Utriainen and Morris *BMC Bioinformatics* (2023) 24:134
https://doi.org/10.1186/s12859-023-05225-z

BMC Bioinformatics

SOFTWARE **Open Access**

clusterMaker2: a major update to *clusterMaker*, a multi-algorithm clustering app for Cytoscape

Maija Utriainen[1] and John H. Morris[2*]

*Correspondence:
scooter@cgl.ucsf.edu

[1] Maastricht University, Maastricht, NL, USA
[2] Department of Pharmaceutical Chemistry, University of California San Francisco, San Francisco, CA, USA

Abstract

Background: Since the initial publication of *clusterMaker*, the need for tools to analyze large biological datasets has only increased. New datasets are significantly larger than a decade ago, and new experimental techniques such as single-cell transcriptomics continue to drive the need for clustering or classification techniques to focus on portions of datasets of interest. While many libraries and packages exist that implement various algorithms, there remains the need for clustering packages that are easy to use, integrated with visualization of the results, and integrated with other commonly used tools for biological data analysis. *clusterMaker2* has added several new algorithms, including two entirely new categories of analyses: node ranking and dimensionality reduction. Furthermore, many of the new algorithms have been implemented using the Cytoscape *jobs* API, which provides a mechanism for executing remote jobs from within Cytoscape. Together, these advances facilitate meaningful analyses of modern biological datasets despite their ever-increasing size and complexity.

Results: The use of *clusterMaker2* is exemplified by reanalyzing the yeast heat shock expression experiment that was included in our original paper; however, here we explored this dataset in significantly more detail. Combining this dataset with the yeast protein–protein interaction network from STRING, we were able to perform a variety of analyses and visualizations from within *clusterMaker2*, including Leiden clustering to break the entire network into smaller clusters, hierarchical clustering to look at the overall expression dataset, dimensionality reduction using UMAP to find correlations between our hierarchical visualization and the UMAP plot, fuzzy clustering, and cluster ranking. Using these techniques, we were able to explore the highest-ranking cluster and determine that it represents a strong contender for proteins working together in response to heat shock. We found a series of clusters that, when re-explored as fuzzy clusters, provide a better presentation of mitochondrial processes.

Conclusions: *clusterMaker2* represents a significant advance over the previously published version, and most importantly, provides an easy-to-use tool to perform clustering and to visualize clusters within the Cytoscape network context. The new algorithms should be welcome to the large population of Cytoscape users, particularly the new dimensionality reduction and fuzzy clustering techniques.

Keywords: Clustering, Community detection, Network analysis, Visualization, Cytoscape

Background

High-throughput techniques to generate large proteomic, genomic, metabolomic and interactome datasets provide a wealth of information about basic biological processes as well as human diseases. More recently, with the advent of single-cell transcriptomics and spatially resolved proteomics, the size and complexity of these datasets have exploded. One of the techniques historically used for the analysis of high-throughput biological data has been clustering to categorize large numbers of data points into significantly smaller numbers of groups, where all of the members of the groups are similar or have similar features or behaviors. While many libraries and packages exist that implement various algorithms, there remains the need for clustering packages that are easy to use, integrated with visualization of the results, and integrated with other commonly used tools for biological data analysis.

Clustering algorithms have been used for over 2 decades to analyze microarray data [1], find complexes in protein–protein interaction data [2, 3], and more recently, to categorize single-cell data into cell types [4–6]. The goal of clustering algorithms is to group similar data together, but the definition of similarity may depend on the specific use case. For example, traditional hierarchical clustering of microarray data looks for similarity in expression patterns, while clustering of protein–protein interaction networks aims to group nodes based on how closely connected they are. Below, we discuss these two major approaches to clustering algorithms as well as dimensionality reduction and ranking approaches, with a focus on their application to nodes and edges in a network.

Attribute clustering algorithms

Attribute clustering algorithms group nodes based on the similarity of the attributes of the nodes or a distance metric calculated using an edge weight. Examples of attribute clustering algorithms include hierarchical [1], k-means [7, 8], HOPACH [9], PAM [10], AutoSome [11], and Transitivity Clustering [12]. The results of these algorithms are generally presented as heatmaps, or dendrograms associated with heat maps (e.g., for hierarchical clusterings). The categories may be used to group nodes in a network context, but often they are used primarily as visualization aids since the groupings may be independent of the network topology.

