

UCSF

UC San Francisco Electronic Theses and Dissertations

Title

Integrating protein similarity networks and orthogonal information for understanding protein origins and function

Permalink

<https://escholarship.org/uc/item/4rv9w55q>

Author

Barber, Alan Edgel

Publication Date

2012

Peer reviewed|Thesis/dissertation

Integrating protein similarity networks and orthogonal information for
understanding protein origins and function

by

Alan Edgel Barber II

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Pharmaceutical Sciences and Pharmacogenomics

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, SAN FRANCISCO

Copyright 2012

by

Alan Edgel Barber II

*For my wife, Melissa M. Barber, without whose support none of this would have
been possible.*

Acknowledgements

There are numerous people whose hard work and involvement have contributed to the substance of this dissertation. First among them was my PI and dissertation chair, Professor Patricia Babbitt, whose guidance provided the intellectual foundation for the work, and whose involvement was instrumental in its execution. Much of the work presented here was a result of different collaborations. The alkaline phosphatase work presented in chapter 3 is a modified version of a paper developed in conjunction with the lab of Professor Daniel Herschlag at Stanford University, specifically with Dr. Jonathan Lasilla, a postdoctoral researcher from the Herschlag lab who will be co-first author on the submitted paper and contributed significantly to the writing and figures; and with Helen Wiersma-Koch, a graduate student from the same lab. Dr. Michael Hicks, a postdoctoral researcher in our lab, also contributed to the figures are writing in this work. All of the caspase work presented in chapter 4 is a small portion of an ongoing collaboration with the lab of James Wells. Specifically, I worked with Dr. Emily Crawford, who was a graduate student in his lab. I would also like to thank all the people who offered valuable time and effort and helped to prepare or bug test Pythoscape for release, specifically Dr. John “Scooter” Morris, Javona White Bear, Dr. Michael Hicks, Dr. Sunil Ohja, Dr. Suwen Zhao, Dr. Patrick Frantom and Jeff Yunes.

More generally, I would like to thank all the members of the Babbitt lab for all their help and advice in developing these projects over the last several years. I also would like to thank everyone who maintains and directs the PSPG program at UCSF, particularly Professor Frank Szoka and now Professor Deanna Kroetz. My thanks also to Debbie Acoba-Idlebi who administrates the program. They helped me arrive at a point where I am ready to take the next steps in my career.

Finally, I would like to acknowledge the funding that made this work possible. I was supported by an informatics fellowship from the Pharmaceutical Research and Manufacturers of America, a fellowship from the Achievement Rewards for College Scientists Foundation and NIH Training Grant T32 GM007175. Dr. Patricia Babbitt, the supervisor of this work, supported me with NIH Grant R01 GM60595.

Note to the reader

A shortened version of Chapter 2 is in press in the journal *Bioinformatics* with the following reference: Barber AE 2nd, Babbitt PC. Pythoscape: A framework for generation of large protein similarity networks. *Bioinformatics*. (*In Press*). The coauthor listed in this publication directed and supervised the research.

A modified version of Chapter 3 derives from the following manuscript *in preparation*: “Barber AE 2nd*, Lasilla JK*, Weirisma-Koch HI, Hicks MA, Herschlag

D, Babbitt PC. Evolution of functional diversity in the alkaline phosphatase superfamily. (*in preparation*).” Alan Barber and Dr. Jonathan Lasilla were co-first authors. Alan Barber carried out the computational experiments. Alan Barber and Jonathan Lasilla both contributed significantly to the writing. Helen Wiersma-Koch and Dr. Michael Hicks also made contributions to the writing. In the material presented in this dissertation, Alan Barber contributed the majority of the writing, but significant sections were written by Dr. Jonathan Lasilla. Alan Barber created all the figures and tables, except for figure 1 and appendix D, table D.1, which were created by Dr. Jonathan Lasilla and appendix D, table D.3 which was created by Helen Weirisma-Koch. Dr. Jonathan Lasilla also created all protein topology diagrams which are used throughout the figures in chapter 3. Dr. Daniel Herschlag and Dr. Patricia Babbitt directed and supervised the research.

Chapter 4 is novel unpublished research from the author with help from Dr. Emily Crawford to create the portion of appendix E, table E.1 which refers to literature references of kinase targets of caspases.

Patricia C. Babbitt

Professor of Bioengineering and Therapeutic Sciences

Abstract

Integrating protein similarity networks and orthogonal information for
understanding protein origins and function

Alan Edgel Barber II

Biology's entrance into the genomic age has meant dramatic changes. Biologists once carried out painstaking, low-throughput experiments, but now often rely on massive high-throughput experimental centers and 'big data'. In modern biology, the quantity of data scientists can create vastly outstrips their corresponding ability to analyze and understand its full meaning. This means that one of most pressing current challenges is to create methods that can manage, organize and visualize massive datasets with the goal of assisting biologists in creating and testing hypothesis.

The computational solution presented in this dissertation is that of the protein similarity network (PSNs) and its implementation and usage. These networks are constructed by using an all-by-all pairwise comparison of a protein entity or feature, of which a network can be visualized. These networks assist in showing proteins of interest within their context, whether it is in a sequence, structure or functional context; and in creating hypothesis about how the data of interest relate to the much larger whole.

First, Pythoscape will be presented which is a novel software framework for the creation, modification and output of large PSNs. It will be described along with an overview and description of the architecture of the framework, as well as an example using the glutathione transferase superfamily to show the power of the framework in investigating the sequence and structure relationships of large protein superfamilies.

Second, an application of Pythoscape to the alkaline phosphatase superfamily is presented. PSNs are used to generate evolutionary hypothesis for this large protein superfamily. These networks, in conjunction with phylogenetic trees, are used to propose an evolutionary model that can annotate protein function more accurately and which also demonstrates the complexity of evolution in large mechanistically diverse enzyme superfamilies.

Finally, an application of Pythoscape to the kinase superfamily is presented. We use PSNs to study how members of this superfamily are targeted by caspases, proteases that are activated during apoptosis. This preliminary research demonstrates that sequence similarity and function do not always track and that other orthogonal sources of information may be necessary for accurate annotation.

Table of contents

Preface

Copyright notice	ii
Dedication	iii
Acknowledgements	iv
Note to the reader	v
Abstract	vii
Table of contents	ix
List of tables	xiv
List of figures	xv

Chapter 1. Introduction 1

1.1. Biology in the era of ‘big data’	2
1.2. Investigating protein evolution in large datasets	4
1.3. Scope of the dissertation	6

Chapter 2. Pythoscape: A framework for generation of large protein

similarity networks 8

2.1. Abstract	9
2.2. Introduction	9
2.3. Framework description	11
2.4. Framework implementation	12
2.5. Example Usage	15
2.6. Conclusion	19
2.7. Methods	20

Chapter 3. Evolution of functional diversity in the alkaline phosphatase

superfamily 22

3.1. Author contributions	23
3.2. Abstract.....	23
3.3. Introduction	24
3.4. Results.....	27
3.4.1. Global sequence relationships in the AP superfamily track with known active site diversity	27
3.4.2. The conserved structural core of the AP superfamily	29
3.4.3. A minimal precursor evolutionary model for the AP superfamily	33
3.4.4. An evaluation of the minimal precursor hypothesis using phylogenetic reconstruction of two/three metal ion AP superfamily members.....	35
3.5. Discussion	39
3.5.1. An evolutionary model for the AP superfamily	39
3.5.2. Strategies for functional divergence in the AP superfamily.....	42
3.5.3. Definition of functional families in the NPP-like sub-subgroup.....	43
3.6. Conclusions	46
3.7. Methods	47
3.7.1. Structure alignments.....	47
3.7.2. Sequence similarity network of AP superfamily.....	48
3.7.3. Sequence similarity network of 2/3 metal enzymes.....	49
3.7.4. Sequence alignments	51
3.7.5. Two/three metal group tree.....	51
Chapter 4. Investigating caspase substrate evolution using kinases as a model system.....	53
4.1. Abstract.....	54
4.2. Introduction	54

4.3. Results.....	57
4.3.1. Human STE kinases are statistically enriched for caspase substrates.....	57
4.3.2. Cleavage sites are only rarely conserved between human STE kinases.....	60
4.3.3. Preliminary computational prediction of kinase targets of caspases.....	62
4.4. Discussion	66
4.4.1. Motif conservation is a poor computational tool to predict caspase targets, except in cases of close homologues	67
4.4.2. Computational prediction using motif identification is currently insufficient to identify putative target sets.....	68
4.6. Conclusions	69
4.7. Methods	70
4.7.1. Sequence similarity network of kinases	70
4.7.2. Enrichment analysis.....	70
4.7.3. Kinase motif alignments.....	71
4.7.3. Substrate scoring using motif scores	72
Bibliography.....	73
Appendix A. Supplementary information for chapter 2	75
Appendix B. Pythoscape documentation.....	79
B.1. Introduction.....	80
B.2. General documentation	80
B.2.1. Installation instructions	80
B.2.2. Framework overview	82
B.2.3. Plug-in usage	86
B.2.4. Network creation pipeline	97
B.2.5. Writing plug-ins	108

B.2.6. Setting up a MySQL interface.....	115
B.2.7. Pre-defined regular expressions.....	116
B.2.8. Parallel resources	119
B.2.9. Delete plug-ins	122
B.2.10. Input plug-ins.....	125
B.2.11. Output plug-ins.....	168
B.3. Tutorials.....	188
B.3.1. Scripting with objects tutorial	188
B.3.2. Small network tutorial with local interface	196
B.3.3. Large network tutorial with MySQL interface	211
Appendix C. Pythoscape source code	228
C.1. Introduction	229
C.2. ./ (root) modules	230
C.3. ./mysql_resources code.....	233
C.4. ./parallel_resources modules	286
C.5. ./src packages	309
C.5.1. ./src/pythoscape/auxiliary modules.....	309
C.5.2. ./src/pythoscape/interface modules	340
C.5.3. ./src/pythoscape/main modules	382
C.5.4. ./src/pythoscape/plugin/delete modules.....	433
C.5.5. ./src/pythoscape/plugin/input modules	437
C.5.5. ./src/pythoscape/plugin/input_bio modules.....	501
C.5.6. ./src/pythoscape/plugin/output modules	518
C.5.7. ./src/pythoscape/plugin/output_bio modules.....	546
C.5.8. ./src/pythoscape/plugin/output_r modules	546

Appendix D. Supplemental information for chapter 3.....	552
Appendix E. Supplemental information for chapter 4.....	582
Publishing agreement.....	586

List of tables

Table 2.1. Pythoscape plug-ins provided in version 1.01b.	13
Table 4.1. Fisher's exact test to determine positive enrichment of caspase targets in Manning kinase groups.	60
Table 4.2. Table of top ten predictions from the composite scoring method, their cellular location and if they are a known caspase target	65
Table A.1. Benchmarks for edge calculations.....	78
Table D.1. Enzymatic activities of AP superfamily members.....	553
Table D.2. Experimentally characterized two/three metal subgroup enzymes.....	558
Table D.3. Motif comparison analysis of NPP-like homologs	562
Table D.5. Data for identifying leaves for 2/3 metal subgroup tree.....	574
Table E.1. Kinases that are cleaved by caspases.	583

List of figures

Figure 1.1. Growth of sequences in UniProt.....	3
Figure 2.1. Sequence similarity network of the GST superfamily generated by Pythoscape and visualized in Cytoscape.	16
Figure 2.2. Structure similarity network of GST structures colored by SwissProt GST family.	18
Figure 3.1. Catalysis in the AP superfamily.	26
Figure 3.2. Sequence similarity network of the AP superfamily with experimentally determined protein structures indicated.	28
Figure 3.3. Conserved structure and active site features of members of the AP superfamily.....	31
Figure 3.4. Sequence alignment of selected AP superfamily members for which three-dimensional structures have been determined.....	33
Figure 3.5. Sequence similarity network of the AP superfamily main cluster colored by eccentricity highlights central, minimal enzymes.	34
Figure 3.6. Bayesian phylogenetic tree of the proteins in the 2/3 metal subgroup of the AP superfamily.	36
Figure 3.7. Minimal precursor model for evolution of functional diversity in the AP superfamily.....	40
Figure 3.7. Model for evolution of functional diversity in the AP superfamily.	40
Figure 3.8. Predicting substrate specificity in NPP-like homologues.	44

Figure 4.1. Sequence similarity network of the Manning kinome colored by caspase cleavages.....	58
Figure 4.2. Little conservation of known caspase cleavage sites in STE kinases.	61
Figure 4.3. ROC plot of methods to score entire protein sequences as targets of caspases.....	63
Figure A.1. Correlation plot between TM-score and $\log_{10}(-\log_{10}(\text{E-value}))$	76
Figure A.2. Correlation between ideal representative $-\log_{10}(\text{Evalue})$ node distances and ideal full network $-\log_{10}(\text{Evalue})$ node distances for GST sequences.....	77
Figure D.1. Sequence similarity network of the 2/3 metal subgroup of the AP superfamily colored by experimentally determined functions.	554
Figure D.2 Sequence similarity network of the AP superfamily with superkingdom indicated.....	556
Figure D.3. Structure similarity network of the 2 and 3 metal AP superfamily enzymes colored by experimentally determined function.....	557
Figure D.4. Sequence similarity network of the 2/3 metal subgroup of the AP superfamily with NPP-like homolog set indicated.....	560

Chapter 1. Introduction

1.1. Biology in the era of 'big data'

The quest to sequence the first human genome, completed in 2001(1, 2), marked a revolutionary transformation in the way that science would be carried out. While it certainly wasn't the first published experiment to generate a large quantity of data yet to be interpreted for biological meaning, it was a watershed moment that made clear that science had fully entered the age of 'big data'. The data scientists can create now vastly outstrips their corresponding ability to analyze and understand the full meaning of the data created. The number of predicted and known sequences of proteins in UniProt (3) has now ballooned exponentially to over 22 million as of May 14, 2012 (figure 1.1), with only 536,029 in the well curated portion of UniProt, the SwissProt database. This problem of large sets of unknown sequences will only continue to become more intractable as metagenomic sequencing efforts increase. As of August 24, 2012, the IMG/M (4) contained over 7 billion predicted gene sequences, with the vast majority coming from bacterial metagenomic sequencing projects. Not included in these numbers are the other large databases of biological information including expression levels, protein-protein interactions and various proteomic datasets, such as degradomics.

With all the effort now employed with the sole purpose of generating data, it is reasonable to ask if all of this data will ultimately prove useful, especially since much of it remains un-analyzed. The human genome is widely credited with providing the foundation for a new understanding of human health (5). With the 1000 genomes project (6), each individual's genome can be compared with a set

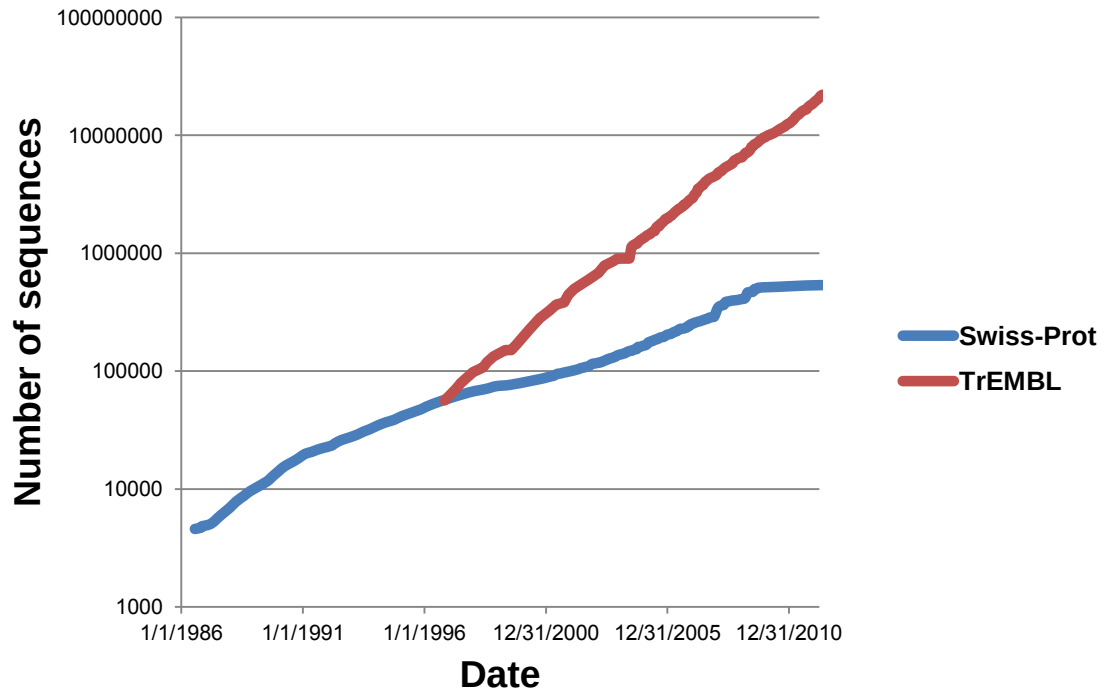


Figure 1.1. Growth of sequences in UniProt. This plot demonstrates the exponential growth of the uncurated TrEMBL sequence database as opposed to the slower growth of the curated Swiss-Prot sequence database. The data to make this plot was kindly provided by Elisabeth Gasteiger at the Swiss Institute of Bioinformatics.

of references to determine how their specific variation can give rise to phenotype, a central requirement for creating a more personalized experience in medicine. Large metagenomic projects are providing a window into how the chemistry of life is carried out in environments as divergent as the human gut (7) or heavy metal contaminated groundwater (8). Ultimately, the goal is to use this knowledge to learn how to locate and manufacture the next generation of chemical products and pharmaceutical therapies (9).

But, these goals are dependent on accurate, large-scale inference of how gene sequence gives rise to function. A recent study has shown that in some superfamilies the majority of the functional annotations of predicted genes in the major database are inaccurate (10). This calls into question the usefulness of all the data being collected if accurate annotations are not computationally tractable. The prevalence of questionable annotations deposited in large databases means that one of the most pressing, current challenges is to create methods that can manage, organize and visualize massive datasets with the end goal of assisting scientists to create and test hypothesis. Only with the ability to rapidly understand data on a large scale will modern science be able to overcome the gulf between the quantity of data that exists and its meaning.

1.2. Investigating protein evolution in large datasets

Much of the misannotation in publically-available databases is due to misunderstanding at what level of significance of sequence similarity function is conserved between two genes (10). Because every protein family/superfamily evolves at a unique rate and has unique sequence determinates of function when compared to every other family/superfamily, this answer must be determined contextually for each annotation transfer (11). The complexity of determining the factors on a case-by-case basis constitutes a key obstacle when attempting to improve the accuracy of function prediction. SwissProt, for example, has specific rules that are developed by a curator for annotating each family/superfamily (12). This procedure has resulted in a much lower rate of misannotation that other less

curated databases (10). The Babbitt lab employs a more specific processes developed for enzymes, which is to assign specific determinates of chemistry at various levels of a hierarchy, and then specifically assign target proteins to a given level of the hierarchy (13). This hierarchy developed is hand-curated and based on the specific evolutionary constraints of delivering enzymatic function for each protein superfamily. This prevents the common problem of over-annotating where the sequence and structure requirements for a specific function are not present, but the sequence is similar to a known function (10).

To match sequence and structure features with evolutionary models, traditionally phylogenetic trees are used, such as using an 'evolutionary trace' methodology (14). And while there are resources to investigate evolution using large scale automated trees (15, 16), two significant technical hurdles prevent easy creation, curation and interpretation of these trees. First, one of the most informative features of trees is the use of methods capable of determining the statistical likelihood of the tree topology outputted, such as Bayesian (17) or maximum likelihood (18) methods. These methods do not scale well to large sets of proteins because of the difficulty in sampling the large topology space of unique trees, which can lead to convergence issues. The second hurdle is that trees require the presence of a sequence alignment and tree accuracy directly depends on alignment accuracy. The computational complexity in determining multiple sequence alignments for large sets of high divergently sequences means that generating these alignments is only possible with manual intervention and curation (19).

To solve these issues, instead of using trees as an initial investigatory method, the Babbitt lab now uses protein similarity networks (PSNs) as the preferred method for large scale protein sequence, structure and function investigation. These networks are constructed by using an all-by-all pairwise comparison of a protein entity or feature, which network can be visualized using a program such as Cytoscape (20). In the visualized network, each protein (or a group of proteins in the case of a representative network) is represented by node and each edge represents a score of similarity. In most cases, the networks can be thresholded to show and calculate layouts solely based on certain sets of edges that pass a given level of statistical significance. While no evolutionary model is present in a PSN, they do scale well to large sets, do not require the use of alignments and have been useful in a number of superfamilies to generate initial hypothesis. These can then be investigated more thoroughly with much more rigorous methods, like phylogenetic trees (21).

1.3. Scope of the dissertation

The focus of this dissertation will be on the creation of PSNs and their use to investigate the evolution of large protein superfamilies.

Chapter 2 is a description of Pythoscape, a novel software framework for the creation, modification and output of large PSNs. The chapter will provide an overview and description of the architecture of the framework, as well as an example using the GSTs to show the power of the framework in investigating the sequence and structure relationships of large protein superfamilies. This chapter

is an expansion of a paper now in press, Barber AE 2nd, Babbitt PC.

Pythoscape: A framework for generation of large protein similarity networks.

Bioinformatics. (In Press).

Chapter 3 is an application of Pythoscape to the alkaline phosphatase superfamily, a mechanistically diverse enzyme superfamily whose members catalyze mostly phosphoester or sulfoester hydrolysis. In this chapter, we demonstrate the use of PSNs to generate evolutionary hypothesis for large protein superfamilies, which we can further test using phylogenetic trees. We propose that this evolutionary model can be used to annotate more accurately protein function. This model also shows the complexity of evolution in large mechanistically diverse enzyme superfamilies.

Chapter 4 is an application of Pythoscape to the kinase superfamily. This superfamily is important in many cell signaling pathways and does not vary by mechanism; rather, the members vary by substrate targeting and. We use PSNs to study how members of this superfamily are targeted by caspases, proteases that are activated during apoptosis. This preliminary research demonstrates that sequence similarity and function do not always track and that other orthogonal sources of information may be necessary for accurate annotation. In this case, it is because the function of interest, caspase cleavage, is determined by small linear motifs that evolve very rapidly. In the case of the kinases studied, homologous kinases that are caspase targets often do not share homologous cleavage sites. We propose that pathway membership networks might be a more useful methodology in annotating caspase targets, instead of PSNs.

Chapter 2. Pythoscape: A framework for generation of large protein similarity networks

2.1. Abstract

Pythoscape is a framework implemented in Python to process large protein similarity networks for visualization in other software packages. Protein similarity networks are graphical representations of sequence, structural, and other similarities among proteins for which pairwise all-by-all similarity connections have been calculated. Nodes are used to represent one or more protein sequences or structures and edges drawn between two nodes represent the aforementioned measures of their similarity. Mapping of biological and other information to network nodes or edges enables hypothesis creation about sequence-structure-function relationships across sets of related proteins. Pythoscape provides several options to calculate pairwise similarities for input sequences or structures, apply filters to network edges, define sets of similar nodes and their associated data as single nodes (termed representative nodes) for compression of network information, and output data or formatted files for visualization. The Pythoscape framework is available for download at: <http://www.rbvi.ucsf.edu/trac/Pythoscape>

2.2. Introduction

The rapid growth of databases of protein information (e.g., sequences and structures) provides both new opportunities and challenges for analysis and clustering by similarity. For example, global analysis of entire superfamilies and association of their members with biological information and other types of metadata has become a useful tool for functional annotation and discovery (21).

At present, some of these sets contain many thousands of divergent sequences and they continue to grow larger with new sequencing projects. This size results in fast exploration on the large scale becoming less feasible using traditional approaches such as alignments and trees.

PSNs enable analysis and visualization of structure-function relationships in large protein data sets by clustering of individual protein sets for more complex analysis while summarizing 'connectivity' relationships among the clusters. Mapping orthogonal sources of biological information onto PSNs then provides a powerful way to view functional trends across the set that can be interpreted in the context of their similarities. Previous work published in the Babbitt lab has described various uses of these networks to describe functional properties of proteins in the context of their sequence and structure relationships as well as a statistical validation of the method (22).

Already existing are databases like The Similarity Matrix of Proteins (23) which store pairwise similarities that a user can download to construct PSNs. There are also plug-ins available with software such as Cytoscape (20) that allow creation of small PSNs (24). What has not existed until now is a software platform capable of creating and managing large PSNs. As already noted, sequence databases only continue to grow in size, which means that solutions for large datasets will only become more necessary in the future. Pythoscape is a new extensible software framework implemented in Python, designed to create and manipulate large PSNs with the explicit goal of being a tool to make biological hypotheses in the age of 'big data'.

2.3. Framework description

There are two major difficulties in designing software to work with PSNs. The first is the heterogeneity of possible useful metadata types and data transformations. Many unique annotations or calculations based on those annotations - some of which will only be discovered to be useful in the future - could be necessary to associate with PSNs for hypothesis generation. The second difficulty is that unlike sparser networks such as interaction networks, PSNs are frequently close to complete, often requiring storage, management and rapid calculation of large quantities of data, which requires that appropriate storage solutions can be chosen based on network size or developed as needed. This points to the development of an extensible framework instead of a static software package as the optimal solution, since the software will need to be developing constantly to remain useful.

The philosophy guiding successful framework design required the designer to identify 'design hotspots' where extensibility will be allowed for the framework to maximize usefulness. The designer must also identify design constraints where the framework will specify required usage parameters. These constraints simplify the implementation of code in the 'hotspots', focusing future developers' and users' efforts only on implementing truly novel code, which prevents unnecessary reimplementations.

The 'hotspots' defined in the Pythoscape framework based on identified engineering issues are (1) data addition or transformation, which can be

achieved using plug-ins; and (2) data storage, which can be achieved using a storage interface. The constrained code is found within a set of environment modules, which provide a developer or user all the classes necessary to instantiate graph objects, interact with them using plug-ins and pass them into storage interfaces for data storage, retrieval or deletion.

2.4. Framework implementation

For the user interested in generating large networks without doing any development, the Pythoscape framework comes pre-packaged with a core set of plug-ins (table 2.1) and two interfaces: (1) a local interface that requires no additional configuration and (2) a MySQL-based interface which can be used for large networks and requires additional setup. No development is needed to create simple network associated with useful metadata. For software developers, Pythoscape provides the previously described environment modules for rapid modification along with well-documented application programming interfaces (APIs) for development of additional plug-ins using new sources of metadata or interfaces for new, perhaps faster or more reliable, storage options.

An example of a useful feature with no user modification needed is the ability of Pythoscape could create a large network; then, using representative nodes and edges, it can compress the information and output smaller summary networks for visualization (figure 2.1A & B). Pythoscape allows the user to choose the mean, max or minimum edge as the representative edge score between two groups of represented nodes. All represented information is stored

Table 2.1. Pythoscape plug-ins provided in version 1.01b.

Plug-in name	Type	Description
AddAttributesByGreatestSum	Input	Create representative node attributes by calculating the node attribute with the greatest count
AddAttributesByIfAny	Input	Create representative node attributes by concatenating all attributes in nodes represented
AddBLASTEEdgesFromLocalBLAST	Input	Calculate BLAST edges
AddBLASTEEdgesFromTableRun	Input	Add BLAST edges from a table file
AddEdgesToRepnodNetwork	Input	Calculate representative edges
CalcNodeSize	Input	Calculate how many nodes are in a representative node
CalculateTMAAlignScores	Input	Calculate structure similarity using TM-align
CreateCDHITRepnodes	Input	Use CD-HIT to cluster nodes for representative networks
DeleteEdgesFromFile	Delete	Delete edges as specified by a file

ImportAttributeTable	Input	Input attributes from a file
ImportFromFastaFile	Input	Input sequences from fasta file
ImportFromUniProt	Input	Import node attributes from UniProt
ImportIdentifierTable	Input	Input node identifiers from table file
OutputAttributeClusters	Output	Output attributes from representative nodes
OutputAttributeTable	Output	Output table file of node attributes
OutputIdentifierTable	Output	Output table of node identifiers
OutputCorrelation	Output	Output correlation plot between two edge attributes
OutputXGMML	Output	Output XGMML file for visualization
PDBImportFromCSVFile	Input	Import structures from PDB database into Pythoscape
SequenceTableRun	Output	Output set-up files for parallel sequence edge run
StructureTableRun	Output	Output set-up files for parallel structure edge run

in the database so that the user can output selected sub-networks using the entire set of sequences in each.

Additionally, Pythoscape has plug-ins for creating structure similarity networks and for generating correlations for edge distances between networks generated from a set of sequences and a corresponding set of available structures (table 2.1, figure 2.2 and appendix A, figure A.1).

Documentation and tutorials are provided on the website to instruct new users on the proper use of Pythoscape. A copy of the documentation and tutorials is provided in this dissertation in appendix B. The source code is also available on the website for download and a full copy of the source code for Pythoscape v.1.01b is provided in appendix C. Within the downloadable source code, Pythoscape contains various tools to parallelize the process of creating networks. A benchmark on one of these tools, `parallel_blast2seq.py`, is found in appendix A, table A.1.

2.5. Example Usage

Glutathione transferases (GSTs) are enzymes that typically catalyze the addition of glutathione to substrate compounds. They play roles in many biological processes, including metabolism of endogenous compounds and xenobiotics such as drugs. Of the thousands of GSTs that have been identified, the physiological substrates of only a small proportion are known; thus, they are

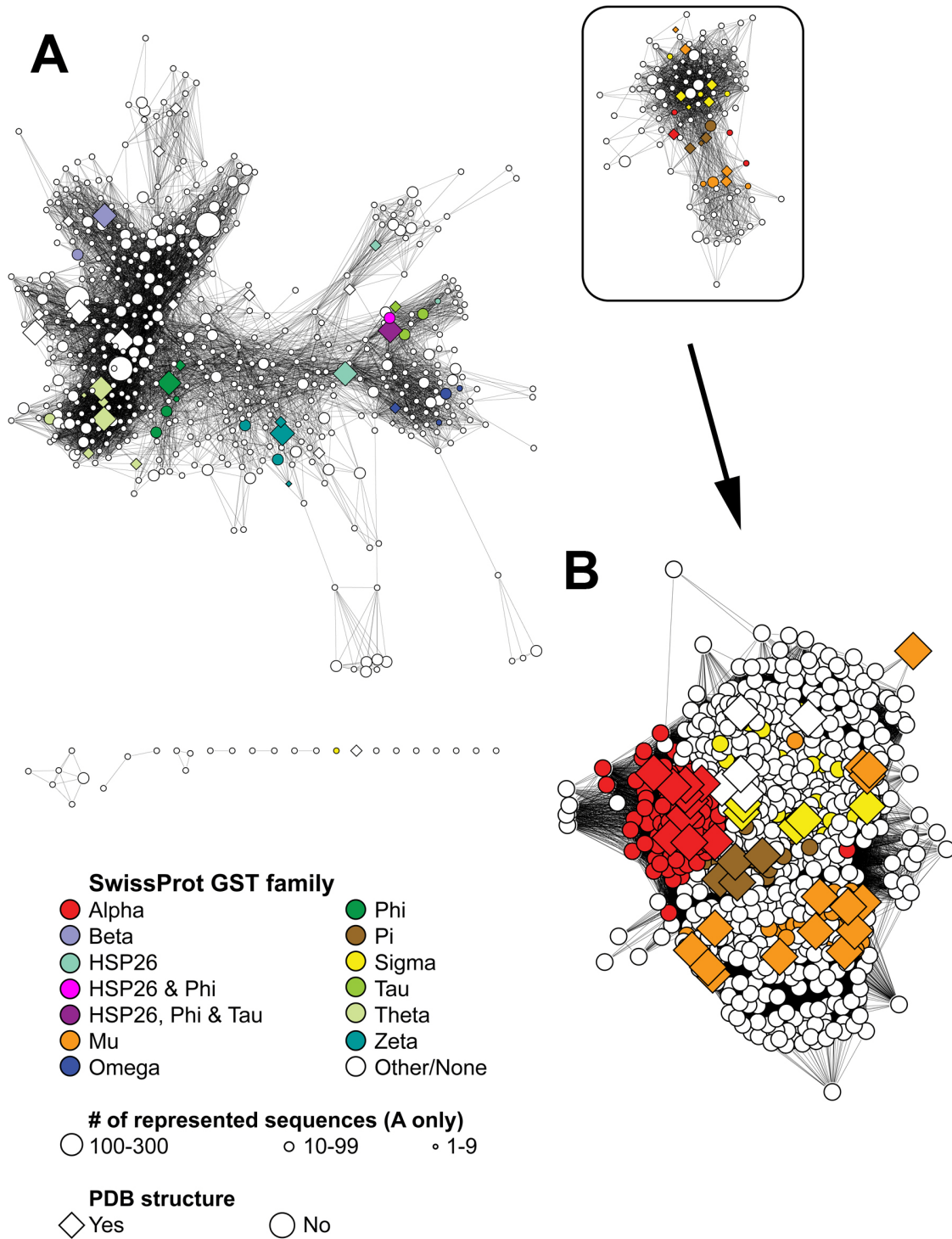


Figure 2.1. Sequence similarity network of the GST superfamily generated by Pythoscape and visualized in Cytoscape. To compact the view for this figure, networks were laid out using the organic layout in Cytoscape rather than the distances computed from a similarity metric. 664 representative nodes are

used to describe pairwise relationships among 7,447 sequences. **A)** Representative network with functional classes colored, if annotated by SwissProt in a family (3). Family membership is indicated if one or more sequences in the abstracted node are associated with that family. Similarity between representative nodes is calculated as the mean $-\log_{10}(\text{E-value})$ of all pairwise BLAST+ (25) alignments < 1 between all sequences abstracted by each representative node. Edges are shown if the mean $-\log_{10}(\text{E-value})$ for pairwise comparisons of the sequences within a representative node is greater than a threshold of 13; edges at this threshold represent alignments with a median 31% identity over 205 residues. **B)** Full, non-abstracted network for the group of GSTs found mostly in eukaryotes (boxed in A). This network contains 702 sequences and only edges with pairwise BLAST $-\log_{10}(\text{E-value})$ alignment score of greater than 13.

principally classified into putative functional classes according to enzymatic, structural, and other features (26). Recently, PSNs have been used to summarize and guide a global interpretation of GST sequence and structure relationships (27).

A PSN of GST sequences is shown in figure 2.1A. It illustrates how representative nodes computed by Pythoscape enable analysis of PSNs too large to be visualized in total while retaining their value for developing hypotheses from sequence similarities across the whole set. For comparison, individual clusters of interest can be outputted with all nodes present (figure 2.1B). This full, non-abstracted network (representing a node for each sequence) shows a similar pattern of relationships to those shown in the corresponding representative node network (boxed in figure 1A). The correlation between the ideal representative node mean distances calculated in Pythoscape and the corresponding full network ideal distance for figure 2.1A is provided in appendix

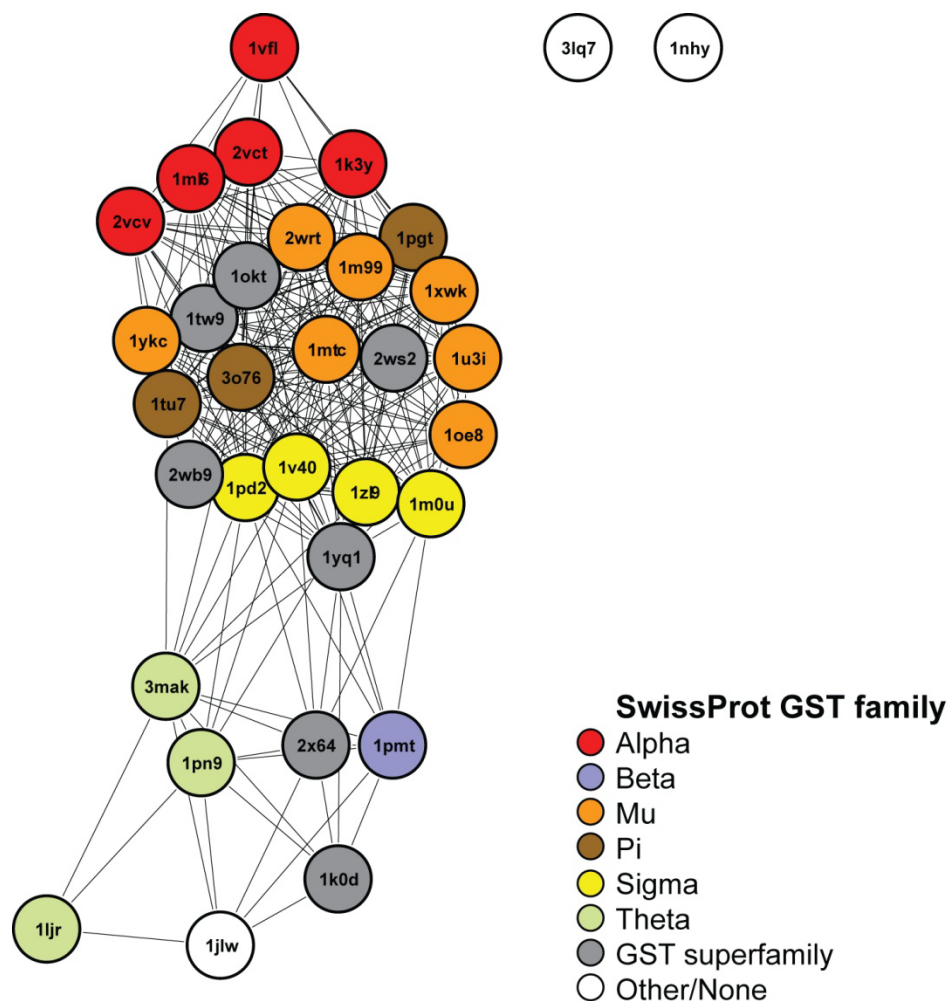


Figure 2.2. Structure similarity network of GST structures colored by SwissProt GST family. Edges between nodes are drawn only if the TM-score > 0.80 (median number of aligned residues = 187; median RMSD = 2.32Å) for that edge. Structures are classified to SwissProt families as in figure 2.1. Each node is labeled by the PDB structure it represents. The network is visualized using the organic layout in Cytoscape. The correlation between edge distance for this structure similarity network and the corresponding $-\log_{10}(\text{E-value})$ edge in the sequence similarity network in figure 2.1A is plotted in appendix A, figure A.1. While the distances are similar, there is much less coverage in the structure network.

A, figure A.2. A quantitative description of the relationships between filtered networks and full networks has also recently been described elsewhere for some example systems (22), but these differences appear also to depend on the specific system analyzed. While “missing data” is an inherent feature of representative nodes, the trade-off is in visualizing similarity relationships across very large datasets that would not be practically achievable due to memory and speed limitations in their calculation.

The network shown in figure 2.1A demonstrates another issue in the use of representative nodes that could complicate interpreting relationships between functional features and sequence similarity. In the example given here, some GST families are represented by multiple representative nodes, while other representative nodes contain multiple SwissProt families (HSP26, Phi and Tau), obscuring how sequence similarity tracks with annotation. Thus, a general recommendation is that analysis using representative networks be accompanied by examination of the relevant parts of the corresponding full networks.

2.6. Conclusion

Pythoscape is a software framework to efficiently create and manage protein similarity networks. It has been designed to facilitate sequence-structure-function hypothesis creation by allowing functional attributes and other metadata to be visualized in the context of large PSNs. An example of its use using the GST superfamily has been presented and demonstrates that the relationship between functional annotation and sequence and structure similarity can be quite

complex. Tutorials, documentation, source code, and future development plans are available at: <http://www.rbvi.ucsf.edu/trac/Pythoscape>. Tutorials and documentation are also available in appendix B and the source code is available in appendix C.

2.7. Methods

Pythoscape v1.01b was used to create and annotate the networks (Ref!). The network depicted in figure 2.1 was generated using all full-length protein sequences from Pfam 24.0 GST, GST_N (PF02798) (28). Figure 2.1A contains 664 representative nodes for 7,447 sequences from the Pfam GST family and 10,881 representative edges representing 2,125,783 edges. One representative node is depicted for each cluster of sequences defined by CD-HIT v4.5.6 (29) using a 40% sequence identity threshold. An indication of the number of sequences represented in each representative node is given in the legend for figure 2.1. The network depicted in figure 2.1B contains the full network from the boxed cluster in figure 2.1A. It contains the full set of 702 sequences represented. For the network depicted in figure 2.3, pairwise similarities for a non-redundant set of 33 structures from the Pfam 26.0 GST_N family (PF02798) (28) were calculated using TM-align (30).

All networks are visualized in Cytoscape v2.8.3 using the organic layout. This layout uses only node connectivity to calculate layout position and not ideal distance. This results in a more compact network for viewing. As reported elsewhere, layout positions for PSNs using the organic layout are similar to those

calculated using a force directed layout computed from ideal distances (22) (supplementary information table 1 with Pearson's correlation (R) ranging from 0.838 to 0.924 for eight different systems). Nodes are colored according to SwissProt GST family classifications. Annotations for SwissProt GST family and PDB structures were taken from UniProt release 2012_07 (3). White nodes are those from the Pfam set that are not annotated in the SwissProt database, nor are these annotated with any of the main classes shown in the figure legend.

Chapter 3. Evolution of functional diversity in the alkaline phosphatase superfamily

3.1. Author contributions

This chapter contains a modified set of excerpts from the paper: “Barber AE 2nd*, Lasilla JK*, Weirsmma-Koch HI, Hicks MA, Herschlag D, Babbitt PC. Evolution of functional diversity in the alkaline phosphatase superfamily. (*in preparation*) *These authors contributed equally to this work.” For a full explanation of the input of each of the authors, see “Note to the reader” in the dissertation preface.

3.2. Abstract

Mechanistically diverse enzyme superfamilies are composed of evolutionarily-related enzymes that share common mechanistic features yet catalyze different chemical reactions. Studying of the evolution of these types of superfamilies provides a unique opportunity to understand how nature has modified specific catalytic scaffolds to catalyze numerous enzymatic reactions. The alkaline phosphatase (AP) superfamily provides an especially compelling system because its founding member, AP, is a prototypic phosphoryl transfer catalyst with a well-characterized mechanism. Homologues of AP catalyze a range of phosphoryl and sulfuryl transfer reactions including phosphatases, sulfatases, phosphodiesterases and phosphomutases.

We present an investigation of the AP superfamily using PSNs, which demonstrates that significant active site features, such as the presence of auxiliary domains, residues involved in catalysis and metal ion count cluster over sequence and structure space. We combine this analysis with a phylogenetic

tree, which suggests an evolutionary model for the divergence of many functions from a common ancient minimal ancestor. We show how a robust understanding of the evolutionary divergence of function can assist in identifying the determinants of mechanism for enzymatic families. Finally, we demonstrate how this model can be used to propose putative functional boundaries in sequence space.

3.3. Introduction

A mechanistically diverse enzyme superfamily is a group of homologous enzymes that retain a conserved active site feature with a conserved mechanistic capability, but have divergent overall reaction specificities (31). These types of superfamilies provide ideal systems to understand how many diverse possible protein functions can evolve from a single starting point, along with the evolutionary strategies employed. This knowledge can be used not only for protein function prediction, a vexing present problem given the exploding number of un-annotated or mis-annotated sequences in public databases, but to rationally engineer proteins (21, 32).

The alkaline phosphatase (AP) superfamily provides an especially compelling system in which to take advantage of evolutionary relationships to understand specialization of enzymatic function. Its founding member, *Escherichia coli* AP, represents a prototypic phosphoryl transfer catalyst and is among the most well characterized enzymes (33). The evolutionary and structural relationships between AP and a set of phosphatases, sulfatases,

phosphodiesterases, and phosphomutases were first identified by Galperin and coworkers through the observation of conserved metal ion binding residues (34). These enzymes catalyze a range of phosphoryl and sulfuryl transfer reactions that include different transition states, varied substrate negative charge and variations in charge distributions, differences in the number and types of esterifying groups, and cleavage of P-O, S-O, P-H, and P-C bonds (figure 3.1, appendix D, table D.1). Yet all of these reactions are united by the use of at least one metal ion and an alkoxide nucleophile to attack at sulfur or phosphorus, forming a covalent enzyme-phosphate or enzyme-sulfate species (figure 3.1).

This common chemistry has been diversified through various strategies. For example, mutations to modify in the number of bound metal ions from 3 to 2 in AP dramatically change the substrate specificity profile (35). Auxiliary domains in the phosphomutases directly participate in the catalytic mechanism by interacting and reorienting the substrate during catalysis (36). In other members of the superfamily, like nucleotide pyrophosphatase/phosphodiesterase (NPP), residues and loops that are part of the core fold near the active site directly participate in binding and positioning the substrate (37). The diversity of identified strategies complicates organization and prediction of function within the superfamily, but presents an opportunity to use large scale methods and analysis with newly developed scalable computational tools, such as PSNs, along with the vast data from ongoing sequencing projects to develop models of functional evolution.

In this chapter, we present a model for the evolution of enzymatic specialization in the AP superfamily, which merges hypotheses from the clustering of functional features with respect to sequence and structure similarity gleaned from PSNs with phylogenetic relationships inferred from phylogenetic trees. From this evolutionary model, we can organize the superfamily into functional groupings based on shared functional characteristics and shared common ancestry. We demonstrate how all modern day functions could have arisen from modifications to a minimal precursor and how identification of those modifications can be used to map functional boundaries.

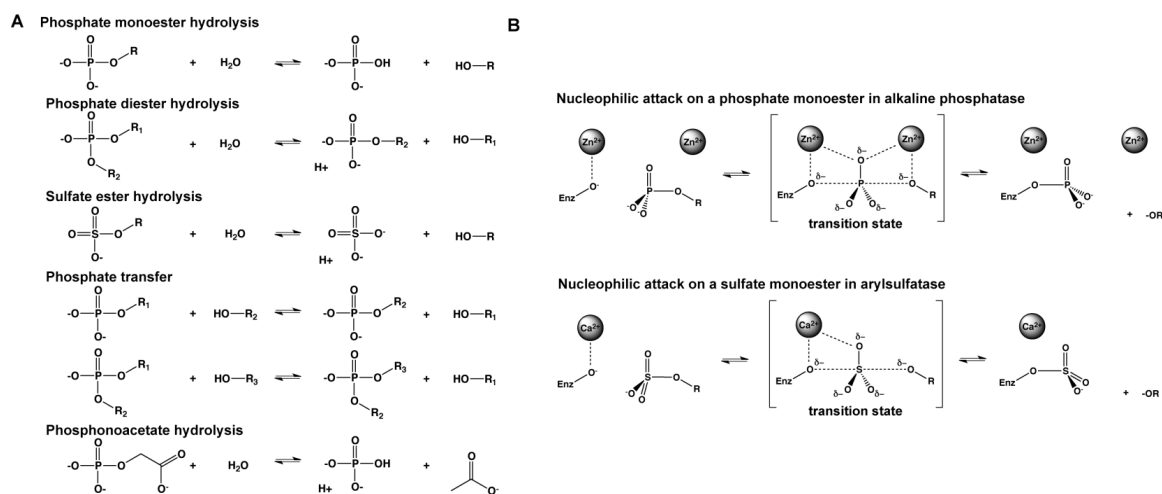


Figure 3.1. Catalysis in the AP superfamily. **A)** Known reaction types catalyzed by members of the AP superfamily. **B)** AP superfamily members share the common chemical step of attack by an alkoxide nucleophile at phosphorus or sulfur to form a covalent intermediate.

3.4. Results

We first define the AP superfamily and present a PSN of all known members and discuss how known structural features cluster with respect to sequence similarity. We then present an alignment of a structural superposition of all known structurally characterized AP superfamily members and discuss the common topology. Given robust structural coverage as well as relatively wide scale functional characterization of the 2/3 metal enzymes, we are able to align the core domain across this subgroup. We use this alignment to generate a phylogenetic tree and model of functional divergence for the superfamily.

3.4.1. Global sequence relationships in the AP superfamily track with known active site diversity

Using the Pfam (28) clan Alk_phosphatase (CL0088) as a basis for identifying sequences that are part of the AP superfamily, we calculated a representative sequence PSN of a set of 24,403 putative superfamily members. This PSN contains three connected clusters with structurally characterized members (figure 3.2). This first cluster contains enzymes that are known to bind a single metal ion and which are annotated as acid phosphatases. The characterized member of this cluster, PDB structure: 2D1g, is involved in helping a pathogen to elude host defenses by hydrolyzing phosphorylated substrates (38). The second, largest cluster contains many known functions of both one and two metal ion binding enzymes. These enzymes cluster separately onto two

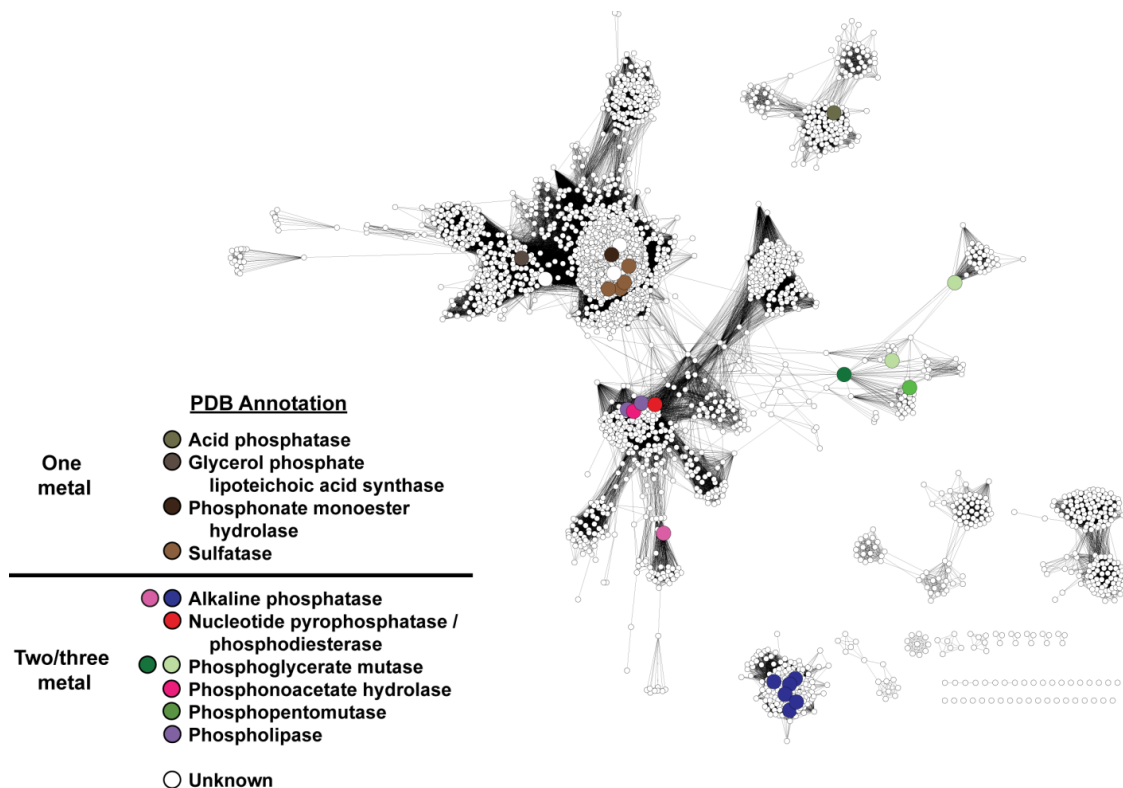


Figure 3.2. Sequence similarity network of the AP superfamily with experimentally determined protein structures indicated. A representative set of 4590 nodes was selected from the 24403 protein sequence superfamily, as determined by Pfam models. Each node represents a set of proteins with >40% identity. Edges between nodes are drawn only if the maximum similarity of edge weights greater than $-\log_{10}(\text{E-value}) > 13.1$ (median alignment length = 422 residues; median percent identity of pairwise comparisons = 30%). The network is visualized using the organic layout in Cytoscape. Coloring is based as indicated in the color annotation key if any protein in a representative node has been structurally characterized and has the given annotation.

different lobes of this main cluster indicating a fundamental sequence divergence between the two sets of enzymes. Finally, the last cluster with structurally characterized members is the cluster which contains the canonical alkaline phosphatase sequences and contains enzymes which bind three metal ions. This

separation demonstrates a fundamental sequence divergence between these enzymes and the other one and two metal-ion binding enzymes.

While the clustering patterns clearly track with active site characteristics, the relationship between annotated enzymatic function and PSN topology is more complex. As an example, two clusters which have known members annotated as “alkaline phosphatases” are seen. The canonical set, as previously explained, clusters apart from the main cluster. A non-canonical structurally characterized “alkaline phosphatase” is also found in the main cluster. This enzyme has been shown to hydrolyze phosphates in alkaline conditions, but only binds two metal ions and must stabilize the transition state in a method differing from the canonical enzymes that use a water bound by the third metal to interact with the substrate during catalysis (37, 39). As a further example, enzymes known to catalyze sulfatase hydrolysis are only found in the one metal lobe of the main cluster, but enzymes that catalyze phosphate ester hydrolysis are found ubiquitously throughout all three clusters with characterized enzymes (figure 3.2, appendix D, table D.1).

3.4.2. The conserved structural core of the AP superfamily

Members of the AP superfamily share a structural core that anchors crucial side chains for catalytic function. This core motif consists of a central beta sheet, typically of eight strands, and typically six conserved helices (figure 3.3). In all known members, two side chain ligands for catalytic metal ions are situated

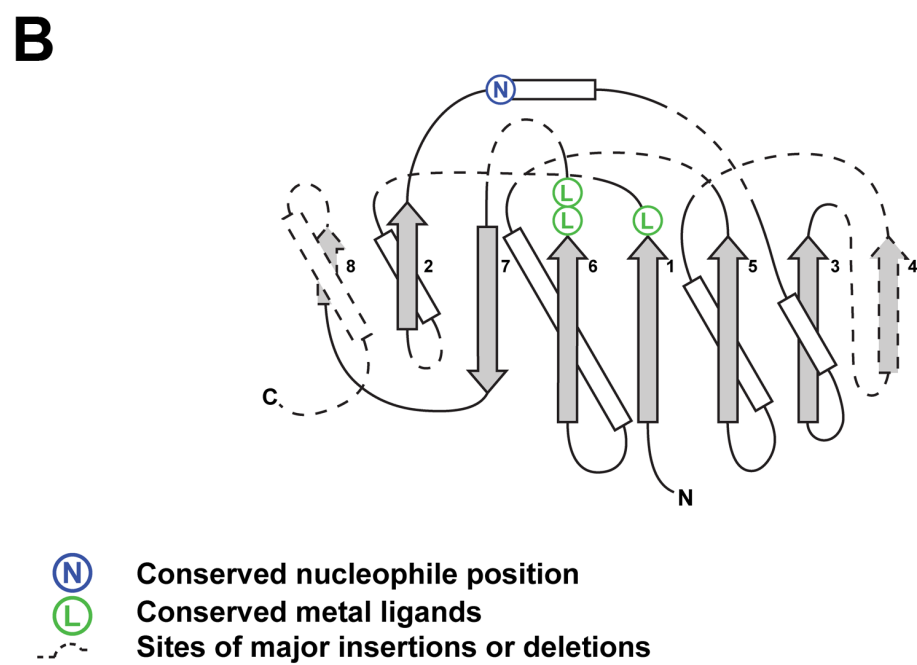
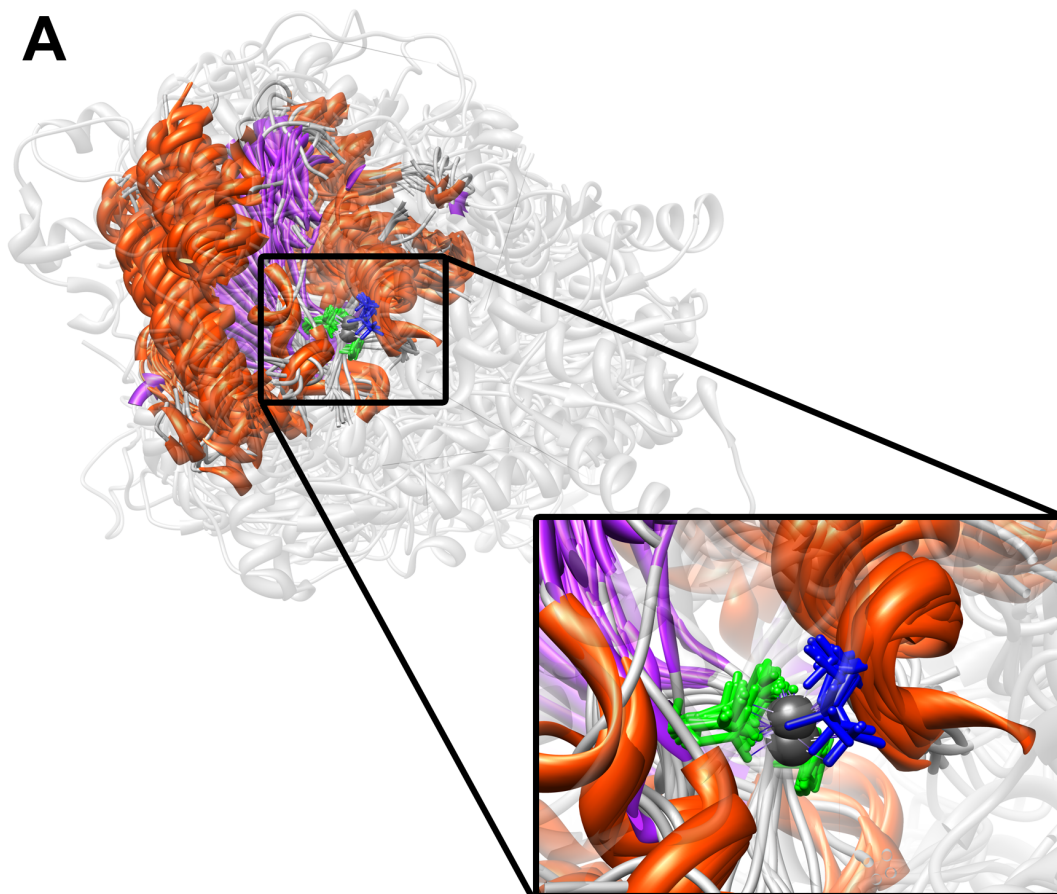


Figure 3.3. Conserved structure and active site features of members of the AP superfamily. **A)** Superposition of a non-redundant set of protein structures representing the AP superfamily. A full list of the aligned structures and an alignment of regions surrounding the active site is provided in figure 3.4. Structurally conserved beta-sheets are colored purple, structurally conserved alpha-helices are colored orange, structurally conserved loops are colored grey, regions that are not structurally conserved are transparent, residues that act as metal ligands are shown and colored green, residues that act as the nucleophile are colored in blue and the conserved metal is shown and colored grey. An inset provides a zoomed in view of the conserved active site residues. **B)** A topology diagram of the core conserved fold indicating the conserved active site residues.

at the C-terminus of strand six. These two residues, along with an additional side chain metal ligand at the C-terminus of strand 1, coordinate a divalent metal ion that lies adjacent in structure to an alkoxide nucleophile side chain. This nucleophilic residue is positioned at the N-terminus of a structurally conserved helix in all known cases and may be serine, threonine, or a cysteine side chain that is post-translationally modified to formylglycine.

A sequence alignment based on a structural superposition of a non-redundant set of protein structures demonstrates sequence conservation of the nucleophile and metal ligands. The core aligned regions around the active site provide a signature for all AP superfamily members that allowed Galperin, *et. al.* to identify that the enzymes are related despite very low sequence identity (figure 3.4, 28). This alignment also clusters the structures into two groupings based on conserved active site configurations. The first major subgroup in the superfamily is the one metal ion binding proteins representing the one metal ion binding proteins from the central cluster as well as the acid phosphatases. The second major subgroup is the two and three metal ion binding proteins. While they bind

Minimal scaffold enzyme	Q5LG14	[30]	L	I	A	L	D	G	[33]	S	S	A	I	N	W	A	[109]	D	H	-	-	T	G	H	[39]	M	T	A	D	H	G	[5]	H	[61]	
Acid phosphatase	2d1g	[38]	I	I	F	Q	E	N	[129]	P	S	T	P	G	A	L	[171]	H	-	G	G	Y	S	-	-	[28]	I	I	Y	D	D	S	[21]	-	[89]
Sulfatase-like	2w8d 2vqr 1p49 1fsu 1hdh 1auk	[35]	Y	V	S	L	E	S	[41]	K	T	S	D	A	E	F	[111]	H	-	F	-	-	P	-	-	[53]	M	Y	G	D	H	Y	[23]	-	[153]
		[36]	L	I	V	V	D	Q	[42]	P	C	G	P	A	R	A	[157]	H	P	-	-	A	F	-	-	[99]	F	T	S	D	H	G	[10]	K	[177]
		[9]	L	V	M	A	D	D	[37]	L	C	T	P	S	R	A	[211]	H	T	-	-	A	L	-	-	[45]	F	T	S	D	Q	G	[11]	-	[228]
		[9]	L	V	M	A	D	D	[37]	L	C	T	P	S	R	A	[211]	H	T	-	-	P	L	-	-	[45]	F	T	S	D	Q	G	[11]	-	[228]
		[8]	V	I	V	A	D	D	[35]	T	C	S	P	T	R	S	[156]	H	W	-	-	P	L	-	-	[99]	F	M	S	D	N	G	[47]	-	[170]
		[8]	V	I	V	A	D	D	[35]	T	C	S	P	T	R	S	[156]	H	W	-	-	P	Q	-	-	[99]	F	M	S	D	N	G	[47]	-	[170]
NPP-like	3q3q 3i01 3nkr 2gsu	[44]	A	I	A	V	D	Q	[37]	E	T	C	P	G	H	S	[207]	D	Y	-	-	-	V	G	H	[37]	L	T	A	D	H	G	[143]	H	[74]
		[27]	A	V	C	L	D	G	[36]	F	T	N	P	N	N	L	[139]	D	Y	-	-	-	V	Q	H	[31]	V	T	A	D	H	G	[124]	H	[47]
		[131]	I	F	S	V	D	G	[35]	K	T	F	P	N	L	Y	[98]	D	F	-	-	-	S	G	H	[39]	F	V	G	D	H	G	[113]	H	[392]
		[10]	L	I	S	I	D	G	[33]	L	T	F	P	N	H	Y	[116]	D	E	-	-	-	A	G	H	[39]	V	V	S	D	H	G	[103]	H	[69]
Mutases	1ejj 3m8z 2zkt	[7]	L	I	I	L	D	G	[47]	N	S	E	V	G	H	L	[337]	D	M	-	-	-	V	G	H	[33]	I	T	A	D	H	G	[15]	H	[49]
		[13]	L	V	V	M	D	S	[69]	D	T	M	T	G	H	W	[197]	D	A	-	-	L	F	G	H	[32]	I	T	A	D	H	G	[9]	H	[55]
		[8]	L	I	I	L	D	G	[44]	G	S	D	T	A	H	L	[241]	D	A	-	-	-	A	G	H	[33]	I	T	G	D	H	S	[7]	H	[56]
Alkaline phosphatase	3tq0	[38]	L	L	I	G	D	G	[48]	D	S	A	A	S	A	T	[221]	D	K	-	-	-	Q	D	H	[34]	V	T	A	D	H	A	[40]	H	[40]

Figure 3.4. Sequence alignment of selected AP superfamily members for which three-dimensional structures have been determined. PDB codes are given, except in the case of the minimal scaffold-enzyme, for which the UniProt ID is given. Conserved nucleophiles and metal ion ligands are marked with **N** and **L** respectively. Numbers in brackets indicate the length of amino acid sequence that is not shown in the alignment.

differing numbers of metal ions, they share a common set of six conserved metal ligands, which are structurally conserved.

3.4.3. A minimal precursor evolutionary model for the AP superfamily

Only considering the main cluster of enzymes, a set of enzymes was identified as forming a hub that connects multiple subgroups within the superfamily using centrality analysis (figure 3.5). The hub-like nature of the relationships of these enzymes suggests that the central group have the most general sequence properties of the larger superfamily. Unfortunately, no experimental or structural characterization has been performed on these enzymes, so any inferences about their functional properties can only be based on sequence. Based on an analysis of the topological variations within the AP superfamily, we identified these central enzymes as having minimal fold topologies; that is, these proteins lack domain insertions relative to the other groups of the AP superfamily (figure 3.4), but retain the necessary sequence motif to bind two metal ions along with the nucleophile. Based on this evidence, we hypothesize that these minimal-fold enzymes may represent a modern-day

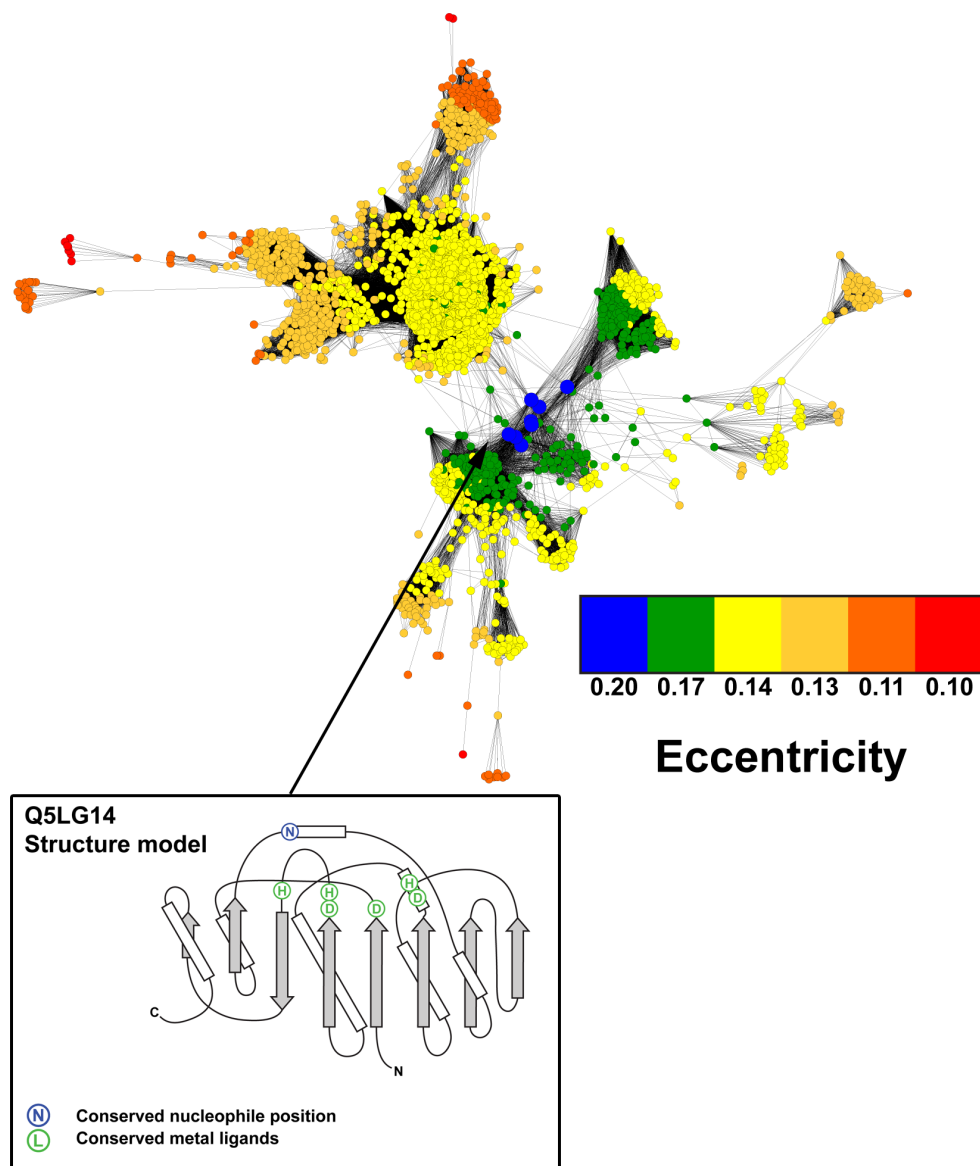


Figure 3.5. Sequence similarity network of the AP superfamily main cluster colored by eccentricity highlights central, minimal enzymes. This network is only the main cluster from figure 3.2 containing a set of 3222 nodes, representing 21626 protein sequences. The edges and layout are is in figure 3.2. Coloring is based on a calculation of node eccentricity (40) which indicates the most central nodes in blue, which have been made larger. A topological model of structure based on sequence alignments for one of the sequences represented by one of the central nodes is shown.

version of an ancestral enzyme that could represent a precursor for the structures and functions in the AP superfamily.

3.4.4. An evaluation of the minimal precursor hypothesis using phylogenetic reconstruction of two/three metal ion AP superfamily members

To evaluate the minimal precursor evolutionary model, we constructed a phylogenetic tree of representative members of the AP superfamily. We focused on the members of the AP superfamily with a conserved two-metal-ion site for several reasons. First, in the AP superfamily sequence similarity network, the one-metal enzymes (including the arylsulfatases, phosphonate monoester hydrolases, glycerol phosphate lipoteichoic acid synthases, and acid phosphatases) clustered separately from the two- and three-metal enzymes (including APs, NPPs, phosphonoacetate hydrolases, phosphopentomutases, phosphoglycerate mutases, minimal diesterases, and NPP-like monoesterases) (figure 3.2). Second, a lack of structural information in sequence clusters that connect one-metal with two-metal enzymes prevented the construction of high quality sequence alignments and thus well-resolved trees that include enzymes from both groups. Lastly, the absolute conservation of six metal ligands in the two/three-metal subgroup (figure 3.4) provided a sequence signal to allow for the construction of a high-confidence curated set of two/three-metal superfamily sequences from UniProt database, which was not possible for the one metal ion superfamily members. Using a PSN of this curated set of enzymes as a basis for

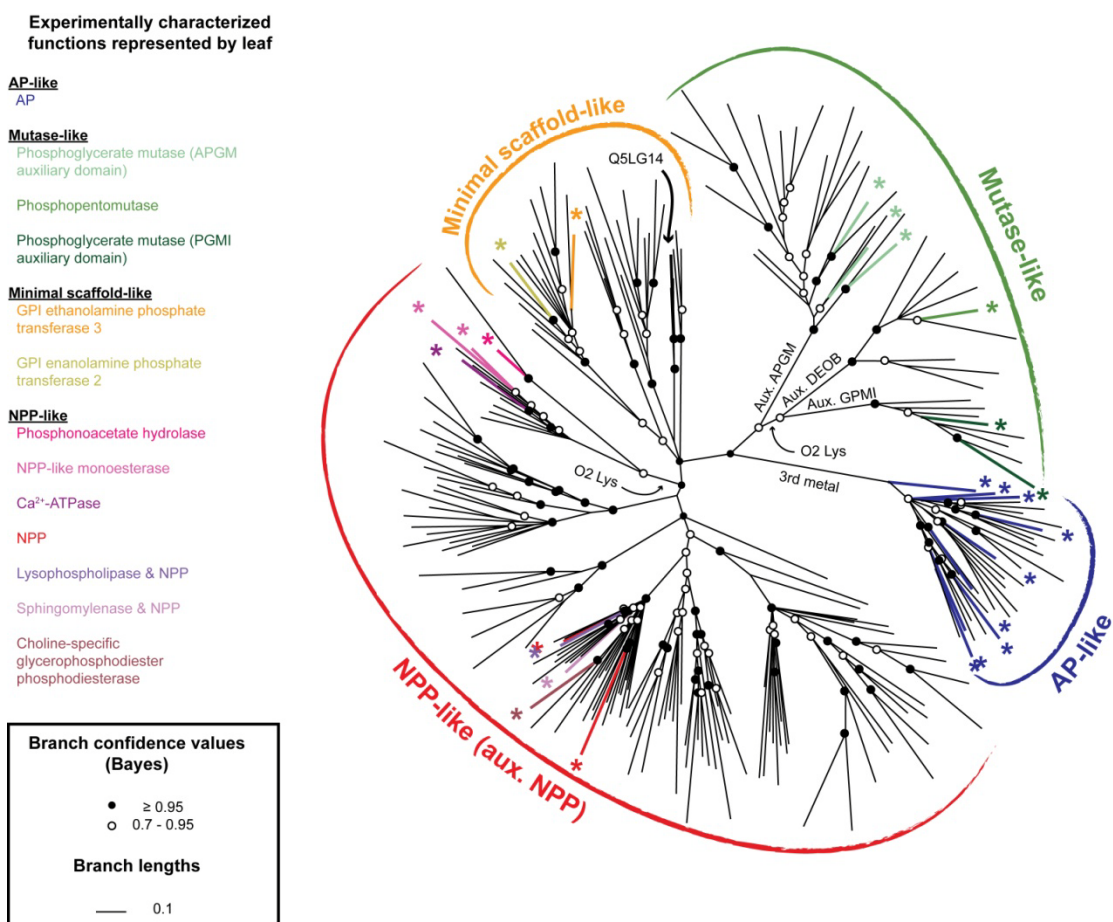


Figure 3.6. Bayesian phylogenetic tree of the proteins in the 2/3 metal subgroup of the AP superfamily. A representative set of 239 proteins was selected from the 7205-protein subgroup by using only proteins where the six metal coordinating residues and nucleophile could be reliably aligned and the sequences shared <45% identity. Leaves are colored and starred according to the function color key if a protein represented by the leaf protein has been experimentally characterized to have such function. The sequences cluster in the tree in four sub-subgroups, NPP-like (red), minimal scaffold-like (orange), mutase-like (green) and AP-like (blue) with similar sequence and structural properties. Structure or active site properties that are common in given clades are indicated with arrows or next to the appropriate branch and discussed further in the text. Branch confidence values are indicated as filled circles (≥ 0.95), open circles (0.7–0.94), or no indication (0.5–0.7). The leaf identities are found in appendix D, table D.4.

analysis (appendix D, figure D.1), we clustered the sequences into groupings that could be reasonably aligned using current automated alignment algorithms. These alignments were, then, aligned to each other in a semi-automated process described in 3.7. Methods. A phylogenetic analysis was performed using this alignment, which is described below and presented in figure 3.6.

All of the major branches of the tree in figure 3.6 contain enzyme sequences present in all domains of life, i.e., Archaea, Eubacteria and Eukaryota (appendix D, figure D.2). This observation suggests that many of the major transformational events in the fold related to present day function could have arisen before the emergence of the major domains of life. Given these ancient divergences, no outgroup can be defined and thus the tree is presented radially. Despite the tree being unrooted, we can still draw several inferences about the evolution of functional groups. For example, mutases and 3-metal ion APs form one high confidence branch of the tree. Experimental characterization has shown these modern-day enzymes to have specificity for monoesters (37, 41). While connections between mutases and 3-metal APs were not seen in the superfamily sequence similarity network (figure 3.2), a relationship is suggested by structure similarity analysis of the two/three metal subgroup (appendix D, figure D.3). In this structure similarity network, the mutases are intermediate with regard to a statistical scoring of the structure alignment between the core domains of the NPP-like structures and the 3-metal APs. As structure is more conserved over evolutionary time-scales than sequence (42), this observation supports the

evolutionary model of this branch presented in the phylogenetic tree. Additionally, there is a high confidence node that distinguishes 3-metal ion and 2-metal ion enzymes on this branch.

All enzymes with the NPP-like auxiliary domain are contained within a single, large branch on the phylogenetic tree and these enzymes can be distinguished from the mutases, APs, and minimal NPPs with high confidence. This observation supports a single emergence for the very similar fold topologies seen in these enzymes. Among the NPP-like enzymes, there is a high confidence branch containing the phosphonoacetate hydrolases and 2-metal phosphatases. There is also a high confidence branch containing the well-characterized NPPs, sphingomyelinase and autotaxin (see 3.5. Discussion), a group of well-studied homologous enzymes present in a wide range of both bacterial and eukaryotic species (43).

Of special interest is the central node, as this may provide the greatest insight into the evolution of the AP superfamily. Several minimal-fold enzymes branch from this central node in addition to the two primary branches discussed above. All of these minimal-fold enzymes remain experimentally uncharacterized, except for a branch that contains glycosylphosphatidylinositol (GPI) ethanolamine phosphate transferases, forms G, N and O (PIGG, PIGN, PIGO) (figure 3.6, appendix D, table D.2). The AP-superfamily fold is fused to another functional domain in some GPI ethanolamine phosphate transferases, but the topology of the AP-superfamily catalytic domain lacks elaboration, as in the topology illustration of a minimal enzyme in figure 3.5 and the other enzymes of

the minimal-fold branch. In addition to forming the central node in the phylogenetic analysis, these minimal-fold enzymes share the highest degree of sequence similarity with the one-metal enzymes, providing a link to sulfatases and other one-metal enzymes. Thus, the minimal-fold enzymes branch directly from a central node in the phylogenetic tree, linking the AP/mutase branch of the tree and the NPP-like branch of the tree, and potentially linking the one-metal enzymes as well. Thus, as further discussed below, the phylogenetic analysis provides independent and strong support for the hypothesis of an ancient enzyme of a minimal enzyme form as a precursor to the AP superfamily.

3.5. Discussion

An evolutionary model was briefly presented in 3.4. Results, but a more full exploration of this model will be presented to demonstrate the use of this information to describe the various evolutionary strategies used in the AP superfamily. This more complete model will then be used to demonstrate how an evolutionary model of the divergence of enzyme mechanism can be used to define functional boundaries for large-scale computational prediction.

3.5.1. An evolutionary model for the AP superfamily

The phylogenetic model allows us to expand upon the hypothesis presented in section 3.4.3 to present a more detailed model for evolution of functional diversity in the AP superfamily. In figure 3.7, fold topologies for canonical enzymes are placed in a simplified representation of the phylogenetic

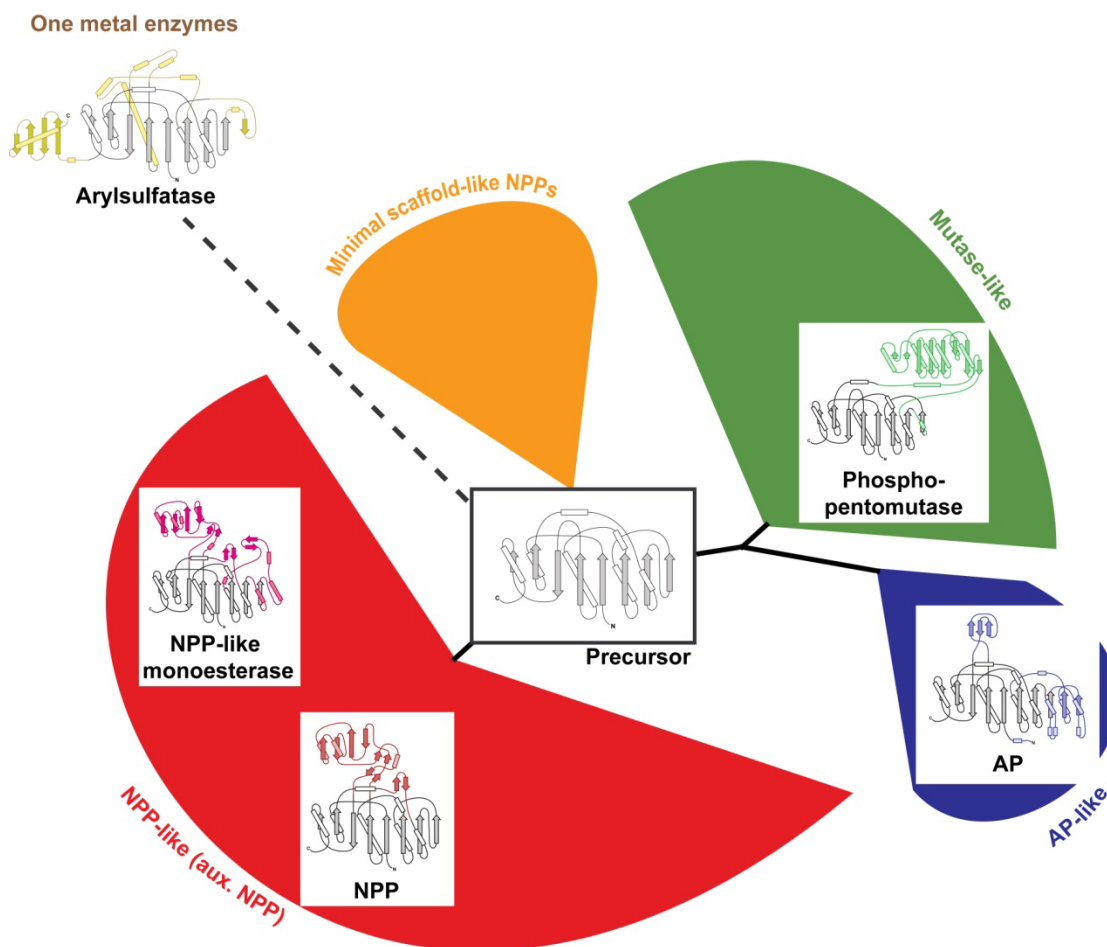


Figure 3.7. Minimal precursor model for evolution of functional diversity in the AP superfamily. Fold topologies of representative enzymes are placed in a simplified representation of the phylogenetic tree shown in figure 3.6.

tree. Branches arise from a minimal-fold precursor protein, the presumed ancestral predecessor to the class of modern-day minimal-fold enzymes. Although the phylogenetic tree is unrooted, this model represents the simplest interpretation of the phylogenetic analysis. In all directions starting with the minimal enzyme fold, insertions are added to the minimal fold to produce new enzymatic entities. As illustrated in figure 3.7, the types and locations of these

insertions differ substantially in each branch of the tree. Further, multiple insertions are required to produce each elaborated fold from a minimal core structure. Thus, although it remains possible that an ancestral precursor to one of the topologically elaborated enzymes *lost* structural elements before acquiring new insertions, the most parsimonious interpretation places the ancestral predecessor to the minimal-fold enzymes as the precursor to all enzymes of the superfamily.

Although the one-metal enzymes could not be included in the phylogenetic analysis due to a lack of requisite structural information, we further speculate that these enzymes were also derived from a minimal-fold predecessor of the modern-day minimal-fold enzymes. We were not able to identify any present-day one-metal enzymes that lack elaboration on the core structure using the Pfam 26.0 Alk_phosphatase clan (CL0088) and searching for a minimal-fold enzyme without the conserved two metal ion binding residues. If such an enzyme does exist, it would differ substantially from other known one-metal AP superfamily members. Thus, the simplest model would place the minimal-fold precursor as the progenitor of the one-metal enzymes in addition to the two-metal enzymes. The ancestors of the present-day minimal-fold enzymes may have bound two metal ions like the modern-day minimal enzymes. In this scenario, the loss of one positively charged metal ion could have accompanied the development of sulfatase activity, as sulfate monoesters carry less negative charge than phosphate monoesters. However, this model is difficult to evaluate because there is currently not enough structural information to allow adequate sequence

alignments between present-day minimal-fold enzymes and the class of one-metal enzymes. Alternatively, minimal-fold enzymes with one-metal sites may have once existed, but were either lost through evolution or remain undetected today.

3.5.2. Strategies for functional divergence in the AP superfamily

Following the evolutionary model presented in figure 3.7 from a minimal fold to the specialized present-day enzymes illustrates that multiple strategies have been used for functional differentiation in the AP superfamily. These strategies include the addition of short insertion regions, the insertion of large nonhomologous auxiliary domains in the mutases, the variation of metal ions and specific side chain contacts, domain fusion and noncovalent interactions with other proteins and domains.

We will briefly focus on examples of these strategies using those instances of active site feature divergence identified in figure 3.6, which appear to be major adducts through which function diverges. The most pronounced instance of functional divergence on the tree comes on the AP/mutase branch. APs catalyze monoesterase reactions using a 3rd metal to stabilize the transition state through indirect contact (by way of a water) with a nonbonding oxygen atom, referred to as O2 in the figure (35). The mutases, on the other hand, catalyze a complex two step reaction involving the participation of an auxiliary domain to rotate and position the substrate (36). Yet, the fundamental nature of the phosphate hydrolysis through the conserved two metal binding site and

nucleophile remains constant. The differences in the catalytic capabilities between these two sets of enzymes are thus due to the various unique elements inserted into a common homologous scaffold.

Three sets of enzyme homologues catalyze monoesterase-like reactions in a similar manner: (1) the APs, (2) the mutases in one half of the mechanism and (3) NPP-like monoesterase. In the case of these three sets of homologues, there are two different solutions that have evolved to interact with the non-bridging oxygen O2. For the APs, the solution just mentioned involving a third metal ion is used. For both some types of mutases and the NPP-like monoesterases a Lys which is structurally equivalent, but evolved independently, provides the equivalent contact as the metal-water in AP.

3.5.3. Definition of functional families in the NPP-like subgroup

So far in this chapter, broad evolutionary models and trends of the AP superfamily have been discussed. The goal of the following section is to demonstrate how this fundamental knowledge can assist in large-scale protein function annotation, an emerging computational challenge in the age of 'big data'.

The binding pockets of NPP-family diesterases recognize different ester substituents, including nucleotides, phospholipid groups, and phosphocholine groups, with diverse biological roles including in bone mineralization, insulin signaling, and cell motility (43). Yet the NPPs maintain the same auxiliary domains, metal sites, and nonbridging oxygen contacts. Where did nature allow

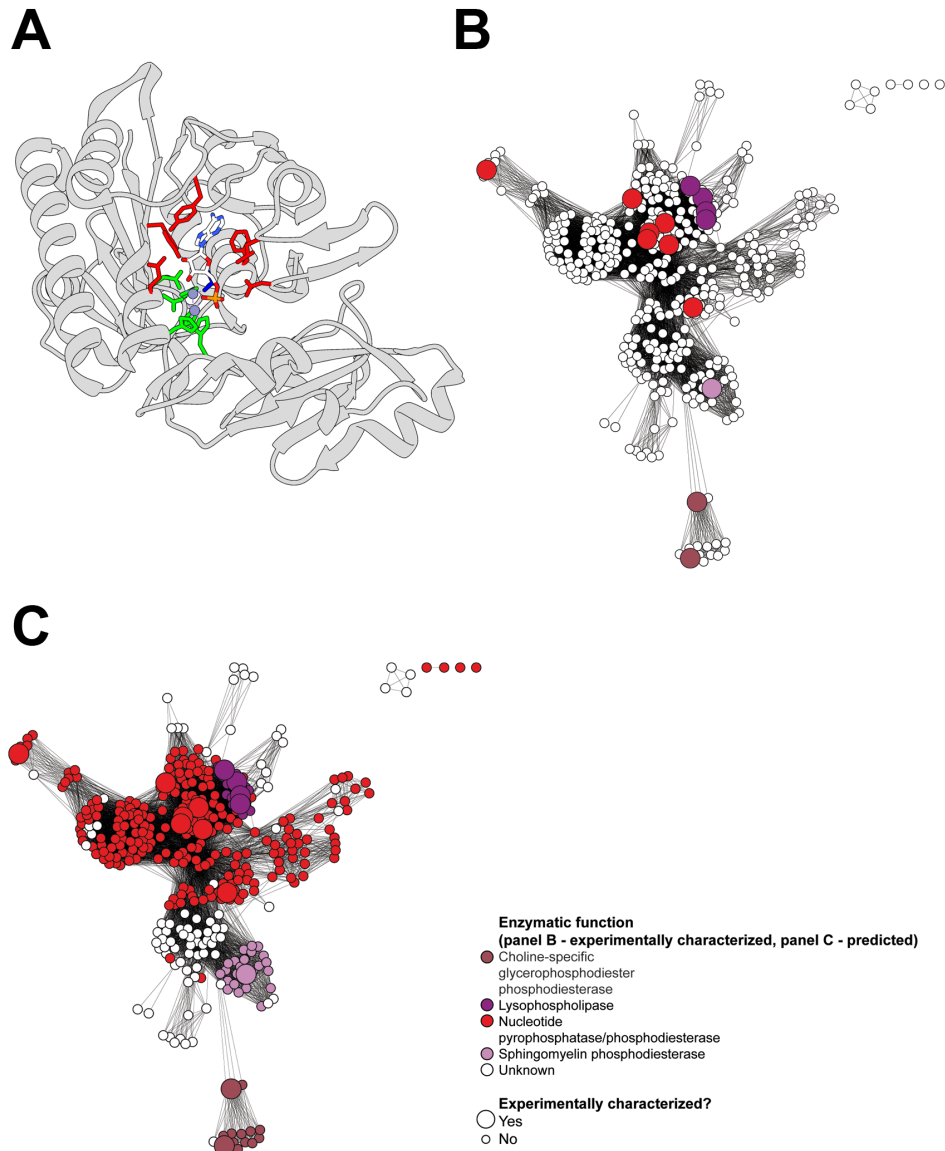


Figure 3.8. Predicting substrate specificity in NPP-like homologues.