Network clustering algorithms

Network clustering algorithms find densely connected regions in a network. Examples of network clustering algorithms include AutoSOME [11], Affinity Propagation [13], Connected Components, GLay [14], MCODE [15], MCL [16, 17], SCPS [18], Transitivity Clustering [12], Leiden [19], Infomap [20], Fast Greedy [21], Leading Eigenvector [22], Label Propagation [23] and Multilevel [24]. Generally, these algorithms are used to break the network up into smaller groups, so rather than using heatmaps to visualize the results, the typical visualization is a clustered network, with multiple disconnected components.

Clustering algorithms such as MCL and Leiden are examples of discrete clustering. That is, the algorithm will place a node either in one group or another based on the parameters and the specifics of the algorithm. However, sometimes nodes are not clearly in one group or another, and, for example, might be strongly associated with more than one cluster. This can lead to an overinterpretation of the clustering results. Fuzzy clustering algorithms allow nodes to be a member of more than one cluster, where membership is a proportional value. For example, the fuzzy C-Means (FCM) algorithm [25, 26], which is (roughly speaking) the fuzzy analog to the k-means clustering algorithm, iteratively calculates cluster centroids and assigns nodes to clusters, repeating until some convergence criterion is reached.

Dimensionality reduction algorithms

Dimensionality reduction techniques are used to reduce the number of input variables, that is, the dimensionality, in a dataset. The higher the dimensionality, the more complicated the analysis and modeling of the dataset becomes. Reducing the number of input variables is important, as the performance of algorithms like clustering and ranking can degrade with high dimensionality. Dimensionality reduction is a common technique and has been used for many years. Algorithms such as principal component analysis (PCA) [27], principal coordinate analysis (PCoA) [28], and multidimensional scaling (MDS) [29–31] are common and have been in use for many years. More recently, new techniques such as t-SNE [32] and UMAP [33] have been gaining popularity, particularly in the context of very large datasets such as single-cell transcriptomics, where both have been used to great effect to visualize cell-type clusters. Other dimensionality reduction techniques include non-negative matrix factorization (NMF) [34, 35], Isomap [36], Linear Embedding [37], and Spectral [38]. In the context of networks, embedding techniques can also be used as a form of layout algorithm to move nodes in a network based on a reduction of a dataset to two dimensions.

Cluster ranking algorithms

The goal of ranking algorithms is to rank the clusters in the network by importance, where importance is measured by a provided attribute value (e.g., expression fold change) [39]. The ranking algorithms in *clusterMaker2* are: Hyperlink-Induced Topic Search (HITS), Multiple Attribute Additive Method (MAA), Multiple Attribute Multiplication Method (MAM), PageRank (PR) and PageRank with Priors (PRWP). The highest-ranking cluster most likely represents a biologically relevant grouping, for example, a cancer biomarker cluster. The more important the cluster, the higher rank it has.

Cytoscape

Cytoscape [40–42] is open-source software used to analyze and visualize biological data. The Cytoscape 3 series was significantly refactored to improve overall performance and modularity. Cytoscape 3.0 was first released in 2013 and is now in its ninth release (Cytoscape 3.9). Cytoscape provides an extensive App application programming interface (API) that allows programmers to extend the native capabilities of Cytoscape with new functionality. The Cytoscape app store [43] currently lists over 350 apps, which extend Cytoscape in a number of ways, from implementing specific algorithms for targeted use cases, to integrating with public data repositories, to adding new visualization capabilities.

In addition to an API for the development of Apps, Cytoscape provides a mechanism called CyREST [44] that allows R and Python-based workflows to directly interact with Cytoscape networks and data and with Cytoscape apps that have exposed commands through Cytoscape automation [45].

The Cytoscape app store [43] currently lists over 30 apps that perform some kind of clustering. However, the user interfaces of these individual apps are very different, and there is no interaction between them. Further, there are other apps that would like to include various clustering algorithms without having to reimplement them.

clusterMaker2

Here we present *clusterMaker2*, a significant update to the Cytoscape app *clusterMaker* [46]. In addition to clustering algorithms provided in the previous version of the app, a variety of frequently used dimensionality reduction techniques and ranking algorithms have been implemented in *clusterMaker2*. Furthermore, many of the new algorithms have been implemented using the Cytoscape *jobs* API, which provides a mechanism for executing remote jobs from within Cytoscape. The algorithms that use this technology calculate the results on a remote server and then send the results to *clusterMaker2* client. *clusterMaker2* is also automatable [45] through Cytoscape commands or CyREST [44], which allows it to be used by other apps in the Cytoscape ecosystem. For example, the stringApp [47] and the AutoAnnotate [48] app both use *clusterMaker2* as a clustering provider through the automation mechanism.*clusterMaker2* has added several new algorithms, including two entirely new categories of analyses: node ranking and dimensionality reduction. In the next several sections, we list the previous algorithms and describe any new algorithms.