A) Crystal structure of NPP from *X. axonopodis* crystallized with adenosine monophosphate ligand (PDB: 2GSU). Conserved metal binding residues are shown in green. Conserved nucleophile is shown in blue. Residues that interact and were used for the motif comparison analysis with the ligand are shown in red. **B)** Sequence PSN of NPP-like homologues. Sequence nodes and edges selected for this network are a subset of the entire 2/3 metal network and are depicted in appendix D, figure D.4. Nodes in this set that were from UniParc were filtered out. Edges between nodes are shown only if $-\log_{10}(\text{E-value}) > 70$ (median alignment length = 401 residues; median percent identifier of pairwise comparisons = 37.9%). This network was visualized in Cytoscape using the organic layout. Sequences with experimental characterization of function are shown. **C)** Network is as in B. Sequences are colored according to predicted

families using residue covariation analysis. This analysis correctly divides all known characterized enzymes into their appropriate family.

variation to produce specific recognition of these different substrates? As with *catalytic* specificity discussed above, multiple strategies were employed to define *substrate* specificity in the NPPs. Both side chain preferences and topological variation produce altered substrate recognition pockets, and exterior domains further modulate biological function.

We analyzed the amino acid preferences for seven substrate-recognition residues in NPPs (figure 3.8A) and found that these preferences predict boundaries for the known enzymatic functions among NPPs. The identities of these seven side chains correlated well with groupings of enzymes in sequence similarity networks (figure 3.8C). Both the sequence similarity networks and motif comparison analyses segregated different experimentally known functions, providing confidence in the resulting clusters (figure 3.8B). The white space in Figure 9B illustrates that several groupings of NPP enzymes remain uncharacterized or minimally characterized experimentally, and this analysis now allows these enzymes to be characterized in a systematic manner, maximizing the potential to understand sequence-activity relationships and identify new biological functions. We present the full motif comparison analysis and additional predictions about NPP substrate specificity in the appendix D, table D.3.

3.6. Conclusions

The extensive mechanistic characterizations of several AP-superfamily enzymes and the large number of diverse functions offer a unique opportunity to provide a detailed computational study of the evolution of a mechanistically diverse superfamily. In this work, we classified sequences in the AP superfamily into a hierarchy where groupings represented shared active site and topological features. We used PSNs to cluster and align sequences in order to calculate a phylogenetic tree of the two/three metal subgroup. We presented a model of evolution for the AP superfamily in which a minimal precursor protein was elaborated into the diverse set of sequences, structures and functions seen in modern day enzymes through the use of several strategies, including loop insertions, domain insertions, loop deletions, point mutations and variable metal binding.

Mapping of sequence, structure, and function annotation within the AP superfamily helped to identify the set of minimal scaffold enzymes that are hypothesized to reflect the ancestral minimal precursor protein. Study of these minimal scaffold proteins could represent an opportunity to understand the early evolution of the superfamily. The network analysis of sequence similarity in the AP superfamily has also elucidated clusters of proteins for which no experimental or structural characterization exists and where further experimental information could guide future efforts in understanding the evolution of the superfamily.

This study raises numerous questions about the evolution of this superfamily and suggests possible paths forward to investigate them. First, what

is the function of the modern-day minimal scaffold proteins? Future experimental characterization of these proteins may help to define the chemistry that a minimal structure can catalyze and suggest a potential progenitor activity for the superfamily. Second, did two-metal or one-metal enzymes arise first and what was the evolutionary path to transition from one type of active site to the other? While the modern day minimal scaffold enzymes are predicted to bind two metals, it is unclear if this reflects the state of the original ancestor minimal proteins. A deeper understanding of the mechanistic differences between one-metal and two-metal sites and may shed light on this question. Finally, how does the structural fold define or influence the range of catalytic and biological activities in the superfamily? This analysis has identified several regions of the AP superfamily that remain uncharacterized. Understanding the full range of catalytic activities of this fold and their relationship to sequence will help to elucidate the intrinsic catalytic potential of the AP superfamily fold and will likely reveal additional enzymes of biological and medical importance.

3.7. Methods

3.7.1. Structure alignments

Structures for the alkaline phosphatase superfamily were collected from Pfam 26.0 (3), clan Alk_phosphatase (CL0088). This set of structures was filtered to a non-redundant set of one structure per UniRef100 release 2011_07 entry (44). This non-redundant set was checked against the VAST structure

similarity search (last check 2/21/2012) to identify additional structures belonging to this fold class (45). A total of 31 structures were collected. One structure for each functional class of enzymes was chosen as a representative for the structure alignment. Two/three metal structures in the superfamily were aligned in UCSF Chimera using the match command on corresponding sets of the six metal coordinating residues and the two coordinated metals (46, 47). Specifically, all two metal enzymes were matched to 3Q3Q.pdb, residues ASP-49, ASP-300, HIS-304, ASP-345, HIS-346, HIS-491, ZN-1559 and ZN-1561. The one metal ion structures were aligned based on the three conserved metal coordinating residues, the conserved metal and the nucleophile. Specifically, all one metal ion enzymes were matched to 1HDH.pdb, residues 13, 51 (atoms CA, CB and OG1), 317, 318 (atoms CA, CB and CG) and 1528. The two/three metal structures were aligned to the one metal based on the match of the three conserved metal coordinating residues, the conserved metal and the nucleophile. The match was between 1HDH.pdb, residues 49, 89 (atoms CA, CB, OG1), 345 and 3Q3Q.pdb. The match -> align program was then used to construct a structure-based sequence alignment using a 4.0 Å cutoff and the criteria to match at least one other structure.

3.7.2. Sequence similarity network of AP superfamily

Sequences were collected using the following HMMs from Pfam (28) that represent the Alk_phosphatase clan (CL0088): Alk_phosphatase (PF00245.14), Metalloenzyme (PF01676.12), Phosphodiester (PF01663.16), DUF1501

(PF07394), DUF229 (PF02995), PglZ (08665), Phosphoesterase (PF04185), Sulfatase (PF00884). These models were run against UniRef100 release 2011_07 (44) using HMMER v3.0 (48) and they were added to the network if they met the bit score listed as the domain gathering cut-off. The entire pairwise BLAST network was calculated using Blast2Seq from BLAST 2.2.24+ with Smith-Waterman trace-back enabled (25). The network was formatted and created using Pythoscape (49). Annotations from UniProt release 2011_07 (3) were collected, including literature citations, taxonomic classifiers, experimental functional characterization, and associated PDB structures. Representative nodes were chosen using CDHIT v4.0 with word count= 2 and percent identity threshold = 0.4 (29). Representative edges were calculated as the max $-\log_{10}(\text{E-value})$ of all edges between nodes represented by two representative nodes. Outputted networks were visualized using Cytoscape v2.8.2 and the organic layout (20).

Eccentricity calculations of central cluster were calculated using the method in this reference (40) using the set of unweighted thresholded edges present at $-\log_{10}(\text{E-value}) > 13.1$.

3.7.3. Sequence similarity network of 2/3 metal enzymes

Sequences were collected using the following HMMs from Pfam (28): Alk_phosphatase (PF00245.14), Metalloenzyme (PF01676.12), Phosphodiesterase (PF01663.16). These models were run against UniRef100 release 2011_07 (44) using HMMER 3.0 (48) and no bit score or E-value thresholds were used.

Instead, hit sequences were added if the model aligned the 2nd through 5th metal coordinating residues (as an example, ASP-300, HIS-304, ASP-345, HIS-346 from 3Q3Q.pdb). Based on sequence alignments (described below) sequences were further filtered out if they did not contain the 1st and 6th metal coordinating residue, ASP and HIS respectively (as an example, ASP-49, HIS-491 from 3Q3Q.pdb), or nucleophile (as an example, THR-89 from 3Q3Q.pdb), SER or THR. The entire pairwise BLAST network was calculated using Blast2Seq from BLAST 2.2.24+ with Smith-Waterman trace-back enabled (25). The network was formatted and created using Pythoscape (49). Annotations from UniProt release 2011_07 (3) were collected, including literature citations, taxonomic classifiers, experimental functional characterization, and associated PDB structures. Representative nodes were chosen using CDHIT v4.0 with word count= 5 and percent identity threshold = 0.7 (29). Representative edges were calculated as the mean $-\log_{10}(\text{E-value})$ of all edges between nodes represented by two representative nodes. Outputted networks were visualized using Cytoscape v2.8.2 and the organic layout (20). Edges that did not represent BLAST E-value connections between alkaline phosphatase superfamily domains were removed. Some nodes represented proteins that were fusions between two members of alkaline phosphatase superfamily members. In this case, only edges to the best-hit superfamily cluster were kept.

3.7.4. Sequence alignments

The sequence similarity network of 2/3 metal enzymes was filtered to include only those edges with $-\log_{10}(\text{Evalue}) > 60$. Those node groupings containing more than 10 sequences were aligned using MAFFT-linsi v6.860 (50). These groups were then aligned as profiles using MUSCLE v3.7 from which functional residues (six metal coordinating residues and the nucleophile) were identified (51). For a given group, if the presence of any of those functionally important residues was ambiguous, the group was removed from further analysis. The sequence profiles were then curated to include only those sequences with all functionally relevant residues. Profile-profile alignments of all sequence groupings together were performed based on network topology. Alignments were refined by eye after each profile-profile alignment depending on known structural information. Non-aligning regions were removed prior to each iteration of the profile-profile alignment.

3.7.5. Two/three metal group tree

Alignments representing all 2/3 metal sequences where the six metal ligands and nucleophile could be reasonably aligned were filtered so that no two aligned sequences shared greater than 45% sequence identity using CD-HIT v4.0 and word count = 2 (29). These alignments only included the core, structurally conserved regions, present across this set of superfamily sequences. The alignment procedure was facilitated because of the strong signal from the six

metal coordinating residues and reasonable structural coverage of major sequence clusters apart from the minimal class.

These alignments were inputted into MrBayes v3.1.14 (17). All parameters were left to default except for temperature, which was lowered to 0.07 to improve convergence; prior frequency model, which was set to WAG; rate distribution, which was set to gamma; and rate bins, which were set to 6 to improve the results given the hypothesized differences in rates between alignment columns. MrBayes was run 6,000,000 generations to a final standard deviation of split frequencies of 0.034. Convergence was also measured using AWTY (52). From these calculations and from the final standard deviation, it was clear that the tree as a whole had not converged, but getting to absolute convergence of a tree representing such divergent sequences would be challenging under any set of circumstances.

In this case, we had confidence to present this tree for the following reasons: (1) the divergences that we considered most important for our proposed evolutionary model, i.e. the central divergences, were ubiquitously present in both runs as measured by MrBayes, (2) the tree as proposed by MrBayes also matches several other lines of evidence as presented in the results section.

Chapter 4. Investigating caspase substrate evolution using kinases as a model system

4.1. Abstract

Discovering targets of caspases, enzymes activated during apoptosis that hydrolyze peptide bonds after aspartate residues, provides a method to understand the biology of cell death and human diseases that relate to cell death. Current high-throughput experimental techniques exist to determine the targets of caspases, but these methodologies are limited in their ultimate reach. But, they provide rich datasets to test and train novel computational methods that can extend current knowledge to predict putative targets that can be verified experimentally.

In this chapter, I present preliminary work developing computational tools to predict caspase targets using sequence similarity, motif conservation and *de novo* motif prediction. Using kinases as a target set, I demonstrate that there is a specific set of kinase homologs, the STE kinases, which are preferentially cleaved. I demonstrate that sequence similarity and motif conservation will be limited to predicting only targets that are very similar homologs to known targets, which will limit their ultimate utility. I also demonstrate that extending current methods that predict which motifs in sequences are cleaved to predicting which protein sequences are actually cleaved in a biological context is still problematic.

4.2. Introduction

Apoptosis, a form of programmed cell death, is a critical cellular pathway known to be found in metazoan species. It is activated to carry out numerous

roles in these organisms including organogenesis, tissue remodeling, regulating cell number, cellular remodeling and elimination of diseased or defective cells. Given the importance of apoptosis in the development and maintenance of complex, multi-cellular life forms, the process of apoptotic activation, regulation and completion is intensely studied in the search for novel therapeutics for disorders such as cancer, autoimmune and neurodegenerative diseases and infections (53, 54).

The exact nature of the regulation and activation of apoptosis can vary widely depending on the organism, but in general, extrinsic or intrinsic cellular signals converge by activating caspases, which are cysteine proteases that cleave a variety of downstream targets, which are only partially known (53, 54). Because the end phenotype of apoptosis depends on the cleavage of the targets of caspases, enumerating these targets will provide a clearer understanding of the mechanism of apoptosis and how to chemically modulate cell death (55). For this reason, there is considerable interest in predicting caspase targets computationally. Most current work focuses on caspase substrate specificity based on small linear motifs in the target surrounding a preferred aspartate in the P1 position (54). For example, it has been shown that caspase specificity depends both on the identities of the surrounding residues and predicted or experimentally determined secondary structure and that these features can be used for computational prediction of cleaved substrates (54, 55).

Traditionally, one of the most successful computational techniques to predict function has been the use of evolutionary inference. That is, if two

biological entities are homologous (that is they are not only similar, but share a common ancestry), then aspects of function are often conserved between the two entities. So far this technique has not been integrated into computational predictors of caspase substrates, most likely because the evolution is acting on short linear motifs, which are known to evolve relatively quickly (56). This rapid evolution can complicate locating and having confidence in the prediction of homologous sites.

In this chapter, I will focus on the computational prediction of caspase substrates using target site homology and human kinases as a model system for testing. Kinases are enzymes involved in signaling pathways and have been shown to be targets of caspases whose function is modulated upon cleavage. Moreover, several kinases, including protein kinase C, ROCK1 and PAK2, are activated on cleavage and contribute to driving apoptosis to completion (57). These gain of function cleavages are ideal targets to study because they are thought to be more likely to be conserved because of the functional consequence of cleavage.

Even though it is a reasonable hypothesis, I show that motif conservation and sequence similarity of targets will be a poor predictor of cleavage, except in cases of near homology, because of the rapid evolution of the short linear motifs. This holds true even in gain of function cleavages. I show that current motif prediction methodologies are currently not powerful enough to be used for large scale prediction of caspase substrates in a biological context. I propose these sorts of *ab initio* predictions (that is, without evidence of homology) of cleavage

could be more useful when coupled with other orthogonal biological information, such as cellular compartment of the target. I also propose that protein interaction networks could be used instead of sequence similarity networks to enhance predictive algorithms.

4.3. Results

I first demonstrate that human STE kinases are the only statistically enriched group of human kinase homologues. Further, I demonstrate that there is almost no conservation across the caspase target sites in this functionally important set of homologues and propose that the cleavage motifs evolved independently from each other. Finally, I posit the use of computational caspase motif prediction to predict novel kinase targets and show that current methods are not sufficient for the task.

4.3.1. Human STE kinases are statistically enriched for caspase substrates

Contrary to what was expected, when querying high-throughput determined human caspase targets (58) for Pfam (28) domain enrichment using a Chi-Squared test, Pkinase (PF00069) domains were not enriched in the target set (appendix E, figure E.1). This was surprising given the known importance of kinases as targets of caspases (57).

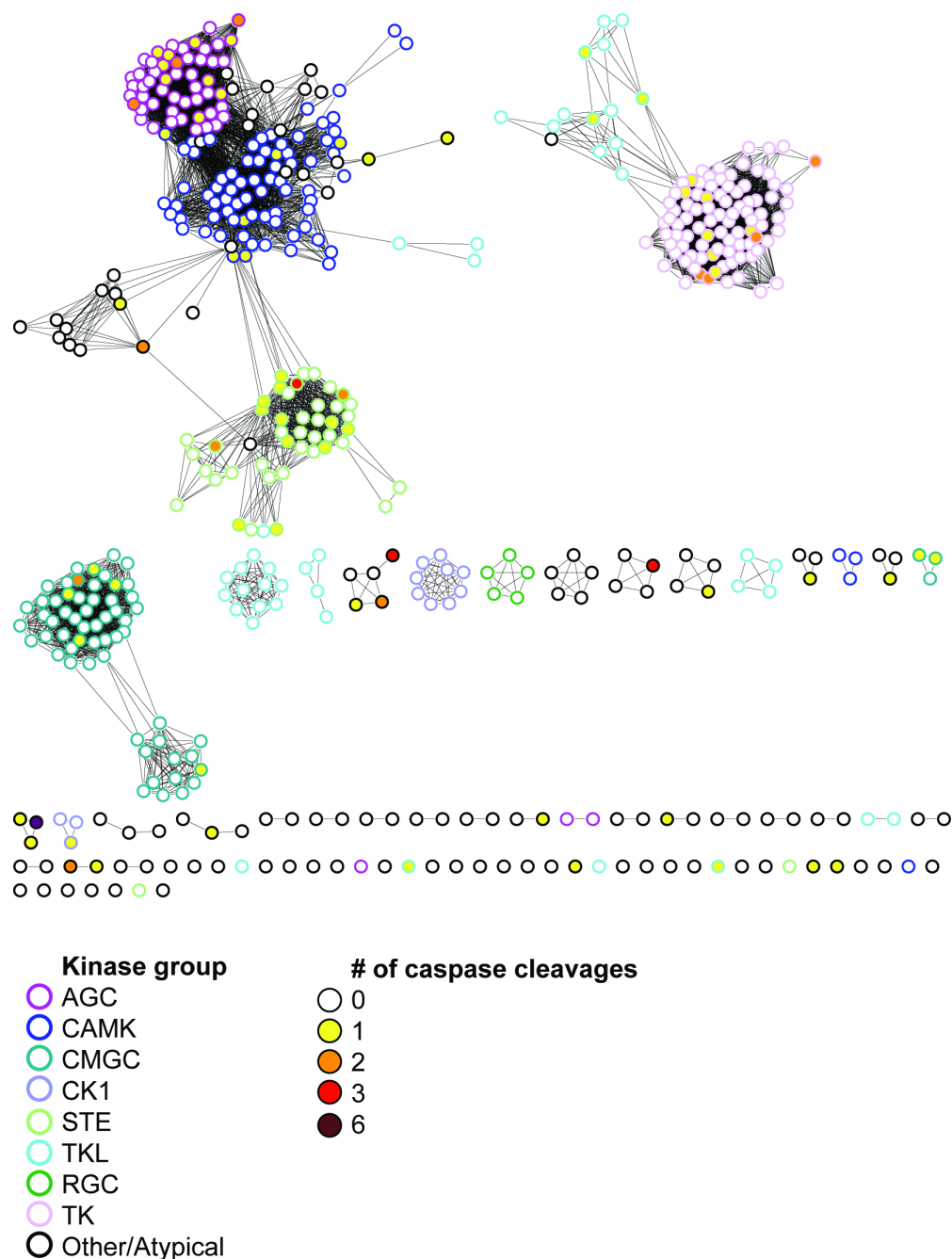


Figure 4.1. Sequence similarity network of the Manning kinome colored by caspase cleavages. Sequence PSN of the 515 full length *Manning, et al.* kinase sequences (59). Edges between nodes are drawn only if $-\log_{10}(\text{E-value}) > 40$ (median alignment length = 280 residues; median percent identity of pairwise comparisons = 35.8%). The network is visualized using the organic layout in Cytoscape. Node outline coloring is based on the Manning classification. Node interior coloring is based on the number of known caspase cleavage sites for that

target protein. Some groups like the CAMK group are barely cleaved, while other groups, like the STE group are frequently cleaved.

To investigate this further, I created a sequence PSN of human kinases using the Manning kinase set and colored by the manning classification, a classification based on predicted common ancestry (59). To annotate this PSN, I used a database of the number of cleavages for each human kinase known based on a combined set of high throughput data and literature searching (appendix E, table E.1). This network demonstrates that certain homologous groups of kinases are known to be cleaved with high frequency and some have very few known cleavages (figure 4.1). While it would be difficult to interpret the negative information, as high throughput studies are known to have enrichment biases, it is possible to look for trends in the known data.

Performing a Fisher's exact test and correcting for multiple testing on the enrichment of targets in the kinase groups shows that in one set, the STE kinases, has a statistically significant enrichment of caspase targets (table 4.1). STE kinases are a homologous group of kinases overwhelmingly found in the MAP kinase pathway and implicated in the regulation of cellular proliferation, differentiation and death (59, 60).

Table 4.1. Fisher's exact test to determine positive enrichment of caspase targets in Manning kinase groups.

Group	Number cleaved	Total proteins	P-value (raw)	P-value (corrected)
AGC	12	63	0.24	
CAMK	6	80	0.99	
CMGC	8	61	0.75	
STE	14	47	0.0074	0.05
TKL	5	43	0.82	
TK	11	90	0.70	
CK1	1	12	0.86	

4.3.2. Cleavage sites are only rarely conserved between human STE kinases

Even though a group of homologous kinases, the STE kinases, showed a statistically significant enrichment of caspase targets, this doesn't necessarily imply that the sites that give rise to those cleavages represent homologous sites. To test whether the sites were conserved between STE kinases, I used meNOG alignments from the EGGNOG database (61) to check for alignment of known cleavage sites. While these alignments do not cover all possible all-by-all combinations, they do align near homologues. Because the sites are usually in regions that are only shared between near homologues, this is not thought to skew the analysis. This was checked by aligning all human STE kinases in

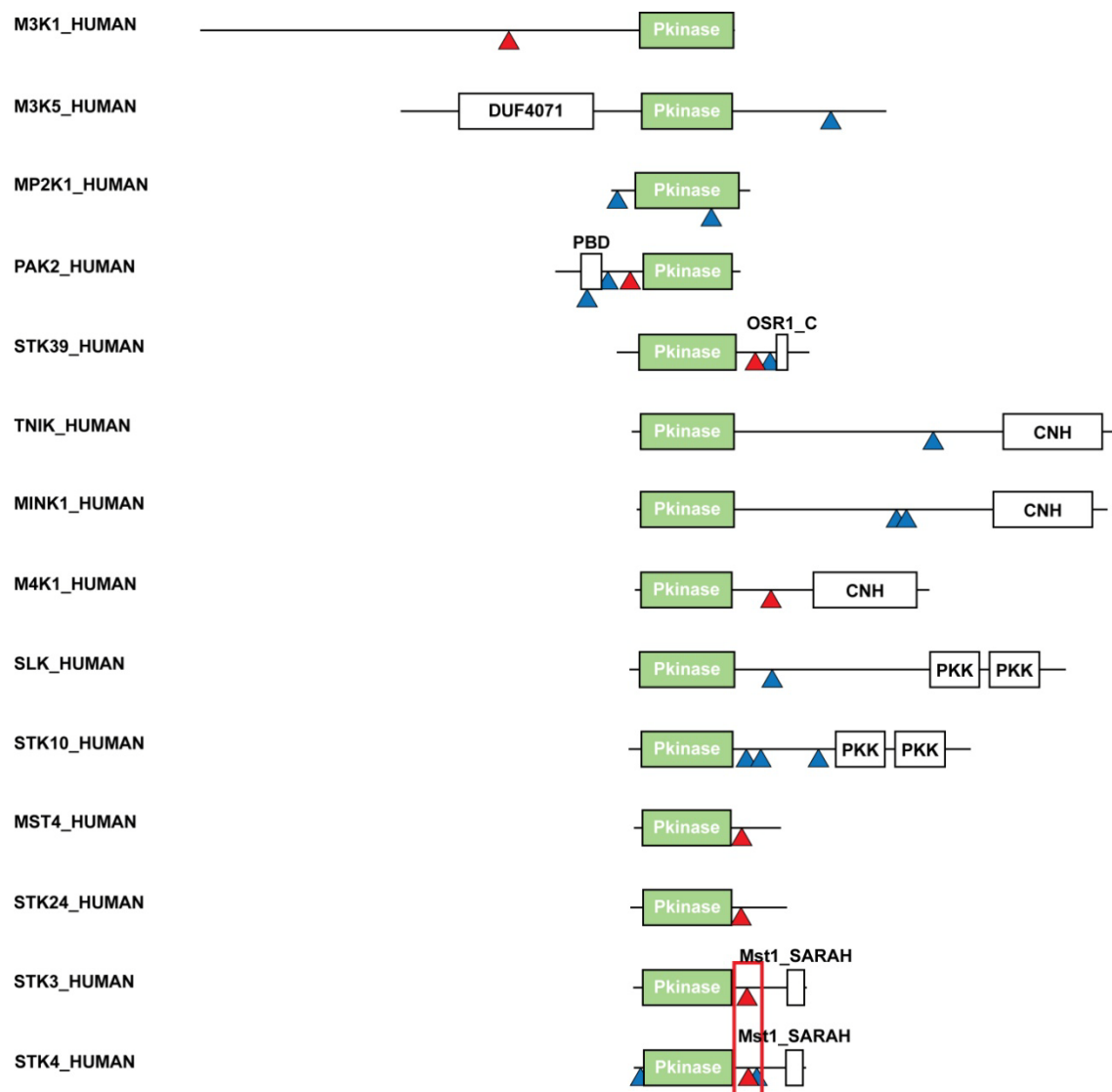


Figure 4.2. Little conservation of known caspase cleavage sites in STE kinases. Each known cleaved STE kinase is depicted as a cartoon where the length of the line is scaled to the length of the protein sequence. Pfam A (28) domains and their scaled length and position are depicted as boxes with the name of the domain given with the absolutely conserved Pkinase domain in light green. Caspase cleavage sites are indicated with triangles; red indicates an experimentally and functionally characterized cleavage and blue indicates a cleavage from the high throughput data. The single pair of cleavages that align according to EGGNOG (61) are highlighted with a red box.

MAFFT-linsi (50) and confirming that these alignments do not under represent all possible motif alignments (data not shown). This analysis was mapped onto a diagram of Pfam domains to show that motifs are not conserved between homologous proteins that are cleaved, usually because they are in different portions of the protein (figure 4.2). In only one case, was the motif conserved between homologous cleaved STE kinases. This contradicts the expected result that homologs with the same gain-of-function impact upon cleavage would have conserved cleavage sites. In fact, it appears that in the great majority of instances, these sites with similar end results in homologous proteins would have evolved independently of one another.

4.3.3. Preliminary computational prediction of kinase targets of caspases

To predict kinase targets of caspases not using conservation, I instead needed to rely on algorithms that predict protein substrates using other information. Previously published methods have been able to make predictions about the suitability of an inputted eight amino acid motif to be cleaved by caspases using sequence and structure information (62). But to predict which protein substrates are cleaved during apoptosis, these methods would need to be extended to predict whether or not any site on a protein substrate is cleaved within its relevant biological context. While this is a much more difficult problem that could depend on many factors outside of just motif content including when and where the protein substrate is expressed, I first investigated how accurately

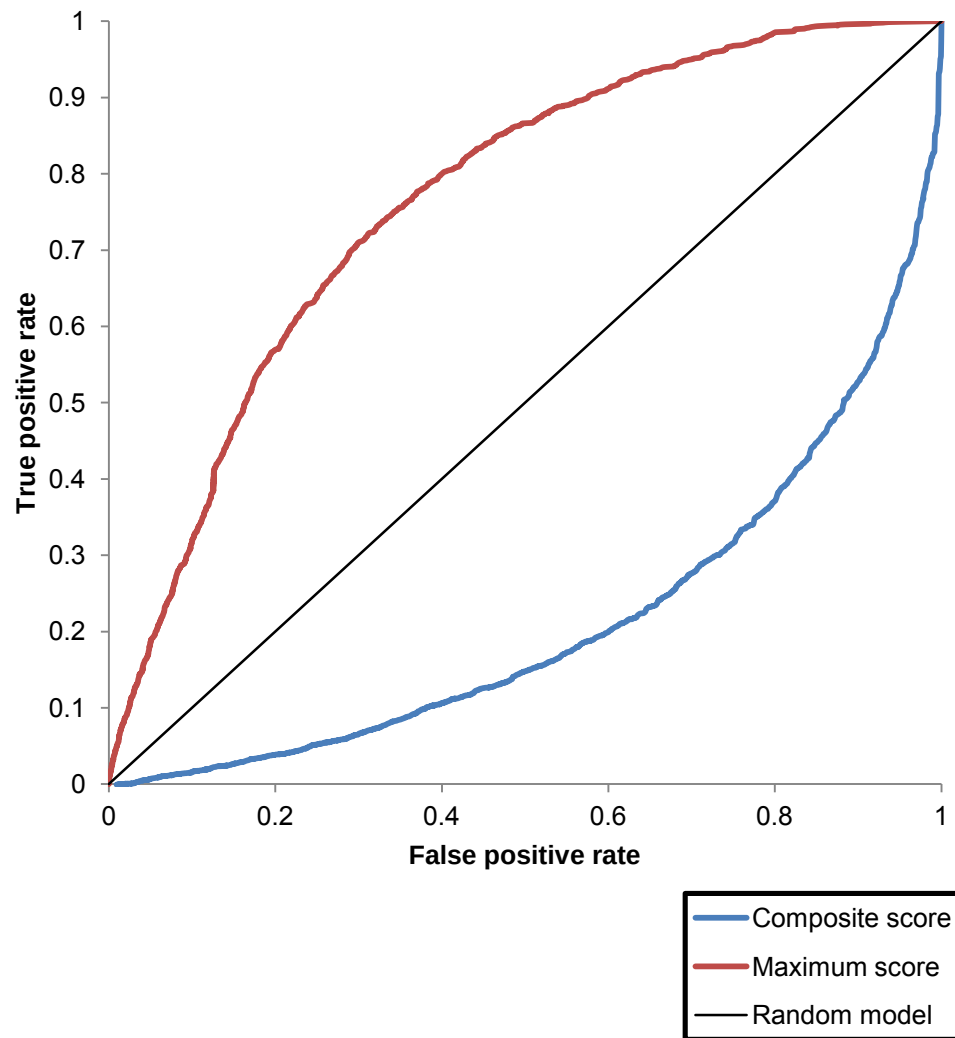


Figure 4.3. ROC plot of methods to score entire protein sequences as targets of caspases. Two methods to predict caspase protein targets are compared. The maximum score model (red) uses the maximum SVM score of any possible cleaved motif as the score for the target. The composite score model (blue) uses a probabilistic score using a combination of all possible cleaved motifs as the score for the target. The curve for a random model (black) is given to compare to. The maximum score is somewhat predictive and the composite score is negatively predictive.

protein targets could be predicted just using algorithms that focus on the motifs.

Using the calculated support vector machine (SVM) scores for all possible

human eight amino acid sequence motifs with aspartate at the P1 position (62), I tested whether these scores could predict a known set of caspase substrates from high throughput experiments of human cell lines (58). For the sake of this exercise, I assumed that all substrates not known to be cleaved are, in fact, not cleaved. While this is certainly an incorrect assumption, this approximation gives a basis to begin investigating the value of this predictive method. In addition, the results presented in this section are only preliminary, as I have calibrated and predicted on the same set of data. For the results from these tests to be considered robust and accurate, I would need to perform cross-validation tests. Ultimately, the desire for these preliminary results was to provide insight into the nature of the challenges that need to be overcome to build useful predictive methods.

For the first method, I scored each putative target sequence based on the maximum SVM score from the Barkan, et al. method of any possible cleaved motif in that target. For the second method, I used a composite score where we calculate the probability that any given target is cleaved based on the following formula:

$$1 - p_{seq} = \prod_i (1 - p_{seq,i}) \quad (4.1)$$

where p_{seq} is the probability that a target is cleaved and $p_{seq,i}$ is the probability that each motif in the target sequence is cleaved. This equation calculates the

Table 4.2. Table of top ten predictions from the composite scoring method, their cellular location and if they are a known caspase target

UniProt ID	% chance cleaved (composite scoring method)	Cellular location	Known target?
Q9NZW4	0.97	Extracellular	NO
Q9NZR2	0.83	Membrane	NO
P98164	0.82	Membrane	NO
Q09666	0.8	Nucleus	YES
Q8NEZ4	0.77	Nucleus	YES
Q07954	0.77	Membrane	NO
P13611	0.76	Extracellular	NO
Q9Y6V0	0.76	Membrane	NO
Q8N3K9	0.75	Cytoplasm	NO
Q6UB99	0.74	Nucleus	NO

probability that a target isn't cleaved and then converts that value to the probability that the substrate is cleaved. This strategy is used because the probability of not being cleaved is the simple product of all the probabilities that any given motif isn't cleaved. For this equation to be true, it has to be assumed that all motifs are independent of one another.

The receiver-operator characteristic curves (ROC) of these methods demonstrate a surprising result. The first method, the maximum score method, is slightly predictive with an area under the curve (AUC) of 0.76 (figure 4.3). The

second method, the composite score method, is negatively predictive as the curve is concave under the random model line (figure 4.3). This implies that the method assumed to incorporate more information about all possible motifs counter-intuitively better predicts the opposite of what we would expect. When investigating the top ten results from this method, i.e. the targets with the highest likelihood of being cleaved, many of the targets with incorrect predictions were incorrectly predicted because they are not accessible to caspases during apoptosis (table 4.2). Six of the ten are either membrane-imbedded receptors or extra-cellular proteins. This implies that this method in its current form will not work because it lacks all the appropriate information to properly classify targets. A discussion of possible information types that could be added to make a more predictive algorithm is included in the discussion in 4.4.2.

4.4. Discussion

Though the results are preliminary, they point to two conclusions. The first is that using motif conservation to predict novel cleavage sites will only be useful in limited circumstances where there are close homologs of known targets. The second is that current computational techniques to identify putative cleaved motifs that do not take homology into consideration will only be useful as large-scale computational tools if they are combined with additional information that increase their predictive power.

4.4.1. Motif conservation is a poor computational tool to predict caspase targets, except in cases of close homologues

While motif conservation was initially thought to be useful as a method to predict caspase target substrates, it has been shown to be useful in only a limited number of cases in the kinases studied here. A recent experimental exploration of caspase motif and target conservation among two vertebrate species, human and mouse, and two invertebrate species, *C. elegans* and *D. melanogaster*, demonstrated much the same result (63). In the work for this study, which I contributed to but is not included in this dissertation, it was demonstrated that caspase target motif conservation was frequently seen between closely related species, i.e. human and mouse. In the cases of comparing humans to invertebrate species, caspase target motifs were only rarely conserved. Instead, more often the same protein was cleaved in a different location. In many of the other cases when the same protein substrate was not conserved, the two proteins were in the same pathway. The results in this paper combined with the results in this chapter show that using motif conservation for large scale prediction of caspase sites would be problematic. Instead, other sorts of information, like pathway membership which do not require motifs to be conserved, will need to be used.

As an example, in these results, it is demonstrated that STE kinases are statistically enriched for caspase cleavage sites (table 4.1), but that this not due to conserved cleavage motifs (figure 4.2). This implies that sequence similarity and homology between the targets might not underlie the true reason for this

enrichment. Another, perhaps more likely reason for this statistical enrichment, could be that the great majority of STE kinases are found in a single pathway, the MAP kinase pathway (59, 60). The ultimate cause for this enrichment might be that there is selective pressure for members of this pathway to be cleaved during apoptosis. Further exploration would have to be performed to validate that hypothesis, but is reasonable to postulate that the more valuable method to predict targets wouldn't be sequence PSNs, but pathway networks.

4.4.2. Computational prediction using motif identification is currently insufficient to identify putative target sets

The results demonstrate that current methods are not up to the task of predicting caspase substrates on the whole proteome yet, but they also demonstrate what improvements could be made to enhance the predictive power of these algorithms (figure 4.3, table 4.2). First, the integration of cellular context information could enhance the ability of the algorithm to distinguish between targets that caspases could cleave from those that are not accessible during apoptosis. Many of the high scoring false positives are proteins that could be weeded out during this process with context information (table 4.2). Second, the integration of cell type information could also enhance the ability to train the algorithm appropriately. The experiments that created this dataset were done in specific cells, but the training is currently done agnostically to that information. Training based on a subset of proteins that could have actually been cleaved based on cell type could enhance the power of the algorithm. Finally, as

discussed in the previous section, pathway membership is thought to be an important factor in which proteins are cleaved, as pathway membership is conserved among targets over longer distances than either homologous relationships or motif conservation (63).

If such a method could be constructed, an immediate application would be in expanding the small sets of currently known targets in species other than humans. For example, there are enough known targets in *D. melanogaster* to provide data for training an algorithm that had proven to be effective in humans (63). By comparing predicted sets in human to predicted sets in *D. melanogaster*, it would be possible to investigate if certain pathways have been targeted in a much larger and more complete way than previous experimental investigations have been able to deduce.

4.6. Conclusions

Progress in understanding how activated caspases are able to transform a normal cell to an apoptotic body would be dramatically accelerated by knowing the complete complement of caspase targets. While experimental work has made initial progress, computational algorithms could assist in filling out the still unknown details.

Using kinases as a test system, I have shown that motif conservation would only be of limited assistance in this algorithm as the motifs evolve too quickly to be of use to annotate evolutionarily distant homologs. Methods that use motif predictions to predict substrates are also currently not able to robustly

distinguish targets from non-targets, but the data suggest that these methods could be augmented using other known information such as cellular context or pathway membership to create much more predictive algorithms.

4.7. Methods

4.7.1. Sequence similarity network of kinases

Full length sequences and grouping annotations were collected from the Manning, et al. set (59). Cleavage annotations were collected from a hand curated literature dataset (appendix E, table E.1) that contains 36 known cleavages and 44 protein targets (some where the exact site of cleavage is unknown) and from an experimental high-throughput dataset that contains 2,175 known cleavages and 1,527 protein targets (58). The entire pairwise BLAST network was calculated using Blast2Seq from BLAST 2.2.24+ with Smith-Waterman trace-back enabled (25). The network was formatted and created using Pythoscape (49). Outputted networks were visualized using Cytoscape v2.8.2 and the organic layout (20).

4.7.2. Enrichment analysis

Statistical enrichments for the full high-throughput dataset (58) were calculated by calculating the sum of each Pfam domain (28) found in the hit UniProt 2012_06 (3) records and in all UniProt records. Chi-square tests were

done using the Yates's correction, where the observed number of domains was the sum from all hits and the expected number was the proportion of total domains found in UniProt to the total number of proteins multiplied by the total number of hit proteins. P-values were only calculated for domains where the observed number was greater than 10. Each p-value was corrected using the Bonferonni method, i.e. multiplied by the number of tests.

Statistical enrichments for the caspase targets in the kinase dataset were calculated using the Fisher's exact test based on the groupings published in the Manning, et al. reference (59). Cleavage annotations were identical to 4.7.1. The observed number was the number of annotations of cleavage for each group and the expected number was the proportion of total cleaved kinases to the total number of kinases multiplied by the total group number. P-values that were <0.01 were corrected using the Bonferonni method, i.e. multiplied by the number of tests.

4.7.3. Kinase motif alignments

Kinase motif alignments were taken from the corresponding their meNOGs in the EGGNOG database (61). The motifs were considered to be aligned if the P1 aspartate in both motifs was aligned in the meNOG alignment. Because the meNOG alignments were not a comprehensive all-by-all set, we did a MAFFT-linsi alignment (50) of all STE kinases to verify that this assessment wasn't biased.

4.7.3. Substrate scoring using motif scores

For these methods, we used the Barkan, et al. predicted human dataset (62). In this dataset, 518,731 motif SVM score predictions are made from a set of 20,041 putative target sequences. 2,128 motif SVM score predictions are made for sites in 1,528 protein targets that are known to be cleaved in the experimental dataset from 4.7.1. To calculate the ROC plots for the maximum score, we scored each of the 20,041 protein targets based on the maximum SVM score of any motif in that protein. Using the TP (true positive) number, the FN (false negative) number, FP (false positive) number and TN (true negative) number, we calculated and plotted the TP (true positive) rate, $TP/(TP+FN)$, and false positive rate, $FP/(FP+TN)$, for each possible score cutoff.

To calculate the composite score, we binned all possible scores based on the reported false positive rate into bins for each 0.01 increment. For each bin, we calculated the probability that each score in that bin would be cleaved using the expected number of true positives and false positives to be the difference between the minimum and maximum of the bin, R , multiplied by the total number. So, $p_{seq,i} = (\text{number cleaved})/(\text{total number})$, would be $p_{seq,i} = (2,128 * R_{TPR})/((2,128 * R_{TPR}) + (516,603 * R_{FPR}))$. The calculated $p_{seq,i}$ values were used to create a ROC plot just as for the maximum score.

Bibliography

1. E. S. Lander *et al.*, *Nature* **409**, 860 (Feb 15, 2001).
2. J. C. Venter *et al.*, *Science* **291**, 1304 (Feb 16, 2001).
3. *Nucleic Acids Res* **40**, D71 (Jan, 2012).
4. V. M. Markowitz *et al.*, *Nucleic Acids Res* **40**, D123 (Jan, 2012).
5. E. S. Lander, *Nature* **470**, 187 (Feb 10, 2011).
6. *Nature* **467**, 1061 (Oct 28, 2010).
7. M. Arumugam *et al.*, *Nature* **473**, 174 (May 12, 2011).
8. C. L. Hemme *et al.*, *ISME J* **4**, 660 (May, 2010).
9. M. Bredel, E. Jacoby, *Nat Rev Genet* **5**, 262 (Apr, 2004).
10. A. M. Schnoes, S. D. Brown, I. Dodevski, P. C. Babbitt, *PLoS Comput Biol* **5**, e1000605 (Dec, 2009).
11. J. A. Eisen, *Genome Res* **8**, 163 (Mar, 1998).
12. M. Biswas *et al.*, *Brief Bioinform* **3**, 285 (Sep, 2002).
13. S. C. Pegg *et al.*, *Biochemistry* **45**, 2545 (Feb 28, 2006).
14. H. Yao *et al.*, *J Mol Biol* **326**, 255 (Feb 7, 2003).
15. N. Furnham *et al.*, *Nucleic Acids Res* **40**, D776 (Jan, 2012).
16. N. Krishnamurthy, D. P. Brown, D. Kirshner, K. Sjolander, *Genome Biol* **7**, R83 (2006).
17. G. Altekar, S. Dwarkadas, J. P. Huelsenbeck, F. Ronquist, *Bioinformatics* **20**, 407 (Feb 12, 2004).
18. Z. Yang, *Mol Biol Evol* **24**, 1586 (Aug, 2007).
19. C. Kemena, C. Notredame, *Bioinformatics* **25**, 2455 (Oct 1, 2009).
20. M. E. Smoot, K. Ono, J. Ruschinski, P. L. Wang, T. Ideker, *Bioinformatics* **27**, 431 (Feb 1, 2011).
21. S. D. Brown, P. C. Babbitt, *J Biol Chem* **287**, 35 (Jan 2, 2012).
22. H. J. Atkinson, J. H. Morris, T. E. Ferrin, P. C. Babbitt, *PLoS One* **4**, e4345 (2009).
23. T. Rattei *et al.*, *Nucleic Acids Res* **38**, D223 (Jan, 2010).
24. T. Wittkop *et al.*, *Nat Protoc* **6**, 285 (Mar, 2011).
25. C. Camacho *et al.*, *BMC Bioinformatics* **10**, 421 (2009).
26. B. Mannervik, U. H. Danielson, *CRC Crit Rev Biochem* **23**, 283 (1988).
27. H. J. Atkinson, P. C. Babbitt, *Biochemistry* **48**, 11108 (Nov 24, 2009).
28. M. Punta *et al.*, *Nucleic Acids Res* **40**, D290 (Jan, 2012).
29. W. Li, A. Godzik, *Bioinformatics* **22**, 1658 (Jul 1, 2006).
30. Y. Zhang, J. Skolnick, *Nucleic Acids Res* **33**, 2302 (2005).
31. M. E. Glasner, J. A. Gerlt, P. C. Babbitt, *Curr Opin Chem Biol* **10**, 492 (Oct, 2006).
32. M. E. Glasner, J. A. Gerlt, P. C. Babbitt, *Adv Enzymol Relat Areas Mol Biol* **75**, 193 (2007).
33. J. E. Coleman, *Annu Rev Biophys Biomol Struct* **21**, 441 (1992).
34. M. Y. Galperin, A. Bairoch, E. V. Koonin, *Protein Sci* **7**, 1829 (Aug, 1998).
35. J. G. Zalatan, T. D. Fenn, D. Herschlag, *J Mol Biol* **384**, 1174 (Dec 31, 2008).
36. T. D. Panosian *et al.*, *J Biol Chem* **286**, 8043 (Mar 11, 2011).
37. J. G. Zalatan, T. D. Fenn, A. T. Brunger, D. Herschlag, *Biochemistry* **45**, 9788 (Aug 15, 2006).
38. R. L. Felts, T. J. Reilly, J. J. Tanner, *J Biol Chem* **281**, 30289 (Oct 6, 2006).

39. S. C. Bihani *et al.*, *PLoS One* **6**, e22767 (2011).
40. G. Scardoni, M. Petterlini, C. Laudanna, *Bioinformatics* **25**, 2857 (Nov 1, 2009).
41. R. Breathnach, J. R. Knowles, *Biochemistry* **16**, 3054 (Jul 12, 1977).
42. C. Chothia, A. M. Lesk, *EMBO J* **5**, 823 (Apr, 1986).
43. C. Stefan, S. Jansen, M. Bollen, *Trends Biochem Sci* **30**, 542 (Oct, 2005).
44. B. E. Suzek, H. Huang, P. McGarvey, R. Mazumder, C. H. Wu, *Bioinformatics* **23**, 1282 (May 15, 2007).
45. J. F. Gibrat, T. Madej, S. H. Bryant, *Curr Opin Struct Biol* **6**, 377 (Jun, 1996).
46. E. C. Meng, E. F. Pettersen, G. S. Couch, C. C. Huang, T. E. Ferrin, *BMC Bioinformatics* **7**, 339 (2006).
47. E. F. Pettersen *et al.*, *J Comput Chem* **25**, 1605 (Oct, 2004).
48. R. Durbin, S. R. Eddy, A. Krogh, G. Michison, *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. (Cambridge University Press, 1998).
49. A. E. Barber, 2nd, P. C. Babbitt, *Bioinformatics*, (In press.).
50. K. Katoh, K. Kuma, H. Toh, T. Miyata, *Nucleic Acids Res* **33**, 511 (2005).
51. R. C. Edgar, *Nucleic Acids Res* **32**, 1792 (2004).
52. J. A. Nylander, J. C. Wilgenbusch, D. L. Warren, D. L. Swofford, *Bioinformatics* **24**, 581 (Feb 15, 2008).
53. Y. Fuchs, H. Steller, *Cell* **147**, 742 (Nov 11, 2011).
54. E. D. Crawford, J. A. Wells, *Annu Rev Biochem* **80**, 1055 (Jun 7, 2011).
55. S. Mahrus *et al.*, *Cell* **134**, 866 (Sep 5, 2008).
56. V. Neduva, R. B. Russell, *FEBS Lett* **579**, 3342 (Jun 13, 2005).
57. M. Kurokawa, S. Kornbluth, *Cell* **138**, 838 (Sep 4, 2009).
58. E. D. Crawford, J. E. Seaman, *et al.*, (In preparation).
59. G. Manning, D. B. Whyte, R. Martinez, T. Hunter, S. Sudarsanam, *Science* **298**, 1912 (Dec 6, 2002).
60. G. Pearson *et al.*, *Endocr Rev* **22**, 153 (Apr, 2001).
61. S. Powell *et al.*, *Nucleic Acids Res* **40**, D284 (Jan, 2012).
62. D. T. Barkan *et al.*, *Bioinformatics* **26**, 1714 (Jul 15, 2010).
63. E. D. Crawford *et al.*, *Cell Death Differ*, (Aug 24, 2012).
64. L. F. Olguin, S. E. Askew, A. C. O'Donoghue, F. Hollfelder, *J Am Chem Soc* **130**, 16547 (Dec 10, 2008).
65. B. van Loo *et al.*, *Proc Natl Acad Sci U S A* **107**, 2740 (Feb 16, 2010).
66. P. J. O'Brien, D. Herschlag, *Biochemistry* **40**, 5691 (May 15, 2001).
67. P. J. O'Brien, D. Herschlag, *J Am Chem Soc* **120**, 12369 (1998).
68. K. Yang, W. W. Metcalf, *Proc Natl Acad Sci U S A* **101**, 7919 (May 25, 2004).
69. J. K. Lassila, D. Herschlag, *Biochemistry* **47**, 12853 (Dec 2, 2008).
70. A. Kim *et al.*, *Biochemistry* **50**, 3481 (May 3, 2011).

Appendix A. Supplementary information for chapter 2

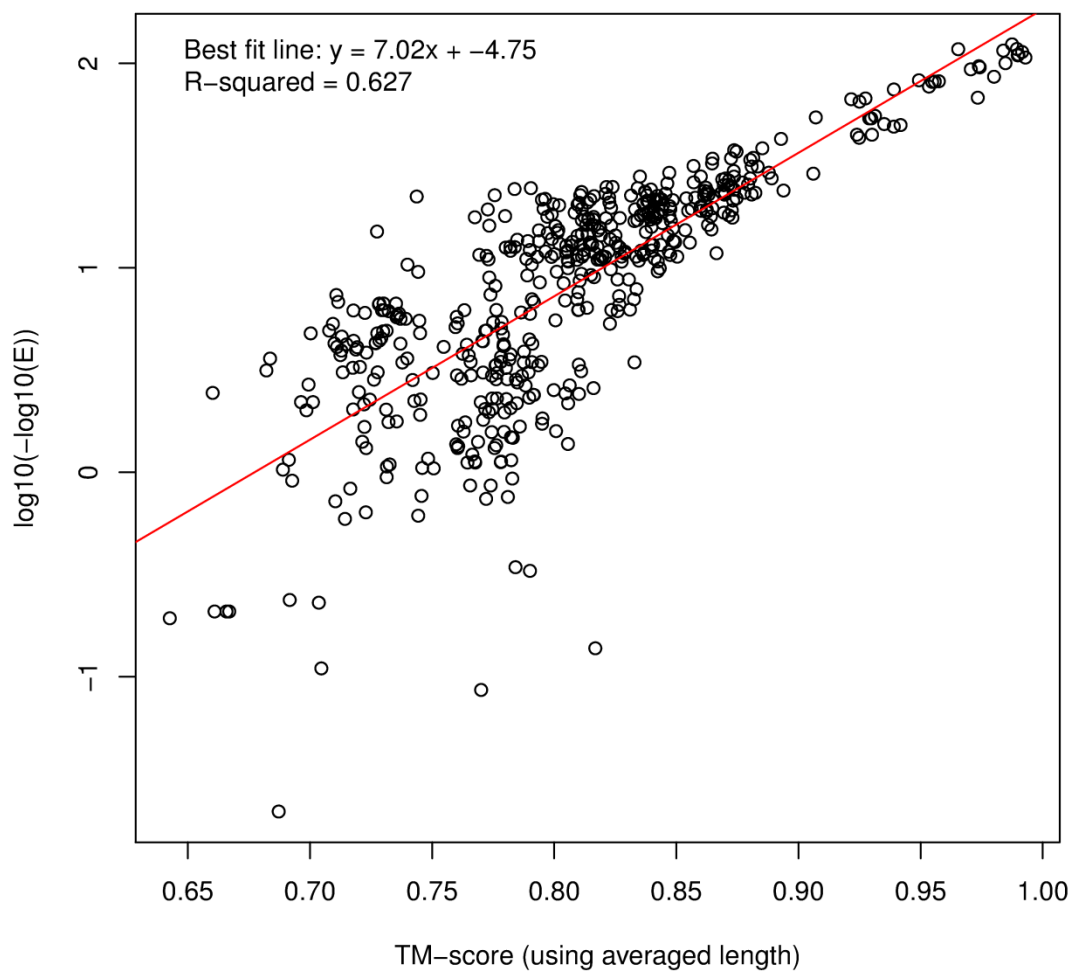


Figure A.1. Correlation plot between TM-score and $\log_{10}(-\log_{10}(E\text{-value}))$. Edges between sequence and their associated structure nodes that have both an E-value score < 1 and a TM-align score are plotted and show a roughly linear correlation. This figure is an example of the output from the output_correlation.py Pythoscape plug-in.

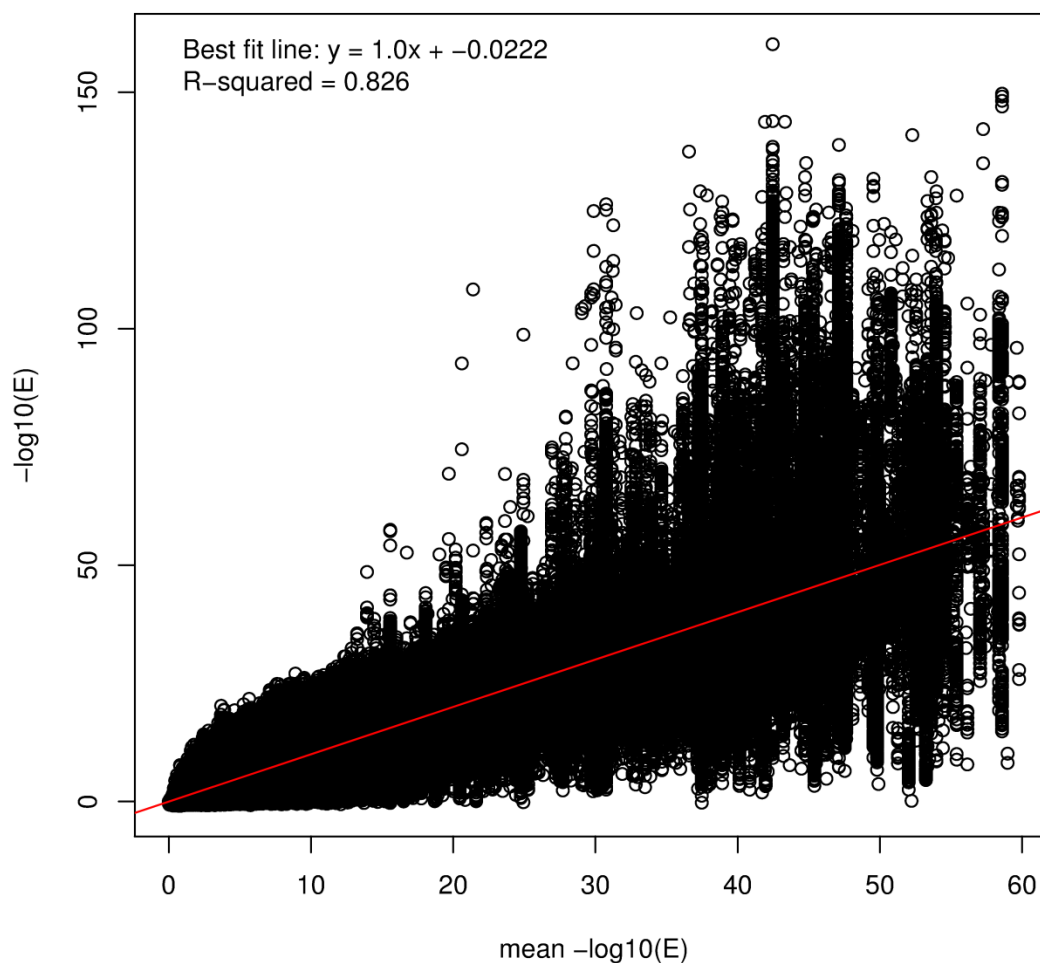


Figure A.2. Correlation between ideal representative $-\log_{10}(\text{Evalue})$ node distances and ideal full network $-\log_{10}(\text{Evalue})$ node distances for GST sequences. The plot of mean edge scores between representative nodes and their associated represented edge scores that both have an E-value < 1 and a mean $-\log_{10}(E) < 60$ shows a linear correlation. Outlier edges scores with mean $-\log_{10}(E) > 60$ were not shown as the scores are highly influenced by sequence length. Of 13,083,599 edge scores represented by the representative network used to create figure 1, 13,080,754 are plotted here. This figure was generated by the output_correlation.py Pythoscape plug-in.

Table A.1. Benchmarks for edge calculations. These calculations were performed using BI2Seq from BLAST+ v2.2.25 and parallel_blast2seq.py from the Pythoscape package on a system with 8 Intel(R) Xeon(R) X7560s at 2.27GHz (64 cores available) and 129,004 MB RAM. Each run as performed using a single edge input file for the purposes of the benchmark. Further speedup is possible with larger edge runs by splitting the run into several input files. In practice, we have routinely used Pythoscape to create networks of up to 50,000 nodes with roughly a billion edges using a large, shared cluster.

# nodes (protein sequences)	# edges (BI2Seq runs)	# threads used	time (seconds)	# edges/thread/second
10	45	2	4	5.625
10	45	4	2	5.625
10	45	8	1	5.625
100	4,950	2	323	5.625
100	4,950	4	153	7.662
100	4,950	8	77	8.088
1000	499,500	2	67072	3.724
1000	499,500	4	33525	3.725
1000	499,500	8	22184	2.815

Appendix B. Pythoscape documentation

B.1. Introduction

The material presented here is from release 1.01b of Pythoscape and downloaded from the website: <http://www.rbvi.ucsf.edu/trac/Pythoscape>. The full URL to access the web version of each section is provided. Code blocks are red text.

B.2. General documentation

B.2.1. Installation instructions

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/InstallationInstructions>

As mentioned on the front page, the following are the requirements to install Pythoscape:

== Required software ==

Python 2.6 or 2.7

(Pythoscape currently does not support Python 3.0+)

== Optional software ==

Biopython is required for any plug-in in the *_bio package plug-ins. Without Biopython installed, you cannot use these plug-ins. You can download Biopython from <http://biopython.org>.

MySQLdb v1.2.3 is required to use the MySQL based storage interface. You can download MySQLdb from <http://sourceforge.net/projects/mysql-python/>. (You must also have a MySQL database server installed and access to it.)

rpy v.1.0.3 is required to use the *_r package plug-ins. This provides access to plotting information from Pythoscape in R, as well as doing statistical tests on networks. You can download rpy from <http://rpy.sourceforge.net/index.html>. (You must also have R installed.)

Specific plug-ins may have other requirements. See the plug-in documentation for more information.

Currently, only source is available for OSes that have access to a Linux-like terminal. Other distributions will be released in the future. The setup.py script will check to see if you have Biopython, rpy and MySQLdb installed. It will not install packages if you lack the appropriate corresponding dependencies. You can

always re-install the program again after you install the appropriate dependencies to access them.

To install Pythoscape, un-tar the file:

```
tar -xvzf pythoscape-v1.01.tar.gz
```

Add Pythoscape to your Python installation:

```
python setup.py install
```

Depending on your system, you might need sudo privileges to run the previous command.

B.2.2. Framework overview

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/FrameworkOverview>

Pythoscape is a framework written in Python for assembling, curating and outputting large and often complex similarity networks. To do this, there are three main parts.

Plug-in <-> Environment <-> Storage interface

1. Environment: The environment defines the classes that allow for common handling between any storage interface and any plug-in. It can also constrain the plug-in and storage interface to only act on certain sets of network objects.

2. Storage interface: This interface extracts data from the environment and manipulates the underlying data.

3. Plug-in: These small classes specify data to be inputted, manipulated or outputted. Because the environment and storage interfaces are standardized, any plug-in should work on any interface.

=== Environments ===

1. PythoscapeEnvironment - This environment allows the plug-in or user to manipulate all stored data.

2. PythoscapeNetwork - This environment constrains Pythoscape to a previously defined set of data (a network).

=== Storage interfaces ===

There are currently two storage interfaces available. Others will be released as they are finished.

1. LocalInterface - This interface stores Pythoscape data on the local hard drive using Python shelves. It is not recommended to use this interface for large networks (>999 nodes) as it is more of a hack than a robust database infrastructure.

2. MySQLInterface - This interface stored Pythoscape data in a MySQL database. This interface is currently recommended for large, complex networks.

=== Plug-in ===

Many plug-ins are available. See the PluginTableOfContents page for more information.

== Objects in Pythoscape ==

Object name	Description	Required arguments (push)	Required arguments (pull)	Complex arguments	Optional arguments
Node	Basic unit of network - id must be unique	id		identifiers, networks	
Edge	Object that indicates the connection between two nodes	source_node, target_node		networks	filter, weight

Network	Nodes and edges can belong to networks	network_name			network_type
Attribute	Attributes are attached to nodes, edges or networks to describe their properties	element_obj, attribute_name, attribute_value	element_obj		
Identifier	Identifiers are attached to nodes to associate them with other databases	identifier_type, identifier_value	identifier_type, identifier_value		
Filter	A filter is attached to a querying edge to only fetch edges that match the filter	identifier_type, identifier_value	identifier_type, identifier_value		

Pythoscape requires that required parameters be inputted to create objects to either add them to the database (push) or query them from the database (pull). Complex parameters involve attaching other objects as described below in the tutorial. Optional parameters can be used at the user's preference.

== Scripting directly with the Pythoscape interface ==

For a tutorial on using objects to script with the Pythoscape interface, see [ScriptingWithObjects](#).

B.2.3. Plug-in usage

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/UsePlugins>

The following instructions will help you to properly configure a run file to execute a Pythoscape plug-in.

== 1. Create a run file. ==

Pythoscape can either be controlled from the Python interactive terminal or through run files. In general, it is recommended to control Pythoscape using run files as it is easiest and most efficient. To create a run file, make a directory where you plan to perform the analysis. Inside the directory, make a file called 'run.py' (don't include quotes).

```
#Import Pythoscape
```

```
from pythoscape.auxiliary.re_patterns import RE_PATTERNS
```

```
import pythoscape.main.environments as env
```

```
#Import plug-ins
```

```
#Create interface and environment
```

```
#Create plug-ins
```

```
#Execute plug-ins
```

Notice that there are five sections: `#Import Pythscape`, `#Import plug-ins`, `#Create interface and environment`, `#Create plug-ins` and `#Execute plug-ins`. We will fill in each of these sections to successfully execute a plug-in. To successfully execute the plug-in, you must fill in all the sections before ever running the script.

== 2. Choose an interface. ==

Pythscape stores the network through the use of an interface. Currently, there are two interfaces available, a local interface which stores the network in Python shelves in a specified directory and a MySQL interface which stores the network in a MySQL database. Other interfaces are being implemented to accommodate larger networks.

The local interface is ideal for initial testing of network interfaces and working with smaller databases. Because it uses local files and is quite slow, this interface is not appropriate for large networks that require optimized database software to index and manipulate them.

Specifically, small networks could be thought of networks with no more than 999 nodes where the complete network is no larger than ~500,000 edges. Even with these restrictions, if there are many (>3) edge attributes, the processes could move slowly enough that it is worth it to move to the MySQL interface.

When in doubt, use the MySQL interface!

Be aware that all network information gets saved in the location that you indicate. If you want to have separate networks in unique locations for organizational ease, you must have a separate database/file for each of them. If you prefer to have any set of created networks in a single database/file, you use the same interface parameters for those networks. If you choose to have several networks in a single database/file, be sure to use the PythoscapeNetwork environment (discussed in the next section) to organize those networks.

If you are going to use the local interface, import the interface module and create an interface, like such:

```
#Import Pythoscape
from pythoscape.auxiliary.re_patterns import RE_PATTERNS
import pythoscape.main.environments as env
import pythoscape.interface.local_interface as l_i

#Import plug-ins

#Create interface and environment
my_interface = l_i.LocalInterface(<save directory>)

#Create plug-ins

#Execute plug-ins
```

Replace <save directory> with a save path. This directory should exist, but nothing should be in it, unless you are loading a previously created network. As an example:

```
my_interface = l_i.local_interface('/home/myname/mynetwork')
```

If you are going to use the MySQL interface, import the interface module and create an interface, like such:

```
#Import Pythoscape

from pythoscape.auxiliary.re_patterns import RE_PATTERNS

import pythoscape.main.environments as env

import pythoscape.interface.mysql_interface as m_i

#Import plug-ins

#Create interface and environment

my_interface = m_i.MySQLInterface(<database_name>)

#Create plug-ins

#Execute plug-ins
```

To use the MySQL interface, you must have a MySQL database for your data already set up. If this has not been done, refer to CreatePythoscapeMysql to create this database before continuing. <database name> is the required argument for the MySQL

interface. Other database parameters must be entered as necessary including username, password, host and port. Host defaults to 'localhost' and port defaults to 3306 if these parameters aren't entered. As an example:

```
my_interface = \
m_i.MySQLInterface('mydatabase',user='myuser',passwd='mypass',host='localhost',port\
=3306)
```

For the rest of this page, the code blocks will assume that the local interface was used. Either interface is compatible with the code added in the later sections.

== 3. Choose an environment ==

The environment determines which set of nodes and edges a plug-in has access to. If you use the more general PythoscapeEnvironment, Pythoscape provides access to all nodes, edges and attributes. This is useful when you are adding node, edges or attributes for later network partitioning or when you are deciding which nodes or edges go in a specific network. Often you can do all the initial steps of network creation using the PythoscapeEnvironment and appropriate plug-ins. Afterwards, you can partition into specific networks once all data is collected. Using PythoscapeEnvironment at the beginning is not appropriate if you plan on having several distinct networks in a single database or file. In this case, you should specify which network you wish to work with using PythoscapeNetwork when running all plug-ins.

PythoscapeNetwork restricts access to a specific set of nodes, edges and attributes that is associated with a single network. This is useful when you only want a specific set of nodes, edges and attributes. For example, when you are outputting a network for analysis with another program or performing edge statistics on a given set.

PythoscapeNetwork is also necessary if you intend on having several distinct datasets in the same database file as mentioned above.

Finally, certain plug-ins only work with specific environments. Check the documentation of the plug-in that you are planning on using to make sure that it is compatible with the environment that you want to use with it.

If you are going to use PythoscapeEnvironment, change your run file to the following:

```
#Import Pythoscape
from pythoscape.auxiliary.re_patterns import RE_PATTERNS
import pythoscape.main.environments as env
import pythoscape.interface.local_interface as l_i

#Import plug-ins

#Create interface and environment
my_interface = l_i.LocalInterface(<save directory>)
my_pytho = env.PythoscapeEnvironment(my_interface)

#Create plug-ins
```



```
#Execute plug-ins
```

If you are going to use PythoscapeNetwork, change your run file to the following:

```
#Import Pythoscape
```

```
from pythoscape.auxiliary.re_patterns import RE_PATTERNS
```

```
import pythoscape.main.environments as env
```

```
import pythoscape.interface.local_interface as l_i
```

```
#Import plug-ins
```

```
#Create interface and environment
```

```
my_interface = l_i.LocalInterface(<save directory>)
```

```
my_net = env.PythoscapeNetwork(<network\
```

```
name>,my_interface,network_type=<network type>,create_network=True)
```

```
#Create plug-ins
```

```
#Execute plug-ins
```

The first two parameters, <network name> and my_interface are required. The other two are optional and can be left off.

Required:

Replace <network name> with the name of the network to create. "This name needs to be less than 13 characters long to work in the MySQL interface."

Optional:

Replace <network type> with the network type, if you are creating the network and want the type specified.

If this is the first time you use the network and it hasn't been created yet, `create_network=True` needs to be in the line. If the network already exists, `create_network=True` is optional.

As an example:

```
my_net =\
env.PythoscapeNetwork('my_network',my_interface,network_type='my_type',create_net\
work=True)
```

For the purposes of simplicity, we will use `PythoscapeEnvironment` for step 4. See step 5 for the complete code with both environments.

== 4. Choose plug-in. ==

Decide which plug-in to run. For instructions on using specific plug-ins or to locate plug-ins, use PluginTableOfContents and locate the documentation for the plug-in of interest. Import the plug-in module and create the plug-in:

```
#Import Pythoscape

from pythoscape.auxiliary.re_patterns import RE_PATTERNS

import pythoscape.main.environments as env
import pythoscape.interface.local_interface as l_i

#Import plug-ins

import <plugin path>

#Create interface and environment

my_interface = l_i.LocalInterface(<save directory>)
my_pytho = env.PythoscapeEnvironment(my_interface)

#Create plug-ins

plugin = <plugin module>.<plugin>()

#Execute plug-ins
```

Replace <plugin path>, <plugin module> and <plugin> with the information about the plug-in path and name. To give an example:

```
import pythoscape.plugin.input.test_plugin as t_p
```

and

```
plugin = t_p.test_plugin()
```

== 5. Execute plug-in. ==

Execute the plug-in. Until this step, no manipulations are done on the network. You can execute as many plug-ins in a row in the code as is wanted. In this example and in all other examples in the tutorial, only one plug-in is executed. If you trying something for the first time, it is recommended to only execute one plug-in at a time. If something goes wrong, it is easier to isolate the problem.

If you are using PythoscapeEnvironment, the following code is used:

```
#Import Pythoscape
from pythoscape.auxiliary.re_patterns import RE_PATTERNS
import pythoscape.main.environments as env
import pythoscape.interface.local_interface as l_i

#Import plug-ins
import <plugin path>

#Create interface and environment
my_interface = l_i.LocalInterface(<save directory>)
```

```
my_pytho = env.PythoscapeEnvironment(my_interface)
```

```
#Create plug-ins
```

```
plugin = <plugin module>.<plugin>()
```

```
#Execute plug-ins
```

```
my_pytho.execute_plugin(plugin)
```

If you are using PythoscapeNetwork, the following code is used:

```
#Import Pythoscape
```

```
from pythoscape.auxiliary.re_patterns import RE_PATTERNS
```

```
import pythoscape.main.environments as env
```

```
import pythoscape.interface.local_interface as l_i
```

```
#Import plug-ins
```

```
import <plugin path>
```

```
#Create interface and environment
```

```
my_interface = l_i.LocalInterface(<save directory>)
```

```
my_net = env.PythoscapeNetwork(<network\  
name>,my_interface,network_type=<network type>,create_network=True)
```

```
#Create plug-ins
```

```
plugin = <plugin module>.<plugin>()
```

```
#Execute plug-ins  
my_net.execute_plugin(plugin)
```

Notice how in either case, the method `execute_plugin` is used to run a plug-in.

The preceding code blocks, both for `PythoscapeEnvironment` and `PythoscapeNetwork` are complete. All run files must have all the preceding elements in them in order to function correctly. You can now run them with the `python` command. For example, if you saved the complete run file from above as `'run.py'` (no quotes), you would execute the step in Pythoscape by executing the run file in the command line.

```
python run.py
```

B.2.4. Network creation pipeline

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/PipelineAndPlugins>

The pipeline is a series of steps and corresponding plug-ins that can be used to create, modify and output a network.

On this page, we will use the concept of run files as described in `UsePlugins`. We will also use query files as explained in `ScriptingWithObjects`. (If you haven't reviewed both these sections, it would be a good idea before continuing.) In each step, there is a list of plug-ins that can be used to accomplish this step. To find

the instructions on using each plug-in, find the plug-in documentation in `PluginTableOfContents`.

Plug-ins currently released, documented and ready are listed in each section.

Future plug-ins will be included on this page as they are released.

== Step 1: Add nodes (sequences, structures, etc.) ==

Use the following plug-ins to add appropriate nodes into the network:

Add: sequences from a fasta file as nodes

Use: `ImportFromFastaFile`

Location: `pythoscape.plugin.input.import_sequences`

Documentation: `InputPlugins#ImportFromFastaFile`

Add: import PDB files from local PDB database

Use: `PDBImportFromCSVFile`

Location: `pythoscape.plugin.input_bio.import_structures`

Documentation: `InputPlugins#PDBImportFromCSVFile`

== Step 2: Calculate complete network ==

Calculate all between-node connections using one of the following plug-ins.

If you are calculating edges for a particularly large sequence network, it is recommended to use the `OutputPlugins#SequenceTableRun` plug-in and calculate the edges on an external cluster or server in a parallel fashion and then input the information using an edge input plug-in like `InputPlugins#AddBLASTEEdgesFromTableRun`. The speed of calculating edges using plug-ins depends both on the method that the plug-in uses and the implementation of that method execution in the plug-in. This means that it could take days or even weeks for a slow plug-in to calculate a complete large network. See the documentation for the specific plug-in for more information.

Add: local BLAST run

Use: `AddBLASTEEdgesFromLocalBLAST`

Location: `pythoscape.plugin.input.add_local_blast`

Documentation: `InputPlugins#AddBLASTEEdgesFromLocalBLAST`

Add: edges from BLAST table run

Use: `AddBLASTEEdgesFromTableRun`

Location: `pythoscape.plugin.input.add_blast_files`

Documentation: `InputPlugins#AddBLASTEEdgesFromTableRun`

Add: calculated structure similarity edges using TM-Align

Use: `CalculateTMAlignScores`

Location: `pythoscape.plugin.input.calculate_struc_edges`

Documentation: `InputPlugins#CalculateTMAlignScores`

Do: output set up files for a remote all-by-all sequence edge run

Use: `SequenceTableRun`

Location: `pythoscape.plugin.output.output_table_runs`

Documentation: `OutputPlugins#SequenceTableRun`

Do: output set up files for a remote all-by-all structure edge run

Use: `StructureTableRun`

Location: `pythoscape.plugin.output.output_table_runs`

Documentation: `InputPlugins#StructureTableRun`

== Step 3: Generate edge statistics ==

To know how many edges you are dealing with and appropriate steps forward, the first necessary piece of data is an edge count. To calculate an edge count, make a query file (`query.py`) and paste the following commands:

(Be sure to replace `<interface>`, `<create interface>`, `<environment>`, and `<create environment>` below to the appropriate values. See the `UsePlugins` documentation for information on setting these lines appropriately)

```

import main.environments as env

import <interface>

#Create interface
<create interface>

#Make edge query
edge_query = <interface name>.Edge()

# If you have been using PythoscapeNetwork, then you need the following two
# lines. Uncomment them
# and fill in the appropriate values. If you have been using
# PythoscapeEnvironment, leave them commented out.

# net = <interface name>.Network(network_name=<network name>)
# edge_query.add_networks([net])

#Count edges
print len([True for edge in <interface name>.pull(edge_query)])

```

To run the query script:

python query.py

This will print how many edges are in the network. Default [http://www.cytoscape.org Cytoscape] 32-bit configurations, the preferred network visualization program, cannot handle networks with an edge count too much above 200,000. In general, it is reasonable to consider 200,000 a good upper limit for networks to output on a typical 32-bit desktop installation. If you are using 64-bit [http://www.cytoscape.org Cytoscape], the edge limit is much higher and depends on the system setup and the amount of RAM. As a general rule, we shoot for an upper limit of 1,000,000 edges on a 64-bit system. Networks with edge counts about 1,000,000 can be opened, but manipulation becomes difficult.

== Step 4: Add node attributes ==

There are several plug-ins that add node attributes depending on what information needs to be included for visualization.

Add: Annotations from UniProt

Use: ImportFromUniProt

Location: pythoscape.plugin.input_bio.add_uniprot_info

Documentation: InputPlugins#ImportFromUniProt

Add: Annotations from table

Use: ImportAttributeTable

Location: pythoscape.plugin.input.add_attribute_table

Documentation: InputPlugins#ImportAttributeTable

== Step 5: Filter edges (Optional) ==

This step is only necessary if the number of edges is greater than what you would like to output.

First, to check if your chosen edge threshold restricts the number of edges to an amount that you would like to output, make a new query.

(Be sure to replace <interface>, <create interface>, <environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import main.environments as env
```

```
import <interface>
```

```
#Create interface
```

```
<create interface>
```

```

#Make edge filter

edge_filt = <interface name>.Filter(filter_name=<edge attribute>,direction = <'>' \
or '<'>',filter_value=<filter value>)

#Make edge query

edge_query = <interface name>.Edge(filter=edge_filt)

# If you have been using PythoscapeNetwork, then you need the following two
# lines. Uncomment them

# and fill in the appropriate values. If you have been using
# PythoscapeEnvironment, leave them commented out.

# net = <interface name>.Network(network_name=<network name>)

# edge_query.add_networks([net])

#Count edges

print len([True for edge in <interface name>.pull(edge_query)])

```

Filter required variables:

<edge attribute> = edge attribute that you want to filter on, an example would be '-log10(E)'

<'>' or '<'> = pick '>' or '<' depending on if you want edges with the attribute greater or less than the filter value

<filter value> = the value of the filter

The query file is constructed and run above, except now we include a filter. As an example, the line:

```
edge_filter = <interface name>.Filter(filter_name='-log10(E)',direction =\
'>',filter_value=5)
```

would output all edges where the value of attribute $-\log_{10}(E)$ is greater than 5.

To run the query script:

```
python query.py
```

== Step 6: Make a representative node network (Optional) ==

If there are still too many edges and you would like to visualize protein similarity relationships at relaxed thresholds, you can create representative nodes. The first step is to create a new network of filtered nodes that will stand in the set of

entire nodes. The follow plug-ins can do this depending on the network you are working with.

Add: Representative nodes that stand in for all nodes in a sequence network within a provided sequence identity cutoff. Calculation is done with CD-HIT.

Use: CreateCDHITRepnodes

Location: `pythoscape.plugin.input.make_cdhit_repnodes`

Documentation: `InputPlugins#CreateCDHITRepnodes`

Next, we will determine representative edges for the new representative node network. The plug-in is as follows:

Add: calculated representative edge set

Use: AddEdgesToRepnodeNetwork

Location: `pythoscape.plugin.input.add_repnode_edges`

Documentation: `InputPlugins#AddEdgesToRepnodeNetwork`

== Step 7: Add representative node attributes (Optional) ==

Now that you have selected representative nodes, you need to create node attributes that reflect the properties of the nodes that the representative nodes represent.

Add: represented attributes concatenated to a list

Use: `AddAttributesByIfAny`

Location: `pythoscape.plugin.input.add_renode_atts`

Documentation: `InputPlugins#AddAttributesByIfAny`

Add: represented attributes cropped to the most represented one

Use: `AddAttributesByGreatestSum`

Location: `pythoscape.plugin.input.add_renode_atts`

Documentation: `InputPlugins#AddAttributesByGreatestSum`

Add: number of nodes represented by a representative node

Use: `CalcNodeSize`

Location: `pythoscape.plugin.input.renode_stats`

Documentation: `InputPlugins#CalcNodeSize`

== Step 8: Output Network ==

Finally, output the network to be visualized or analyzed. Remember that if you have made a representative node network, you should output the representative node network using `PythoscapeNetwork`, instead of all the nodes and edges using `PythoscapeEnvironment`.

Do: output the network as an XGMML that can be opened in Cytoscape

Use: OutputXGMML

Location: pythoscape.plugin.output.output_xgmml

Documentation: OutputPlugins#OutputXGMML

B.2.5. Writing plug-ins

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/WritingPlugins>

The class to use to create plug-ins is called PythoscapePlugin. You can inherit and configure this class to write your own plug-ins. Two sample plug-ins are below. This example code can guide the creation of new plug-ins. To understand the basics of writing a plug-in, review the ScriptingWithObjects tutorial. It is also helpful to understand the concept of the Python generator:

<http://docs.python.org/tutorial/classes.html#generators>.

There is also a function included in this class called hasher. This function outputs the hexdigest of a SHA-224 hash. This can be used by plug-ins as a standard function to create node ids from a specific attribute. To import the hasher, use the following import:

```
from pythoscape.main.plugin import hasher
```

```
== Example plug-in 1 (!ExamplePlugin): ==
```

This plug-in will demonstrate the basics of setting up a plug-in. For the great majority of plug-ins, this template should suffice.

```
from pythoscape.main.plugin import PythoscapePlugin

class ExamplePlugin(PythoscapePlugin):

    def init(self):

        #-- Plugin required code--

        #Don't change this line of code.
        PythoscapePlugin.init(self,locals())

        #Is your plugin compatible with the base Pythoscape environment?
        # (True/False)
        #This is generally true.
        self.Pythoscape = True

        #Is your plugin compatible with the PythoscapeNetwork environment?
        # (True/False)
        #This is often true, except in cases where the plugin needs to be able to
        #work with multiple sub-networks.
        self.PythoscapeNetwork = True

        #Set a description of your plugin for logging.
```

```

self.description = 'Plugin description here'

#Plugin specific

#Add your own init variables and functions here!


#Setup the functions. Any function that you want run on the data needs to
# be setup.

#The plugins are run in the order that they are set up.

#The first variable for self.setup is either 'pull','push' or 'delete' depending on
#what
#will be yielded by the function.

#You can rename the function, the second variable, to whatever you want.

#The names below are
#just suggestions.

self.setup('pull',self.get_data)

self.setup('push',self.add_data)

self.setup('delete',self.del_data)


def get_data(self):

    #This is an example 'pull' function.

    #The first step is to yield a selection network object. If you are uncertain
    #about what a selection
    #object looks like, see the documentation for graph_interface.py or the

```

```
#ScriptingWithObjects tutorial.
```

```
#For example, the empty node below will return all nodes in the data
```

```
#source.
```

```
yield self.Node()
```

```
#The selected objects are yielded in self.outgoing.
```

```
for node in self.outgoing:
```

```
    pass
```

```
def add_data(self):
```

```
    #This is an example 'push' function.
```

```
    #Every object yielded in this function will be added into the data source.
```

```
    yield self.Node(id='example')
```

```
def del_data(self):
```

```
    #This is an example 'delete' function.
```

```
    #To use the delete function, yield selection network objects. If you are
```

```
    # uncertain about what a selection
```

```
    #object looks like, see the documentation for graph_interface.py or the
```

```
    #ScriptingWithObjects tutorial.
```

```
    #For example, the empty node below will delete all nodes in the data
```

```
    #source.
```

```
    yield self.Node()
```

== Example plug-in 2 (ExampleEdgePlugin): ==

This plug-in will demonstrate the use of batch functions. Batch functions can be used to run blocks of functions with a single variable. This is useful when you want to run several functions at a time and not have to store data in RAM. For example, if you want to modify all the edges, but are afraid that there wouldn't be enough RAM to hold all the edge information at one time.

```
class ExampleEdgePlugin(PythoscapePlugin):

    def init(self):

        #-- Plugin required code--

        #Don't change this line of code.
        PythoscapePlugin.init(self,locals())

        #Is your plugin compatible with the base Pythoscape environment?
        #(True/False)

        #This is generally true.
        self.Pythoscape = True

        #Is your plugin compatible with the PythoscapeNetwork environment?
        #(True/False)

        #This is often true, except in cases where the plugin needs to be able to
        #work with multiple sub-networks.
```

```

self.PythoscapeNetwork = True

#Set a description of your plugin for logging.

self.description = 'Modify edges'


#Plugin specific

#Add your own init variables and functions here!

self._nodes = []


#Setup the functions. Any function that you want run on the data needs to
#be setup.

#The plugins are run in the order that they are set up.

#The first variable for self.setup is either 'pull','push' or 'delete' depending on
#what
#will be yielded by the function.

#You can rename the function, the second variable, to whatever you want.

#The names below are
#just suggestions.

self.setup('pull',self.get_nodes)


#Setup batch functions. The first argument is a list of tuples. The tuples are
#the same variables that would be used for a regular setup. In the case of a
#batch
#run, these functions will require an input. Pythoscape will run them in the

```

```

#order

#that you specify in the list. The second argument is a string

#that is the name of the class attribute that contains the batch of objects

#that will be passed into each function in the order provided.

self.batch_setup([('pull',self.get_edges),('push',self.yield_edges)],'_nodes')


def get_nodes(self):

    #Get all nodes names and store them.

    yield self.Node()

    for node in self.outgoing:

        self._nodes.append(node)


def get_edges(self,node):

    #Get all edges that are from a single node.

    yield self.Edge(source_node=node)

    self._curr_edges = []

    for edge in self.outgoing:

        self._curr_edges.append(edge)


def yield_edges(self,node):

    #Add edge attribute

    for edge in self._curr_edges:

```

```
yield\  
self.Attribute(element_obj=edge,attribute_value=True,attribute_name='modify')
```

B.2.6. Setting up a MySQL interface

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/CreatePythoscapeMysql>

== Step 1) Make database ==

Make sure that the MySQL server and client is installed on the machine where you want to store the information. Because Pythoscape uses stored procedures and views, you will need MySQL 5.0 or higher.

If you are a system administrator:

Create a database for yourself to use. Log into MySQL using the client and the appropriate parameters and create a database.

```
CREATE DATABASE <database_name>;
```

Replace <database_name> with the name of the database you want to make.

If you are not a system administrator:

Ask your system administrator to create a database for you as well as provide you a username and password. Also ask for permission to use stored procedures. These are often disabled by default.

== Step 2) Format database ==

Locate the `mysql_template.sql` file in the Pythoscape package. It should be in the `mysql_resources` directory. Format the database to work with Pythoscape by running the following command:

```
mysql <required parameters> < /path/to/mysql_template.sql
```

Replace `<required parameters>` with the appropriate log-in information and `/path/to/mysql_template.sql` with the appropriate path

Example:

```
mysql -u myusername -p mydatabase <\  
/home/myhome/pythoscape/mysql_resources/mysql_template.sql
```

B.2.7. Pre-defined regular expressions

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/IncludedRegularExpressions>

Pythoscape uses regular expressions to find identifiers in titles and descriptions. Because regular expressions are not intuitive, several have been included in Pythoscape to simplify the pipeline. To use one of these patterns, make sure you have the following line somewhere in the first section of your run file:

```
from pythoscape.auxiliary.re_patterns import RE_PATTERNS
```

RE_PATTERNS is a Python dictionary that defines these strings. To use this dictionary, instead of inputting the regular expression, use the convention RE_PATTERNS[<pattern name>] where <pattern name> is the name of the pattern you would like to use. A table of included pattern names is below.

Pattern name: 'SFLD_GI'

Pattern description: This pattern locates GIs from files from the SFLD. It should also work with general GIs, but this is untested.

Pattern name: 'SFLD_EFD'

Pattern description: This pattern locates EFD numbers outputted from the SFLD.

Pattern name: 'SFLD UniProtEntry'

Pattern description: This pattern locates UniProt entry names in an SFLD-like title.

Pattern name: 'SFLD_Enzyme'

Pattern description: This pattern is used for GI like titles using the prefix 'lcl|' and suffix '|'.

Pattern name: 'Uniref100 with prefix'

Pattern description: This pattern pulls out the whole UniRef ID including the 'UniRef100_' tag at the beginning.

Pattern name: 'Uniref100 without prefix'

Pattern description: This pattern pulls out the UniRef ID but leaves off the 'UniRef100_' tag.

Pattern name: 'UniprotKB'

Pattern description: This pattern pulls out a UniProt KB ID - the 6 number letter one.

Pattern name: 'Uniprot/Pfam full length'

Pattern description: This pattern pulls out a UniProt KB AC from a full length sequence download from Pfam.

Pattern name: 'Uniprot/Pfam alignment'

Pattern description: This pattern pulls out a UniProt KB ID from a Pfam alignment sequence header.

Pattern name: 'PDB/NCBI'

Pattern description: This pattern pulls out a PDB ID from an nr sequence title.

B.2.8. Parallel resources

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/ParallelResources>

Scripts for doing massively parallel edge runs are included in the parallel_resources directory of the Pythoscape downloadable package. See below for instructions on their usage.

== parallel_blast2seq.py ==

Description:

The file parallel_blast2seq.py is used to calculate these edges and requires that the BLAST+ v.2.2.25 or above be installed on the machine to use it. You can download this package from:

http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download. You must have at least Python 2.6 installed to use this program.

Benchmarks done on the performance of this code have shown that on a machine with an Intel Xeon 2.27 GHz processor, approximately 360 edges can be calculated per minute per core used. This translates to around 500,000 edges per day per core used. Keep this in mind when deciding how many threads to use when calculating what size of network. For rapid calculation of large networks (> 10 million edges), it is a good idea to have at least a hundred processor cores available.

There are two ways to run this script; the first is on a machine where you have access to many cores in the shell. The second is to use a queuing system. If you indicate that you only want to calculate on 1 processor, then the script can be used effectively with HPC management systems like Sun Grid Engine with an appropriate submission script and one table file per processor. The contents of that submission script will depend on the specific setup of your cluster, so contact your system administrator for assistance.

Do not have any files in the run directory that have the table file extension other than the table files or the script will run into problems.

Execution description:

```
python parallel_blast2seq.py <# processors> <seq_file> <table_file_ext>  
<output_file_ext> <Optional: --gap_open value> <Optional: --gap_extend value>
```

<Optional: --comp_based True/False> <Optional: --use_best True/False>

<Optional: --filter -log10(E)_filter> <Optional: --blastp_exe blastp_exe>

Required arguments:

<# processors> - number of processors to use to calculate edges

<seq_file> - file of fasta formatted sequences outputted from Pythoscape, called sequences.fa by default

<table_file_ext> - extension of table files, all files with this extension in the directory will be assumed to be table files for the run

<output_file_ext> - extension of output files to create, 'out' (no quotes) is a good idea.

Optional arguments:

--gap_open: value for gap open penalty, 11 by default

--gap_extend: value for gap extend penalty, 1 by default

--comp_based: composition based statistics, True by default

--use_best: use the best edge or worst edge of the pair calculated between two nodes (they are asymmetric if composition based statistics is on), False by default

--filter: only write out edges that have a $-\log_{10}(E)$ value greater than the value given, default None

--blastp_exe: the location of the blastp executable, the script assumes that blastp is in the directory. ./blastp by default

Example execution:

```
python parallel_blast2seq.py 6 sequences.fa tab out
```

This would run edges on 6 cores using the sequence file sequences.fa, all files named .tab in the directory and output corresponding .out files.

B.2.9. Delete plug-ins

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/DeletePlugins>

== DeleteEdgesFromFile ==

Description:

This plug-in deletes all edges defined in an input file. By default, the plug-in expects the file lines to look as follows:

```
source_node target_node
```

There is an additional option to filter edge ids that are copied from the Cytoscape edge table. If this is the case, the file lines look like this:

```
source_node (pp) target_node
```

Environments:

This plug-in will work with both PythoscapeEnvironment and PythoscapeNetwork. Using PythoscapeNetwork only deletes edges from a given network and does not delete them from the general environment.

Example execution:

(Be sure to replace <interface>, <create interface>, <environment>, and <create environment> below to the appropriate values. See the UsePlugins for information on setting these lines appropriately)


```

import main.environments as env

import <interface>

#Import plug-ins

import plugin.delete.delete_edges as d_e

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin = d_e.DeleteEdgesFromFile(<file_name>,pp_edges=<True or False>)

#Execute plug-ins

<environment>.execute_plugin(plugin)

```

Required arguments:

<file_name> = name of file that contains the edges to be deleted

Optional arguments:

<pp_edges> = set to True if edge is of form: source (pp) target

Plug-in example:

```
plugin = d_e.DeleteEdgesFromFile('my_edges.txt')
```

This example would delete the edges defined by my_edges.txt.

B.2.10. Input plug-ins

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/InputPlugins>

== AddAttributesByGreatestSum ==

Description:

This plug-in adds representative node attributes based on 'greatest sum' voting of the represented nodes' attributes. This means that it takes all the possibilities of the attribute in the group of represented nodes and adds the one with the most occurrences. Currently the plug-in adds whichever one comes first if there is a tie. Future versions of the plug-in will solve that.

Environments:

This plug-in will only work with PythoscapeEnvironment. If you want to have this plug-in operate on a sub-network, use the initial_net argument.

Example execution:

(Be sure to replace <interface>, <create interface>, <environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env

import <interface>

#Import plug-ins

import pythoscape.plugin.input.add_renode_atts as a_r_a

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin =

a_r_a.AddAttributesByGreatestSum(<network>,<renode_att>,<calc_atts>,\
<new_att_name>,initial_net=<initial_net>)
```

#Execute plug-ins

<environment>.execute_plugin(plugin)

Required arguments:

<network> - name of representative node network

<reptime_att> - name of attribute that indexes representative nodes

<calc_atts> - list of attributes from which you will find the greatest common value

-- if you just want this done on one attribute, make a list of one

<new_att_name> - name of created representative node attribute

Optional arguments:

<initial_net> - default None, if you set this to the name of a network, it will only consider nodes from that network

Because you can't run this plug-in using NetworkEnvironment, <initial_net> is used to constrain the plug-in to a specific sub-network.