Implementation

clusterMaker2 is an application available for use in the Cytoscape environment. Cytoscape is an open-source network visualization software platform. *clusterMaker2* extends Cytoscape by providing the functionalities needed for clustering, dimensionality reduction, and ranking. *clusterMaker2* is written in Java. *clusterMaker2* also provides new capabilities to use remote servers to execute algorithms asynchronously. In the next sections, we discuss relevant implementation details of each new algorithm, beginning with some details that cross all of the algorithms.

Matrix API

For the algorithms implemented in Java, we realized that most if not all of the modern algorithms require some sort of matrix manipulation. To facilitate this and avoid duplication within the code, we implemented an internal matrix API that supports different backend implementations. Currently, the API supports a simple implementation of matrices as two-dimensional Java arrays, a faster implementation that uses the oj! Algorithms [49], and an implementation that includes sparse matrices from parallel colt [50]. In either case, convenience routines support the creation and manipulation of matrices from nodes and their attributes as well as edges.

Remote (asynchronous) execution

Some of the network clustering and dimensionality reduction techniques described below were implemented using org.cytoscape.jobs package and a RESTful API. The Cytoscape jobs API provides a mechanism for executing jobs from within Cytoscape. It provides the framework for Cytoscape apps to marshal data, submit a remote job, check on the status of the submitted job, fetch the results, and unmarshal the data [49].

The front-end sends the graph data to the server and gets the analyzed data back. The options menu of each algorithm allows specifying a timeout period other than the default of 20 s, after which the algorithm shifts to running asynchronously in the background.

Falcon (https://falconframework.org/) is a Python-based web API framework for building app backends and microservices. clusterMaker2 uses Falcon to handle REST calls through Web Server Gateway Interface (WSGI).

Automation

clusterMaker2 exposes a wide variety of commands [45] that may be used through CyREST [44], including the RCy3 and py4cytoscape wrappers. *clusterMaker2* exposes four namespaces: **cluster** (Table 1), for network and node attribute cluster algorithms; **clusterdimreduce** (Table 2), for dimensionality reduction algorithms; **clusterrank** (Table 3), for cluster ranking algorithms; and **clusterviz** (Table 4) for the visualizations *clusterMaker2* provides. The *clusterMaker2* website (http://www.rbvi.ucsf.edu/cytoscape/clusterMaker2/) provides more details about the commands and their arguments.

Attribute clustering algorithms

In the latest version of *clusterMaker2*, we have added two new attribute clustering methods, PAM [10] and HOPACH [9]. These algorithms are similar to the existing k-means algorithm and were added by detailed transcoding of the original R implementation of HOPACH and an implementation of PAM based on the published description of the algorithm. Both were tested against R implementations of the algorithms.

Network clustering algorithms

The following algorithms have been added in the latest version of *clusterMaker2*, using the remote execution mechanism described above: Leiden [19], Infomap [20], Fast Greedy [21], Leading Eigenvector [22], Label Propagation [23] and Multilevel clusterer [24]. In each case, the server-side implementation takes advantage of the python-igraph package for the algorithm itself (see Table 5).

As discussed above, sometimes nodes are not clearly in one group or another, and, for example, might be strongly associated with more than one cluster. We introduced two fuzzy clustering algorithms, which allow nodes to be a member of more than one cluster and where membership is a proportional value. The first of these is the well-known Fuzzy C-Means (FCM) algorithm [25, 26], which we implemented in Java based on the published method. The second fuzzy algorithm, which we call a cluster "fuzzifier," was developed and implemented by the *clusterMaker2* authors. It is based on the observation that the most expensive part of the fuzzy c-means calculation is the iterative determination of the centroid of each cluster. *clusterMaker2*