Plug-in example:

```
plugin = a_r_a.AddAttributesByIfAny('my_rep_network','my_repnod_index',\
'chosen_attribute','representative chosen_attribute')
```

This plug-in would calculate representative attributes for my_rep_network using my_repnod_index as the node index. The attribute 'representative chosen_attribute' would be created by doing 'greatest sum' voting on the attribute 'chosen_attribute'.

== AddAttributesByIfAny ==

Description:

This plug-in adds representative node attributes based on 'if-any' voting of the represented nodes' attributes. This means that it takes all the possibilities of the attribute in the group of represented nodes and concatenates that list together for the representative node attribute.

Environments:

This plug-in will only work with PythoscapeEnvironment. If you want to have this plug-in operate on a sub-network, use the initial_net argument.

Example execution:

(Be sure to replace <interface>,<create interface>,<environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env

import <interface>

#Import plug-ins

import pythoscape.plugin.input.add_renode_atts as a_r_a

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin = a_r_a.AddAttributesByIfAny(<network>,<renode_att>,<calc_att>,\
<new_att_name>,<initial_net>,<is_iden>=<True or\
False>,<concat>=<concat>)
```

#Execute plug-ins

<environment>.execute_plugin(plugin)

Required arguments:

<network> - name of representative node network

<reptime_att> - name of attribute that indexes representative nodes

<calc_att> - name of attribute to make the representative attribute from

<new_att_name> - name of created representative node attribute

Optional arguments:

<initial_net> - default None, if you set this to the name of a network, it will only consider nodes from that network

Because you can't run this plug-in using NetworkEnvironment, <initial_net> is used to constrain the plug-in to a specific sub-network.

<is_ident> - default False, set this to true if want to do this calculation on an identifier and calc_att is the name of an identifier

<concat> - default ',', set this to another value if you want the plug-in to concatenate options using something other than ','

Plug-in example:

```
plugin = a_r_a.AddAttributesByIfAny('my_rep_network','my_repnod_index',\
'chosen_attribute','representative chosen_attribute')
```

This plug-in would calculate representative attributes for my_rep_network using my_repnod_index as the node index. The attribute 'representative chosen_attribute' would be created by doing 'if-any' voting on the attribute 'chosen_attribute'.

== AddBLASTEEdgesFromLocalBLAST ==

Description:

This plug-in uses a local BLAST run to add BLAST edges to Pythoscape.

Warning: This plug-in currently does not run BLAST in parallel. That means that it could take a very long time to calculate a large network.

Environments:

This plug-in will work with both PythoscapeEnvironment and PythoscapeNetwork. Using PythoscapeNetwork adds the inputted edges into a specific network.

Plug-in software requirements:

This plug-in was designed to work with BLAST+ v.2.2.25 and above. You can download this package from:

http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download

Example execution:

(Be sure to replace <interface>, <create interface>, <environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```

import pythoscape.main.environments as env

import <interface>

#Import plug-ins

import pythoscape.plugin.input.add_local_blast as a_l_b

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin = a_l_b.AddBLASTEEdgesFromLocalBLAST(<blast_exe>,<include_atts>,

                                             filter_att=<filter_att>,

                                             filter_dir=<'>' or '<'>,

                                             filter_value=<filter_value>,

                                             sequence_att=<sequence_att>)

#Execute plug-ins

<environment>.execute_plugin(plugin)

```

Required arguments:

<blast_exe> - location of the blast_p executable

<include_atts> - list of edge attributes to include

The include_atts argument requires a list of any of the following: 'bit_score', 'm*n', 'alignment_len', 'alignment_identities', 'query_start', 'query_end', 'subject_start', 'subject_end', '-ln(E)', '-log10(E)', '% id'. The attributes included in this list will be added to the network while the others will be skipped.

Optional arguments:

<filter_att> - default None, edge attribute to filter on

<filter_dir> - default None, either '>' or '<'

<filter_value> - default None, value to test filter against

<sequence_att> - default 'sequence', attribute name of protein sequences

Plug-in example:

```
plugin = a_b_f.AddBLASTEdgesFromLocalBLAST('/home/myuser/blast/blastp',\
include_atts=['-log10(E)'])
```

This example would look for the blast executable at /home/myuser/blast/blastp and only adds the -log10(E) score to the database.

== AddBLASTEEdgesFromTableRun ==

Description:

This plug-in uses a table-like file to add BLAST edges to Pythoscape. The table file is expected to have the following columns as shown below from a BLAST sequence similarity comparison. See ParallelResources for more information on executing cluster runs. || indicates a new column.

|| query_node_id || subject_node_id || bit_score || m*n || alignment_len ||
alignment_identities || query_start || query_end || subject_start || subject_end

Environments:

This plug-in will work with both PythoscapeEnvironment and PythoscapeNetwork. Using PythoscapeNetwork adds the inputted edges into a specific network.

Example execution:

(Be sure to replace <interface>,<create interface>,<environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env

import <interface>

#Import plug-ins

import pythoscape.plugin.input.add_blast_files as a_b_f

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin =\

a_b_f.AddBLASTEdgesFromTableRun(<directory>,<include_atts>,file_ext=\

<file_ext>,filter_att=<filter_att>,filter_dir=<'>' or\

'<'>,filter_value=<filter_value>,restart=<True or False>)

#Execute plug-ins
```

`<environment>.execute_plugin(plugin)`

Required arguments:

`<directory>` - directory location on disk to find all results files

`<include_atts>` - list of edge attributes to include

The `include_atts` argument requires a list of any of the following: 'bit_score', 'm*n', 'alignment_len', 'alignment_identities', 'query_start', 'query_end', 'subject_start', 'subject_end', '-ln(E)', '-log10(E)', '% id'. The attributes included in this list will be added to the network while the others will be skipped.

Optional arguments:

`<file_ext>` - default 'out', if the files are not named with the extension '.out', then this needs to be set to the appropriate extension

If you want to filter incoming edges to speed up the process by not importing and indexing unnecessary edges, use the filter options. All three must be set for the filter to work.

`<filter_att>` - default None, edge attribute to filter on

<filter_dir> - default None, either '>' or '<'

<filter_value> - default None, value to test filter against

If the run crashed for any reason and you want to restart it in the middle, use the restart argument. This option does not work when you are using the above filter. You cannot restart currently if you are using the filtering option.

<restart> - default False, True if you want to restart the edge import run

Plug-in example:

```
plugin = a_b_f.AddBLASTEdgesFromTableRun('/home/myuser/mydata/tables',\
include_atts=['-log10(E)'])
```

This example would look for results files in the directory /home/myuser/mydata/tables and only adds the $-\log_{10}(E)$ score to the database. This saves time until the edges can be filtered. Other edge attributes can be added later.

== AddEdgesToReprnodeNetwork ==

Description:

This plug-in adds edges to a representative node network. It summarizes the information of all the edges that connect all nodes between two representative nodes. It calculates three attributes: 1) how many edges are being represented, 2) the mean value of the attribute and 3) the minimum or maximum value of the attribute. The edge attributes added have the form '<network> <attribute>' for their name where <network> is the name of the representative network name and <attribute> is the specific attribute added. For example, if you did this analysis on the representative node network called 'my_rep_net', you would get three edge attributes 'my_rep_net mean', 'my_rep_net min/max' and 'my_rep_net total'. Right now, these names can't be customized.

Environments:

This plug-in will only work with PythoscapeEnvironment. If you want to have this plug-in operate on a sub-network, use the initial_net argument.

Example execution:

(Be sure to replace <interface>,<create interface>,<environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env
```

```
import <interface>
```

```
#Import plug-ins
```

```
import pythoscape.plugin.input.add_repnod_edges as a_r_e
```

```
#Create interface and environment
```

```
<create interface>
```

```
<create environment>
```

```
#Create plug-ins
```

```
plugin = a_r_e.AddEdgesToRepnodNetwork(self,<network>,<repnod_att>,\n<edge_att_name>,<minum>=<True or False>,<filt_dir>=<'<', '<=', '>' or '>='>,<filt_value>=<filt_value>,<initial_net>=<initial_net>,<pre_filt>=<pre_filt>)
```

```
#Execute plug-ins
```

```
<environment>.execute_plugin(plugin)
```

Required arguments:

<network> - name of representative node network

<reinode_att> - name of attribute that indexes representative nodes

<edge_att_name> - name of edge attribute to base representative edge on

Optional arguments:

<minum> - default False, this argument is used to indicate whether this plug-in calculate a minimum or maximum value for edge attribute. False indicates to calculate a maximum value, True indicates a minimum value

If you want to filter what edges are included in this calculation, set filter_dir and filter_value. As a default, it will use edge_att_name as filter_att.

<filter_dir> - default None, either '>', '>=', '<=' or '<'

<filter_value> - default None, value to test filter against

<initial_net> - default None, if you set this to the name of a network, it will only consider nodes from that network

Because you can't run this plug-in using !NetworkEnvironment, <initial_net> is used to constrain the plug-in to a specific sub-network.

<pre_filt> - default None, if you want to filter on something other than edge_att_name as filter_att set the attribute name here.

Plug-in example:

```
plugin = a_r_a.AddAttributesByIfAny('my_rep_network','my_repnod_index',\
'-log10(E)',filt_dir='>',filt_value=0)
```

This example would calculate representative edges using $-\log_{10}(E)$ as the basis of the calculations. It will leave out any edges with $-\log_{10}(E) < 0$. The representative node network used would be my_rep_network which has the representative node index of my_repnod_index.

== CalcNodeSize ==

Description:

This plug-in adds the number of nodes in a representative node as an attribute.

Environments:

This plug-in will only work with PythoscapeEnvironment. If you want to have this plug-in operate on a sub-network, use the `initial_net` argument.

Example execution:

(Be sure to replace `<interface>`, `<create interface>`, `<environment>`, and `<create environment>` below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env

import <interface>

#Import plug-ins

import pythoscape.plugin.input.repnodel_stats as r_s

#Create interface and environment

<create interface>

<create environment>
```

#Create plug-ins

plugin =

r_s.CalcNodeSize(<network>,<reinode_att>,<size_att>,initial_net=<initial_net>)

#Execute plug-ins

<environment>.execute_plugin(plugin)

Required arguments:

<network> - name of representative node network

<reinode_att> - name of attribute that indexes representative nodes

<size_att> - name of node attribute that will contain the calculated size

Optional arguments:

<initial_net> - default None, if you set this to the name of a network, it will only consider nodes from that network

Because you can't run this plug-in using NetworkEnvironment, <initial_net> is used to constrain the plug-in to a specific sub-network.

Plug-in example:

```
plugin = r_s.CalcNodeSize('my_rep_network','my_repnodex_index','repnodex size')
```

This example would calculate the number of nodes in each representative node for the representative node network my_rep_network. It would use the representative node index my_repnodex_index and save the attribute with the name repnodex size.

== CalculateTMAAlignScores ==

Description:

This plug-in calculates edges between nodes that represent structures using the program TM-Align.

Environments:

This plug-in will work with both PythoscapeEnvironment and PythoscapeNetwork. Using PythoscapeNetwork restricts the inputted attributes to those nodes in a given network.

Example execution:

(Be sure to replace <interface>, <create interface>, <environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env

import <interface>

#Import plug-ins

import pythoscape.plugin.input.calculate_structure_edges as c_s_e

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin =\

c_s_e.CalculateTMAlignScores(<tm_align>,path_att=<path_att>,att_prefix=\

<att_prefix>)

#Execute plug-ins
```

`<environment>.execute_plugin(plugin)`

Required arguments:

`<tm_align>` = TM-Align executable path

Optional arguments:

`<path_att>` = default 'path', the attribute that holds the path to the structure file

`<att_prefix>` = default "", prefix to add to the beginning to attribute names from parser

Plug-in example:

`plugin = c_s_e.CalculateTMAAlignScores('talign')`

This example would calculate edges using the TM-Align executable at path "talign".

== CreateCDHITRepnodes ==

Description:

This plug-in creates a new meta-node network where the representative sequence (the longest sequence) chosen by CD-HIT stands in for all nodes in the same CD-HIT cluster. This plug-in creates the network if it doesn't already exist.

Plug-in software requirements:

You must have installed CD-HIT. This plug-in was tested on CD-HIT version 4.5.6. You can download CD-HIT from: <http://code.google.com/p/cdhit/>

Environments:

This plug-in will only work with PythoscapeEnvironment. If you want to have this plug-in operate on a sub-network, use the `initial_net` argument.

Example execution:

(Be sure to replace <interface>, <create interface>, <environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```

import pythoscape.main.environments as env

import <interface>

#Import plug-ins

import pythoscape.plugin.input.make_cdhit_repnodes as m_c_r

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin =\

m_c_r.CreateCDHITRepnodes(<network_name>,<cdhit_exe>,n=<n>,c=<c>,\
sequence_att=<sequence_att>,repnod_att=<repnod_att>,initial_net=\
<initial_net>,network_type=<network_type>,include_lengths=<include_lengths>)

#Execute plug-ins

<environment>.execute_plugin(plugin)

```

Required arguments:

<network_name> - network to create that contains representative nodes.

"Remember that this name has to be under 13 characters for the MySQL interface! " This network will be created if it doesn't already exist.

<cdhit_exe> - path to CDHIT executable. It is the executable file named 'cdhit' that comes with the program.

Optional arguments:

<n> - default is 2, word size for use in CDHIT. Set n to a new value to change it use a different word size.

<c> - default is 0.4, sequence identity cutoff for CDHIT to use. Set c to a new value to increase clustering stringency.

<sequence_att> - default is 'sequence', attribute that Pythoscape will use to locate sequences.

<reinode_att> - default is 'cdhit <c>' where <c> is the sequence identity cutoff, attribute that Pythoscape will use to store the cluster index number. For example, if you set the threshold to 0.5, the representative node attribute name would be 'cdhit 0.5'. If you want to have a custom attribute name, set this parameter to the desired name.

<initial_net> - default None, if you set this to the name of a network, it will only consider nodes from that network

Because you can't run this plug-in using NetworkEnvironment, <initial_net> is used to constrain the plug-in to a specific sub-network.

<network_type> - default None, if you set this, it will include a network_type in the network created

<include_lengths> - default None, if you input an integer here, it will not include sequences below a given length in this analysis. This is useful if you do not want to include sequence fragments in your representative node network.

Plug-in example:

```
plugin = m_c_r.CreateCDHITRepnodes('my_rep_net','/my/cdhit/directory/cdhit')
```

This example create a representative node network called my_rep_net using the CD-HIT executable at /my/cdhit/directory/cdhit and make represnetative nodes using a 40% sequence identity cutoff.

== !ImportAttributeTable ==

Description:

This plug-in uses a table-like file to add attributes into Pythoscape. These attributes are added as string attributes. The layout of the file should be multiple columns, the first column being the key to match to, either node id or node identifier. The rest of the columns should be attributes. The top row should identify the attributes. An example layout is below:

id_or_identifier_to_match_to attribute_to_add attribute_to_add_2 ...

Environments:

This plug-in will work with both PythoscapeEnvironment and PythoscapeNetwork. Using PythoscapeNetwork restricts the inputted attributes to those nodes in a given network.

Example execution:

(Be sure to replace <interface>,<create interface>,<environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env
```

```
import <interface>
```

```
#Import plug-ins
```

```
import pythoscape.plugin.input.add_attribute_table as a_a_t
```

```
#Create interface and environment
```

```
<create interface>
```

```
<create environment>
```

```
#Create plug-ins
```

```
plugin = a_a_t.ImportAttributeTable(<filename>,csv = <True or\
False>,<match_against> = <match_against>)
```

```
#Execute plug-ins
```

```
<environment>.execute_plugin(plugin)
```

Required arguments:

<filename> = file name of identifier table

Optional arguments:

csv - default False, if True the plugin interprets the file as a csv

match_against = default None, set to the name of the identifier name if the first column has identifiers and not node ids

Plug-in example:

```
plugin =\  
a_a_t.ImportAttributeTable('my_file.txt',match_against='existing_identifer')
```

This example would add attributes from my_file.txt to Pythoscape. It would add them to the nodes that match with the 'existing_identifier' identifier from the first column.

== ImportFromFastaFile ==

Description:

This plug-in takes sequences from a fasta file and adds them into a network as nodes. Hashes the sequences with the built-in standard `main.plugin.hasher` to generate ids.

Environments:

This plug-in will work with both `PythoscapeEnvironment` and `PythoscapeNetwork`. If you are storing multiple distinct networks in a single database/file, then use `PythoscapeNetwork`.

Example execution:

(Be sure to replace `<interface>`, `<create interface>`, `<environment>`, and `<create environment>` below to the appropriate values. See the `UsePlugins` documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env
```

```
import <interface>
```

```
#Import plug-ins
```

```
import pythoscape.plugin.input.import_sequences as i_s
```



```
#Create interface and environment
```

```
<create interface>
```

```
<create environment>
```

```
#Create plug-ins
```

```
plugin =\
```

```
i_s.ImportFromFastaFile(<sequence_file>,id_re=<id_re>,id_name=<id_name>,\  
sequence_attr=<sequence_attr>,flat_id=<True or False>)
```

```
#Execute plug-ins
```

```
<environment>.execute_plugin(plugin)
```

Required arguments:

<sequence_file> - fasta file name

Optional arguments:

If both id_re and id_name are set, then a node identifier is also added to the node when it is added to the network. For example, if you wanted to identify a sequence as having a certain GI number, this function is used. Because regular expressions are used for this function and are not intuitive, several have already been included in Pythoscape for convenience. See the documentation in

IncludedRegularExpressions for more information. If you set flat_id to True, then you only have to set id_name because it uses the whole fasta description line as the identifier value.

<id_re> - default None, regular expression that locates the node identifier in the fasta title

<id_name> - default None, type of node identifier recognized by the regular expression

<flat_id> - default False, deposit whole description line as identifier value

<sequence_attr> - default 'sequence', name of attribute that sequences are deposited under

Plug-in example:

```
plugin =\
i_s.ImportFromFastaFile('my_seqs.fa',id_re=RE_PATTERNS['SFLD_GI'],\
id_name='GI' )
```

This example would pull sequences from my_seqs.fa and use the SFLD_GI regular expression to add GI identifiers to the nodes.

== ImportFromUniProt ==

Description:

This plug-in imports annotations from UniProt sequence records. The annotations it currently imports are as follows:

- * uniprot sequence
- * organism
- * description
- * accessions
- * family
- * subfamily
- * superfamily
- * pfam
- * pdb
- * taxonomy

Currently, this is done using module defined dictionaries and function called `make_info`. In the future this will be changed to a much more extensible method of using a wrapper class on top of the Biopython UniProt record object.

Environments:

This plug-in will work with both `PythoscapeEnvironment` and `PythoscapeNetwork`.

Example execution:

(Be sure to replace `<interface>`, `<create interface>`, `<environment>`, and `<create environment>` below to the appropriate values. See the `UsePlugins` documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env
```

```
import <interface>
```

```
#Import plug-ins
```

```
import pythoscape.plugin.input.input_bio.add_uniprot_info as a_u_i
```

```
#Create interface and environment
```

```
<create interface>
```

<create environment>

#Create plug-ins

```
plugin = a_u_i.ImportFromUniProt(data_file=<data_file>,structure=<True/False>,\n\n                                uniprot_id=<uniprot_id>,entry_name=<True/False>,\n\n                                taxonomy_classifiers=<taxonomy_classifiers>,\n\n                                node_file=<node_file>,name_file=<name_file>)
```

#Execute plug-ins

```
<environment>.execute_plugin(plugin)
```

Required arguments:

None

Optional arguments:

<data_file> - default is None. If this is set then the plug-in reads the records from the file. If not, it pulls the records directly from UniProt servers. If it is a file, it expects the raw file format.

<structure> - default is False. Is this a structure network?

<uniprot_id> - default is 'UniprotKB'. What is a attribute name for the UniProt AC number?

<entry_name> - default is False. Match the identifier against entry_name?

<taxonomy classifiers> - list of levels of the taxonomy that you want included, like 'kingdom' or 'phylum'

<node_file> - path to NCBI node file

<name_file> - path to NCBI name file

Plug-in example:

```
plugin = a_u_i.ImportFromUniProt(data_file='my_data.txt',uniprot_id='Uniprot',\  
                                taxonomy_classifiers=['kingdom','phylum'],\  
                                node_file='node.dmp',name_file='name.dmp')
```

This example will add annotations using the downloaded file of raw UniProt data, my_data.txt and map to UniProt ACs in the identifier Uniprot. It will add taxonomic data for kingdoms and phylum using the two NCBI files node.dmp and name.dmp.

== ImportIdentifierTable ==

Description:

This plug-in uses a table-like file to add identifiers into Pythoscape. The layout of the file should be two columns, the first column being the key to match to, either node id or node identifier, and the second column being the new identifier to add.

id_or_identifier_to_match_to identifier_to_add

Environments:

This plug-in will work with both PythoscapeEnvironment and PythoscapeNetwork. Using PythoscapeNetwork restricts the inputted identifiers to those nodes in a given network.

Example execution:

(Be sure to replace <interface>, <create interface>, <environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```

import pythoscape.main.environments as env

import <interface>

#Import plug-ins

import pythoscape.plugin.input.add_attribute_table as a_a_t

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin = a_a_t.ImportIdentifierTable(<match_to>,<filename>,dont_skip = <True\
or False>,match_against = <match_against>)

#Execute plug-ins

<environment>.execute_plugin(plugin)

```

Required arguments:

<match_to> = attribute name of identifier to add

<filename> = file name of identifier table

Optional arguments:

dont_skip = default False, set to True to add first list from file (the first line is assumed to be column headings)

match_against = default None, set to the name of the identifier name if the first column has identifiers and not node ids

Plug-in example:

```
plugin = a_a_t.ImportIdentifierTable('new_identifier','my_file.txt',match_against=\
'existing_identifier')
```

This example would add 'new_identifier' identifiers from the second column from my_file.txt to Pythoscape. It would add them to the nodes that match with the 'existing_identifier' identifier from the first column.

== PDBImportFromCSVFile ==

Description:

This plug-in imports PDB chains using a local mirror of the PDB database and a CSV file. The CSV file to import structures into Pythoscape has the following form:

pdb_id_1,model_1,chain_1,identifier_1(optional)

pdb_id_2,model_2,chain_2,identifier_2(optional)

...

The identifier column only needs to be included if you are associating the structure nodes with pre-existing sequence nodes.

Environments:

This plug-in will work with both PythoscapeEnvironment and PythoscapeNetwork. Using PythoscapeNetwork restricts the inputted identifiers to those nodes in a given network.

Example execution:

(Be sure to replace <interface>,<create interface>,<environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env
```

```
import <interface>
```

```
#Import plug-ins
```

```
import pythoscape.plugin.input_bio.import_structures as i_s
```

```
#Create interface and environment
```

```
<create interface>
```

```
<create environment>
```

```
#Create plug-ins
```

```
plugin =\
```

```
i_s.PDBImportFromCSVFile(<pdb_path>,<csv_file>,<local_path>,path_att=\
```

```
<path_att>,no_files=<True or\
```

```
False>,add_to_sequence_network=<add_to_sequence_network>)
```

```
#Execute plug-ins
```

```
<environment>.execute_plugin(plugin)
```

Required arguments:

<pdb_path> = path to root of local copy of PDB database

<csv_file> = path to csv file describing input structures (see above)

<local_path> = path to directory to store inputted structure chains

Optional arguments:

<path_att> = default 'path', attribute to store structure path

<no_files> = default False, if True no files are copied

<add_to_sequence_network> = default None, if a value is set no new nodes are made and instead node attributes are added to the node that has the identifier_value given in the csv with the identifier_type specified by the value set here

Plug-in example:

```
plugin = i_s.PDBImportFromCSVFile('/pdb_root','struc.csv','/home/strucs')
```

This example would pull from the PDB mirror at /pdb_root the structures given in struc.csv and put the copied chains into /home/strucs.

B.2.11. Output plug-ins

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/OutputPlugins>

== CorrelationPlot ==

Description:

This plug-in calculates a correlation plot of two edge attributes using the R statistics package and Rpy as the python to R interface.

Environments:

This plug-in will work with both PythoscapeEnvironment and PythoscapeNetwork. Using PythoscapeNetwork restricts the output to those nodes in a given network.

Example execution:

(Be sure to replace <interface>,<create interface>,<environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env
```

```
import <interface>
```

```
#Import plug-ins
```

```
import pythoscape.plugin.output_r.output_correlation as o_c
```

```
#Create interface and environment
```

```
<create interface>
```

```
<create environment>
```

```
#Create plug-ins
```

```
plugin = o_c.CorrelationPlot(<att_x>,<att_y>,<outfile>,without_rsqr=<True or\
False>,transform_x=<transform_x>,transform_y=<transform_y>,att_x_name=\
<att_x_name>,att_y_name=<att_y_name>)
```

```
#Execute plug-ins
```

```
<environment>.execute_plugin(plugin)
```

Required arguments:

<att_x> = name of edge attribute for x axis

<att_y> = name of edge attribute for y axis

<outfile> = name of output pdf file

Optional arguments:

<without_rsqu> = default False, if True does not calculate rsq for correlation plot

<transform_x> = default None, function or callable object that transforms the x edge attribute

<transform_y> = default None, function or callable object that transforms the y edge attribute

<att_x_name> = default None, renames x attribute on output plot

<att_y_name> = default None, renames y attribute on output plot

Plug-in example:

```
plugin = plugin = o_c.CorrelationPlot('edge_att_1','edge_att_2','output.pdf')
```

This example would output a pdf plot to 'output.pdf' of edge_att_1 on the x-axis and edge_att_2 on the y-axis and the r-square correlation between them.

== OutputAttributeClusters ==

Description:

This plug-in outputs a file that gives information about what nodes or attributes are either a given set of representative nodes or all representative nodes. The output file has the following form:

Representative node id

!-----

Node_id_1 (tab) Given attribute/identifier

Node_id_2 (tab) Given attribute/identifier

...

Environments:

This plug-in will only work with PythoscapeEnvironment.

Example execution:

(Be sure to replace <interface>, <create interface>, <environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env
```

```
import <interface>
```

```
#Import plug-ins
```

```
import pythoscape.plugin.output.output_renode_info as o_r_i
```

```
#Create interface and environment
```

```
<create interface>
```

```
<create environment>
```

```
#Create plug-ins
```

```
plugin = o_r_i.OutputAttributeClusters(<network>,<reinode_att>,<output_file>,\
output_att = <output_att>,is_identifier = <True or False>, in_reinodes =\
<in_reinodes>,no_titles = <True or False>)
```

#Execute plug-ins

```
<environment>.execute_plugin(plugin)
```

Required arguments:

<network> - name of representative node network

<reinode_att> - name of attribute that indexes representative nodes

<output_file> - name of file to output

Optional arguments:

<output_att> = default None, name of attribute or identifier to output along with node names

<is_identifier> = default False, True if <output_att> is an identifier type

<in_repnodes> = default False, set to a file name if you want to restrict to a set of representative nodes. This file should be one representative node id per line.

<no_titles> = default False, if set to True, the representative node header lines are not included in the output file

Plug-in example:

```
plugin = o_r_i.OutputAttributeClusters('my_network','cdhit\
0.7','my_output.txt',output_att='hit?')
```

This would output the values of attribute 'hit?' for all the represented nodes that are defined in the representative node network 'my_network' with representative node indexing attribute 'cdhit 0.7'. The output would be written to 'my_output.txt'.

== OutputAttributeTable ==

Description:

This plug-in outputs a table-like file that describes attributes for each node in the following way:

attribute 1 attribute 2 etc.

node id

Environments:

This plug-in will work with both PythoscapeEnvironment and PythoscapeNetwork. Using PythoscapeNetwork restricts the output to those nodes in a given network.

Example execution:

(Be sure to replace <interface>, <create interface>, <environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env
```

```
import <interface>
```

```
#Import plug-ins
```

```
import pythoscape.plugin.output.output_attribute_tables as o_a_t
```

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin =\

```
o_a_t.OutputAttributeTable(<output_file>,attribute_names=<attribute_names>,\n                           justify=<justify>,leave_out=<leave_out>,\n                           csv=<True or False>))
```

#Execute plug-ins

<environment>.execute_plugin(plugin)

Required arguments:

<output_file> = name of output file

Optional arguments:

<attribute_names> = default None, list of attributes to output, skip others

<justify> = default 15, number of spaces to justify columns

<leave_out> = default None, list of attributes not to output, use others

<csv> = default False, set to True to output a comma separated value file instead of a table

Plug-in example:

```
plugin = o_a_t.OutputAttributeTable('myfile.txt')
```

This example would output the table to the file myfile.txt.

== OutputIdentifierTable ==

Description:

This plug-in outputs a table-like file that describes identifiers for each node in the following way:

identifier 1 identifier 2 etc.

node id

If a single node has multiple identifiers of the same type, they are each outputted on a different line.

Environments:

This plug-in will work with both PythoscapeEnvironment and PythoscapeNetwork. Using PythoscapeNetwork restricts the output to those nodes in a given network.

Example execution:

(Be sure to replace <interface>,<create interface>,<environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```
import pythoscape.main.environments as env
```

```
import <interface>
```

```
#Import plug-ins
```

```
import pythoscape.plugin.output.output_attribute_tables as o_a_t
```

```
#Create interface and environment
```

```
<create interface>
```

```
<create environment>
```

```
#Create plug-ins
```

```
plugin = o_a_t.OutputIdentifierTable(<output_file>,csv=<True or False>)
```

```
#Execute plug-ins
```

```
<environment>.execute_plugin(plugin)
```

Required arguments:

<output_file> = name of output file

Optional arguments:

<csv> = default False, set to True to output a comma separated value file instead of a table

Plug-in example:

```
plugin = o_a_t.OutputIdentifierTable('myfile.txt')
```

This example would output the table to the file myfile.txt.

== OutputXGMML ==

Description:

This plug-in outputs a network held in Pythoscape into a XGMML file that can be opened using Cytoscape.

Environments:

This plug-in will work with both PythoscapeEnvironment and PythoscapeNetwork. Using PythoscapeNetwork restricts the edges to the nodes in the specified network. Make sure to use PythoscapeNetwork if you have created and want to output a meta-node network or a network of selected nodes.

Example execution:

(Be sure to replace <interface>, <create interface>, <environment>, and <create environment> below to the appropriate values. See the UsePlugins documentation for information on setting these lines appropriately)

```

import pythoscape.main.environments as env

import <interface>

#Import plug-ins

import pythoscape.plugin.output.output_xgmml as o_x

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin = o_x.OutputXGMML(<output>,<label>,<filt_name=<filt_name>,<filt_dir =\
<'>','>','=','<' or '<='>,<filt_value=<filt_value>)

#Execute plug-ins

<environment>.execute_plugin(plugin)

```

Required arguments:

<output> - name of file into which network will be outputted

<label> - XGMML label of network. In Cytoscape, this is the title of the network.

Optional arguments:

The three parameters `filt_name`, `filt_dir` and `filt_value` are used to pre-filter the edges that are outputted into the network. Any edge that doesn't meet the filter will not be included.

`<filt_name>` - edge attribute that you want to filter on

`<filt_dir>` - pick '>', '>=', '<' or '<=' depending on if you want edges with the attribute greater or less than the filter value

`<filt_value>` - the value of the filter

Plug-in example:

```
plugin = o_x.OutputXGMML('my_net.xgmml','My network!','\  
filt_name='-log10(E)',filt_dir='>',filt_value=5)
```

This would output the network to the file `my_net.xgmml` with the label 'My network!'. Only edges that have the edge attribute $-\log_{10}(E)$ of greater than 5 will be included in the file.

== SequenceTableRun ==

Description:

This plug-in outputs all edges in the complete network in numbered table file and sequence file so that edges can be calculated outside of Pythoscape in parallel. For more information on resources provided to run edges in parallel, see [ParallelResources](#).

Environments:

This plug-in will work with both `PythoscapeEnvironment` and `PythoscapeNetwork`. Using `PythoscapeNetwork` restricts the outputted edges to the nodes in the specified network.

Example execution:

(Be sure to replace `<interface>`, `<create interface>`, `<environment>`, and `<create environment>` below to the appropriate values. See the [UsePlugins](#) documentation for information on setting these lines appropriately)

```

import pythoscape.main.environments as env

import <interface>

#Import plug-ins

import pythoscape.plugin.output.output_table_runs as o_t_r

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin =\ o_t_r.SequenceTableRun(<directory>,<runs_in_file>,sequence_att=\
<sequence_att>,seq_file=<seq_file>,compress=<True or\
False>,new_nodes_only=<True or False>)

#Execute plug-ins

<environment>.execute_plugin(plugin)

```

Required arguments:

<directory> - location on disk to output table files

<runs_in_file> - number of edges to calculate in each table

Optional arguments:

<sequence_att> - default 'sequence', if the sequence attribute is not named 'sequence', this needs to be set to the appropriate value

<seq_file> - default <directory>/sequences.fa; this is the name and location of the file that contains all the sequences

<compress> - default False, should the plug-in compress all the outputs with gzip?

<new_nodes_only> - default False, set this to True if you just added nodes to the an already existing network and you only want to output edges that correspond to those nodes.

Plug-in example:

```
plugin = o_t_r.SequenceTableRun('/home/myuser/mydata/tables',100000)
```

This example would output all the tables in the directory /home/myuser/mydata/tables and each table file would run 100000 edges (a reasonable number for a large network).

== StructureTableRun ==

Description:

This plug-in outputs all edges in the complete network in numbered table file and structure file names so that edges can be calculated outside of Pythoscape in parallel. For more information on resources provided to run edges in parallel, see [ParallelResources](#).

Environments:

This plug-in will work with both `PythoscapeEnvironment` and `PythoscapeNetwork`. Using `PythoscapeNetwork` restricts the outputted edges to the nodes in the specified network.

Example execution:

(Be sure to replace `<interface>`, `<create interface>`, `<environment>`, and `<create environment>` below to the appropriate values. See the [UsePlugins](#) documentation for information on setting these lines appropriately)

```

import pythoscape.main.environments as env

import <interface>

#Import plug-ins

import pythoscape.plugin.output.output_table_runs as o_t_r

#Create interface and environment

<create interface>

<create environment>

#Create plug-ins

plugin =

o_t_r.SequenceTableRun(<directory>,<runs_in_file>,path_att=<path_att>)

#Execute plug-ins

<environment>.execute_plugin(plugin)

```

Required arguments:

<directory> - location on disk to output table files

<runs_in_file> - number of edges to calculate in each table

Optional arguments:

<path_att> - default 'path', if the structure path att is something other than path, then this needs to be set

Plug-in example:

```
plugin = o_t_r.StructureTableRun('/home/myuser/mydata/tables',1000)
```

This example would output all the tables in the directory
/home/myuser/mydata/tables and each table file would run 1000 edges (a
reasonable number for a large structure network).

B.3. Tutorials

B.3.1. Scripting with objects tutorial

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/ScriptingWithObjects>

For this exercise, we will create a small test network to work from. Once you have tried several of the commands on a small test network, you can try them on the larger network of your choosing. All of the following commands can be run in either a run file or directly scripting in the Python interpreter. I prefer to make a

run file and then call the file using python <file name>, because then I can quickly change commands and try again when there is a syntax mistake.

Let's begin.

```
#Import interface
```

```
import pythoscape.interface.local_interface as l_i
```

```
#Create interface
```

```
my_interface = l_i.LocalInterface(<save_directory>)
```

For simplicities sake, we will use LocalInterface during this tutorial. For more information on the various interfaces and their use, see FrameworkOverview and UsePlugins. The previous script creates this interface. All data input and export in Pythoscape is done through a Pythoscape interface. Save the file as above and continue to the next step without executing it. (If you do execute it, no harm is done – it will just create an empty storage file.) Make sure that the directory you input already exists. It will save the storage files in that directory.

```
#Import interface
```

```
import pythoscape.interface.local_interface as l_i
```

```
#Create interface
```

```

my_interface = l_i.LocalInterface(<save_directory>)

#Create network

my_net = my_interface.Network(network_name='test net',\
network_type='test type')

my_interface.push(my_net)

#Create node

my_node = my_interface.Node(id='123456')

my_interface.push(my_node)

```

If you run a file with the previous code, it will create a network and node. The following code will show you how to access them.

```

#Import interface

import pythoscape.interface.local_interface as l_i

#Create interface

my_interface = l_i.LocalInterface(<save_directory>)

#Get nodes

for node in my_interface.pull(my_interface.Node()):

    print node

```

This code queries the network using the 'pull' function. This queries based on what is present in the object that you include in the pull. If you pull a node that is empty, it will include all nodes in the generator that is created. If you pull with a node containing a defined id, it will only pull ones that match the parameters given. If you do a pull on a more complex object with multiple parameters (an edge, for example), it will pull based on the fields that you include and leave all the others as wildcards. If we query for an object that doesn't exist in the database, as below, nothing is returned.

```
#Import interface
import pythoscape.interface.local_interface as l_i

#Create interface
my_interface = l_i.LocalInterface(<save_directory>)

#Get node '345678'
for node in my_interface.pull(my_interface.Node(id = '345678')):
    print node
```

In the next example, we will add more objects so that we can try more advanced queries.

```
#Import interface

import pythoscape.interface.local_interface as l_i


#Create interface

my_interface = l_i.LocalInterface(<save_directory>)


#Add nodes

my_node = my_interface.Node(id='123456')
my_node_2 = my_interface.Node(id='7891011')
my_interface.push(my_node_2)
my_node_3 = my_interface.Node(id='155669')
my_interface.push(my_node_3)


#Add edges

my_edge =\
my_interface.Edge(source_node=my_node,target_node=my_node_2)
my_interface.push(my_edge)

my_edge_2 =\
my_interface.Edge(source_node=my_node_2,target_node=my_node_3)
my_interface.push(my_edge_2)

my_edge_3 =\
my_interface.Edge(source_node=my_node,target_node=my_node_3)
my_interface.push(my_edge_3)
```

```
#Add attributes
```

```
my_att = my_interface.Attribute(element_obj=my_edge,\n\nattribute_name='test name',attribute_value=5)\n\nmy_att_2 = my_interface.Attribute(element_obj=my_edge_2,\n\nattribute_name='test name',attribute_value=3)\n\nmy_att_3 = my_interface.Attribute(element_obj=my_edge_3,\n\nattribute_name='test name',attribute_value=1)\n\nmy_interface.push(my_att)\n\nmy_interface.push(my_att_2)\n\nmy_interface.push(my_att_3)
```

In the previous code block, we added several nodes, edges and attributes to the network. Now we can query the edges in this network using a filter.

```
#Import interface
```

```
import pythoscape.interface.local_interface as l_i
```

```
#Create interface
```

```
my_interface = l_i.LocalInterface(<save_directory>)
```

```
#Create filter
```

```
my_filter = my_interface.Filter(filter_name='test name',direction='<',filter_value=3)
```

```
#Pull edges with filter

for edge in my_interface.pull(my_interface.Edge(filter=my_filter)):

    print edge
```

The next section will demonstrate complex parameters. You can add either identifiers or networks to an option using the <object>.add_identifiers or <object>.add_networks functions. We will first add some complex parameters.

```
#Import interface

import pythoscape.interface.local_interface as l_i

#Create interface

my_interface = l_i.LocalInterface(<save_directory>)

#Add identifier and network to node

my_node = my_interface.Node(id='123456')

my_node.add_identifiers([my_interface.Identifier(identifier_type='test_iden',\
identifier_value='123456')])

my_node.add_networks([my_interface.Network(network_name='test net')])

my_interface.push(my_node)

#Get nodes
```

```
for node in my_interface.pull(my_interface.Node()):  
    print node
```

You can query using complex parameters as well.

```
#Import interface  
import pythoscape.interface.local_interface as l_i  
  
#Create interface  
my_interface = l_i.LocalInterface(<save_directory>)  
  
#Create query node  
my_node = my_interface.Node(id='123456')  
my_node.add_networks([my_interface.Network(network_name='my test')])  
  
#Use query node to pull data  
for node in my_interface.pull(my_node):  
    print node
```

The `interface.delete()` behaves just like `pull`, except that it deletes the selection instead of querying it. You can also clear the entire database by doing the following.


```
#Import interface

import pythoscape.interface.local_interface as l_i

#Create interface

my_interface = l_i.LocalInterface(<save_directory>)

#Clear database

my_interface.clear()
```

B.3.2. Small network tutorial with local interface

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/LocalInterfaceSequenceNetwork>

This tutorial uses the local interface to create a representative network of Sigma and sigma-like glutathione transferases. You can download the sequence file, `sigma_gst.fasta`, from the tutorial webpage in the "attachments" section.

This tutorial assumes that you have looked through `FrameworkOverview`, `UsePlugins`, `PipelineAndPlugins` and `ScriptingWithObjects` to get a basic sense of how Pythoscape works.

To begin this tutorial, Pythoscape should have been installed as instructed in the `InstallationInstructions`. You must have installed Biopython before installing

Pythoscape as we will use the plug-in ImportFromUniProt. You need not have installed rpy or MySQLdb.

This tutorial will take a couple hours of computer run time to complete using the file sigma_gst.fasta. If you want a quick run through the tutorial without a long computation time just to learn the steps, use the file small_sigma_gst.fasta attached below.

== Step 1: Add sequences to network ==

Make a run.py script as below and execute it. There is a line that sets up the interface with the argument <save directory>. This must be changed to a directory on your hard disk to save the network information files.

```
#Import Pythoscape
```

```
from pythoscape.auxiliary.re_patterns import RE_PATTERNS
```

```
import pythoscape.main.environments as env
```

```
import pythoscape.interface.local_interface as l_i
```

```
#Import plug-ins
```

```
import pythoscape.plugin.input.import_sequences as i_s
```

```
#Create interface and environment
```

```

my_interface = l_i.LocalInterface(<save directory>)
my_pytho = env.PythoscapeEnvironment(my_interface)

#Create plug-ins

#Import sequences

plugin_1 =\
i_s.ImportFromFastaFile('sigma_gst.fasta',id_re=RE_PATTERNS['UniprotKB'],\
id_name='Uniprot')

#Execute plug-ins

my_pytho.execute_plugin(plugin_1)

```

The name of the fasta sequence file we are going to import is sigma_gst.fasta. We will use the UniprotKB regular expression to parse the sequence tags and call those identifiers 'Uniprot'. If you want to make a network with a different sequence file, just use that sequence file in place of the one here. Be aware that if you use a different sequence file, you might have to change the regular expression that is used to parse the sequence headers. For a full list of included regular expressions, see IncludedRegularExpressions.

== Step 2: Add edges to network ==

Change the run.py file to the following and execute it to add edges to the network.

```
#Import Pythoscape
```

```
import pythoscape.main.environments as env
```

```
import pythoscape.interface.local_interface as l_i
```

```
#Import plug-ins
```

```
import pythoscape.plugin.input.add_local_blast as a_l_b
```

```
#Create interface and environment
```

```
my_interface = l_i.LocalInterface(<save directory>)
```

```
my_pytho = env.PythoscapeEnvironment(my_interface)
```

```
#Create plug-ins
```

```
#Input edges to Pythoscape
```

```
plugin_2 = a_l_b.AddBLASTEEdgesFromLocalBLAST(<blastp\  
executable>,include_atts=['-log10(E)'])
```

```
#Execute plug-ins
```

```
my_pytho.execute_plugin(plugin_2)
```

As before, you must set <save directory>. The above plug-in adds the edges into the interface. Replace <blastp executable> with the location of the BLAST+ blastp executable. The option include_atts is a list of edge attributes that you want to include in the network. The available options are: 'bit_score', 'm*n', 'alignment_len', 'alignment_identities', 'query_start', 'query_end', 'subject_start', 'subject_end', 'bit_score', '-ln(E)', '-log10(E)'. In this example, we are just adding -log10(E). We use log transforms of the E-value because reliable storage of very small exponential numbers can be difficult.

== Step 3: Add information from UniProt ==

Change the run.py file to the following and execute it.

```
#Import Pythoscape

import pythoscape.main.environments as env
import pythoscape.interface.local_interface as l_i

#Import plug-ins

import pythoscape.plugin.input_bio.add_uniprot_info as a_u_i

#Create interface and environment

my_interface = l_i.LocalInterface(<save directory>)

my_pytho = env.PythoscapeEnvironment(my_interface)
```

```
#Create plug-ins
```

```
#Input edges to Pythoscape
```

```
plugin_3 = a_u_i.ImportFromUniProt(uniprot_id='Uniprot')
```

```
#Execute plug-ins
```

```
my_pytho.execute_plugin(plugin_3)
```

Warning: This step will only work if you had Biopython installed on your computer before installing Pythoscape. If you get the error: "ImportError: No module named input_bio.add_uniprot_info", this was the problem. Make sure that you have Biopython installed and then re-install Pythoscape.

As before, you must set <save directory>. The above plug-in adds information from Uniprot. It adds the following node attributes (not including taxonomic information): uniprot sequence, organism, description, accessions, family, subfamily, superfamily, pdb, pfam. To add taxonomic information, include the classifiers you want included as a list for taxonomy classifiers. To add taxonomic information, you also need two files from the NCBI taxonomy database, the nodes.dmp file and the names.dmp file. This example does not add taxonomic information. For more information on doing so, see [InputPlugins#ImportFromUniProt](#).

== Step 4: Create a representative node network ==

Change the run.py file to the following and execute it.

```
#Import Pythoscape
```

```
import pythoscape.main.environments as env
```

```
import pythoscape.interface.local_interface as l_i
```

```
#Import plug-ins
```

```
import pythoscape.plugin.input.make_cdhit_repnodes as m_c_r
```

```
#Create interface and environment
```

```
my_interface = l_i.LocalInterface(<save directory>)
```

```
my_pytho = env.PythoscapeEnvironment(my_interface)
```

```
#Create plug-ins
```

```
#Create representative network
```

```
plugin_4 = m_c_r.CreateCDHITRepnodes('rep-net',<cd-hit  
executable>,c=0.7,n=5)
```

```
#Execute plug-ins
```

```
my_pytho.execute_plugin(plugin_4)
```

As before, you must set <save directory>. The above plug-in creates a representative node network using CDHIT. Replace <cd-hit executable> with the path of the cdhit executable. As stated in the documentation for the plug-in, this should come from the CDHIT package v4.5.6 or greater. The first two plugin arguments are mandatory, the name of the network and then the location of the CDHIT executable. In this plug-in, the values of c and n for CDHIT default to 0.4 and 2 respectively. If you want to change them, you must include them in the arguments used to call the plug-in. This plug-in creates a new PythoscapeNetwork called 'rep-net' and an attribute called 'cdhit 0.7'. The attribute reflects the value of c. If you do not put in any value for c and the default of 0.4 is used, the attribute is called 'cdhit 0.4'.

== Step 5: Calculate the size of the representative nodes ==

Change the run.py file to the following and execute it.

```
#Import Pythoscape
```

```
import pythoscape.main.environments as env
```

```
import pythoscape.interface.local_interface as l_i
```

```
#Import plug-ins
```



```

import pythoscape.plugin.input.repnnode_stats as r_s

#Create interface and environment

my_interface = l_i.LocalInterface(<save directory>)

my_pytho = env.PythoscapeEnvironment(my_interface)


#Create plug-ins


#Create representative network

plugin_5 = r_s.CalcNodeSize('rep-net','cdhit 0.7','node size')


#Execute plug-ins

my_pytho.execute_plugin(plugin_5)

```

As before, you must set <save directory>. The above plugin adds a node attribute to the representative nodes that reflects how many nodes it represents. There are three required arguments, the name of your representative node PythoscapeNetwork, the node attribute that numbers the clusters and the node attribute name you want created to hold this value.

== Step 6: Calculate representative node edges ==

Change the run.py file to the following and execute it.

```

#Import Pythoscape

import pythoscape.main.environments as env

import pythoscape.interface.local_interface as l_i


#Import plug-ins

import pythoscape.plugin.input.add_repnod_edges as a_r_e


#Create interface and environment

my_interface = l_i.LocalInterface(<save directory>)

my_pytho = env.PythoscapeEnvironment(my_interface)


#Create plug-ins


#Create representative network

plugin_6 = a_r_e.AddEdgesToRepnodNetwork('rep-net','cdhit 0.7',\
'-log10(E)',filt_dir='>',filt_value=0)


#Execute plug-ins

my_pytho.execute_plugin(plugin_6)

```

As before, you must set <save directory>. The above plug-in calculates representative node edges, adds them to the PythoscapeNetwork and adds edge

attributes that correspond to those edges. There are three required arguments, the name of your representative node PythoscapeNetwork, the node attribute that numbers the clusters and the edge attribute value that you want to base these edges on. You can also include a filter if you want by adding `filt_dir = <filter direction>` and `filt_value = <filter value>`. We have added a filter of `-log10(E) > 0` because we assume that E-values greater than 1 are not useful and should be ignored.

This plugin makes three edge attributes for all representative edges called 'rep-net mean', 'rep-net min/max' and 'rep-net total'. The name of the representative node network is added to 'mean', 'min/max' and 'total', so the edge attribute names will change to correspond to the network you create.

Mean is the mean edge, min/max is either the minimum or maximum edge value as you indicate, and total is the number of edges that the edge represents. For more information on this, see the plug-in documentation: [InputPlugins#AddEdgesToRepnodNetwork](#).

== Step 7: Add representative node attributes ==

Change the `run.py` file to the following and execute it.

```
#Import Pythoscape
```

```

import pythoscape.main.environments as env

import pythoscape.interface.local_interface as l_i

#Import plug-ins

import pythoscape.plugin.input.add_repnod_atts as a_r_a

#Create interface and environment

my_interface = l_i.LocalInterface(<save directory>)

my_pytho = env.PythoscapeEnvironment(my_interface)

#Create plug-ins

#Create representative node attributes

plugin_7 = a_r_a.AddAttributesByIfAny('rep-net','cdhit 0.7','family',\
'repnod family')

#Execute plug-ins

my_pytho.execute_plugin(plugin_7)

```

As before, you must set <save directory>.

The above plug-in calculates attributes for representative nodes. It uses 'if/any' voting, which means that it concatenates a list of all the possible options that a node attribute can be for all members of the group.

There are four required arguments, the name of your representative node PythoscapeNetwork, the node attribute that numbers the clusters, the node attribute value that you want to base the new representative value on and the new name of the representative attribute.

== Step 8: Output the representative node network (Optional) ==

Change the run.py file to the following and execute it.

```
#Import Pythoscape
import pythoscape.main.environments as env
import pythoscape.interface.local_interface as l_i

#Import plug-ins
import pythoscape.plugin.output.output_xgmml as o_x

#Create interface and environment
my_interface = l_i.LocalInterface(<save directory>)
my_pytho = env.PythoscapeEnvironment(my_interface)
```

```

my_net = env.PythoscapeNetwork('rep-net',my_interface)

#Create plug-ins

#Output network
out_plugin_1 = o_x.OutputXGMML('gst-sigma-rep-net-20.xgmml',\
'sigma gst representative node network @ 20',filt_name='rep-net
mean',filt_dir='>',filt_value=20)

#Execute plug-ins

my_net.execute_plugin(out_plugin_1)

```

As before, you must set <save directory>.

The above plug-in outputs the network in XGMML format so that you can view it in Cytoscape. For more information on visualizing networks in Cytoscape, see <http://www.cytoscape.org>. The first two arguments are required. The first is that name of the output file and the second is the title of the network. You can also add an edge filter by filling in the values of `filt_name`, `filt_dir` and `filt_value` as appropriate.

Notice that we are going to run this plug-in on the new `PythoscapeNetwork` that we created and not the general environment. If you want to output the entire

network and not representative nodes, use the PythoscapeEnvironment as in the below example.

== Step 9: Output the full node network ==

Change the run.py file to the following and execute it.

```
#Import Pythoscape

import pythoscape.main.environments as env
import pythoscape.interface.local_interface as l_i

#Import plug-ins

import pythoscape.plugin.output.output_xgmml as o_x

#Create interface and environment

my_interface = l_i.LocalInterface(<save directory>)
my_pytho = env.PythoscapeEnvironment(my_interface)

#Create plug-ins

#Output network

out_plugin_2 = o_x.OutputXGMML('gst-sigma-20.xgmml',\
'sigma gst network @ 20',filt_name='-log10(E)',filt_dir='>',filt_value=20)
```

```
#Execute plug-ins
```

```
my_pytho.execute_plugin(out_plugin_2)
```

As before, you must set <save directory>.

In the attachments section on the webpage for the tutorial is a file called `run_template.py`, which is all the code from the entire tutorial. For it to work, you must change all the "<>" arguments to their appropriate values as explained above.

B.3.3. Large network tutorial with MySQL interface

<http://www.rbvi.ucsf.edu/trac/Pythoscape/wiki/MySQLInterfaceSequenceNetwork>

This tutorial uses the MySQL interface to create a representative network of glutathione transferases. You can download the sequence file from the bottom of the tutorial webpage in the "attachments" section. The sequence file has been compressed in .gz format.

This tutorial assumes that you have looked through FrameworkOverview, UsePlugins, PipelineAndPlugins and ScriptingWithObjects to get a basic sense of how Pythoscape works.

To begin this tutorial, Pythoscape should have been installed as instructed in the InstallationInstructions. You must have installed Biopython and MySQLdb before installing Pythoscape as we will use the plug-in !ImportFromUniProt and the MySQL interface. You need not have installed rpy.

As a warning, this tutorial will take a couple days running time even on a server with several processors. If you want a fast tutorial, try the other tutorial that doesn't use the full GST set.

== Step 1: Set up the MySQL database ==

Follow the instructions in CreatePythoscapeMysql to create and format a Pythoscape MySQL database.

== Step 2: Add sequences to network ==

Make a run.py script as below and execute it. You must also set <database name> to the name of your database, <user name> to your MySQL user name and <password> to your MySQL password in the line that creates the MySQL

interface. You must also set <host> if you are logging onto a host not on the machine you are running this command. If you are running it on the same machine, you can leave off this argument.

```
#Import Pythoscape

from pythoscape.auxiliary.re_patterns import RE_PATTERNS

import pythoscape.main.environments as env

import pythoscape.interface.mysql_interface as m_i

import pythoscape.plugin.input.import_sequences as i_s


#Import plug-ins


#Create interface and environment

my_interface = m_i.MySQLInterface(<database name>,\
user=<user name>,passwd=<password>,host=<host>)

my_pytho = env.PythoscapeEnvironment(my_interface)


#Create plug-ins


#Import sequences

plugin_1 = i_s.ImportFromFastaFile('gst.fasta',\

id_re=RE_PATTERNS['Uniprot/Pfam full length'],id_name='Uniprot')
```

```
#Execute plug-ins
```

```
my_pytho.execute_plugin(plugin_1)
```

The name of the fasta sequence file we are going to import is euk_gst.fasta. We will use the UniprotKB regular expression to parse the sequence tags and call those identifiers 'Uniprot'. If you want to make a network with a different sequence file, just use that sequence file in place of the one here. Be aware that if you use a different sequence file, you might have to change the regular expression that is used to parse the sequence headers. For a full list of included regular expressions, see IncludedRegularExpressions.

== Step 3: Output the BLAST run files for a parallel run. ==

Change the run.py file to the following and execute it.

```
#Import Pythoscape
```

```
import pythoscape.main.environments as env
```

```
import pythoscape.interface.mysql_interface as m_i
```

```
import pythoscape.plugin.output.output_table_runs as o_t_r
```

```
#Import plug-ins
```

```
#Create interface and environment
```

```

my_interface = m_i.MySQLInterface(<database name>,\
user=<user name>,passwd=<password>,host=<host>)
my_pytho = env.PythoscapeEnvironment(my_interface)

#Create plug-ins

#Output BLAST runs
plugin_2 = o_t_r.SequenceTableRun('<current location>/runs',50000)

#Execute plug-ins
my_pytho.execute_plugin(plugin_2)

```

As before, you must set <database name>, <user name>, <password> and <host>. Make a directory called 'runs' in your working directory. This plug-in will output the required files to do a parallel blast2seq run into that directory. Replace <current location>/runs with the appropriate value. 50000 is the number of individual blast2seq runs to do on one processor. The value to input here will depend entirely on how many processors you have access to and how big the network is. The best value would be edges to calculate/processors.

For instructions on running edges using parallel resources, see the instructions in [ParallelResources](#).

== Step 4: Add edges to network ==

Change the run.py file to the following and execute it.

```
import pythoscape.main.environments as env
import pythoscape.interface.mysql_interface as m_i
import pythoscape.plugin.input.add_blast_files as a_b_f

#Import plug-ins

#Create interface and environment
my_interface = m_i.MySQLInterface(<database name>,\
user=<user name>,passwd=<password>,host=<host>)
my_pytho = env.PythoscapeEnvironment(my_interface)

#Create plug-ins

#Input edges to Pythoscape
plugin_3 = a_b_f.AddBLASTEdgesFromTableRun('<dir>/runs',\
include_atts=['-log10(E)'])

#Execute plug-ins
```

```
my_pytho.execute_plugin(plugin_3)
```

As before, you must set <database name>, <user name>, <password> and <host>. The above plug-in adds the edges into the interface. Replace '<dir>/runs' with the location of your .out files. The option include_atts is a list of edge attributes that you want to include in the network. The available options are: 'bit_score', 'm*n', 'alignment_len', 'alignment_identities', 'query_start', 'query_end', 'subject_start', 'subject_end', '-ln(E)', '-log10(E)', '% id'.

== Step 5: Add information from !UniProt ==

Change the run.py file to the following and execute it.

```
#Import Pythoscape

import pythoscape.main.environments as env

import pythoscape.interface.mysql_interface as m_i

import pythoscape.plugin.input_bio.add_uniprot_info as a_u_i


#Import plug-ins


#Create interface and environment

my_interface = m_i.MySQLInterface(<database name>,\
user=<user name>,passwd=<password>,host=<host>)
```

```

my_pytho = env.PythoscapeEnvironment(my_interface)

#Create plug-ins

#Input UniProt info to Pythoscape
plugin_4 = a_u_i.ImportFromUniProt(uniprot_id='Uniprot')

#Execute plug-ins
my_pytho.execute_plugin(plugin_4)

```

As before, you must set <database name>, <user name>, <password> and <host>. The above plug-in adds information from Uniprot. It adds the following node attributes (not including taxonomic information): uniprot sequence, organism, description, accessions, family, subfamily, superfamily, pdb, pfam. To add taxonomic information, include the classifiers you want included as a list for taxonomy classifiers. To add taxonomic information, you also need two files from the NCBI taxonomy database, the nodes.dmp file and the names.dmp file. This example does not add taxonomic information. For more information on doing so, see InputPlugins#ImportFromUniProt.

== Step 6: Create a representative node network ==

Change the run.py file to the following and execute it.

```

#Import Pythoscape

import pythoscape.main.environments as env

import pythoscape.interface.mysql_interface as m_i

import pythoscape.plugin.input.make_cdhit_repnodes as m_c_r


#Import plug-ins


#Create interface and environment

my_interface = m_i.MySQLInterface(<database name>,\
user=<user name>,passwd=<password>,host=<host>)

my_pytho = env.PythoscapeEnvironment(my_interface)


#Create plug-ins


#Create representative network

plugin_5 = m_c_r.CreateCDHITRepnodes('rep-net',<cd-hit executable>)


#Execute plug-ins

my_pytho.execute_plugin(plugin_5)

```

As before, you must set <database name>, <user name>, <password> and <host>. The above plug-in creates a representative node network using CDHIT.

Replace <cd-hit executable> with the path of the cdhit executable. As stated in the documentation for the plug-in, this should come from the CDHIT package v4.5.6 or greater. The first two plug-in arguments are mandatory, the name of the network and then the location of the CDHIT executable. In this plug-in, the values of c and n for CDHIT default to 0.4 and 2 respectively. If you want to change them, you must include them in the arguments used to call the plug-in. This plug-in creates a new PythoscapeNetwork called 'rep-net' and an attribute called 'cdhit 0.4', which is the default name of this attribute.

== Step 7: Calculate the size of the representative nodes ==

Change the run.py file to the following and execute it.

```
#Import Pythoscape
import pythoscape.main.environments as env
import pythoscape.interface.mysql_interface as m_i
import pythoscape.plugin.input.repnodel_stats as r_s

#Import plug-ins

#Create interface and environment
my_interface = m_i.MySQLInterface(<database name>,\
user=<user name>,passwd=<password>,host=<host>)
```

```
my_pytho = env.PythoscapeEnvironment(my_interface)
```

```
#Create plug-ins
```

```
#Calculate number of nodes in a representative node
```

```
plugin_6 = r_s.CalcNodeSize('rep-net','cdhit 0.4','node size')
```

```
#Execute plug-ins
```

```
my_pytho.execute_plugin(plugin_6)
```

As before, you must set <database name>, <user name>, <password> and <host>. The above plug-in adds a node attribute to the representative nodes that reflects how many nodes it represents. There are three required arguments, the name of your representative node PythoscapeNetwork, the node attribute that numbers the clusters and the node attribute name you want created to hold this value.

== Step 8: Calculate representative node edges ==

Change the run.py file to the following and execute it.

```
#Import Pythoscape
```

```
import pythoscape.main.environments as env
```

```

import pythoscape.interface.mysql_interface as m_i

import pythoscape.plugin.input.add_repnod_edges as a_r_e


#Import plug-ins


#Create interface and environment

my_interface = m_i.MySQLInterface(<database name>,\
user=<user name>,passwd=<password>,host=<host>)

my_pytho = env.PythoscapeEnvironment(my_interface)


#Create plug-ins


#Add representative edges

plugin_7 = a_r_e.AddEdgesToRepnodNetwork('rep-net',\
'cdhit 0.4','-log10(E)',filt_dir='>',filt_value=0)


#Execute plug-ins

my_pytho.execute_plugin(plugin_7)

```

As before, you must set <database name>, <user name>, <password> and <host>. The above plug-in calculates representative node edges, adds them to the PythoscapeNetwork and adds edge attributes that correspond to those edges. There are three required arguments, the name of your representative

node PythoscapeNetwork, the node attribute that numbers the clusters and the edge attribute value that you want to base these edges on. You can also include a filter if you want by adding `filt_dir = <filter direction>` and `filt_value = <filter value>`. We have added a filter of $-\log_{10}(E) > 0$ because we assume that E-values greater than 1 are not useful and should be ignored.

This plug-in makes three edge attributes for all representative edges called 'rep-net mean', 'rep-net min/max' and 'rep-net total'. The name of the representative node network is added to 'mean', 'min/max' and 'total', so the edge attribute names will change to correspond to the network you create.

Mean is the mean edge, min/max is either the minimum or maximum edge value as you indicate, and total is the number of edges that the edge represents. For more information on this, see the plug-in documentation:

`InputPlugins#AddEdgesToReprnodeNetwork`.

== Step 9: Add representative node attributes ==

Change the run.py file to the following and execute it.

```
#Import Pythoscape
import pythoscape.main.environments as env
import pythoscape.interface.mysql_interface as m_i
```

```

import pythoscape.plugin.input.add_repnod_atts as a_r_a

#Import plug-ins

#Create interface and environment

my_interface = m_i.MySQLInterface(<database name>,\
user=<user name>,passwd=<password>,host=<host>)

my_pytho = env.PythoscapeEnvironment(my_interface)

#Create plug-ins

#Create representative node attributes

plugin_8 = a_r_a.AddAttributesByIfAny('rep-net','cdhit 0.4','family',\
'repnod family')

#Execute plug-ins

my_pytho.execute_plugin(plugin_8)

```

As before, you must set <database name>, <user name>, <password> and <host>.

The above plug-in calculates attributes for representative nodes. It uses 'if/any' voting, which means that it concatenates a list of all the possible options that a node attribute can be for all members of the group.

There are four required arguments, the name of your representative node PythoscapeNetwork, the node attribute that numbers the clusters, the node attribute value that you want to base the new representative value on and the new name of the representative attribute.

== Step 10: Output the representative node network ==

Change the run.py file to the following and execute it.

```
#Import Pythoscape

import pythoscape.main.environments as env

import pythoscape.interface.mysql_interface as m_i

import pythoscape.plugin.output.output_xgmml as o_x


#Import plug-ins


#Create interface and environment

my_interface = m_i.MySQLInterface(<database name>,\
user=<user name>,passwd=<password>,host=<host>)
```

```

my_pytho = env.PythoscapeEnvironment(my_interface)
my_net = env.PythoscapeNetwork('rep-net',my_interface)

#Create plug-ins

#Output network
out_plugin = o_x.OutputXGMML('gst-rep-net-13.xgmml',\
'gst representative node network',filt_name='rep-net\
mean',filt_dir='>',filt_value=13)

#Execute plug-ins
my_net.execute_plugin(out_plugin)

```

As before, you must set <database name>, <user name>, <password> and <host>.

The above plug-in outputs the network in XGMML format so that you can view it in Cytoscape. For more information on visualizing networks in Cytoscape, see <http://www.cytoscape.org>. The first two arguments are required. The first is that name of the output file and the second is the title of the network. You can also add an edge filter by filling in the values of filt_name, filt_dir and filt_value as appropriate.

Notice that we are going to run this plug-in on the new PythoscapeNetwork that we created and not the general environment. If you want to output the entire network and not representative nodes, use the PythoscapeEnvironment.

Appendix C. Pythoscape source code

C.1. Introduction

This section contains the source code for all the relevant Python modules for release 1.01b of Pythoscape. The organization of this section follows the organization of the downloaded code and is introduced below. Code that is required for Pythoscape to run and provided in the package, but was not an original creation of the author is not included here. Also, `__init__.py` files were omitted.

There are 3 major directories within the Pythoscape source root directory:

- 1) `mysql_resources` | contains sql source for use with the `MysqlInterface`
- 2) `parallel_resources` | contains python code for large, parallel runs
- 3) `src` | contains the installed Pythoscape modules

There are 9 major packages in the `root/src/pythoscape`, which are as follows:

- 1) `auxiliary` | auxiliary tools, functions and classes for use in Pythoscape
- 2) `interface` | storage interface classes
- 3) `main` | the main classes and functions to run Pythoscape
- 4) `plugin/delete` | plug-ins that delete data
- 5) `plugin/input` | plug-ins that input data
- 6) `plugin/input_bio` | plug-ins that input data and require BioPython
- 7) `plugin/output` | plug-ins that output data
- 8) `plugin/output_bio` | plug-ins that output data and require BioPython
- 9) `plugin/output_r` | plug-ins that output data and require R and rpy

C.2. ./ (root) modules

./setup.py

```
1  # Author: Alan Barber
2  # Created on: March 17, 2012
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: The installation file for Pythoscape.
7  '''
8
9  from sys import version
10 from distutils.core import setup
11
12 def try_import(name):
13     '''Tries to import a module. If it can't, returns
   false.
14
15     Arguments:
16         name - module name
17
18     returns:
19         module name or False
20     '''
21     try:
22         return __import__(name)
23     except ImportError:
24         return False
25
26 if version < '2.6':
27     print 'Pythoscape is not compatible with versions of
   Python earlier than 2.6.'
28 elif version >= '3.0':
29     print 'Pythoscape is not yet compatible with Python
   3.0+.'
30 else:
31     default_modules = [
32         'pythoscape.interface.local_interface',
33         'pythoscape.interface.standard_interface',
34         'pythoscape.plugin.__init__']
35     default_packages = ['pythoscape',
36                         'pythoscape.auxiliary',
37                         'pythoscape.main',
38                         'pythoscape.plugin.input',
39                         'pythoscape.plugin.output']
```

./setup.py

```
39     if try_import('rpy'):
40         default_packages.append(
41             'pythoscape.plugin.output_r')
42     if try_import('Bio'):
43         default_packages += [
44             'pythoscape.plugin.output_bio',
45             'pythoscape.plugin.input_bio']
46     if try_import('MySQLdb'):
47         default_modules.append(
48             'pythoscape.interface.mysql_interface')
49     setup(name='pythoscape',
50           version='1.0',
51           description='A framework for creating and
52 managing protein similarity networks',
53           author='Alan E. Barber II',
54           author_email='alan.barber@ucsf.edu',
55           url='http://plato.cgl.ucsf.edu/trac/Pythoscape',
56           package_dir={'': 'src'},
57           packages=default_packages,
58           py_modules=default_modules)
```

C.3. ./mysql_resources code

./mysql_resources/mysql_template.sql

```
1  -- Author: Alan Barber
2  -- This file is for use in generating the tables, views
   and stored procedures necessary for the MySQL interface.
3  -- Usage: mysql [database] < mysql_template.sql
4  -- Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
5
6  -- -----
7  -- Tables
8  -- -----
9
10 /*!40101 SET @
   OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
11 /*!40101 SET @
   OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
12 /*!40101 SET @
   OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
13 /*!40101 SET NAMES utf8 */;
14 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
15 /*!40103 SET TIME_ZONE='+00:00' */;
16 /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
   UNIQUE_CHECKS=0 */;
17 /*!40014 SET @
   OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
   FOREIGN_KEY_CHECKS=0 */;
18 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
   SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
19 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
20
21 --
22 -- Table structure for table `attribute`
23 --
24
25 DROP TABLE IF EXISTS `attribute`;
26 SET @saved_cs_client      = @@character_set_client;
27 SET character_set_client  = utf8;
28 CREATE TABLE `attribute` (
29   `attribute_id` int(10) unsigned NOT NULL,
30   `object_id` int(10) unsigned NOT NULL,
31   `attribute_blob` blob,
32   UNIQUE KEY `attribute_id` (`attribute_id`,`object_id`)
33 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
34 SET character_set_client  = @saved_cs_client;
35
```

./mysql_resources/mysql_template.sql

```
36  --
37  -- Dumping data for table `attribute`
38  --
39
40  LOCK TABLES `attribute` WRITE;
41  /*!40000 ALTER TABLE `attribute` DISABLE KEYS */;
42  /*!40000 ALTER TABLE `attribute` ENABLE KEYS */;
43  UNLOCK TABLES;
44
45  --
46  -- Table structure for table `attribute_description`
47  --
48
49  DROP TABLE IF EXISTS `attribute_description`;
50  SET @saved_cs_client      = @@character_set_client;
51  SET character_set_client = utf8;
52  CREATE TABLE `attribute_description` (
53    `attribute_id` int(10) unsigned NOT NULL auto_increment,
54    `attribute_name` varchar(100) NOT NULL,
55    `attribute_object` enum('node','edge','network',
56    'identifier') NOT NULL,
57    `data_type` enum('int','string','double','bool') NOT
58    NULL,
59    PRIMARY KEY (`attribute_id`),
60    UNIQUE KEY `attribute_object` (`attribute_object`,`
61    attribute_name`)
62  ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
63  SET character_set_client = @saved_cs_client;
64
65  --
66  -- Dumping data for table `attribute_description`
67  --
68
69  LOCK TABLES `attribute_description` WRITE;
70  /*!40000 ALTER TABLE `attribute_description` DISABLE
71  KEYS */;
72  /*!40000 ALTER TABLE `attribute_description` ENABLE KEYS
73  */;
74  UNLOCK TABLES;
```


./mysql_resources/mysql_template.sql

```
75 DROP TABLE IF EXISTS `edge`;
76 SET @saved_cs_client      = @@character_set_client;
77 SET character_set_client = utf8;
78 CREATE TABLE `edge` (
79   `edge_id` int(10) unsigned NOT NULL auto_increment,
80   `source_node_id` int(10) unsigned NOT NULL,
81   `target_node_id` int(10) unsigned NOT NULL,
82   PRIMARY KEY (`edge_id`),
83   UNIQUE KEY `source_node_id` (`source_node_id`,`
target_node_id`)
84 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
85 SET character_set_client = @saved_cs_client;
86
87 --
88 -- Dumping data for table `edge`
89 --
90
91 LOCK TABLES `edge` WRITE;
92 /*!40000 ALTER TABLE `edge` DISABLE KEYS */;
93 /*!40000 ALTER TABLE `edge` ENABLE KEYS */;
94 UNLOCK TABLES;
95
96 --
97 -- Table structure for table `edge2network`
98 --
99
100 DROP TABLE IF EXISTS `edge2network`;
101 SET @saved_cs_client      = @@character_set_client;
102 SET character_set_client = utf8;
103 CREATE TABLE `edge2network` (
104   `edge_id` int(10) unsigned NOT NULL,
105   `network_id` int(10) unsigned NOT NULL,
106   UNIQUE KEY `edge_id` (`edge_id`,`network_id`)
107 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
108 SET character_set_client = @saved_cs_client;
109
110 CREATE UNIQUE INDEX e2n_reverse_index ON edge2network (
network_id,edge_id);
111
112 --
113 -- Dumping data for table `edge2network`
114 --
115
116 LOCK TABLES `edge2network` WRITE;
```

./mysql_resources/mysql_template.sql

```
117  /*!40000 ALTER TABLE `edge2network` DISABLE KEYS */;
118  /*!40000 ALTER TABLE `edge2network` ENABLE KEYS */;
119  UNLOCK TABLES;
120
121  --
122  -- Table structure for table `identifier`
123  --
124
125  DROP TABLE IF EXISTS `identifier`;
126  SET @saved_cs_client      = @@character_set_client;
127  SET character_set_client = utf8;
128  CREATE TABLE `identifier` (
129    `attribute_id` int(10) unsigned NOT NULL,
130    `identifier_value` varchar(100) NOT NULL,
131    `node_id` int(10) unsigned NOT NULL,
132    UNIQUE KEY `attribute_id` (`attribute_id`,`
133    identifier_value`)
134  ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
135  SET character_set_client = @saved_cs_client;
136
137  CREATE UNIQUE INDEX identifier_node_index ON identifier (
138  node_id,attribute_id,identifier_value);
139
140  --
141  -- Dumping data for table `identifier`
142  --
143
144  LOCK TABLES `identifier` WRITE;
145  /*!40000 ALTER TABLE `identifier` DISABLE KEYS */;
146  /*!40000 ALTER TABLE `identifier` ENABLE KEYS */;
147  UNLOCK TABLES;
148
149  --
150  -- Table structure for table `network`
151  --
152
153  DROP TABLE IF EXISTS `network`;
154  SET @saved_cs_client      = @@character_set_client;
155  SET character_set_client = utf8;
156  CREATE TABLE `network` (
157    `network_id` int(10) unsigned NOT NULL auto_increment,
158    `network_name` varchar(12) NOT NULL,
159    `kind` varchar(12) default NULL,
160    PRIMARY KEY (`network_id`),
```

./mysql_resources/mysql_template.sql

```
159     UNIQUE KEY `network_name` (`network_name`)
160 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
161 SET character_set_client = @saved_cs_client;
162
163 --
164 -- Dumping data for table `network`
165 --
166
167 LOCK TABLES `network` WRITE;
168 /*!40000 ALTER TABLE `network` DISABLE KEYS */;
169 /*!40000 ALTER TABLE `network` ENABLE KEYS */;
170 UNLOCK TABLES;
171
172 --
173 -- Table structure for table `node`
174 --
175
176 DROP TABLE IF EXISTS `node`;
177 SET @saved_cs_client      = @@character_set_client;
178 SET character_set_client = utf8;
179 CREATE TABLE `node` (
180     `node_id` int(10) unsigned NOT NULL auto_increment,
181     `node_hash` varchar(75) NOT NULL,
182     PRIMARY KEY (`node_id`),
183     UNIQUE KEY `node_hash` (`node_hash`)
184 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
185 SET character_set_client = @saved_cs_client;
186
187 --
188 -- Dumping data for table `node`
189 --
190
191 LOCK TABLES `node` WRITE;
192 /*!40000 ALTER TABLE `node` DISABLE KEYS */;
193 /*!40000 ALTER TABLE `node` ENABLE KEYS */;
194 UNLOCK TABLES;
195
196 --
197 -- Table structure for table `node2network`
198 --
199
200 DROP TABLE IF EXISTS `node2network`;
201 SET @saved_cs_client      = @@character_set_client;
202 SET character_set_client = utf8;
```

./mysql_resources/mysql_template.sql

```
203 CREATE TABLE `node2network` (  
204     `node_id` int(10) unsigned NOT NULL,  
205     `network_id` int(10) unsigned NOT NULL,  
206     UNIQUE KEY `node_id` (`node_id`,`network_id`)  
207 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;  
208 SET character_set_client = @saved_cs_client;  
209  
210 CREATE UNIQUE INDEX n2n_reverse_index ON node2network (  
    network_id,node_id);  
211  
212 --  
213 -- Dumping data for table `node2network`  
214 --  
215  
216 LOCK TABLES `node2network` WRITE;  
217 /*!40000 ALTER TABLE `node2network` DISABLE KEYS */;  
218 /*!40000 ALTER TABLE `node2network` ENABLE KEYS */;  
219 UNLOCK TABLES;  
220 /*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;  
221  
222 /*!40101 SET SQL_MODE=@OLD_SQL_MODE */;  
223 /*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS  
    */;  
224 /*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;  
225 /*!40101 SET  
    CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;  
226 /*!40101 SET  
    CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;  
227 /*!40101 SET  
    COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;  
228 /*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;  
229  
230 -- Author: Alan Barber  
231 -- This file only updates the views and stored  
    procedures that are used by the MySQL interface.  
232 -- The reason for a separate script to do this is for  
    bug and script updates. This code can  
233 -- update the the underlying functions without changing  
    or dropping tables with existing data.  
234 -- Usage: mysql [database] < stored_procs.sql  
235  
236 --  
-----  
-----
```

./mysql_resources/mysql_template.sql

```
237  -- Views
238  --
239  -----
240
241  DROP VIEW IF EXISTS `network_attribute`;
242  CREATE VIEW network_attribute AS
243  SELECT n.network_name,n.kind as network_type,a_d.
244  attribute_name,a_d.data_type,a.attribute_blob as
245  attribute_value,
246  n.network_id,a_d.attribute_id
247  FROM attribute AS a, attribute_description AS a_d,
248  network AS n
249  WHERE a.attribute_id = a_d.attribute_id AND n.network_id
250  = a.object_id
251  AND a_d.attribute_object = "network";
252
253  DROP VIEW IF EXISTS `node_identifier`;
254  CREATE VIEW node_identifier AS
255  SELECT nd.node_hash as id,a_d.attribute_name as
256  identifier_type,i.identifier_value as identifier_value,nd
257  .node_id
258  FROM node AS nd
259  LEFT JOIN (identifier i, attribute_description a_d)
260  ON (i.node_id = nd.node_id AND a_d.attribute_id = i.
261  attribute_id AND a_d.attribute_object = "identifier");
262
263  DROP VIEW IF EXISTS `node_attribute`;
264  CREATE VIEW node_attribute AS
265  SELECT n.node_hash as id,a_d.attribute_name,a_d.data_type
266  ,a.attribute_blob as attribute_value,
267  n.node_id,a_d.attribute_id
268  FROM attribute AS a, attribute_description AS a_d, node
269  AS n
270  WHERE a.attribute_id = a_d.attribute_id AND n.node_id = a
271  .object_id
272  AND a_d.attribute_object = "node";
273
274  DROP VIEW IF EXISTS `edge_filter`;
275  CREATE VIEW edge_filter AS
276  SELECT e.edge_id,n1.node_hash as source_node,n2.node_hash
277  as target_node,
278  a_d.attribute_name as algorithm, a.attribute_blob as
279  weight_value
```

./mysql_resources/mysql_template.sql

```
267 FROM edge AS e, attribute AS a, attribute_description AS
    a_d, node AS n1, node AS n2
268 WHERE a.object_id = e.edge_id AND a_d.attribute_id = a.
    attribute_id AND a_d.attribute_object = "edge"
269 AND n1.node_id = e.source_node_id AND n2.node_id = e.
    target_node_id;
270
271 DROP VIEW IF EXISTS `edge_filter_network`;
272 CREATE VIEW edge_filter_network AS
273 SELECT e.edge_id,n1.node_hash as source_node,n2.node_hash
    as target_node,
274 a_d.attribute_name as algorithm, a.attribute_blob as
    weight_value,
275 net.network_name,net.kind as network_type
276 FROM edge AS e, attribute AS a, attribute_description AS
    a_d,
277 node AS n1, node AS n2, network AS net, edge2network AS
    en
278 WHERE a.object_id = e.edge_id AND a_d.attribute_id = a.
    attribute_id AND a_d.attribute_object = "edge"
279 AND n1.node_id = e.source_node_id AND n2.node_id = e.
    target_node_id
280 AND e.edge_id = en.edge_id AND net.network_id = en.
    network_id;
281
282 DROP VIEW IF EXISTS `edge_nofilter`;
283 CREATE VIEW edge_nofilter AS
284 SELECT e.edge_id,n1.node_hash as source_node,n2.node_hash
    as target_node
285 FROM edge AS e, node AS n1, node AS n2
286 WHERE n1.node_id = e.source_node_id AND n2.node_id = e.
    target_node_id;
287
288 DROP VIEW IF EXISTS `edge_nofilter_network`;
289 CREATE VIEW edge_nofilter_network AS
290 SELECT e.edge_id,n1.node_hash as source_node,n2.node_hash
    as target_node,
291 net.network_name,net.kind as network_type
292 FROM edge AS e, node AS n1, node AS n2, network AS net,
    edge2network AS en
293 WHERE n1.node_id = e.source_node_id AND n2.node_id = e.
    target_node_id
294 AND e.edge_id = en.edge_id AND net.network_id = en.
    network_id;
```

./mysql_resources/mysql_template.sql

```
295
296  --
      -----
      -----
297  -- Stored procedures
298  --
      -----
      -----
299
300  DELIMITER //
301
302  DROP PROCEDURE IF EXISTS `AddNode` //
303  CREATE PROCEDURE AddNode (
304  IN nodehash VARCHAR(75)
305  )
306  interior:BEGIN
307
308  IF nodehash IS NULL
309  THEN LEAVE interior;
310  END IF;
311
312  SELECT * FROM node WHERE node_hash = nodehash;
313  IF FOUND_ROWS() = 0
314  THEN INSERT INTO node (node_hash) VALUES (nodehash);
315  END IF;
316  END //
317
318  DROP PROCEDURE IF EXISTS `AddNetwork` //
319  CREATE PROCEDURE AddNetwork (
320  IN netname VARCHAR(20),
321  IN netkind VARCHAR(20)
322  )
323  interior:BEGIN
324
325  IF netname IS NULL
326  THEN LEAVE interior;
327  END IF;
328
329  SELECT * FROM network WHERE network_name = netname;
330  IF FOUND_ROWS() = 0
331  THEN INSERT INTO network (network_name,kind) VALUES (
332  netname,netkind);
333  ELSE
334  UPDATE network SET kind = netkind WHERE network_name =
```

./mysql_resources/mysql_template.sql

```
netname;
334 END IF;
335 END //
336
337 DROP PROCEDURE IF EXISTS `AddNodeToNetwork` //
338 CREATE PROCEDURE AddNodeToNetwork(
339 IN netname VARCHAR(20),
340 IN nodehash VARCHAR(75)
341 )
342 BEGIN
343
344 SET @netid = NULL;
345 SET @nodeid = NULL;
346 SET @netflag = NULL;
347 SELECT node_id FROM node WHERE node_hash = nodehash LIMIT
    1 INTO @nodeid;
348 SELECT network_id FROM network WHERE network_name =
    netname LIMIT 1 INTO @netid;
349
350 IF @netid IS NOT NULL AND @nodeid IS NOT NULL
351 THEN SELECT node_id FROM node2network WHERE network_id =
    @netid AND node_id = @nodeid LIMIT 1 INTO @netflag;
352 END IF;
353
354 IF @netid IS NOT NULL AND @nodeid IS NOT NULL AND @
    netflag IS NULL
355 THEN INSERT INTO node2network (network_id,node_id) VALUES
    (@netid,@nodeid);
356 END IF;
357
358 END //
359
360 DROP PROCEDURE IF EXISTS `AddEdge` //
361 CREATE PROCEDURE AddEdge (
362 IN sourcenode VARCHAR(75),
363 IN targetnode VARCHAR(75)
364 )
365 interior:BEGIN
366
367 SET @sourcenodeid = NULL;
368 SET @targetnodeid = NULL;
369 SELECT node_id FROM node WHERE node_hash = sourcenode
    LIMIT 1 INTO @sourcenodeid;
370 SELECT node_id FROM node WHERE node_hash = targetnode
```


./mysql_resources/mysql_template.sql

```

LIMIT 1 INTO @targetnodeid;
371
372 IF @sourcenodeid IS NULL OR @targetnodeid IS NULL
373 THEN LEAVE interior;
374 END IF;
375
376 SELECT * FROM edge WHERE source_node_id = @sourcenodeid
AND target_node_id = @targetnodeid;
377 IF FOUND_ROWS() = 0
378 THEN INSERT INTO edge (source_node_id,target_node_id)
VALUES (@sourcenodeid,@targetnodeid);
379 END IF;
380 END //
381
382 DROP PROCEDURE IF EXISTS `AddEdgeToNetwork` //
383 CREATE PROCEDURE AddEdgeToNetwork(
384 IN netname VARCHAR(20),
385 IN sourcenode VARCHAR(75),
386 IN targetnode VARCHAR(75)
387 )
388 BEGIN
389
390 SET @netid = NULL;
391 SET @edgeid = NULL;
392 SET @netflag = NULL;
393 SELECT e.edge_id
394 FROM edge AS e, node AS n1, node AS n2
395 WHERE n2.node_id = e.target_node_id AND n1.node_id = e.
source_node_id
396 AND n1.node_hash = sourcenode AND n2.node_hash =
targetnode
397 LIMIT 1
398 INTO @edgeid;
399 SELECT network_id FROM network WHERE network_name =
netname LIMIT 1 INTO @netid;
400
401 SELECT @edgeid;
402 SELECT @netid;
403
404 IF @netid IS NOT NULL AND @edgeid IS NOT NULL
405 THEN SELECT edge_id FROM edge2network WHERE network_id =
@netid AND edge_id = @edgeid LIMIT 1 INTO @netflag;
406 END IF;
407
```

./mysql_resources/mysql_template.sql

```
408 IF @netid IS NOT NULL AND @edgeid IS NOT NULL AND @
    netflag IS NULL
409 THEN INSERT INTO edge2network (network_id,edge_id) VALUES
    (@netid,@edgeid);
410 END IF;
411
412 END //
```

```
413
414 DROP PROCEDURE IF EXISTS `AddAttribute` //
415 CREATE PROCEDURE AddAttribute(
416 IN netname VARCHAR(12),
417 IN sourcenode VARCHAR(75),
418 IN targetnode VARCHAR(75),
419 IN attributename VARCHAR(100),
420 IN attobject ENUM('network','node','edge'),
421 IN attdatatype ENUM('double','int','string','bool'),
422 IN attributeblob BLOB
423 )
424 interior:BEGIN
425
426 SET @attributeid = NULL;
427 SET @objectid = NULL;
428 SELECT attribute_id FROM attribute_description WHERE
    attribute_name = attributename AND attribute_object =
    attobject LIMIT 1 INTO @attributeid;
429
430 IF attobject = 'network' AND netname IS NOT NULL
431 THEN SELECT network_id FROM network WHERE network_name =
    netname LIMIT 1 INTO @objectid;
432 ELSEIF attobject = 'node' AND sourcenode IS NOT NULL
433 THEN SELECT node_id FROM node WHERE node_hash =
    sourcenode LIMIT 1 INTO @objectid;
434 ELSEIF attobject = 'edge' AND sourcenode IS NOT NULL and
    targetnode IS NOT NULL
435 THEN SELECT e.edge_id
436 FROM edge AS e, node AS n1, node AS n2
437 WHERE n2.node_id = e.target_node_id AND n1.node_id = e.
    source_node_id
438 AND n1.node_hash = sourcenode AND n2.node_hash =
    targetnode
439 LIMIT 1
440 INTO @objectid;
441 END IF;
442
```

./mysql_resources/mysql_template.sql

```
443 IF @objectid IS NULL
444 THEN LEAVE interior;
445 END IF;
446
447 IF @attributeid IS NULL
448 THEN INSERT INTO attribute_description (attribute_name,
attribute_object,data_type) VALUES (attributename,
attobject,attdatatype);
449 SET @attributeid = LAST_INSERT_ID();
450 END IF;
451
452 SELECT * FROM attribute WHERE attribute_id = @
attributeid AND object_id = @objectid;
453 IF FOUND_ROWS() = 0
454 THEN INSERT INTO attribute (attribute_id,object_id,
attribute_blob) VALUES (@attributeid,@objectid,
attributeblob);
455 ELSE
456 UPDATE attribute SET attribute_blob = attributeblob WHERE
attribute_id = @attributeid AND object_id = @objectid;
457 END IF;
458 END //
459
460 DROP PROCEDURE IF EXISTS `AddIdentifier` //
461 CREATE PROCEDURE AddIdentifier(
462 IN nodehash VARCHAR(75),
463 IN attributename VARCHAR(100),
464 IN identiervalue VARCHAR(100)
465 )
466 interior:BEGIN
467
468 SET @attributeid = NULL;
469 SET @nodeid = NULL;
470 SELECT node_id FROM node WHERE node_hash = nodehash LIMIT
1 INTO @nodeid;
471
472 IF @nodeid IS NULL
473 THEN LEAVE interior;
474 END IF;
475
476 SELECT attribute_id FROM attribute_description WHERE
attribute_name = attributename AND attribute_object =
'identifier' LIMIT 1 INTO @attributeid;
477
```

./mysql_resources/mysql_template.sql

```
478 IF @attributeid IS NULL
479 THEN INSERT INTO attribute_description (attribute_name,
attribute_object,data_type) VALUES (attributename,
'identifier','string');
480 SET @attributeid = LAST_INSERT_ID();
481 END IF;
482
483 SELECT * FROM identifier WHERE attribute_id = @
attributeid AND identifier_value = identiervalue;
484 IF FOUND_ROWS() = 0
485 THEN INSERT INTO identifier (attribute_id,
identifier_value,node_id) VALUES (@attributeid,
identiervalue,@nodeid);
486 ELSE
487 UPDATE identifier SET node_id = @nodeid WHERE
attribute_id = @attributeid AND identifier_value =
identiervalue;
488 END IF;
489 END //
490
491 DROP PROCEDURE IF EXISTS `SelectNetwork` //
492 CREATE PROCEDURE SelectNetwork(
493 IN netname VARCHAR(20),
494 IN netkind VARCHAR(20)
495 )
496 BEGIN
497
498 SET @where_stmt = CONCAT('network_name = "',netname,'"');
499 SET @where_stmt_2 = CONCAT('kind = "',netkind,'"');
500
501 IF @where_stmt IS NULL
502 THEN SET @where_stmt = '1 = 1';
503 END IF;
504
505 IF @where_stmt_2 IS NULL
506 THEN SET @where_stmt_2 = '1 = 1';
507 END IF;
508
509 SET @select_stmt = CONCAT_WS(' ','SELECT
network_name,kind as network_type FROM network WHERE',@
where_stmt,'AND',@where_stmt_2);
510 PREPARE exec_stmt FROM @select_stmt;
511 EXECUTE exec_stmt;
512
```

./mysql_resources/mysql_template.sql

```
513 END //
514
515 DROP PROCEDURE IF EXISTS `SelectNetworkAttribute` //
516 CREATE PROCEDURE SelectNetworkAttribute(
517     IN netname VARCHAR(20),
518     IN attributename VARCHAR(100)
519 )
520 BEGIN
521
522     IF attributename IS NOT NULL
523     THEN SET @where_stmt = CONCAT('attribute_name = "',
524                                     attributename, '"');
525     ELSE SET @where_stmt = '1 = 1';
526     END IF;
527
528     IF netname IS NOT NULL
529     THEN SET @where_stmt_2 = CONCAT('network_name = "',
530                                     netname, '"');
531     ELSE SET @where_stmt_2 = '1 = 1';
532     END IF;
533
534     SET @select_stmt = CONCAT_WS(' ', 'SELECT
535                                     network_name, network_type, attribute_name, data_type, attribu
536                                     te_value
537     FROM network_attribute WHERE', @where_stmt, 'AND', @
538     where_stmt_2);
539
540     PREPARE exec_stmt FROM @select_stmt;
541     EXECUTE exec_stmt;
542
543 END //
544
545 DROP PROCEDURE IF EXISTS `SelectNodeInNetwork` //
546 CREATE PROCEDURE SelectNodeInNetwork(
547     IN netname VARCHAR(12),
548     IN nodehash VARCHAR(75),
549     IN attributename VARCHAR(100),
550     IN identiervalue VARCHAR(100)
551 )
552 interior:BEGIN
553
554     SET @nodeid = NULL;
555     IF nodehash IS NOT NULL
556     THEN SELECT node_id FROM node WHERE node_hash = nodehash
557     LIMIT 1 INTO @nodeid;
```

./mysql_resources/mysql_template.sql

```
551 END IF;
552
553 SET @nodeid2 = NULL;
554 IF attributename IS NOT NULL AND identiervalue IS NOT
NULL
555 THEN SELECT i.node_id FROM identifier as i,
attribute_description as a_d
556 WHERE a_d.attribute_id = i.attribute_id AND a_d.
attribute_object = 'identifier'
557 AND a_d.attribute_name = attributename AND i.
identifier_value = identiervalue
558 LIMIT 1 INTO @nodeid2;
559 END IF;
560
561 IF @nodeid IS NOT NULL AND @nodeid2 IS NOT NULL AND @
nodeid != @nodeid2
562 THEN LEAVE interior;
563 ELSEIF @nodeid IS NOT NULL AND @nodeid2 IS NOT NULL AND @
nodeid = @nodeid2
564 THEN SET @nodeid2 = NULL;
565 END IF;
566
567 IF @nodeid IS NOT NULL
568 THEN SET @where_stmt = CONCAT('node_id = ',@nodeid);
569 ELSE SET @where_stmt = '1 = 1';
570 END IF;
571
572 IF @nodeid2 IS NOT NULL
573 THEN SET @where_stmt_2 = CONCAT('node_id = ',@nodeid2);
574 ELSE SET @where_stmt_2 = '1 = 1';
575 END IF;
576
577 IF netname IS NOT NULL
578 THEN SET @net_stmt = CONCAT('JOIN (node2network nn,
network net)
579 ON (n_i.node_id = nn.node_id AND net.network_id =
nn.network_id AND net.network_name = "',netname,'"');
580 ELSE SET @net_stmt = 'LEFT JOIN (node2network nn,
network net)
581 ON (n_i.node_id = nn.node_id AND net.network_id =
nn.network_id)';
582 END IF;
583
584 SET @select_stmt = CONCAT_WS(' ', 'SELECT
```

./mysql_resources/mysql_template.sql

```
net.network_name,net.kind as
network_type,n_i.id,n_i.identifier_type,n_i.identifier_val
ue
585 FROM (SELECT * FROM node_identifier WHERE',
586 @where_stmt,'AND',@where_stmt_2,') as n_i',@net_stmt);
587 PREPARE exec_stmt FROM @select_stmt;
588 EXECUTE exec_stmt;
589
590 END //
591
592 DROP PROCEDURE IF EXISTS `SelectNodeAttributeInNetwork` //
593 CREATE PROCEDURE SelectNodeAttributeInNetwork(
594 IN netname VARCHAR(12),
595 IN nodehash VARCHAR(75),
596 IN attributename VARCHAR(100)
597 )
598 BEGIN
599
600 IF attributename IS NOT NULL
601 THEN SET @where_stmt = CONCAT('n_a.attribute_name = "',
attributename,'"');
602 ELSE SET @where_stmt = '1 = 1';
603 END IF;
604
605 IF nodehash IS NOT NULL
606 THEN SET @where_stmt_2 = CONCAT('n_a.id = "',nodehash,'"
');
607 ELSE SET @where_stmt_2 = '1 = 1';
608 END IF;
609
610 SET @select_stmt = NULL;
611
612 IF netname IS NULL
613 THEN SET @select_stmt = CONCAT_WS(' ','SELECT
id,attribute_name,data_type,attribute_value
614 FROM node_attribute AS n_a WHERE',@where_stmt,'AND',@
where_stmt_2);
615 ELSE SET @select_stmt = CONCAT('SELECT
n_a.id,n_a.attribute_name,n_a.data_type,n_a.attribute_valu
e
616 FROM node_attribute AS n_a, node2network AS nn, network
AS net
617 WHERE n_a.node_id = nn.node_id AND net.network_id =
nn.network_id AND
```

./mysql_resources/mysql_template.sql

```
618 net.network_name = '',netname,'' AND ',@where_stmt,' AND
    ',@where_stmt_2);
619 END IF;
620
621 IF @select_stmt IS NOT NULL THEN
622     PREPARE exec_stmt FROM @select_stmt;
623     EXECUTE exec_stmt;
624 END IF;
625
626 END //
```

```
627
628 DROP PROCEDURE IF EXISTS `SelectEdgeInNetwork` //
```

```
629 CREATE PROCEDURE SelectEdgeInNetwork(
630     IN netname VARCHAR(12),
631     IN sourcenode VARCHAR(75),
632     IN targetnode VARCHAR(75),
633     IN edgeweight DOUBLE,
634     IN weightdirection ENUM('<','>'),
635     IN edgeweightatt VARCHAR(100)
636 )
637 interior:BEGIN
638
639     IF sourcenode IS NOT NULL
640     THEN SET @where_stmt = CONCAT('source_node = ',
        sourcenode, '');
641     ELSE SET @where_stmt = '1 = 1';
642     END IF;
643
644     IF targetnode IS NOT NULL
645     THEN SET @where_stmt_2 = CONCAT('target_node = ',
        targetnode, '');
646     ELSE SET @where_stmt_2 = '1 = 1';
647     END IF;
648
649     IF edgeweight IS NULL OR weightdirection IS NULL OR
        edgeweightatt IS NULL
650     THEN SET @filter_flag = 0;
651     ELSE SET @filter_flag = 1;
652     END IF;
653
654     IF @filter_flag = 0 THEN
655         IF netname IS NULL
656             THEN SET @select_stmt = CONCAT_WS(' ','SELECT
        net.network_name,net.kind as network_type,
```


./mysql_resources/mysql_template.sql

```
657         e_f.source_node,e_f.target_node
658         FROM (SELECT * FROM edge_nofilter WHERE',
659         @where_stmt,'AND',@where_stmt_2,') as e_f LEFT
JOIN (edge2network en, network net)
660         ON (e_f.edge_id = en.edge_id AND net.network_id
= en.network_id)');
661     ELSE
662         SET @select_stmt = CONCAT('SELECT
network_name,network_type,
663         source_node,target_node
664         FROM edge_nofilter_network WHERE ',
665         @where_stmt,' AND ',@where_stmt_2,' AND
network_name = "',netname,'"');
666     END IF;
667     PREPARE exec_stmt FROM @select_stmt;
668     EXECUTE exec_stmt;
669     LEAVE interior;
670     END IF;
671
672     SET @attributetype = NULL;
673     SELECT data_type FROM attribute_description WHERE
attribute_name = edgeweightatt AND attribute_object =
'edge' LIMIT 1 INTO @attributetype;
674     IF @attributetype IS NULL OR @attributetype = 'string'
675     THEN LEAVE interior;
676     END IF;
677
678     SET @where_stmt_3 = CONCAT('algorithm = "',edgeweightatt,
'");
679     SET @where_stmt_4 = CONCAT('weight_value ',
weightdirection,' ',edgeweight);
680
681     IF netname IS NULL
682     THEN SET @select_stmt = CONCAT_WS(' ','SELECT
net.network_name,net.kind as network_type,
683     e_f.source_node,e_f.target_node,e_f.algorithm,e_f.weight_v
alue
684     FROM (SELECT * FROM edge_filter WHERE',@where_stmt,'AND',
@where_stmt_2,'AND',
685     @where_stmt_3,'AND',@where_stmt_4,') as e_f LEFT JOIN
(edge2network en, network net)
686     ON (e_f.edge_id = en.edge_id AND net.network_id =
en.network_id)');
687     ELSE
```

./mysql_resources/mysql_template.sql

```
688 SET @select_stmt = CONCAT('SELECT
network_name,network_type,
689 source_node,target_node,algorithm,weight_value
690 FROM edge_filter_network WHERE ',@where_stmt,' AND ',@
where_stmt_2,' AND ',
691 @where_stmt_3,' AND ',@where_stmt_4,' AND network_name =
'',netname,'');
692 END IF;
693
694 PREPARE exec_stmt FROM @select_stmt;
695 EXECUTE exec_stmt;
696 END //
697
698 DROP PROCEDURE IF EXISTS `SelectEdgeAttributeInNetwork` //
699 CREATE PROCEDURE SelectEdgeAttributeInNetwork(
700 IN netname VARCHAR(12),
701 IN sourcenode VARCHAR(75),
702 IN targetnode VARCHAR(75),
703 IN attributename VARCHAR(100),
704 IN edgeweight DOUBLE,
705 IN weightdirection ENUM('<','>'),
706 IN edgeweightatt VARCHAR(100)
707 )
708 interior:BEGIN
709
710 IF attributename IS NOT NULL
711 THEN SET @where_stmt = CONCAT('a_d.attribute_name = ',
attributename,'');
712 ELSE SET @where_stmt = '1 = 1';
713 END IF;
714
715 IF sourcenode IS NOT NULL
716 THEN SET @where_stmt_2 = CONCAT('n1.node_hash = ',
sourcenode,'');
717 ELSE SET @where_stmt_2 = '1 = 1';
718 END IF;
719
720 IF targetnode IS NOT NULL
721 THEN SET @where_stmt_3 = CONCAT('n2.node_hash = ',
targetnode,'');
722 ELSE SET @where_stmt_3 = '1 = 1';
723 END IF;
724
725 SET @attributetype = NULL;
```

./mysql_resources/mysql_template.sql

```
726 IF edgeweightatt IS NOT NULL THEN
727 SELECT data_type FROM attribute_description WHERE
attribute_name = edgeweightatt AND attribute_object =
'edge' LIMIT 1 INTO @attributetype;
728 END IF;
729
730 IF @attributetype = 'string'
731 THEN SET @attributetype = NULL;
732 END IF;
733
734 IF edgeweightatt IS NOT NULL AND weightdirection IS NOT
NULL AND edgeweight IS NOT NULL
735 THEN SET @where_stmt_4 = CONCAT('e_f.algorithm = "',
edgeweightatt,'" AND e_f.weight_value ',weightdirection,
' ',edgeweight);
736 SET @edge_table = 'edge_filter';
737 ELSE SET @where_stmt_4 = '1 = 1';
738 SET @edge_table = 'edge_nofilter';
739 END IF;
740
741 IF netname IS NOT NULL
742 THEN SET @net_stmt = ', edge2network en, network net';
743 SET @net_stmt_2 = CONCAT('e_f.edge_id = en.edge_id AND
net.network_id = en.network_id AND net.network_name = "',
netname,'"');
744 ELSE SET @net_stmt = '';
745 SET @net_stmt_2 = '1 = 1';
746 END IF;
747
748 IF edgeweightatt IS NOT NULL AND weightdirection IS NOT
NULL AND edgeweight IS NOT NULL
749 THEN SET @select_stmt = CONCAT_WS(' ', 'SELECT
a_d.data_type,a_d.attribute_name,a.attribute_blob as
attribute_value,e_f.source_node,e_f.target_node,e_f.algori
thm,e_f.weight_value
750 FROM',@edge_table,'AS e_f, attribute_description AS a_d,
attribute AS a',@net_stmt,
751 'WHERE a.object_id = e_f.edge_id AND a_d.attribute_id =
a.attribute_id AND a_d.attribute_object = "edge" AND',
752 @where_stmt,'AND',@where_stmt_2,'AND',@where_stmt_3,'AND'
,@where_stmt_4,'AND',@net_stmt_2);
753 ELSE SET @select_stmt = CONCAT_WS(' ', 'SELECT
a_d.data_type,a_d.attribute_name,a.attribute_blob as
attribute_value,e_f.source_node,e_f.target_node
```

```
754 FROM',@edge_table,'AS e_f, attribute_description AS a_d,  
attribute AS a',@net_stmt,  
755 'WHERE a.object_id = e_f.edge_id AND a_d.attribute_id =  
a.attribute_id AND a_d.attribute_object = "edge" AND',  
756 @where_stmt,'AND',@where_stmt_2,'AND',@where_stmt_3,'AND'  
,@where_stmt_4,'AND',@net_stmt_2);  
757 END IF;  
758  
759 PREPARE exec_stmt FROM @select_stmt;  
760 EXECUTE exec_stmt;  
761  
762 END //  
763  
764 DROP PROCEDURE IF EXISTS `DeleteAttribute` //  
765 CREATE PROCEDURE DeleteAttribute(  
766 IN netname VARCHAR(12),  
767 IN sourcenode VARCHAR(75),  
768 IN targetnode VARCHAR(75),  
769 IN attributename VARCHAR(100),  
770 IN atttype ENUM('network','node','edge')  
771 )  
772 BEGIN  
773  
774 IF attributename IS NOT NULL  
775 THEN SET @where_stmt = CONCAT('a_d.attribute_name = "',  
attributename,'"');  
776 ELSE SET @where_stmt = '1 = 1';  
777 END IF;  
778  
779 IF sourcenode IS NOT NULL AND atttype = 'node'  
780 THEN SET @where_stmt_2 = CONCAT('nd.node_hash = "',  
sourcenode,'"');  
781 ELSEIF netname IS NOT NULL AND atttype = 'network'  
782 THEN SET @where_stmt_2 = CONCAT('net.network_name = "',  
netname,'"');  
783 ELSE SET @where_stmt_2 = '1 = 1';  
784 END IF;  
785  
786 IF targetnode IS NOT NULL  
787 THEN SET @where_stmt_3 = CONCAT('n2.node_hash = "',  
targetnode,'"');  
788 ELSE SET @where_stmt_3 = '1 = 1';  
789 END IF;  
790
```

./mysql_resources/mysql_template.sql

```
791 SET @where_stmt_4 = CONCAT('a_d.attribute_object = "',
atttype, '"');
792
793 SET @initial_stmt = 'DELETE a.* FROM attribute AS a
794 JOIN attribute_description a_d
795 ON a.attribute_id = a_d.attribute_id';
796
797 IF atttype = 'node'
798 THEN SET @select_stmt = CONCAT_WS(' ',@initial_stmt,
799 'JOIN node nd
800 ON nd.node_id = a.object_id
801 WHERE',@where_stmt,'AND',@where_stmt_2,'AND',@
where_stmt_4);
802 ELSEIF atttype = 'edge'
803 THEN SET @select_stmt = CONCAT_WS(' ',@initial_stmt,
804 'JOIN edge e
805 ON a.object_id = e.edge_id
806 JOIN node n1
807 ON n1.node_id = e.source_node_id
808 JOIN node n2
809 ON n2.node_id = e.target_node_id
810 WHERE',@where_stmt,'AND',@where_stmt_2,'AND',@
where_stmt_3,'AND',@where_stmt_4);
811 ELSEIF atttype = 'network'
812 THEN SET @select_stmt = CONCAT_WS(' ',@initial_stmt,
813 'JOIN network net
814 ON a.object_id = net.network_id
815 WHERE',@where_stmt,'AND',@where_stmt_2,'AND',@
where_stmt_4);
816 ELSE SET @select_stmt = CONCAT_WS(' ',@initial_stmt,
817 'WHERE',@where_stmt);
818 END IF;
819
820 PREPARE exec_stmt FROM @select_stmt;
821 EXECUTE exec_stmt;
822
823 END //
824
825 DROP PROCEDURE IF EXISTS `DeleteEdgeInNetwork` //
826 CREATE PROCEDURE DeleteEdgeInNetwork(
827 IN netname VARCHAR(12),
828 IN sourcenode VARCHAR(75),
829 IN targetnode VARCHAR(75),
830 IN edgeweight DOUBLE,
```

./mysql_resources/mysql_template.sql

```
831  IN weightdirection ENUM('<','>','<=','>='),
832  IN edgeweightatt VARCHAR(100)
833  )
834  interior:BEGIN
835
836  IF netname IS NOT NULL
837  THEN SET @where_stmt = CONCAT('net.network_name = "',
      netname, '"');
838  ELSE SET @where_stmt = '1 = 1';
839  END IF;
840
841  IF sourcenode IS NOT NULL
842  THEN SET @where_stmt_2 = CONCAT('n1.node_hash = "',
      sourcenode, '"');
843  ELSE SET @where_stmt_2 = '1 = 1';
844  END IF;
845
846  IF targetnode IS NOT NULL
847  THEN SET @where_stmt_3 = CONCAT('n2.node_hash = "',
      targetnode, '"');
848  ELSE SET @where_stmt_3 = '1 = 1';
849  END IF;
850
851  IF edgeweight IS NULL OR weightdirection IS NULL OR
      edgeweightatt IS NULL
852  THEN SET @delete_stmt = CONCAT_WS(' ','DELETE en.*
853  FROM edge2network AS en, network AS net,edge AS e, node
      AS n1, node AS n2
854  WHERE e.edge_id = en.edge_id AND en.network_id =
      net.network_id AND
855  n1.node_id = e.source_node_id AND n2.node_id =
      e.target_node_id AND',
856  @where_stmt,'AND',@where_stmt_2,'AND',@where_stmt_3);
857  PREPARE exec_stmt FROM @delete_stmt;
858  EXECUTE exec_stmt;
859  LEAVE interior;
860  END IF;
861
862  SET @attributetype = NULL;
863  SELECT data_type FROM attribute_description WHERE
      attribute_name = edgeweightatt AND attribute_object =
      'edge' LIMIT 1 INTO @attributetype;
864  IF @attributetype IS NULL OR @attributetype = 'string'
865  THEN LEAVE interior;
```

./mysql_resources/mysql_template.sql

```
866 END IF;
867
868 SET @where_stmt_4 = CONCAT('a_d.attribute_name = "',
    edgeweightatt,'" AND a.attribute_value ',weightdirection,
    ' ',edgeweight);
869
870 SET @delete_stmt = CONCAT_WS(' ', 'DELETE en.*
871 FROM edge AS e, attribute AS a, attribute_description AS
    a_d, node AS n1,
872 node AS n2, edge2network AS en, network AS net
873 WHERE a.object_id = e.edge_id AND a_d.attribute_id =
    a.attribute_id AND a_d.attribute_object = "edge"
874 AND n1.node_id = e.source_node_id AND n2.node_id =
    e.target_node_id AND
875 AND e.edge_id = en.edge_id AND en.network_id =
    net.network_id AND',
876 @where_stmt, 'AND', @where_stmt_2, 'AND', @where_stmt_3, 'AND'
    , @where_stmt_4);
877 PREPARE exec_stmt FROM @delete_stmt;
878 EXECUTE exec_stmt;
879 END //
880
881 DROP PROCEDURE IF EXISTS `DeleteNodeInNetwork` //
882 CREATE PROCEDURE DeleteNodeInNetwork(
883 IN netname VARCHAR(12),
884 IN nodehash VARCHAR(75)
885 )
886 BEGIN
887
888 IF netname IS NOT NULL
889 THEN SET @where_stmt = CONCAT('net.network_name = "',
    netname, '"');
890 ELSE SET @where_stmt = '1 = 1';
891 END IF;
892
893 IF nodehash IS NOT NULL
894 THEN SET @where_stmt_2 = CONCAT('nd.node_hash = "',
    nodehash, '"');
895 ELSE SET @where_stmt_2 = '1 = 1';
896 END IF;
897
898 SET @delete_stmt = CONCAT_WS(' ', 'DELETE nn.*
899 FROM node as nd
900 LEFT JOIN (node2network nn, network net)
```

./mysql_resources/mysql_template.sql

```
901  ON (nd.node_id = nn.node_id AND net.network_id =
    nn.network_id)
902  WHERE',@where_stmt,'AND',@where_stmt_2);
903  PREPARE exec_stmt FROM @delete_stmt;
904  EXECUTE exec_stmt;
905
906  END //
```

```
907
908  DROP PROCEDURE IF EXISTS `DeleteNode` //
```

```
909  CREATE PROCEDURE DeleteNode(
910  IN nodehash VARCHAR(75),
911  IN attributename VARCHAR(100),
912  IN identiervalue VARCHAR(100)
913  )
914  interior:BEGIN
915
916  SET @nodehash2 = NULL;
917  IF attributename IS NOT NULL AND identiervalue IS NOT
    NULL
918  THEN SELECT nd.node_hash FROM identifier as i
919  JOIN attribute_description a_d
920  ON a_d.attribute_id = i.attribute_id
921  JOIN node nd
922  ON nd.node_id = i.node_id
923  WHERE a_d.attribute_object = 'identifier' AND a_d.
    attribute_name = attributename AND i.identifier_value =
    identiervalue
924  LIMIT 1 INTO @nodehash2;
925  END IF;
926
927  IF nodehash IS NOT NULL AND @nodehash2 IS NOT NULL AND
    nodehash != @nodehash2
928  THEN LEAVE interior;
929  ELSEIF nodehash IS NULL AND @nodehash2 IS NOT NULL
930  THEN SET nodehash = @nodehash2;
931  END IF;
932
933  CALL deleteNodeInNetwork(NULL,nodehash);
934  CALL deleteAttribute(NULL,nodehash,NULL,NULL,'node');
935  CALL deleteEdge(nodehash,NULL);
936  CALL deleteEdge(NULL,nodehash);
937
938  SET @where_stmt = CONCAT('nd.node_hash = ',nodehash,'');
939  IF @where_stmt IS NULL
```


./mysql_resources/mysql_template.sql

```
940 THEN SET @where_stmt = '1 = 1';
941 END IF;
942
943 SET @delete_stmt = CONCAT('DELETE i.*
944 FROM node AS nd, identifier AS i
945 WHERE i.node_id = nd.node_id AND ',@where_stmt);
946 PREPARE exec_stmt FROM @delete_stmt;
947 EXECUTE exec_stmt;
948
949 SET @delete_stmt_2 = CONCAT('DELETE nd.* FROM node AS nd
950 WHERE ',@where_stmt);
951 PREPARE exec_stmt_2 FROM @delete_stmt_2;
952 EXECUTE exec_stmt_2;
953
954
955 END //
956
957 DROP PROCEDURE IF EXISTS `DeleteNetwork` //
958 CREATE PROCEDURE DeleteNetwork(
959 IN netname VARCHAR(20)
960 )
961 BEGIN
962 CALL deleteNodeInNetwork(netname,NULL);
963 CALL deleteEdgeInNetwork(netname,NULL,NULL,NULL,NULL
964 );
965 CALL deleteAttribute(netname,NULL,NULL,NULL,'network');
966
967 SET @where_stmt = CONCAT('network_name = "',netname,'"');
968 IF @where_stmt IS NULL
969 THEN SET @where_stmt = '1 = 1';
970 END IF;
971
972 SET @select_stmt = CONCAT_WS(' ','DELETE FROM network
973 WHERE ',@where_stmt);
974 PREPARE exec_stmt FROM @select_stmt;
975 EXECUTE exec_stmt;
976
977 END //
978
979 DROP PROCEDURE IF EXISTS `DeleteEdge` //
980 CREATE PROCEDURE DeleteEdge(
981 IN sourcenode VARCHAR(75),
982 IN targetnode VARCHAR(75)
983 )
```

./mysql_resources/mysql_template.sql

```
981 interior:BEGIN
982
983 CALL deleteEdgeInNetwork(NULL, sourcenode, targetnode, NULL,
NULL, NULL);
984 CALL deleteAttribute(NULL, sourcenode, targetnode, NULL,
'edge');
985
986 IF sourcenode IS NOT NULL
987 THEN SET @where_stmt = CONCAT('n1.node_hash = "',
sourcenode, '"');
988 ELSE SET @where_stmt = '1 = 1';
989 END IF;
990
991 IF targetnode IS NOT NULL
992 THEN SET @where_stmt_2 = CONCAT('n2.node_hash = "',
targetnode, '"');
993 ELSE SET @where_stmt_2 = '1 = 1';
994 END IF;
995
996 SET @delete_stmt = CONCAT_WS(' ', 'DELETE e.*
997 FROM edge AS e, node AS n1, node AS n2
998 WHERE n1.node_id = e.source_node_id AND n2.node_id =
e.target_node_id
999 AND', @where_stmt, 'AND', @where_stmt_2);
1000 PREPARE exec_stmt FROM @delete_stmt;
1001 EXECUTE exec_stmt;
1002
1003
1004 END //
1005
1006 DROP PROCEDURE IF EXISTS `ClearAll` //
1007 CREATE PROCEDURE ClearAll()
1008
1009 BEGIN
1010
1011 DELETE FROM node;
1012 DELETE FROM edge;
1013 DELETE FROM attribute;
1014 DELETE FROM network;
1015 DELETE FROM node2network;
1016 DELETE FROM edge2network;
1017 DELETE FROM attribute_description;
1018 DELETE FROM identifier;
1019 ALTER TABLE node AUTO_INCREMENT = 1;
```

./mysql_resources/mysql_template.sql

```
1020 ALTER TABLE edge AUTO_INCREMENT = 1;  
1021 ALTER TABLE network AUTO_INCREMENT = 1;  
1022 ALTER TABLE attribute_description AUTO_INCREMENT = 1;  
1023  
1024 END //
```

./mysql_resources/stored_procs.sql

```
1  -- Author: Alan Barber
2  -- This file only updates the views and stored
   procedures that are used by the MySQL interface.
3  -- The reason for a seperate script to do this is for
   bug and script updates. This code can
4  -- update the the underlying functions without changing
   or dropping tables with existing data.
5  -- Usage: mysql [database] < stored_procs.sql
6  -- Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
7
8  --
   -----
   -----
9  -- Views
10 --
   -----
   -----
11
12 DROP VIEW IF EXISTS `network_attribute`;
13 DROP TABLE IF EXISTS `network_attribute`;
14 CREATE VIEW network_attribute AS
15 SELECT n.network_name,n.kind as network_type,a_d.
   attribute_name,a_d.data_type,a.attribute_blob as
   attribute_value,
16 n.network_id,a_d.attribute_id
17 FROM attribute AS a, attribute_description AS a_d,
   network AS n
18 WHERE a.attribute_id = a_d.attribute_id AND n.network_id
   = a.object_id
19 AND a_d.attribute_object = "network";
20
21 DROP VIEW IF EXISTS `node_identifier`;
22 DROP TABLE IF EXISTS `node_identifier`;
23 CREATE VIEW node_identifier AS
24 SELECT nd.node_hash as id,a_d.attribute_name as
   identifier_type,i.identifier_value as identifier_value,nd
   .node_id
25 FROM node AS nd
26 LEFT JOIN (identifier i, attribute_description a_d)
27 ON (i.node_id = nd.node_id AND a_d.attribute_id = i.
   attribute_id AND a_d.attribute_object = "identifier");
28
29 DROP VIEW IF EXISTS `node_attribute`;
```

./mysql_resources/stored_procs.sql

```
30 DROP TABLE IF EXISTS `node_attribute`;
31 CREATE VIEW node_attribute AS
32 SELECT n.node_hash as id,a_d.attribute_name,a_d.data_type
   ,a.attribute_blob as attribute_value,
33 n.node_id,a_d.attribute_id
34 FROM attribute AS a, attribute_description AS a_d, node
   AS n
35 WHERE a.attribute_id = a_d.attribute_id AND n.node_id = a
   .object_id
36 AND a_d.attribute_object = "node";
37
38 DROP VIEW IF EXISTS `edge_filter`;
39 DROP TABLE IF EXISTS `edge_filter`;
40 CREATE VIEW edge_filter AS
41 SELECT e.edge_id,n1.node_hash as source_node,n2.node_hash
   as target_node,
42 a_d.attribute_name as algorithm, a.attribute_blob as
   weight_value
43 FROM edge AS e, attribute AS a, attribute_description AS
   a_d, node AS n1, node AS n2
44 WHERE a.object_id = e.edge_id AND a_d.attribute_id = a.
   attribute_id AND a_d.attribute_object = "edge"
45 AND n1.node_id = e.source_node_id AND n2.node_id = e.
   target_node_id;
46
47 DROP VIEW IF EXISTS `edge_filter_network`;
48 DROP TABLE IF EXISTS `edge_filter_network`;
49 CREATE VIEW edge_filter_network AS
50 SELECT e.edge_id,n1.node_hash as source_node,n2.node_hash
   as target_node,
51 a_d.attribute_name as algorithm, a.attribute_blob as
   weight_value,
52 net.network_name,net.kind as network_type
53 FROM edge AS e, attribute AS a, attribute_description AS
   a_d,
54 node AS n1, node AS n2, network AS net, edge2network AS
   en
55 WHERE a.object_id = e.edge_id AND a_d.attribute_id = a.
   attribute_id AND a_d.attribute_object = "edge"
56 AND n1.node_id = e.source_node_id AND n2.node_id = e.
   target_node_id
57 AND e.edge_id = en.edge_id AND net.network_id = en.
   network_id;
58
```

./mysql_resources/stored_procs.sql

```
59 DROP VIEW IF EXISTS `edge_nofilter`;
60 DROP TABLE IF EXISTS `edge_nofilter`;
61 CREATE VIEW edge_nofilter AS
62 SELECT e.edge_id,n1.node_hash as source_node,n2.node_hash
   as target_node
63 FROM edge AS e, node AS n1, node AS n2
64 WHERE n1.node_id = e.source_node_id AND n2.node_id = e.
   target_node_id;
65
66 DROP VIEW IF EXISTS `edge_nofilter_network`;
67 DROP TABLE IF EXISTS `edge_nofilter_network`;
68 CREATE VIEW edge_nofilter_network AS
69 SELECT e.edge_id,n1.node_hash as source_node,n2.node_hash
   as target_node,
70 net.network_name,net.kind as network_type
71 FROM edge AS e, node AS n1, node AS n2, network AS net,
   edge2network AS en
72 WHERE n1.node_id = e.source_node_id AND n2.node_id = e.
   target_node_id
73 AND e.edge_id = en.edge_id AND net.network_id = en.
   network_id;
74
75 --
-----
-----
76 -- Stored procedures
77 --
-----
-----
78
79 DELIMITER //
80
81 DROP PROCEDURE IF EXISTS `AddNode` //
82 CREATE PROCEDURE AddNode (
83 IN nodehash VARCHAR(75)
84 )
85 interior:BEGIN
86
87 IF nodehash IS NULL
88 THEN LEAVE interior;
89 END IF;
90
91 SELECT * FROM node WHERE node_hash = nodehash;
92 IF FOUND_ROWS() = 0
```

./mysql_resources/stored_procs.sql

```
93  THEN INSERT INTO node (node_hash) VALUES (nodehash);
94  END IF;
95  END //
96
97  DROP PROCEDURE IF EXISTS `AddNetwork` //
98  CREATE PROCEDURE AddNetwork (
99  IN netname VARCHAR(20),
100 IN netkind VARCHAR(20)
101 )
102 interior:BEGIN
103
104 IF netname IS NULL
105 THEN LEAVE interior;
106 END IF;
107
108 SELECT * FROM network WHERE network_name = netname;
109 IF FOUND_ROWS() = 0
110 THEN INSERT INTO network (network_name,kind) VALUES (
    netname,netkind);
111 ELSE
112 UPDATE network SET kind = netkind WHERE network_name =
    netname;
113 END IF;
114 END //
115
116 DROP PROCEDURE IF EXISTS `AddNodeToNetwork` //
117 CREATE PROCEDURE AddNodeToNetwork(
118 IN netname VARCHAR(20),
119 IN nodehash VARCHAR(75)
120 )
121 BEGIN
122
123 SET @netid = NULL;
124 SET @nodeid = NULL;
125 SET @netflag = NULL;
126 SELECT node_id FROM node WHERE node_hash = nodehash LIMIT
    1 INTO @nodeid;
127 SELECT network_id FROM network WHERE network_name =
    netname LIMIT 1 INTO @netid;
128
129 IF @netid IS NOT NULL AND @nodeid IS NOT NULL
130 THEN SELECT node_id FROM node2network WHERE network_id =
    @netid AND node_id = @nodeid LIMIT 1 INTO @netflag;
131 END IF;
```

./mysql_resources/stored_procs.sql

```
132
133 IF @netid IS NOT NULL AND @nodeid IS NOT NULL AND @
netflag IS NULL
134 THEN INSERT INTO node2network (network_id,node_id) VALUES
(@netid,@nodeid);
135 END IF;
136
137 END //
138
139 DROP PROCEDURE IF EXISTS `AddEdge` //
140 CREATE PROCEDURE AddEdge (
141 IN sourcenode VARCHAR(75),
142 IN targetnode VARCHAR(75)
143 )
144 interior:BEGIN
145
146 SET @sourcenodeid = NULL;
147 SET @targetnodeid = NULL;
148 SELECT node_id FROM node WHERE node_hash = sourcenode
LIMIT 1 INTO @sourcenodeid;
149 SELECT node_id FROM node WHERE node_hash = targetnode
LIMIT 1 INTO @targetnodeid;
150
151 IF @sourcenodeid IS NULL OR @targetnodeid IS NULL
152 THEN LEAVE interior;
153 END IF;
154
155 SELECT * FROM edge WHERE source_node_id = @sourcenodeid
AND target_node_id = @targetnodeid;
156 IF FOUND_ROWS() = 0
157 THEN INSERT INTO edge (source_node_id,target_node_id)
VALUES (@sourcenodeid,@targetnodeid);
158 END IF;
159 END //
160
161 DROP PROCEDURE IF EXISTS `AddEdgeToNetwork` //
162 CREATE PROCEDURE AddEdgeToNetwork(
163 IN netname VARCHAR(20),
164 IN sourcenode VARCHAR(75),
165 IN targetnode VARCHAR(75)
166 )
167 BEGIN
168
169 SET @netid = NULL;
```


./mysql_resources/stored_procs.sql

```
170 SET @edgeid = NULL;
171 SET @netflag = NULL;
172 SELECT e.edge_id
173 FROM edge AS e, node AS n1, node AS n2
174 WHERE n2.node_id = e.target_node_id AND n1.node_id = e.
    source_node_id
175 AND n1.node_hash = sourcenode AND n2.node_hash =
    targetnode
176 LIMIT 1
177 INTO @edgeid;
178 SELECT network_id FROM network WHERE network_name =
    netname LIMIT 1 INTO @netid;
179
180 SELECT @edgeid;
181 SELECT @netid;
182
183 IF @netid IS NOT NULL AND @edgeid IS NOT NULL
184 THEN SELECT edge_id FROM edge2network WHERE network_id =
    @netid AND edge_id = @edgeid LIMIT 1 INTO @netflag;
185 END IF;
186
187 IF @netid IS NOT NULL AND @edgeid IS NOT NULL AND @
    netflag IS NULL
188 THEN INSERT INTO edge2network (network_id,edge_id) VALUES
    (@netid,@edgeid);
189 END IF;
190
191 END //
192
193 DROP PROCEDURE IF EXISTS `AddAttribute` //
194 CREATE PROCEDURE AddAttribute(
195 IN netname VARCHAR(12),
196 IN sourcenode VARCHAR(75),
197 IN targetnode VARCHAR(75),
198 IN attributename VARCHAR(100),
199 IN attobject ENUM('network','node','edge'),
200 IN attdatatype ENUM('double','int','string','bool'),
201 IN attributeblob BLOB
202 )
203 interior:BEGIN
204
205 SET @attributeid = NULL;
206 SET @objectid = NULL;
207 SELECT attribute_id FROM attribute_description WHERE
```

./mysql_resources/stored_procs.sql

```
attribute_name = attributename AND attribute_object =
attobject LIMIT 1 INTO @attributeid;
208
209 IF attobject = 'network' AND netname IS NOT NULL
210 THEN SELECT network_id FROM network WHERE network_name =
netname LIMIT 1 INTO @objectid;
211 ELSEIF attobject = 'node' AND sourcenode IS NOT NULL
212 THEN SELECT node_id FROM node WHERE node_hash =
sourcenode LIMIT 1 INTO @objectid;
213 ELSEIF attobject = 'edge' AND sourcenode IS NOT NULL and
targetnode IS NOT NULL
214 THEN SELECT e.edge_id
215 FROM edge AS e, node AS n1, node AS n2
216 WHERE n2.node_id = e.target_node_id AND n1.node_id = e.
source_node_id
217 AND n1.node_hash = sourcenode AND n2.node_hash =
targetnode
218 LIMIT 1
219 INTO @objectid;
220 END IF;
221
222 IF @objectid IS NULL
223 THEN LEAVE interior;
224 END IF;
225
226 IF @attributeid IS NULL
227 THEN INSERT INTO attribute_description (attribute_name,
attribute_object,data_type) VALUES (attributename,
attobject,attdatatype);
228 SET @attributeid = LAST_INSERT_ID();
229 END IF;
230
231 SELECT * FROM attribute WHERE attribute_id = @
attributeid AND object_id = @objectid;
232 IF FOUND_ROWS() = 0
233 THEN INSERT INTO attribute (attribute_id,object_id,
attribute_blob) VALUES (@attributeid,@objectid,
attributeblob);
234 ELSE
235 UPDATE attribute SET attribute_blob = attributeblob WHERE
attribute_id = @attributeid AND object_id = @objectid;
236 END IF;
237 END //
238
```

./mysql_resources/stored_procs.sql

```
239 DROP PROCEDURE IF EXISTS `AddIdentifier` //
240 CREATE PROCEDURE AddIdentifier(
241     IN nodehash VARCHAR(75),
242     IN attributename VARCHAR(100),
243     IN identiervalue VARCHAR(100)
244 )
245 interior:BEGIN
246
247     SET @attributeid = NULL;
248     SET @nodeid = NULL;
249     SELECT node_id FROM node WHERE node_hash = nodehash LIMIT
        1 INTO @nodeid;
250
251     IF @nodeid IS NULL
252     THEN LEAVE interior;
253     END IF;
254
255     SELECT attribute_id FROM attribute_description WHERE
        attribute_name = attributename AND attribute_object =
        'identifier' LIMIT 1 INTO @attributeid;
256
257     IF @attributeid IS NULL
258     THEN INSERT INTO attribute_description (attribute_name,
        attribute_object,data_type) VALUES (attributename,
        'identifier','string');
259     SET @attributeid = LAST_INSERT_ID();
260     END IF;
261
262     SELECT * FROM identifier WHERE attribute_id = @
        attributeid AND identifier_value = identiervalue;
263     IF FOUND_ROWS() = 0
264     THEN INSERT INTO identifier (attribute_id,
        identifier_value,node_id) VALUES (@attributeid,
        identiervalue,@nodeid);
265     ELSE
266     UPDATE identifier SET node_id = @nodeid WHERE
        attribute_id = @attributeid AND identifier_value =
        identiervalue;
267     END IF;
268     END //
269
270 DROP PROCEDURE IF EXISTS `SelectNetwork` //
271 CREATE PROCEDURE SelectNetwork(
272     IN netname VARCHAR(20),
```

./mysql_resources/stored_procs.sql

```
273  IN netkind VARCHAR(20)
274  )
275  BEGIN
276
277  SET @where_stmt = CONCAT('network_name = "',netname,'"');
278  SET @where_stmt_2 = CONCAT('kind = "',netkind,'"');
279
280  IF @where_stmt IS NULL
281  THEN SET @where_stmt = '1 = 1';
282  END IF;
283
284  IF @where_stmt_2 IS NULL
285  THEN SET @where_stmt_2 = '1 = 1';
286  END IF;
287
288  SET @select_stmt = CONCAT_WS(' ', 'SELECT
network_name,kind as network_type FROM network WHERE',@
where_stmt,'AND',@where_stmt_2);
289  PREPARE exec_stmt FROM @select_stmt;
290  EXECUTE exec_stmt;
291
292  END //
293
294  DROP PROCEDURE IF EXISTS `SelectNetworkAttribute` //
295  CREATE PROCEDURE SelectNetworkAttribute(
296  IN netname VARCHAR(20),
297  IN attributename VARCHAR(100)
298  )
299  BEGIN
300
301  IF attributename IS NOT NULL
302  THEN SET @where_stmt = CONCAT('attribute_name = "',
attributename,'"');
303  ELSE SET @where_stmt = '1 = 1';
304  END IF;
305
306  IF netname IS NOT NULL
307  THEN SET @where_stmt_2 = CONCAT('network_name = "',
netname,'"');
308  ELSE SET @where_stmt_2 = '1 = 1';
309  END IF;
310
311  SET @select_stmt = CONCAT_WS(' ', 'SELECT
network_name,network_type,attribute_name,data_type,attribu
```

./mysql_resources/stored_procs.sql

```
te_value
312 FROM network_attribute WHERE',@where_stmt,'AND',@
    where_stmt_2);
313 PREPARE exec_stmt FROM @select_stmt;
314 EXECUTE exec_stmt;
315
316 END //
317
318 DROP PROCEDURE IF EXISTS `SelectNodeInNetwork` //
319 CREATE PROCEDURE SelectNodeInNetwork(
320     IN netname VARCHAR(12),
321     IN nodehash VARCHAR(75),
322     IN attributename VARCHAR(100),
323     IN identiervalue VARCHAR(100)
324 )
325 interior:BEGIN
326
327     SET @nodeid = NULL;
328     IF nodehash IS NOT NULL
329     THEN SELECT node_id FROM node WHERE node_hash = nodehash
330         LIMIT 1 INTO @nodeid;
331     END IF;
332
333     SET @nodeid2 = NULL;
334     IF attributename IS NOT NULL AND identiervalue IS NOT
335     NULL
336     THEN SELECT i.node_id FROM identifier as i,
337         attribute_description as a_d
338     WHERE a_d.attribute_id = i.attribute_id AND a_d.
339         attribute_object = 'identifier'
340     AND a_d.attribute_name = attributename AND i.
341         identifier_value = identiervalue
342     LIMIT 1 INTO @nodeid2;
343     END IF;
344
345     IF @nodeid IS NOT NULL AND @nodeid2 IS NOT NULL AND @
346     nodeid != @nodeid2
347     THEN LEAVE interior;
348     ELSEIF @nodeid IS NOT NULL AND @nodeid2 IS NOT NULL AND @
349     nodeid = @nodeid2
350     THEN SET @nodeid2 = NULL;
351     END IF;
352
353     IF @nodeid IS NOT NULL
```

./mysql_resources/stored_procs.sql

```
347 THEN SET @where_stmt = CONCAT('node_id = ',@nodeid);
348 ELSE SET @where_stmt = '1 = 1';
349 END IF;
350
351 IF @nodeid2 IS NOT NULL
352 THEN SET @where_stmt_2 = CONCAT('node_id = ',@nodeid2);
353 ELSE SET @where_stmt_2 = '1 = 1';
354 END IF;
355
356 IF netname IS NOT NULL
357 THEN SET @net_stmt = CONCAT('JOIN (node2network nn,
network net)
358 ON (n_i.node_id = nn.node_id AND net.network_id =
nn.network_id AND net.network_name = "',netname,'"'));
359 ELSE SET @net_stmt = 'LEFT JOIN (node2network nn,
network net)
360 ON (n_i.node_id = nn.node_id AND net.network_id =
nn.network_id)';
361 END IF;
362
363 SET @select_stmt = CONCAT_WS(' ','SELECT
net.network_name,net.kind as
network_type,n_i.id,n_i.identifier_type,n_i.identifier_val
ue
364 FROM (SELECT * FROM node_identifier WHERE',
365 @where_stmt,'AND',@where_stmt_2,') as n_i',@net_stmt);
366 PREPARE exec_stmt FROM @select_stmt;
367 EXECUTE exec_stmt;
368
369 END //
370
371 DROP PROCEDURE IF EXISTS `SelectNodeAttributeInNetwork` //
372 CREATE PROCEDURE SelectNodeAttributeInNetwork(
373 IN netname VARCHAR(12),
374 IN nodehash VARCHAR(75),
375 IN attributename VARCHAR(100)
376 )
377 BEGIN
378
379 IF attributename IS NOT NULL
380 THEN SET @where_stmt = CONCAT('n_a.attribute_name = "',
attributename,'"');
381 ELSE SET @where_stmt = '1 = 1';
382 END IF;
```

./mysql_resources/stored_procs.sql

```
383
384 IF nodehash IS NOT NULL
385 THEN SET @where_stmt_2 = CONCAT('n_a.id = "',nodehash,'"
');
386 ELSE SET @where_stmt_2 = '1 = 1';
387 END IF;
388
389 SET @select_stmt = NULL;
390
391 IF netname IS NULL
392 THEN SET @select_stmt = CONCAT_WS(' ','SELECT
id,attribute_name,data_type,attribute_value
393 FROM node_attribute AS n_a WHERE',@where_stmt,'AND',@
where_stmt_2);
394 ELSE SET @select_stmt = CONCAT('SELECT
n_a.id,n_a.attribute_name,n_a.data_type,n_a.attribute_valu
e
395 FROM node_attribute AS n_a, node2network AS nn, network
AS net
396 WHERE n_a.node_id = nn.node_id AND net.network_id =
nn.network_id AND
397 net.network_name = "',netname,'" AND ',@where_stmt,' AND
',@where_stmt_2);
398 END IF;
399
400 IF @select_stmt IS NOT NULL THEN
401 PREPARE exec_stmt FROM @select_stmt;
402 EXECUTE exec_stmt;
403 END IF;
404
405 END //
406
407 DROP PROCEDURE IF EXISTS `SelectEdgeInNetwork` //
408 CREATE PROCEDURE SelectEdgeInNetwork(
409 IN netname VARCHAR(12),
410 IN sourcenode VARCHAR(75),
411 IN targetnode VARCHAR(75),
412 IN edgeweight DOUBLE,
413 IN weightdirection ENUM('<','>'),
414 IN edgeweightatt VARCHAR(100)
415 )
416 interior:BEGIN
417
418 IF sourcenode IS NOT NULL
```

./mysql_resources/stored_procs.sql

```
419 THEN SET @where_stmt = CONCAT('source_node = "',
    sourcenode, '");
420 ELSE SET @where_stmt = '1 = 1';
421 END IF;
422
423 IF targetnode IS NOT NULL
424 THEN SET @where_stmt_2 = CONCAT('target_node = "',
    targetnode, '");
425 ELSE SET @where_stmt_2 = '1 = 1';
426 END IF;
427
428 IF edgeweight IS NULL OR weightdirection IS NULL OR
    edgeweightatt IS NULL
429 THEN SET @filter_flag = 0;
430 ELSE SET @filter_flag = 1;
431 END IF;
432
433 IF @filter_flag = 0 THEN
434     IF netname IS NULL
435     THEN SET @select_stmt = CONCAT_WS(' ', 'SELECT
    net.network_name, net.kind as network_type,
436     e_f.source_node, e_f.target_node
437     FROM (SELECT * FROM edge_nofilter WHERE ',
438     @where_stmt, 'AND ', @where_stmt_2, ') as e_f LEFT
    JOIN (edge2network en, network net)
439     ON (e_f.edge_id = en.edge_id AND net.network_id
    = en.network_id)');
440     ELSE
441     SET @select_stmt = CONCAT('SELECT
    network_name, network_type,
442     source_node, target_node
443     FROM edge_nofilter_network WHERE ',
444     @where_stmt, ' AND ', @where_stmt_2, ' AND
    network_name = "', netname, '");
445     END IF;
446 PREPARE exec_stmt FROM @select_stmt;
447 EXECUTE exec_stmt;
448 LEAVE interior;
449 END IF;
450
451 SET @attributetype = NULL;
452 SELECT data_type FROM attribute_description WHERE
    attribute_name = edgeweightatt AND attribute_object =
    'edge' LIMIT 1 INTO @attributetype;
```


./mysql_resources/stored_procs.sql

```
453 IF @attributetype IS NULL OR @attributetype = 'string'
454 THEN LEAVE interior;
455 END IF;
456
457 SET @where_stmt_3 = CONCAT('algorithm = "',edgeweightatt,
    ''););
458 SET @where_stmt_4 = CONCAT('weight_value ',
    weightdirection, ' ',edgeweight);
459
460 IF netname IS NULL
461 THEN SET @select_stmt = CONCAT_WS(' ','SELECT
    net.network_name,net.kind as network_type,
462 e_f.source_node,e_f.target_node,e_f.algorithm,e_f.weight_v
    alue
463 FROM (SELECT * FROM edge_filter WHERE',@where_stmt,'AND',
    @where_stmt_2,'AND',
464 @where_stmt_3,'AND',@where_stmt_4,') as e_f LEFT JOIN
    (edge2network en, network net)
465 ON (e_f.edge_id = en.edge_id AND net.network_id =
    en.network_id)'););
466 ELSE
467 SET @select_stmt = CONCAT('SELECT
    network_name,network_type,
468 source_node,target_node,algorithm,weight_value
469 FROM edge_filter_network WHERE ',@where_stmt,' AND ',@
    where_stmt_2,' AND ',
470 @where_stmt_3,' AND ',@where_stmt_4,' AND network_name =
    "',netname,'"););
471 END IF;
472
473 PREPARE exec_stmt FROM @select_stmt;
474 EXECUTE exec_stmt;
475 END //
476
477 DROP PROCEDURE IF EXISTS `SelectEdgeAttributeInNetwork` //
478 CREATE PROCEDURE SelectEdgeAttributeInNetwork(
479 IN netname VARCHAR(12),
480 IN sourcenode VARCHAR(75),
481 IN targetnode VARCHAR(75),
482 IN attributename VARCHAR(100),
483 IN edgeweight DOUBLE,
484 IN weightdirection ENUM('<','>'),
485 IN edgeweightatt VARCHAR(100)
486 )
```

```
487 interior:BEGIN
488
489 IF attributename IS NOT NULL
490 THEN SET @where_stmt = CONCAT('a_d.attribute_name = "',
    attributename, '"');
491 ELSE SET @where_stmt = '1 = 1';
492 END IF;
493
494 IF sourcenode IS NOT NULL
495 THEN SET @where_stmt_2 = CONCAT('n1.node_hash = "',
    sourcenode, '"');
496 ELSE SET @where_stmt_2 = '1 = 1';
497 END IF;
498
499 IF targetnode IS NOT NULL
500 THEN SET @where_stmt_3 = CONCAT('n2.node_hash = "',
    targetnode, '"');
501 ELSE SET @where_stmt_3 = '1 = 1';
502 END IF;
503
504 SET @attributetype = NULL;
505 IF edgeweightatt IS NOT NULL THEN
506 SELECT data_type FROM attribute_description WHERE
    attribute_name = edgeweightatt AND attribute_object =
    'edge' LIMIT 1 INTO @attributetype;
507 END IF;
508
509 IF @attributetype = 'string'
510 THEN SET @attributetype = NULL;
511 END IF;
512
513 IF edgeweightatt IS NOT NULL AND weightdirection IS NOT
    NULL AND edgeweight IS NOT NULL
514 THEN SET @where_stmt_4 = CONCAT('e_f.algorithm = "',
    edgeweightatt, '" AND e_f.weight_value ',weightdirection,
    ' ',edgeweight);
515 SET @edge_table = 'edge_filter';
516 ELSE SET @where_stmt_4 = '1 = 1';
517 SET @edge_table = 'edge_nofilter';
518 END IF;
519
520 IF netname IS NOT NULL
521 THEN SET @net_stmt = ', edge2network en, network net';
522 SET @net_stmt_2 = CONCAT('e_f.edge_id = en.edge_id AND
```

./mysql_resources/stored_procs.sql

```
net.network_id = en.network_id AND net.network_name = '',
netname, ''');
523 ELSE SET @net_stmt = '';
524 SET @net_stmt_2 = '1 = 1';
525 END IF;
526
527 IF edgeweightatt IS NOT NULL AND weightdirection IS NOT
NULL AND edgeweight IS NOT NULL
528 THEN SET @select_stmt = CONCAT_WS(' ', 'SELECT
a_d.data_type,a_d.attribute_name,a.attribute_blob as
attribute_value,e_f.source_node,e_f.target_node,e_f.algori
thm,e_f.weight_value
529 FROM',@edge_table,'AS e_f, attribute_description AS a_d,
attribute AS a',@net_stmt,
530 'WHERE a.object_id = e_f.edge_id AND a_d.attribute_id =
a.attribute_id AND a_d.attribute_object = "edge" AND',
531 @where_stmt,'AND',@where_stmt_2,'AND',@where_stmt_3,'AND'
,@where_stmt_4,'AND',@net_stmt_2);
532 ELSE SET @select_stmt = CONCAT_WS(' ', 'SELECT
a_d.data_type,a_d.attribute_name,a.attribute_blob as
attribute_value,e_f.source_node,e_f.target_node
533 FROM',@edge_table,'AS e_f, attribute_description AS a_d,
attribute AS a',@net_stmt,
534 'WHERE a.object_id = e_f.edge_id AND a_d.attribute_id =
a.attribute_id AND a_d.attribute_object = "edge" AND',
535 @where_stmt,'AND',@where_stmt_2,'AND',@where_stmt_3,'AND'
,@where_stmt_4,'AND',@net_stmt_2);
536 END IF;
537
538 PREPARE exec_stmt FROM @select_stmt;
539 EXECUTE exec_stmt;
540
541 END //
```

```
542
543 DROP PROCEDURE IF EXISTS `DeleteAttribute` //
```

```
544 CREATE PROCEDURE DeleteAttribute(
545 IN netname VARCHAR(12),
546 IN sourcenode VARCHAR(75),
547 IN targetnode VARCHAR(75),
548 IN attributename VARCHAR(100),
549 IN atttype ENUM('network','node','edge')
550 )
551 BEGIN
552
```

```
553 IF attributename IS NOT NULL
554 THEN SET @where_stmt = CONCAT('a_d.attribute_name = "',
    attributename, '"');
555 ELSE SET @where_stmt = '1 = 1';
556 END IF;
557
558 IF sourcenode IS NOT NULL AND atttype = 'node'
559 THEN SET @where_stmt_2 = CONCAT('nd.node_hash = "',
    sourcenode, '"');
560 ELSEIF netname IS NOT NULL AND atttype = 'network'
561 THEN SET @where_stmt_2 = CONCAT('net.network_name = "',
    netname, '"');
562 ELSE SET @where_stmt_2 = '1 = 1';
563 END IF;
564
565 IF targetnode IS NOT NULL
566 THEN SET @where_stmt_3 = CONCAT('n2.node_hash = "',
    targetnode, '"');
567 ELSE SET @where_stmt_3 = '1 = 1';
568 END IF;
569
570 SET @where_stmt_4 = CONCAT('a_d.attribute_object = "',
    atttype, '"');
571
572 SET @initial_stmt = 'DELETE a.* FROM attribute AS a
573 JOIN attribute_description a_d
574 ON a.attribute_id = a_d.attribute_id';
575
576 IF atttype = 'node'
577 THEN SET @select_stmt = CONCAT_WS(' ',@initial_stmt,
578 'JOIN node nd
579 ON nd.node_id = a.object_id
580 WHERE',@where_stmt,'AND',@where_stmt_2,'AND',@
    where_stmt_4);
581 ELSEIF atttype = 'edge'
582 THEN SET @select_stmt = CONCAT_WS(' ',@initial_stmt,
583 'JOIN edge e
584 ON a.object_id = e.edge_id
585 JOIN node n1
586 ON n1.node_id = e.source_node_id
587 JOIN node n2
588 ON n2.node_id = e.target_node_id
589 WHERE',@where_stmt,'AND',@where_stmt_2,'AND',@
    where_stmt_3,'AND',@where_stmt_4);
```

./mysql_resources/stored_procs.sql

```
590 ELSEIF atttype = 'network'
591 THEN SET @select_stmt = CONCAT_WS(' ',@initial_stmt,
592 'JOIN network net
593 ON a.object_id = net.network_id
594 WHERE',@where_stmt,'AND',@where_stmt_2,'AND',@
where_stmt_4);
595 ELSE SET @select_stmt = CONCAT_WS(' ',@initial_stmt,
596 'WHERE',@where_stmt);
597 END IF;
598
599 PREPARE exec_stmt FROM @select_stmt;
600 EXECUTE exec_stmt;
601
602 END //
```

```
603
604 DROP PROCEDURE IF EXISTS `DeleteEdgeInNetwork` //
```

```
605 CREATE PROCEDURE DeleteEdgeInNetwork(
606 IN netname VARCHAR(12),
607 IN sourcenode VARCHAR(75),
608 IN targetnode VARCHAR(75),
609 IN edgeweight DOUBLE,
610 IN weightdirection ENUM('<','>','<=','>='),
611 IN edgeweightatt VARCHAR(100)
612 )
613 interior:BEGIN
614
615 IF netname IS NOT NULL
616 THEN SET @where_stmt = CONCAT('net.network_name = "',
netname,'"');
617 ELSE SET @where_stmt = '1 = 1';
618 END IF;
619
620 IF sourcenode IS NOT NULL
621 THEN SET @where_stmt_2 = CONCAT('n1.node_hash = "',
sourcenode,'"');
622 ELSE SET @where_stmt_2 = '1 = 1';
623 END IF;
624
625 IF targetnode IS NOT NULL
626 THEN SET @where_stmt_3 = CONCAT('n2.node_hash = "',
targetnode,'"');
627 ELSE SET @where_stmt_3 = '1 = 1';
628 END IF;
629
```

```
630 IF edgeweight IS NULL OR weightdirection IS NULL OR
    edgeweightatt IS NULL
631 THEN SET @delete_stmt = CONCAT_WS(' ', 'DELETE en.*
632 FROM edge2network AS en, network AS net, edge AS e, node
    AS n1, node AS n2
633 WHERE e.edge_id = en.edge_id AND en.network_id =
    net.network_id AND
634 n1.node_id = e.source_node_id AND n2.node_id =
    e.target_node_id AND',
635 @where_stmt, 'AND', @where_stmt_2, 'AND', @where_stmt_3);
636 PREPARE exec_stmt FROM @delete_stmt;
637 EXECUTE exec_stmt;
638 LEAVE interior;
639 END IF;
640
641 SET @attributetype = NULL;
642 SELECT data_type FROM attribute_description WHERE
    attribute_name = edgeweightatt AND attribute_object =
    'edge' LIMIT 1 INTO @attributetype;
643 IF @attributetype IS NULL OR @attributetype = 'string'
644 THEN LEAVE interior;
645 END IF;
646
647 SET @where_stmt_4 = CONCAT('a_d.attribute_name = "',
    edgeweightatt, '" AND a.attribute_value ', weightdirection,
    ' ', edgeweight);
648
649 SET @delete_stmt = CONCAT_WS(' ', 'DELETE en.*
650 FROM edge AS e, attribute AS a, attribute_description AS
    a_d, node AS n1,
651 node AS n2, edge2network AS en, network AS net
652 WHERE a.object_id = e.edge_id AND a_d.attribute_id =
    a.attribute_id AND a_d.attribute_object = "edge"
653 AND n1.node_id = e.source_node_id AND n2.node_id =
    e.target_node_id AND
654 AND e.edge_id = en.edge_id AND en.network_id =
    net.network_id AND',
655 @where_stmt, 'AND', @where_stmt_2, 'AND', @where_stmt_3, 'AND'
    , @where_stmt_4);
656 PREPARE exec_stmt FROM @delete_stmt;
657 EXECUTE exec_stmt;
658 END //
659
660 DROP PROCEDURE IF EXISTS `DeleteNodeInNetwork` //
```

./mysql_resources/stored_procs.sql

```
661 CREATE PROCEDURE DeleteNodeInNetwork(  
662 IN netname VARCHAR(12),  
663 IN nodehash VARCHAR(75)  
664 )  
665 BEGIN  
666  
667 IF netname IS NOT NULL  
668 THEN SET @where_stmt = CONCAT('net.network_name = "',  
netname, '"');  
669 ELSE SET @where_stmt = '1 = 1';  
670 END IF;  
671  
672 IF nodehash IS NOT NULL  
673 THEN SET @where_stmt_2 = CONCAT('nd.node_hash = "',  
nodehash, '"');  
674 ELSE SET @where_stmt_2 = '1 = 1';  
675 END IF;  
676  
677 SET @delete_stmt = CONCAT_WS(' ', 'DELETE nn.*  
678 FROM node as nd  
679 LEFT JOIN (node2network nn, network net)  
680 ON (nd.node_id = nn.node_id AND net.network_id =  
nn.network_id)  
681 WHERE', @where_stmt, 'AND', @where_stmt_2);  
682 PREPARE exec_stmt FROM @delete_stmt;  
683 EXECUTE exec_stmt;  
684  
685 END //  
686  
687 DROP PROCEDURE IF EXISTS `DeleteNode` //  
688 CREATE PROCEDURE DeleteNode(  
689 IN nodehash VARCHAR(75),  
690 IN attributename VARCHAR(100),  
691 IN identiervalue VARCHAR(100)  
692 )  
693 interior:BEGIN  
694  
695 SET @nodehash2 = NULL;  
696 IF attributename IS NOT NULL AND identiervalue IS NOT  
NULL  
697 THEN SELECT nd.node_hash FROM identifier as i  
698 JOIN attribute_description a_d  
699 ON a_d.attribute_id = i.attribute_id  
700 JOIN node nd
```

./mysql_resources/stored_procs.sql

```
701 ON nd.node_id = i.node_id
702 WHERE a_d.attribute_object = 'identifier' AND a_d.
attribute_name = attributename AND i.identifier_value =
identifiervalue
703 LIMIT 1 INTO @nodehash2;
704 END IF;
705
706 IF nodehash IS NOT NULL AND @nodehash2 IS NOT NULL AND
nodehash != @nodehash2
707 THEN LEAVE interior;
708 ELSEIF nodehash IS NULL AND @nodehash2 IS NOT NULL
709 THEN SET nodehash = @nodehash2;
710 END IF;
711
712 CALL deleteNodeInNetwork(NULL,nodehash);
713 CALL deleteAttribute(NULL,nodehash,NULL,NULL,'node');
714 CALL deleteEdge(nodehash,NULL);
715 CALL deleteEdge(NULL,nodehash);
716
717 SET @where_stmt = CONCAT('nd.node_hash = "',nodehash,'"');
718 IF @where_stmt IS NULL
719 THEN SET @where_stmt = '1 = 1';
720 END IF;
721
722 SET @delete_stmt = CONCAT('DELETE i.*
723 FROM node AS nd, identifier AS i
724 WHERE i.node_id = nd.node_id AND ',@where_stmt);
725 PREPARE exec_stmt FROM @delete_stmt;
726 EXECUTE exec_stmt;
727
728 SET @delete_stmt_2 = CONCAT('DELETE nd.* FROM node AS nd
729 WHERE ',@where_stmt);
730 PREPARE exec_stmt_2 FROM @delete_stmt_2;
731 EXECUTE exec_stmt_2;
732
733 END //
734
735 DROP PROCEDURE IF EXISTS `DeleteNetwork` //
736 CREATE PROCEDURE DeleteNetwork(
737 IN netname VARCHAR(20)
738 )
739 BEGIN
740 CALL deleteNodeInNetwork(netname,NULL);
```


./mysql_resources/stored_procs.sql

```
741  CALL deleteEdgeInNetwork(netname,NULL,NULL,NULL,NULL,
    );
742  CALL deleteAttribute(netname,NULL,NULL,NULL,'network');
743
744  SET @where_stmt = CONCAT('network_name = "',netname,'"');
745  IF @where_stmt IS NULL
746  THEN SET @where_stmt = '1 = 1';
747  END IF;
748
749  SET @select_stmt = CONCAT_WS(' ', 'DELETE FROM network
    WHERE',@where_stmt);
750  PREPARE exec_stmt FROM @select_stmt;
751  EXECUTE exec_stmt;
752
753  END //
754
755  DROP PROCEDURE IF EXISTS `DeleteEdge` //
756  CREATE PROCEDURE DeleteEdge(
757  IN sourcenode VARCHAR(75),
758  IN targetnode VARCHAR(75)
759  )
760  interior:BEGIN
761
762  CALL deleteEdgeInNetwork(NULL,sourcenode,targetnode,NULL,
    NULL,NULL);
763  CALL deleteAttribute(NULL,sourcenode,targetnode,NULL,
    'edge');
764
765  IF sourcenode IS NOT NULL
766  THEN SET @where_stmt = CONCAT('n1.node_hash = "',
    sourcenode,'"');
767  ELSE SET @where_stmt = '1 = 1';
768  END IF;
769
770  IF targetnode IS NOT NULL
771  THEN SET @where_stmt_2 = CONCAT('n2.node_hash = "',
    targetnode,'"');
772  ELSE SET @where_stmt_2 = '1 = 1';
773  END IF;
774
775  SET @delete_stmt = CONCAT_WS(' ', 'DELETE e.*
    FROM edge AS e, node AS n1, node AS n2
    WHERE n1.node_id = e.source_node_id AND n2.node_id =
    e.target_node_id
```

./mysql_resources/stored_procs.sql

```
778 AND',@where_stmt,'AND',@where_stmt_2);
779 PREPARE exec_stmt FROM @delete_stmt;
780 EXECUTE exec_stmt;
781
782
783 END //
784
785 DROP PROCEDURE IF EXISTS `ClearAll` //
786 CREATE PROCEDURE ClearAll()
787
788 BEGIN
789
790 DELETE FROM node;
791 DELETE FROM edge;
792 DELETE FROM attribute;
793 DELETE FROM network;
794 DELETE FROM node2network;
795 DELETE FROM edge2network;
796 DELETE FROM attribute_description;
797 DELETE FROM identifier;
798 ALTER TABLE node AUTO_INCREMENT = 1;
799 ALTER TABLE edge AUTO_INCREMENT = 1;
800 ALTER TABLE network AUTO_INCREMENT = 1;
801 ALTER TABLE attribute_description AUTO_INCREMENT = 1;
802
803 END //
```

C.4. ./parallel_resources modules

./parallel_resources/parallel_blast2seq.py

```
1  # Author: Alan Barber
2  # Created on: November 16, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: The file parallel_blast2seq.py is used to
   calculate these edges and requires
7  that the BLAST+ v.2.2.25 or above be installed on the
   machine to use it. You can download
8  this package from:
   http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=
   BlastDocs&DOC_TYPE=Download.
9  You must have at least Python 2.6 installed to use this
   program.
10
11  Benchmarks done on the performance of this code have
   shown that on a machine with an Intel Xeon 2.67
12  Ghz processor, approximately 600 edges can be calculated
   per minute per core used. This translates to
13  approximately 864,000 edges per day per core used. Keep
   this in mind when deciding how many threads to
14  use when calculating what size of network. For rapid
   calculation of large networks (> 10 million edges),
15  it is a good idea to have at least a hundred processor
   cores available.
16
17  There are two ways to run this script, the first is on a
   machine where you have access to many cores in
18  the shell. The second is to use a queuing system. If you
   indicate that you only want to calculate on 1
19  processor, then the script can be used effectively with
   HPC management systems like Sun Grid Engine with
20  an appropriate submission script and one table file per
   processor. The contents of that submission script
21  will depend on the specific setup of your cluster, so
   contact your system administrator for assistance.
22
23  Do not have any files in the run directory that have the
   table file extension other than the table files
24  or the script will run into problems.
25
26  Much of the code was taken from various parts of
   Pythoscape and put into one file for easy execution.
```

`./parallel_resources/parallel_blast2seq.py`

```
27
28 Functions:
29 find_files - Returns the full path of all files in a
    directory with a given extension.
30 make_data_table - Prints a data table file
31 write_fasta_file - Creates a fasta file
32 read_in_fa - Reads in a fasta file
33 apply_filter_to_data - Takes two parsed BLAST dictionary
    results and applies filter
34 parse_bl2seq - Parses the output of a bl2seq file into a
    dictionary
35 run_bl2seq - Returns a dictionary of information from a
    pairwise BLAST comparison
36 run_bl2seq_star - Runs a bl2seq job using a list of
    arguments
37 parse_bl2seq_star - Runs a parse_bl2seq job using a list
    of arguments
38 make_seqs - Makes sequence dictionaries for run_bl2seq
    from infile and a sequence dictionary
39 run_blasts - Runs all blasts using an edge infile
40
41 Classes:
42 TextTags - Pythoscape text parsing engine
43
44 Module attributes:
45 COLUMNS - list of columns of information that are pulled
    from BLAST output
46
47 Usage:
48 python parallel_blast2seq.py <# processors> <seq_file>
    <table_file_ext> <output_file_ext>
49 <Optional: --gap_open value> <Optional: --gap_extend
    value> <Optional: --comp_based True/False>
50 <Optional: --use_best True/False> <Optional: --filter
    -log10(E)_filter>
51 <Optional: --blastp_exe blastp_exe>
52
53 Required arguments:
54 <# processors> - number of processors to use to
    calculate edges
55 <seq_file> - file of fasta formatted sequences outputted
    from Pythoscape, called sequences.fa by default
56 <table_file_ext> - extension of table files, all files
    with this extension in the directory will be assumed
```

`./parallel_resources/parallel_blast2seq.py`

```
57         to be table files for the run
58 <output_file_ext> - extension of output files to create,
    'out' (no quotes) is a good idea.
59
60 Optional arguments:
61 --gap_open: value for gap open penalty, 11 by default
62 --gap_extend: value for gap extend penalty, 1 by default
63 --comp_based: composition based statistics, True by
    default
64 --use_best: use the best edge or worst edge of the pair
    calculated
65         between two nodes (they are asymmetric if
    composition based statistics is on), False by default
66 --filter: only write out edges that have a -log10(E)
    value greater than the value given, default None
67 --blastp_exe: the location of the blastp executable, the
    script assumes that blastp is in the directory.
68         ./blastp by default
69
70 Example execution:
71 python parallel_blast2seq.py 6 sequences.fa tab out
72
73 This would run edges on 6 cores using the sequence file
    sequences.fa, all files named .tab in the
74 directory and output corresponding .out files.
75 '''
76
77 import sys
78 import subprocess
79 import os
80 import math
81 import multiprocessing
82 import itertools
83
84 COLUMNS = ['bit_score', 'm*n', 'alignment_len',
    'alignment_ids',
85             'query_start', 'query_end', 'subject_start',
    'subject_end']
86
87 def find_files(ext, directory):
88     '''Returns the full path of all files in a directory
    with a given extension.
89
90     Arguments:
```

./parallel_resources/parallel_blast2seq.py

```
91         ext - file extension of files to find
92         directory - directory to search
93
94     Returns:
95         list of full file paths
96     '''
97     return filter(lambda x: len(x) > len(ext)+1 and x[-
len(ext)-1:] == '{0}'.format(ext),
98                 ['{0}/{1}'.format(directory,item) for
item in os.listdir(directory)])
99
100 def make_data_table(outfile,column_size,rows,columns,
matrix,csv=False):
101     '''Prints a data table file
102
103     Arguments:
104         outfile - name of file to output
105         column_size - column justification (except
for first)
106         rows - sequence of all row key names
107         columns - sequence of all column key names
108         matrix - dictionary of sequence values (e.g.
list) indexed by (row,column)
109
110     Arguments (with defaults):
111         csv (default is false) - if True, creates a csv
file instead of a space delimited table
112
113     Example output File:
114         Column 1    Column 2    Column 3
115     Row 1    Value 1,1    Value 1,2    Value 1,3a
116                                     Value 1,3b
117     Row 2                                     Value 2,2
118     (etc...)
119     '''
120     f = open(outfile,'w')
121     first_width = max([len(str(row)) for row in rows])
122     if not csv:
123         first_row = '{0:>{width}}'.format(' ',width=
first_width)
124         first_row = ''.join([first_row]+'{0:>{width}}'.
format(column,width=column_size)
125                                for column in
columns)+'\n'])
```

./parallel_resources/parallel_blast2seq.py

```
126         else:
127             first_row = ','.join([' ']+[str(column) for
column in columns]+['\n'])
128             f.write(first_row)
129             for row_id in rows:
130                 start,i = True,0
131                 while 1:
132                     to_continue = False
133                     if start and not csv:
134                         row = '{0:>{width}}'.format(row_id,width=
first_width)
135                     elif start and csv:
136                         row = '{0}'.format(row_id)
137                     elif not start and not csv:
138                         row = '{0:>{width}}'.format(' ',width=
first_width)
139                     else:
140                         row = ' '
141                     column_data = []
142                     for column in columns:
143                         if (row_id,column) in matrix:
144                             if len(matrix[row_id,column]) > i:
145                                 column_data.append(matrix[row_id,
column][i])
146                         else:
147                             column_data.append(' ')
148                         if len(matrix[row_id,column]) > i+1:
149                             to_continue = True
150                     else:
151                         column_data.append(' ')
152                     if not csv:
153                         row = ''.join([row]+['{0:>{width}}'.
format(column,width=column_size)
154                                     for column in
column_data]+['\n'])
155                     else:
156                         row = ','.join([row]+[str(column) for
column in column_data])
157                         row = '{0}\n'.format(row)
158                     f.write(row)
159                     if not to_continue:
160                         break
161                     start = False
162                     i += 1
```


./parallel_resources/parallel_blast2seq.py

```
163     f.close()
164
165     def write_fasta_file(fa_file, seqs, file_exists=False):
166         ''' Creates a fasta file -- of the form >title
167         sequence
168
169         It creates the fasta file named using fa_file using
170         the dictionary-like
171         seqs, where the key is the title and the value is
172         the sequence.
173
174         Arguments:
175             fa_file - name of fasta file to output
176             seqs - dictionary of output sequences
177
178         Arguments (with defaults):
179             file_exists (default is False) - if true adds
180             the sequence to existing file
181         '''
182
183         option = 'a' if file_exists else 'w'
184         with open(fa_file, option) as f:
185             for seq in seqs:
186                 f.write('>{0}\n{1}\n'.format(seq, seqs[seq]))
187
188     def read_in_fa(fa_file):
189         '''Reads in a single fasta file.
190
191         It returns a dictionary of sequences where the key
192         is the title and the value is the sequence.
193
194         Arguments:
195             fa_file - name of fasta file to input
196
197         Returns:
198             dictionary where the key is the title and the
199             value is the sequence
200         '''
201
202         f = open(fa_file)
203         seqs = {}
204         for line in f:
205             line = line.strip()
206             if line and line[0] == '>':
```

./parallel_resources/parallel_blast2seq.py

```
201         curr_seq = line[1:]
202         seqs[curr_seq] = ''
203     elif line:
204         seqs[curr_seq] = ''.join([seqs[curr_seq], line
205 ])
206     return seqs
207
208 class TextTags(object):
209     '''TextTags takes a text string and tags lines based
210     on the parameters
211     used it set_position_tags. Line splitting tags can
212     be set using
213     set_split_tags. Iterating through the class gives
214     dictionaries
215     from each line of the defined data.
216
217     Public methods:
218     set_position_tag - sets a line tag that indicates
219     data positions
220     set_split_tag - sets a data tag to a line that
221     indicates how to find it
222     reset - resets internal iterator to first line
223
224     Example Usage:
225     my_tt = TextTags(['* This sentence starts with stars
226     and has a 1 = 1 equation.'])
227     my_tt.set_position_tag('double star','**',go_for=1)
228     my_tt.set_split_tag('double
229     star',[( 'equals',10)],splits=[' '])
230     for item in my_tt:
231         print item
232     '''
233
234     #Parameters to be saved from set_position_tag
235     POSITION_PARAMS = ['name','start_tag','lines_to_data',
236                       'stop_tagging','multiple_tags',
237                       'empty_tag',
238                       'empty_in_multi','go_for','z']
239
240     #Parser states set in _match_to_tag
241     SAVED_STATES = ['start','started','keep_going',
242                     'repeater','saved_lines','stop']
243
244     def __init__(self,data,split_on=None):
```

./parallel_resources/parallel_blast2seq.py

```
235         '''Create a new instance of TextBlock.
236
237         Arguments:
238             data -- string or list of strings
239
240         Arguments (with defaults):
241             split_on (default is None): If a single
string is inputed, this is how to split it
242                                     to prepare the
list of strings. If None, no split is
243                                     performed
244         '''
245
246         if split_on: data = data.split(split_on)
247         self.data = list(data)
248         self._line_tags = {}
249         self._split_tags = {}
250         self._pulled_tags = {}
251         self._data_tags = [None]*len(self.data)
252         self._lp = {}
253         self._lpp = dict([(pos_param,i) for i,pos_param
in enumerate(self.POSITION_PARAMS)])
254         self._ls = {}
255         self._lsp = dict([(state,i) for i,state in
enumerate(self.SAVED_STATES)])
256         self._i = 0
257         self._zs = {}
258         self._current_tag = None
259         self._errors = set([])
260
261     def __str__(self):
262         '''Overwrites __str__ when printing TextTags
263
264         Returns:
265             a description of how each line will be parsed
266         '''
267
268         out = []
269         for return_data in self:
270             if return_data:
271                 out_data = '\n'.join(['{0}:{1}'.format(dp
,return_data[dp]) for dp in return_data]+[''])
272             else:
273                 out_data = ''
```

./parallel_resources/parallel_blast2seq.py

```
274         out = out + ['Line:\n{0}\n'.format(self.data[
self._i]),
275                                     'Assigned tag:\n{0}\n'.format(
self._current_tag),
276                                     'Parsed data:\n',out_data,'\n']
277         self.reset()
278         return ''.join(out)
279
280     def __iter__(self):
281         '''Overwrites __iter__ to yield all data
dictionaries parsed from text.
282
283         Yields:
284             dictionary of {data_name:data} of all parsed
hits in each line
285         '''
286
287         for line in self.data:
288             self._get_tag()
289             if self._current_tag:
290                 yield self._make_data_dict()
291             else:
292                 yield {}
293             self._i += 1
294
295     def _make_data_dict(self):
296         '''Private function that makes a the output
dictionary from each line.
297
298         Do not call as this function is subject to
removal or change!'''
299
300         outline = [self.data[self._i]]
301         if self._current_tag in self._split_tags:
302             for split_on in self._split_tags[self.
_current_tag]:
303                 outline = [item.strip().split(split_on)
for item in outline]
304                 outline = [item for item in itertools.
chain(*outline) if item != '']
305                 try:
306                     return dict([(name,outline[location]) for
name,location
307                                     in self._pulled_tags[self.
```

./parallel_resources/parallel_blast2seq.py

```

    _current_tag]])
308         except IndexError:
309             if self._current_tag in self._errors:
310                 return {}
311             else:
312                 try:
313                     raise IndexError;
314                 except IndexError as err:
315                     if not err.args:
316                         err.args=(' ',)
317                     err.args = (err.args[0] + ""
318 The split definition given: {0}
319 cannot be performed on line: {1}
320 "".format(self._pulled_tags[self._current_tag],outline
321 ),)+err.args[1:]
322                 raise
323             elif self._current_tag in self._pulled_tags:
324                 line = outline[0].strip()
325                 return {self._pulled_tags[self._current_tag][
326 0][0]:line}
327             else:
328                 return {}
329
330     def _apply_z(self, tag_set):
331         '''Private function that figures out tag with
332 highest z.
333
334 Do not call as this function is subject to
335 removal or change!'''
336
337         tag_set = filter(lambda x: x != None, tag_set)
338         if len(tag_set) == 1:
339             return tag_set[0]
340         elif len(tag_set) == 0:
341             return None
342         z_set = [self._zs[tag] for tag in tag_set]
343         max_tag = max(z_set)
344         if z_set.count(max_tag) > 1:
345             raise AmbiguousZ(self.data[self._i], tag_set)
346         return tag_set[z_set.index(max_tag)]
347
348     def _get_tag(self):
349         '''Private function that sets the current
350 position tag.
```

./parallel_resources/parallel_blast2seq.py

```
346
347         Do not call as this function is subject to
removal or change!'''
348
349         if self._i == 0:
350             for tn in self._ls:
351                 self._ls[tn][self._lsp['start']] = False
352                 self._ls[tn][self._lsp['saved_lines']] =
self._lp[tn][self._lpp['lines_to_data']]*1
353                 if self._lp[tn][self._lpp['multiple_tags'
]]:
354                     self._ls[tn][self._lsp['repeater']] =
len(self._lp[tn][self._lpp['name']])
355                     for tp in self._lp:
356                         if self._lp[tp][self._lpp['multiple_tags'
]]:
357                             for one_tag in tp:
358                                 self._zs[one_tag] = self._lp[tp][
self._lpp['z']]
359                             else:
360                                 self._zs[tp] = self._lp[tp][self._lpp
['z']]
361                     if self._i < len(self.data):
362                         return_data = map(self._match_to_tag,[
tag_name for tag_name in self._lp])
363                         self._current_tag = self._apply_z(return_data)
364
365         def _get_run_info(self,tag):
366             '''Private function that retrieves the position
tagging state.
367
368             Do not call as this function is subject to
removal or change!'''
369
370             return_data = dict([(name,self._ls[tag][self._lsp
[name]]) for name in self._lsp])
371             return_data.update(dict([(name,self._lp[tag][self
._lpp[name]]) for name in self._lpp]))
372             return return_data
373
374         def _update_run_info(self,tag,tag_info):
375             '''Private function that updates the current
position tagging state.
376
```

./parallel_resources/parallel_blast2seq.py

```
377         Do not call as this function is subject to
removal or change!'''
378
379         for i,state in enumerate(self.SAVED_STATES):
380             self._ls[tag][i] = tag_info[state]
381
382         def _match_to_tag(self,tag):
383             '''Private function that matches a line to a
given position tag.
384
385             Do not call as this function is subject to
removal or change!'''
386
387             return_tag = None
388             run_ps = self._get_run_info(tag)
389             line = self.data[self._i]
390             if line.startswith(run_ps['start_tag']):
391                 run_ps['start'],run_ps['started'] = True,True
392                 run_ps['keep_going'] = run_ps['go_for']
393                 if run_ps['saved_lines'] != ['lines_to_data']:
394                     run_ps['saved_lines'] = run_ps[
'lines_to_data']*1
395                 if not run_ps['saved_lines'] and run_ps[
'stop_tagging'] != None\
396                     and line.startswith(run_ps['stop_tagging'])
and run_ps['started']:
397                     run_ps['stop'] = True
398                     if run_ps['start'] and not run_ps['stop']:
399                         if not run_ps['saved_lines']:
400                             do_return = True
401                             if not line.strip():
402                                 do_return = False
403                             if run_ps['empty_tag'] or (run_ps[
'empty_in_multi'] and\
404                                 run_ps['multiple_tags']):
405                                 do_return = True
406                             if run_ps['keep_going'] == 1:
407                                 run_ps['start'] = False
408                             elif run_ps['keep_going']:
409                                 run_ps['keep_going'] -= 1
410                             if not run_ps['multiple_tags'] and
do_return:
411                                 return_tag = run_ps['name']
412                             elif do_return:
```