Table 1 clusterMaker2 cluster commands

Namespace	Command	Description	Type
cluster	ap	Affinity propagation	Network clusterer
	autosome_heatmap	AutoSOME attribute clustering	Attribute clusterer
	autosome_network	AutoSOME network clustering	Network clusterer
	bestneighbor	Best neighbor filter	Cluster FIlter
	connectedcomponents	Connected components	Network clusterer
	cuttingedge	Cutting edge filter	Cluster FIlter
	density	Density filter	Cluster FIlter
	fastgreedy	Fast greedy (remote)	Network clusterer
	fcml	Fuzzy C-means cluster	Fuzzy cluster
	featurevector	Create correlation network from node attributes	Network clusterer
	fuzzifier	Cluster fuzzifier	Fuzzy cluster
	getcluster	Get an attribute cluster result	Utility
	getnetworkcluster	Get a cluster network cluster result	Utility
	glay	Community cluster (GLay)	Network clusterer
	hair_cut	HairCut filter	Cluster FIlter
	hascluster	Test to see if this network has a cluster of the requested type	Utility
	hierarchical	Hierarchical cluster	Attribute clusterer
	hopach	HOPACH-PAM cluster	Attribute clusterer
	infomap	Infomap (remote)	Network clusterer
	kmeans	K-means cluster	Attribute clusterer
	kmedoid	K-medoid cluster	Attribute clusterer
	labelpropagation	Label propagation (remote)	Network clusterer
	leadingeigenvector	Leading eigenvector (remote)	Network clusterer
	leiden	Leiden clusterer (remote)	Network clusterer
	mcl	MCL cluster	Network clusterer
	mcode	MCODE cluster	Network clusterer
	multilevel	Multilevel cluster (remote)	Network clusterer
	PAM Cluster	Partition around medoids (PAM) cluster	Attribute clusterer
	scps	Spectral clustering of protein sequences	Network clusterer
	transclust	Transitivity clustering	Network clusterer

Table 2 clusterMaker2 clusterdimreduce commands

Namespace	Command	Description
clusterdimreduce	isomap	Isomap (remote)
	lle	Local linear embedding (remote)
	mds	MDS (remote)
	pca	Principal component analysis
	pcoa	Principal coordinate analysis
	spectral	Spectral (remote)
	tsne	t-distributed stochastic neighbor
	tsneremote	tSNE (remote)
	umap	UMAP (remote)

Table 3 clusterMaker2 clusterrank commands

Namespace	Command	Description
clusterrank	HITS	Create rank from the hyperlink induced topic search algorithm
	MAA	Create rank from multiple nodes and edges (additive sum)
	MAM	Create rank from multiple nodes and edges (multiply sum)
	PR	Create rank from the PageRank algorithm
	PRWP	Create rank from the PageRankWithPriors algorithm

Table 4 clusterMaker2 clusterviz commands

Namespace	Command	Description
clusterviz	attributeview	Create new network from attributes
	clusterview	Create new network from clusters
	createRankingPanel	Show results from ranking clusters
	createResultsPanel	Create results panel from clusters
	destroyRankingPanel	Hide results from ranking clusters
	destroyResultsPanel	Destroy all cluster results panels
	heatmapview	JTree HeatMapView (unclustered)
	knnview	JTree KnnView
	linkSelection	Link selection across networks
	treeview	JTree TreeView
	unlinkSelection	Unlink the selection across networks

already provides several algorithms to provide a first-round segregation of nodes into clusters, and given those clusters, it is relatively easy to calculate a centroid and then evaluate proportional measurement based on those centroids. The cluster "fuzzifier," then, takes as its input a previous clustering performed by *clusterMaker2*, such as MCL, and then "fuzzifies" it by reassigning nodes to clusters proportionally. The algorithm assumes that edge values exist and are distances (not weights) and then calculates the distance from each node to each cluster centroid using a Java implementation of the algorithm defined by the R usedist [54] package's distance_to_centroid function.

Dimensionality reduction algorithms

Some of the dimensionality reduction techniques in *clusterMaker2* are coded in Java and implemented completely in the app, whereas others are implemented partially using the Cytoscape API and partially on the server. The Java dimensionality reduction techniques in *clusterMaker2* are: PCA, PCoA, and t-SNE [32]. The ones using REST implementations are: Isomap [36], Linear Embedding [37], MDS [29–31], Spectral [38], t-SNE [32], and UMAP [33]. t-SNE has been implemented in both ways.

Table 5 clusterMaker2 Cluster Algorithms. Italics algorithms are new since publication of the last paper