./parallel_resources/parallel_blast2seq.py

```
413         return_tag = run_ps['name'][len(
run_ps['name'])-run_ps['repeater']]
414         run_ps['repeater'] -= 1
415     else:
416         run_ps['saved_lines'] -= 1
417     if run_ps['repeater'] == 0:
418         run_ps['repeater'] = len(run_ps['name'])
419     self._update_run_info(tag, run_ps)
420     if return_tag != None:
421         return return_tag
422
423     def set_position_tag(self, name, start_tag,
lines_to_data=0,
424                         stop_tagging=None, multiple_tags=
False, empty_tag=False,
425                         empty_in_multi=False, go_for=None
, z=0):
426         '''Sets a position tag, which tags a given line
as a certain type of line
427         given the parameters.
428
429         Arguments:
430         name - tag name or list of names (if using a
list of names, multiple_tags
431             must be set to True)
432         start_tag - where to start tagging lines with
the tag (if this is set,
433             but no other optional arguments are
set, then all lines
434             after the first start hit will be
tagged)
435
436         Arguments (with defaults):
437         lines_to_data (default 0) - how many lines after
the line with the start_tag
438             to start tagging
439         stop_tagging (default None) - line start that
indicates to stop tagging
440         multiple_tags (default False) - name is not one
tag, but a list of tags
441             that will be
used in order
442         empty_tag (default False) - empty lines won't be
tagged if False
```


./parallel_resources/parallel_blast2seq.py

```
443         empty_in_multi (default False) - empty lines
will be skipped in a multiple
444                                     tag set if set
to False
445         go_for (default None) - if set to a number, this
will limit the number of lines
446                                     tagged to the number given
447         z (default 0) - z of the tag. Only one position
tag can be given per line,
448                                     the tag with the highest z will
be the set tag.
449         '''
450         if multiple_tags:
451             name = tuple(name)
452             self._lp[name] = [None]*len(self.POSITION_PARAMS)
453         for i,parameter in enumerate(self.POSITION_PARAMS
):
454             if parameter in locals():
455                 self._lp[name][i] = locals()[parameter]
456             self._ls[name] = [None]*len(self.SAVED_STATES)
457
458     def reset(self):
459         '''Resets the iterator to the start.
460
461         Arguments:
462             None'''
463         self._i = 0
464         for name in self._lp:
465             self._ls[name] = [None]*len(self.SAVED_STATES)
466
467     def set_split_tag(self,name,pulled_elements,splits=
None,no_errors=False):
468         '''Sets a split tag, which indicates how to find
the data point
469         that is being searched for.
470
471         Arguments:
472             name - position tag to use for split tag
473             pulled_elements - list of a set of tuples of the
following form,
474             [(split_tag_name,split_tag_location)]
475
476         Arguments (with defaults):
```

./parallel_resources/parallel_blast2seq.py

```
477         splits (default None) - set of splits to use on
the line, used in order
478                                     to the flattened set of
all data
479                                     If this is None and if
you want the whole line
480                                     pulled_elements should
look like:
481                                     [(split_tag_name,0)]
482         no_errors (default False) - if True, instead of
throwing an error when
483                                     the line can't be
parsed, it returns an empty
484                                     dictionary
485         '''
486
487         if splits != None:
488             self._split_tags[name] = splits
489         if no_errors:
490             self._errors.add(name)
491         self._pulled_tags[name] = pulled_elements
492
493     def apply_filter_to_data(options,use_best=False,
val_filter=None):
494         '''Takes two parsed BLAST dictionary results and
applies filter
495
496         Arguments:
497             options - tuple of two parsed BLAST dictionaries
from parse_bl2seq
498
499         Arguments (with defaults):
500             use_best (default False) - use the best e-value
or the worse e-value.
501                                     Default of False uses
the worse one.
502             val_filter (default None) - if this is set,
result is only returned
503                                     if -log10(E) is
above this amount.
504
505         Returns:
506             single dictionary or None, if neither meet
criteria
```

./parallel_resources/parallel_blast2seq.py

```
507     '''
508
509     evals = [None, None]
510     for i, option in enumerate(options):
511         if 'm*n' in option and 'bit_score' in option:
512             test_val = -math.log10(float(option['m*n']))
513 + float(option['bit_score'])*math.log10(2)
514             if val_filter != None and test_val <
515 val_filter:
516                 pass
517             else:
518                 evals[i] = test_val
519
520     if evals[0] == None and evals[1] == None:
521         return None
522     elif None in evals:
523         return options[evals.index(evals[0] or evals[1])]
524     elif evals[1] < evals[0]:
525         options = [options[1], options[0]]
526     if use_best:
527         return options[1]
528     else:
529         return options[0]
530
531 def parse_bl2seq(bl2seq_outs, use_best=False, val_filter=
532 None):
533     ''' Parses the output of a bl2seq file into a
534     dictionary
535
536     Arguments:
537         bl2seq_out - an iterator that yields the lines
538     of a blast output
539
540     Arguments (with defaults):
541         use_best (default False) - use the best e-value
542     or the worse e-value.
543
544         Default of False uses
545     the worse one.
546
547         val_filter (default None) - if this is set,
548     result is only returned
549
550         if -log10(E) is
551     above this amount.
552
553     Returns:
```

./parallel_resources/parallel_blast2seq.py

```
542         dictionary of data from the bl2seq file
543     '''
544
545     options = []
546     for tt in bl2seq_outs:
547         tt.set_position_tag('query', 'Query=', go_for=1)
548         tt.set_position_tag('subject', 'Subject=', go_for=1)
549         tt.set_position_tag('score line', ' Score',
stop_tagging=' Identities')
550         tt.set_position_tag('identities line', '
Identities', go_for=1, stop_tagging='Query ')
551         tt.set_position_tag('query lines', 'Query ', go_for
=1, stop_tagging=' Score')
552         tt.set_position_tag('subject lines', 'Sbjct',
go_for=1, stop_tagging=' Score')
553         tt.set_position_tag('search space', 'Effective',
go_for=1)
554         tt.set_split_tag('query', [('query', 1)], splits=['
'])
555         tt.set_split_tag('subject', [('subject', 1)], splits
=[' '])
556         tt.set_split_tag('score line', [('bit_score', 2)],
splits=[' '])
557         tt.set_split_tag('identities line', [(
'alignment_ids', 2), ('alignment_len', 3)], splits=[' ', '/'])
558         tt.set_split_tag('query lines', [('query_start', 1
), ('query_end', 3)], splits=[' '], no_errors=True)
559         tt.set_split_tag('subject lines', [(
'subject_start', 1), ('subject_end', 3)], splits=[' '],
no_errors=True)
560         tt.set_split_tag('search space', [('m*n', 4)],
splits=[' '])
561
562         all_data = {}
563         for parsed_line in tt:
564             if 'query_start' in parsed_line and
'query_start' in all_data:
565                 del parsed_line['query_start']
566             if 'subject_start' in parsed_line and
'subject_start' in all_data:
567                 del parsed_line['subject_start']
568             all_data.update(parsed_line)
569             options.append(all_data)
570
```

./parallel_resources/parallel_blast2seq.py

```
571     best = apply_filter_to_data(options,use_best,
val_filter)
572     if best:
573         return_data = {}
574         title = '{0} {1}'.format(best['query'],best[
'subject'])
575         for datap in COLUMNS:
576             return_data[title,datap] = best[datap]
577         return title,return_data
578     else:
579         return None
580
581 def run_bl2seq(seqs,blast_exe=None,gap_open=None,
gap_extend=None,no_comp_based=False):
582     ''' Returns a dictionary of information about the
worst (or best score) from a pairwise
583     BLAST comparison
584
585     Arguments:
586         seqs - dictionary of the two sequences to compare
587         blast_exe - where the blastp executable is
588
589     Arguments (with defaults):
590         gap_open (default is 11) - gap open penalty
591         gap_extend (default is 1) - gap extend penalty
592         no_comp_based (default is False) - if True,
turns off composition based statistics
593         use_best (default is False) - if True, returns
info on best scoring comparison
594
595     Returns:
596         dictionary of data from the bl2seq file
597     '''
598
599     gap_open = gap_open if gap_open != None else '11'
600     gap_extend = gap_extend if gap_extend != None else '1'
601     comp_based = '0' if no_comp_based else '2'
602     blast_exe = blast_exe if blast_exe != None else
'./blastp'
603     seq_names = seqs.keys()
604     fn_1 = ''.join([seq_names[0][0:10],seq_names[1][0:10
],'-1.fa'])
605     fn_2 = ''.join([seq_names[0][0:10],seq_names[1][0:10
],'-2.fa'])
```

./parallel_resources/parallel_blast2seq.py

```
606     write_fasta_file(fn_1,{seq_names[0]:seqs[seq_names[0]
    ]})
607     write_fasta_file(fn_2,{seq_names[1]:seqs[seq_names[1]
    ]})
608     cline = '{0} -query {1} -subject {2} -gapopen {3}
    -gapextend {4} -comp_based_stats {5} -use_sw_tback'.
    format(blast_exe,fn_1,fn_2,gap_open,gap_extend,comp_based)
609     child = subprocess.Popen(cline,shell=True,stdout=
    subprocess.PIPE,close_fds=True)
610     my_out,my_err = child.communicate()
611     cline = '{0} -query {1} -subject {2} -gapopen {3}
    -gapextend {4} -comp_based_stats {5} -use_sw_tback'.
    format(blast_exe,fn_2,fn_1,gap_open,gap_extend,comp_based)
612     child = subprocess.Popen(cline,shell=True,stdout=
    subprocess.PIPE,close_fds=True)
613     my_out_2,my_err = child.communicate()
614     os.remove(fn_1)
615     os.remove(fn_2)
616     return my_out,my_out_2
617
618 def run_bl2seq_star(args):
619     ''' Runs a bl2seq job using a list of arguments --
    this is for proper parallelization
620
621     Arguments:
622         args - list of args
623     '''
624
625     return run_bl2seq(*args)
626
627 def parse_bl2seq_star(args):
628     ''' Runs a parse_bl2seq job using a list of
    arguments -- this is for proper parallelization
629
630     Arguments:
631         args - list of args
632     '''
633
634     return parse_bl2seq(*args)
635
636 def make_seqs(infile,seqs):
637     '''Makes sequence dictionaries for run_bl2seq from
    infile and the total sequence dictionary
638
```

./parallel_resources/parallel_blast2seq.py

```
639     Arguments:
640         infile - name of edge list file
641         seqs - sequence dictionary containing all
sequences
642
643     Returns:
644         list of 2 sequence dictionaries that correspond
to infile
645     '''
646
647     seq_groups = []
648     with open(infile) as f:
649         for line in f:
650             line = line.strip().split()
651             seq_groups.append({line[0]:seqs[line[0]], line
[1]:seqs[line[1]]})
652     return seq_groups
653
654 def run_blasts(infile,outfile,p_num,seqs,gap_open,
gap_extend,no_comp_based,val_filter,use_best,blastp_exe):
655     '''Runs a set of bl2seq runs given an edge infile
656
657     Arguments:
658         infile - name of edge list file
659         outfile - name of file to output with results
660         p_num - number of processors to use
661         seqs - dictionary of sequences
662         gap_open - gap open penalty
663         gap_extend - gap extend penalty
664         no_comp_based - turn off/on composition based
statistics
665         val_filter - -log10(e-value) filter to apply,
None if none
666         use_best - use the best or worse e-value, True
if best
667         blastp_exe - location of blastp executable from
BLAST+
668
669     '''
670
671     if p_num > 1:
672         pool = multiprocessing.Pool(processes=p_num)
673     elif p_num < 1:
674         print 'Number of processors greater than 0 must
```

./parallel_resources/parallel_blast2seq.py

```
be set.'
675     return None
676     repeat_args = []
677     for arg in [blastp_exe, gap_open, gap_extend,
no_comp_based, use_best, val_filter]:
678         repeat_args.append(itertools.repeat(arg))
679     blast_args = [make_seqs(infile, seqs)] + repeat_args[0
:4]
680     if p_num > 1:
681         blast_results = pool.map(run_bl2seq_star,
itertools.izip(*blast_args))
682     else:
683         blast_results = map(run_bl2seq_star, itertools.
izip(*blast_args))
684     blast_results = [[(TextTags(result_1, split_on='\n'),
TextTags(result_2, split_on='\n')) for result_1, result_2
in blast_results]]
685     parse_args = blast_results + repeat_args[4:]
686     if p_num > 1:
687         parsed_results = pool.map(parse_bl2seq_star,
itertools.izip(*parse_args))
688     else:
689         parsed_results = map(parse_bl2seq_star, itertools.
izip(*parse_args))
690     parsed_results = filter(lambda x: x != None,
parsed_results)
691     rows, matrix = [], {}
692     for result in parsed_results:
693         rows.append(result[0])
694         matrix.update(dict([(key, [result[1][key]]) for
key in result[1]]))
695     make_data_table(outfile, 15, rows, COLUMNS, matrix)
696
697 if __name__ == '__main__':
698     if len(sys.argv) < 5:
699         print 'Incomplete function call --
parallel_blast2seq.py [# processors] [seq_file]
[table_file_ext] [output_file_ext] [Optional: --gap_open
value] [Optional: --gap_extend value] [Optional:
--comp_based True/False] [Optional: --use_best
True/False] [Optional: --filter -log10(E) filter]
[Optional: --blastp_exe blastp_exe]'
700     else:
701         p_num = int(sys.argv[1])
```


./parallel_resources/parallel_blast2seq.py

```
702         seq_loc = sys.argv[2]
703         tab_ext = sys.argv[3]
704         out_ext = sys.argv[4]
705         gap_open = '11'
706         gap_extend = '1'
707         no_comp_based = False
708         val_filter = None
709         use_best = False
710         blastp_exe = None
711         for i, arg in enumerate(sys.argv[5:]):
712             if arg == '--gap_open': gap_open = str(int(
sys.argv[i+6]))
713             elif arg == '--gap_extend': gap_extend = str(
int(sys.argv[i+6]))
714             elif arg == '--comp_based':
715                 if sys.argv[i+6] == 'False':
716                     no_comp_based = True
717             elif arg == '--filter': val_filter = float(
sys.argv[i+6])
718             elif arg == '--use_best': use_best = bool(sys
.argv[i+6])
719             elif arg == '--blastp_exe': blastp_exe = sys.
argv[i+6]
720             all_files = find_files(tab_ext, os.getcwd())
721             out_files = ['.'.join([infile.split('{0}'.format
(tab_ext))[0].split('/')[ -1], out_ext]) for infile in
all_files]
722             seqs = read_in_fa(seq_loc)
723             for infile, outfile in zip(all_files, out_files):
724                 run_blasts(infile, outfile, p_num, seqs, gap_open
, gap_extend, no_comp_based, val_filter, use_best, blastp_exe)
```

C.5. ./src packages

C.5.1. ./src/pythoscape/auxiliary modules

./src/pythoscape/auxiliary/blast_tools.py

```
1  # Author: Alan Barber
2  # Created on: March 11, 2012
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains general use functions
   that are used to run
7  and interpret BLAST. These functions have been separated
   out here because they
8  can be generally useful.
9
10 Functions:
11 get_sequence - Pulls a sequence from a BLAST database
12 '''
13
14 import subprocess
15 from parse_tools import get_name
16 from text_tags import TextTags
17 from file_tools import write_fasta_file
18
19 def get_sequence(database, title, database_re=None,
   blastdbcmd_exe=None):
20     ''' Returns sequence from BLAST formatted database.
21
22     Arguments:
23         database - the path to the BLAST database to
   search
24         title - the title of the sequence to retrieve
25
26     Arguments (with defaults):
27         database_re (default is None) - if database_re,
   the variable is used as a pattern to
28                                     search the title
   for the actual title in the BLAST
29                                     database
30         blastdbcmd_exe (default is 'blastdbcmd') - if
   blastdbcmd_exe, then that path is used
31                                     to
   run blastdbcmd
32
33     Returns:
34         sequence text
35     '''
36
```

./src/pythoscape/auxiliary/blast_tools.py

```
37     name = get_name(title,database_re) if database_re
else title
38     blastdbcmd = blastdbcmd_exe if blastdbcmd_exe else
'blastdbcmd'
39     cline = '{0} -db {1} -entry "{2}"'.format(blastdbcmd,
database,name)
40     child = subprocess.Popen(cline,shell=True,stdout=
subprocess.PIPE,close_fds=True)
41     my_out,my_err = child.communicate()
42     return ''.join(my_out.split('\n')[1:])
43
44 def parse_bl2seq(bl2seq_out):
45     ''' Parses the output of a bl2seq file into a
dictionary
46
47     Arguments:
48         bl2seq_out - an iterator that yields the lines
of a blast output
49
50     Returns:
51         dictionary of data from the bl2seq file
52     '''
53
54     tt = TextTags(bl2seq_out)
55     tt.set_position_tag('query','Query=',go_for=1)
56     tt.set_position_tag('subject','Subject=',go_for=1)
57     tt.set_position_tag('score line',' Score',
stop_tagging=' Identities')
58     tt.set_position_tag('identities line',' Identities',
go_for=1,stop_tagging='Query ')
59     tt.set_position_tag('query lines','Query ',go_for=1,
stop_tagging=' Score')
60     tt.set_position_tag('subject lines','Sbjct',go_for=1,
stop_tagging=' Score')
61     tt.set_position_tag('search space','Effective',go_for
=1)
62     tt.set_split_tag('query',[(('query',1)],splits=[' '])
63     tt.set_split_tag('subject',[(('subject',1)],splits=['
'])
64     tt.set_split_tag('score line',[(('bit_score',2)],
splits=[' '])
65     tt.set_split_tag('identities line',[(
'alignment_identities',2),('alignment_len',3)],splits=['
','/'])
66     tt.set_split_tag('query lines',[(('query_start',1),('
```

```

./src/pythoscape/auxiliary/blast_tools.py

'query_end',3]],splits=[' '],no_errors=True)
67     tt.set_split_tag('subject_lines',[(('subject_start',1
),('subject_end',3)],splits=[' '],no_errors=True)
68     tt.set_split_tag('search space',[(('m*n',4)],splits=[
' '])

69
70     all_data = {}
71     for parsed_line in tt:
72         if 'query_start' in parsed_line and 'query_start'
in all_data:
73             del parsed_line['query_start']
74             if 'subject_start' in parsed_line and
'subject_start' in all_data:
75                 del parsed_line['subject_start']
76                 all_data.update(parsed_line)
77     return all_data
78
79 def run_bl2seq(seqs,blast_exe,gap_open=None,gap_extend=
None,no_comp_based=False,use_best=False):
80     ''' Returns a dictionary of information about the
worst (or best score) from a pairwise
81     BLAST comparison
82
83     Arguments:
84         seqs - dictionary of the two sequences to compare
85         blast_exe - where the blastp executable is
86
87     Arguments (with defaults):
88         gap_open (default is 11) - gap open penalty
89         gap_extend (default is 1) - gap extend penalty
90         no_comp_based (default is False) - if True,
turns off composition based statistics
91         use_best (default is False) - if True, returns
info on best scoring comparison
92
93     Returns:
94         dictionary of data from the bl2seq file
95     '''
96
97     gap_open = gap_open if gap_open != None else '11'
98     gap_extend = gap_extend if gap_extend != None else '1'
99     comp_based = '0' if no_comp_based else '2'
100     seq_names = seqs.keys()
101     write_fasta_file('seq1.fa',{seq_names[0]:seqs[
seq_names[0]]})

```

./src/pythoscape/auxiliary/blast_tools.py

```
102     write_fasta_file('seq2.fa',{seq_names[1]:seqs[
seq_names[1]]})
103     cline = '{0} -query seq1.fa -subject seq2.fa
-gapopen {1} -gapextend {2} -comp_based_stats {3}
-use_sw_tback'.format(blast_exe,gap_open,gap_extend,
comp_based)
104     child = subprocess.Popen(cline,shell=True,stdout=
subprocess.PIPE,close_fds=True)
105     my_out,my_err = child.communicate()
106     option_1 = parse_bl2seq(iter(my_out.split('\n')))
107     cline = '{0} -query seq1.fa -subject seq2.fa
-gapopen {1} -gapextend {2} -comp_based_stats {3}
-use_sw_tback'.format(blast_exe,gap_open,gap_extend,
comp_based)
108     child = subprocess.Popen(cline,shell=True,stdout=
subprocess.PIPE,close_fds=True)
109     my_out,my_err = child.communicate()
110     option_2 = parse_bl2seq(iter(my_out.split('\n')))
111     if 'bit_score' in option_1 and 'bit_score' in
option_2 and option_2['bit_score']<option_1['bit_score']:
112         option_1,option_2 = option_2,option_1
113     elif not 'bit_score' in option_1 and 'bit_score' in
option_2:
114         return option_2
115     elif not 'bit_score' in option_2 and 'bit_score' in
option_1:
116         return option_1
117     elif not 'bit_score' in option_2 and not 'bit_score'
in option_1:
118         return None
119     if use_best:
120         return option_2
121     else:
122         return option_1
123
124 #Testing code and example usage
125 if __name__ == '__main__':
126     test_blast2seq = ['BLASTP 2.2.25+', '', '', 'Query=
28eeca152bd5c7a59dc72d6874a9296c0f5cdadcef6bc3653f335229',
127                       '', 'Length=218', '', 'Subject=
ca8bff01becf6df0c9626cf6636ecd55852a328c8955cbe3ce164744',
128                       '', 'Length=149', '', '',
129                       ' Score = 213 bits (543), Expect
= 1e-60, Method: Compositional matrix adjust.',
130                       ' Identities = 110/217 (51%),
```

./src/pythoscape/auxiliary/blast_tools.py

```
Positives = 129/217 (60%), Gaps = 69/217 (31%)', '',
131         'Query 1
MAMILGYWNVRLTHPIRLLLEYTDSNYEEKRYVMGDAPNFDRSQWLSEKFNGLGLDIP
NL 60',
132         '          M MILGYW++RGL H
IRLLLEYTDS+YEEK+Y MGDAP++DRSQWL+EKF LGLD PNL',
133         'Sbjct 1
MPMILGYWDIRGLAHAIRLLLEYTDSSYEEKKYTMGDAPDYDRSQWLNEKFKLGLDFP
NL 60',
134         '', 'Query 61
PYLIDGSHKVTQSNAILRYLGRKHNLCGETEEERIRVDTLENQVMDTRIHLMI VCCSP
DF 120',
135         '          PYLIDG+HK+TQSNAIL Y+
RKHNLCGETEEE+IRVD LENQ MD + L ++C +P+F',
136         'Sbjct 61
PYLIDGAHKITQSNAILCYIARKHNLCGETEEEEKIRVDILENQTMDSNHMQLGMICYNP
EF 120',
137         '', 'Query 121
EKQKPEFLKSIPEKMKIYSEFLGKRPWFAGDKVTYVDFLAYDILDQYRMFEPECLDAF
PN 180',
138
139         'Sbjct
-----
-- ',
140         '', 'Query 181
LKDFLARFEGLKKISAYMKSSSFLPRPVFTKIPQWGT 217',
141         '          GL+KISAYMKSS
FLPRPVF+K+ WG ',
142         'Sbjct 121
-----GLEKISAYMKSSRFLPRPVFSKMAVWGN 148', '', '', '',
143         'Lambda K H', '
0.323 0.140 0.438 ', '', 'Gapped',
144         'Lambda K H', ' 0.267
0.0410 0.140 ', '',
145         'Effective search space used:
25870', '', '', '', '', 'Matrix: BLOSUM62',
146         'Gap Penalties: Existence: 11,
Extension: 1', 'Neighboring words threshold: 11',
147         'Window for multiple hits: 40', '']
148 print parse_bl2seq(test_blast2seq)
149
```

./src/pythoscape/auxiliary/file_tools.py

```
1  # Author: Alan Barber
2  # Created on: March 11, 2012
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains general use functions
   that are used to manipulate files.
7  These functions have been separated out here because
   they can be generally useful.
8
9  Functions:
10 write_table_files - Writes a table file and a sequence
   file from a list of sequences and a list of edges
11 write_fasta_file - Writes a fasta file given a sequence
   dictionary.
12 find_files - Returns a list of all the files in a given
   directory with a given extension
13 pull_file_floats - Returns a list of all the lines of a
   file cast as floats
14 make_data_table - outputs data as a table file
15 '''
16
17 import gzip
18 import os
19
20 def write_table_files(directory,current_edges,i,compress=
   False):
21     ''' Creates a table file. These are specifically
   used to break up
22     numerous edge runs into smaller queues that can be
   run in parallel on
23     clusters.
24
25     It creates a table file using the current_edges
   variable. It assumes that
26     an edge is defined as a sequence of two objects. The
   format of the file
27     is one line per edge, sequence[0] sequence[1] on
   each line in that order.
28
29     Arguments:
30         directory - directory to output files
31         current_edges - sequence of edges, each of which
   is a sequence of two
```


./src/pythoscape/auxiliary/file_tools.py

```
32             text blocks denoting each name
33             i - name of table and sequence file, i.e.
[i].tab and [i].seq
34
35     Arguments (with defaults):
36     compress (default is False) - if true compresses
the files using gzip
37     '''
38
39     if not compress:
40         f_out = open('{0}/{1}.tab'.format(directory,i),
'w')
41     else:
42         f_out = gzip.open('{0}/{1}.tab.gz'.format(
directory,i), 'wb')
43     for edge in current_edges:
44         f_out.write('{0} {1}\n'.format(edge[0],edge[1]))
45     f_out.close()
46
47
48 def write_fasta_file(fa_file,seqs,file_exists=False):
49     ''' Creates a fasta file -- of the form >title
sequence
50
51     It creates the fasta file named using fa_file using
the dictionary-like
52     seqs, where the key is the title and the value is
the sequence.
53
54     Arguments:
55         fa_file - name of fasta file to output
56         seqs - dictionary of output sequences
57
58     Arguments (with defaults):
59         file_exists (default is False) - if true adds
the sequence to existing file
60     '''
61
62     option = 'a' if file_exists else 'w'
63     with open(fa_file,option) as f:
64         for seq in seqs:
65             f.write('>{0}\n{1}\n'.format(seq,seqs[seq]))
66
67 def find_files(ext, directory):
68     '''Returns the full path of all files in a directory
```

./src/pythoscape/auxiliary/file_tools.py

with a given extension.

```
69
70     Arguments:
71         ext - file extension of files to find
72         directory - directory to search
73
74     Returns:
75         list of full file paths
76     '''
77
78     return filter(lambda x: len(x) > len(ext)+1 and x[-
79 len(ext)-1:] == '{0}'.format(ext),
80                  ['{0}/{1}'.format(directory,item) for
81 item in os.listdir(directory)])
82
83 def pull_file_floats(file_name):
84     '''Returns a list of all the values in a list file
85     cast as floats.
86
87     Arguments:
88         file_name - location of file
89
90     Returns:
91         list of floats
92     '''
93     return_data = []
94     with open(file_name) as f:
95         for line in f:
96             return_data.append(float(line.strip()))
97     return return_data
98
99 def make_data_table(outfile,column_size,rows,columns,
100 matrix, csv=False):
101     '''Prints a data table file
102
103     Arguments:
104         outfile - name of file to output
105         column_size - column justification (except
106 for first)
107         rows - sequence of all row key names
108         columns - sequence of all column key names
109         matrix - dictionary of sequence values (e.g.
110 list) indexed by (row,column)
111
112     Arguments (with defaults):
```

./src/pythoscape/auxiliary/file_tools.py

```
107         csv (default is false) - if True, creates a csv
file instead of a space delimited table
108
109         Example output File:
110             Column 1    Column 2    Column 3
111         Row 1    Value 1,1    Value 1,2    Value 1,3a
112                                     Value 1,3b
113         Row 2                                     Value 2,2
114         (etc...)
115     '''
116     f = open(outfile, 'w')
117     first_width = max([len(str(row)) for row in rows])+2
118     if not csv:
119         first_row = '{0:>{width}}'.format(' ', width=
first_width)
120         first_row = ''.join([first_row]+['{0:>{width}}'.
format(column,width=column_size)
121                                     for column in
columns]+['\n'])
122     else:
123         first_row = ','.join([' ']+[str(column) for
column in columns]+['\n'])
124     f.write(first_row)
125     for row_id in rows:
126         start, i = True, 0
127         while 1:
128             to_continue = False
129             if start and not csv:
130                 row = '{0:>{width}}'.format(row_id,width=
first_width)
131             elif start and csv:
132                 row = '{0}'.format(row_id)
133             elif not start and not csv:
134                 row = '{0:>{width}}'.format(' ', width=
first_width)
135             else:
136                 row = ' '
137             column_data = []
138             for column in columns:
139                 if (row_id,column) in matrix:
140                     if len(matrix[row_id,column]) > i:
141                         column_data.append(matrix[row_id,
column][i])
142                     else:
143                         column_data.append(' ')
```

./src/pythoscape/auxiliary/file_tools.py

```
144             if len(matrix[row_id,column]) > i+1:
145                 to_continue = True
146             else:
147                 column_data.append(' ')
148             if not csv:
149                 row = ''.join([row]+'{0:>{width}}'.
format(column,width=column_size)
150                                     for column in
column_data)+'\n'])
151             else:
152                 row = ','.join([row]+[str(column) for
column in column_data])
153                 row = '{0}\n'.format(row)
154                 f.write(row)
155                 if not to_continue:
156                     break
157                 start = False
158                 i += 1
159         f.close()
```

./src/pythoscape/auxiliary/main_tools.py

```
1  # Author: Alan Barber
2  # Created on: March 11, 2012
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains general use functions
   that are used by the main
7  Pythoscape classes. These functions have been separated
   out here because they
8  can be generally useful.
9
10 Functions:
11 nonstring_iterable - returns True/False if an object is
   a nonstring iterable object
12 get_one_attr - returns an attribute of an object if the
   attribute exists or None if it doesn't exist
13 and_lambda - does AND on two variables
14 or_lambda - does OR on two variables
15 make_comp_lambda - Returns a lambda function that can
   compare two objects
16 return_internal_attr - Calls an internal object method
   based on a path
17 '''
18
19 def nonstring_iterable(test_obj):
20     '''Returns True/False if an object is a nonstring
   iterable object.
21
22     Arguments:
23         test_obj - object to test
24     '''
25     if isinstance(test_obj, basestring):
26         return False
27     try:
28         iter(test_obj)
29     except TypeError:
30         return False
31     else:
32         return True
33
34 def get_one_attr(func, attr):
35     '''Returns an attribute of an object if the
   attribute exists or None if it doesn't exist.
36
```

./src/pythoscape/auxiliary/main_tools.py

```
37     Arguments:
38         func - object to check for attribute
39         attr - attribute to look for
40
41     returns:
42         the attribute if it exists
43         None if it doesn't
44     '''
45
46     return getattr(func,attr) if hasattr(func,attr) else
None
47
48 def and_lambda(*xs):
49     '''Performs an and on all inputed arguments in order
50
51     Arguments:
52         *xs - arguments
53
54     returns:
55         and of all arguments
56     '''
57     return reduce(lambda x,y: x and y,xs)
58
59
60 def or_lambda(*xs):
61     '''Performs an or on all inputed arguments in order
62
63     Arguments:
64         *xs - arguments
65
66     returns:
67         or of all arguments
68     '''
69     return reduce(lambda x,y: x or y,xs)
70
71 def make_comp_lambda(condition,value):
72     '''Returns a lambda function that will compare a
73     variable to the value provided
74     using the comparison function provided.
75
76     Arguments:
77         condition - Appropriate values are
78         '>','<','!=','<>','==','>=' or '<='.
79         If another value is provided,
80         None will be returned.
```

./src/pythoscape/auxiliary/main_tools.py

```
78         value - value to compare to
79
80     Returns:
81         corresponding lambda function
82     '''
83     if condition == '>':
84         return lambda x: x > value
85     elif condition == '<':
86         return lambda x: x < value
87     elif condition == '>=':
88         return lambda x: x >= value
89     elif condition == '<=':
90         return lambda x: x <= value
91     elif condition == '==':
92         return lambda x: x == value
93     elif condition == '!=' or condition == '<>':
94         return lambda x: x != value
95     else:
96         return None
97
98 def return_internal_attr(obj,*args):
99     '''Calls an internal object method based on a path
100
101     Arguments:
102         obj - object with method to call
103         *args - path to method
104
105     Returns:
106         called method if it exists or None if not
107
108     Example usage:
109
110     return_internal_attr(obj_to_search,attribute,sub_attribute
111     ,method)
112
113     '''
114     answer = reduce(get_one_attr,[obj]+list(args))
115     return answer() if answer else None
```

./src/pythsocape/auxiliary/parse_tools.py

```
1  # Author: Alan Barber
2  # Created on: March 11, 2012
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains general use functions
   that are used for text parsing.
7  These functions have been separated out here because
   they can be generally useful.
8
9  Functions:
10 give_lines - Takes a text stream and returns a generator
   that yields individual lines
11 get_name - Returns an RE pattern match in text if there
   is only one match
12 '''
13
14 import re
15
16 def give_lines(out_stream):
17     ''' Generator that yields lines delimited by slash-n
   from a text stream.
18
19     Arguments:
20         out_stream - text stream
21
22     Yields:
23         delimited text blocks'''
24
25     growing = ''
26     for char in out_stream:
27         if char == "\n":
28             yield growing
29             growing = ''
30         else:
31             growing = '{0}{1}'.format(growing, char)
32
33 def get_name(title, name_re, take_first=False):
34     ''' Returns match from regular expression search if
   and only if there is one match.
35     If not, the function throws an IDError.
36
37     Arguments:
38         title - the text to search
```


./src/pythsocape/auxiliary/parse_tools.py

```
39         name_re - the regular expression pattern
40
41     Arguments (with defaults):
42         take_first (default is False) - if True,
43         overrides error checking for multiple hits and gives
44         first hit
45
46     Returns:
47         match text
48     '''
49     matches = re.findall(re.compile(name_re),title)
50     if take_first and len(matches) > 0:
51         pass
52     elif len(matches) != 1:
53         try:
54             raise ValueError;
55         except ValueError as err:
56             if not err.args:
57                 err.args=(' ',)
58             err.args = (err.args[0] + ""
59 The regular expression failed. It needs to find exactly
60 one match and it found
61 {0} matches. The search string was {1} against the RE
62 {2} and the matches were {3}
63 """".format(len(matches),title,name_re,matches),)+err.args
64 [1:]
65
66         raise
67     return matches[0]
68
69 #Testing code and example usage
70 if __name__ == '__main__':
71     #Testing get_name on a title that works.
72     print get_name('D8TWR1_VOLCA/23-311',
73 '^'[A-Z0-9]{6}[_][A-Z]{5}(?=[/]))
74
75     #Testing get_name on a title that doesn't work.
76     print get_name('8TWR1_VOLCA/23-311',
77 '^'[A-Z0-9]{6}[_][A-Z]{5}(?=[/]))
```

./src/pythoscape/auxiliary/re_patterns.py

```
1  # Author: Alan Barber
2  # Created on: September 9, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains a single dictionary
   that defines various RE patterns.
7  The dictionary is called RE_PATTERNS and has key, value
   pairs corresponding to a type
8  of RE pattern and the pattern definition.
9
10 These patterns are used specifically in locating the
   name of a node given a specific
11 string that contains the node name. Several plug-ins use
   these patterns as inputs
12 to properly locate and pass data. Instead of having the
   user determine in each
13 case the appropriate RE string (given that they can be
   complex to determine),
14 useful patterns will be pre-defined in this file.
15
16 It is intended that this file will grow as further
   patterns are needed.
17
18 Current patterns:
19
20 Pattern ID | Use case
21 -----
22 'SFLD_GI' | This pattern locates GIs
   from files from the SFLD. It should also work with
   general GIs, but this is untested.
23 'SFLD_EFD' | This pattern locates EFD
   numbers outputted from the SFLD.
24 'SFLD UniProtEntry' | This pattern locates
   Uniprot entry names in an SFLD-like title.
25 'SFLD_Enzyme' | This pattern is used for
   GI like titles using the prefix 'lcl|' and suffix '|'
26 'Uniref100 with prefix' | This pattern pulls out the
   whole UnirefID including the 'UniRef100_' tag at the
   beginning.
27 'Uniref100 without prefix' | This pattern pulls out the
   UniRefID but leaves off the 'UniRef100_' tag.
28 'UniprotKB' | This pattern pulls out a
   UniprotKB AC - the 6 number letter one.
```

./src/pythoscape/auxiliary/re_patterns.py

```
29 'Uniprot/Pfam full length' | This pattern pulls out a
   UniprotKB AC from a full length sequence download from
   Pfam.
30 'Uniprot/Pfam alignment' | This pattern pulls out a
   UniprotKB ID from a Pfam alignment sequence header.
31 'PDB/NCBI' | This pattern pulls out a
   PDB ID from an nr sequence title.
32 'Uniprot ID-like' | This pattern pulls out a
   UniProt ID if it comes first.
33 '''
34
35 RE_PATTERNS = {'SFLD_GI':
   '(<=[Gg][Ii][|])\d{1,12}(?=[|])|(<=[Gg][Ii][|])\d{1,12}'
   ,
36               'SFLD_EFD':
   '(<=[EFDID[|])\d{1,12}(?=[|])|(<=[EFDID[|])\d{1,12}',
37               'SFLD_UniProtEntry':
   '(<=[Ss][Pp][|])[A-Za-z0-9]{3,6}[_][A-Za-z0-9]{3,5}',
38               'SFLD_Enzyme':
   '(<=[l][c][l][|])\d{1,12}(?=[|])',
39               'Uniref100 with prefix':
   'UniRef100[_]UPI\w{10}|UniRef100[_]\w{6}[-]\d{1,2}|UniRef1
   00[_]\w{6}',
40               'Uniref100 without prefix':
   '(<=[UniRef100[_])UPI\w{10}|(<=[UniRef100[_])\w{6}[-]\d{1,
   2}|(<=[UniRef100[_])\w{6}',
41               'UniprotKB':
   '(<=[sp[|])\w{6}|(<=[tr[|])\w{6}',
42               'Uniprot/Pfam full length':
   '(<=[(|])\w{6}(?=[|])',
43               'Uniprot/Pfam alignment':
   '^[A-Z0-9]{2,6}[_][A-Z0-9]{3,5}(?=[/])',
44               'Uniprot ID-like':
   '^[a-zA-Z0-9]{2,6}[_][a-zA-Z0-9]{3,5}',
45               'PDB/NCBI':
   '(<=[Pp][Dd][Bb][|])[A-Z0-9]{4}(?=[|])'}
```

./src/pythoscape/auxiliary/text_tags.py

```
1  # Author: Alan Barber
2  # Created on: March 11, 2012
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains the class TextTags and
   its related functions.
7  It is a class used to tag text streams and then parse
   them based on the tags
8  given.
9
10 For example usage, see the __name__ = '__main__' section
   of the module.
11
12 Classes:
13 AmbiguousZ - Error when it isn't clear which line tag
   should be used.
14 TextTags - Class to use tags to pull out data from a
   text block.
15 '''
16
17 from itertools import chain
18
19 class AmbiguousZ(Exception):
20     '''Exception thrown when there isn't a clear tag for
   a line.
21
22     Arguments:
23     line - the data line where there isn't a clear
   tag'''
24
25     def __init__(self, line, tag_set):
26         self.line = line
27         self.tag_set = tag_set
28
29     def __str__(self):
30         return """The line '{0}' in the data doesn't
   have a clear tag.
31             Use Z to set one. Current
   tags:\n{1}\n""".format(self.line, self.tag_set)
32
33 class TextTags(object):
34     '''TextTags takes a text string and tags lines based
   on the parameters
```

./src/pythoscape/auxiliary/text_tags.py

```
35     used it set_position_tags. Line splitting tags can
    be set using
36     set_split_tags. Iterating through the class gives
    dictionaries
37     from each line of the defined data.
38
39     Public methods:
40     set_position_tag - sets a line tag that indicates
    data positions
41     set_split_tag - sets a data tag to a line that
    indicates how to find it
42     reset - resets internal iterator to first line
43
44     Example Usage:
45     my_tt = TextTags(['* This sentence starts with stars
    and has a 1 = 1 equation.'])
46     my_tt.set_position_tag('double star','*',go_for=1)
47     my_tt.set_split_tag('double
    star',[(('equals',10)],splits=[' '])
48     for item in my_tt:
49         print item
50     '''
51
52     #Parameters to be saved from set_position_tag
53     POSITION_PARAMS = ['name','start_tag','lines_to_data',
54                       'stop_tagging','multiple_tags',
    'empty_tag',
55                       'go_for','z']
56
57     #Parser states set in _match_to_tag
58     SAVED_STATES = ['start','started','keep_going',
    'repeater','saved_lines','stop']
59
60     def __init__(self,data,split_on=None):
61         '''Create a new instance of TextBlock.
62
63         Arguments:
64             data -- string or list of strings
65
66         Arguments (with defaults):
67             split_on (default is None): If a single
    string is inputed, this is how to split it
68                                     to prepare the
    list of strings. If None, no split is
69                                     performed
```

./src/pythoscape/auxiliary/text_tags.py

```
70         '''
71
72         if split_on: data = data.split(split_on)
73         self.data = list(data)
74         self._line_tags = {}
75         self._split_tags = {}
76         self._pulled_tags = {}
77         self._data_tags = [None]*len(self.data)
78         self._lp = {}
79         self._lpp = dict([(pos_param,i) for i,pos_param
in enumerate(self.POSITION_PARAMS)])
80         self._ls = {}
81         self._lsp = dict([(state,i) for i,state in
enumerate(self.SAVED_STATES)])
82         self._i = 0
83         self._zs = {}
84         self._current_tag = None
85         self._errors = set([])
86
87     def __str__(self):
88         '''Overwrites __str_ when printing TextTags
89
90         Returns:
91             a description of how each line will be parsed
92         '''
93
94         out = []
95         for return_data in self:
96             if return_data:
97                 out_data = '\n'.join(['{0}:{1}'.format(dp
,return_data[dp]) for dp in return_data]+[''])
98             else:
99                 out_data = ''
100             out = out + ['Line:\n{0}\n'.format(self.data[
self._i]),
101                             'Assigned tag:\n{0}\n'.format(
self._current_tag),
102                             'Parsed data:\n',out_data,'\n']
103             self.reset()
104         return ''.join(out)
105
106     def __iter__(self):
107         '''Overwrites __iter__ to yield all data
108         dictionaries parsed from text.
```

./src/pythoscape/auxiliary/text_tags.py

```
109         Yields:
110             dictionary of {data_name:data} of all parsed
hits in each line
111         '''
112
113         for line in self.data:
114             self._get_tag()
115             if self._current_tag:
116                 yield self._make_data_dict()
117             else:
118                 yield {}
119             self._i += 1
120
121         def _make_data_dict(self):
122             '''Private function that makes a the output
dictionary from each line.
123
124             Do not call as this function is subject to
removal or change!'''
125
126             outline = [self.data[self._i]]
127             if self._current_tag in self._split_tags:
128                 for split_on in self._split_tags[self.
_current_tag]:
129                     outline = [item.strip().split(split_on)
for item in outline]
130                     outline = [item for item in chain(*
outline) if item != '']
131                 try:
132                     return dict([(name, outline[location]) for
name, location
133                                     in self._pulled_tags[self.
_current_tag]])
134                 except IndexError:
135                     if self._current_tag in self._errors:
136                         return {}
137                     else:
138                         try:
139                             raise IndexError;
140                         except IndexError as err:
141                             if not err.args:
142                                 err.args=(' ',)
143                             err.args = (err.args[0] + ""
The split definition given: {0}
144 cannot be performed on line: {1}
```

./src/pythoscape/auxiliary/text_tags.py

```
146     """.format(self._pulled_tags[self._current_tag],outline
    ),)+err.args[1:]
147         raise
148     elif self._current_tag in self._pulled_tags:
149         line = outline[0].strip()
150         return {self._pulled_tags[self._current_tag][
0][0]:line}
151     else:
152         return {}
153
154     def _apply_z(self,tag_set):
155         '''Private function that figures out tag with
highest z.
156
157         Do not call as this function is subject to
removal or change!'''
158
159         tag_set = filter(lambda x: x != None, tag_set)
160         if len(tag_set) == 1:
161             return tag_set[0]
162         elif len(tag_set) == 0:
163             return None
164         z_set = [self._zs[tag] for tag in tag_set]
165         max_tag = max(z_set)
166         if z_set.count(max_tag) > 1:
167             raise AmbiguousZ(self.data[self._i],tag_set)
168         return tag_set[z_set.index(max_tag)]
169
170     def _get_tag(self):
171         '''Private function that sets the current
position tag.
172
173         Do not call as this function is subject to
removal or change!'''
174
175         if self._i == 0:
176             for tn in self._ls:
177                 self._ls[tn][self._lsp['start']] = False
178                 self._ls[tn][self._lsp['saved_lines']] =
self._lp[tn][self._lpp['lines_to_data']]*1
179                 if self._lp[tn][self._lpp['multiple_tags'
]]:
180                     self._ls[tn][self._lsp['repeater']] =
len(self._lp[tn][self._lpp['name']])
181                     for tp in self._lp:
```


./src/pythoscape/auxiliary/text_tags.py

```
182         if self._lp[tp][self._lpp['multiple_tags']
183         ]]:
184             for one_tag in tp:
185                 self._zs[one_tag] = self._lp[tp][
186                 self._lpp['z']]
187             else:
188                 self._zs[tp] = self._lp[tp][self._lpp
189                 ['z']]
190         if self._i < len(self.data):
191             return_data = map(self._match_to_tag, [
192             tag_name for tag_name in self._lp])
193             self._current_tag = self._apply_z(return_data)
194
195         def _get_run_info(self, tag):
196             '''Private function that retrieves the position
197             tagging state.
198
199             Do not call as this function is subject to
200             removal or change!'''
201
202             return_data = dict([(name, self._ls[tag][self._lsp
203             [name]]) for name in self._lsp])
204             return_data.update(dict([(name, self._lp[tag][self
205             ._lpp[name]]) for name in self._lpp]))
206             return return_data
207
208         def _update_run_info(self, tag, tag_info):
209             '''Private function that updates the current
210             position tagging state.
211
212             Do not call as this function is subject to
213             removal or change!'''
214
215             for i, state in enumerate(self.SAVED_STATES):
216                 self._ls[tag][i] = tag_info[state]
217
218         def _match_to_tag(self, tag):
219             '''Private function that matches a line to a
220             given position tag.
221
222             Do not call as this function is subject to
223             removal or change!'''
224
225             return_tag = None
226             run_ps = self._get_run_info(tag)
```

./src/pythoscape/auxiliary/text_tags.py

```
215         line = self.data[self._i]
216         if line.startswith(run_ps['start_tag']):
217             run_ps['start'], run_ps['started'] = True, True
218             run_ps['keep_going'] = run_ps['go_for']
219             if run_ps['saved_lines'] != ['lines_to_data']:
220                 run_ps['saved_lines'] = run_ps[
221 'lines_to_data']*1
222         if not run_ps['saved_lines'] and run_ps[
223 'stop_tagging'] != None\
224         and line.startswith(run_ps['stop_tagging'])
225 and run_ps['started']:
226             run_ps['stop'] = True
227             if run_ps['start'] and not run_ps['stop']:
228                 if not run_ps['saved_lines']:
229                     do_return = True
230                     if not line.strip():
231                         do_return = False
232                     if run_ps['empty_tag']:
233                         do_return = True
234                     if run_ps['keep_going'] == 1:
235                         run_ps['start'] = False
236                     elif run_ps['keep_going']:
237                         run_ps['keep_going'] -= 1
238                     if not run_ps['multiple_tags'] and
239 do_return:
240                 return_tag = run_ps['name']
241                 elif run_ps['multiple_tags'] and run_ps[
242 'name'][(len(run_ps['name'])-run_ps['repeater'])] == None:
243                     run_ps['repeater'] -= 1
244                 elif do_return:
245                     return_tag = run_ps['name'][(len(
246 run_ps['name'])-run_ps['repeater'])]
247                     run_ps['repeater'] -= 1
248                 else:
249                     run_ps['saved_lines'] -= 1
250                 if run_ps['repeater'] == 0:
251                     run_ps['repeater'] = len(run_ps['name'])
252                 self._update_run_info(tag, run_ps)
253                 if return_tag != None:
254                     return return_tag
255
256     def set_position_tag(self, name, start_tag,
257 lines_to_data=0,
258 stop_tagging=None, multiple_tags=
259 False, empty_tag=False,
```

./src/pythoscape/auxiliary/text_tags.py

```
252         go_for=None, z=0):
253     '''Sets a position tag, which tags a given line
254     as a certain type of line
255     given the parameters.
256     Arguments:
257     name - tag name or list of names (if using a
258     list of names, multiple_tags
259         must be set to True)
260     start_tag - where to start tagging lines with
261     the tag (if this is set,
262         but no other optional arguments are
263     set, then all lines
264         after the first start hit will be
265     tagged)
266     Arguments (with defaults):
267     lines_to_data (default 0) - how many lines after
268     the line with the start_tag
269         to start tagging
270     stop_tagging (default None) - line start that
271     indicates to stop tagging
272     multiple_tags (default False) - name is not one
273     tag, but a list of tags
274         that will be
275     used in order, None indicates
276         a line to skip
277     tagging
278     empty_tag (default False) - empty lines won't be
279     tagged if False
280     go_for (default None) - if set to a number, this
281     will limit the number of lines
282         tagged to the number given
283     z (default 0) - z of the tag. Only one position
284     tag can be given per line,
285         the tag with the highest z will
286     be the set tag.
287     '''
288     if multiple_tags:
289         name = tuple(name)
290         self._lp[name] = [None]*len(self.POSITION_PARAMS)
291         for i, parameter in enumerate(self.POSITION_PARAMS
292 ):
293             if parameter in locals():
294                 self._lp[name][i] = locals()[parameter]
```

./src/pythoscape/auxiliary/text_tags.py

```
282         self._ls[name] = [None]*len(self.SAVED_STATES)
283
284     def reset(self):
285         '''Resets the iterator to the start.
286
287         Arguments:
288             None'''
289         self._i = 0
290         for name in self._lp:
291             self._ls[name] = [None]*len(self.SAVED_STATES)
292
293     def set_split_tag(self,name,pulled_elements,splits=
None,no_errors=False):
294         '''Sets a split tag, which indicates how to find
the data point
295         that is being searched for.
296
297         Arguments:
298             name - position tag to use for split tag
299             pulled_elements - list of a set of tuples of the
following form,
300
301             [(split_tag_name,split_tag_location)]
302
303             Arguments (with defaults):
304             splits (default None) - set of splits to use on
the line, used in order
305
306             to the flattened set of
all data
307
308             If this is None and if
you want the whole line
309
310             pulled_elements should
look like:
311
312             [(split_tag_name,0)]
313
314             no_errors (default False) - if True, instead of
throwing an error when
315
316             the line can't be
parsed, it returns an empty
317
318             dictionary
'''
319
320     if splits != None:
321         self._split_tags[name] = splits
322     if no_errors:
323         self._errors.add(name)
```

./src/pythoscape/auxiliary/text_tags.py

```
317         self._pulled_tags[name] = pulled_elements
318
319
320     #
321     if __name__ == '__main__':
322
323         test_text = """This is complex text.
324         ** This sentence starts with stars and has a 1 = 1
325         equation.
326         + Sometimes they start with + and end with -
327         Can you parse out the .
328         Can you parse out the ?
329         How about the &&!&&"""
330         tt_test = TextTags(test_text, split_on='\n')
331         tt_test.set_position_tag('starting', 'This', go_for=1)
332         tt_test.set_position_tag('double star', '**', go_for=1)
333         tt_test.set_position_tag('plus', '+', go_for=1)
334         tt_test.set_position_tag(['period', 'question mark',
335         'ampersand'], ' ', multiple_tags=True)
336         tt_test.set_split_tag('double star', [('equals', 10)],
337         splits=[' '])
338         tt_test.set_split_tag('plus', [('plus', 5), ('minus', 9
339         )], splits=[' '])
340         tt_test.set_split_tag('period', [('period', 5)], splits
341         =[' '])
342         tt_test.set_split_tag('question mark', [('question
343         mark', 5)], splits=[' '])
344         tt_test.set_split_tag('ampersand', [('exclamation
345         mark', 3)], splits=[' ', '&'])
346         print tt_test
```

./src/pythoscape/auxiliary/xml_tools.py

```
1  # Author: Alan Barber
2  # Created on: March 17, 2012
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains general use functions
   that are used for xml processing.
7  These functions have been separated out here because
   they can be generally useful.
8
9  Functions:
10  get_xml_type - finds the appropriate xgmml type for an
   object, uses XGMML_RULES
11  output_elemtree - uses the elemtree package to output a
   XGMML formatted network file
12  '''
13
14  from xml.etree import ElementTree
15
16  #Used by get_xgmml_type to figure out a mapping between
   Python objects and xgmml objects
17  XGMML_RULES = [(bool, "boolean"),
18                  (str, "string"),
19                  (int, "integer"),
20                  (float, "real"),
21                  (long, "integer")]
22
23  def get_xgmml_type(thing):
24      '''Finds the appropriate xgmml type for an object,
   uses XGMML_RULES
25
26      Arguments:
27          object - find the appropriate XGMML type for
   this object
28
29      Returns:
30          string of XGMML type
31      '''
32
33      for (rule, returnType) in XGMML_RULES:
34          if isinstance(thing, rule): return returnType
35
36  def output_elemtree(nodes, edges, output, label):
37      '''Uses the elemtree package to output a XGMML
```

./src/pythoscape/auxiliary/xml_tools.py

formatted network file

```
38
39     Arguments:
40         nodes - dictionary-like container of nodes
41                where key:value =
42         node_name: [(attribute1_name, attribute1_value), ...]
43         edges - dictionary-like container of edges
44                where key:value =
45         (node_name_1, node_name_2): [(attribute1_name, attribute1_val
46         ue), ...]
47
48         output - output file name
49         label - what to label the network in the XGMML
50
51     Returns:
52         Nothing
53     '''
54     with open(output, 'w') as f:
55         f.write("<?xml version='1.0' ?>\n<graph
56         id='outputFile' label='{0}'
57         xmlns='http://www.cs.rpi.edu/XGMML'>\n".format(label))
58         for node in nodes:
59             new_node = ElementTree.Element('node')
60             new_node.attrib['id'] = str(node)
61             new_node.attrib['label'] = str(node)
62             for attribute in nodes[node]:
63                 new_attribute = ElementTree.Element('att')
64                 new_attribute.attrib['value'] = str(
65                 attribute[1])
66                 new_attribute.attrib['name'] = str(
67                 attribute[0])
68                 new_attribute.attrib['type'] =
69                 get_xgmml_type(attribute[1])
70                 new_node.append(new_attribute)
71             one_node = ElementTree.ElementTree(element=
72             new_node)
73             one_node.write(f)
74             f.write('\n')
75             for edge in edges:
76                 new_edge = ElementTree.Element('edge')
77                 new_edge.attrib['id'] = ''.join([str(edge[0
78                 ]), ',', str(edge[1])])
79                 new_edge.attrib['label'] = ''.join([str(edge[
80                 0]), ',', str(edge[1])])
81                 new_edge.attrib['source'] = str(edge[0])
82                 new_edge.attrib['target'] = str(edge[1])
```

./src/pythoscape/auxiliary/xml_tools.py

```
71         for attribute in edges[edge]:
72             new_attribute = ElementTree.Element('att')
73             new_attribute.attrib['value'] = str(
attribute[1])
74             new_attribute.attrib['name'] = str(
attribute[0])
75             new_attribute.attrib['type'] =
get_xgmmL_type(attribute[1])
76             new_edge.append(new_attribute)
77             one_edge = ElementTree.ElementTree(element=
new_edge)
78             one_edge.write(f)
79             f.write('\n')
80             f.write('</graph>\n')
```


C.5.2. ./src/pythoscape/interface modules

./src/pythoscape/interface/local_interface.py

```
1  # Author: Alan Barber
2  # Created on: July 14, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: The module contains the local interface. It
   mimics the behavior
7  of the MySQL interface using flat files containing
   Python shelves. It does not
8  require the more extensive setup needed to create and
   manage MySQL databases.
9  This interface is ideal for initial testing of network
   interfaces and working
10 with smaller databases. Because it uses flat files and
   loads portions of the entire
11 network into memory at a single time, this interface is
   not appropriate for large
12 networks that require database software to index and
   manipulate them.
13
14 Specifically, small networks could be thought of
   networks with no more than 999 nodes
15 where the complete network is no larger than ~500,000
   edges. Even with these restrictions,
16 if there are many (>10) edge attributes and if the
   computer has limited RAM, there
17 could be memory issues.
18
19 When in doubt, use the MySQL interface!
20
21 The LocalInterface class was written as a general
   modifiable Pythoscape storage solution.
22 For that reason, several dictionaries contain the
   definitions for the specifics of the graph
23 structure. If the graph structure is modified for any
   reason, these values should be changed
24 instead of a wholesale rewrite of the interface.
25
26 This interface is at the moment too complicated and
   bloated. It was written as a quick and
27 dirty band-aid to solve the problem of non-MySQL
   storage. Eventually, there will be two new
28 solutions: 1) an object-oriented LocalInterface that
   does away with the bloat and
```

./src/pythoscape/interface/local_interface.py

```
29 2) An interface built on NoSQL database solutions.
30
31 Functions:
32 match -- matches unwrapped objects against a query
33         objects. If there is a match, then true
34         is returned. If not, false is returned.
35
36 Classes:
37 NoAttributeError -- Exception for when a required
38 attribute is not present in the network
39 LocalInterface -- Interface class (see description above)
40
41 Example usage:
42 my_loc_inter = LocalInterface(<directory>)
43
44 For general usage of any interface including a
45 description of how to push, pull and delete
46 network objects, see the graph_interface documentation.
47 '''
48
49 import shelve
50 from itertools import product, chain
51
52 from standard_interface import StandardInterface
53
54 # Constants -- These are for use with private functions.
55 # It is intended for these to be changed in accordance with
56 # any changes in the graph structure.
57
58 #INFORMATION -- Information stored in each storage
59 shelf. Keys are names of different
60 #               network objects and values are tuples
61 #               corresponding to the data stored
62 #               in each shelf if the object is stored.
63
64 INFORMATION = {'Node_attribute':('id','attribute_name',
65                                 'attribute_value'),
66               'Network_attribute':('network_name',
67                                    'attribute_name','attribute_value'),
68               'Edge_attribute':('source_node',
69                                 'target_node','attribute_name','attribute_value'),
70               'Node':('id','identifiers','networks'),
71               'Edge':('source_node','target_node',
72                       'weight','filter','networks'),
73               'Network':('network_name','network_type'),
```

./src/pythoscape/interface/local_interface.py

```
64         'Attribute':('attribute_name',
        'element_obj','attribute_value'),
65         'Identifier':('identifier_type',
        'identifier_value'),
66         'Filter':('filter_name','direction',
        'filter_value'),
67         'Weight':('weight_value','algorithm')}}
68
69 #COMPLEX -- Defines attributes from above that are
        complex attributes,
70 #         e.g. attributes where a single object can
        have multiple.
71
72 COMPLEX = {'identifiers':('identifier_type','Identifier',
        '_identifier'),
73            'networks':('network_name','Network',
        '_network')}}
74
75 #KEYED_BY -- Indicates the network object values that
        are used to define a unique object. Keys
76 #         are shelves corresponding to a network
        object and the value is a list
77 #         of network object values used to key the
        storage of the object.
78
79 KEYED_BY = {'Node':['id'],
80             'Edge':['source_node','target_node'],
81             'Network':['network_name'],
82             'Node_attribute':['id','attribute_name'],
83             'Network_attribute':['network_name',
        'attribute_name'],
84             'Edge_attribute':['source_node','target_node',
        'attribute_name']}]
85
86 #INDEXES -- Shelves used to store indexes to find
        appropriate network objects. Keys
87 #         are shelf names and values are a list of
        attributes that have to be
88 #         indexed.
89
90 INDEXES = {'Node':['id','networks','identifiers'],
91            'Edge':['source_node','networks'],
92            'Network':['network_name','network_type'],
93            'Node_attribute':['id','attribute_name'],
94            'Network_attribute':['network_name',
```

```

./src/pythoscape/interface/local_interface.py

'attribute_name'],
95     'Edge_attribute':['source_node',
'attribute_name']]
96
97 #COMPLEX_INDEXES -- Values to use to find complex indexes.
98
99 COMPLEX_INDEXES = {'network_name':'networks',
100                   'identifier_type':'identifiers'}
101
102 #FIND_COMPLEX_INDEXES -- Attributes that don't store
complex indexes, but
103 #                      need to have them searched
anyway.
104
105 FIND_COMPLEX_INDEXES = {'Node_attribute':'Node',
106                        'Edge_attribute':'Edge'}
107
108 #DOUBLE_INDEX -- Network object values that must be
indexed under a different name to
109 #              function properly, like target_node
ids. Key is value to be indexed as
110 #              another, value is what the value is to
be indexed as.
111
112 DOUBLE_INDEX = {'target_node':'source_node'}
113
114 #PULL_T00 -- What other objects pull be looked up and
pulled with a certain object is pulled.
115 #          Key is object being pulled, value is object
to look up and also pull.
116
117 PULL_T00 = {'Node_attribute':'Node',
118            'Network_attribute':'Network',
119            'Edge_attribute':'Edge'}
120
121 #CHECK -- These need to be checked to make sure that
they already exist in the database if
122 #          they are to be set as part of an object.
123
124 CHECK = {'Node':['network_name'],
125         'Edge':['source_node','target_node'],
126         'Node_attribute':['id'],
127         'Network_attribute':['network_name']}
128
129 #DOUBLE_CHECK -- Just like check, but they have to be

```

./src/pythoscape/interface/local_interface.py

```
checked as a group.
130 #
131
132 DOUBLE_CHECK = {'Edge_attribute':[('source_node',
133                                     'target_node')]]}
134
135 #OBJ_TRANS -- In the case that one of the functions
136 #             needs to find the corresponding object
137 #             for an object attribute, it looks here.
138 #             For example, the attributes mentioned
139 #             in CHECK are searched here.
140
141 OBJ_TRANS = {'network_name':'Network',
142              'id':'Node',
143              ('source_node','target_node'):'Edge',
144              'source_node':'Node',
145              'target_node':'Node'}
146
147 #NOT_STANDARD -- These are objects passed by the
148 #               standard interface but need to be stored as
149 #               part of the object given, not a
150 #               seperate object.
151
152 NOT_STANDARD = {'Network_node':'Node',
153                'Network_edge':'Edge',
154                'Identifier':'Node',
155                'Network_node_attribute':'Node_attribute',
156                'Network_edge_attribute':'Edge_attribute'}
157
158 #FIND_NS -- This is the object that needs to be looked
159 #           for to properly set the non-standard
160 #           object.
161
162 FIND_NS = {'Network_node':'Network',
163            'Network_edge':'Network',
164            'Identifier':'Identifier'}
165
166 #LINKED -- Objects that need to be checked to be to see
167 #          if they need to be deleted when the
168 #          key object is deleted.
169
170 LINKED = {'Node':['Edge','Node_attribute'],
171           'Edge':['Edge_attribute'],
172           'Network':['Network_node','Network_edge',
173                     'Network_attribute']}
```

./src/pythoscape/interface/local_interface.py

```
166
167 #LINKED_MODIFY -- Synonyms for the LINKED check.
168
169 LINKED_MODIFY = {'id': 'source_node'}
170
171 def make_index(sequence, boundry):
172     ''' Takes a sequence and converts it to a single
173     string with a boundary joining it.
174     Standardizes making indexes with a single
175
176     Arguments:
177         sequence - sequence of items
178         boundry - how to join the items
179
180     Returns:
181         string of joined items
182     '''
183     return boundry.join([str(item) for item in sequence])
184
185 class NoAttributeError(Exception):
186     ''' Exception to be called when a required attribute
187     is not present in the
188     stored network.
189
190     Arguments:
191         obj -- Name of object that is throwing error
192         att_name -- Name of attribute missing
193         required -- Required value that is missing'''
194
195     def __init__(self, obj, att_name, required):
196         self.obj = obj
197         self.att_name = att_name
198         self.required = required
199
200     def __str__(self):
201         return '''A/an {0} is calling for a {1}, {2},
202         that doesn't exist in the database.
203         This is a fatal error.''.format(self.
204         obj, self.att_name, self.required)
205
206 class LocalInterface(StandardInterface):
207     '''Class that describes the local interface. For an
208     extended explanation,
209     see the documentation for the module. Public methods
210     are as required
```

```

./src/pythoscape/interface/local_interface.py

206     by the standard interface: select, add, remove,
clear, finish.
207
208     Public methods:
209     select -- select all network objects that match
object
210     add -- add network object to network
211     remove -- delete all network objects that match
object
212     clear -- delete entire network
213     finish -- code to run after any action to
finalize action
214     '''
215
216     def __init__(self,direc):
217         '''Create a new instance of LocalInterface.
218
219         Arguments:
220         direc - directory where to save the data
shelves'''
221
222         StandardInterface.__init__(self)
223         self.direc = direc
224         self.index_locations = {}
225         for obj_name in INDEXES:
226             self.index_locations[obj_name] =
'{0}/{1}_index.slf'.format(self.direc,obj_name)
227         self.indexes = {}
228         self.data_locations = {}
229         for obj_name in INDEXES:
230             self.data_locations[obj_name] =
'{0}/{1}_data.slf'.format(self.direc,obj_name)
231         self.data = {}
232         self._open_shelves()
233         self.boundary = '|'
234
235     def _open_shelves(self,clear=False):
236         '''Private function to create all the shelves
and indexes that are needed
237         to find and store data.
238
239         Do not call as this function is subject to
removal or change!'''
240         for obj_name in INDEXES:
241             self.data[obj_name] = shelve.open(self.

```


./src/pythoscape/interface/local_interface.py

```
data_locations[obj_name])
242         self.indexes[obj_name] = shelve.open(self.
index_locations[obj_name])
243         if clear:
244             self.indexes[obj_name].clear()
245             self.data[obj_name].clear()
246
247     def _make_unwrapped(self, obj, obj_name):
248         '''Private function that unwraps an object and
then fixes the object
249         to not include attributes that shouldn't be
matched to.
250
251         Do not call as this function is subject to
removal or change!'''
252
253         unwrapped = obj.unwrap()
254         filter=None
255         if 'filter_name' in unwrapped:
256             filter = (unwrapped['filter_name'],unwrapped[
'direction'],unwrapped['filter_value'])
257             del unwrapped['filter_name']
258             del unwrapped['direction']
259             del unwrapped['filter_value']
260         for item in unwrapped.keys():
261             if unwrapped[item] == None: del unwrapped[
item]
262         return unwrapped,filter
263
264     def _build_index(self,unwrapped_obj,obj_name):
265         '''Private function to make an index from an
unwrapped obj.
266
267         Do not call as this function is subject to
removal or change!'''
268
269         return make_index(tuple([unwrapped_obj[elem] for
elem in KEYED_BY[obj_name]]),
                             self.boundary)
270
271     def _build_data_value(self,unwrapped_obj,obj_name):
272         '''Private function to prepare the object for
storage.
273
274         Do not call as this function is subject to
275         removal or change!'''
```

./src/pythoscape/interface/local_interface.py

```
removal or change!'''
276         return_obj = [None]*len(INFORMATION[obj_name])
277         for i,info in enumerate(INFORMATION[obj_name]):
278             if info in COMPLEX:
279                 new_obj_name = COMPLEX[info][1]
280                 if new_obj_name in KEYED_BY and not False
in \
281                     [item in unwrapped_obj for item in
KEYED_BY[new_obj_name]]:
282                     return_obj[i] = set([tuple([
unwrapped_obj[item] for item in KEYED_BY[new_obj_name]]))
283                     elif new_obj_name not in KEYED_BY and not
False in \
284                     [item in unwrapped_obj for item in
INFORMATION[new_obj_name]]:
285                     return_obj[i] = set([tuple([
unwrapped_obj[item] for item in INFORMATION[new_obj_name
]]))
286                     else:
287                         return_obj[i] = set([])
288                     elif info not in unwrapped_obj:
289                         pass
290                     else:
291                         return_obj[i] = unwrapped_obj[info]
292         return return_obj
293
294     def _add_to_indexes(self,unwrapped_obj,obj_name):
295         '''Private function to add an object to the
appropriate indexes.
296
297         Do not call as this function is subject to
removal or change!'''
298         add_indexes = list(set(unwrapped_obj.keys()) &
set(DOUBLE_INDEX.keys()))
299         add_indexes = INDEXES[obj_name]+add_indexes
300         while len(add_indexes):
301             index_name = add_indexes.pop()
302             if index_name in COMPLEX:
303                 add_indexes = add_indexes + list(
INFORMATION[COMPLEX[index_name][1]])
304             elif index_name in unwrapped_obj:
305                 if index_name in DOUBLE_INDEX: put_name =
DOUBLE_INDEX[index_name]
306                 else: put_name = index_name
307                 total_index = make_index([put_name,
```

```

./src/pythoscape/interface/local_interface.py

unwrapped_obj[index_name]],self.boundary)
308         if not total_index in self.indexes[
obj_name]:
309             self.indexes[obj_name][total_index] =
set([])
310             curr_set = self.indexes[obj_name][
total_index]
311             curr_set.add(self._build_index(
unwrapped_obj,obj_name))
312             self.indexes[obj_name][total_index] =
curr_set
313
314     def _save_to_data(self,unwrapped_obj,obj_name):
315         '''Private function to add an object to the shelf.
316
317         Do not call as this function is subject to
removal or change!'''
318
319         data_index = self._build_index(unwrapped_obj,
obj_name)
320         data_point = self._build_data_value(unwrapped_obj
,obj_name)
321         if data_index in self.data[obj_name]:
322             current_data = self.data[obj_name][data_index]
323             for i,info in enumerate(INFORMATION[obj_name
]):
324                 if info in COMPLEX:
325                     current_data[i] = current_data[i] |
data_point[i]
326                 else:
327                     self.data[obj_name][data_index][i] =
data_point[i]
328             self.data[obj_name][data_index] = current_data
329         else:
330             self.data[obj_name][data_index] = data_point
331             self._add_to_indexes(unwrapped_obj,obj_name)
332
333     def _check_filter(self,edge_filter,to_match):
334         '''Private function to check if an edge passes a
filter and return edge
335         weights that do.
336
337         Do not call as this function is subject to
removal or change!'''
338         filter_index = make_index([to_match['source_node'

```

./src/pythoscape/interface/local_interface.py

```
],to_match['target_node'],
339                                     edge_filter[0]],self.
boundary)
340     try:
341         filter_check = self.data['Edge_attribute'][
filter_index]
342     except KeyError:
343         return None
344     weight = {'algorithm':filter_check[2],
345              'weight_value':filter_check[3]}
346     if edge_filter[1] == '<' and filter_check[3] <
edge_filter[2]:
347         return weight
348     elif edge_filter[1] == '<=' and filter_check[3]
<= edge_filter[2]:
349         return weight
350     elif edge_filter[1] == '>' and filter_check[3] >
edge_filter[2]:
351         return weight
352     elif edge_filter[1] == '>=' and filter_check[3]
>= edge_filter[2]:
353         return weight
354     else: return None
355
356     def _find_indexes(self,unwrapped_obj,obj_name):
357         '''Private function that finds all indexes that
match a given
358         object definition.
359
360         Do not call as this function is subject to
removal or change!'''
361         if not unwrapped_obj:
362             return self.data[obj_name].keys()
363         else:
364             all_indexes = set(self.data[obj_name].keys())
365             for obj_key in unwrapped_obj:
366                 ret_index,prim_flag = True,False
367                 if obj_key in INDEXES[obj_name] and
obj_key in DOUBLE_INDEX:
368                     test_index = make_index([DOUBLE_INDEX
[obj_key],unwrapped_obj[obj_key]],self.boundary)
369                 elif obj_key in INDEXES[obj_name] or (
obj_key in COMPLEX_INDEXES and COMPLEX_INDEXES[obj_key]\
370                     in INDEXES[obj_name]):
371                     test_index = make_index([obj_key,
```

./src/pythoscape/interface/local_interface.py

```
unwrapped_obj[obj_key]],self.boundary)
372         elif obj_name in FIND_COMPLEX_INDEXES and
        obj_key in COMPLEX_INDEXES:
373             test_index = make_index([obj_key,
unwrapped_obj[obj_key]],self.boundary)
374             prim_flag = True
375         else:
376             ret_index = False
377             if ret_index and prim_flag and test_index
in self.indexes[FIND_COMPLEX_INDEXES[obj_name]]:
378                 new_indexes = self.indexes[
FIND_COMPLEX_INDEXES[obj_name]][test_index]
379                 if obj_name in DOUBLE_CHECK:
380                     new_indexes = [item for item in
chain(*map(lambda x: x.split(self.boundary),new_indexes))]
381                     possibilities = set([])
382                     for new_index in new_indexes:
383                         possibilities = possibilities |
self.indexes[obj_name][make_index([KEYED_BY[obj_name][0],
new_index],self.boundary)]
384                         all_indexes = all_indexes &
possibilities
385                     elif ret_index and test_index in self.
indexes[obj_name]:
386                         all_indexes = all_indexes & self.
indexes[obj_name][test_index]
387                     elif ret_index:
388                         all_indexes = set([])
389                     return all_indexes
390
391     def _delete_obj(self,unwrapped_obj,obj_name):
392         '''Private function that deletes the appropriate
objects from
393         the data shelves.
394
395         Do not call as this function is subject to
removal or change!'''
396         non_stand = False
397         if obj_name in NOT_STANDARD:
398             non_stand = FIND_NS[obj_name]
399             obj_name = NOT_STANDARD[obj_name]
400             all_indexes = self._find_indexes(unwrapped_obj,
obj_name)
401             if non_stand:
402                 match_to = self._find_indexes(unwrapped_obj,
```

./src/pythoscape/interface/local_interface.py

```
non_stand)
403     all_ns_objs = [tuple(self.data[non_stand][
obj_index]) for obj_index in match_to]
404     if non_stand in KEYED_BY:
405         all_ns_objs = [tuple([one_obj[INFORMATION
[non_stand].index(one_key)]\
406                               for one_key in
KEYED_BY[non_stand]])\
407                               for one_obj in all_ns_objs]
408     ns_index = None
409     for comp_obj in COMPLEX:
410         if non_stand in COMPLEX[comp_obj]:
411             ns_index = INFORMATION[obj_name].
index(comp_obj)
412         if ns_index != None:
413             for obj_index, ns_obj in product(
all_indexes, all_ns_objs):
414                 if ns_obj in self.data[obj_name][
obj_index][ns_index]:
415                     curr_item = self.data[obj_name][
obj_index]
416                     curr_item[ns_index].remove(ns_obj)
417                     self.data[obj_name][obj_index] =
curr_item
418             else:
419                 for obj_index in all_indexes:
420                     del self.data[obj_name][obj_index]
421
422     def _fix_indexes(self, unwrapped_obj, obj_name,
s_indexes):
423         '''Private function that deletes the appropriate
index values
424         that correspond to the objects that are being
deleted.
425
426         Do not call as this function is subject to
removal or change!'''
427     non_stand = False
428     if obj_name in NOT_STANDARD:
429         non_stand = NOT_STANDARD[obj_name]
430     if not unwrapped_obj:
431         self.indexes[obj_name].clear()
432     elif non_stand:
433         to_del = [make_index([item, unwrapped_obj[item
]], self.boundary) for item in unwrapped_obj]
```

./src/pythoscape/interface/local_interface.py

```
434         for index_test in to_del:
435             if index_test in self.indexes[non_stand]:
436                 del self.indexes[non_stand][
index_test]
437             else:
438                 to_del = []
439                 for index_set in self.indexes[obj_name]:
440                     new_set = self.indexes[obj_name][
index_set] - (s_indexes & self.indexes[obj_name][
index_set])
441                     if new_set:
442                         self.indexes[obj_name][index_set] =
new_set
443                     else:
444                         to_del.append(index_set)
445                 for index_set in to_del:
446                     del self.indexes[obj_name][index_set]
447
448     def _del_linked_objs(self, unwrapped_obj, obj_name):
449         '''Private function that deletes the appropriate
linked objects
450         that no longer should exist.
451
452         Do not call as this function is subject to
removal or change!'''
453         if obj_name in LINKED:
454             for new_obj in LINKED[obj_name]:
455                 submit_obj = unwrapped_obj.copy()
456                 for modify in LINKED_MODIFY:
457                     if modify in submit_obj:
458                         submit_obj[LINKED_MODIFY[modify]]
= submit_obj[modify]
459                         del submit_obj[modify]
460                 if obj_name in NOT_STANDARD:
461                     index_name = NOT_STANDARD[new_obj]
462                 else: index_name = obj_name
463                 self._del_linked_objs(submit_obj, new_obj)
464                 starting_indexes = self._find_indexes(
submit_obj, index_name)
465                 self._delete_obj(submit_obj, new_obj)
466                 self._fix_indexes(submit_obj, new_obj,
starting_indexes)
467
468     def _build_objects(self, flat_obj, obj_name):
469         '''Private function that takes the flat
```

./src/pythoscape/interface/local_interface.py

```
information stored
470         in the shelf and creates the Pythoscape-like
dictionary
471         ready to be sent to the StandardInterface.
472
473         Do not call as this function is subject to
removal or change!'''
474
475         return_obj, send_flag = {}, True
476         for subobj_name, subobj_value in zip(INFORMATION[
obj_name], flat_obj):
477             if subobj_name not in COMPLEX:
478                 return_obj[subobj_name] = subobj_value
479             elif subobj_value:
480                 send_flag = False
481             if obj_name in PULL_T00:
482                 item_index = self._build_index(return_obj,
PULL_T00[obj_name])
483                 return_obj.update(self._build_objects(self.
data[PULL_T00[obj_name]] [item_index], PULL_T00[obj_name])).
next())
484             if send_flag:
485                 yield return_obj
486             else:
487                 for subobj_name, subobj_value in zip(
INFORMATION[obj_name], flat_obj):
488                     if subobj_name in COMPLEX:
489                         for data_point in subobj_value:
490                             if COMPLEX[subobj_name][1] in
KEYED_BY:
491                                 item_index = make_index(
data_point, self.boundary)
492                                 data_point = self.data[
COMPLEX[subobj_name][1]] [item_index]
493                                 return_obj.update(self.
_build_objects(list(data_point),
494
COMPLEX[subobj_name][1]).next())
495                                 yield return_obj
496
497         def _check_exists(self, check_obj, obj_type):
498             '''Private function that checks to make sure
that a specific
499             object already exists in the database.
500
```


./src/pythoscape/interface/local_interface.py

```
501         Do not call as this function is subject to
removal or change!'''
502         if isinstance(check_obj,tuple):
503             check_obj = list(check_obj)
504         else:
505             check_obj = [check_obj]
506         obj_index = make_index(check_obj,self.boundary)
507         return obj_index in self.data[obj_type]
508
509     def _run_checks(self,unwrapped_obj,obj_name):
510         '''Private function that checks to make sure
that the appropriate
511         data already exists to add a new object into the
shelf.
512
513         Do not call as this function is subject to
removal or change!'''
514         if obj_name in CHECK:
515             for to_check in CHECK[obj_name]:
516                 if to_check in unwrapped_obj and not self
._check_exists(unwrapped_obj[to_check],OBJ_TRANS[to_check
]):
517                     raise NoAttributeError(obj_name,
to_check,unwrapped_obj[to_check])
518                 elif obj_name in DOUBLE_CHECK:
519                     check_objs,obj_types = [],[]
520                     for double_ch in DOUBLE_CHECK[obj_name]:
521                         check_obj = tuple([unwrapped_obj[subobj]
for subobj in double_ch])
522                         check_objs.append(check_obj)
523                         obj_types.append(OBJ_TRANS[double_ch])
524                     for sub_obj_name in double_ch:
525                         if sub_obj_name in CHECK:
526                             check_objs.append(unwrapped_obj[
sub_obj_name])
527                             check_objs.append(CHECK[
sub_obj_name])
528                     for check_obj,obj_type in zip(check_objs,
obj_types):
529                         if not self._check_exists(check_obj,
obj_type):
530                             raise NoAttributeError(obj_name,
obj_type,check_obj)
531
532     def add(self,obj,obj_name=None):
```

./src/pythoscape/interface/local_interface.py

```
533         '''Adds object to data repository. Function
required for standard_interface.
534
535         Arguments:
536         obj -- obj to be added to repository
537
538         Arguments (with defaults):
539         obj_name (default is None) - if set, this
overwrites the obj_name used to something other than the
540                                     obj_name of the obj
541         '''
542
543         if obj_name == None: obj_name = obj.get_name()
544         if obj_name in NOT_STANDARD:
545             obj_name = NOT_STANDARD[obj_name]
546         new_obj, edge_filter = self._make_unwrapped(obj,
obj_name)
547         self._run_checks(new_obj, obj_name)
548         self._save_to_data(new_obj, obj_name)
549
550         def select(self, obj, obj_name=None):
551             '''Selects objects from data repository.
Function required for standard_interface.
552             For information about how to format object for
selection/deletion, see graph_interface documentation.
553
554             Arguments:
555             obj -- selection/deletion formatted Pythoscape
object
556
557             Arguments (with defaults):
558             obj_name (default is None) - if set, this
overwrites the obj_name used to something other than the
559                                     obj_name of the obj
560
561             Yields:
562             Pythoscape objects that match the inputted object
563             '''
564
565             if obj_name == None: obj_name = obj.get_name()
566             if obj_name in NOT_STANDARD:
567                 obj_name = NOT_STANDARD[obj_name]
568             unwrapped_obj, edge_filter = self._make_unwrapped(
obj, obj_name)
569             all_indexes = self._find_indexes(unwrapped_obj,
```

./src/pythoscape/interface/local_interface.py

```
obj_name)
570     for obj_index in all_indexes:
571         found_obj = self.data[obj_name][obj_index]
572         for built_obj in self._build_objects(
found_obj,obj_name):
573             if not edge_filter:
574                 yield built_obj
575             else:
576                 weight = self._check_filter(
edge_filter,built_obj)
577                 if weight:
578                     built_obj.update(weight)
579                     yield built_obj
580
581     def remove(self,obj,obj_name=None):
582         '''Deletes objects from data repository.
Function required for standard_interface.
For information about how to format object for
selection/deletion, see graph_interface documentation.
583
584         Arguments:
585         obj -- selection/deletion formatted Pythoscape
object
586
587         Arguments (with defaults):
588         obj_name (default is None) - if set, this
overwrites the obj_name used to something other than the
obj_name of the obj
589
590         '''
591
592         if obj_name == None: obj_name = obj.get_name()
593         unwrapped_obj,edge_filter = self._make_unwrapped(
obj,obj_name)
594         if obj_name in NOT_STANDARD:
595             index_name = NOT_STANDARD[obj_name]
596         else: index_name = obj_name
597         starting_indexes = self._find_indexes(
unwrapped_obj,index_name)
598         self._del_linked_objs(unwrapped_obj,obj_name)
599         self._delete_obj(unwrapped_obj,obj_name)
600         self._fix_indexes(unwrapped_obj,obj_name,
starting_indexes)
601
602     def finish(self):
603         '''Run after all objects are submitted to an
interface.
```

./src/pythoscape/interface/local_interface.py

```
604         '''
605         for obj_name in INDEXES:
606             self.indexes[obj_name].sync()
607             self.data[obj_name].sync()
608
609         def clear(self):
610             '''Deletes all data in the interface. Function
611             required for the standard_interface.
612             '''
613             self._open_shelves(clear=True)
614
615         if __name__ == '__main__':
616             pass
```

`./src/pythoscape/interface/mysql_interface.py`

```
1  # Author: Alan Barber
2  # Created on: July 14, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains the MySQL interface.
   This interface uses a MySQL
7  database to store all information used in Pythoscape.
   This class requires that a MySQL
8  server already be installed and the custom Pythoscape
   database already be created on that
9  server. This class also required the MySQLdb Python
   package. This code was tested using
10 MySQLdb 1.2.3, but other version should work as well.
11
12 To set up a Pythoscape on an already existing MySQL
   server, follow the following instructions.
13 Log into MySQL and create a database:
14
15 CREATE DATABASE <database name>;
16
17 After creating the database, format it using the
   provided SQL template database
18 in the auxiliary package called mysql_template.sql. Using
   the command line outside
19 of MySQL, do the following:
20
21 mysql <required arguments for your MySQL server>
   <database name> < mysql_template.sql
22
23 After you have formatted the database, you are ready to
   use the mysql_interface. A general
24 recommendation is to create a different database for
   each set of proteins you are going to
25 analyze. This isn't required, as you can create
   different networks of different protein sets,
26 but this becomes confusing and difficult because the
   searching methods can become clunky
27 with large complex sets of objects.
28
29 The MySQLInterface class was written as a general
   modifiable Pythoscape storage solution.
30 For that reason, several dictionaries contain the
   definitions for how Pythoscape interacts
```

./src/pythoscape/interface/mysql_interface.py

```
31 with the custom MySQL database template. Advanced users
32 can modify both if they choose
33 to modify the graph structure.
34 The MySQL interface is currently the best solution for
35 large networks. It suffers from a
36 speed issue. Namely, depending on the setup of your
37 MySQL server, the queries run on a single
38 processor and it can take a long time to deposit large
39 quantities of data into the database.
40 Future plans involve developing a NoSQL solution that
41 will speed up this process for
42 large networks.
43 Further development on this class will need to make it
44 object oriented instead of function
45 driven.
46 Classes:
47 MySQLInterface -- Interface class (see description above)
48 Example usage:
49 my_loc_inter = MySQLInterface(<database
50 name>,user=<username>,passwd=<password>,port=<port>,host=<
51 host>)
52 For general usage of any interface including a
53 description of how to push, pull and delete
54 network objects, see the graph_interface documentation.
55 '''
56
57 import MySQLdb.cursors
58 import MySQLdb
59 from MySQLdb import OperationalError, ProgrammingError,
60 InterfaceError
61 from standard_interface import StandardInterface
62 from warnings import filterwarnings
63
64 #Suppress warnings
65 filterwarnings('ignore',category=MySQLdb.Warning)
66
67 class MySQLInterface(StandardInterface):
68
69     '''Class that describes the MySQL interface. For an
70     extended explanation,
```

`./src/pythoscape/interface/mysql_interface.py`

```
65     see the documentation for the module. Public methods
are as required
66     by the standard interface: select, add, remove,
clear, finish.
67
68     Public methods:
69     test_type -- tests to see what time of object an
attribute value is
70     force_type -- forces a returned blob to have the
type specified
71     pull_lookups -- takes a standard Pythoscape
object and makes it compatible with the stored procedures
72     re_connect -- re-connects to the MySQL database
73     select -- select all network objects that match
object
74     add -- add network object to network
75     remove -- delete all network objects that match
object
76     clear -- delete entire network
77     finish -- code to run after any action to
finalize action
78     '''
79
80     # Constants -- These are for use with private
functions. It is intended for these to be changed in
accordance with
81     # any changes in the graph structure.
82
83     #ATTRIBUTES -- Standard interface objects that are
treated as a standard "attribute" object.
84     #           Key is object name and value is the
value of the attribute.
85
86     ATTRIBUTES = {'Node_attribute':'attribute_value',
87                  'Edge_attribute':'attribute_value',
88                  'Network_attribute':'attribute_value'}
89
90     #LOOK_UPS -- Where to find object to template
translations. Key is the function and
91     #           value is the name of the dictionary.
92
93     LOOK_UPS = {'add':'MYSQL_ADD_OBJ_DEFS',
94               'select':'MYSQL_SEL_OBJ_DEFS',
95               'delete':'MYSQL_DEL_OBJ_DEFS'}
96
```

./src/pythoscape/interface/mysql_interface.py

```
97     #MYSQL_ADD_FUNCTIONS -- Stored procedures to add
objects to database. Key is the object
98     #                               and value is the name of the
stored procedure.
99
100     MYSQL_ADD_FUNCTIONS = {'Node': 'AddNode',
101                             'Edge': 'AddEdge',
102                             'Node_attribute':
'AddAttribute',
103                             'Edge_attribute':
'AddAttribute',
104                             'Network_attribute':
'AddAttribute',
105                             'Network_edge':
'AddEdgeToNetwork',
106                             'Network': 'AddNetwork',
107                             'Network_node':
'AddNodeToNetwork',
108                             'Identifier': 'AddIdentifier'}
109
110     #MYSQL_SEL_FUNCTIONS -- Stored procedures to select
objects from the database. Key is the object
111     #                               and value is the name of the
stored procedure.
112
113     MYSQL_SEL_FUNCTIONS = {'Node': 'SelectNodeInNetwork',
114                             'Network': 'SelectNetwork',
115                             'Edge': 'SelectEdgeInNetwork',
116                             'Node_attribute':
'SelectNodeAttributeInNetwork',
117                             'Network_attribute':
'SelectNetworkAttribute',
118                             'Edge_attribute':
'SelectEdgeAttributeInNetwork',
119                             'Network_node_attribute':
'SelectNodeAttributeInNetwork',
120                             'Network_edge_attribute':
'SelectEdgeAttributeInNetwork'}
121
122     #MYSQL_DEL_FUNCTIONS -- Stored procedures to delete
objects from the database. Key is the object
123     #                               and value is the name of the
stored procedure.
124
125     MYSQL_DEL_FUNCTIONS = {'Node': 'DeleteNode',
```


./src/pythoscape/interface/mysql_interface.py

```
126                                     'Network_node':
'DeleteNodeInNetwork',
127                                     'Network': 'DeleteNetwork',
128                                     'Edge': 'DeleteEdge',
129                                     'Network_edge':
'DeleteEdgeInNetwork',
130                                     'Attribute': 'DeleteAttribute'}
131
132     #MYSQL_ADD_OBJ_DEFS -- Database object "add"
definitions. Key is the obj and the value is list of
attributes
133     #                                     that are arguments for the
object's stored procedures.
134
135     MYSQL_ADD_OBJ_DEFS = {'Node': ['id'],
136                           'Identifier': ['id',
'identifier_type', 'identifier_value'],
137                           'Network_node': ['network_name', 'id'
],
138                           'Network': ['network_name',
'network_type'],
139                           'Edge': ['source_node', 'target_node'
],
140                           'Network_edge': ['network_name',
'source_node', 'target_node'],
141                           'Node_attribute': [None, 'id', None,
'attribute_name', 'attribute_elem', 'data_type',
'attribute_value'],
142                           'Edge_attribute': [None,
'source_node', 'target_node', 'attribute_name',
'attribute_elem', 'data_type', 'attribute_value'],
143                           'Network_attribute': ['network_name'
, None, None, 'attribute_name', 'attribute_elem', 'data_type',
'attribute_value']}]
144
145     #MYSQL_SEL_OBJ_DEFS -- Database object "select"
definitions. Key is the obj and the value is list of
attributes
146     #                                     that are arguments for the
object's stored procedures.
147
148     MYSQL_SEL_OBJ_DEFS = {'Node': ['network_name', 'id',
'identifier_type', 'identifier_value'],
149                           'Network': ['network_name',
'network_type'],
```

./src/pythoscape/interface/mysql_interface.py

```
150         'Edge':['network_name',
'source_node','target_node','filter_value','direction',
'filter_name'],
151         'Node_attribute':[
'source_node','target_node','filter_value','direction',
'filter_name'],
152         'Network_attribute':[
'source_node','target_node','filter_value','direction',
'filter_name'],
153         'Edge_attribute':[
'source_node','target_node','filter_value','direction',
'filter_name'],
154         'Network_node_attribute':[
'source_node','target_node','filter_value','direction',
'filter_name'],
155         'Network_edge_attribute':[
'source_node','target_node','filter_value','direction',
'filter_name']
156     }
157     #MYSQL_DEL_OBJ_DEFS -- Database object "delete"
definitions. Key is the obj and the value is list of
attributes
158     # that are arguments for the
object's stored procedures.
159
160     MYSQL_DEL_OBJ_DEFS = {'Node':['id','identifier_type',
'identifier_value'],
161                           'Network_node':['network_name',
'id'],
162                           'Network':['network_name'],
163                           'Network_edge':['network_name',
'source_node','target_node','filter_name','direction',
'filter_value'],
164                           'Edge':['source_node',
'target_node'],
165                           'Attribute':['network_name',
'source_node','target_node','attribute_name',
'attribute_elem']}
166
167     #TYPES -- List of types that attribute values can
be. Used to properly store and retrieve blobs.
168     # (Python type,stored type name)
169
170     TYPES = [(str,'string'),(int,'int'),(float,'double'
),(unicode,'string'),(bool,'bool')]
171
```

```

./src/pythoscape/interface/mysql_interface.py

172     #CLEAR_FUNCTION -- Name of stored procedure that
clears that database.
173
174     CLEAR_FUNCTION = 'ClearAll'
175
176     def __init__(self,db,user=None,passwd=None,port=None,
host=None):
177         '''Create a new instance of MySQLInterface.
178
179         Arguments:
180             db - name of Pythoscape database
181
182         Arguments (with defaults):
183             user (default is None) - MySQL user name
184             passwd (default is None) - MySQL password
185             port (default is 3306) - MySQL port
186             host (default is 'localhost') - MySQL host
187         '''
188
189         self.db = db
190         self.user = user
191         self.passwd = passwd
192         self.host = host if host else 'localhost'
193         self.port = port if port else 3306
194         self.Database = None
195         self.Cursor = None
196         self.connect(add=True)
197         StandardInterface.__init__(self)
198
199     def re_connect(self,add=False):
200         '''Reconnect to the MySQL database. Tries twice.
201         '''
202         try:
203             self.connect(add)
204         except OperationalError:
205             self.connect(add)
206
207     def connect(self,add=False):
208         '''Set the MySQL Arguments and connect to MySQL
database.
209         '''
210         if self.Database:
211             self.finish()
212         if not add:
213             cursor_to_use = MySQLdb.cursors.SSDictCursor

```

./src/pythoscape/interface/mysql_interface.py

```
214         else:
215             cursor_to_use = MySQLdb.cursors.DictCursor
216             self.Database = MySQLdb.connect(user=self.user,
217                                             passwd=self.
218                                             passwd,
219                                             db=self.db, port=
220                                             self.port,
221                                             host=self.host,
222                                             cursorclass=cursor_to_use)
223             self.Cursor = self.Database.cursor()
224
225         def type_test(self,value):
226             '''Return which of the types defined in TYPES
227             the value is.
228
229             Arguments:
230                 value - value to find type
231
232             Returns:
233                 string from TYPES that corresponds to the
234                 matching type
235             '''
236             for v_t in self.TYPES:
237                 if type(value) == v_t[0]:
238                     return v_t[1]
239
240         def force_type(self,value,value_type):
241             '''Return value cast as the given type from TYPES.
242
243             Arguments:
244                 value - value to cast
245                 value_type - type to cast value as
246
247             Returns:
248                 value cast as the new type
249             '''
250             for v_t in self.TYPES:
251                 if value_type == v_t[1]:
252                     return v_t[0](value)
253
254         def pull_lookups(self,obj_name,obj,function):
255             '''Takes an object and prepares the arguments
256             for a MySQL query.
```

./src/pythoscape/interface/mysql_interface.py

```
253         Arguments:
254             obj_name - obj_name of object to prepare
255             obj - obj to make lookups from
256             function - 'add','select' or 'delete' -
choose the appropriate function
257
258         Returns:
259             tuple of prepared lookups
260             '''
261         return_data = []
262         obj_defs = obj.unwrap()
263         if hasattr(obj, 'element_obj'):
264             obj_defs['attribute_elem'] = obj.element_obj.
get_name().lower()
265         if obj_name in self.ATTRIBUTES and self.
ATTRIBUTES[obj_name] in obj_defs:
266             obj_defs['data_type'] = self.type_test(
obj_defs[self.ATTRIBUTES[obj_name]])
267             if 'id' in obj_defs:
268                 obj_defs['source_node'] = obj_defs['id']
269             if function in self.LOOK_UPS:
270                 for el_attr in getattr(self, self.LOOK_UPS[
function])[obj_name]:
271                     if el_attr in obj_defs: return_data.
append(obj_defs[el_attr])
272                     else: return_data.append(None)
273             return tuple(return_data)
274
275     def add(self, obj, obj_name=None):
276         '''Adds object to data repository. Function
required for standard_interface.
277
278         Arguments:
279             obj -- obj to be added to repository
280
281         Arguments (with defaults):
282             obj_name (default is None) - if set, this
overwrites the obj_name used to something other than the
obj_name of the obj
283
284             '''
285         if obj_name == None: obj_name = obj.get_name()
286         try:
287             self.Cursor.callproc(self.MYSQL_ADD_FUNCTIONS
[obj_name], self.pull_lookups(obj_name, obj, 'add'))
288         except (ProgrammingError, InterfaceError):
```

./src/pythoscape/interface/mysql_interface.py

```
289         self.re_connect(add=True)
290         self.Cursor.callproc(self.MYSQL_ADD_FUNCTIONS
[obj_name],self.pull_lookups(obj_name,obj, 'add'))
291         while self.Cursor.nextset() is not None: pass
292         self.Database.commit()
293
294     def _run_select(self,obj,obj_name,lookups):
295         '''Private function that calls the select stored
procedure.
296
297         Do not call as this function is subject to
removal or change!'''
298         self.Cursor.callproc(self.MYSQL_SEL_FUNCTIONS[
obj_name],lookups)
299         while 1:
300             result = self.Cursor.fetchone()
301             if result:
302                 yield result
303             else: break
304         while self.Cursor.nextset() is not None: pass
305
306     def select(self,obj,obj_name=None):
307         '''Selects objects from data repository.
Function required for standard_interface.
308         For information about how to format object for
selection/deletion, see graph_interface documentation.
309
310         Arguments:
311         obj -- selection/deletion formatted Pythoscape
object
312
313         Arguments (with defaults):
314         obj_name (default is None) - if set, this
overwrites the obj_name used to something other than the
315             obj_name of the obj
316
317         Yields:
318         Pythoscape objects that match the inputted object
'''
319
320         self.re_connect()
321         if obj_name == None: obj_name = obj.get_name()
322         lookups = self.pull_lookups(obj_name,obj, 'select')
323         for item in self._run_select(obj,obj_name,lookups
):
324             if 'attribute_value' in item and 'data_type'
```

./src/pythoscape/interface/mysql_interface.py

```
in item:
325         item['attribute_value'] = self.force_type
(item['attribute_value'], item['data_type'])
326         if 'weight_value' in item:
327             item['weight_value'] = float(item[
'weight_value'])
328         yield item
329         if obj_name == 'Edge' and lookups[1] and not
lookups[2]:
330             lookups = list(lookups)
331             lookups[2], lookups[1] = lookups[1], None
332             for item in self._run_select(obj, obj_name,
tuple(lookups)):
333                 yield item
334             self.finish()
335
336     def remove(self, obj, obj_name=None):
337         '''Deletes objects from data repository.
Function required for standard_interface.
338         For information about how to format object for
selection/deletion, see graph_interface documentation.
339
340         Arguments:
341         obj -- selection/deletion formatted Pythoscape
object
342
343         Arguments (with defaults):
344         obj_name (default is None) - if set, this
overwrites the obj_name used to something other than the
345             obj_name of the obj
346         '''
347         self.re_connect()
348         if obj_name == None: obj_name = obj.get_name()
349         self.Cursor.callproc(self.MYSQL_DEL_FUNCTIONS[
obj_name], self.pull_lookups(obj_name, obj, 'delete'))
350         while self.Cursor.nextset() is not None: pass
351         self.Database.commit()
352         self.finish()
353
354     def finish(self):
355         '''Run after any object is submitted to the
interface.
356         Function required for standard_interface, but
doesn't have to do anything if this isn't required.
357         '''
```

./src/pythoscape/interface/mysql_interface.py

```
358         try:
359             self.Database.close()
360         except ProgrammingError:
361             pass
362
363     def clear(self):
364         '''Deletes all data in the interface. Function
365         required for the standard_interface.
366         '''
367         self.re_connect()
368         self.Cursor.callproc(self.CLEAR_FUNCTION, ())
369         self.finish()
370 if __name__ == '__main__':
371
372     pass
```


./src/pythoscape/interface/standard_interface.py

```
1  # Author: Alan Barber
2  # Created on: July 14, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains the standard
   interface. This interface inherits from the
7  graph interface and handles data so that it can be
   passed into a "database"-like interface.
8  In this case, a "database"-like interface is one where
   complex network objects have to
9  be cropped down to ones where there is only a single
   relationship provided at a time.
10 This allows these data points to be stored one at a
   time. The same is true for selection,
11 in reverse. This module expects that relationships are
   provided one at a time and then
12 it builds the entire complex object to be passed back to
   the plugin.
13
14 This interface also introduces a few new objects that
   must be handled by inheriting classes:
15 Network_node -- a node that has a single network
   associated with it
16 Network_edge -- an edge that has a single network
   associated with it
17 Node_attribute -- an attribute associated with a node
18 Edge_attribute -- an attribute associated with an edge
19 Network_attribute -- an attribute associated with a
   network
20
21 This interface doesn't explicitly link to a storage
   source and so it cannot be used by itself.
22 As such the functions add, select and remove must be set
   by an inheriting class.
23 The overwritten functions must be overwritten by
   functions that have the following form:
24
25 def add(self, obj, obj_name=None):
26     return None
27
28 def select(self, obj, obj_name=None):
29     yield selected_obj
30
```

./src/pythoscape/interface/standard_interface.py

```
31 def remove(self,obj,obj_name=None):
32     return None
33
34 This class also provides an example for how to properly
   inherit the graph_interface.
35
36 Classes:
37 StandardInterface -- Interface class (see description
   above)
38
39 Example usage:
40 my_loc_inter = StandardInterface()
41
42 For general usage of any interface including a
   description of how to push, pull and delete
43 network objects, see the graph_interface documentation.
44 '''
45
46 from pythoscape.main.graph_interface import GraphInterface
47 from pythoscape.main.dict_property import dict_property
48
49 class StandardInterface(GraphInterface):
50
51     '''Class that describes the MySQL interface. For an
   extended explanation,
52     see the documentation for the module. Public methods
   are as required
53     by the standard interface: select, add, remove,
   clear, finish.
54
55     Public methods:
56         add -- must be overwritten (see above)
57         select -- must be overwritten (see above)
58         remove -- must be overwritten (see above)
59         node_interface -- required node interface
60         network_interface -- required network interface
61         edge_interface -- required edge interface
62         attribute_interface -- required attribute
   interface
63     '''
64
65     def __init__(self):
66
67         GraphInterface.__init__(self)
68
```

./src/pythoscape/interface/standard_interface.py

```
69     def add(self, obj, obj_name=None):
70         '''Adds object to data repository. Function
71         required for standard_interface.
72         This function only contains pass and needs to be
73         overwritten to be functional.
74
75         Arguments:
76         obj -- obj to be added to repository
77
78         Arguments (with defaults):
79         obj_name (default is None) - if set, this
80         overwrites the obj_name used to something other than the
81         obj_name of the obj
82         '''
83         pass
84
85     def select(self, obj, obj_name=None):
86         '''Selects objects from data repository.
87         Function required for standard_interface.
88         For information about how to format object for
89         selection/deletion, see graph_interface documentation.
90         This function only returns an empty list and
91         needs to be overwritten to be functional.
92
93         Arguments:
94         obj -- selection/deletion formatted Pythoscape
95         object
96
97         Arguments (with defaults):
98         obj_name (default is None) - if set, this
99         overwrites the obj_name used to something other than the
100         obj_name of the obj
101
102         Yields:
103         Pythoscape objects that match the inputted object
104         '''
105         return []
106
107     def remove(self, obj, obj_name=None):
108         '''Deletes objects from data repository.
109         Function required for standard_interface.
110         For information about how to format object for
111         selection/deletion, see graph_interface documentation.
112         This function only contains pass and needs to be
113         overwritten to be functional.
```

./src/pythoscape/interface/standard_interface.py

```
103
104     Arguments:
105     obj -- selection/deletion formatted Pythoscape
object
106
107     Arguments (with defaults):
108     obj_name (default is None) - if set, this
overwrites the obj_name used to something other than the
109                                     obj_name of the obj
110     '''
111     pass
112
113     @dict_property
114     def node_interface():
115         '''Standard and required Pythoscape object
interface (Node) which contains
116             the fset, fget and fdel methods.'''
117         def fset(self, key, value):
118             '''Standard and required fset method. Adds
node and associated information to storage.
119
120             Arguments:
121             key -- Node object to be added to
storage
122
123                 value -- assumed to be True
124             '''
125             new_node = key.do_copy()
126             new_node.clear_networks()
127             new_node.clear_identifiers()
128             self.add(new_node, 'Node')
129             if key.networks():
130                 for network in key.networks():
131                     new_node = key.do_copy()
132                     new_node.clear_identifiers()
133                     new_node.crop_networks(network)
134                     self.add(new_node, 'Network_node')
135             if key.identifiers():
136                 for identifier in key.identifiers():
137                     new_node = key.do_copy()
138                     new_node.clear_networks()
139                     new_node.crop_identifiers(identifier)
140                     self.add(new_node, 'Identifier')
141         def fget(self, key):
142             '''Standard and required fget method.
Selects node and associated information from storage.
```

./src/pythoscape/interface/standard_interface.py

```
142
143         Arguments:
144             key -- Node object to defines nodes
to select from storage
145
146         Yields:
147             selected nodes
148         '''
149         current_node = None
150         for result in self.select(key):
151             if current_node == None:
152                 current_node = self.Node(result)
153             elif result['id'] != current_node.id:
154                 yield current_node
155                 current_node = self.Node(result)
156             if 'identifier_type' in result and result
['identifier_type']:
157                 current_node.add_identifiers([self.
Identifier(result)])
158             if 'network_name' in result and result[
'network_name']:
159                 current_node.add_networks([self.
Network(result)])
160             if current_node != None:
161                 yield current_node
162             else:
163                 raise StopIteration
164         def fdel(self, key):
165             '''Standard and required fdel method.
Deletes node and associated information in storage.
166
167             Arguments:
168                 key -- Node object to defines nodes
to delete from storage
169             '''
170             if key.networks():
171                 for network in key.networks():
172                     new_node = key.do_copy()
173                     new_node.crop_networks(network)
174                     self.remove(new_node, 'Network_node')
175             else:
176                 self.remove(key, 'Node')
177         return locals()
178
179     @dict_property
```

./src/pythoscape/interface/standard_interface.py

```
180     def network_interface():
181         '''Standard and required Pythoscape object
182         interface (Network) which contains
183         the fset, fget and fdel methods.'''
184         def fset(self, key, value):
185             '''Standard and required fset method. Adds
186             network and associated information to storage.
187
188             Arguments:
189                 key -- Network object to be added to
190                 storage
191                 value -- assumed to be True
192             '''
193             self.add(key, 'Network')
194         def fget(self, key):
195             '''Standard and required fget method.
196             Selects network and associated information from storage.
197
198             Arguments:
199                 key -- Network object to defines
200                 nodes to select from storage
201
202             Yields:
203                 selected networks
204             '''
205             for result in self.select(key):
206                 yield self.Network(result)
207         def fdel(self, key):
208             '''Standard and required fdel method.
209             Deletes network and associated information in storage.
210
211             Arguments:
212                 key -- Network object to defines
213                 nodes to delete from storage
214             '''
215             if not key.network_name and key.network_type:
216                 try:
217                     raise ValueError;
218                 except ValueError as err:
219                     if not err.args:
220                         err.args=(' ',)
221                     err.args = (err.args[0] + ""
222                                 This graph interface does not allow a network to be
223                                 deleted based on type without a name.
224                                 """,)+err.args[1:]
```

./src/pythoscape/interface/standard_interface.py

```
217         raise
218         self.remove(key)
219     return locals()
220
221     @dict_property
222     def edge_interface():
223         '''Standard and required Pythoscape object
224         interface (Edge) which contains
225         the fset, fget and fdel methods.'''
226         def fset(self, key, value):
227             '''Standard and required fset method. Adds
228             edge and associated information to storage.
229
230             Arguments:
231                 key -- Edge object to be added to
232                 storage
233                 value -- assumed to be True
234             '''
235             new_edge = key.do_copy()
236             new_edge.clear_networks()
237             self.add(new_edge, 'Edge')
238             if key.networks():
239                 for network in key.networks():
240                     new_edge = key.do_copy()
241                     new_edge.crop_networks(network)
242                     self.add(new_edge, 'Network_edge')
243             #ADD weight input code
244         def fget(self, key):
245             '''Standard and required fget method.
246             Selects edge and associated information from storage.
247
248             Arguments:
249                 key -- Edge object to defines nodes
250                 to select from storage
251
252             Yields:
253                 selected edges
254             '''
255             current_edge = None
256             for result in self.select(key):
257                 result['source_node'] = self.Node(id=
258                 result['source_node'])
259                 result['target_node'] = self.Node(id=
260                 result['target_node'])
261                 if 'algorithm' in result and
```

./src/pythoscape/interface/standard_interface.py

```
'weight_value' in result:
255         result['weight'] = self.Weight(result)
256         if current_edge == None:
257             current_edge = self.Edge(result)
258         elif result['source_node'] !=
current_edge.source_node or result['target_node'] !=
current_edge.target_node:
259             yield current_edge
260             current_edge = self.Edge(result)
261             if 'network_name' in result and result[
'network_name']:
262                 current_edge.add_networks([self.
Network(result)])
263             if current_edge != None:
264                 yield current_edge
265                 raise StopIteration
266
267         def fdel(self, key):
268             '''Standard and required fdel method.
Deletes edge and associated information in storage.
269
270             Arguments:
271             key -- Edge object to defines nodes
to delete from storage
272             '''
273             if key.networks():
274                 for network in key.networks():
275                     new_edge = key.do_copy()
276                     new_edge.crop_networks(network)
277                     self.remove(new_edge, 'Network_edge')
278             else:
279                 self.remove(key, 'Edge')
280             return locals()
281
282         @dict_property
283         def attribute_interface():
284             '''Standard and required Pythoscape object
interface (Attribute) which contains
285             the fset, fget and fdel methods.'''
286             def fset(self, key, value):
287                 '''Standard and required fset method. Adds
attribute to storage.
288
289                 Arguments:
290                 key -- Attribute object to be added
```


./src/pythoscape/interface/standard_interface.py

```
to storage
291         value -- assumed to be True
292         """
293         att_type = '{0}_{1}'.format(key.element_obj.
get_name(), 'attribute')
294         if key.element_obj.get_name() == 'Node' or
key.element_obj.get_name() == 'Edge':
295             key.element_obj.clear_networks()
296             if key.element_obj.get_name() == 'Node':
297                 key.element_obj.clear_identifiers()
298             self.add(key, att_type)
299         def fget(self, key):
300             '''Standard and required fget method.
Selects attribute and associated information from storage.
301
302             Arguments:
303                 key -- Attribute object to defines
nodes to select from storage
304
305             Yields:
306                 selected attributes
307             """
308             att_type = '{0}_{1}'.format(key.element_obj.
get_name(), 'attribute')
309             if hasattr(key.element_obj, 'networks') and
key.element_obj.networks():
310                 func = self.select(key, '{0}_{1}'.format(
'Network', att_type.lower()))
311             else:
312                 func = self.select(key, att_type)
313             for result in func:
314                 if key.element_obj.get_name() == 'Edge':
315                     result['source_node'] = self.Node(id=
result['source_node'])
316                     result['target_node'] = self.Node(id=
result['target_node'])
317                 if 'algorithm' in result and
'weight_value' in result:
318                     result['weight'] = self.Weight(
result)
319                 result['element_obj'] = getattr(self, key.
element_obj.get_name())(result)
320                 yield self.Attribute(result)
321         def fdel(self, key):
322             '''Standard and required fdel method.
```

./src/pythoscape/interface/standard_interface.py

```
Deletes attributes in storage.
323
324         Arguments:
325             key -- Attribute object to defines
nodes to delete from storage
326         """
327         self.remove(key, 'Attribute')
328         return locals()
329
330 if __name__ == '__main__':
331
332     pass
```

C.5.3. ./src/pythoscape/main modules

./src/pythoscape/main/environments.py

```
1  # Author: Alan Barber
2  # Created on: September 28, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains the two environments
   used in Pythoscape to modify
7  graphs stored in interfaces by running plug-ins.
8
9  PythoscapeEnvironment is used to access or modify the
   entire graph. PythoscapeNetwork is used
10 to access or modify a previously defined subset of the
   graph.
11
12 Classes:
13 NetworkError -- Error used by PythoscapeNetwork.
14 PythoscapeEnvironment -- Environment used to modify or
   access any part of the graph.
15 PythoscapeNetwork -- Environment used to modify or
   access a pre-defined subset of the graph.
16
17 PythoscapeEnvironment example usage:
18 #instantiate
19 my_pytho = PythoscapeEnvironment(<interface>)
20 #run plugin
21 my_pytho.execute_plugin(<plug-in>)
22
23 PythoscapeNetwork example usage:
24 #instantiate
25 my_net = PythoscapeNetwork(<network name>,<interface>)
26 #run plugin
27 my_net.execute_plugin(<plug-in>)
28 '''
29
30 from errors import ObjectError, RequiredError, OptionError
31 import datetime
32 from network_objects import NETWORK_DEFS
33
34 class NetworkError(Exception):
35     ''' Exception that is used by the PythoscapeNetwork
   class. Specifically, this error is used
36     when a network doesn't exist that is being referenced.
37
38     Arguments:
```

./src/pythoscape/main/environments.py

```
39         value - name of unknown referenced network'''
40
41     def __init__(self,value):
42         self.value = value
43     def __str__(self):
44         return ''.join(["The following network, ",str(
self.value),", does not exist. This is a fatal error."])
45
46 class PythoscapeEnvironment(object):
47     '''PythoscapeEnvironment class. This environment is
used to modify or access any part
48     of the graph.
49
50     Public methods (should be only used for inherited
environment classes and not generally):
51         run_command -- runs a command on a single object
52         set_plugin -- sets up a plugin to run
53
54     Public methods (general use):
55         execute_plugin -- run a plugin
56         print_logs -- print environment logs
57
58     PythoscapeEnvironment example usage:
59     #instantiate
60     my_pytho = PythoscapeEnvironment(<interface>)
61     #run plugin
62     my_pytho.execute_plugin(<plug-in>)
63     '''
64
65     #Commands available to use by graph_interface
66     COMMANDS = set(['push','pull','delete'])
67
68     def __init__(self, graph_interface):
69         '''Create a new instance of PythoscapeEnvironment.
70
71         Arguments:
72             graph_interface -- instantiated
graph_interface'''
73
74         self.graph_interface = graph_interface
75         self.plugin = None
76         self.plugin_environment = 'Pythoscape'
77         self.graph_interface.push(self.graph_interface.
Network(network_name='*blank*'))
78         self.graph_interface.finish()
```

./src/pythoscape/main/environments.py

```
79
80     def run_command(self,command,obj):
81         '''Runs a command on an obj using the
graph_interface.
82
83         Arguments:
84         obj -- obj to be run
85         command -- command to be run
86         '''
87         if hasattr(obj,'get_name') and obj.get_name() in
NETWORK_DEFS:
88             return getattr(self.graph_interface,command)(
obj)
89         elif len(obj) == 2:
90             return getattr(self.graph_interface,command)(
obj[1],obj[0])
91         else:
92             raise ObjectError(str(obj))
93
94     def execute_plugin(self,plugin,comments=None,
comments_off=False):
95         '''Executes a plugin.
96
97         Arguments:
98         plugin -- plugin to be run
99
100        Arguments (with defaults):
101        comments (default is None) -- comments to
include in logs with plugin run
102        comments_off (default is False) -- turn off
plugin logging
103        '''
104         if not getattr(plugin,self.plugin_environment):
105             raise RequiredError('plugin.type',self.
plugin_environment)
106         self.plugin = plugin
107         if not comments_off:
108             if not comments: comments = ''
109             comments = ' '.join(['Plugin run:',self.
plugin.__class__.__name__, 'Plugin description',self.
plugin.description,
110                                'Comments:',comments,
'Arguments:',str(self.plugin._arguments)])
111         network_att = self.graph_interface.Network(
network_name='*blank*')
```

./src/pythoscape/main/environments.py

```
112         comment_att = self.graph_interface.Attribute(  
element_obj=network_att, attribute_name=str(datetime.  
datetime.now()), attribute_value=comments)  
113         self.graph_interface.push(comment_att)  
114         self.graph_interface.finish()  
115         for command, func in self.plugin:  
116             if command not in self.COMMANDS:  
117                 raise OptionError(command, 'command type')  
118             for obj in func:  
119                 pos_gen = self.run_command(command, obj)  
120                 if pos_gen: self.plugin.outgoing = pos_gen  
121                 self.graph_interface.finish()  
122  
123     def print_logs(self, pos_file=None):  
124         '''Print environment logs.  
125  
126         Arguments (with defaults):  
127         pos_file (default is None) -- print comments to  
a file instead of screen  
128         '''  
129  
130         network_name = self.network_name if hasattr(self,  
'network_name') else '*blank*'  
131         network_att = self.graph_interface.Network(  
network_name=network_name)  
132         comment_att = self.graph_interface.Attribute(  
element_obj=network_att)  
133         gen = self.graph_interface.pull(comment_att)  
134         if pos_file:  
135             f = open(pos_file, 'w')  
136             for att in gen: f.write('%s\n%s\n\n' %(att.  
attribute_name, att.attribute_value))  
137             f.close()  
138         else:  
139             for att in gen: print '%s\n%s\n\n' %(att.  
attribute_name, att.attribute_value)  
140  
141     class PythoscapeNetwork(PythoscapeEnvironment):  
142         '''PythoscapeNetwork class. This class is used to  
access or modify a previously  
143         defined subset of the graph.  
144  
145         Public methods:  
146         execute_plugin -- run a plugin  
147
```

./src/pythoscape/main/environments.py

```
148     PythoscapeNetwork example usage:
149     #instantiate
150     my_net = PythoscapeNetwork(<network name>,<interface>)
151     #run plugin
152     my_net.execute_plugin(<plug-in>)
153     '''
154
155     #NETWORK_INCLUDED is a dictionary that defines how
156     networks attributes are handled.
157     #Key is the object, value is None is has network as
158     attribute or it is the attribute name
159     #of the attribute that holds the network.
160     NETWORK_INCLUDED = {'Edge':None, 'Node':None,
161     'Attribute':'element_obj'}
162
163     def __init__(self, name, graph_interface,
164     network_type=None, create_network=False):
165         '''Create a new instance of PythoscapeNetwork.
166
167         Arguments:
168             name - network name
169             graph_interface -- instantiated
170             graph_interface
171
172         Arguments (with defaults):
173             network_type (default is None) -- network
174             type, not necessary unless you want it associated
175             when
176             creating the network
177             create_network (default is False) -- create
178             the network as well as access it
179         '''
180
181         PythoscapeEnvironment.__init__(self,
182         graph_interface)
183         self.plugin_environment = 'PythoscapeNetwork'
184         self.access_net = self.graph_interface.Network(
185         network_name=name,network_type=network_type)
186         if create_network:
187             self.graph_interface.push(self.access_net,
188             'Network')
189             self.graph_interface.finish()
190
191     def _new_generator(self,gen_to_modify):
192         '''Private function that makes a new generator
```


./src/pythoscape/main/environments.py

```
from the yielded objects
    with the network information included.
182
183
184     Do not call as this function is subject to
removal or change!'''
185     for item in gen_to_modify:
186         item_name = item.get_name()
187         if item_name in self.NETWORK_INCLUDED:
188             if not self.NETWORK_INCLUDED[item_name]:
189                 item.clear_networks()
190                 item.add_networks(self.network_def)
191             else:
192                 getattr(item, self.NETWORK_INCLUDED[
item_name]).clear_networks()
193                 getattr(item, self.NETWORK_INCLUDED[
item_name]).add_networks(self.network_def)
194         yield item
195
196     def execute_plugin(self, plugin, comments=None,
comments_off=False):
197         '''Executes a plugin.
198
199         Arguments:
200         plugin -- plugin to be run
201
202         Arguments (with defaults):
203         comments (default is None) -- comments to
include in logs with plugin run
204         comments_off (default is False) -- turn off
plugin logging
205         '''
206
207         if not getattr(plugin, self.plugin_environment):
208             raise RequiredError('plugin.type', self.
plugin_environment)
209         self.plugin = plugin
210         if not comments_off:
211             if not comments: comments = ''
212             comments = ' '.join(['Plugin run:', self.
plugin.__class__.__name__, 'Plugin description', self.
plugin.description,
213                                 'Comments:', comments,
'Arguments:', str(self.plugin._arguments)])
214             comment_att = self.graph_interface.Attribute(
element_obj=self.access_net, attribute_name=str(datetime.
```

./src/pythoscape/main/environments.py

```
datetime.now()),attribute_value=comments)
215         self.graph_interface.push(comment_att)
216         self.graph_interface.finish()
217     for command,func in self.plugin:
218         if command not in self.COMMANDS:
219             raise OptionError(command,'command type')
220         try:
221             self.network_def = [self.graph_interface.
pull(self.access_net, 'Network').next()]
222             self.graph_interface.finish()
223         except StopIteration:
224             raise NetworkError(self.access_net.
network_name)
225     for obj in func:
226         obj_name = obj.get_name()
227         if obj_name in self.NETWORK_INCLUDED:
228             to_mod = getattr(obj,self.
NETWORK_INCLUDED[obj_name]) if self.NETWORK_INCLUDED[
obj_name] else obj
229             to_mod.clear_networks()
230             to_mod.add_networks(self.network_def)
231             pos_gen = self.run_command(command,obj)
232             if pos_gen: self.plugin.outgoing = self.
_new_generator(pos_gen)
233         self.graph_interface.finish()
```

./src/pythoscape/main/errors.py

```
1  # Author: Alan Barber
2  # Created on: September 28, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains custom errors used by
   Pythoscape.
7
8  Classes:
9  UnwrapError -- when an object cannot be unwrapped due to
   complex attributes
10 ObjectError -- if an object not defined is called
11 MissingRequiredElement -- if a required attribute to
   create an object is not given
12 NodeEqualityError -- if an edge does not connect two
   unequal nodes
13 AttributeError -- if an inappropriate object is
   attached to an attribute
14 OptionError -- a provided option is not allowed by an
   object
15 RequiredError -- a requirement for a class is not
   defined (used by plugin management)
16
17 Example usage:
18 Nothing in this class is called directly.
19 '''
20
21 class ComplexElementError(Exception):
22     '''Exception thrown when a defined object includes
   complex elements that are not also
23     defined.
24
25     Arguments:
26         element - complex attribute'''
27     def __init__(self, element):
28         self.element = element
29     def __str__(self):
30         return "All complex elements must also be
   defined. "
31
32 class UnwrapError(Exception):
33     '''Exception thrown when an object cannot be
   unwrapped due to a surplus of
34     complex attributes, (i.e. greater than 1 in a list).
```

```
35
36 Arguments:
37     element - complex attribute
38     value - number of items instead of 1'''
39 def __init__(self, element, value):
40     self.element = element
41     self.value = value
42 def __str__(self):
43     return ''.join(["While unwrapping, the following
complex attribute, ", str(self.element), ", was longer
than 1 attribute. It had ", str(self.value), " items. This
is a fatal error."])
44
45 class ObjectError(Exception):
46     '''Exception thrown if there is an attempt to call
an object that is not defined.
47
48     Arguments:
49         value -- name of object'''
50
51     def __init__(self, value):
52         self.value = value
53     def __str__(self):
54         return ''.join(["The following object, ", str(self
.value), ", is not allowed. This is a fatal error."])
55
56 class MissingRequiredElement(Exception):
57     '''Exception thrown if a required object attribute
is not provided.
58
59     Arguments:
60         value -- name of object attribute
61         element -- name of object'''
62
63     def __init__(self, value, element):
64         self.value = value
65         self.element = element
66     def __str__(self):
67         return ''.join(["The object attribute, ", str(self
.value), ", is required to create the object, ", str(self.
element), ". This is a fatal error."])
68
69 class NodeEqualityError(Exception):
70     '''Exception thrown if an edge does not connect two
unequal nodes.
```

```
71
72     Arguments:
73         None'''
74
75     def __str__(self):
76         return "An edge must connect two non-equal
77 nodes. This is a fatal error."
78
79 class AttributeError(Exception):
80     '''Exception thrown if an inappropriate object is
81 attached to an attribute.
82
83     Arguments:
84         value -- name of inappropriate object'''
85
86     def __init__(self,value):
87         self.value = value
88     def __str__(self):
89         return ''.join(["The following class type, ",str(
90 self.value),", is not allowed to have attributes. This
91 is a fatal error."])
92
93 class OptionError(Exception):
94     '''Exception thrown if a provided option is not
95 allowed by an object.
96
97     Arguments:
98         value -- value of provided option
99         element -- object attribute that does not allow
100 option given'''
101
102     def __init__(self,value,element):
103         self.value = value
104         self.element = element
105     def __str__(self):
106         return ''.join(["The following option is not
107 allowed, ",str(self.value),", for the argument, ", str(
108 self.element),". This is a fatal error."])
109
110 class RequiredError(Exception):
111     '''Exception thrown if a requirement for a class is
112 not defined. This is used
113 by the plugin management tests.
114
115     Arguments:
```

./src/pythoscape/main/errors.py

```
107         value -- given value
108         required -- class requirement'''
109
110     def __init__(self,value,required):
111         self.value = value
112         self.required = required
113     def __str__(self):
114         return ''.join(["The following value, ",str(self.
value),", does not meet the requirement, ", str(self.
element),". This is a fatal error."])
```

./src/pythoscape/graph.py

```
1  # Author: Alan Barber
2  # Created on: September 28, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: The graph class packages all the network
   object classes from network object into one class that
   can be
7  easily inherited.
8
9  Classes:
10 Graph -- Packages up all networks objects for easy
   inheritance and adds the ability to call objects using
11         a variety of methods using create_netobj.
12
13 Functions:
14 create_netobj -- Creates a new populated object from
   inputs.
15
16 Graph example usage:
17 my_node = Graph.Node(id='12345')
18 my_edge = Graph.Edge(my_node)
19 '''
20
21 from errors import ObjectError
22 from network_objects import NETWORK_DEFS
23
24 def create_netobj(element, elem_obj=None, **k):
25     '''Takes an object definition from either arguments
   or a dictionary and returns the created object.
26
27     Arguments:
28         element -- object name
29
30     Arguments (with defaults):
31         elem_obj (default is None) -- object definitions
   from dictionary
32
33     Other arguments:
34         **k -- object definitions
35
36     Returns:
37         instantiated object
38     '''
```

./src/pythoscape/graph.py

```
39
40     if element not in NETWORK_DEFS:
41         raise ObjectError(element)
42     if elem_obj:
43         if isinstance(elem_obj, NETWORK_DEFS[element]):
44             return elem_obj
45         elif hasattr(elem_obj, 'setdefault'):
46             k.update(elem_obj)
47         else:
48             raise TypeError("When inputting an object, a
dictionary-like object or a network object must be used
to create an object.")
49     if not k:
50         return NETWORK_DEFS[element](*[None]*NETWORK_DEFS
[element].ELEMENT_LEN)
51     attributes = []
52     for el_attr in NETWORK_DEFS[element].ELEMENTS:
53         if el_attr in NETWORK_DEFS[element].REDIRECT and
el_attr in k and k[el_attr]:
54             attributes.append(create_netobj(NETWORK_DEFS[
element].REDIRECT[el_attr], k[el_attr]))
55         else:
56             attributes.append(k[el_attr] if el_attr in k
else None)
57     return NETWORK_DEFS[element](*attributes)
58
59 class Graph(object):
60     '''Graph class. This is a class that is a container
for all the network object classes.
61     It exists for easy inheritance of all the classes as
a group. The four main object classes
62     are packaged here and are described more extensively
in the documentation for the
63     network_objects module. Those classes are Node,
Edge, Network, Attribute.
64
65     Public methods:
66         None
67
68     Example Usage:
69     my_node = Graph.Node(id='12345')
70     my_edge = Graph.Edge(my_node)
71     '''
72
73     def __init__(self):
```


./src/pythoscape/graph.py

```
74         for element in NETWORK_DEFS:
75             setattr(self,element,self._element_interface(
element))
76
77     def _element_interface(self,element):
78         def c_netobj(elem_obj=None, **k):
79             return create_netobj(element,elem_obj,**k)
80         return c_netobj
81
82
```

./src/pythoscape/main/graph_interface.py

```
1  # Author: Alan Barber
2  # Created on: September 28, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains the graph_interface
   class, which extends the graph class into a common
   interface
7  for network storage. This module is non-functional as
   provided. It must be inherited by a more specific class
   which
8  provides the functions to properly access specific data
   sets. Instructions are provided below on extending this
   class.
9
10 This class (or its inheritors) is used by the
   environment classes to run plugins or can be used
   directly to interact
11 with data stored by Pythoscape.
12
13 Using a configured graph_interface
14 -----
15 ---
16 Once graph_interface has been configured for a specific
   data source, the following operations can
17 manipulate the data in the source.
18
19 To add data into the graph:
20 graph_interface.push(<Pythoscape object>)
21
22 To select data from the graph:
23 data_gen = graph_interface.pull(<Pythoscape selection
   object>)
24 for obj in data_gen:
25     print obj
26
27 To delete data from the graph:
28 graph_interface.delete(<Pythoscape selection object>)
29
30 A Pythoscape selection object is an object where defined
   object attributes represent searching parameters and
   empty object attributes represent wildcards. For
   example, the node selection object Node(id='12345')
   would search for a
```

./src/pythoscape/main/graph_interface.py

```
31 node with id of '12345'. The node selection object
   Node() would search for any node with any id.
32
33 Push and delete do not return anything. Select returns a
   generator of all the selected objects.
34
35 To delete all data in the interface:
36 graph_interface.clear()
37
38 Configuring a graph_interface
39 -----
   --
40 The graph_interface has four null object interfaces that
   must be overwritten to interact with the
41 data source.
42
43 Those are graph_interface.node_interface,
   graph_interface.edge_interface,
   graph_interface.network_interface
44 and graph_interface.attribute_interface. Each of these
   accepts objects based a "dict_property" system.
45 The function dict_property should be imported from the
   module main.dict_property and used as below.
46
47 @dict_property
48 def example_interface():
49     """ """
50     def fset(self, key, value):
51         pass
52     def fget(self, key):
53         yield None
54     def fdel(self, key):
55         pass
56     return locals()
57
58 The fset function should store the respective object.
   The fget function should yield the selected objects.
59 The fdel function should delete the selected objects.
60
61 For a functioning version of this, see the standard
   interface.
62
63 The class method graph_interface.finish can also be
   overwritten if need be. This method is run
64 after each object is passed to the interface. If the
```

./src/pythoscape/main/graph_interface.py

```
65 data source needs to process the data
66 after each object is pass to it, code can be written
67 here to manage that.
68
69 Functions:
70 dict_property - used allow DictProperty to be used as a
71 decorator
72 null_interface - an empty interface that is set to
73 GraphInterface that needs to be overwritten
74
75 Classes:
76 DictProperty - Dictionary property class
77 GraphInterface - Class used to interact with stored data.
78
79 Example usage:
80 See above
81 '''
82
83 from errors import ObjectError, MissingRequiredElement,
84 UnwrapError
85 from graph import Graph
86 from pythoscape.auxiliary.main_tools import
87 return_internal_attr
88 from dict_property import dict_property
89
90 #These are the interfaces which should be created. This
91 list should reflect the base objects which can be stored.
92 INTERFACES = ['Node', 'Edge', 'Attribute', 'Network']
93
94 @dict_property
95 def null_interface():
96     '''This is the null interface used as a place holder
97     for a real interface.
98
99     Returns:
100         locals to define dict_property
101         '''
102
103     def fset(self, key, value):
104         '''Set object in interface
105
106     Arguments:
107         key -- object
108         value -- assumed True
109         '''
```

./src/pythoscape/main/graph_interface.py

```
102         pass
103
104     def fget(self, key):
105         '''Select objects from interface.
106
107         Arguments:
108             key -- selection object
109
110         Yields:
111             selected objects
112         '''
113
114         yield None
115
116     def fdel(self, key):
117         '''Select objects from interface.
118
119         Arguments:
120             key -- selection object
121
122         Yields:
123             selected objects
124         '''
125         pass
126
127     return locals()
128
129 class GraphInterface(Graph):
130     '''GraphInterface class. For a full description of
131     the class, see module documentation.
132     In brief, this class extends the graph class into a
133     common interface for network storage.
134     Some methods needs to be overwritten for this class
135     to be functional:
136
137     node_interface, edge_interface, network_interface, attribute_
138     interface, finish, clear
139
140     Public methods:
141         node_interface -- dict_property interface for
142         nodes (null interface in this case)
143         edge_interface -- dict_property interface for
144         edges (null interface in this case)
145         network_interface -- dict_property interface for
146         networks (null interface in this case)
```

`./src/pythoscape/main/graph_interface.py`

```
139         attribute_interface -- dict_property interface
for attributes (null interface in this case)
140         push -- add object to data source
141         pull -- select objects from data source
142         delete -- delete objects from data source
143         finish -- perform this action after every data
source modification (pass in this case)
144         clear -- delete all data from data source (pass
in this case)
145
146         Example Usage:
147
148         Using a configured graph_interface
149
-----
---
150         Once graph_interface has been configured for a
specific data source, the following operations can
151         manipulate the data in the source.
152
153         To add data into the graph:
154         graph_interface.push(<Pythoscape object>)
155
156         To select data from the graph:
157         data_gen = graph_interface.pull(<Pythoscape
selection object>)
158         for obj in data_gen:
159             print obj
160
161         To delete data from the graph:
162         graph_interface.delete(<Pythoscape selection object>)
163
164         A Pythoscape selection object is an object where
defined object attributes represent searching arguments
and
165         empty object attributes represent wildcards. For
example, the node selection object Node(id='12345')
would search for a
166         node with id of '12345'. The node selection object
Node() would search for any node with any id.
167
168         Push and delete do not return anything. Select
returns a generator of all the selected objects.
169
170         To delete all data in the interface:
```

./src/pythoscape/main/graph_interface.py

```
171     graph_interface.clear()
172
173     Configuring a graph_interface
174
175     -----
176     --
177     The graph_interface has four null object interfaces
178     that must be overwritten to interact with the
179     data source.
180
181     Those are graph_interface.node_interface,
182     graph_interface.edge_interface,
183     graph_interface.network_interface
184     and graph_interface.attribute_interface. Each of
185     these accepts objects based a "dict_property" system.
186     The function dict_property should be imported from
187     this module and used as below.
188
189     @dict_property
190     def example_interface():
191         """ """
192         def fset(self, key, value):
193             pass
194         def fget(self, key):
195             yield None
196         def fdel(self, key):
197             pass
198         return locals()
199     '''
200
201     def __init__(self, warnings=False):
202         '''Create a new instance of GraphInterface.
203
204         Arguments (with defaults):
205         warnings (default is False) -- If True,
206         prints warnings as well as errors.'''
207         Graph.__init__(self)
208         self.warnings = warnings
209         for element in INTERFACES:
210             if not hasattr(self, '_' + element.lower() +
211                             'interface'):
212                 setattr(self, '_' + element.lower() +
213                         'interface', null_interface)
214
215     def _fix_request(self, obj, obj_name=None):
```

./src/pythoscape/main/graph_interface.py

```
206         '''Private function that makes sure an object is
properly configured.
207
208         Do not call as this function is subject to
removal or change!'''
209         if obj_name == None:
210             obj_name = obj.get_name()
211         if obj_name not in INTERFACES:
212             raise ObjectError(obj_name)
213         obj = getattr(self,obj_name)(obj)
214         return obj
215
216     def push(self,obj,obj_name=None):
217         '''Adds object to data repository.
218
219         Arguments:
220             obj -- obj to be added to repository
221
222         Arguments (with defaults):
223             obj_name (default is None) - if set, this
overwrites the obj_name used to something other than the
224             obj_name of the
obj
225         '''
226         obj = self._fix_request(obj,obj_name)
227         obj_name = obj.get_name()
228         if self.warnings:
229             disallowed = obj.push_disallowed()
230             if disallowed: print ''.join([disallowed, '
in ',obj_name,' is/are not read as an input. It/They
will not be stored. This is a warning.'])
231         push_test = obj.push_required()
232         if push_test:
233             raise MissingRequiredElement(push_test,
obj_name)
234         getattr(self,'_'.join([obj_name.lower(),
'interface']))[obj] = True
235
236     def pull(self,obj,obj_name=None):
237         '''Selects objects from data repository.
238
239         Arguments:
240             obj -- selection/deletion formatted
Pythoscape object
241
```


./src/pythoscape/main/graph_interface.py

```
242         Arguments (with defaults):
243             obj_name (default is None) - if set, this
overwrites the obj_name used to something other than the
244                                     obj_name of the
obj
245
246         Yields:
247             Pythoscape objects that match the inputted
object
248         '''
249         obj = self._fix_request(obj,obj_name)
250         obj_name = obj.get_name()
251         complex_atts = []
252         for el_attr in obj.COMPLEX:
253             complex_atts.append(return_internal_attr(obj,
el_attr))
254         if 'element_obj' in obj.ELEMENTS:
255             for complex_attr in obj.element_obj.COMPLEX:
256                 complex_atts.append(return_internal_attr(
obj,'element_obj',complex_attr))
257         for complex_attr in complex_atts:
258             if complex_attr and len(complex_attr) > 1:
259                 raise UnwrapError(complex_attr.get_name(),
obj_name)
260         for return_obj in getattr(self,'_'.join([obj_name
.lower(),'interface']))[obj]:
261             yield getattr(self,obj_name)(return_obj)
262
263     def delete(self,obj,obj_name=None):
264         '''Deletes objects from data repository.
265
266         Arguments:
267             obj -- selection/deletion formatted
Pythoscape object
268
269         Arguments (with defaults):
270             obj_name (default is None) - if set, this
overwrites the obj_name used to something other than the
271                                     obj_name of the
obj
272         '''
273         obj = self._fix_request(obj,obj_name)
274         obj_name = obj.get_name()
275         del getattr(self,'_'.join([obj_name.lower(),
'interface']))[obj]
```

./src/pythoscape/main/graph_interface.py

```
276
277     def finish(self):
278         '''Run after any object is submitted to the
interface.
279         '''
280         pass
281
282     def clear(self):
283         '''Deletes all data in the interface.
284         '''
285         pass
```

./src/pythoscape/main/network_objects.py

```
1  # Author: Alan Barber
2  # Created on: December 3, 2011
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module defines the base NetworkObject
   class that is used as the base class
7  for all the objects used to define a network in
   Pythoscape.
8
9  There are four main network object classes:
10
11  1) Nodes are the fundamental units of the graph and
   refer to protein or protein characteristics.
12  2) Edges are a measure of similarity between two
   proteins (i.e. nodes).
13  3) Networks are subgraphs of nodes and edges that allow
   you to partition the graph into working sections.
14  4) Attributes are descriptors that can be attached to
   either nodes, edges or networks.
15
16  Each of these objects can have a variety of three types
   of arguments depending on what should be inputted:
17
18  1) Value arguments -- a value, such as a string, integer
   or float
19  2) Object arguments -- another Pythoscape object, like a
   Node
20  3) Complex arguments -- these are modified using the
   complex parameter interface as described below
21
22  These four main objects are represented in Pythoscape in
   the following way:
23
24  Object name | Value arguments | Object
   arguments (object) | Complex arguments |
25  -----
26  Node | id | identifiers,
   | | networks |
27  Edge | |
   source_node (Node), target_node (Node) |
   networks |
```

./src/pythoscape/main/network_objects.py

```
28         |                                     | filter
   (Filter), weight (Weight)         |
29 Network      | network_name, network_type
   |
   |                                     |
30 Attribute    | attribute_name, attribute_value |
   element_obj (Node, Edge or Network)
   |                                     |
31
32 There are three auxiliary objects which are used as
   specific modifiers of the main objects:
33
34 Object name | Value arguments                                     |
35 -----
36 Identifier  | identifier_type, identifier_value
   |
37 Weight      | weight_value, algorithm
   |
38 Filter      | filter_name, direction, filter_value
   |
39
40 There are two ways to instantiate a network object:
41
42 1) Use a dictionary of class arguments:
43 my_node = Node({id:'test_id'})
44
45 2) Define class arguments:
46 my_node = Node(id='test_id')
47
48 Object arguments are other Pythoscape objects. Example:
49
50 my_node_1 = Node(id='test_id_1')
51 my_node_2 = Node(id='test_id_2')
52 my_edge =
   Edge(source_node=my_node_1, target_node=my_node_2)
53
54 Complex arguments are manipulated using the complex
   parameter interface. A complex
55 parameter is treated like as a non-redundant set of the
   network objects. There are
56 four functions allowed: add, delete, crop and clear.
   They are accessed as methods
57 on the object which contains them with the form
   <function>_<complex parameter>.
58 Add and delete require sequences of objects, crop
```

./src/pythoscape/main/network_objects.py

requires a single object.

As an example:

```
my_network = Network(network_name='test_network')
my_node.add_networks([my_network])
my_node.crop_networks(my_network)
my_node.delete_networks([my_network])
my_node.clear_networks()
```

To gain access to all the defined object classes, import the NETWORK_DEFS dictionary, which has all 7 keyed by their names.

Functions:

- edge_enforce - enforce function for edges
- no_nulls - enforce function for no_nulls objects
- attribute_enforce - enforce function for attributes
- filter_enforce - enforce function for filters

Classes:

- ObjectDefinition - Used to define network objects
- NetworkMixer - Meta-class used to mix network objects
- NetworkObject - Base class for network objects (do not call directly)

Module attributes:

NETWORK_DEFS - Dictionary of all defined network object classes for use by other modules.

Example usage:

See above

'''

```
from errors import UnwrapError, ObjectError
from copy import copy
from itertools import chain
from operator import itemgetter
from pythoscape.auxiliary.main_tools import get_one_attr,
or_lambda
from pythoscape.auxiliary.external_tools import
add_first_att_to_class_function
from errors import MissingRequiredElement,
NodeEqualityError, AttributeError, OptionError
```

./src/pythoscape/main/network_objects.py

```
97
98 class ObjectDefinition(object):
99     '''ObjectDefinition class. This class is used to set
100     the various parameters
101     that are needed to define a network object.
102
103     Public methods:
104         make_class - this method calls NetworkMixer to
105         create the defined object
106
107     Example use:
108         Node_def =
109         ObjectDefinition(['id'],complex_elements=['Network','Ident
110         ifier'],compare_overwrite='id',
111                                     push_required=['id'])
112     '''
113     def __init__(self,simple_elements,complex_elements=
114     None,
115                 redirect=None,collapse=None,enforce=None,
116                 compare_overwrite=None,push_required=
117     None,
118                 push_disallowed=None,complex_key=None):
119
120         self.variables = {}
121         self.variables['ELEMENTS'] = simple_elements
122         self.variables['COMPARE_OVERWRITE'] =
123         compare_overwrite
124         self.variables['COMPLEX'] = complex_elements if
125         complex_elements else []
126         self.variables['REDIRECT'] = redirect if redirect
127         else {}
128         self.variables['COLLAPSE'] = collapse if collapse
129         else {}
130         self.variables['ENFORCE'] = enforce if enforce
131         else []
132         self.variables['PUSH_REQUIRED'] = push_required
133         if push_required else []
134         self.variables['PUSH_DISALLOWED'] =
135         push_disallowed if push_disallowed else []
136         self.variables['COMPLEX_KEY'] = complex_key
137         self.variables['ELEMENT_LEN'] = len(
138         simple_elements)
139
140     def make_class(self,obj_name):
141         return NetworkMixer(obj_name,(None,),self.
```

./src/pythoscape/main/network_objects.py

```
variables)

128
129 class NetworkMixer(type):
130     '''NetworkMixer class. This is a meta-class which is
used to create all
131     the network object classes based on the
NetworkObject class.
132
133     Public methods:
134         None
135
136     Example use:
137         new_class =
NetworkMixer('NewNetObj', (None, ), {'method_to_add', method_t
o_add})
138     '''
139
140     def __new__(cls, name, bases, dct):
141         dct['__name__'] = name
142         for index, el_attr in enumerate(dct['ELEMENTS']):
143             dct[el_attr] = property(itemgetter(index))
144         new_cls = type.__new__(cls, name, (NetworkObject,
tuple), dct)
145         for el_attr in dct['COMPLEX']:
146             setattr(new_cls, 'add_{0}s'.format(el_attr.
lower()),
147                     add_first_att_to_class_function(
new_cls._add_complex_atts, el_attr))
148             setattr(new_cls, 'delete_{0}s'.format(el_attr.
lower()),
149                     add_first_att_to_class_function(
new_cls._remove_complex_atts, el_attr))
150             setattr(new_cls, 'crop_{0}s'.format(el_attr.
lower()),
151                     add_first_att_to_class_function(
new_cls._crop_complex_atts, el_attr))
152             setattr(new_cls, 'clear_{0}s'.format(el_attr.
lower()),
153                     add_first_att_to_class_function(
new_cls._clear_complex_atts, el_attr))
154             setattr(new_cls, '{0}s'.format(el_attr.lower
()),
155                     add_first_att_to_class_function(
new_cls._complex_att_interface, el_attr))
156         return new_cls
```

./src/pythoscape/main/network_objects.py

```
157
158 class NetworkObject(tuple):
159     '''NetworkObject class. This is the foundational
160     class which all the network objects
161     are created from. The code is based on the concept
162     of the "super-tuple" or tuple
163     with named fields. Do not call directly, use
164     ObjectDefinition to define the network object
165     which can then be properly mixed.
166
167     Public methods
168     do_copy -- make a copy of the network object
169     get_name -- return the name of the super-tuple
170
171     class
172     unwrap -- return a copy of the network object
173     that is serializable,
174     a dictionary with the named fields shown
175     modify -- makes a new copy of the object with
176     fields changed
177
178     Example usage:
179     Do not call directly!
180     '''
181
182     ENFORCE = []
183     ELEMENTS = []
184     COMPLEX = {}
185     REDIRECT = {}
186     COLLAPSE = {}
187     ELEMENT_LEN = 0
188     COMPARE_OVERWRITE = None
189     PUSH_REQUIRED = []
190     PUSH_DISALLOWED = []
191     COMPLEX_KEY = None
192
193     __slots__ = ()
194
195     def __new__(cls, *args):
196         '''Overwrites __new__ to create a new instance
197         of class cls
198
199         Returns:
200         new cls
201         '''
202         cls.ELEMENT_LEN = len(cls.ELEMENTS)
```


./src/pythoscape/main/network_objects.py

```
195         if len(args) != cls.ELEMENT_LEN:
196             raise TypeError, '%s takes exactly %s
arguments (%s given)' % (
197                 cls.__name__,
198                 cls.ELEMENT_LEN,
199                 len(args))
200         prepped_cls = tuple.__new__(cls,args)
201         for el_attr in cls.COMPLEX:
202             setattr(prepped_cls,'_{0}'.format(el_attr.
lower()),{ })
203         for enf_func in cls.ENFORCE:
204             prepped_cls = enf_func(prepped_cls)
205         return prepped_cls
206
207     def __len__(self):
208         '''Overwrites __len__ to provide a count of all
objects and values including
209         objects in complex arguments
210
211         Returns:
212             integer (length value)
213         '''
214
215         return tuple.__len__(self) + len(self.COMPLEX)
216
217     def __repr__(self):
218         '''Overwrites __repr__ to print all objects and
values contained within an
219         object -- including sub-objects.
220
221         Returns:
222             string representation of object
223         '''
224
225         attr_strings = []
226         for el_attr,item in self:
227             if el_attr in self.COMPLEX:
228                 el_attr = '{0}s'.format(el_attr)
229                 attr_strings.append('=' .join([el_attr,repr(
item)]))
230         return '%s(%s)' %(self.__class__.__name__,', '.
join(attr_strings))
231
232     def __iter__(self):
233         '''Overwrites __iter__ to yield all objects and
```

./src/pythoscape/main/network_objects.py

```
values contained within an
234     object -- including sub-objects.
235
236     Yields:
237         all contained values and sub-objects
238     '''
239
240     for el_attr in self.ELEMENTS:
241         if el_attr in self.REDIRECT and getattr(self,
el_attr):
242             yield el_attr, getattr(self, el_attr)
243             elif getattr(self, el_attr) != None:
244                 yield el_attr, getattr(self, el_attr)
245         for el_attr in self.COMPLEX:
246             if getattr(self, '{0}s'.format(el_attr.lower
247             ())))():
248                 yield el_attr, getattr(self, '{0}s'.format(
el_attr.lower()))()
249
250     def __copy__(self):
251         '''Overwrites __copy__ to properly return a copy
of network object
252
253         Returns:
254             copy of self
255         '''
256
257         attrs = [None]*self.ELEMENT_LEN
258         for i, el_attr in enumerate(self.ELEMENTS):
259             if getattr(self, el_attr):
260                 attrs[i] = copy(getattr(self, el_attr))
261         elem_copy = NETWORK_DEFS[self.get_name()](*attrs)
262         for el_attr in self.COMPLEX:
263             new_attrs = [copy(attr) for attr in getattr(
self, '{0}s'.format(el_attr.lower()))()]
264             get_attr(elem_copy, 'add_{0}s'.format(el_attr.
lower()))(new_attrs)
265         return elem_copy
266
267     def __eq__(self, other):
268         '''Overwrites __eq__ to properly compare Network
Objects. Node comparisons are based on id.
269         If not a node, it just compares tuples.
270
271         Returns:
```

./src/pythoscape/main/network_objects.py

```
271         True/False
272     '''
273
274     try:
275         return self[self.COMPARE_OVERWRITE] == other[
other.COMPARE_OVERWRITE]
276     except (AttributeError, TypeError):
277         return tuple.__eq__(self, other)
278
279     def __lt__(self, other):
280         '''Overwrites __lt__ to properly compare Network
Objects. Node comparisons are based on id.
281         If not a node, it just compares tuples.
282
283         Returns:
284             True/False
285         '''
286
287         try:
288             return self[self.COMPARE_OVERWRITE] < other[
other.COMPARE_OVERWRITE]
289         except (AttributeError, TypeError):
290             return tuple.__lt__(self, other)
291
292     def __ne__(self, other):
293         '''Overwrites __ne__ to properly compare Network
Objects. Node comparisons are based on id.
294         If not a node, it just compares tuples.
295
296         Returns:
297             True/False
298         '''
299
300         try:
301             return self[self.COMPARE_OVERWRITE] != other[
other.COMPARE_OVERWRITE]
302         except (AttributeError, TypeError):
303             return tuple.__ne__(self, other)
304
305     def __gt__(self, other):
306         '''Overwrites __ge__ to properly compare Network
Objects. Node comparisons are based on id.
307         If not a node, it just compares tuples.
308
309         Returns:
```

./src/pythoscape/main/network_objects.py

```
310         True/False
311     '''
312
313     return not (self.__lt__(other) or self.__eq__(
other))
314
315     def __le__(self, other):
316         '''Overwrites __le__ to properly compare Network
Objects. Node comparisons are based on id.
If not a node, it just compares tuples.
317
318         Returns:
319             True/False
320         '''
321
322         return self.__eq__(other) or self.__lt__(other)
323
324     def __ge__(self, other):
325         '''Overwrites __ge__ to properly compare Network
Objects. Node comparisons are based on id.
If not a node, it just compares tuples.
326
327         Returns:
328             True/False
329         '''
330
331         return self.__eq__(other) or self.__gt__(other)
332
333     def get_name(self):
334         '''Returns name of object
335
336         Returns:
337             name of object
338         '''
339
340         return self.__class__.__name__
341
342     def do_copy(self):
343         '''Returns a copy of the object.
344
345         Returns:
346             copy of object
347         '''
348
349         return copy(self)
```

./src/pythoscape/main/network_objects.py

```
352
353     def _add_complex_atts(self, att_name, complex_atts):
354         '''Adds objects into the set of the complex
parameter. Private method - do not call.
'''
355
356
357         for complex_att in complex_atts:
358             try:
359                 comp_name = complex_att.get_name()
360             except AttributeError:
361                 raise ObjectError(complex_att)
362             else:
363                 if comp_name != att_name:
364                     raise ObjectError(comp_name)
365                 if not getattr(complex_att, complex_att.
COMPLEX_KEY) in getattr(self, '_{0}'.format(att_name.lower
()))):
366                     getattr(self, '_{0}'.format(att_name.lower
()))[getattr(complex_att, complex_att.COMPLEX_KEY)] = set
([])
367                     getattr(self, '_{0}'.format(att_name.lower
()))[getattr(complex_att, complex_att.COMPLEX_KEY)].add(
complex_att)
368
369     def _remove_complex_atts(self, att_name, complex_atts):
370         '''Deletes objects from the set of the complex
arguments. Private method -- do not call.
'''
371
372
373         for complex_att in complex_atts:
374             getattr(self, '_{0}'.format(att_name.lower
()))[getattr(complex_att, complex_att.COMPLEX_KEY)].remove
(complex_att)
375
376     def _crop_complex_atts(self, att_name, complex_att):
377         '''Crop objects in complex parameter to the
single inputted object if it previously existed in the
set.
'''
378         Private method -- do not call.
379
380
381         if getattr(complex_att, complex_att.COMPLEX_KEY)
in getattr(self, '_{0}'.format(att_name.lower())):
382             setattr(self, '_{0}'.format(att_name.lower
()), {getattr(complex_att, complex_att.COMPLEX_KEY): set([
```

./src/pythoscape/main/network_objects.py

```
complex_att]]))
383
384     def _clear_complex_atts(self, att_name):
385         '''Clear all objects in the complex parameter.
386         Private method -- do not call.
387         '''
388         getattr(self, '_{0}'.format(att_name.lower())).
clear()
389
390     def _complex_att_interface(self, att_name):
391         '''Output the current set of objects from the
392         complex parameter. Private method -- do not call.
393         '''
394         return [item for item in chain(*map(self.
_list_all, getattr(self, '_{0}'.format(att_name.lower())).
values()))]
395
396     def _list_all(self, x):
397         '''Returns x in list if COMPLEX and not in list
398         if not. Private method -- do not call.
399         '''
400         return x.__class__.__name__ in self.COMPLEX and [
x] or x
401
402     def _get_internal_attr(self, path):
403         '''Returns x in list if COMPLEX and not in list
404         if not. Private method -- do not call.
405         '''
406         return reduce(get_one_attr, [self]+list(path))
407
408     def _push_check(self, item):
409         '''Returns x in list if COMPLEX and not in list
410         if not. Private method -- do not call.
411         '''
412         if item[0] == 'path':
413             return self._get_internal_attr(item[1:])
414         elif item[0] == 'choice':
415             choices = [self._push_check(['path']+list(
choice))] for choice in item[1:]
416             return reduce(or_lambda, [False]+choices)
417         elif item[0] == 'not':
418             return get_one_attr(self, item[1]) != item[2]
419         elif item[0] == 'is':
```

./src/pythoscape/main/network_objects.py

```
417         return get_one_attr(self,item[1]) == item[2]
418     else:
419         return get_one_attr(self,item)
420
421     def modify(self,**k):
422         '''Returns a new object with modifications made
423         based on the arguments given.
424
425         Other arguments:
426         **k -- object definitions
427
428         Returns:
429         new network object with modified arguments.
430         '''
431         attrs = [None]*self.ELEMENT_LEN
432         for i,el_attr in enumerate(self.ELEMENTS):
433             if getattr(self,el_attr) and el_attr not in k:
434                 attrs[i] = copy(getattr(self,el_attr))
435             elif el_attr in k:
436                 attrs[i] = k[el_attr]
437         elem_copy = NETWORK_DEFS[self.get_name()](*attrs)
438         for el_attr in self.COMPLEX:
439             getattr(elem_copy,'add_{0}s'.format(el_attr.
440 lower()))(getattr(self,'{0}s'.format(el_attr.lower()))())
441         return elem_copy
442
443     def unwrap(self,full=False):
444         '''Returns a dictionary form of the object. This
445         is useful for serialization.
446         If full is false, there can only be a maximum of
447         one object in each
448         complex parameter.
449
450         Arguments (with defaults):
451         full (default is False) -- return complex
452         arguments as tuples and output all of them.
453
454         Returns:
455         dictionary form of object
456         '''
457
458         return_dict = {}
459         for el_attr in self.COMPLEX:
460             if not full and len(getattr(self,'{0}s'.
```

./src/pythoscape/main/network_objects.py

```
format(el_attr.lower())))) > 1:
457         raise UnwrapError(el_attr, len(getattr(
self, '{0}s'.format(el_attr.lower()))))
458     for el_attr, value in self:
459         if el_attr in self.COMPLEX:
460             if not full:
461                 for item_name, item in value[0]:
462                     return_dict[item_name] = item
463             else:
464                 return_dict[el_attr] = [tuple([
subitem[1] for subitem in item])
465                                         for item in
value]
466         elif el_attr in self.COLLAPSE:
467             return_dict[el_attr] = getattr(value, self
.COLLAPSE[el_attr])
468         elif hasattr(value, 'unwrap'):
469             return_dict.update(value.unwrap())
470         else:
471             return_dict[el_attr] = value
472     return return_dict
473
474     def push_required(self):
475         ''' Method does a check on the current state of
the object. If all object arguments are set that are
476         necessary for the object to be pushed, method
returns None. If not, method returns the name of the
missing
477         object argument.
478
479         Arguments:
480             None
481
482         Returns:
483             argument name if set as PUSH_REQUIRED and
not defined
484             None if all PUSH_REQUIRED arguments are
defined
485         '''
486         for item in self.PUSH_REQUIRED:
487             if not self._push_check(item):
488                 return item
489         return None
490
491     def push_disallowed(self):
```


./src/pythoscape/main/network_objects.py

```
492         ''' Method does a check on the current state of
493         the object. If object arguments are set that will not
494         be pushed as part of the object, it returns them
495         as a list.
496
497         Arguments:
498             None
499
500         Returns:
501             list of arguments that will not be pushed
502             because they are disallowed
503         '''
504         return [self._get_internal(disallow) and disallow
505                 [-1] for disallow in self.PUSH_DISALLOWED]
506
507 #Enforcement functions
508
509 def edge_enforce(e):
510     ''' Function is an object enforcement function used
511     to check that edges do not point from one node
512     to the same node. It also sets as default the
513     "lesser" node as the target node so that edges
514     are standardized.
515
516     Arguments:
517         e - edge for enforcement check
518
519     Returns:
520         edge that has been fixed
521     '''
522     if (e.source_node or e.target_node) and e.source_node
523     == e.target_node:
524         raise NodeEqualityError()
525     if e.source_node < e.target_node:
526         e = e.modify(source_node=e.target_node,
527                      target_node=e.source_node)
528     return e
529
530 def no_nulls(elem_obj):
531     ''' Function is an object enforcement function used
532     to check that none of the object
533     arguments are None.
534
535     Arguments:
536         elem_obj - network object
```

./src/pythoscape/main/network_objects.py

```
528
529         Returns:
530             network object that has been checked
531         '''
532         for el_attr in elem_obj.ELEMENTS:
533             if getattr(elem_obj, el_attr) == None:
534                 raise MissingRequiredElement(el_attr, elem_obj
.get_name())
535         return elem_obj
536
537     def attribute_enforce(a):
538         ''' Function is an object enforcement function used
to check that the attribute
539         has an object attached to it and that it is an
appropriate object, as well as
540         set all attributes values of networks as strings.
541
542         Arguments:
543             a - attribute
544
545         Returns:
546             attribute that has been checked and fixed
547         '''
548         if not getattr(a, 'element_obj'):
549             raise MissingRequiredElement('element_obj',
'attribute')
550         if a.element_obj.get_name() == 'Edge' or a.
element_obj.get_name() == 'Node':
551             pass
552         elif a.element_obj.get_name() == 'Network':
553             if not isinstance(a.attribute_value, str):
554                 a.modify(attribute_value=str(a.
attribute_value))
555         else:
556             raise AttributeError(a.element_obj.__class__.
__name__)
557         return a
558
559     def filter_enforce(f):
560         ''' Function is an object enforcement function used
to check that a filter
561         is using one of the appropriate directions, '>',
'>', '<=' or '>='.
562
563         Arguments:
```

./src/pythoscape/main/network_objects.py

```
564         f - filter
565
566         Returns:
567             checked filter
568     """
569     if f.direction != '<' and f.direction != '>' and f.
direction != '<=' and f.direction != '>=':
570         raise OptionError(f.direction, 'direction')
571     return f
572
573 #Global object definitions and created object classes
574
575 Node_def = ObjectDefinition(['id'], complex_elements=[
    'Network', 'Identifier'], compare_overwrite='id',
576                             push_required=['id'])
577 Edge_def = ObjectDefinition(['source_node', 'target_node',
    'weight', 'filter'],
578                             complex_elements=['Network'],
579                             redirect={'source_node':
    'Node',
580                                     'target_node':
    'Node'}},
581                             collapse={'source_node': 'id',
582                                     'target_node': 'id'},
583                             enforce=[edge_enforce],
584                             push_required=[('path',
    'source_node', 'id'), ('path', 'target_node', 'id')],
585                             push_disallowed=['filter', (
    'source_node', 'networks'),
586                                             (
    'source_node', 'identifiers'),
587                                             (
    'target_node', 'networks'),
588                                             (
    'target_node', 'identifiers'))])
589 Network_def = ObjectDefinition(['network_name',
    'network_type'],
590                                push_required=[
    'network_name'],
591                                complex_key='network_name')
592 Attribute_def = ObjectDefinition(['attribute_name',
    'element_obj', 'attribute_value'],
593                                   enforce=[
    attribute_enforce],
594                                   push_required=[('not',
```

./src/pythoscape/main/network_objects.py

```
'attribute_value',None), 'attribute_name',
595                                     ['choice'
,('element_obj','source_node','id'),
596                                     (
'element_obj','id'),
597                                     (
'element_obj','network_name')]],
598                                     push_disallowed=[(
'element_obj','identifiers'),
599                                     (
'element_obj','networks'),
600                                     (
'element_obj','source_node','networks'),
601                                     (
'element_obj','source_node','identifiers'),
602                                     (
'element_obj','target_node','networks'),
603                                     (
'element_obj','target_node','identifiers')]]
604 Identifier_def = ObjectDefinition(['identifier_type',
'identifier_value'],enforce=[no_nulls],
605                                     complex_key=
'identifier_type')
606 Filter_def = ObjectDefinition(['filter_name','direction',
'filter_value'],
607                                     enforce=[no_nulls,
filter_enforce])
608 Weight_def = ObjectDefinition(['weight_value','algorithm'
],enforce=[no_nulls])
609
610 NETWORK_DEFS = {'Node':Node_def.make_class('Node'),
611                 'Edge':Edge_def.make_class('Edge'),
612                 'Network':Network_def.make_class(
'Network'),
613                 'Attribute':Attribute_def.make_class(
'Attribute'),
614                 'Identifier':Identifier_def.make_class(
'Identifier'),
615                 'Filter':Filter_def.make_class('Filter'),
616                 'Weight':Weight_def.make_class('Weight')}
617
618 #Testing code and example usage
619 if __name__ == '__main__':
620     test_node_1 = NETWORK_DEFS['Node']('test_node_1')
621     print test_node_1
```

./src/pythoscape/main/network_objects.py

```
622     test_node_2 = NETWORK_DEFS['Node']('test_node_2')
623     print test_node_2
624     test_network_1 = NETWORK_DEFS['Network'](
'test_network',None)
625     print test_network_1
626     test_network_2 = NETWORK_DEFS['Network'](
'test_network_2',None)
627     test_node_1.add_networks([test_network_1,
test_network_2])
628     print test_node_1
629     test_node_1.crop_networks(test_network_1)
630     print test_node_1
631     print test_node_1._network
```

./src/pythoscape/main/plugin.py

```
1  # Author: Alan Barber
2  # Created on: May 3, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains the classes required
7  to make plugins that can properly be used by
8  the Pythoscape environments. These classes do not need
9  to be used directly to use Pythoscape, as the
10 base plugin class doesn't do anything. These classes are
11 meant to be inherited and reconfigured.
12
13 The class to use to create plugins is PythoscapePlugin.
14 Two sample plugins are below. This example code
15 can guide the creation of new plugins.
16
17 ExamplePlugin -- This plugin will demonstrate the basics
18 of setting up a plugin
19 ExmapleEdgePlugin -- This plugin will demonstrate the
20 use of batch functions. Batch functions
21 can be used to run blocks of functions with a single
22 variable. This is useful when you want to
23 run several functions all at a time and not have to
24 store data in RAM. For example, if you want
25 to modify all the edges, but are afraid that there
26 wouldn't be enough RAM to hold all the edge
27 information at a time.
28
29 #!Code example start!
30
31 from pythoscape.main.plugin import PythoscapePlugin
32
33 class ExamplePlugin(PythoscapePlugin):
34     def __init__(self):
35         #-- Plugin required code--
36
37         #Don't change this line of code.
38         PythoscapePlugin.__init__(self, locals())
39         #Is your plugin compatible with the base
40         Pythoscape environment? (True/False)
41         #This is generally true.
42         self.Pythoscape = True
```

`./src/pythoscape/main/plugin.py`

```
35         #Is your plugin compatible with the
PythoscapeNetwork environment? (True/False)
36         #This is often true, except in cases where the
plugin needs to be able to work with multiple
sub-networks.
37         self.PythoscapeNetwork = True
38         #Set a description of your plugin for logging.
39         self.description = 'Plugin description here'
40
41         #Plugin specific
42         #Add your own init variables and functions here!
43
44         #Setup the functions. Any function that you want
run on the data needs to be setup.
45         #The plugins are run in the order that they are
set up.
46         #The first variable for self.setup is either
'pull', 'push' or 'delete' depending on what
47         #will be yielded by the function.
48         #You can rename the function, the second
variable, to whatever you want. The names below are
49         #just suggestions.
50         self.setup('pull', self.get_data)
51         self.setup('push', self.add_data)
52         self.setup('delete', self.del_data)
53
54         def get_data(self):
55             #This is an example 'pull' function.
56             #The first step is to yield a selection network
object. If you are uncertain about what a selection
57             #object looks like, see the documentation for
graph_interface. For example, the empty node below
58             #will return all nodes in the data source.
59             yield self.Node()
60             #The selected objects are yielded in
self.outgoing.
61             for node in self.outgoing:
62                 pass
63
64         def add_data(self):
65             #This is an example 'push' function.
66             #Every object yielded in this function will be
added into the data source.
67             yield self.Node(id='example')
68
```

./src/pythoscape/main/plugin.py

```
69     def del_data(self):
70         #This is an example 'delete' function.
71         #To use the delete function, yield selection
network objects. If you are uncertain about what a
selection
72         #object looks like, see the documentation for
graph_interface. For example, the empty node below
73         #will delete all nodes in the data source.
74         yield self.Node()
75
76 class ExampleEdgePlugin(PythoscapePlugin):
77
78     def __init__(self):
79
80         #-- Plugin required code--
81
82         #Don't change this line of code.
83         PythoscapePlugin.__init__(self, locals())
84         #Is your plugin compatible with the base
Pythoscape environment? (True/False)
85         #This is generally true.
86         self.Pythoscape = True
87         #Is your plugin compatible with the
PythoscapeNetwork environment? (True/False)
88         #This is often true, except in cases where the
plugin needs to be able to work with multiple
sub-networks.
89         self.PythoscapeNetwork = True
90         #Set a description of your plugin for logging.
91         self.description = 'Modify edges'
92
93         #Plugin specific
94         #Add your own init variables and functions here!
95         self._nodes = []
96
97         #Setup the functions. Any function that you want
run on the data needs to be setup.
98         #The plugins are run in the order that they are
set up.
99         #The first variable for self.setup is either
'pull','push' or 'delete' depending on what
100         #will be yielded by the function.
101         #You can rename the function, the second
variable, to whatever you want. The names below are
102         #just suggestions.
```


./src/pythoscape/main/plugin.py

```
103         self.setup('pull',self.get_nodes)
104
105         #Setup batch functions. The first argument is a
106         #list of tuples. The tuples are
107         #the same variables that would be used for a
108         #regular setup. In the case of a batch
109         #run, these functions will require an input.
110         #Pythoscape will run them in the order
111         #that you specify in the list. The second
112         #argument is a string
113         #that is the name of the class attribute that
114         #contains the batch of objects
115         #that will be passed into each function in the
116         #order provided.
117
118     self.batch_setup([('pull',self.get_edges),('push',self.yie
119     ld_edges)], '_nodes')
120
121     def get_nodes(self):
122         #Get all nodes names and store them.
123         yield self.Node()
124         for node in self.outgoing:
125             self._nodes.append(node)
126
127     def get_edges(self,node):
128         #Get all edges that are from a single node.
129         yield self.Edge(source_node=node)
130         self._curr_edges = []
131         for edge in self.outgoing:
132             self._curr_edges.append(edge)
133
134     def yield_edges(self,node):
135         #Add edge attribute
136         for edge in self._curr_edges:
137             yield
138         self.Attribute(element_obj=edge,attribute_value=True,attri
139         bute_name='modify')
140
141     #!Code example end!
142
143     There is also a function included in this class called
144     hasher. This function outputs the hexdigest of a SHA-224
145     hash.
146
147     This can be used as a standard function to create node
148     ids from a specific attribute.
```

./src/pythoscape/main/plugin.py

```
135
136 Classes:
137 SingleCommand -- Wrapper class for single commands.
138 BatchCommand -- Wrapper class for batch commands.
139 PythoscapePlugin -- Class to inherit to create plugins
    for Pythoscape. See instructions above.
140
141 Functions:
142 hasher -- returns hexdigest of SHA-224 hash
143 '''
144
145 from itertools import product
146 import hashlib
147 from graph import Graph
148
149 #returns hexdigest of SHA-224 hash
150 hasher = lambda x: hashlib.sha224(x).hexdigest()
151
152 class SingleCommand(object):
153     '''Wrapper class for single commands
154
155     Public methods:
156         None
157     '''
158
159     def __init__(self, command, func):
160         '''Creates a new instance of SingleCommand
161
162         Parameters:
163             command -- Pythoscape command
164             func -- plugin function'''
165
166         self._func = func
167         self._command = command
168
169     def __iter__(self):
170         '''Creates __iter__ to yield all commands and
    run functions.
171
172         Yields:
173             command as sting, run function
174         '''
175
176         yield self._command, self._func()
```

./src/pythoscape/main/plugin.py

```
178
179 class BatchCommand(object):
180     '''Wrapper class for batch commands
181
182     Public methods:
183         None
184     '''
185
186     def __init__(self, funcs, func_iter):
187         '''Creates a new instance of BatchCommand
188
189         Parameters:
190             funcs -- list of tuples, each tuple should
191             look like a single command -- (command,function)
192             func_iter -- list of variables that are
193             needed for the batch functions'''
194
195         self._funcs = funcs
196         self._func_iter = func_iter
197
198     def __iter__(self):
199         '''Creates __iter__ to yield all commands and
200         run functions.
201
202         Yields:
203             command as sting, run function
204         '''
205
206         for var,cmd_tup in product(self._func_iter,self.
207         _funcs):
208             yield cmd_tup[0],cmd_tup[1](var)
209
210 class PythoscapePlugin(Graph):
211     '''Class to inherit to create plugins for
212     Pythoscape. For instructions in inheriting
213     and properly coding plugins see the module
214     documentation.
215
216     Public methods:
217         setup -- setup a single command function
218         batch_setup -- setup a batch command
219     '''
220
221     def __init__(self,plugin_arguments):
```

./src/pythoscape/main/plugin.py

```
217         '''Creates a new instance of PythoscapePlugin
218
219         Parameters:
220             plugin_arguments -- locals() from plugin,
used for plugin logging'''
221
222         Graph.__init__(self)
223         self.environ = None
224         self.description = None
225         self._commands = []
226         self.outgoing = None
227         self._batch_funcs = []
228         self._batch_iter = []
229         if 'self' in plugin_arguments: del
plugin_arguments['self']
230         self._arguments = plugin_arguments
231         self.Pythoscape = False
232         self.PythoscapeNetwork = False
233
234         def __iter__(self):
235             '''Creates __iter__ to yield all commands and
run functions.
236
237             Yields:
238                 command as sting, run function
239                 ...
240
241             for command_obj in self._commands:
242                 for command,func in command_obj:
243                     yield command,func
244             for funcs,one_func_iter in zip(self._batch_funcs,
self._batch_iter):
245                 for command,func in BatchCommand(funcs,
getattr(self,one_func_iter)):
246                     yield command,func
247
248         def setup(self,command,func):
249             '''Adds a single command function to the queue.
250
251             Parameters:
252                 command -- For command options, see the
module documentation.
253                 func -- Function to be run. This function
may not have any
254                     input parameters other than self.
```

./src/pythoscape/main/plugin.py

```
255         '''
256         self._commands.append(SingleCommand(command, func))
257
258     def batch_setup(self, funcs, func_iter):
259         '''Adds a batch of commands to the queue.
260
261         Parameters:
262             funcs -- Tuple of (command,function). For
263 command options, see the module
264             documentation. Each function must
265 have one input parameter other
266 than self.
267             func_iter -- String the indicates an object
268 attribute of the plugin
269 that contains an iterable of
270 variables. These variables are
271 fed to each function when they
272 are run.
273         '''
274         self._batch_funcs.append(funcs)
275         self._batch_iter += [func_iter]
```

C.5.4. ./src/pythoscape/plugin/delete modules

./src/pythoscape/plugin/delete/delete_edges.py

```
1  # Author: Alan Barber
2  # Created on: October 8, 2011
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plugins used to delete
   edges.
7
8  Classes:
9  DeleteEdgesFromFile - use a file to delete edges
10 '''
11
12 from pythoscape.main.plugin import PythoscapePlugin
13
14 class DeleteEdgesFromFile(PythoscapePlugin):
15     '''Description: This plug-in deletes all edges
   defined in an input file.
16     By default, the plugin expects the the file lines to
   look as follows:
17         source_node target_node
18
19     There is an additional option to filter edge ids
   that are copied from
20     the Cytoscape edge table. If this is the case, the
   file lines look like this:
21         source_node (pp) target_node
22
23     Environments: This plug-in will work with both
   PythoscapeEnvironment and
24     PythoscapeNetwork. Using PythoscapeNetwork restricts
   the outputted edges
25     to the nodes in the specified network.
26
27     Example execution:
28     (Be sure to replace <interface>,<create
   interface>,<environment>,
29     and <create environment> below to the appropriate
   values. See the general
30     documentation for information on setting these lines
   appropriately)
31
32     #!Example begin!
33     import pythoscape.main.environments as env
34     import <interface>
```

./src/pythoscape/plugin/delete/delete_edges.py

```
35
36     #Import plug-ins
37     import pythoscape.plugin.delete.delete_edges as d_e
38
39     #Create interface and environment
40     <create interface>
41     <create environment>
42
43     #Create plug-ins
44     plugin =
d_e.DeleteEdgesFromFile(<file_name>,pp_edges=<True or
False>)
45
46     #Execute plug-ins
47     <environment>.execute_plugin(plugin)
48     #!Example end!
49
50     Required arguments:
51     <file_name> = name of file that contains the edges
to be deleted
52
53     Optional arguments:
54     <pp_edges> = default False, set to True if edge is
of form: source (pp) target
55
56     Plug-in example:
57     plugin = d_e.DeleteEdgesFromFile('my_edges.txt')
58
59     This example would delete the edges defined by
my_edges.txt.
60     '''
61
62     def __init__(self,file_name,pp_edges=False):
63         '''Create a new instance of DeleteEdgesFromFile
64
65         Required arguments:
66             file_name = name of file that contains the
edges to be deleted
67
68         Optional arguments:
69             pp_edges = set to True if edge is of form:
source (pp) target
70         '''
71         #Required
72         PythoscapePlugin.__init__(self,locals())
```


./src/pythoscape/plugin/delete/delete_edges.py

```
73         self.Pythoscape = True
74         self.PythoscapeNetwork = True
75         self.description = 'Delete edges from a file.'
76
77         #Plugin specific
78         self.file_name = file_name
79         self.pp_edges = pp_edges
80         self.setup('delete',self.delete_edges)
81
82     def delete_edges(self):
83         '''Plug-in method to delete edges.
84
85         Required arguments:
86             None
87
88         Yields:
89             edges to be deleted
90         '''
91         with open(self.file_name) as f:
92             for line in f:
93                 edge_name = line.strip().split()
94                 if self.pp_edges:
95                     edge_name.pop(1)
96                 yield self.Edge(source_name=edge_name[0],
target_name=edge_name[1])
```

C.5.5. ./src/pythoscape/plugin/input modules

./src/pythoscape/plugin/input/add_attribute_table.py

```
1  # Author: Alan Barber
2  # Created on: June 8, 2011
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plugins that use
   table-like files to add attributes to Pythoscape.
7
8  Classes:
9  ImportIdentifierTable - use a table-like file to add
   identifiers into Pythoscape
10 ImportAttributeTable - use a table-like file to add
   attributes into Pythoscape
11 '''
12
13 from pythoscape.main.plugin import PythoscapePlugin
14
15 class ImportIdentifierTable(PythoscapePlugin):
16     '''Description: This plug-in uses a table-like file
   to add identifiers
17     into Pythoscape. The layout of the file should be
   two columns, the first
18     column being the key to match to, either node id or
   node identifier, and
19     the second column being the new identifier to add.
20
21     id_or_identifier_to_match_to identifier_to_add
22
23     Environments: This plug-in will work with both
   PythoscapeEnvironment and
24     PythoscapeNetwork. Using PythoscapeNetwork restricts
   the inputted identifiers
25     to those nodes in a given network.
26
27     Example execution:
28     (Be sure to replace <interface>,<create
   interface>,<environment>,
29     and <create environment> below to the appropriate
   values. See the general
30     documentation for information on setting these lines
   appropriately)
31
32     #!Example begin!
33     import pythoscape.main.environments as env
```

./src/pythoscape/plugin/input/add_attribute_table.py

```
34     import <interface>
35
36     #Import plug-ins
37     import pythoscape.plugin.input.add_attribute_table
as a_a_t
38
39     #Create interface and environment
40     <create interface>
41     <create environment>
42
43     #Create plug-ins
44     plugin =
a_a_t.ImportIdentifierTable(<match_to>,<filename>,
45                             dont_skip =
<True or False>,
46                             match_against =
<match_against>)
47
48     #Execute plug-ins
49     <environment>.execute_plugin(plugin)
50     #!Example end!
51
52     Required arguments:
53     <match_to> = attribute name of identifier to add
54     <filename> = file name of identifier table
55
56     Optional arguments:
57     dont_skip = default False, set to True to add first
list from file
58             (the first line is assumed to be column
headings)
59     match_against = default None, set to the name of the
identifier name
60             if the first column has identifiers
and not node
61             ids
62
63     Plug-in example:
64     plugin =
a_a_t.ImportIdentifierTable('new_identifier','my_file.txt'
,match_against='existing_identifer')
65
66     This example would add 'new_identifier' identifiers
from the second column from my_file.txt
67     to Pythoscape. It would add them to the nodes that
```

./src/pythoscape/plugin/input/add_attribute_table.py

```
68     match with the 'existing_identifier'
69     identifier from the first column.
70     '''
71     def __init__(self, match_to, filename,
72                  dont_skip = False, match_against=None):
73         '''Create a new instance of ImportIdentifierTable
74
75         Required arguments:
76             match_to = attribute name of identifier to add
77             filename = file name of identifier table
78
79         Optional arguments:
80             dont_skip = default False, set to True to
81             add first list from file
82                         (the first line is assumed to be
83             column headings)
84             match_against = default None, set to the
85             name of the identifier name
86                         if the first column has
87             identifiers and not node
88                         ids
89         '''
90         #Required
91         PythoscapePlugin.__init__(self, locals())
92         self.Pythoscape = True
93         self.PythoscapeNetwork = True
94         self.description = 'Adds identifiers based on
95         another identifier'
96
97         #Plugin specific
98         self._ids = {}
99         self.m_a = match_against
100        self.m_t = match_to
101        self.filename = filename
102        self.dont_skip = dont_skip
103        self.setup('pull', self.get_ids)
104        self.setup('push', self.add_ids)
105
106        def get_ids(self):
107            '''Plug-in method to get keys to match to,
108            either identifier values or node ids
109
110            Required arguments:
111                None
```

./src/pythoscape/plugin/input/add_attribute_table.py

```
106
107         Yields:
108             self.Node()
109         '''
110         yield self.Node()
111         for node in self.outgoing:
112             if self.m_a:
113                 for identifier in node.identifiers():
114                     if identifier.identifier_type == self
115             .m_a:
116                 self._ids[identifier.
117 identifier_value] = node.id
118             else:
119                 self._ids[node.id] = node.id
120
121         def add_ids(self):
122             '''Plug-in method to add identifiers into
123 Pythoscape
124
125         Required arguments:
126             None
127
128         Yields:
129             nodes with new identifiers
130         '''
131         with open(self.filename) as f:
132             if not self.dont_skip:
133                 f.next()
134             for line in f:
135                 line = line.strip().split()
136                 if line[0] in self._ids:
137                     id_att = self.Identifier(
138 identifier_type=self.m_t, identifier_value=line[1])
139                     node = self.Node(id=self._ids[line[0
140 ]])
141                     node.add_identifiers([id_att])
142                     yield node
143
144     class ImportAttributeTable(PythoscapePlugin):
145         '''Description: This plug-in uses a table-like file
146 to add attributes
147         into Pythoscape. These attributes are added as
148 string attributes. The
149         layout of the file should be multiple columns, the
150 first
```

`./src/pythoscape/plugin/input/add_attribute_table.py`

```
143     column being the key to match to, either node id or
144     node identifier. The rest
145     of the columns should be attributes. The top row
146     should identify the
147     attributes. An example layout is below:
148
149     id_or_identifier_to_match_to attribute_to_add
150     attribute_to_add_2 ...
151
152     Environments: This plug-in will work with both
153     PythoscapeEnvironment and
154     PythoscapeNetwork. Using PythoscapeNetwork restricts
155     the inputted attributes
156     to those nodes in a given network.
157
158     Example execution:
159     (Be sure to replace <interface>,<create
160     interface>,<environment>,
161     and <create environment> below to the appropriate
162     values. See the general
163     documentation for information on setting these lines
164     appropriately)
165
166     #!Example begin!
167     import pythoscape.main.environments as env
168     import <interface>
169
170     #Import plug-ins
171     import pythoscape.plugin.input.add_attribute_table
172     as a_a_t
173
174     #Create interface and environment
175     <create interface>
176     <create environment>
177
178     #Create plug-ins
179     plugin = a_a_t.ImportAttributeTable(<filename>,
180                                         csv = <True or
181                                         False>,
182                                         match_against =
183                                         <match_against>)
184
185     #Execute plug-ins
186     <environment>.execute_plugin(plugin)
187     #!Example end!
```

./src/pythoscape/plugin/input/add_attribute_table.py

```
177
178     Required arguments:
179     <filename> = file name of identifier table
180
181     Optional arguments:
182     csv - default False, if True the plugin interprets
the file
183         as a csv
184     match_against = default None, set to the name of the
identifier name
185         if the first column has identifiers
and not node
186         ids
187
188     Plug-in example:
189     plugin =
a_a_t.ImportAttributeTable('my_file.txt',match_against='ex
isting_idenfifer')
190
191     This example would add attributes from my_file.txt
to Pythoscape. It would add them
192     to the nodes that match with the
'existing_identifier' identifier from the first column.
193     '''
194
195     def __init__(self,filename,
196                  csv = False,match_against=None):
197         '''Create a new instance of ImportAttributeTable
198
199         Required arguments:
200         filename = file name of identifier table
201
202         Optional arguments:
203         csv - default False, if True the plugin
interprets the file
204             as a csv
205         match_against = default None, set to the
name of the identifier name
206             if the first column has
identifiers and not node
207             ids
208         '''
209         #Required
210         PythoscapePlugin.__init__(self,locals())
211         self.Pythoscape = True
```


./src/pythoscape/plugin/input/add_attribute_table.py

```
212         self.PythoscapeNetwork = True
213         self.description = 'Adds attributes using table'
214
215         #Plugin specific
216         self._ids = {}
217         self.m_a = match_against
218         self.filename = filename
219         self.csv = csv
220         self.setup('pull',self.get_ids)
221         self.setup('push',self.add_atts)
222
223     def get_ids(self):
224         '''Plug-in method to get keys to match to,
either identifier values or node ids
225
226         Required arguments:
227             None
228
229         Yields:
230             self.Node()
231         '''
232         yield self.Node()
233         for node in self.outgoing:
234             if self.m_a:
235                 for identifier in node.identifiers():
236                     if identifier.identifier_type == self
.m_a:
237                         self._ids[identifier.
identifier_value] = node.id
238             else:
239                 self._ids[node.id] = node.id
240
241     def add_atts(self):
242         '''Plug-in method to add attributes into
Pythoscape
243
244         Required arguments:
245             None
246
247         Yields:
248             new attributes
249         '''
250         with open(self.filename) as f:
251             first_line=f.next()
252             if self.csv:
```

./src/pythoscape/plugin/input/add_attribute_table.py

```
253         first_line = first_line.strip().split(',')
254     else:
255         first_line = first_line.strip().split()
256     first_line = first_line[1:]
257     for line in f:
258         if self.csv:
259             line = line.strip().split(',')
260         else:
261             line = line.strip().split()
262         if line[0] in self._ids:
263             for entry, entry_name in zip(line[1:],
first_line):
264                 node = self.Node(id=self._ids[
line[0]])
265                 yield self.Attribute(element_obj=
node,
266 attribute_name=entry_name,
267 attribute_value=entry)
```

./src/pythoscape/plugin/input/add_blast_files.py

```
1  # Author: Alan Barber
2  # Created on: November 22, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plugins that input
   nodes from BLAST table files.
7
8  Functions:
9  get_data - parses BLAST data from table runs
10
11  Classes:
12  AddBLASTEdgesFromTableRun - this plug-in adds edges from
   a table-like file that of BLAST-like edges
13  '''
14
15  from pythoscape.main.plugin import PythoscapePlugin
16  from pythoscape.auxiliary.file_tools import find_files
17  import math
18
19  def get_data(line_data,outs,float_atts,score_atts):
20      '''Parses a BLAST table line into a value dictionary
   using a pre-defined list of attributes
21
22          Required arguments:
23          line_data - list of data from line to be
   interpreted
24          outs - key of names for each item in line_data
25          float_atts - which of the items in line_data
   should be turned to floats
26          score_atts - how to calculate scores using
   'bit_score' and 'm*n',
27                      dict with score name as key and
   lambda function as value
28
29          Returns:
30          dictionary of all attributes and scores
31      '''
32      return_data = dict([(att_name,None) for att_name in
   outs+score_atts.keys()])
33      for i,att_value in enumerate(line_data):
34          if outs[i] == 'bit_score' and att_value == 'None'
   :
35              att_value = None
```

./src/pythoscape/plugin/input/add_blast_files.py

```
36         elif outs[i] in float_atts and att_value != None:
37             att_value = float(att_value)
38         elif att_value != None:
39             att_value = int(att_value)
40             return_data[outs[i]] = att_value
41         if return_data['m*n'] != None and return_data[
42             'bit_score'] != None:
43             for score_type in score_atts:
44                 return_data[score_type] = score_atts[
45                     score_type](return_data['m*n'], return_data['bit_score'])
46         return return_data
47
48 class AddBLASTEdgesFromTableRun(PythoscapePlugin):
49     '''Description: This plug-in uses a table-like file
50     to add BLAST
51     edges to Pythoscape. The table file is expected to
52     have the following
53     columns in the order listed below from a BLAST
54     sequence similarity comparison.
55     The module parallel_blast2seq.py in the remote
56     directory automatically
57     makes these table files.
58
59     query_node_id
60     subject_node_id
61     bit_score
62     m*n
63     alignment_len
64     alignment_identities
65     query_start
66     query_end
67     subject_start
68     subject_end
69
70     Environments: This plug-in will work with both
71     PythoscapeEnvironment and
72     PythoscapeNetwork. Using PythoscapeNetwork adds the
73     inputted edges
74     into a specific network.
75
76     Example execution:
77     (Be sure to replace <interface>,<create
78     interface>,<environment>,
79     and <create environment> below to the appropriate
80     values. See the general
```

./src/pythoscape/plugin/input/add_blast_files.py

```
71     documentation for information on setting these lines
    appropriately)
72
73     #!Example begin!
74     import pythoscape.main.environments as env
75     import <interface>
76
77     #Import plug-ins
78     import pythoscape.plugin.input.add_blast_files as
a_b_f
79
80     #Create interface and environment
81     <create interface>
82     <create environment>
83
84     #Create plug-ins
85     plugin =
a_b_f.AddBLASTEdgesFromTableRun(<directory>,<include_atts>
    ,
86
87     file_ext=<file_ext>,
88
89     filter_att=<filter_att>,
90
91     filter_dir=<'>' or '<'>,
92
93     filter_value=<filter_value>,
94
95     restart=<True or False>)
96
97     #Execute plug-ins
98     <environment>.execute_plugin(plugin)
99     #!Example end!
100
101     Required arguments:
102
103     <directory> - directory location on disk to find all
    results files
104     <include_atts> - list of edge attributes to include
105
106     The include_atts argument requires a list of any of
    the following:
107
108     'bit_score', 'm*n', 'alignment_len', 'alignment_identities', '
    query_start', 'query_end', 'subject_start',
```

./src/pythoscape/plugin/input/add_blast_files.py

```
103     'subject_end', '-ln(E)', '-log10(E)'
104
105     The attributes included in this list will be added
to the network while the others will be skipped.
106
107
108     Optional arguments:
109
110     <file_ext> - default 'out', if the files are not
named with the extension '.out',
111         then this needs to be set to the
appropriate extension
112
113     If you want to filter incoming edges to speed up the
process by not importing and indexing unnecessary
114     edges, use the filter options. All three must be set
for the filter to work.
115
116     <filter_att> - default None, edge attribute to
filter on
117     <filter_dir> - default None, either '>' or '<'
118     <filter_value> - default None, value to test filter
against
119
120     If the run crashed for any reason and you want to
restart it in the middle, use the restart argument.
121     This option does not work when you are using the
above filter. You cannot restart currently if you are
122     using the filtering option.
123
124     <restart> - default False, True if you want to
restart the edge import run
125
126     Plug-in example:
127     plugin =
a_b_f.AddBLASTEdgesFromTableRun('/home/myuser/mydata/table
s',include_atts=['-log10(E)'])
128
129     This example would look for results files in the
directory /home/myuser/mydata/tables and only adds the
130     -log10(E) score to the database. This saves time
until the edges can be filtered. Other edge attributes
131     be added later.
132     '''
133
```

./src/pythoscape/plugin/input/add_blast_files.py

```
134     #OUTS - list of table attributes in order of column
135     OUTS = ['bit_score', 'm*n', 'alignment_len',
'alignment_identities', 'query_start', 'query_end',
'subject_start', 'subject_end']
136
137     #FLOAT_ATTS - which attributes from OUTS should be
cast as floats
138     FLOAT_ATTS = set(['bit_score', 'alignment_identities'])
139
140     #SCORE_ATTS - dictionary of scores to calculate
using the data from OUTS; key is score name, value is
lambda function
141     SCORE_ATTS = {'-ln(E)': lambda mn, b_s: -math.log(mn) +
(b_s*math.log(2)),
'-log10(E)': lambda mn, b_s: -math.log10(
mn) + (b_s*math.log10(2))}
142
143
144     def __init__(self, directory, include_atts, file_ext=
None, filter_att=None, filter_dir=None, filter_value=None,
restart=False):
145         '''Create a new instance of
AddBLASTEEdgesFromTableRun
146
147         Required arguments:
148             <directory> - directory location on disk to
find all results files
149             <include_atts> - list of edge attributes to
include; must include one/more of the following:
150
'bit_score', 'm*n', 'alignment_len', 'alignment_identities', '
query_start',
151
'query_end', 'subject_start', 'subject_end', '-ln(E)', '-log10
(E)'
152
153         Optional arguments:
154             <file_ext> - default 'out', if the files are
not named with the extension '.out',
155             then this needs to be set to
the appropriate extension
156             <filter_att> - edge attribute to filter on
157             <filter_dir> - either '>' or '<'
158             <filter_value> - value to test filter against
159             <restart> - default False, True if you want
to restart the edge import run
```

./src/pythoscape/plugin/input/add_blast_files.py

```
160         '''
161         #Required
162         PythoscapePlugin.__init__(self, locals())
163         self.Pythoscape = True
164         self.PythoscapeNetwork = True
165         self.description = 'Add BLAST data from a
cluster run.'
166
167         #Plugin specific
168         self.directory = directory
169         self.include_atts = set(include_atts)
170         self.file_ext = file_ext if file_ext else 'out'
171         self.filter_att = filter_att
172         self.filter_value = filter_value
173         self.filter_dir = filter_dir
174         if self.filter_att in self.SCORE_ATTS and self.
filter_value != None:
175             self.filter_on = True
176         else:
177             self.filter_on = False
178         if self.filter_dir != '>' and self.filter_dir !=
'<':
179             self.filter_on = False
180         self.to_do = set([])
181         if restart:
182             self.setup('pull', self.edges_exist)
183             self.setup('push', self.get_edges)
184
185         def edges_exist(self):
186             '''Plug-in method to get keys to test if the
last edge in input files has been added
187             to Pythoscape
188
189             Required arguments:
190                 None
191
192             Yields:
193                 last edge of all table files in directory
indicated
194             '''
195             for out_f in find_files(self.file_ext, self.
directory):
196                 with open(out_f) as open_f:
197                     last_line = None
198                     for line in open_f:
```


./src/pythoscape/plugin/input/add_blast_files.py

```
199         if line != '\n': last_line = line
200         last_line = last_line.strip().split()
201         yield self.Edge(source_node=self.Node(id=
last_line[0]),
202                        target_node=self.Node(id=
last_line[1]))
203     try:
204         test = self.outgoing.next()
205     except StopIteration:
206         self.to_do.add(out_f)
207
208     def get_edges(self):
209         '''Plug-in method to input edges from table files
210
211         Required arguments:
212             None
213
214         Yields:
215             edges from files
216             edge attributes indicated
217         '''
218         if self.to_do: all_fs = self.to_do
219         else: all_fs = find_files(self.file_ext,self.
directory)
220         for out_f in all_fs:
221             with open(out_f) as open_f:
222                 open_f.next()
223                 for line in open_f:
224                     data = line.strip().split()
225                     all_data = get_data(data[2:],self.
OUTS,self.FLOAT_ATTS,self.SCORE_ATTS)
226                     add_edges = False
227                     if not self.filter_on:
228                         add_edges = True
229                     elif self.filter_att in self.
SCORE_ATTS and all_data[self.filter_att] != None:
230                         if self.filter_dir == '>' and
all_data[self.filter_att] > self.filter_value:
231                             add_edges = True
232                         if self.filter_dir == '<' and
all_data[self.filter_att] < self.filter_value:
233                             add_edges = True
234                     if add_edges:
235                         try:
236                             edge = self.Edge(source_node=
```

./src/pythoscape/plugin/input/add_blast_files.py

```
self.Node(id=data[0]),
237                                     target_node=
self.Node(id=data[1]))
238         except IndexError:
239             print 'In file {0}, line
"{1}" is not complete.'.format(out_f, line)
240         else:
241             yield edge
242             for att in self.include_atts:
243                 if att in self.OUTS or
att in self.SCORE_ATTS:
244                     yield self.Attribute(
element_obj=edge,
245
attribute_name=att,
246
attribute_value=all_data[att])
247                 if '% id' in self.
include_atts \
248                     and all_data['query_start']
and all_data['query_end']\
249                     and all_data['subject_start']
and all_data['subject_end']:
250                     min_len = min([all_data[
'query_end']-all_data['query_start']+1,
251                                     all_data[
'subject_end']-all_data['subject_start']+1])
252                     yield self.Attribute(
element_obj=edge,
253
attribute_name='% id',
254
attribute_value=all_data['alignment_identities']/float(
min_len))
```

./src/pythoscape/plugin/input/add_local_blast.py

```
1  # Author: Alan Barber
2  # Created on: April 7, 2012
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plugins that input
   nodes from local BLAST runs.
7
8  Functions:
9  get_data - parses BLAST data from table runs
10
11  Classes:
12  AddBLASTEgesFromLocalBLAST - this plug-in adds edges
   from a local BLAST run.
13  '''
14
15  from pythoscape.main.plugin import PythoscapePlugin
16  from pythoscape.auxiliary.blast_tools import run_bl2seq
17  import math
18  import itertools
19
20  def fix_data(data, included_atts, float_atts, score_atts):
21      '''Transforms data points into the appropriate types
   and adds E-value scores.
22
23          Required arguments:
24              data - data dictionary from run_bl2seq
25              included_atts -atts to fix
26              float_atts - which of the items in line_data
   should be turned to floats,
27                      all others will be cast as ints
28              score_atts - how to calculate scores using
   'bit_score' and 'm*n',
29                      dict with score name as key and
   lambda function as value
30
31          Returns:
32              transformed data points
33      '''
34
35      for d_p in data:
36          if d_p in float_atts:
37              data[d_p] = float(data[d_p])
38          elif data[d_p] != None and d_p in included_atts:
```

./src/pythoscape/plugin/input/add_local_blast.py

```
39         data[d_p] = int(data[d_p])
40     if data['m*n'] != None and data['bit_score'] != None:
41         for score_type in score_atts:
42             data[score_type] = score_atts[score_type](
data['m*n'],data['bit_score'])
43     return data
44
45 class AddBLASTEEdgesFromLocalBLAST(PythoscapePlugin):
46     '''Description: This plug-in uses a local BLAST run
to add BLAST
47     edges to Pythoscape. Warning: This plug-in currently
does not run
48     BLAST in parallel. That means that it could take a
very long time to
49     calculate a large network.
50
51     Environments: This plug-in will work with both
PythoscapeEnvironment and
52     PythoscapeNetwork. Using PythoscapeNetwork adds the
inputted edges
53     into a specific network.
54
55     Plug-in software requirements: This plug-in was
designed to work with
56     BLAST+ v.2.2.25 and above. You can download this
package from:
57
58     http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE\_TYPE=
BlastDocs&DOC\_TYPE=Download
59
60     Example execution:
61     (Be sure to replace <interface>,<create
interface>,<environment>,
62     and <create environment> below to the appropriate
values. See the general
63     documentation for information on setting these lines
appropriately)
64
65     #!Example begin!
66     import pythoscape.main.environments as env
67     import <interface>
68
69     #Import plug-ins
70     import pythoscape.plugin.input.add_local_blast as
```

./src/pythoscape/plugin/input/add_local_blast.py

```
a_l_b
71
72     #Create interface and environment
73     <create interface>
74     <create environment>
75
76     #Create plug-ins
77     plugin =
a_l_b.AddBLASTEdgesFromLocalBLAST(<blast_exe>,<include_atts>,
78
79     filter_att=<filter_att>,
80
81     filter_dir=<'>' or '<'>,
82
83     filter_value=<filter_value>,
84
85     sequence_att=<sequence_att>)
86
87     #Execute plug-ins
88     <environment>.execute_plugin(plugin)
89     #!Example end!
90
91     Required arguments:
92
93     <blast_exe> - location of the blast_p executable
94     <include_atts> - list of edge attributes to include
95
96     The include_atts argument requires a list of any of
97     the following:
98
99     'bit_score', 'm*n', 'alignment_len', 'alignment_identities', '
100     query_start', 'query_end', 'subject_start',
101     'subject_end', '-ln(E)', '-log10(E)'
102
103     The attributes included in this list will be added
104     to the network while the others will be skipped.
105
106     Optional arguments:
107
108     <filter_att> - default None, edge attribute to
109     filter on
110     <filter_dir> - default None, either '>' or '<'
111     <filter_value> - default None, value to test filter
112     against
```

./src/pythoscape/plugin/input/add_local_blast.py

```
103     <sequence_att> - default 'sequence', attribute name
of protein sequences
104
105     Plug-in example:
106     plugin =
a_b_f.AddBLASTEdgesFromLocalBLAST('/home/myuser/blast/blas
tp',include_atts=['-log10(E)'])
107
108     This example would look for the blast executable at
/home/myuser/blast/blastp and only adds the
109     -log10(E) score to the database.
110     '''
111
112     #OUTS - list of table attributes in order of column
113     OUTS = set(['bit_score','m*n','alignment_len',
'alignment_identities',
114                 'query_start','query_end','subject_start'
,'subject_end'])
115
116     #FLOAT_ATTTS - which attributes from OUTS should be
cast as floats
117     FLOAT_ATTTS = set(['bit_score','alignment_identities'])
118
119     #SCORE_ATTTS - dictionary of scores to calculate
using the data from OUTS; key is score name, value is
lambda function
120     SCORE_ATTTS = {'-ln(E)':lambda mn,b_s:-math.log(mn) +
(b_s*math.log(2)),
121                   '-log10(E)':lambda mn,b_s:-math.log10(
mn) + (b_s*math.log10(2))}
122
123     def __init__(self,blast_exe,include_atts,
124                 filter_att=None,filter_dir=None,
filter_value=None,sequence_att=None):
125         '''Create a new instance of
AddBLASTEdgesFromLocalBLAST
126
127         Required arguments:
128         <blast_exe> - location of the blast_p
executable
129         <include_atts> - list of edge attributes to
include; must include one/more of the following:
130
'bit_score','m*n','alignment_len','alignment_identities','
query_start',
```

./src/pythoscape/plugin/input/add_local_blast.py

```
131 'query_end', 'subject_start', 'subject_end', '-ln(E)', '-log10
    (E)'
132
133     Optional arguments:
134         <filter_att> - edge attribute to filter on
135         <filter_dir> - either '>' or '<'
136         <filter_value> - value to test filter against
137         <sequence_att> - default 'sequence',
attribute name of protein sequences
138     '''
139
140     #Required
141     PythoscapePlugin.__init__(self, locals())
142     self.Pythoscape = True
143     self.PythoscapeNetwork = True
144     self.description = 'Add BLAST data from a local
run.'
145
146     #Plugin specific
147     self.blast_exe = blast_exe
148     self.include_atts = set(include_atts)
149     self.filter_att = filter_att
150     self.filter_value = filter_value
151     self.filter_dir = filter_dir
152     if self.filter_att in self.SCORE_ATTS and self.
filter_value != None:
153         self.filter_on = True
154     else:
155         self.filter_on = False
156     if self.filter_dir != '>' or self.filter_dir !=
'<':
157         self.filter_on = False
158     self.sequence_att = sequence_att if sequence_att
else 'sequence'
159     self._sequences = {}
160     self.setup('pull', self.get_sequences)
161     self.setup('push', self.get_edges)
162
163     def get_sequences(self):
164         '''Pulls all nodes and the corresponding sequences
165
166         Required arguments:
167             None
168
```

./src/pythoscape/plugin/input/add_local_blast.py

```
169         Yields:
170             all node sequence attribute search
171         '''
172         yield self.Attribute(element_obj=self.Node(),
attribute_name=self.sequence_att)
173         for attribute in self.outgoing:
174             self._sequences[attribute.element_obj.id] =
attribute.attribute_value
175
176     def get_edges(self):
177         '''Plug-in method to input edges from table files
178
179         Required arguments:
180             None
181
182         Yields:
183             edges from blast run
184             edge attributes indicated
185         '''
186
187         for node_1,node_2 in itertools.combinations(self.
_sequences.keys(),2):
188             seqs = {node_1:self._sequences[node_1],node_2
:self._sequences[node_2]}
189             edge_data = run_bl2seq(seqs,self.blast_exe)
190             add_edges = False
191             if edge_data:
192                 edge_data = fix_data(edge_data,self.OUTS,
self.FLOAT_ATTS,self.SCORE_ATTS)
193                 if not self.filter_on:
194                     add_edges = True
195                 elif self.filter_att in self.SCORE_ATTS
and edge_data[self.filter_att] != None:
196                     if self.filter_dir == '>' and
edge_data[self.filter_att] > self.filter_value:
197                         add_edges = True
198                     if self.filter_dir == '<' and
edge_data[self.filter_att] < self.filter_value:
199                         add_edges = True
200                 if add_edges:
201                     edge = self.Edge(source_node=self.Node(id
=node_1),
202                                     target_node=self.Node(id
=node_2))
203                     yield edge
```


./src/pythoscape/plugin/input/add_local_blast.py

```
204         for att in self.include_atts:
205             if att in self.OUTS or att in self.
SCORE_ATTS:
206                 yield self.Attribute(element_obj=
edge,
207
attribute_name=att,
208
attribute_value=edge_data[att])
209             if '% id' in self.include_atts \
210                 and edge_data['query_start'] and
edge_data['query_end']\
211                 and edge_data['subject_start'] and
edge_data['subject_end']:
212                 min_len = min([edge_data[
'query_end']-edge_data['query_start']+1,
213                               edge_data[
'subject_end']-edge_data['subject_start']+1])
214                 yield self.Attribute(element_obj=
edge,
215
attribute_name='% id',
216
attribute_value=edge_data['alignment_identities']/float(
min_len))
```

./src/pythoscape/plugin/input/add_repnod_atts.py

```
1  # Author: Alan Barber
2  # Created on: March 1, 2011
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plugins that calculate
   representative node attributes
7  based on the constituent node attributes.
8
9  Classes:
10  AddAttributesByIfAny - This plug-in calculates
   representative node attributes by 'if-any' voting
11  AddAttributesByGreatestSum - This plug-in calculates
   representative node attributes by greatest sum
12                               voting
13  '''
14
15  from pythoscape.main.plugin import PythoscapePlugin
16
17  class AddAttributesByIfAny(PythoscapePlugin):
18      '''Description: This plug-in adds representative
   node attributes
19      based on 'if-any' voting of the represented nodes'
   attributes. This means
20      that it takes all the possibilities of the attribute
   in the group
21      of represented nodes and concatenates that list
   together for
22      the representative node attribute.
23
24      Environments: This plug-in will only work with
   PythoscapeEnvironment.
25      If you want to have this plug-in operate on a
   sub-network, use the
26      initial_net argument.
27
28      Example execution:
29      (Be sure to replace <interface>,<create
   interface>,<environment>,
30      and <create environment> below to the appropriate
   values. See the general
31      documentation for information on setting these lines
   appropriately)
32
```

./src/pythoscape/plugin/input/add_repnod_atts.py

```
33      #!Example begin!
34      import pythoscape.main.environments as env
35      import <interface>
36
37      #Import plug-ins
38      import pythoscape.plugin.input.add_repnod_atts as
a_r_a
39
40      #Create interface and environment
41      <create interface>
42      <create environment>
43
44      #Create plug-ins
45      plugin =
a_r_a.AddAttributesByIfAny(<network>,<repnod_att>,<calc_a
tt>,<new_att_name>,
46
initial_net=<initial_net>,is_iden=<True or False>,
47                                concat=<concat>)
48
49      #Execute plug-ins
50      <environment>.execute_plugin(plugin)
51      #!Example end!
52
53      Required arguments:
54
55      <network> - name of representative node network
56      <repnod_att> - name of attribute that indexes
representative nodes
57      <calc_att> - name of attribute to make the
representative attribute from
58      <new_att_name> - name of created representative node
attribute
59
60      Optional arguments:
61
62      <initial_net> - default None, if you set this to the
name of a network, it will only consider
63                        nodes from that network
64
65      Because you can't run this plug-in using
NetworkEnvironment, <initial_net> is used to constrain
66      the plug-in to a specific sub-network.
67
68      <is_iden> - default False, set this to true if want
```

./src/pythoscape/plugin/input/add_repnod_atts.py

```
to do this calculation on an identifier
    and calc_att is the name of an identifier
69
70
71     <concat> - default ',', set this to another value if
you want the plug-in to concatenate options
72         using something other than ','
73
74     Plug-in example:
75     plugin =
a_r_a.AddAttributesByIfAny('my_rep_network','my_repnod_in
dex','chosen_attribute',
76                             'representative
chosen_attribute')
77
78     This plug-in would calculate representative
attributes for my_rep_network using my_repnod_index
79     as the node index. The attribute 'representative
chosen_attribute' would be created by doing
80     'if-any' voting on the attribute 'chosen_attribute'.
81     '''
82
83     def __init__(self,network,repnod_att,calc_att,
new_att_name,initial_net=None,is_iden=False,
84                 concat=None):
85         '''Create a new instance of
AddBLASTEEdgesFromTableRun
86
87         Required arguments:
88         <network> - name of representative node
network
89         <repnod_att> - name of attribute that
indexes representative nodes
90         <calc_att> - name of attribute to make the
representative attribute from
91         <new_att_name> - name of created
representative node attribute
92
93         Optional arguments:
94         <initial_net> - default None, if you set
this to the name of a network, it will only consider
95             nodes from that network
96         <is_iden> - default False, set this to true
if want to do this calculation on an identifier
97             and calc_att is the name of an
identifier
```

./src/pythoscape/plugin/input/add_repnod_atts.py

```
98         <concat> - default ',', set this to another
value if you want the plug-in to concatenate options
99         using something other than ','
<directory> - directory location on disk to find all
100         results files
101     '''
102
103     #Required
104     PythoscapePlugin.__init__(self, locals())
105     self.Pythoscape = True
106     self.PythoscapeNetwork = False
107     self.description = 'Add new representative node
attribute with "if-any" voting'
108
109     #Plugin specific
110     self.repnod_n = self.Network(network_name=
network)
111     self.initial_n = self.Network(network_name=
initial_net) if initial_net else None
112     self.repnod_att = repnod_att
113     self.calc_att = calc_att
114     self.new_att_name = new_att_name
115     self.is_iden = is_iden
116     self.concat = concat if concat != None else ','
117     self.clusters = {}
118     self.nodes_in_clus = {}
119     self.repnodes = {}
120     self.atts = {}
121     self.setup('pull', self.pull_info)
122     self.setup('push', self.make_atts)
123
124     def pull_info(self):
125         '''Plug-in method to pull all of the
representative node index attributes,
126         target attributes and nodes.
127
128         Required arguments:
129             None
130
131         Yields:
132             Pulling objects for representative node
index attributes, representative network
133             nodes, target attributes and, if needed,
nodes in initial_net
134         '''
```

./src/pythoscape/plugin/input/add_repnod_atts.py

```
135
136         net_node = self.Node()
137         if self.initial_n:
138             net_node.add_networks([self.initial_n])
139         yield self.Attribute(element_obj=net_node,
attribute_name=self.repnod_att)
140         for att in self.outgoing:
141             self.clusters[att.element_obj.id] = att.
attribute_value
142             self.nodes_in_clus[att.attribute_value] =
self.nodes_in_clus.get(att.attribute_value, [])+ [att.
element_obj.id]
143         if self.is_iden:
144             yield net_node
145         for node in self.outgoing:
146             all_idens = []
147             for iden in node.identifiers():
148                 if iden.identifier_type == self.
calc_att:
149                 all_idens.append(iden.
identifier_value)
150             if all_idens:
151                 self.atts[node.id] = self.concat.join
(all_idens)
152             else:
153                 yield self.Attribute(element_obj=net_node,
attribute_name=self.calc_att)
154             for att in self.outgoing:
155                 self.atts[att.element_obj.id] = att.
attribute_value
156                 net_node = self.Node()
157                 net_node.add_networks([self.repnod_n])
158                 yield net_node
159                 for node in self.outgoing:
160                     if node.id in self.clusters:
161                         self.repnodes[self.clusters[node.id]] =
node.id
162
163         def make_atts(self):
164             '''Plug-in method to submit representative
attributes
165
166             Required arguments:
167                 None
168
```

./src/pythoscape/plugin/input/add_repnod_atts.py

```
169         Yields:
170             representative attributes
171         '''
172
173         for clus_num in self.nodes_in_clus:
174             add_atts = set([])
175             for node_id in self.nodes_in_clus[clus_num]:
176                 if node_id in self.atts:
177                     add_atts.add(str(self.atts[node_id]))
178             yield self.Attribute(element_obj=self.Node(id
= self.repnodes[clus_num]),
179                                 attribute_value=self.
concat.join(list(add_atts)),
180                                 attribute_name=self.
new_att_name)
181
182     class AddAttributesByGreatestSum(PythoscapePlugin):
183         '''Description: This plug-in adds representative
node attributes
184             based on 'greatest sum' voting of the represented
nodes' attributes. This means
185             that it takes all the possibilities of the attribute
in the group
186             of represented nodes and adds the one with the most
occurrences.
187             Currently the plug-in adds a which ever one comes
first is there
188             is a tie. Future versions of the plug-in will solve
that.
189
190             Environments: This plug-in will only work with
PythoscapeEnvironment.
191             If you want to have this plug-in operate on a
sub-network, use the
192             initial_net argument.
193
194             Example execution:
195             (Be sure to replace <interface>,<create
interface>,<environment>,
196             and <create environment> below to the appropriate
values. See the general
197             documentation for information on setting these lines
appropriately)
198
199             #!Example begin!
```

./src/pythoscape/plugin/input/add_repnod_atts.py

```
200     import pythoscape.main.environments as env
201     import <interface>
202
203     #Import plug-ins
204     import pythoscape.plugin.input.add_repnod_atts as
a_r_a
205
206     #Create interface and environment
207     <create interface>
208     <create environment>
209
210     #Create plug-ins
211     plugin =
a_r_a.AddAttributesByGreatestSum(<network>,<repnod_atts>,<
calc_atts>,<new_att_name>,
212
initial_net=<initial_net>)
213
214     #Execute plug-ins
215     <environment>.execute_plugin(plugin)
216     #!Example end!
217
218     Required arguments:
219
220     <network> - name of representative node network
221     <repnod_atts> - name of attribute that indexes
representative nodes
222     <calc_atts> - list of attributes from which you will
find the greatest common value --
223                 if you just want this done on one
attribute, make a list of one
224     <new_att_name> - name of created representative node
attribute
225
226     Optional arguments:
227
228     <initial_net> - default None, if you set this to the
name of a network, it will only consider
229                 nodes from that network
230
231     Because you can't run this plug-in using
NetworkEnvironment, <initial_net> is used to constrain
232     the plug-in to a specific sub-network.
233
234     Plug-in example:
```


./src/pythoscape/plugin/input/add_repnod_atts.py

```
235     plugin =
a_r_a.AddAttributesByIfAny('my_rep_network', 'my_repnod_in
dex', ['chosen_attribute'],
236                                     'representative
chosen_attribute')
237
238     This plug-in would calculate representative
attributes for my_rep_network using my_repnod_index
239     as the node index. The attribute 'representative
chosen_attribute' would be created by doing
240     'greatest sum' voting on the attribute
'chosen_attribute'.
241     '''
242
243     def __init__(self, network, repnod_att, calc_atts,
new_att_name, initial_net=None):
244         '''Create a new instance of
AddAttributesByGreatestSum
245
246         Required arguments:
247         <network> - name of representative node
network
248         <repnod_att> - name of attribute that
indexes representative nodes
249         <calc_atts> - list of attributes from which
you will find the greatest common value --
250         if you just want this done on
one attribute, make a list of one
251         <new_att_name> - name of created
representative node attribute
252
253         Optional arguments:
254         <initial_net> - default None, if you set
this to the name of a network, it will only consider
255         nodes from that network
256         '''
257
258         #Required
259         PythoscapePlugin.__init__(self, locals())
260         self.Pythoscape = True
261         self.PythoscapeNetwork = False
262         self.description = 'Add new repnod attribute
with greatest score voting of possibilities given'
263
264         #Plugin specific
```

./src/pythoscape/plugin/input/add_repnod_atts.py

```
265         self.repnod_n = self.Network(network_name=
network)
266         self.initial_n = self.Network(network_name=
initial_net) if initial_net else None
267         self.repnod_att = repnod_att
268         self.calc_atts = calc_atts
269         self.new_att_name = new_att_name
270         self.clusters = {}
271         self.nodes_in_clus = {}
272         self.repnodes = {}
273         self_atts = {}
274         self.setup('pull',self.pull_info)
275         self.setup('push',self.make_atts)
276
277     def pull_info(self):
278         '''Plug-in method to pull all of the
representative node index attributes,
279         target attributes and nodes.
280
281         Required arguments:
282             None
283
284         Yields:
285             Pulling objects for representative node
index attributes, representative network
286             nodes, target attributes and, if needed,
nodes in initial_net
287         '''
288
289         net_node = self.Node()
290         if self.initial_n:
291             net_node.add_networks([self.initial_n])
292         yield self.Attribute(element_obj=net_node,
attribute_name=self.repnod_att)
293         for att in self.outgoing:
294             self.clusters[att.element_obj.id] = att.
attribute_value
295             self.nodes_in_clus[att.attribute_value] =
self.nodes_in_clus.get(att.attribute_value, [])+ [att.
element_obj.id]
296         for i,att_name in enumerate(self.calc_atts):
297             yield self.Attribute(element_obj=net_node,
attribute_name=att_name)
298         for att in self.outgoing:
299             if not self.clusters[att.element_obj.id]
```

./src/pythoscape/plugin/input/add_repnod_atts.py

```
in self.atts:
300         self.atts[self.clusters[att.
element_obj.id]] = [0]*len(self.calc_atts)
301         self.atts[self.clusters[att.element_obj.
id]][i] = self.atts[self.clusters[att.element_obj.id]][i]
\
302
        + att.attribute_value
303         net_node = self.Node()
304         net_node.add_networks([self.repnod_n])
305         yield net_node
306         for node in self.outgoing:
307             self.repnodes[self.clusters[node.id]] = node.
id
308
309         def make_atts(self):
310             '''Plug-in method to submit representative
attributes
311
312             Required arguments:
313                 None
314
315             Yields:
316                 representative attributes
317             '''
318
319             for cluster_num in self.atts:
320                 max_att_i = self.atts[cluster_num].index(max(
self.atts[cluster_num]))
321                 yield self.Attribute(element_obj=self.Node(id
=self.repnodes[cluster_num]),
322                                     attribute_value=self.
calc_atts[max_att_i],
323                                     attribute_name=self.
new_att_name)
324
325
326
```

./src/pythoscape/plugin/input/add_repnnode_edges.py

```
1  # Author: Alan Barber
2  # Created on: December 1, 2011
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plugins that
   recalculate edges for a representative node network.
7
8  Functions:
9  calculate_running_mean - calculates a mean over a set of
   numbers without storing the numbers as
10                          they are yielded
11
12  Classes:
13  AddEdgesToRepnnodeNetwork - Adds edges to a
   representative node network based on edges connecting
14                          represented nodes
15  '''
16  from pythoscape.main.plugin import PythoscapePlugin
17  import math
18
19  def calculate_running_mean(n,curr_mean,value,log10=False):
20      '''Calculates a running mean.
21
22          Required arguments:
23          n - current size of set
24          curr_mean - previous calculated mean
25          value - new value
26
27          Optional arguments:
28          log10 - default False, calculates the log10
   of the value before adding to the mean
29
30          Returns:
31          running mean as float
32      '''
33
34      if log10:
35          value = math.log10(value)
36      if curr_mean == 0:
37          return value/float(n)
38      else:
39          return ((curr_mean*(n-1))+value)/float(n)
40
```

./src/pythoscape/plugin/input/add_repnod_edges.py

```
41 class AddEdgesToRepnodNetwork(PythoscapePlugin):
42     '''Description: This plug-in adds edges to a
    representative node network.
43     It summarizes the information of all the edges that
    connect all nodes
44     between two representative nodes. It calculates
    three attributes:
45     1) how many edges are being represented, 2) the mean
    value of the attribute and
46     3) the minimum or maximum value of the attribute.
    The edge attributes added
47     have the form '<network> <attribute>' for their name
    where <network> is the name
48     of the representative network name and <attribute>
    is the specific attribute
49     added. For example, if you did this analysis on the
    representative node
50     network called 'my_rep_net', you would get three
    edge attributes
51     'my_rep_net mean', 'my_rep_net min/max' and
    'my_rep_net total'. Right now,
52     these names can't be customized.
53
54     Environments: This plug-in will only work with
    PythoscapeEnvironment.
55     If you want to have this plug-in operate on a
    sub-network, use the
56     initial_net argument.
57
58     Example execution:
59     (Be sure to replace <interface>,<create
    interface>,<environment>,
60     and <create environment> below to the appropriate
    values. See the general
61     documentation for information on setting these lines
    appropriately)
62
63     #!Example begin!
64     import pythoscape.main.environments as env
65     import <interface>
66
67     #Import plug-ins
68     import pythoscape.plugin.input.add_repnod_edges as
    a_r_e
69
```

./src/pythoscape/plugin/input/add_repnod_edges.py

```
70     #Create interface and environment
71     <create interface>
72     <create environment>
73
74     #Create plug-ins
75     plugin =
a_r_e.AddEdgesToRepnodNetwork(<network>,<repnod_att>,<ed
ge_att_name>,
76                                     minum=<True
or False>,<filt_dir>=<'<', '<=', '>' or '>='>,<
77     filt_value=<filt_value>,<initial_net>=<initial_net>,<
78     pre_filt=<pre_filt>)
79
80     #Execute plug-ins
81     <environment>.execute_plugin(plugin)
82     #!Example end!
83
84     Required arguments:
85
86     <network> - name of representative node network
87     <repnod_att> - name of attribute that indexes
representative nodes
88     <edge_att_name> - name of edge attribute to base
representative edge on
89
90     Optional arguments:
91
92     <minum> - default False, this argument is used to
indicate whether this plug-in calculate
93     a minimum or maximum value for edge attribute. False
indicates to calculate a maximum value,
94     True indicates a minimum value
95
96     If you want to filter what edges are included in
this calculation, set filter_dir and filter_value.
97     As a default, it will use edge_att_name as filter_att.
98
99     <filter_dir> - default None, either '>', '>=', '<='
or '<'
100     <filter_value> - default None, value to test filter
against
101
102     <initial_net> - default None, if you set this to the
```

`./src/pythoscape/plugin/input/add_repnod_edges.py`

```
103         name of a network, it will only consider
104             nodes from that network
105
106         Because you can't run this plug-in using
107         NetworkEnvironment, <initial_net> is used to constrain
108         the plug-in to a specific sub-network.
109
110         <pre_filt> - default None, if you want to filter on
111         something other than edge_att_name as filter_att
112             set the attribute name here.
113         <log_transform> - default False, if you want to
114         perform a log10 transform of the attribute
115             before calculating the mean.
116
117         Plug-in example:
118         plugin =
119         a_r_a.AddAttributesByIfAny('my_rep_network', 'my_repnod_in
120         dex', '-log10(E)',
121
122         filt_dir='>', filt_value=0)
123
124         This example would calculate representative edges
125         using -log10(E) as the basis of the calculations.
126         It will leave out any edges with -log10(E) < 0. The
127         representative node network used would be
128         my_rep_network which has the representative node
129         index of my_repnod_index.
130         '''
131
132         def __init__(self, network, repnod_att, edge_att_name,
133             minum=False, filt_dir=None, filt_value=
134             None, initial_net=None,
135             pre_filt=None, log_transform=False):
136             '''Create a new instance of
137             AddEdgesToRepnodNetwork
138
139             Required arguments:
140
141             <network> - name of representative node
142             network
143             <repnod_att> - name of attribute that
144             indexes representative nodes
145             <edge_att_name> - name of edge attribute to
146             base representative edge on
```

./src/pythoscape/plugin/input/add_repnnode_edges.py

```
133         Optional arguments:
134
135         <minum> - default False, Should this plug-in
136         calculate a minimum or maximum value for edge attribute?
137         False indicates to calculate a
138         maximum value, True indicates a minimum value
139         <filter_dir> - default None, either '>',
140         '>=', '<=' or '<'
141         <filter_value> - default None, value to test
142         filter against
143         <initial_net> - default None, if you set
144         this to the name of a network, it will only consider
145         nodes from that network
146         <pre_filt> - default None, if you want to
147         filter on something other than edge_att_name as filter_att
148         set the attribute name here.
149         <log_transform> - default False, if you want
150         to perform a log10 transform of the attribute
151         before calculating the mean.
152         '''
153
154         #Required
155         PythoscapePlugin.__init__(self, locals())
156         self.Pythoscape = True
157         self.PythoscapeNetwork = False
158         self.description = 'Add edges and edge info for
159         representative-edges into network.'
160
161         #Plugin specific
162         self.repnnode_n = self.Network(network_name =
163         network)
164         self.repnnode_att = repnode_att
165         self.filt_name = edge_att_name
166         if pre_filt == None:
167             pre_filt = edge_att_name
168         if pre_filt and filt_dir and filt_value != None:
169             self.filt = self.Filter(filter_name=pre_filt,
170             direction=filt_dir, filter_value=filt_value)
171         else: self.filt = None
172         self.minum = minum
173         self.initial_net = initial_net
174         self.repnodes = {}
175         self.clusters = {}
176         self.repedges = {}
177         self.ATT_NAMES = ['{0} {1}'.format(network,
```


./src/pythoscape/plugin/input/add_repnod_edges.py

```
168         self.log_transform = log_transform
169         self.setup('pull',self.pull_info)
170         self.setup('push',self.add_info)
171
172     def pull_info(self):
173         '''Plug-in method to pull all of the
174         representative node index attributes, nodes
175         edge attributes. It also figures out the mean,
176         min/max and number of the edge
177         attribute for each pair of representative nodes.
178
179         Required arguments:
180             None
181
182         Yields:
183             Pulling objects for representative node
184             index attributes, representative network
185             nodes, target edge attributes and, if
186             needed, nodes in initial_net
187         '''
188         att_node = self.Node()
189         if self.initial_net:
190             att_node.add_networks([self.Network(
191                 network_name=self.initial_net)])
192         yield self.Attribute(element_obj=att_node,
193                             attribute_name=self.repnod_att)
194         for att in self.outgoing:
195             self.clusters[att.element_obj.id] = att.
196             attribute_value
197             if len(self.clusters) == 0:
198                 raise ValueError("Attribute \"{0}\" probably
199                 does not exist!".format(self.repnod_att))
200             net_node = self.Node()
201             net_node.add_networks([self.repnod_n])
202             yield net_node
203             for node in self.outgoing:
204                 self.repnodes[self.clusters[node.id]] = node.
205                 id
206             yield self.Attribute(element_obj=self.Edge(filter
207                             =self.filt),attribute_name=self.filt_name)
208             for att in self.outgoing:
209                 if att.element_obj.source_node.id in self.
210                 clusters and \
211                 att.element_obj.target_node.id in self.
```

./src/pythoscape/plugin/input/add_repnod_edges.py

```
clusters:
201         c_source = self.clusters[att.element_obj.
source_node.id]
202         c_target = self.clusters[att.element_obj.
target_node.id]
203         if c_source != c_target:
204             if c_source > c_target:
205                 c_source,c_target = c_target,
c_source
206             if not (c_source,c_target) in self.
repedges:
207                 self.repedges[c_source,c_target]
= [0,0,0]
208                 self.repedges[c_source,c_target][1]
+= 1
209                 self.repedges[c_source,c_target][0] =
calculate_running_mean(self.repedges[c_source,c_target][
1],
210
self.repedges[c_source,c_target][0
],
211
att.attribute_value,
212
log10=self.log_transform)
213         if (self.minum and att.
attribute_value < self.repedges[c_source,c_target][2]) \
214             or (not self.minum and att.
attribute_value > self.repedges[c_source,c_target][2]):
215             self.repedges[c_source,c_target][
2] = att.attribute_value
216
def add_info(self):
217     '''Plug-in method to submit representative
attributes
218
219     Required arguments:
220         None
221
222     Yields:
223         representative edgea and edge attributes;
224         mean, min/max and number
225     '''
226     for m_edge in self.repedges:
227         source_node,target_node = self.Node(id = self
```

./src/pythoscape/plugin/input/add_repnod_edges.py

```
.repnodes[m_edge[0]]),self.Node(id = self.repnodes[m_edge
228         curr_edge = self.Edge(source_node =
source_node,target_node = target_node)
229         curr_edge.add_networks([self.repnod_n])
230         yield curr_edge
231         for i,att in enumerate(self.ATT_NAMES):
232             yield self.Attribute(element_obj=
curr_edge,
233                                     attribute_name = att,
234                                     attribute_value =
self.repedges[m_edge][i])
235
```

./src/pythoscape/plugin/input/calculate_struc_edges.py

```
1  # Author: Alan Barber
2  # Created on: September 1, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plug-ins that
   calculate protein structure similarity
7  as edges.
8
9  Functions:
10 parse_tmaligned_out - parse TM-Align output
11
12 Classes:
13 CalculateTMAlignScores - Uses TM-Align to calculate edge
   attributes between structure nodes.
14 '''
15
16 from pythoscape.main.plugin import PythoscapePlugin
17 from itertools import combinations
18 from pythoscape.auxiliary.parse_tools import give_lines
19 from pythoscape.auxiliary.text_tags import TextTags
20 import subprocess
21
22 def parse_tmaligned_out(tma_stream):
23     '''Parses a TM-Align output file using TextTags.
24
25     Required arguments:
26         tma_stream - direct character stream from
   stdout
27
28     Returns:
29         attribute list of (item,value) tuples
30     '''
31
32     line_gen = give_lines(tma_stream)
33     tt = TextTags(line_gen)
34     tt.set_position_tag('data line','Aligned length',
   go_for=1)
35     tt.set_position_tag(['TM-score 1','TM-score 2',
   'TM-score 3'],'TM-score',
36                        stop_tagging='(You',
   multiple_tags=True)
37     tt.set_position_tag('alignment','(":"',lines_to_data=
   2,go_for=1)
```

./src/pythoscape/plugin/input/calculate_struc_edges.py

```
38     tt.set_split_tag('TM-score 1',[(('TM-score (length 1)'
39 ,1)], ' '))
40     tt.set_split_tag('TM-score 2',[(('TM-score (length 2)'
41 ,1)], ' '))
42     tt.set_split_tag('TM-score 3',[(('TM-score (average
length)',1)], ' '))
43     tt.set_split_tag('data line',[(('Aligned length',2),('
44 'RMSD',4),('Seq_ID',6)], ' '))
45     tt.set_split_tag('alignment',[(('Aligned residues',
46 None)])
47
48     atts = []
49     for line in tt:
50         for item in line:
51             if line[item].endswith(','):
52                 line[item] = line[item][:-1]
53             if item == 'Aligned length':
54                 line[item] = int(line[item])
55             elif item.startswith('TM-score') or item ==
56 'RMSD' or item == 'Seq_ID':
57                 line[item] = float(line[item])
58             elif item == 'Aligned residues':
59                 line[item] = line[item].count(',')
60             atts.append((item, line[item]))
61     return atts
62
63 class CalculateTMAlignScores(PythoscapePlugin):
64     '''Description: This plug-in calculates edges
65 between nodes that represent
66 structures using the program TM-Align.
67
68     Environments: This plug-in will work with both
69 PythoscapeEnvironment and
70 PythoscapeNetwork. Using PythoscapeNetwork restricts
71 the inputted identifiers
72 to those nodes in a given network.
73
74     Example execution:
75     (Be sure to replace <interface>,<create
76 interface>,<environment>,
77 and <create environment> below to the appropriate
78 values. See the general
79 documentation for information on setting these lines
80 appropriately)
```

./src/pythoscape/plugin/input/calculate_struc_edges.py

```
71     #!Example begin!
72     import pythoscape.main.environments as env
73     import <interface>
74
75     #Import plug-ins
76     import
pythoscape.plugin.input.calculate_structure_edges as c_s_e
77
78     #Create interface and environment
79     <create interface>
80     <create environment>
81
82     #Create plug-ins
83     plugin =
c_s_e.CalculateTMAlignScores(<tm_align>,path_att=<path_att
>,
84
att_prefix=<att_prefix>)
85
86     #Execute plug-ins
87     <environment>.execute_plugin(plugin)
88     #!Example end!
89
90     Required arguments:
91     <tm_align> = TM-Align executable path
92
93     Optional arguments:
94     <path_att> = default 'path', the attribute that
holds the path to the structure file
95     <att_prefix> = default '', prefix to add to the
beginning to attribute names from parser
96
97     Plug-in example:
98     plugin = c_s_e.CalculateTMAlignScores('tmalign')
99
100     This example would calculate edges using the
TM-Align executable at path "tmalign".
101     '''
102
103     def __init__(self,tm_align,path_att=None,att_prefix=
None):
104         '''Create a new instance of CalculateTMAlignScores
105
106         Required arguments:
107         <tm_align> = TM-Align executable path
```

./src/pythoscape/plugin/input/calculate_struc_edges.py

```
108
109         Optional arguments:
110             <path_att> = default 'path', the attribute
that holds the path to the structure file
111             <att_prefix> = default '', prefix to add to
the beginning to attribute names from parser
112         '''
113
114         #Required
115         PythoscapePlugin.__init__(self, locals())
116         self.Pythoscape = True
117         self.PythoscapeNetwork = True
118         self.description = 'Add tm-align edges to
network.'
119
120         #Plugin specific
121         self.tm_align = tm_align
122         self.path_att = path_att if path_att else 'path'
123         self.att_prefix = att_prefix if att_prefix else ''
124         self._nodes = {}
125         self.setup('pull', self.get_nodes)
126         self.setup('push', self.add_edges)
127
128     def get_nodes(self):
129         '''Pulls all nodes with path atts
130
131         Required arguments:
132             None
133
134         Yields:
135             path att searching attribute
136         '''
137
138         yield self.Attribute(attribute_name=self.path_att
,element_obj=self.Node())
139         for node_att in self.outgoing:
140             self._nodes[node_att.element_obj.id] =
node_att.attribute_value
141
142     def add_edges(self):
143         '''Pulls all nodes with path atts
144
145         Required arguments:
146             None
147
```

./src/pythoscape/plugin/input/calculate_struc_edges.py

```
148         Yields:
149             Edges and TM-align edge attributes
150         """
151
152         for node_one,node_two in combinations(self._nodes
153 .keys(),2):
154             cline = '{0} {1} {2} -a'.format(self.tm_align
155 ,self._nodes[node_one],self._nodes[node_two])
156             child = subprocess.Popen(cline,shell=True,
157 stdout=subprocess.PIPE,close_fds=True)
158             my_out,my_err = child.communicate()
159             edge = self.Edge(source_node=self.Node(id=
160 node_one),
161                             target_node=self.Node(id=
162 node_two))
163             yield edge
164             for att in parse_tmalign_out(my_out):
165                 yield self.Attribute(element_obj=edge,
166 attribute_name=''.join([self.att_prefix,att[0]]),
167                             attribute_value=att[
168 1])
```


./src/pythoscape/plugin/input/import_sequences.py

```
1  # Author: Alan Barber
2  # Created on: August 31, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plugins inputs nodes
   from protein sequences.
7
8  Functions:
9  parse_fasta - parses a fasta file
10
11  Classes:
12  AddEdgesToRepnodNetwork - Adds edges to a
   representative node network based on edges connecting
13                               represented nodes
14  '''
15
16  import re
17  from pythoscape.main.plugin import PythoscapePlugin, hasher
18  from pythoscape.auxiliary.parse_tools import get_name
19
20  def parse_fasta(f_name):
21      '''Parses a fasta file into a generator of
   description line, sequence tuples.
22
23      Arguments:
24          f_name - name of fasta file
25
26      Yields:
27          tuple of description line, sequence for each
   sequence in fasta file
28      '''
29      with open(f_name) as f_open:
30          seq_growing = ''
31          seq_name = None
32          for line in f_open:
33              line = line.strip()
34              if line and line[0] == '>':
35                  if seq_name:
36                      yield seq_name, seq_growing
37                      seq_name = line[1:]
38                      seq_growing = ''
39                  elif line:
40                      seq_growing = '{0}{1}'.format(seq_growing
```

./src/pythoscape/plugin/input/import_sequences.py

```
,line)
41         yield seq_name, seq_growing
42
43     class ImportFromFastaFile(PythoscapePlugin):
44         '''Description: This plug-in takes sequences from a
45         fasta file and
46         adds them into a network as nodes. Hashes the
47         sequences with the
48         built in standard main.plugin.hasher to generate ids.
49
50         Environments: This plug-in will work with both
51         PythoscapeEnvironment and
52         PythoscapeNetwork. If you are storing multiple
53         distinct networks in a single
54         database/file, then use PythoscapeNetwork.
55
56         Example execution:
57         (Be sure to replace <interface>,<create
58         interface>,<environment>,
59         and <create environment> below to the appropriate
60         values. See the general
61         documentation for information on setting these lines
62         appropriately)
63
64         #!Example begin!
65         import pythoscape.main.environments as env
66         import <interface>
67
68         #Import plug-ins
69         import pythoscape.plugin.input.import_sequences as i_s
70
71         #Create interface and environment
72         <create interface>
73         <create environment>
74
75         #Create plug-ins
76         plugin =
77         i_s.ImportFromFastaFile(<sequence_file>,id_re=<id_re>,id_n
78         ame=<id_name>,
79
80         sequence_attr=<sequence_attr>,flat_id=<True or False>)
81
82         #Execute plug-ins
83         <environment>.execute_plugin(plugin)
84         #!Example end!
```

./src/pythoscape/plugin/input/import_sequences.py

```
75
76     Required arguments:
77
78     <sequence_file> - fasta file name
79
80     Optional arguments:
81
82     If both id_re and id_name are set, then a node
83     identifier is also added to the node when
84     it is added to the network. For example, if you
85     wanted to identify a sequence as having a
86     certain GI number, this function is used. Because
87     regular expressions are used for this
88     function and are not intuitive, several have already
89     been included in Pythoscape for
90     convenience. See the documentation in
91     auxiliary.re_patterns for more information. If
92     you set flat_id to True, then you only have to set
93     id_name because it uses the whole
94     fasta description line as the identifier value.
95
96     <id_re> - default None, regular expression that
97     locates the node identifier in the fasta title
98     <id_name> - default None, type of node identifier
99     recognized by the regular expression
100     <flat_id> - default False, deposit whole description
101     line as identifier value
102
103     <sequence_attr> - default 'sequence', name of
104     attribute that sequences are deposited under
105
106     Plug-in example:
107     plugin =
108     i_s.ImportFromFastaFile('my_seqs.fa',id_re=RE_PATTERNS['SFLD_GI'],id_name='GI' )
109
110     This example would pull sequences from my_seqs.fa
111     and use the SFLD_GI regular expression to
112     add GI identifiers to the nodes.
113     '''
114     def __init__(self,sequence_file,flat_id=False,id_re=
115     None,id_name=None,sequence_attr=None):
116         '''Create a new instance of ImportFromFastaFile
```

./src/pythoscape/plugin/input/import_sequences.py

```
106         Required arguments:
107
108         <sequence_file> - fasta file name
109
110         Optional arguments:
111
112         <id_re> - default None, regular expression
113         that locates the node identifier in the fasta title
114         <id_name> - default None, type of node
115         identifier recognized by the regular expression
116         <flat_id> - default False, deposit whole
117         description line as identifier value
118         <sequence_attr> - default 'sequence', name
119         of attribute that sequences are deposited under
120         '''
121
122         #Required
123         PythoscapePlugin.__init__(self, locals())
124         self.Pythoscape = True
125         self.PythoscapeNetwork = True
126         self.description = 'Import from fasta file'
127
128         #Plugin specific
129         self.sequence_file = sequence_file
130         self.id_re = id_re
131         self.id_name = id_name
132         self.flat_id = flat_id
133         self.sequence_attr = sequence_attr if
134         sequence_attr else 'sequence'
135         self.setup('push', self.get_sequence_info)
136
137         def get_sequence_info(self):
138             '''Plug-in method to push all sequence nodes
139             with identifiers and
140             sequence attributes.
141
142             Required arguments:
143             None
144
145             Yields:
146             nodes with identifier, if specified
147             sequence attributes
148             '''
149             re_pattern = None
150             if self.id_re and self.id_name: re_pattern = re.
```

./src/pythoscape/plugin/input/import_sequences.py

```
compile(self.id_re)
145         for seq_name, seq in parse_fasta(self.
sequence_file):
146             curr_node = self.Node(id=hasher(seq))
147             ident = None
148             if re_pattern:
149                 ident = self.Identifier(identifier_type=
self.id_name,
150                                     identifier_value=
get_name(seq_name.strip(), self.id_re))
151             elif self.flat_id and self.id_name:
152                 ident = self.Identifier(identifier_type=
self.id_name, identifier_value=seq_name.strip())
153             if ident != None:
154                 curr_node.add_identifiers([ident])
155             yield curr_node
156             node_attr = self.Attribute(element_obj=
curr_node, attribute_name=self.sequence_attr,
attribute_value=seq)
157             yield node_attr
```

./src/pythoscape/plugin/input/make_cdhit_repnodes.py

```
1  # Author: Alan Barber
2  # Created on: November 2, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plugins that make
   representative nodes using CD-HIT.
7
8  Functions:
9  index_shortened - takes a list of ids and shortens them
   to a given length as a dictionary
10                     shortened id is the key, full length
   id is the value
11
12  Classes:
13  CreateCDHITRepnodes - Makes a new representative node
   network using CD-HIT
14  '''
15
16  from pythoscape.main.plugin import PythoscapePlugin
17  import subprocess
18  import tempfile
19  import os
20
21  def index_shortened(ids, length):
22      '''Shortens a list of id and provides a mapping
   dictionary
23
24      Arguments:
25          ids - list of ids
26          lenght - length to shorten ids to
27
28      Returns:
29          dictionary of shortened ids, key is shortened
   id, value is original id
30      '''
31      return dict([(id[:length],id) for id in ids])
32
33  class CreateCDHITRepnodes(PythoscapePlugin):
34      '''Description: This plug-in creates a new
   representative node network where the
35          representative sequence (the longest sequence)
   chosen by CD-HIT stands in
36          for all nodes of the same CD-HIT cluster. This
```

`./src/pythoscape/plugin/input/make_cdhit_repnodes.py`

```
37     plug-in creates the network
38     if it doesn't already exist.
39
40     Plug-in software requirements: You must have
41     installed CD-HIT. This plug-in
42     was tested on CD-HIT version 4.5.6. You can download
43     CD-HIT from:
44     http://http://code.google.com/p/cdhit/
45
46     Environments: This plug-in will only work with
47     PythoscapeEnvironment.
48     If you want to have this plug-in operate on a
49     sub-network, use the
50     initial_net argument.
51
52     Example execution:
53     (Be sure to replace <interface>,<create
54     interface>,<environment>,
55     and <create environment> below to the appropriate
56     values. See the general
57     documentation for information on setting these lines
58     appropriately)
59
60     #!Example begin!
61     import pythoscape.main.environments as env
62     import <interface>
63
64     #Import plug-ins
65     import pythoscape.plugin.input.make_cdhit_repnodes
66     as m_c_r
67
68     #Create interface and environment
69     <create interface>
70     <create environment>
71
72     #Create plug-ins
73     plugin =
74     m_c_r.CreateCDHITRepnodes(<network_name>,<cdhit_exe>,n=<n>
75     ,c=<c>,sequence_att=<sequence_att>,
76
77     repnode_att=<repnode_att>,initial_net=<initial_net>,
78
79     network_type=<network_type>,include_lengths=<include_lengths>)
80
81     67
```

./src/pythoscape/plugin/input/make_cdhit_repnodes.py

```
68
69     #Execute plug-ins
70     <environment>.execute_plugin(plugin)
71     #!Example end!
72
73     Required arguments:
74
75     <network_name> - network to create that contains
representative nodes. Remember that this
76                     name has to be under 13 characters
for the MySQL interface! This network will
77                     be created if it doesn't already
exist.
78
79     <cdhit_exe> - path to CDHIT executable. It is the
executable file named 'cdhit' that comes
80                     with the program.
81
82     Optional arguments:
83
84     <n> - default is 2, word size for use in CDHIT. Set
n to a new value to change it use a different
85         word size.
86
87     <c> - default is 0.4, sequence identity cutoff for
CDHIT to use. Set c to a new value to increase
88         clustering stringency.
89
90     <sequence_att> - default is 'sequence', attribute
that Pythoscape will use to locate sequences.
91
92     <repline_att> - default is 'cdhit <c>' where <c> is
the sequence identity cutoff, attribute that
93                     Pythoscape will use to store the
cluster index number. For example, if you set the
94                     threshold to 0.5, the representative
node attribute name would be 'cdhit 0.5'. If
95                     you want to have a custom attribute
name, set this parameter.
96
97     <initial_net> - default None, if you set this to the
name of a network, it will only consider
98                     nodes from that network
99
100    <network_type> - default None, if you set this, it
```



```

./src/pythoscape/plugin/input/make_cdhit_repnodes.py

will include a network_type in the network
101         created
102
103     <include_lengths> - default None, if you input an
integer here, it will not include sequences
104         below a given length in this
analysis. This is useful if you do not want to
105         include sequence fragments in
your representative node network.
106
107     Because you can't run this plug-in using
NetworkEnvironment, <initial_net> is used to constrain
108     the plug-in to a specific sub-network.
109
110     Plug-in example:
111     plugin =
m_c_r.CreateCDHITRepnodes('my_rep_net', '/my/cdhit/director
y/cdhit')
112
113     This example create a representative node network
called my_rep_net using the CD-HIT executable
114     at /my/cdhit/directory/cdhit and make represnetative
nodes using a 40% sequence identity cutoff.
115     '''
116
117     def __init__(self, network_name, cdhit_exe,
118                   n=None, c=None, network_type=None,
sequence_att=None, repnode_att=None,
119                   initial_net=None, include_lengths=None):
120         '''Create a new instance of CreateCDHITRepnodes
121
122         Required arguments:
123
124         <network_name> - network to create that
contains representative nodes. Remember that this
125         name has to be under 13
characters for the MySQL interface! This network will
126         be created if it doesn't
already exist.
127
128         <cdhit_exe> - path to CDHIT executable. It
is the executable file named 'cdhit' that comes
129         with the program.
130
131         Optional arguments:

```

./src/pythoscape/plugin/input/make_cdhit_repnodes.py

```
132
133         <n> - default is 2, word size for use in
CDHIT. Set n to a new value to change it use a different
134         word size.
135
136         <c> - default is 0.4, sequence identity
cutoff for CDHIT to use. Set c to a new value to increase
137         clustering stringency.
138
139         <sequence_att> - default is 'sequence',
attribute that Pythoscape will use to locate sequences.
140
141         <repnode_att> - default is 'cdhit <c>' where
<c> is the sequence identity cutoff, attribute that
142         Pythoscape will use to store
the cluster index number. For example, if you set the
143         threshold to 0.5, the
representative node attribute name would be 'cdhit 0.5'.
If
144         you want to have a custom
attribute name, set this parameter to the desired name.
145
146         <initial_net> - default None, if you set
this to the name of a network, it will only consider
147         nodes from that network
148
149         <network_type> - default None, if you set
this, it will include a network_type in the network
150         created
151
152         <include_lengths> - default None, if you
input an integer here, it will not include sequences
153         below a given length in
this analysis. This is useful if you do not want to
154         include sequence
fragments in your representative node network.
155         '''
156         #Required
157         PythoscapePlugin.__init__(self, locals())
158         self.Pythoscape = True
159         self.PythoscapeNetwork = False
160         self.description = 'Make a new representative
node network using CDHIT.'
161
162         #Plugin specific
```

./src/pythoscape/plugin/input/make_cdhit_repnodes.py

```
163         self.cdhit_exe = cdhit_exe
164         self.n = n if n else 2
165         self.c = c if c else 0.40
166         self.sequence_att = sequence_att if sequence_att
else 'sequence'
167         self.repnode_att = repnode_att if repnode_att
else 'cdhit {0}'.format(self.c)
168         self.repnode_n = self.Network(network_name =
network_name, network_type = network_type)
169         self.initial_n = self.Network(network_name =
initial_net) if initial_net else None
170         self.inc_len = include_lengths
171         self.sequences = {}
172         self.indexes = None
173         self.setup('pull', self.pull_sequences)
174         self.setup('push', self.add_cdhit)
175
176     def pull_sequences(self):
177         '''Pulls all nodes and the corresponding
sequences to make the representative network
from.
178
179         Required arguments:
180             None
181
182         Yields:
183             all node search, sequence attribute search
184             ...
185
186         pull_node = self.Node()
187         if self.initial_n: pull_node.add_networks([self.
initial_n])
188         yield self.Attribute(attribute_name=self.
sequence_att, element_obj=pull_node)
189         to_index = []
190         for node_att in self.outgoing:
191             if self.inc_len != None and (len(node_att.
attribute_value) < self.inc_len[0]\
192             or len(node_att.attribute_value) > self.
inc_len[1]):
193                 pass
194             else:
195                 self.sequences[node_att.element_obj.id] =
node_att.attribute_value
196                 to_index.append(node_att.element_obj.id)
```

./src/pythoscape/plugin/input/make_cdhit_repnodes.py

```
198         self.indexes = index_shortened(to_index,19)
199
200     def add_cdhit(self):
201         '''Adds attributes to define representative nodes
202
203         Required arguments:
204             None
205
206         Yields:
207             node attribute of representative node index
208             representative nodes with representative
209             network name attached
210             '''
211         yield self.repnode_n
212         with tempfile.NamedTemporaryFile() as f:
213             for seq in self.sequences:
214                 f.write('>{0}\n{1}\n'.format(seq,self.
215 sequences[seq]))
216                 f.flush()
217                 cline = '{0} -i {1} -o cdhit.temp -c {2} -n
218 {3}'.format(self.cdhit_exe,
219
220                 f.name,
221
222                 self.c,
223
224                 self.n)
225                 child = subprocess.Popen(cline, shell=True,
226 close_fds=True)
227                 my_out,my_err = child.communicate()
228                 f = open('cdhit.temp.clstr')
229                 cluster_num = None
230                 while 1:
231                     try:
232                         line = f.next()
233                     except StopIteration:
234                         break
235                     else:
236                         if line[0] == '>':
237                             cluster_num = int(line.strip().split
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

./src/pythoscape/plugin/input/make_cdhit_repnodes.py

```
234             node_obj = self.Node(id = id)
235             yield self.Attribute(element_obj=
node_obj,
236                                     attribute_name=
self.repnod_att,
237                                     attribute_value=
cluster_num)
238             if line.strip().split()[-1] == '*':
239                 node_obj.add_networks([self.
repnod_n])
240             yield node_obj
241         f.close()
242         os.remove('cdhit.temp')
243         os.remove('cdhit.temp.bak.clstr')
244         os.remove('cdhit.temp.clstr')
245
246
```

./src/pythoscape/plugin/input/reprnode_stats.py

```
1  # Author: Alan Barber
2  # Created on: February 2, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plugins calculate
   statistics on representative nodes.
7
8  Classes:
9  CalcNodeSize - Add the number of nodes represented by a
   representative node as an attribute.
10 '''
11
12 from pythoscape.main.plugin import PythoscapePlugin
13
14 class CalcNodeSize(PythoscapePlugin):
15     '''Description: This plug-in adds the number of
   nodes in a representative node
16     as an attribute.
17
18     Environments: This plug-in will only work with
   PythoscapeEnvironment.
19     If you want to have this plug-in operate on a
   sub-network, use the
20     initial_net argument.
21
22     Example execution:
23     (Be sure to replace <interface>,<create
   interface>,<environment>,
24     and <create environment> below to the appropriate
   values. See the general
25     documentation for information on setting these lines
   appropriately)
26
27     #!Example begin!
28     import pythoscape.main.environments as env
29     import <interface>
30
31     #Import plug-ins
32     import pythoscape.plugin.input.reprnode_stats as r_s
33
34     #Create interface and environment
35     <create interface>
36     <create environment>
```

./src/pythoscape/plugin/input/reptime_stats.py

```
37
38     #Create plug-ins
39     plugin =
r_s.CalcNodeSize(<network>,<reptime_att>,<size_att>,initial_
l_net=<initial_net>)
40
41
42     #Execute plug-ins
43     <environment>.execute_plugin(plugin)
44     #!Example end!
45
46     Required arguments:
47
48     <network> - name of representative node network
49     <reptime_att> - name of attribute that indexes
representative nodes
50     <size_att> - name of node attribute that will
contain the calculated size
51
52     Optional arguments:
53
54     <initial_net> - default None, if you set this to the
name of a network, it will only consider
55                     nodes from that network
56
57     Because you can't run this plug-in using
NetworkEnvironment, <initial_net> is used to constrain
58     the plug-in to a specific sub-network.
59
60     Plug-in example:
61     plugin =
r_s.CalcNodeSize('my_rep_network','my_reptime_index','reptime
ode size')
62
63     This example would calculate the number of nodes in
each representative node for the representative node
64     network my_rep_network. It would use the
representative node index my_reptime_index and save the
65     attribute with the name reptime size.
66     '''
67
68     def __init__(self,network,reptime_att,size_att,
initial_net=None):
69         '''Create a new instance of CalcNodeSize
70
```

./src/pythoscape/plugin/input/repnodestats.py

```
71         Required arguments:
72
73         <network> - name of representative node
network
74         <repnodestats> - name of attribute that
indexes representative nodes
75         <size_attr> - name of node attribute that
will contain the calculated size
76
77         Optional arguments:
78
79         <initial_net> - default None, if you set
this to the name of a network, it will only consider
80                         nodes from that network
81         '''
82
83         #Required
84         PythoscapePlugin.__init__(self, locals())
85         self.Pythoscape = True
86         self.PythoscapeNetwork = False
87         self.description = 'Add cluster size as
attribute.'
88
89         #Plugin specific
90         self.repnodes = {}
91         self.clusters = {}
92         self.repnodestats = repnodestats
93         self.repnodestats_n = self.Network(network_name =
network)
94         self.initial_n = self.Network(network_name =
initial_net) if initial_net else None
95         self.size_attr = size_attr
96         self.setup('pull', self.pull_info)
97         self.setup('push', self.add_info)
98
99         def pull_info(self):
100             '''Plug-in method to pull all of the
representative node index attributes and
101                 calculate how many nodes are in each
representative node.
102
103         Required arguments:
104             None
105
106         Yields:
```


./src/pythoscape/plugin/input/repnodestats.py

```
107         Pulling objects for representative node
index attributes
108         '''
109
110         net_node = self.Node()
111         net_node.add_networks([self.repnodestats_n])
112         yield self.Attribute(element_obj=net_node,
attribute_name=self.repnodestats_att)
113         for att in self.outgoing:
114             self.repnodes[att.attribute_value] = att.
element_obj.id
115         all_node = self.Node()
116         if self.initial_n:
117             all_node.add_networks([self.initial_n])
118         yield self.Attribute(element_obj=all_node,
attribute_name=self.repnodestats_att)
119         for att in self.outgoing:
120             self.clusters[att.attribute_value] = self.
clusters.get(att.attribute_value, 0)+1
121
122         def add_info(self):
123             '''Plug-in method to yield all calculated
representative node sizes
124
125             Required arguments:
126                 None
127
128             Yields:
129                 Node attributes of representative node sizes
130             '''
131
132             for cluster in self.clusters:
133                 node = self.Node(id = self.repnodes[cluster])
134                 yield self.Attribute(element_obj=node,
attribute_name=self.
size_att,
135                                     attribute_value=self.
clusters[cluster])
136
137
```

C.5.5. ./src/pythoscape/plugin/input_bio modules

./src/pythoscape/plugin/input_bio/add_uniprot_info.py

```
1  # Author: Alan Barber
2  # Created on: October 19, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plugins that add
   annotations found in UniProt
7  records. This module needs a lot of work to re-write the
   code to use an object-
8  oriented Biopython UniProt record parser.
9
10  Functions:
11  make_info - pulls relevant annotations from a BioPython
   UniProt record object
12  read_dmp_file - makes a dictionary of an NCBI dmp file
13  link_taxonomic - uses NCBI name and node dmp files to
   link taxonomic ids to
14                  hierarchy levels
15
16  Classes:
17  ImportFromUniProt - Imports annotations from UniProt
   sequence records
18  '''
19
20  from Bio import SwissProt, Expasy
21  from pythoscape.main.plugin import PythoscapePlugin
22
23  #Annotations to add and which object attribute they are
   in.
24  INFORMATION = {'uniprot sequence':'sequence',
25                 'organism':'organism',
26                 'description':'description',
27                 'accessions':'accessions',
28                 'family':'comments',
29                 'subfamily':'comments',
30                 'superfamily':'comments',
31                 'pfam':'cross_references',
32                 'pdb':'cross_references',
33                 'taxonomy':'organism_classification'}
34
35  #If the annotation is a merged list, how to re-split it.
36  SPLIT_LIST = {'family':'',
37                'subfamily':'',
38                'superfamily':''}
```

./src/pythoscape/plugin/input_bio/add_uniprot_info.py

```
39
40
41 #Is the annotation in a list of annotations, and how to
  find it
42 IN_LIST = {'family':'SIMILARITY',
43            'subfamily':'SIMILARITY',
44            'superfamily':'SIMILARITY',
45            'pfam':'Pfam',
46            'pdb':'PDB'}
47
48 #How to split a text annotation
49 SPLIT_VALUE = {'family':'.',
50               'subfamily':'.',
51               'superfamily':'.'}
52
53 #What text end to take off
54 CHOOSE_END = {'family':' family',
55               'subfamily':' subfamily',
56               'superfamily':' superfamily'}
57
58 #What text beginning to take off
59 REMOVE_START = {'family':' Belongs to the ',
60                 'subfamily':' Belongs to the ',
61                 'superfamily':' Belongs to the '}
62
63 #If the value is a list of values -- how to find the
  appropriate one
64 COMPOUND_VALUE = {'pfam':(0,2),
65                  'pdb':(0,1)}
66
67 TAXONOMY = 'taxonomy'
68
69 #Where to find sequence IDs
70 IDS = set(['accessions'])
71
72 def make_info(record):
73     '''Pulls relevant annotations from a BioPython
  UniProt record object.
74     This needs to be rewritten to be based on an OO
  record parser.
75
76     Arguments:
77         record - Biopython UniProt entry record object
78
79     Yields:
```

./src/pythoscape/plugin/input_bio/add_uniprot_info.py

```
80         annotation name, annotation value
81     '''
82
83     for info in INFORMATION:
84         return_value = getattr(record, INFORMATION[info])
85         if info in IN_LIST:
86             new_value = None
87             for entry in return_value:
88                 if info in SPLIT_LIST: entry = entry.
split(SPLIT_LIST[info])
89                 if entry[0] == IN_LIST[info]:
90                     entry = entry[1:]
91                     if info in COMPOUND_VALUE:
92                         entry = ':'.join(entry[
COMPOUND_VALUE[info][0]:COMPOUND_VALUE[info][1]])
93                     else: entry = entry[0]
94                     if info in SPLIT_VALUE and info in
CHOOSE_END:
95                         entry = entry.split(SPLIT_VALUE[
info])
96                         new_entry = None
97                         for part_entry in entry:
98                             if part_entry.endswith(
CHOOSE_END[info]):
99                                 new_entry = part_entry[:-
len(CHOOSE_END[info])]
100                                 if info in REMOVE_START
and new_entry.startswith(REMOVE_START[info]):
101                                     new_entry = new_entry
[ len(REMOVE_START[info]): ]
102                                 entry = new_entry
103                                 if entry and new_value == None:
104                                     new_value = entry
105                                 elif entry and new_value:
106                                     new_value = ','.join([new_value,
entry])
107             return_value = new_value
108         yield info, return_value
109
110 def read_dmp_file(file_name, key_index, value_index):
111     '''Reads an NCBI dmp file and returns a dictionary
of key, value pairs
112
113     Arguments:
114         file_name - dmp file name
```

./src/pythoscape/plugin/input_bio/add_uniprot_info.py

```
115         key_index - column number of key
116         value_index - column number of value
117
118     Returns:
119         data dictionary
120     '''
121
122     return_data = {}
123     with open(file_name) as f:
124         for line in f:
125             line = line.strip().split('|')
126             return_data[line[key_index].strip()] = line[
value_index].strip()
127     return return_data
128
129 def link_taxonomic(name_file, node_file):
130     '''uses NCBI name and node dmp files to link
taxonomic ids to hierarchy levels
131
132     Arguments:
133         name_file - NCBI name file
134         node_file - NCBI node file
135
136     Returns:
137         dictionary: key = taxonomy name, value =
taxonomy level name
138     '''
139
140     all_names = read_dmp_file(name_file, 1, 0)
141     all_nodes = read_dmp_file(node_file, 0, 2)
142     for key in all_names.keys():
143         all_names[key] = all_nodes[all_names[key]]
144     return all_names
145
146 class ImportFromUniProt(PythoscapePlugin):
147     '''Description: This plug-in imports annotations
from UniProt sequence records.
148     The annotations it currently imports are as follows:
149     uniprot sequence
150     organism
151     description
152     accessions
153     family
154     subfamily
155     superfamily
```

./src/pythoscape/plugin/input_bio/add_uniprot_info.py

```
156     pfam
157     pdb
158     taxonomy
159
160     Currently, this is done using module defined
161     dictionaries and make_info. In the
162     future this will be changed to a much more
163     extensible method of using a
164     wrapper class on top of the Biopython UniProt record
165     object.
166
167     Optional databases: To get taxonomic information,
168     you must have
169     the nodes.dmp and names.dmp files from the following
170     location:
171     ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz
172
173     Environments: This plug-in will only work with either
174     PythoscapeEnvironment or PythoscapeNetwork.
175
176     Example execution:
177     (Be sure to replace <interface>,<create
178     interface>,<environment>,
179     and <create environment> below to the appropriate
180     values. See the general
181     documentation for information on setting these lines
182     appropriately)
183
184     #!Example begin!
185     import pythoscape.main.environments as env
186     import <interface>
187
188     #Import plug-ins
189     import
190     pythoscape.plugin.input.input_bio.add_uniprot_info as
191     a_u_i
192
193     #Create interface and environment
194     <create interface>
195     <create environment>
196
197     #Create plug-ins
198     plugin =
199     a_u_i.ImportFromUniProt(data_file=<data_file>, structure=<T
200     rue/False>,
```

```

./src/pythoscape/plugin/input_bio/add_uniprot_info.py

189 uniprot_id=<uniprot_id>,entry_name=<True/False>,
190 taxonomy_classifiers=<taxonomy_classifiers>,
191 node_file=<node_file>,name_file=<name_file>)
192
193
194     #Execute plug-ins
195     <environment>.execute_plugin(plugin)
196     #!Example end!
197
198     Required arguments:
199
200     None
201
202     Optional arguments:
203
204     <data_file> - default is None. If this is set then
the plug-in reads the records from the file.
205         If not, it pulls the records directly
from UniProt servers. If it is a file,
206         it expects the raw file format.
207
208     <structure> - default is False. Is this a structure
network?
209
210     <uniprot_id> - default is 'UniprotKB'. What is a
attribute name for the UniProt AC number?
211
212     <entry_name> - default is False. Match the
identifier against entry_name?
213
214     <taxonomy classifiers> - list of levels of the
taxonomy that you want included, like 'kindgom' or
'phylum'
215
216     <node_file> - path to NCBI node file
217
218     <name_file> - path to NCBI name file
219
220     Plug-in example:
221     plugin =
a_u_i.ImportFromUniProt(data_file='my_data.txt',uniprot_id
='Uniprot',

```



```

./src/pythoscape/plugin/input_bio/add_uniprot_info.py

222     taxonomy_classifiers=['kingdom', 'phylum'],
223     node_file='node.dmp', name_file='name.dmp')
224
225     This example will add annotations using the
226     downloaded file of raw UniProt data, my_data.txt
227     and map to UniProt ACs in the identifier Uniprot. It
228     will add taxonomic data for
229     kingdoms and phylum using the two NCBI files
230     node.dmp and name.dmp.
231     '''
232
233     def __init__(self, data_file=None, structure=False,
234                   uniprot_id=None, entry_name=False,
235                   taxonomy_classifiers=None, node_file=None
236                   , name_file=None):
237         '''Create a new instance of ImportFromUniProt.
238
239         Required arguments:
240
241         None
242
243         Optional arguments:
244
245         <data_file> - default is None. If this is set
246         then the plug-in reads the records from the file.
247         If not, it pulls the records
248         directly from UniProt servers. If it is a file,
249         it expects the raw file format.
250
251         <structure> - default is False. Is this a
252         structure network?
253
254         <uniprot_id> - default is 'UniprotKB'. What is a
255         attribute name for the UniProt AC number?
256
257         <entry_name> - default is False. Match the
258         identifier against entry_name?
259
260         <taxonomy classifiers> - list of levels of the
261         taxonomy that you want included, like 'kindgom' or
262         'phylum'
263
264         <node_file> - path to NCBI node file

```

./src/pythoscape/plugin/input_bio/add_uniprot_info.py

```
253
254     <name_file> - path to NCBI name file
255     '''
256
257     #Required
258     PythoscapePlugin.__init__(self, locals())
259     self.Pythoscape = True
260     self.PythoscapeNetwork = True
261     self.description = 'Import data from UniProt'
262
263     #Plugin specific
264     self.data_file = data_file
265     self.structure = structure
266     self.uniprot_id = uniprot_id if uniprot_id else
'UniprotKB'
267     self.entry_name = entry_name
268     self.taxonomy_classifiers = set(list(
taxonomy_classifiers)) if taxonomy_classifiers != None
else None
269     if taxonomy_classifiers != None and node_file !=
None and name_file != None:
270         self.tax_links = link_taxonomic(name_file,
node_file)
271     else:
272         self.tax_links = None
273     self._nodes = {}
274     self.setup('pull', self.get_nodes)
275     self.setup('push', self.push_info)
276
277     def get_nodes(self):
278         '''Pulls all nodes and the corresponding UniProt
ACs
279
280         Required arguments:
281             None
282
283         Yields:
284             all node search, UniProt AC identifier search
285             '''
286
287         yield self.Node()
288         for node in self.outgoing:
289             if self.structure:
290                 if not node.id[0:4] in self._nodes: self.
_nodes[node.id[0:4]] = set([])
```

./src/pythoscape/plugin/input_bio/add_uniprot_info.py

```
291         self._nodes[node.id[0:4]].add(node.id)
292     else:
293         for identifier in node.identifiers():
294             if identifier.identifier_type == self
.uniprot_id:
295                 if not identifier.
identifier_value in self._nodes:
296                     self._nodes[identifier.
identifier_value] = set([])
297                     self._nodes[identifier.
identifier_value].add(node.id)
298
299     def push_info(self):
300         '''Adds UniProt annotations
301
302         Required arguments:
303             None
304
305         Yields:
306             UniProt annotations
307         '''
308
309     if self.data_file:
310         f = open(self.data_file)
311         records = SwissProt.parse(f)
312     else:
313         records = []
314         for identifier in self._nodes:
315             handle = Expasy.get_sprot_raw(identifier)
316             try:
317                 record = SwissProt.read(handle)
318             except ValueError:
319                 print 'The following record was not
found or added: {0}'.format(identifier)
320                 records.append(record)
321         for record in records:
322             ids = []
323             if self.structure:
324                 for reference in record.cross_references:
325                     if reference[0] == 'PDB' and
reference[1].lower() in self._nodes:
326                         for id in self._nodes[reference[1]
].lower()]:
327                             ids.append(id)
328             elif not self.entry_name:
```

./src/pythoscape/plugin/input_bio/add_uniprot_info.py

```
329         ids = []
330         for accession in record.accessions:
331             if accession in self._nodes:
332                 ids = ids + list(self._nodes[
accession])
333         elif self.entry_name:
334             if record.entry_name in self._nodes:
335                 ids = list(self._nodes[record.
entry_name])
336         for id in ids:
337             node = self.Node(id=id)
338             for data_pnt,attr_value in make_info(
record):
339                 if data_pnt in IDS and self.structure:
340                     for id_attr in attr_value:
341                         node.add_identifiers([self.
Identifier(identifier_type=self.uniprot_id,
342
                             identifier_value=id_attr)])
343                     yield node
344                 elif data_pnt == TAXONOMY and
attr_value != None and self.taxonomy_classifiers != None\
345                     and self.tax_links != None:
346                     for tax_name in attr_value:
347                         if tax_name in self.tax_links
and self.tax_links[tax_name] in self.
taxonomy_classifiers:
348                             yield self.Attribute(
element_obj = node,
349
                             attribute_name = self.tax_links[tax_name],
350
                             attribute_value = tax_name)
351                 elif data_pnt not in IDS and data_pnt
!= TAXONOMY and attr_value != None:
352                     yield self.Attribute(element_obj
= node,
353
                             attribute_name = data_pnt,
354
                             attribute_value = attr_value)
```

```

./src/pythoscape/plugin/input_bio/import_structures.py

1  # Author: Alan Barber
2  # Created on: October 12, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plug-ins that import
   PDB structure paths into Pythoscape
7
8  Functions:
9  pull_PDB_chain - uses BioPython to open and then extract
   and save a single chain from a PDB
10
11  Classes:
12  PDBImportFromCSVFile - uses a CSV file to import PDB
   structures from a local PDB database.
13  StructureImportFromTextFile - imports structure network
   using text file
14  '''
15
16  from Bio.PDB.PDBParser import PDBParser
17  from Bio.PDB.PDBIO import PDBIO
18  from Bio.PDB.PDBIO import Select
19  from pythoscape.main.plugin import PythoscapePlugin
20  import gzip
21
22  def pull_PDB_chain(pdb_id,pdb_model,pdb_chain,pdb_dir,
   out_pdb):
23      '''pulls a structure from a local PDB database,
   extracts a specific chain
24          and then saves it.
25
26          Required arguments:
27              pdb_id - 4 letter/number PDB id
28              pdb_model - model letter/number
29              pdb_chain - chain letter/number
30              pdb_dir - root location for local mirror of
   PDB structure database
31              out_pdb - output file
32
33          Returns:
34              Nothing
35      '''
36
37      p = PDBParser(PERMISSIVE=1)

```

./src/pythoscape/plugin/input_bio/import_structures.py

```
38     f = gzip.open('{0}/{1}/pdb{2}.ent.gz'.format(pdb_dir,
pdb_id[1:3],pdb_id))
39     s=p.get_structure(pdb_id,f)
40     model=s[pdb_model]
41     selected_chain = model[pdb_chain]
42     io=PDBIO()
43     io.set_structure(s)
44     selection = Select()
45     selection.accept_chain = lambda x: x == selected_chain
46     io.save(out_pdb,selection)
47     f.close()
48
49 class PDBImportFromCSVFile(PythoscapePlugin):
50     '''Description: This plug-in imports PDB chains
using a local mirror
51     of the PDB database and a CSV file. The CSV file to
import structures
52     into Pythoscape has the following form:
53
54     pdb_id_1,model_1,chain_1,identifier_1(optional)
55     pdb_id_2,model_2,chain_2,identifier_2(optional)
56     ...
57
58     The identifier column only needs to be included if
you are associating
59     the structure nodes with pre-existing sequence nodes.
60
61     Environments: This plug-in will work with both
PythoscapeEnvironment and
62     PythoscapeNetwork. Using PythoscapeNetwork restricts
the inputted identifiers
63     to those nodes in a given network.
64
65     Example execution:
66     (Be sure to replace <interface>,<create
interface>,<environment>,
67     and <create environment> below to the appropriate
values. See the general
68     documentation for information on setting these lines
appropriately)
69
70     #!Example begin!
71     import pythoscape.main.environments as env
72     import <interface>
73
```

./src/pythoscape/plugin/input_bio/import_structures.py

```
74     #Import plug-ins
75     import pythoscape.plugin.input_bio.import_structures
    as i_s
76
77     #Create interface and environment
78     <create interface>
79     <create environment>
80
81     #Create plug-ins
82     plugin =
    i_s.PDBImportFromCSVFile(<pdb_path>,<csv_file>,<local_path
    >,<path_att>=<path_att>,<no_files>=<True or
83     False>,<add_to_sequence_network>=<add_to_sequence_network>)
84
85     #Execute plug-ins
86     <environment>.execute_plugin(plugin)
87     #!Example end!
88
89     Required arguments:
90     <pdb_path> = path to root of local copy of PDB
    database
91     <csv_file> = path to csv file describing input
    structures (see above)
92     <local_path> = path to directory to store inputted
    structure chains
93
94     Optional arguments:
95     <path_att> = default 'path', attribute to store
    structure path
96     <no_files> = default False, if True no files are
    copied
97     <add_to_sequence_network> = default None, if a value
    is set no new nodes are made and instead
98     node attributes are
99     added to the node that has the
100     identifier_value given
101     identifier_type
102     specified by the value set here
103
104     Plug-in example:
105     plugin =
```

```

./src/pythoscape/plugin/input_bio/import_structures.py

i_s.PDBImportFromCSVFile('/pdb_root','struc.csv','/home/st
rucs')
105
106     This example would pull from the PDB mirror at
/pdb_root the structures given in
107     struc.csv and put the copied chains into /home/strucs
108     '''
109
110     def __init__(self,pdb_path,csv_file,local_path,
path_att=None,no_files=False,
111                  add_to_sequence_network=False):
112         '''Create a new instance of PDBImportFromCSVFile
113
114         Required arguments:
115             <pdb_path> = path to root of local copy of
PDB database
116             <csv_file> = path to csv file describing
input structures (see above)
117             <local_path> = path to directory to store
inputted structure chains
118
119         Optional arguments:
120             <path_att> = default 'path', attribute to
store structure path
121             <no_files> = default False, if True no files
are copied
122             <add_to_sequence_network> = default None, if
a value is set no new nodes are made and instead
123                                     node attributes
are added to the node that has the
124                                     identifier_value
given in the csv with the
125                                     identifier_type
specified by the value set here
126
127         '''
128
129         #Required
130         PythoscapePlugin.__init__(self,locals())
131         self.Pythoscape = True
132         self.PythoscapeNetwork = True
133         self.description = 'Create structure network
from CSV file and PDB structure database'
134
135         #Plugin specific

```


./src/pythoscape/plugin/input_bio/import_structures.py

```
136         self.pdb_path = pdb_path
137         self.local_path = local_path
138         self.csv_file = csv_file
139         self.path_att = path_att if path_att else 'path'
140         self.no_files = no_files
141         self.add_to_seq = add_to_sequence_network
142         if self.add_to_seq:
143             self.identifier_keys = {}
144             self.setup('pull', self.get_identifiers)
145             self.setup('push', self.get_struc_info)
146
147     def get_identifiers(self):
148         '''Plug-in method to get identifiers and output
149         them
150
151         Required arguments:
152             None
153
154         Yields:
155             All identifier search
156         '''
157         query_node = self.Node()
158         yield query_node
159         for node in self.outgoing:
160             for identifier in node.identifiers():
161                 if identifier.identifier_type == self.
162                 add_to_seq:
163                     self.identifier_keys[identifier.
164                     identifier_value] = node.id
165
166     def get_struc_info(self):
167         '''Plug-in method to add structure nodes and
168         attributes
169
170         Required arguments:
171             None
172
173         Yields:
174             new structure attributes and new structure
175             nodes if <add_to_sequence_network> is None
176         '''
177         with open(self.csv_file) as f:
178             for line in f:
```

./src/pythoscape/plugin/input_bio/import_structures.py

```
176         line = line.strip().split(',')
177         if len(line) > 3:
178             node_id = line[3]
179             line=line[:3]
180         for entry in line: entry.lower()
181         file_path = '{0}/{1}_{2}_{3}.pdb'.format(
self.local_path,*line)
182         if not self.no_files:
183             pull_PDB_chain(line[0],int(line[1]),
line[2],
184                             self.pdb_path,
file_path)
185         ident = self.Identifier(identifier_type=
'pdb structure',identifier_value=line[0])
186         if not self.add_to_seq:
187             curr_node = self.Node(id=':'.join(
line))
188         else:
189             curr_node = self.Node(id=self.
identifier_keys[node_id])
190             curr_node.add_identifiers([ident])
191             yield curr_node
192             node_attr = self.Attribute(element_obj=
curr_node,
193                                         attribute_name
=self.path_att,
194                                         attribute_value=file_path)
195             yield node_attr
196
197
```

C.5.6. ./src/pythoscape/plugin/output modules

./src/pythoscape/plugin/output/output_attribute_tables.py

```
1  # Author: Alan Barber
2  # Created on: February 16, 2011
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plugins that output
   table-like files of attributes and identifiers.
7
8  Classes:
9  OutputIdentifierTable - output a table-like file of
   identifiers
10 OutputAttributeTable - output a table-like file of node
   attributes
11 '''
12
13 from pythoscape.main.plugin import PythoscapePlugin
14 from pythoscape.auxiliary.file_tools import
   make_data_table
15
16 class OutputIdentifierTable(PythoscapePlugin):
17     '''Description: This plug-in outputs a table-like
   file that describes
18         identifiers for each node in the following way:
19
20             identifier 1    identifier 2    etc.
21         node id
22
23         If a single node has multiple identifiers of the
   same type, they are
24         each outputted on a different line.
25
26         Environments: This plug-in will work with both
   PythoscapeEnvironment and
27         PythoscapeNetwork. Using PythoscapeNetwork restricts
   the output
28         to those nodes in a given network.
29
30         Example execution:
31         (Be sure to replace <interface>,<create
   interface>,<environment>,
32         and <create environment> below to the appropriate
   values. See the general
33         documentation for information on setting these lines
   appropriately)
```

./src/pythoscape/plugin/output/output_attribute_tables.py

```
34
35     #!Example begin!
36     import pythoscape.main.environments as env
37     import <interface>
38
39     #Import plug-ins
40     import
pythoscape.plugin.output.output_attribute_tables as o_a_t
41
42     #Create interface and environment
43     <create interface>
44     <create environment>
45
46     #Create plug-ins
47     plugin =
o_a_t.OutputIdentifierTable(<output_file>,csv=<True or
False>)
48
49     #Execute plug-ins
50     <environment>.execute_plugin(plugin)
51     #!Example end!
52
53     Required arguments:
54     <output_file> = name of output file
55
56     Optional arguments:
57     <csv> = default False, set to True to output a comma
seperated value file
58         instead of a table
59
60     Plug-in example:
61     plugin = o_a_t.OutputIdentifierTable('myfile.txt')
62
63     This example would output the table to the file
myfile.txt.
64     '''
65
66
67     def __init__(self,output_file,csv=False):
68         '''Create a new instance of OutputIdentifierTable
69
70         Required arguments:
71         <output_file> = name of output file
72
73         Optional arguments:
```

./src/pythoscape/plugin/output/output_attribute_tables.py

```
74         <csv> = default False, set to True to output
a comma seperated value file
75         instead of a table
76         '''
77
78     #Required
79     PythoscapePlugin.__init__(self, locals())
80     self.Pythoscape = True
81     self.PythoscapeNetwork = True
82     self.description = 'Output a table of node
identifiers.'
83
84     #Plugin specific
85     self.o_file = output_file
86     self.identifiers = set([])
87     self.nodes = set([])
88     self.identifier_keys = {}
89     self.csv = csv
90     self.setup('pull',self.get_identifiers)
91
92     def get_identifiers(self):
93         '''Plug-in method to get identifiers and output
them
94
95         Required arguments:
96             None
97
98         Yields:
99             All identifier search
100         '''
101
102         query_node = self.Node()
103         yield query_node
104         for node in self.outgoing:
105             for identifier in node.identifiers():
106                 self.identifiers.add(identifier.
identifier_type)
107                 self.nodes.add(node.id)
108                 if not (node.id,identifier.
identifier_type) in self.identifier_keys:
109                     self.identifier_keys[node.id,
identifier.identifier_type] = []
110                     self.identifier_keys[node.id,identifier.
identifier_type].append(identifier.identifier_value)
111                 make_data_table(self.o_file,15,self.nodes,self.
```

```

./src/pythoscape/plugin/output/output_attribute_tables.py

identifiers,self.identifier_keys,csv=self.csv)
112
113 class OutputAttributeTable(PythoscapePlugin):
114     '''Description: This plug-in outputs a table-like
file that describes
115     attributes for each node in the following way:
116
117         attribute 1    attribute 2    etc.
118     node id
119
120     Environments: This plug-in will work with both
PythoscapeEnvironment and
121     PythoscapeNetwork. Using PythoscapeNetwork restricts
the output
122     to those nodes in a given network.
123
124     Example execution:
125     (Be sure to replace <interface>,<create
interface>,<environment>,
126     and <create environment> below to the appropriate
values. See the general
127     documentation for information on setting these lines
appropriately)
128
129     #!Example begin!
130     import pythoscape.main.environments as env
131     import <interface>
132
133     #Import plug-ins
134     import
pythoscape.plugin.output.output_attribute_tables as o_a_t
135
136     #Create interface and environment
137     <create interface>
138     <create environment>
139
140     #Create plug-ins
141     plugin =
o_a_t.OutputAttributeTable(<output_file>,attribute_names=<
attribute_names>,
142
justify=<justify>,leave_out=<leave_out>,
143                                     csv=<True or
False>))
144

```

./src/pythoscape/plugin/output/output_attribute_tables.py

```
145     #Execute plug-ins
146     <environment>.execute_plugin(plugin)
147     #!Example end!
148
149     Required arguments:
150     <output_file> = name of output file
151
152     Optional arguments:
153     <attribute_names> = default None, list of attributes
154     to output, skip others
155     <justify> = default 15, number of spaces to justify
156     columns
157     <leave_out> = default None, list of attributes not
158     to output, use others
159     <csv> = default False, set to True to output a comma
160     seperated value file
161     instead of a table
162
163     Plug-in example:
164     plugin = o_a_t.OutputAttributeTable('myfile.txt')
165
166     This example would output the table to the file
167     myfile.txt.
168     '''
169
170     def __init__(self,output_file,attribute_names=None,
171     justify=None,leave_out=None,csv=False):
172         '''Create a new instance of OutputIdentifierTable
173
174         Required arguments:
175         <output_file> = name of output file
176
177         Optional arguments:
178         <attribute_names> = default None, list of
179         attributes to output, skip others
180         <justify> = default 15, number of spaces to
181         justify columns
182         <leave_out> = default None, list of
183         attributes not to output, use others
184         <csv> = default False, set to True to output
185         a comma seperated value file
186         instead of a table
187         '''
188
189     #Required
```


./src/pythoscape/plugin/output/output_attribute_tables.py

```
180         PythoscapePlugin.__init__(self, locals())
181         self.Pythoscape = True
182         self.PythoscapeNetwork = True
183         self.description = 'Output a table of node
attributes.'
184
185         #Plugin specific
186         self.o_file = output_file
187         self.attr_name = attribute_names
188         self.justify = justify if justify else 15
189         self.nodes = set([])
190         self.attr_keys = {}
191         self.leave_out = set(leave_out) if leave_out !=
None else None
192         self.attribute_names = attribute_names if
attribute_names else set([])
193         self.csv = csv
194         self.setup('pull', self.get_attrs)
195
196     def get_attrs(self):
197         '''Plug-in method to get attributes and output
them
198
199         Required arguments:
200             None
201
202         Yields:
203             All attribute search (unless a list is
provided and then they are searched
204             on the list one by one)
205         '''
206
207         if len(list(self.attribute_names)) > 0:
208             to_iter = self.attribute_names
209         else:
210             to_iter = [None]
211         for att_name in to_iter:
212             yield self.Attribute(element_obj=self.Node(),
attribute_name=att_name)
213             for out_attr in self.outgoing:
214                 self.attribute_names.add(out_attr.
attribute_name)
215                 self.nodes.add(out_attr.element_obj.id)
216                 self.attr_keys[out_attr.element_obj.id,
out_attr.attribute_name] = [out_attr.attribute_value]
```

./src/pythoscape/plugin/output/output_attribute_tables.py

```
217         if self.leave_out != None:
218             for item in self.leave_out:
219                 if item in self.attribute_names:
220                     self.attribute_names.remove(item)
221         make_data_table(self.o_file, self.justify, self.
nodes, list(self.attribute_names), self.attr_keys, csv=self.
csv)
```

./src/pythoscape/plugin/output/output_repnode_info.py

```
1  # Author: Alan Barber
2  # Created on: September 15, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plug-ins that output
   data tables or other information
7  about representative nodes and the nodes that they
   represent.
8
9  Classes:
10 OutputAttributeClusters - outputs a file that gives
   information about what nodes or attributes
11                          are in a set of/all
   representative nodes
12  '''
13
14 from pythoscape.main.plugin import PythoscapePlugin
15
16 class OutputAttributeClusters(PythoscapePlugin):
17     '''Description: This plug-in outputs a file that
   gives information about
18     what nodes or attributes are either a given set of
   representative nodes
19     or all representative nodes. The output file has the
   following form:
20
21     Representative node id
22     -----
23     Node_id_1 (tab) Given attribute/identifier
24     Node_id_2 (tab) Given attribute/identifier
25     ...
26
27     Environments: This plug-in will only work with
   PythoscapeEnvironment.
28
29     Example execution:
30     (Be sure to replace <interface>,<create
   interface>,<environment>,
31     and <create environment> below to the appropriate
   values. See the general
32     documentation for information on setting these lines
   appropriately)
33
```

./src/pythoscape/plugin/output/output_repnod_info.py

```
34      #!Example begin!
35      import pythoscape.main.environments as env
36      import <interface>
37
38      #Import plug-ins
39      import pythoscape.plugin.output.output_repnod_info
as o_r_i
40
41      #Create interface and environment
42      <create interface>
43      <create environment>
44
45      #Create plug-ins
46      plugin =
o_r_i.OutputAttributeClusters(<network>,<repnod_att>,<out
put_file>,
47                                  output_att =
<output_att>,is_identifir = <True or False>,
48                                  in_repnodes =
<in_repnodes>,<no_titles = <True or False>)
49
50
51      #Execute plug-ins
52      <environment>.execute_plugin(plugin)
53      #!Example end!
54
55      Required arguments:
56
57      <network> - name of representative node network
58      <repnod_att> - name of attribute that indexes
representative nodes
59      <output_file> - name of file to output
60
61      Optional arguments:
62
63      <output_att> = default None, name of attribute or
identifir to output
64                  along with node names
65
66      <is_identifir> = default False, True if
<output_att> is an identifir type
67
68      <in_repnodes> = default False, set to a file name if
you want to restrict to
69                  a set of representative nodes. This
```

```

./src/pythoscape/plugin/output/output_renode_info.py

file should be one representative
70         node id per line.
71
72     <no titles> = default False, if set to True, the
representative node header lines
73         are not included in the output file
74
75     Plug-in example:
76     plugin =
o_r_i.OutputAttributeClusters('my_network', 'cdhit
0.7', 'my_output.txt',
77
output_att='hit?')
78
79     This would output the values of attribute 'hit?' for
all the represented nodes
80     that are defined in the representative node network
'my_network' with representative
81     node indexing attribute 'cdhit 0.7'. The output
would be written to 'my_output.txt'.
82     '''
83
84     def __init__(self, network, renode_att, output_file,
85                 output_att=None, is_identifier=False,
in_renodes=False, no_titles=False):
86         '''Create a new instance of
OutputAttributeClusters
87
88         Required arguments:
89
90         <network> - name of representative node
network
91         <renode_att> - name of attribute that
indexes representative nodes
92         <output_file> - name of file to output
93
94         Optional arguments:
95
96         <output_att> = default None, name of
attribute or identifier to output
97         along with node names
98
99         <is_identifier> = default False, True if
<output_att> is an identifier type
100

```

./src/pythoscape/plugin/output/output_repnod_info.py

```
101         <in_repnodes> = default False, set to a file
name if you want to restrict to
102         a set of representative
nodes. This file should be one representative
103         node id per line.
104
105         <no_titles> = default False, if set to True,
the representative node header lines
106         are not included in the output
file
107         '''
108
109         #Required
110         PythoscapePlugin.__init__(self, locals())
111         self.Pythoscape = True
112         self.PythoscapeNetwork = False
113         self.description = 'Output node attributes (or
identifiers) that are in a repnode'
114
115         #Plugin specific
116         self.repnod_n = self.Network(network_name=
network)
117         self.repnod_att = repnod_att
118         self.output_att = output_att
119         self.is_iden = is_identifier
120         self.output_file = output_file
121         self.in_repnodes=in_repnodes
122         self.no_titles = no_titles
123         self.clusters = {}
124         self.nodes_in_clus = {}
125         self.repnodes = {}
126         self.atts = {}
127         self.setup('pull',self.pull_info)
128
129         def pull_info(self):
130             '''Plug-in method to pull all of the required
information to create the class specified
131             output file and then make it.
132
133             Required arguments:
134                 None
135
136             Yields:
137                 Pulling objects for representative node
index attributes, representative network
```

./src/pythoscape/plugin/output/output_repnod_info.py

```
138         nodes and if needed target node attributes
or identifiers
139         """
140         yield self.Attribute(element_obj=self.Node(),
attribute_name=self.repnod_att)
141         for att in self.outgoing:
142             self.clusters[att.element_obj.id] = att.
attribute_value
143             self.nodes_in_clus[att.attribute_value] =
self.nodes_in_clus.get(att.attribute_value, [])+ [att.
element_obj.id]
144             if not self.is_iden and self.output_att != None:
145                 yield self.Attribute(element_obj=self.Node(),
attribute_name=self.output_att)
146                 for att in self.outgoing:
147                     self.atts[att.element_obj.id] = att.
attribute_value
148                 elif self.output_att != None:
149                     yield self.Node()
150                     for node in self.outgoing:
151                         for identifier in node.identifiers():
152                             if identifier.identifier_type == self
.output_att:
153                                 self.atts[node.id] = identifier.
identifier_value
154                     net_node = self.Node()
155                     net_node.add_networks([self.repnod_n])
156                     yield net_node
157                     for node in self.outgoing:
158                         if node.id in self.clusters:
159                             self.repnodes[self.clusters[node.id]] =
node.id
160                     repnod_filter = set([])
161                     if self.in_repnodes:
162                         with open(self.in_repnodes) as f:
163                             for line in f:
164                                 repnod_filter.add(self.clusters[line
.strip()])
165                     with open(self.output_file, 'w') as f:
166                         for clus_num in self.nodes_in_clus:
167                             if self.in_repnodes and clus_num not in
repnod_filter:
168                                 pass
169                             else:
170                                 if not self.no_titles:
```

./src/pythoscape/plugin/output/output_repnod_info.py

```
171                                     f.write(  
    '{0}\n-----\n'.format(self  
    .repnodes[clus_num]))  
172         for node_id in self.nodes_in_clus[  
clus_num]:  
173             if self.output_att != None and  
node_id in self.atts:  
174                 f.write('{0}\t{1}\n'.format(  
node_id, self.atts[node_id]))  
175             elif self.output_att == None:  
176                 f.write('{0}\n'.format(  
node_id))  
177         if not self.no_titles:  
178             f.write('\n')  
179
```


./src/pythoscape/plugin/output/output_table_runs.py

```
1  # Author: Alan Barber
2  # Created on: November 2, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plug-ins to output
   table runs that can be used to calculate
7  networks outside Pythoscape in parallel.
8
9  Functions:
10  output_table_edges - breaks up a list of edges into
   various files given a line size limit
11  filtered_combinations - yields all combinations of the
   filter set + the filter set with everything else
12
13  Classes:
14  SequenceTableRun - sets up a parallel run for sequence
   nodes
15  StructureTableRun - sets up a parallel run for structure
   nodes
16  '''
17
18  from pythoscape.main.plugin import PythoscapePlugin
19  import itertools
20  from pythoscape.auxiliary.file_tools import
   write_table_files, write_fasta_file
21
22  def filtered_combinations(iterable, r, filter_on):
23      '''Yields itertools.combinations(filter_first) +
   product(iterable, filter_first), i.e.
24      all combinations of the filter set + the filter set
   with everything else.
25
26      Arguments:
27          iterable - argument to iterate over
28          r - number of objects to provide
29          filter_on - Only provide combinations of these
30
31      Yields:
32          tuple combination
33      '''
34      filter_on = set(filter_on)
35      for combin in itertools.combinations(filter_on, r):
36          yield combin
```

./src/pythoscape/plugin/output/output_table_runs.py

```
37     for combin in itertools.product(set(iterable)-
38                                     filter_on,filter_on,repeat=r-1):
39         yield combin
40
41 def output_table_edges(edges,directory,limit,compress=
42 False):
43     '''Breaks up a list of edges into various files
44     given a line size limit
45
46     Arguments:
47         edges - list of all edges, edges should be
48         length 2 sequences
49         directory - directory to output all files
50         limit - edge number limit per file
51
52     Arguments (with defaults):
53         compress - default False, gzip all files?
54
55     Returns:
56         Nothing
57     '''
58     current_edges,i,j = [],1,1
59     for node_1,node_2 in edges:
60         current_edges.append((node_1,node_2))
61         if j >= limit:
62             write_table_files(directory,current_edges,i,
63 compress)
64             current_edges,j = [],0
65             i += 1
66             j += 1
67         write_table_files(directory,current_edges,i,compress)
68
69 class SequenceTableRun(PythoscapePlugin):
70     '''Description: This plug-in outputs all edges in
71     the complete network
72     in numbered table file and sequence file so that
73     edges can be calculated
74     outside of Pythoscape in parallel.
75
76     Environments: This plug-in will work with both
77     PythoscapeEnvironment
78     and PythoscapeNetwork. Using PythoscapeNetwork
79     restricts the outputted
80     edges to the nodes in the specified network.
```

./src/pythoscape/plugin/output/output_table_runs.py

```
73     Example execution:
74     (Be sure to replace <interface>,<create
interface>,<environment>,
75     and <create environment> below to the appropriate
values. See the general
76     documentation for information on setting these lines
appropriately)
77
78     #!Example begin!
79     import pythoscape.main.environments as env
80     import <interface>
81
82     #Import plug-ins
83     import pythoscape.plugin.output.output_table_runs as
o_t_r
84
85     #Create interface and environment
86     <create interface>
87     <create environment>
88
89     #Create plug-ins
90     plugin =
o_t_r.SequenceTableRun(<directory>,<runs_in_file>,sequence
_att=<sequence_att>,
91
seq_file=<seq_file>,compress=<True or False>)
92
93     #Execute plug-ins
94     <environment>.execute_plugin(plugin)
95     #!Example end!
96
97     Required arguments:
98
99     <directory> - location on disk to output table files
100
101     <runs_in_file> - number of edges to calculate in
each table
102
103     Optional arguments:
104
105     <sequence_att> - default 'sequence', if the sequence
attribute is not named 'sequence',
106                     this needs to be set to the
appropriate value
107
```

./src/pythoscape/plugin/output/output_table_runs.py

```
108     <seq_file> - default <directory>/sequences.fa; this
is the name and location of the
109         file that contains all the sequences
110
111     <compress> - default False, should the plug-in
compress all the outputs with gzip?
112
113     <new_nodes_only> - default False, set this to True
if you just added nodes to the
114         an already existing network and
you only want to output edges
115         that correspond to those nodes.
116
117     <new_node_file> - default None, set this to a file
name to use only new nodes
118         that are identified by the
file as the new node set. The file
119         should be set up as follows:
120         identifier_name
121         identifier_1
122         identifier_2
123         ...
124
125     Plug-in example:
126     plugin =
o_t_r.SequenceTableRun('/home/myuser/mydata/tables',100000
)
127
128     This example would output all the tables in the
directory /home/myuser/mydata/tables
129     and each table file would run 100000 edges (a
reasonable number for a large network).
130     '''
131
132     def __init__(self,directory,runs_in_file,sequence_att
=None,seq_file=None,compress=False,
133         new_nodes_only=False,new_node_file=None):
134         '''Create a new instance of SequenceTableRun
135
136         Required arguments:
137
138         <directory> - location on disk to output
table files
139
140         <runs_in_file> - number of edges to
```

```

./src/pythoscape/plugin/output/output_table_runs.py

calculate in each table
141
142         Optional arguments:
143
144             <sequence_att> - default 'sequence', if the
sequence attribute is not named 'sequence',
145                 this needs to be set to the
appropriate value
146
147             <seq_file> - default
<directory>/sequences.fa; this is the name and location
of the
148                 file that contains all the
sequences
149
150             <compress> - default False, should the
plug-in compress all the outputs with gzip?
151
152             <new_nodes_only> - default False, set this
to True if you just added nodes to the
153                 an already existing
network and you only want to output edges
154                 that correspond to those
nodes.
155
156             <new_node_file> - default None, set this to
a file name to use only new nodes
157                 that are identified by the
file as the new node set. The file
158                 should be set up as follows:
159                 identifier_name
160                 identifier_1
161                 identifier_2
162                 ...
163             '''
164
165         #Required
166         PythoscapePlugin.__init__(self, locals())
167         self.Pythoscape = True
168         self.PythoscapeNetwork = True
169         self.description = 'Output sequences for a
remote similarity run.'
170
171         #Plugin specific
172         self.directory = directory

```

./src/pythoscape/plugin/output/output_table_runs.py

```
173         self.runs = runs_in_file
174         self.sequence_att = sequence_att if sequence_att
else 'sequence'
175         self.seq_file = seq_file if seq_file else
'{0}/sequences.fa'.format(directory)
176         self.compress=compress
177         self.new_nodes = new_nodes_only
178         self.nn_file = new_node_file
179         self._sequences = {}
180         self.setup('pull',self.get_sequences)
181
182     def get_sequences(self):
183         '''Pulls all nodes and the corresponding sequences
184
185         Required arguments:
186             None
187
188         Yields:
189             all node search, possibly all edge search,
sequence attribute search
190         '''
191         yield self.Attribute(element_obj=self.Node(),
attribute_name=self.sequence_att)
192         for attribute in self.outgoing:
193             self._sequences[attribute.element_obj.id] =
attribute.attribute_value
194         if self.new_nodes:
195             if self.nn_file:
196                 ids_to_use = set([])
197                 filt_nodes = set([])
198                 with open(self.nn_file) as f:
199                     iden_name = f.next().strip()
200                     for line in f:
201                         ids_to_use.add(line.strip())
202                 yield self.Node()
203                 for node in self.outgoing:
204                     for identifier in node.identifiers():
205                         if identifier.identifier_type ==
iden_name and identifier.identifier_value in ids_to_use:
206                             filt_nodes.add(node.id)
207             else:
208                 filt_nodes = set(self._sequences.keys())
209                 yield self.Edge()
210                 for edge in self.outgoing:
211                     for id in [edge.source_node.id,edge.
```

```

./src/pythoscape/plugin/output/output_table_runs.py

target_node.id]:
212         if id in filt_nodes: filt_nodes.
remove(id)
213         edge_combo = filtered_combinations(self.
_sequences.keys(),2,filt_nodes)
214     else:
215         edge_combo = itertools.combinations(self.
_sequences.keys(),2)
216         output_table_edges(edge_combo,self.directory,self
.runs,self.compress)
217         write_fasta_file(self.seq_file,self._sequences)
218
219 class StructureTableRun(PythoscapePlugin):
220     '''Description: This plug-in outputs all edges in
the complete network
221         in numbered table file with structure file names so
that edges can be calculated
222         outside of Pythoscape in parallel.
223
224         Environments: This plug-in will work with both
PythoscapeEnvironment
225         and PythoscapeNetwork. Using PythoscapeNetwork
restricts the outputted
226         edges to the nodes in the specified network.
227
228         Example execution:
229         (Be sure to replace <interface>,<create
interface>,<environment>,
230         and <create environment> below to the appropriate
values. See the general
231         documentation for information on setting these lines
appropriately)
232
233         #!Example begin!
234         import pythoscape.main.environments as env
235         import <interface>
236
237         #Import plug-ins
238         import pythoscape.plugin.output.output_table_runs as
o_t_r
239
240         #Create interface and environment
241         <create interface>
242         <create environment>
243

```

./src/pythoscape/plugin/output/output_table_runs.py

```
244         #Create plug-ins
245         plugin =
o_t_r.StructureTableRun(<directory>,<runs_in_file>,path_at
t=<path_att>)
246
247         #Execute plug-ins
248         <environment>.execute_plugin(plugin)
249         #!Example end!
250
251         Required arguments:
252
253         <directory> - location on disk to output table files
254
255         <runs_in_file> - number of edges to calculate in
each table
256
257         Optional arguments:
258
259         <path_att> - default 'path', if the structure path
att is something other than path,
260                     then this needs to be set
261
262         Plug-in example:
263         plugin =
o_t_r.StructureTableRun('/home/myuser/mydata/tables',1000)
264
265         This example would output all the tables in the
directory /home/myuser/mydata/tables
266         and each table file would run 1000 edges (a
reasonable number for a large structure
267         network).
268         '''
269         def __init__(self,directory,runs_in_file,path_att=
None):
270             '''Create a new instance of StructureTableRun
271
272             Required arguments:
273
274             <directory> - location on disk to output
table files
275
276             <runs_in_file> - number of edges to
calculate in each table
277
278             Optional arguments:
```


./src/pythoscape/plugin/output/output_table_runs.py

```
279
280         <path_att> - default 'path', if the
structure path att is something other than path,
281             then this needs to be set
282         '''
283
284         #Required
285         PythoscapePlugin.__init__(self, locals())
286         self.Pythoscape = True
287         self.PythoscapeNetwork = True
288         self.description = 'Output structures for a
remote structural similarity run.'
289
290         #Plugin specific
291         self.directory = directory
292         self.runs = runs_in_file
293         self.path_att = path_att if path_att else 'path'
294         self.structures = []
295         self.setup('pull', self.get_structures)
296
297     def get_structures(self):
298         '''Pulls all nodes and the corresponding paths
299
300         Required arguments:
301             None
302
303         Yields:
304             all node search, path attribute search
305         '''
306         yield self.Attribute(element_obj=self.Node(),
attribute_name=self.path_att)
307         for attribute in self.outgoing:
308             self.structures.append(attribute.
attribute_value.split('/')[ -1])
309             output_table_edges(itertools.combinations(self.
structures, 2),
310                             self.directory, self.runs)
311
312
```

./src/pythoscape/plugin/output/output_xgmml.py

```
1  # Author: Alan Barber
2  # Created on: September 15, 2010
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plug-ins to output
   table runs that can be used to calculate
7  networks outside Pythoscape in parallel.
8
9  Functions:
10 output_table_edges - breaks up a list of edges into
   various files given a line size limit
11
12 Classes:
13 OutputXGMML - outputs an XGMML file from a given
   environment
14 '''
15
16 from pythoscape.main.plugin import PythoscapePlugin
17 from pythoscape.auxiliary.xml_tools import output_elemtree
18
19 class OutputXGMML(PythoscapePlugin):
20     '''Description: This plug-in outputs a network held
   in Pythoscape into
21     a XGMML file that can be opened using Cytoscape.
22
23     Environments: This plug-in will work with both
   PythoscapeEnvironment
24     and PythoscapeNetwork. Using PythoscapeNetwork
   restricts the edges to
25     the nodes in the specified network. Make sure to use
   PythoscapeNetwork
26     if you have created and want to output a meta-node
   network or a network
27     of selected nodes.
28
29     Example execution:
30     (Be sure to replace <interface>,<create
   interface>,<environment>,
31     and <create environment> below to the appropriate
   values. See the general
32     documentation for information on setting these lines
   appropriately)
33
```

`./src/pythoscape/plugin/output/output_xgmml.py`

```
34     #!/Example begin!
35     import pythoscape.main.environments as env
36     import <interface>
37
38     #Import plug-ins
39     import pythoscape.plugin.output.output_xgmml as o_x
40
41     #Create interface and environment
42     <create interface>
43     <create environment>
44
45     #Create plug-ins
46     plugin =
47     o_x.OutputXGMML(<output>,<label>,filt_name=<filt_name>,
48                       filt_dir = '>','>','=','<'
49     or '<=>',filt_value=<filt_value>)
50
51     #Execute plug-ins
52     <environment>.execute_plugin(plugin)
53     #!/Example end!
54
55     Required arguments:
56
57     <output> - name of file into which network will be
58     outputted
59
60     <label> - XGMML label of network. In Cytoscape, this
61     is the title of the network.
62
63     Optional arguments:
64
65     The three parameters filt_name, filt_dir and
66     filt_value are used to pre-filter the
67     edges that are outputted into the network. Any edge
68     that doesn't meet the filter
69     will not be included.
70
71     <filt_name> - edge attribute that you want to filter
72     on
73
74     <filt_dir> - pick '>','>','=','<' or '<=' depending on
75     if you want edges with the
76     attribute greater or less than the filter value
```

```

./src/pythoscape/plugin/output/output_xgmml.py

71     <filt_value> - the value of the filter
72
73     Plug-in example:
74     plugin = o_x.OutputXGMML('my_net.xgmml', 'My
network!', filt_name='-log10(E)',
75                               filt_dir='>', filt_value=5)
76
77     This would output the network to the file
my_net.xgmml with the label 'My network!'. Only edges
78     that have the edge attribute -log10(E) of greater
than 5 will be included in the file.
79     '''
80
81     def __init__(self, output, label, filt_name=None,
filt_dir=None, filt_value=None):
82         '''Create a new instance of OutputXGMML
83
84         Required arguments:
85
86         <output> - name of file into which network
will be outputted
87
88         <label> - XGMML label of network. In
Cytoscape, this is the title of the network.
89
90         Optional arguments:
91
92         <filt_name> - edge attribute that you want
to filter on
93
94         <filt_dir> - pick '>', '>=', '<' or '<='
depending on if you want edges with the
95         attribute greater or less than the filter
value
96
97         <filt_value> - the value of the filter
98         '''
99
100        #Required
101        PythoscapePlugin.__init__(self, locals())
102        self.Pythoscape = True
103        self.PythoscapeNetwork = True
104        self.description = 'Output all nodes and edges
as XGMML'
105

```

./src/pythoscape/plugin/output/output_xgmml.py

```
106         #Plugin specific
107         self._nodes = {}
108         self._edges = {}
109         if filt_name and filt_dir and filt_value != None:
110             self.filt = self.Filter(filter_name=filt_name
111 ,direction=filt_dir,filter_value=filt_value)
112         else: self.filt = None
113         self.output = output
114         self.label = label
115         self.setup('pull',self.get_all)
116
117     def get_all(self):
118         '''Pulls all nodes, node attributes, edge and
119         edge attributes for XGMML output
120
121         Required arguments:
122             None
123
124         Yields:
125             all node search, all node attribute search,
126             all edge search, all edge attribute search
127         '''
128         yield self.Node()
129         for node in self.outgoing:
130             self._nodes[node.id] = []
131             identifier_sets = {}
132             for identifier in node.identifiers():
133                 if not str(identifier.identifier_type) in
134 identifier_sets:
135                     identifier_sets[str(identifier.
136 identifier_type)] = []
137                     identifier_sets[str(identifier.
138 identifier_type)].append(str(identifier.identifier_value))
139                 for set in identifier_sets:
140                     self._nodes[node.id].append((set,','.join
141 (identifier_sets[set])))
142             yield self.Attribute(element_obj=self.Node())
143             for attribute in self.outgoing:
144                 node_name = attribute.element_obj.id
145                 self._nodes[node_name].append((attribute.
146 attribute_name,attribute.attribute_value))
147             yield self.Edge(filter=self.filt)
148             for edge in self.outgoing:
149                 self._edges[edge.source_node.id,edge.
```

./src/pythoscape/plugin/output/output_xgmml.py

```
target_node.id] = []
143     yield self.Attribute(element_obj=self.Edge(filter
= self.filt))
144     for attribute in self.outgoing:
145         edge_name = (attribute.element_obj.
source_node.id, attribute.element_obj.target_node.id)
146         self._edges[edge_name].append((attribute.
attribute_name, attribute.attribute_value))
147         output_elemtree(self._nodes, self._edges, self.
output, self.label)
```

C.5.7. ./src/pythoscape/plugin/output_bio modules

Empty package

C.5.8. ./src/pythoscape/plugin/output_r modules

./src/pythoscape/plugin/output_r/output_correlation.py

```
1  # Author: Alan Barber
2  # Created on: July 7, 2012
3  # Copyright (c) 2012, The Regents of the University of
   California. All rights reserved.
4
5  '''
6  Description: This module contains plug-ins that
   calculate correlations between edge attributes.
7
8  Functions:
9  do_correlation_plot - takes an edge attribute dictionary
   and outputs a plot from R
10
11  Classes:
12  CorrelationPlot - outputs a plot from R that shows the
   correlation between two edge attributes.
13  '''
14
15  from pythoscape.main.plugin import PythoscapePlugin
16  import rpy
17
18  def do_correlation_plot(edge_atts, att_x, att_y):
19      '''Uses R to output a plot that shows the
   correlation between two edge attributes.
20
21          Required arguments:
22          edge_atts - a dictionary of edge attributes
   in the form:
23
24                      dict[edge] = [att_x, att_x]
25          att_x - name of att_x (string)
26          att_y - name of att_y (string)
27
28          Returns:
29          Nothing
30      '''
31
32      xs, ys = [], []
33      for x, y in edge_atts.values():
34          xs.append(x)
35          ys.append(y)
36      rpy.set_default_mode(rpy.NO_CONVERSION)
37      d = rpy.r.data_frame(x=xs, y=ys)
38      model = rpy.r("y ~ x")
39      fitted_model = rpy.r.lm(model, data = d)
40      rpy.set_default_mode(rpy.BASIC_CONVERSION)
```


./src/pythoscape/plugin/output_r/output_correlation.py

```
40     summary = rpy.r.summary(fitted_model)
41     fitted_model = rpy.r.lm(model, data = d)
42     gradient = fitted_model['coefficients']['x']
43     yintercept= fitted_model['coefficients']['
'(Intercept)']
44     rpy.r.plot(x=xs, y=ys, xlab=att_x, ylab=att_y)
45     rpy.r.abline(a=yintercept, b=gradient, col="red")
46     leg_text = 'Best fit line: {:.3}x+{:.3}\nR-squared
is {:.3}'.format(gradient,yintercept,summary[
'adj.r.squared'])
47     rpy.r.legend("topleft", bty="n", legend=leg_text)
48     rpy.r.dev_off()
49
50 class CorrelationPlot(PythoscapePlugin):
51     '''Description: This plug-in calculates a
correlation plot of two edge attributes
52     using the R statistics package and Rpy as the python
to R interface.
53
54     Environments: This plug-in will work with both
PythoscapeEnvironment and
55     PythoscapeNetwork. Using PythoscapeNetwork restricts
the inputted identifiers
56     to those nodes in a given network.
57
58     Example execution:
59     (Be sure to replace <interface>,<create
interface>,<environment>,
60     and <create environment> below to the appropriate
values. See the general
61     documentation for information on setting these lines
appropriately)
62
63     #!Example begin!
64     import pythoscape.main.environments as env
65     import <interface>
66
67     #Import plug-ins
68     import pythoscape.plugin.output_r.output_correlation
as o_c
69
70     #Create interface and environment
71     <create interface>
72     <create environment>
73
```

./src/pythoscape/plugin/output_r/output_correlation.py

```
74     #Create plug-ins
75     plugin =
o_c.CorrelationPlot(<att_x>,<att_y>,<outfile>,without_rsq=
<True or False>,
76
transform_x=<transform_x>,transform_y=<transform_y>,
77
att_x_name=<att_x_name>,att_y_name=<att_y_name>)
78
79     #Execute plug-ins
80     <environment>.execute_plugin(plugin)
81     #!Example end!
82
83     Required arguments:
84     <att_x> = name of edge attribute for x axis
85     <att_y> = name of edge attribute for y axis
86     <outfile> = name of output pdf file
87
88     Optional arguments:
89     <without_rsq> = default False, if True does not
calculate rsq for correlation plot
90     <transform_x> = default None, function or callable
object that transforms the x edge attribute
91     <transform_y> = default None, function or callable
object that transforms the y edge attribute
92     <att_x_name> = default None, renames x attribute on
output plot
93     <att_y_name> = default None, renames y attribute on
output plot
94
95     Plug-in example:
96     plugin =
o_c.CorrelationPlot('edge_att_1','edge_att_2','output.pdf'
)
97
98     This example would output a pdf plot to 'output.pdf'
of edge_att_1 on the x-axis
99     and edge_att_2 on the y-axis and the r-square
correlation between them.
100     '''
101
102     def __init__(self,att_x,att_y,outfile,without_rsq=
False,
103
transform_x=None,transform_y=None,
att_x_name=None,
```

./src/pythoscape/plugin/output_r/output_correlation.py

```
104         att_y_name=None):
105     '''Create a new instance of CorrelationPlot
106
107     Required arguments:
108         <att_x> = name of edge attribute for x axis
109         <att_y> = name of edge attribute for y axis
110         <outfile> = name of output pdf file
111
112     Optional arguments:
113         <without_rsqu> = default False, if True does
114 not calculate rsqu for correlation plot
115         <transform_x> = default None, function or
116 callable object that transforms the x edge attribute
117         <transform_y> = default None, function or
118 callable object that transforms the y edge attribute
119         <att_x_name> = default None, renames x
120 attribute on output plot
121         <att_y_name> = default None, renames y
122 attribute on output plot
123     '''
124
125     #Required
126     PythoscapePlugin.__init__(self, locals())
127     self.Pythoscape = True
128     self.PythoscapeNetwork = True
129     self.description = 'Output edge attribute
130 correlation plot'
131
132     #Plugin specific
133     self.att_x = att_x
134     self.att_y = att_y
135     self.trans_x = transform_x
136     self.trans_y = transform_y
137     self.outfile = outfile
138     self.without_rsqu = without_rsqu
139     self.setup('pull', self.pull_edges)
140
141     def pull_edges(self):
142         '''Pulls two defined edge attributes and outputs
143 plot
144
145         Required arguments:
146             None
147
148         Yields:
```

./src/pythoscape/plugin/output_r/output_correlation.py

```
142         edge att_x, edge att_y
143     '''
144
145     edge_atts = {}
146     yield self.Attribute(attribute_name=self.att_x,
147 element_obj=self.Edge())
147     for att in self.outgoing:
148         edge_atts[att.element_obj.source_node.id, att.
149 element_obj.target_node.id] = [att.attribute_value, None]
149     yield self.Attribute(attribute_name=self.att_y,
150 element_obj=self.Edge())
150     for att in self.outgoing:
151         if (att.element_obj.source_node.id, att.
152 element_obj.target_node.id) in edge_atts:
152         edge_atts[att.element_obj.source_node.id,
153 att.element_obj.target_node.id][1] = att.attribute_value
153     for edge in edge_atts.keys():
154         if edge_atts[edge][1] == None:
155             del edge_atts[edge]
156         else:
157             if self.trans_x != None:
158                 edge_atts[edge][0] = self.trans_x(
159 edge_atts[edge][0])
159             if self.trans_y != None:
160                 edge_atts[edge][1] = self.trans_y(
161 edge_atts[edge][1])
161     rpy.r.pdf(file=self.outfile)
162     do_correlation_plot(edge_atts, self.att_x, self.
163 att_y)
```

Appendix D. Supplemental information for chapter 3

Table D.1. Enzymatic activities of AP superfamily members. References for each activity are given in the table.

Enzyme	Primary or Biological Activity	Other Reported Activities			
		Phosphate/ phosphonate monoester hydrolysis	Phosphate/ phosphonate diester hydrolysis	Sulfate ester hydrolysis	Phosphite oxidation
One-metal	Arylsulfatase	Sulfate monoester hydrolysis	(64)		
	Phosphonate monoester hydrolase	Phosphate diester or phosphonate monoester hydrolysis	(65)	(65)	(65)
	Glycerol phosphate lipoteichoic acid synthase	Transfer of glycerol phosphate from phosphatidylglycerol to polyphosphoglycerol			
	Acid phosphatase	Phosphate monoester hydrolysis			
Two- or three-metal	Alkaline phosphatase	Phosphate monoester hydrolysis		(66)	(67) (68)
	Nucleotide pyrophosphatase/ phosphodiesterase	Phosphate diester hydrolysis	(37)		(69)
	Phosphonoacetate hydrolase	Hydrolysis of phosphonoacetate (P-C bond cleavage)	(70)		(70)
	Phosphopentomutase	Phosphoryl transfer from monoester substrate			
	Phosphoglycerate mutase	Phosphoryl transfer from monoester substrate	(41)		
	Minimal diesterases	Phosphate diester hydrolysis			
	NPP-like monoesterases	Phosphate monoester hydrolysis			

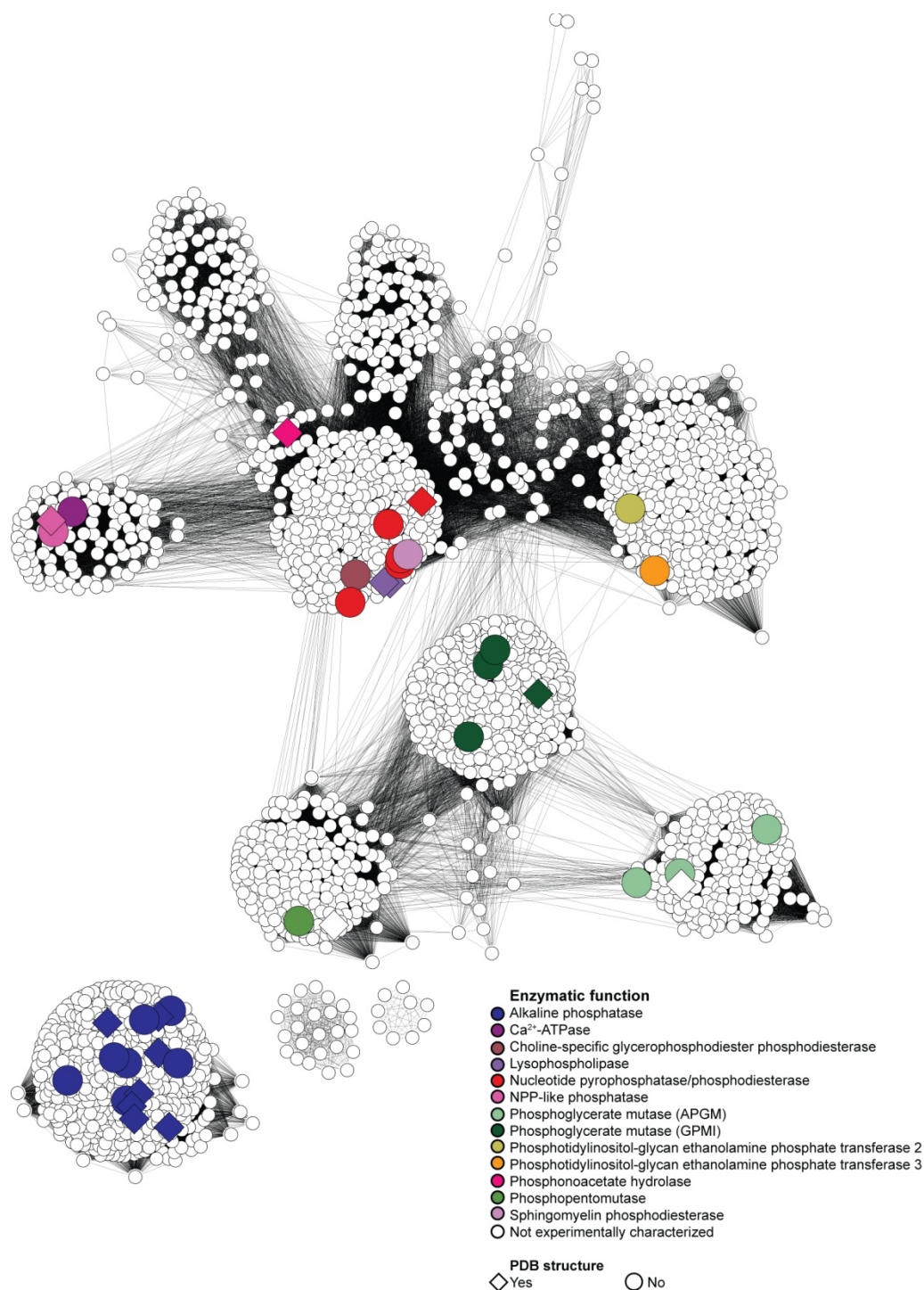


Figure D.1. Sequence similarity network of the 2/3 metal subgroup of the AP superfamily colored by experimentally determined functions. A representative set of 2544 nodes was selected from the 6790-protein subgroup where each node represents a set of proteins with >70% identity. Edges between nodes are drawn only if the mean similarity of edge weights $-\log_{10}(\text{E-value}) > 0$ is $-\log_{10}(\text{E-value}) > 10$ (median alignment length = 236 residues; median percent

identity of pairwise comparisons = 28.3%). The network is visualized using the organic layout in Cytoscape. While the lengths of connecting edges tend to correlate with the relative dissimilarities of each pair of sequences, these distances do not represent a quantitative correlation. Coloring is based as indicated color function key if any protein in a representative node has been experimentally characterized with the given function. If any protein in a representative node has a structure available in the PDB, it is indicated as a diamond shape. Note that AP enzymes do not connect to other two-metal-ion members of the superfamily with an E value of $< 1 \times 10^{-10}$, so they appear disconnected from the other enzymes.

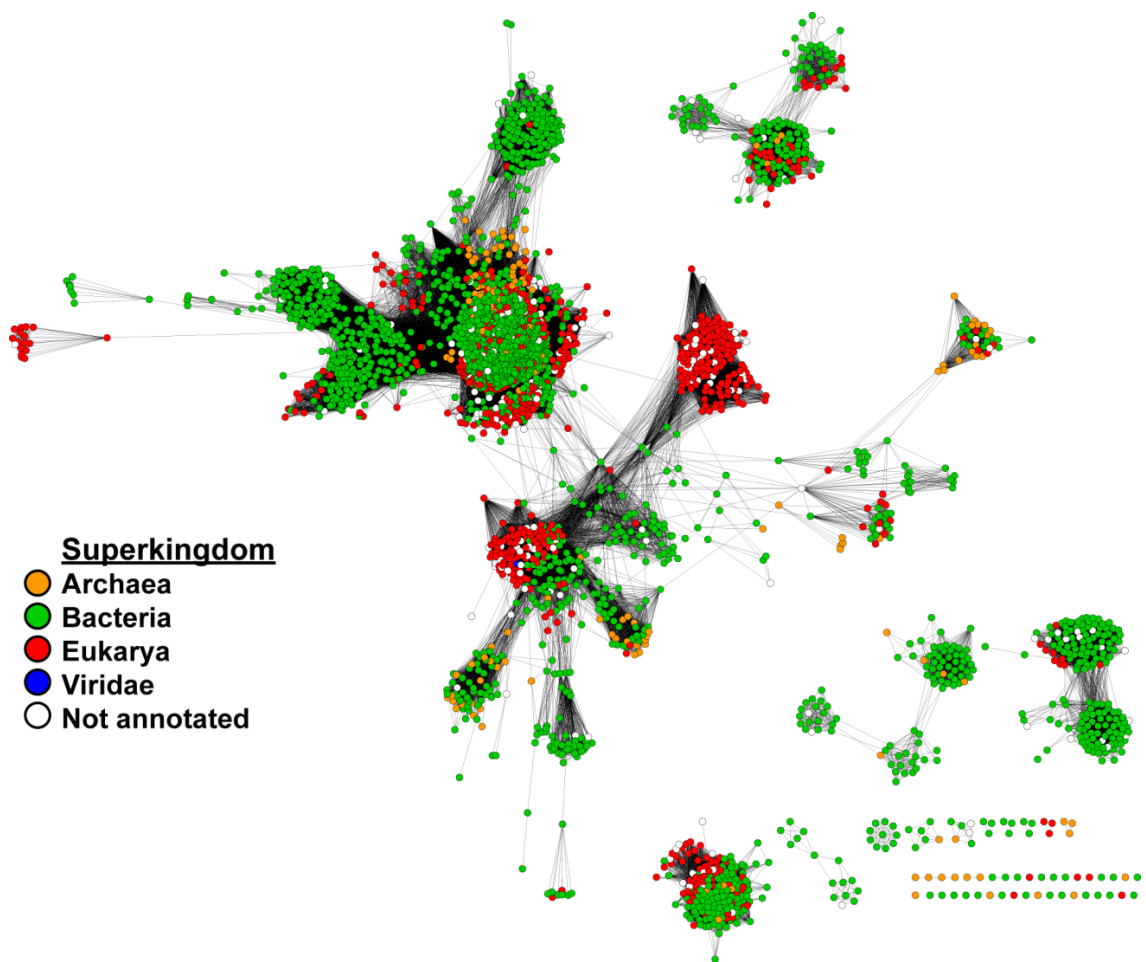


Figure D.2 Sequence similarity network of the AP superfamily with superkingdom indicated. Edges between nodes are drawn only if the maximum similarity of edge weights greater than $-\log_{10}(\text{E-value}) > 13.1$ (median alignment length = 422 residues; median percent identity of pairwise comparisons = 30%). The network is visualized using the organic layout in Cytoscape. Coloring is based as indicated in the color key as to what all superkingdom all sequences represented by any node are part of if they have a superkingdom annotation in UniProt. In white nodes, none of the represented sequences have a superkingdom annotation in UniProt. This network demonstrates that the major topological divisions within the superfamily contain sequences from all superkingdoms demonstrating the ancient nature of these divisions.

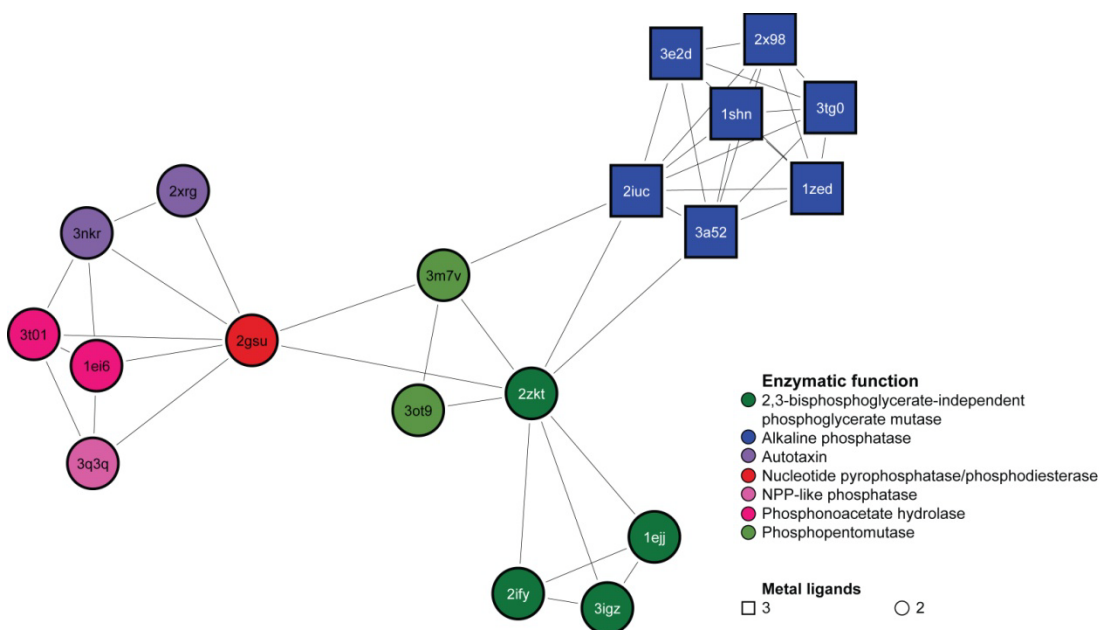


Figure D.3. Structure similarity network of the 2 and 3 metal AP superfamily enzymes colored by experimentally determined function. Edges between nodes are drawn only if the maximum similarity of edge weights greater than FAST N-score > 10 (median alignment length = 208 residues; median RMSD = 3.42Å). The network is visualized using the organic layout in Cytoscape. Coloring is based as indicated in the color key as the experimentally determined function from appendix D, table D.2, where APGM and GMPI are grouped together as 2,3-bisphosphoglycerate-independent phosphoglycerate mutase. This network demonstrates that with respect to the alignment of the core domain, which is the alignment that the edge scores represent, the mutases are intermediate in structure between the NPP-like enzymes and the 3-metal APs.

Table D.2. Experimentally characterized two/three metal subgroup enzymes.

Function annotation	ID	Literature reference (Pubmed ID)	DOI (if no Pubmed ID)
AP	B0R9W3	20438737	
AP	P00634	3910843;2010919;7577993;8652582; 9253408;9533886;10085061	
AP	P19147	2129542	
AP	B5BP20	15973058	
AP	Q1W622	15906116	
AP	Q93P54	11566372;18977465	
AP	E8Z5C8	20575745	
AP	P05186	3532105;11124260	
AP	P15693	2155025	
AP	P19111	8452539	
AP	Q9BHT8	12083516;15333925	
AP	Q8GJK0	12928867	
AP	Q8RQU7	12036047;20057143	
AP	Q9XBW0	10482522	
AP	Q153J0	17989875	
AP	Q9RA56	10556703	
		10672035;17198711	
AP	Q9KWY4	19916164	
AP	P11491	3319783	
APGM	Q59007	12062435	
APGM	O28523	17576516	
APGM	Q60326	12062435	
Ca ²⁺ ATPase/ monoesterase	Q47910	8617788	
Choline phosphodiesterase	Q6UWR7	15788404	
Choline phosphodiesterase	Q8BGN3	15788404	
GPMI	O44742	15234973	
GPMI	P35493	8260624	
GPMI	P35494	8260624	
GPMI	D5DVF4	10388626	
GPMI	P37689	10437801	
GPMI	Q4VWF8	15234973	
GPMI	Q9X519	10388626	
GPMI	Q81X77	17085493	
Lysophospholipase	Q64610	12119361;12633853;15985467;21240271	
Lysophospholipase	Q9R1E6	21240269	
Lysophospholipase	Q9UCR2	21240271	
Lysophospholipase	A1A4K5	12119361	
NPP	Q8PIS1	16893180	
NPP	D9YT79		10.1016/j.jcs.2010.01.009

NPP	P39997	16278456	
NPP	P06802	8223581	
NPP	P22413	8001561	
NPP	P97675	7730366;9096610;11069764	
NPP	P15396	2989287	
NPP-like monoesterase	A1YYW7	18641147	
NPP-like monoesterase	Q60113	7875572	
Phosphonoacetate hydrolase	Q51782	21366328	
Phosphonoacetate hydrolase	Q92UV8	22035792	
Phosphopentomutase	O24821	9692190	
Phosphopentomutase	Q818Z9	22329805	
PIGG	Q5H8A4	15632136	
PIGO	Q9JJI6	10781593	
Sphingomylinase	Q6UWV6	12885774	



Figure D.4. Sequence similarity network of the 2/3 metal subgroup of the AP superfamily with NPP-like homolog set indicated. Network is composed and layed out in the same fashion as appendix D, figure D.1. The sequences colored in red cluster seperately together at an $-\log(\text{E-value}) > 60$ and were chosen as a set of close homologs of NPP where several functions are known to

perform a co-variation analysis. The co-variation analysis of enzymes from this group in SwissProt/TrEMBL is presented in appendix D, table D.3.

Table D.3. Motif comparison analysis of NPP-like homologs

	P91	P94	P123	P155	P160	P174	P205	Match Motif	Function
Q5BKW7	Y	Y	L	Y	E	N	Y	YYXYEXY	Choline phosphodiesterase
F1QVT8	Y	Y	L	Y	E	N	Y	YYXYEXY	Choline phosphodiesterase
Q8BGN3	Y	Y	D	Y	E	V	Y	YYXYEXY	Choline phosphodiesterase
D3ZRI1	Y	Y	D	Y	E	V	Y	YYXYEXY	Choline phosphodiesterase
B0BND0	Y	Y	D	Y	E	V	Y	YYXYEXY	Choline phosphodiesterase
Q58D68	Y	Y	D	Y	E	V	Y	YYXYEXY	Choline phosphodiesterase
Q6UWR7	Y	Y	D	Y	E	V	Y	YYXYEXY	Choline phosphodiesterase
Q5RB45	Y	Y	D	Y	E	V	Y	YYXYEXY	Choline phosphodiesterase
F1NUU6	Y	Y	D	Y	E	V	Y	YYXYEXY	Choline phosphodiesterase
F1N5C8	Y	Y	D	Y	E	V	Y	YYXYEXY	Choline phosphodiesterase
D3YU63	Y	Y	D	Y	E	V	Y	YYXYEXY	Choline phosphodiesterase
F1PZF9	Y	Y	D	Y	E	V	Y	YYXYEXY	Choline phosphodiesterase
D2GY61	Y	Y	D	Y	E	V	Y	YYXYEXY	Choline phosphodiesterase
Q6DDP3	Y	Y	D	Y	E	Y	Y	YYXYEXY	Choline phosphodiesterase
D3TZC6	Y	Y	N	Y	E	Y	Y	YYXYEXY	Choline phosphodiesterase
B0WJ29	F	H	B	M	D	F	Y	XHXM(X(F/Y/W)Y	Unknown
E3WQV5	F	H	B	M	D	F	Y	XHXM(X(F/Y/W)Y	Unknown
A7US29	F	H	B	M	D	F	Y	XHXM(X(F/Y/W)Y	Unknown
E0VI07	F	H	B	M	E	W	Y	XHXM(X(F/Y/W)Y	Unknown
D7EJX0	F	H	B	M	C	Y	Y	XHXM(X(F/Y/W)Y	Unknown
Q0ZSS6	F	H	B	M	D	Y	Y	XHXM(X(F/Y/W)Y	Unknown
Q16VP0	F	H	B	M	N	Y	Y	XHXM(X(F/Y/W)Y	Unknown

Q06K70	F	H	B	M	N	Y	Y	XHXXMX(F/Y/W)Y	Unknown
C3XTL2	D	H	W	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
C3ZM60	F	H	H	M	A	Y	Y	XHXXMX(F/Y/W)Y	Unknown
E2AZ43	F	H	K	M	A	F	Y	XHXXMX(F/Y/W)Y	Unknown
Q566N0	F	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
B5X3I9	F	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
Q4RZ37	F	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
Q0VA77	F	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
F1LXV5	F	H	S	M	D	B	B	XHXXMX(F/Y/W)Y	Unknown
D3YYS8	F	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
Q8BTJ4	F	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
Q9Y6X5	F	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
Q5RAC0	F	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
D2HSX0	F	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
F1RQR7	F	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
A2VDP5	F	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
C3ZM54	F	H	H	M	E	Y	Y	XHXXMX(F/Y/W)Y	Unknown
E2BDY2	F	H	K	M	I	F	Y	XHXXMX(F/Y/W)Y	Unknown
C3ZM61	F	H	H	M	Q	Y	Y	XHXXMX(F/Y/W)Y	Unknown
E9J4S8	F	H	K	M	V	F	Y	XHXXMX(F/Y/W)Y	Unknown
F1R4L7	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
D8KVZ9	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
P84039	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
Q9EQG7	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
Q5DTL0	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
F1N5W4	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
Q5NVN5	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
A8K9X7	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
Q9UJA9	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
F1RQQ6	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
D2HSX1	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown

F1P9H2	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
F1PE50	Y	H	S	M	D	Y	Y	XHXXMX(F/Y/W)Y	Unknown
F1N1R7	S	H	P	F	N	F	Y	SHXXXXY	Sphingomylinase
D3TZC2	S	H	S	F	N	W	Y	SHXXXXY	Sphingomylinase
D3TZC3	S	H	P	F	N	W	Y	SHXXXXY	Sphingomylinase
E2RFT5	S	H	P	F	N	Y	Y	SHXXXXY	Sphingomylinase
F1RZ75	S	H	P	F	N	Y	Y	SHXXXXY	Sphingomylinase
Q6UWV6	S	H	P	F	N	Y	Y	SHXXXXY	Sphingomylinase
Q5EZ72	S	H	P	F	N	Y	Y	SHXXXXY	Sphingomylinase
Q3TIW9	S	H	P	F	N	Y	Y	SHXXXXY	Sphingomylinase
A2A5N7	S	H	P	F	N	Y	Y	SHXXXXY	Sphingomylinase
B7ZWB9	S	H	P	F	N	Y	Y	SHXXXXY	Sphingomylinase
D3Z6V6	S	H	P	F	N	Y	Y	SHXXXXY	Sphingomylinase
C3ZE19	G	H	D	F	S	T	Y	SHXXXXY	Sphingomylinase
Q08C10	S	H	Q	H	A	H	Y	SHXXXXY	Unknown
F1QAC6	S	H	Q	H	A	H	Y	SHXXXXY	Sphingomylinase
Q6ZM30	S	H	Q	H	A	H	Y	SHXXXXY	Sphingomylinase
Q4SAU7	S	H	P	H	A	Y	Y	SHXXXXY	Sphingomylinase
Q6ZM31	S	H	E	H	A	Y	Y	SHXXXXY	Sphingomylinase
A8WGC5	S	H	E	H	A	Y	Y	SHXXXXY	Sphingomylinase
F1PX55	S	H	Y	H	G	D	Y	SHXXXXY	Sphingomylinase
D2HQ64	S	H	S	H	G	D	Y	SHXXXXY	Sphingomylinase
E2R267	S	H	S	H	G	D	Y	SHXXXXY	Sphingomylinase
E1B9U6	S	H	S	H	G	N	Y	SHXXXXY	Sphingomylinase
E1BTU5	S	H	P	H	K	Y	Y	SHXXXXY	Sphingomylinase
Q1MSY7	S	H	P	Y	G	D	Y	SHXXXXY	Sphingomylinase
Q0P3U7	S	H	P	Y	G	D	Y	SHXXXXY	Sphingomylinase
Q15WV6	F	F	B	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q0ANG7	F	F	N	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A3UF17	F	F	V	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B4SPN5	F	H	L	S	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP

B8L390	F	H	L	S	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F0BNV5	F	H	L	S	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q3BR97	F	H	L	S	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D4ST33	F	H	L	S	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q8PIS1	F	H	L	S	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q4UWP2	F	H	L	S	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B0RQX2	F	H	L	S	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F0BL44	F	H	L	S	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B3S3W2	F	H	I	Y	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F4QRJ4	F	H	M	F	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
O69013	F	H	M	F	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C8WEL0	F	H	M	F	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B8H4G4	F	H	M	F	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B4W9U0	F	H	M	F	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D4Z1Z1	F	H	M	F	N	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q1NE09	F	H	M	F	N	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q0BY07	F	H	B	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B3S3W0	F	H	I	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D5HCA9	F	H	L	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q2RZL1	F	H	L	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D0ME80	F	H	L	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B4R8P3	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D9QGN8	F	H	Q	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q4SH02	F	H	V	M	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q6AX80	F	H	M	M	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F1N9T9	F	H	Q	M	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D8KVZ8	F	H	Q	M	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F1P9H3	F	H	D	M	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D3ZWW8	F	H	E	M	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D4A2W1	F	H	E	M	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D3ZY33	F	H	E	M	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP

F1LTZ5	F	H	E	M	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A5P7D4	F	H	M	F	A	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E6WU49	F	H	V	A	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E8RLN6	F	H	L	F	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F4QZD9	F	H	L	F	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F3X317	F	H	N	F	N	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B7QDE2	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B2FHJ5	F	H	L	A	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F4AR16	F	F	B	Y	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B0T2F1	Y	H	M	F	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D5VG07	Y	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F4C295	F	H	M	Y	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B2RHB3	F	H	I	F	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q7MXP5	F	H	I	F	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B7BE80	F	H	M	Y	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E1Z0A5	F	H	M	Y	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D0THL0	F	H	M	Y	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A6LEB9	F	H	M	Y	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C7XBR4	F	H	M	Y	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F0QZ55	F	H	M	Y	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B5CZC1	F	H	M	Y	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F3ZPD7	F	H	M	Y	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E4VQK3	F	H	M	Y	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C6I8E4	F	H	M	Y	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D1JN95	F	H	M	Y	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E6SU48	F	H	M	Y	D	W	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B0NPH5	F	H	M	Y	D	W	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F3PFL1	F	H	M	Y	D	W	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D1PYL3	F	H	L	Y	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E6MTU3	F	H	L	H	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D1QUI0	F	H	L	H	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP

D7N9J6	F	H	L	H	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q5QYI9	F	H	M	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A3WKU6	F	H	M	F	G	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	Unknown
F2G994	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A4CAP7	F	H	M	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F3BHH7	F	H	M	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A0Y3E8	F	H	M	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q3ICX1	F	H	M	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E6RQE9	F	H	M	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E4RTP9	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E4TDC0	G	H	F	M	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D7VZ02	F	H	M	M	A	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	Unknown
E7F2C4	Y	L	F	F	B	B	H	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
Q6PGY9	Y	L	F	F	B	B	H	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
Q7ZXN7	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
B2GU60	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
Q5HZ84	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
Q2TAH6	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
D3ZES5	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
F1LSF4	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
E1BY59	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
A1A4K5	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
F1S280	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
F1PST7	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
E2QT26	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
D2I2V8	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
Q5R6E5	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
E9PHP7	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
Q13822	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
E7EUF1	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
Q64610	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase

Q9R1E6	F	L	L	F	B	B	Y	(Y/F)L(F/L)FBB(Y/H)	Lysophospholipase
B0JZL7	F	H	L	F	D	W	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q5RI82	F	H	L	F	D	W	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
P90754	F	H	S	F	A	Y	Y	FHSFAYY	Unknown
A8Y413	F	H	S	F	A	Y	Y	FHSFAYY	Unknown
E3LQ00	F	H	S	F	A	Y	Y	FHSFAYY	Unknown
E4X1N2	F	H	A	L	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E4XW05	F	H	A	L	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E4XW08	F	H	A	L	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E4X1C1	F	H	A	L	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E4YIQ7	F	H	A	L	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E5SLP6	F	H	F	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E9FS92	F	H	F	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q4SZU4	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q6DYE8	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
P97675	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
P15396	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F1S3P3	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E2R4D7	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D2HL23	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E1C1S1	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D3TZC1	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D3TZC4	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D3TZC5	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D3TZC0	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F1NC25	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E1C1R0	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
P06802	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q924C3	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B3EX52	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F1MNS5	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP

A2T3U8	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F1S3P4	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F1S3P5	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C1FXT3	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B7NZB8	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B1MTF2	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A9X179	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
P22413	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B2RBY8	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D2HL24	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F1PJP0	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q4RQY0	F	H	L	F	D	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E4X3H5	F	H	V	L	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D8RQY1	F	H	F	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D8RQX9	F	H	F	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D8RWV5	F	H	F	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A9SYQ1	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D8RQY2	F	H	F	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D8RWV6	F	H	F	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D7MCI9	F	H	B	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q9SU82	F	H	B	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q84WJ3	F	H	B	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D7MCI7	F	H	B	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D7MCI8	F	H	B	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q9SU81	F	H	B	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C6TGW4	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B8ADR2	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q7F5L2	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D9YT79	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F2CV66	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C5XLV5	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP

B6SGJ5	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q9FS13	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D7MCJ0	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q9SU83	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B9RK58	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B9H8W6	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B9N5F9	F	H	M	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F1L2V1	F	H	K	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E1FVY3	F	H	K	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A8PTF0	F	H	K	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q5R5M5	F	H	L	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A8KA38	F	H	L	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q308M7	F	H	L	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
O14638	F	H	L	F	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q54MJ8	F	H	I	F	G	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	Unknown
E1FIJ3	F	H	I	F	N	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F1KT55	F	H	Y	F	S	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	Unknown
E3LQ02	F	H	A	F	Y	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	Unknown
B6QJU9	F	H	A	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E9F9X2	F	H	H	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E9E4V6	F	H	H	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q6C6J4	F	H	N	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E7R107	F	H	N	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C4Y7Y0	F	H	N	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A5E4H0	F	H	N	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
O94323	F	H	N	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C4QWZ9	F	H	N	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A5DKT7	F	H	N	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q6BTN4	F	H	N	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A3LS92	F	H	N	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C5MCE9	F	H	N	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP

B9WGP4	F	H	N	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q59Q81	F	H	N	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C4YGP5	F	H	N	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B6JWD1	F	H	N	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F4R3T2	F	H	N	M	P	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E3L1R4	F	H	N	M	P	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E3KMZ8	F	H	N	M	P	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B8MRV3	F	H	R	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D5G9X4	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F0XK17	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q2H4Z5	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C7YMK3	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E3Q726	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C4JHL4	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A1C767	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C5FYR6	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E4UVC8	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A6RF96	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C6HR75	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F0U720	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C0P0U9	F	H	Y	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E4ZX38	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q0UVG6	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B2WCX1	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E3RLA2	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B2AT13	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q7SI20	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q2KGE6	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A6S8V5	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A7F1D6	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C9SAX8	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP

C5P370	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
E9CUJ7	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B6GYR8	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A2R5D3	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B8NFZ8	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
Q4X044	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A1DI35	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F2Q625	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F2SBA1	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F2SG30	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D4D4G0	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
D4B5B3	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C5GVW3	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
F2T4L3	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C5JBU1	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C1GT30	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C0S7C0	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C1FZR6	F	H	Y	M	E	Y	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A8NRI7	F	H	Y	M	P	W	Y	FHYMP(W/Y)Y	Unknown
D8PM24	F	H	Y	M	P	W	Y	FHYMP(W/Y)Y	Unknown
E7A092	F	H	Y	M	P	Y	Y	FHYMP(W/Y)Y	Unknown
Q4PBI2	F	H	Y	M	P	Y	Y	FHYMP(W/Y)Y	Unknown
E6R2G5	F	H	Y	M	P	Y	Y	FHYMP(W/Y)Y	Unknown
F5HCS1	F	H	Y	M	P	Y	Y	FHYMP(W/Y)Y	Unknown
A6ZQT4	F	H	B	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C8Z6Y4	F	H	B	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
P39997	F	H	B	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B3LS01	F	H	B	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
B5VHC1	F	H	B	M	E	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A7TLN6	F	H	B	M	F	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	Unknown
P25353	F	H	B	M	D	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP

B5VEX3	F	H	B	M	D	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
C7GW70	F	H	B	M	D	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A6ZTK7	F	H	B	M	D	F	Y	(F/Y)(H/F)XX(E/N/D)(F/Y/W)Y	NPP
A8Y412	F	H	L	F	Y	Y	Y	(F/M)HX(F/L)XXY	Unknown
E3W744	F	H	L	F	Y	Y	Y	(F/M)HX(F/L)XXY	Unknown
P90755	F	H	L	F	Y	Y	Y	(F/M)HX(F/L)XXY	Unknown
F0ZWD1	F	H	N	F	G	F	Y	(F/M)HX(F/L)XXY	Unknown
F0ZWD6	F	H	N	F	S	F	Y	(F/M)HX(F/L)XXY	Unknown
F0ZWD7	F	H	S	F	G	F	Y	(F/M)HX(F/L)XXY	Unknown
D8RQW8	F	H	T	F	Q	Y	Y	(F/M)HX(F/L)XXY	Unknown
D8SYU0	M	H	R	L	E	L	Y	(F/M)HX(F/L)XXY	Unknown
D8S717	M	H	R	L	E	L	Y	(F/M)HX(F/L)XXY	Unknown

Table D.5. Data for identifying leaves for 2/3 metal subgroup tree. Leaves are listed in counter-clockwise order from the specified starting point.

3-metal AP

D5BI89
E4TFF2
E8PNE8
C3JAY9
D8FJU1
C1H7N9
UPI0001F2A152
D0NY52
A5G5J3
E0VGW0
D0SAJ7
Q605T9
E7H3S0
F4P5X7
A0ZGF8
F3ZAI7
A0Z6L8
B1MEC3
B2FJM8
C7CF10
A3ITD9
D7NBY3
A3XKX3
E4TDM9
F4CCK5
A9BFF3
C6BVD8
Q3B154
F4XDL9
B2KE24
E6HE00
B0R9W3

Mutases

(GPML clade)

A0SNX3

C6LQC2

D3SCD0

E1Q077

F0S308

E9D9U0

B1RW85

A2EEX5

(DEOB clade)

C3WF29

UPI0001C1C9C8

Q67S06

D4XV55

E3EBD3

E6PD76

F2JS49

F2JCI5

D3FCZ8

(APGM clade)

B9N1Y2

D0A920

A1S0I2

Q6SI04

A8RRP2

D7DTG5

D3L3F4

F4CQB8

B8DYG6

A2BM20

D2EEB2

D9PEW1

A5FPB9
A5KKM8
Q8PX04
UPI00020A977F
B8DRS5
E8LBL6
F0SZ32
B1QX76
C3NGG9
A6VH25

Minimal scaffold enzymes

(PIGN clade)

E5S0J1
F4JD48
B4PRR2

(Unknown minimal clade 1)

E4W032
D1QTA0
C6W6G8
D2BB38
E4N9G0
D3D2P1
D3PX43
E9T5U0
UPI00020D3C93
D7VUJ6
F3P460
D3PX84
A4APF6

(PIGO & PIGG containing clade)

B8J0M3

Q4HI40
C6IYI2
D0CVZ4
B7G380
A4S7V6
A0C960
A0DP71
Q4QAU7
B6AB65
F4Q2L9
F4P2J9
Q7RN05
C4LXX0
F0W2V9
A1D5E7
D2UZH6
B4LLD0
UPI000206236A
E4M3K2

NPP-like

(phosphonoacetate hydrolase and 2-metal monoesterase clade)

A9G6L6
A1WGW5
A6GJN3
Q60113
D0LQ60
Q1D3Z5
B7VSQ3
E1GVE8
C6X263
A4CH98
D2R876
E8UX69
Q1IUQ5

(Unknown NPP-like clade 1)

UPI000038E241

F4B9D9

D9Q0N7

UPI000038E442

Q9HIJ0

A6G7L2

Q0W935

E1IFX6

A1K612

B9Z574

Q0A4W7

D3RQ95

D3Q9L8

D4YNL4

B0RC17

C5CCK9

F1YNJ6

B7GQI6

C2KQ72

E4CTB9

C7MCH0

C7R4L0

F3NNU4

C7NI94

(Unknown NPP-like clade 2)

Q1AUG9

D3F3W3

A6CBE2

F0DJV9

B0TGQ9

E6TW64

C0ZBA2

(NPP and NPP-like clade)

UPI000180CE4E

Q6ZM31
F1PJP0
F1PJP0
B9RK58
E0VI07
Q06K70
UPI0002061467
E4MAJ6
F3PFL1
E4TDC0
F0ZWD6
B0RQX2
UPI00016E187B
C3ZMZ5
UPI0001CB9A5F
C4QCY6
UPI00016E692F
UPI000180CDFF
UPI000192417B
E3N3Z7
Q0BY07
D2V3Q0
C5E478
B8MRV3
F4R3T2
Q4PBI2
D2V7Q3
E3LQ00
Q2RZL1
UPI0001925120
UPI0001926A77
A7S6K6

(Unknown NPP-like clade 3)

C7PFC1
F3PDX0
F1ZD69

A8RSY5
Q023T8
D5SRM6
UPI0001744497
B9XGV6
E8X061
B0SHK9
E6TU07
F2F4Z5
F0HC33
C0CU39
Q4MQF2
Q03Y80
D4YEF9
D9R3E7
C5C383
D3AEN3
UPI00020ED294
B9Y8B9
D4W691
Q180R2

(Unknown NPP-like clade 4)

Q1PUQ8
D3SSH3
Q9X5R9
D2L164
Q026V4
E8R519
E8R368
D2L481
D7D8A8
Q9YCS9
E1QLZ0
B8DQ47
Q9X5S1
B8CYV9

B0R301
A2BMW3
Q9YCR2
Q1Q787
A7NRY7
A1RUN4
B8FMN0
D0LI00
A3ZY21
E8QX53
D2S3E0
B8CYV7
F0LLW7
F2KQE4
F4G167
E8R8D8

(Unknown NPP-like clade 5)
B6YT22

Appendix E. Supplemental information for chapter 4

Table E.1. Kinases that are cleaved by caspases. Literature references are given for those cleavages where the functional consequence of cleavage is known. Bold identifiers have been experimentally characterized. Black identifiers have known cleavage sites, red identifiers are cleaved, but the exact site is unknown.

	# of caspase cleaves	Functionally characterized	Reference (Pubmed ID)
ABL1_HUMAN	1	X	
AKT1_HUMAN	1		
ARBK1_HUMAN	2		
ATM_HUMAN	1	X	
BCR_HUMAN	1		
BRAF_HUMAN	1		
BRD4_HUMAN	1		
BUB1_HUMAN	1	X	
BUB1B_HUMAN	2	X	16227576
CAMKV_HUMAN	1		
CD11B_HUMAN	1	X	
CDK13_HUMAN	1		
CDK14_HUMAN	1		
CDK6_HUMAN	2		
CHK1_HUMAN	1		
E2AK2_HUMAN	1	X	11555640
E2AK3_HUMAN	1		
EGFR_HUMAN	2	X	11226410;16247443
ERBB2_HUMAN	2		11402024;15087380
FAK1_HUMAN	1	X	8700528;9642276
FAK2_HUMAN	1		
FYN_HUMAN	1	X	
GAK_HUMAN	1		
GRK6_HUMAN	1		
HIPK2_HUMAN	1	X	16601678
KCC2G_HUMAN	1		
KKCC1_HUMAN	1	X	
KKCC2_HUMAN	1		
KPCD_HUMAN	1	X	8557034
KPCD1_HUMAN	1	X	10622742
KPCE_HUMAN	1	X	9431985
KPCL_HUMAN	1	X	10409750
KPCT_HUMAN	1	X	9252332
KPCZ_HUMAN	2	X	10196149

KS6A4_HUMAN	1		
LMTK2_HUMAN	2		
LYN_HUMAN	1	X	11526478
M3K1_HUMAN	1	X	9244305;16427076
M3K13_HUMAN	1		
M3K5_HUMAN	1		
M4K1_HUMAN	1	X	16427076
MARK1_HUMAN	1		
MINK1_HUMAN	2		
MK07_HUMAN	1		
MP2K1_HUMAN	2	X	
MST4_HUMAN	1	X	11741893
MTOR_HUMAN	2		
NEK1_HUMAN	1		
NEK4_HUMAN	2		
PAK2_HUMAN	1	X	
PKN1_HUMAN	1	X	9751706
PKN2_HUMAN	1	X	10926925
PRKDC_HUMAN	3	X	
PRPK_HUMAN	1		
RAF1_HUMAN	1	X	
RET_HUMAN	2	X	10921886
RIOK1_HUMAN	1		
RIPK1_HUMAN	1	X	
RIPK3_HUMAN	1		
ROCK1_HUMAN	1	X	
ROCK2_HUMAN	2		
SFRS9_HUMAN	1	X	10611247
SLK_HUMAN	1		
SRC_HUMAN	1	X	
SRPK1_HUMAN	1	X	
SRPK2_HUMAN	1	X	12417631
STK10_HUMAN	3		
STK24_HUMAN	1	X	
STK39_HUMAN	1	X	
STK3_HUMAN	1	X	9662336
STK4_HUMAN	1	X	
TIF1A_HUMAN	1		
TIF1B_HUMAN	6		
TLK2_HUMAN	1		
TNIK_HUMAN	1		
TRI33_HUMAN	1		

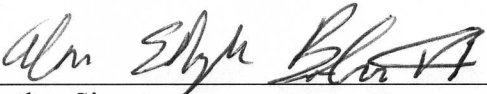
VRK1_HUMAN	1		
WEE1_HUMAN	1	X	9618490
WNK1_HUMAN	3		
ZAP70_HUMAN	1		

Publishing Agreement

It is the policy of the University to encourage the distribution of all theses, dissertations, and manuscripts. Copies of all UCSF theses, dissertations, and manuscripts will be routed to the library via the Graduate Division. The library will make all theses, dissertations, and manuscripts accessible to the public and will preserve these to the best of their abilities, in perpetuity.

Please sign the following statement:

I hereby grant permission to the Graduate Division of the University of California, San Francisco to release copies of my thesis, dissertation, or manuscript to the Campus Library to provide access and preservation, in whole or in part, in perpetuity.



Author Signature

9/12/2012

Date