Type	Algorithm	Description	Source	Details
Attribute Clusterers	AutoSOME	The AutoSOME cluster algorithm [11]	The distributed AutoSOME implementation	Ported directly to clusterMaker by AutoSOME author
	Correlation	*Creates a correlation network based on node attributes*		
	Hierarchical	Standard hierarchical clustering as implemented by Eisen [1]	Cluster 3.0 package from Michiel de Hoon of the University of Tokyo	Ported by clusterMaker authors from C to Java
	k-means	Standard k-means clustering as implemented by Eisen [1] with the addition of silhouette estimation of k	Cluster 3.0 package from Michiel de Hoon of the University of Tokyo	Ported by clusterMaker authors from C to Java. Silhouette implemented by clusterMaker authors
	k-medoid	Modification of k-means from above to use medoid rather than means		Implemented by clusterMaker authors. Silhouette implemented by clusterMaker authors
	HOPACH	*Implementation of HOPACH [9] using PAM for the centroids*		*Implemented by clusterMaker authors*
	PAM	*Partition Around Medoids [10]*		*Implemented by clusterMaker authors*
Network	Affinity Propagation	The message passing-based approach to clustering by Frey and Dueck [13]	Implemented from the algorithm description in the original reference	Implemented by clusterMaker authors
	AutoSOME	The AutoSOME cluster algorithm [11]	The distributed AutoSOME implementation	Ported directly to clusterMaker by AutoSOME author
	Fast-Greedy (remote)	*The fast greedy modularity optimization algorithm for finding community structure [21]*	*python-igraph 0.9.8*	*Implemented on the server using igraph_community_fastgreedy*
	Cluster Fuzzifier	*An algorithm that takes an existing cluster and "fuzzifies" it*		*Developed and implemented by clusterMaker authors*
	Community (GLay)	Newman-Girvan [51] community clustering as implemented by Su et al [14]	The original GLay plugin for Cytoscape	Ported by clusterMaker authors
	Connected Components	Simple division based on connectivity		Implemented by clusterMaker authors
	Fuzzy C-Means cluster	*An implementation of FCM [25, 26]*		*Implemented by clusterMaker authors*
	Infomap (remote)	*Finds the community structure of the network according to the Infomap method [20, 52]*	*python-igraph 0.9.8*	*Implemented on the server using igraph_community_infomap*
	Leiden (remote)	*Finds the community structure of the graph using the Leiden algorithm [19]*	*python-igraph 0.9.8*	*Implemented on the server using igraph_community_leiden*

Table 5 (continued)

Type	Algorithm	Description	Source	Details
	Label Propagation (remote)	*Finds the community structure of the graph according to the label propagation method* [23]	*python-igraph 0.9.8*	*Implemented on the server using igraph_community_label_propagation*
	Leading Eigenvector (remote)	*Newman's leading eigenvector method for detecting community structure* [22]	*python-igraph 0.9.8*	*Implemented on the server using igraph_community_leading_eigenvector*
	MCL	Markov clustering algorithm from van Dongen [16, 17] that uses random-walks to simulate flow	Implemented from original thesis with reference to C implementation for validation of results	Implemented by clusterMaker authors as a parallel algorithm to take advantage of multiple CPU cores
	MCODE	Bader and Hogue [15] algorithm for finding modules in PPI networks	The MCODE Cytoscape plugin	Ported by clusterMaker authors
	Multilevel Cluster (remote)	*Community structure based on the multilevel algorithm* [24]	*python-igraph 0.9.8*	*Implemented on the server using igraph_community_leading_multilevel*
	SCPS	Spectral clustering algorithm for BLAST similarity networks [18]	Implemented from the algorithm description in the original reference using the authors' implementation to validate results	Implemented by clusterMaker authors
	Transitivity Clustering	Transitivity based clustering approach from Wittkop et al. [53]	Ported from Cytoscape TransClust plugin	Ported by original TransClust authors
Ranking	*MAA*	*Multiple nodes and edges (additive sum)*	*Implemented as part of Ranklust* [39]	*Added by Ranklust author*
	MAM	*Multiple nodes and edges (multiply sum)*	*Implemented as part of Ranklust* [39]	*Added by Ranklust author*
	PRWP	*PageRankWithPriors algorithm*	*Implemented as part of Ranklust* [39]	*Added by Ranklust author*
	PR	*PageRank algorithm*	*Implemented as part of Ranklust* [39]	*Added by Ranklust author*
	HITS	*Hyperlinked InducedTopicSearch algorithm with priors*	*Implemented as part of Ranklust* [39]	*Added by Ranklust author*
Dimensionality reduction	*Isomap (remote)*	*Implementation of the Isomap (Isometric Mapping) algorithm* [36]	*scikit-learn 0.24.2*	*Implemented on the server using sklearn.manifold.Isomap*
	Locally Linear Embedding (remote)	*Implementation of the Locally linear embedding (LLE) algorithm* [37]	*scikit-learn 0.24*	*Implemented on the server using sklearn.manifold.LocallyLinearEmbedding*
	MDS (remote)	*Implementation of classic multidimensional scaling* [29–31]	*scikit-learn 0.24*	*Implemented on the server using sklearn.manifold.MDS*

Table 5 (continued)

Type	Algorithm	Description	Source	Details
	Principal Component Analysis	*Implementation of Principal Component Analysis*		*Implemented by clusterMaker2 authors. using OjAlgo library version 49.2.1 to calculate the eigenvalues*
	Principal Coordinate Analysis	*Implementation of Principal Coordinate Analysis (MDS)*		*Implemented by clusterMaker2 authors. using OjAlgo library version 49.2.1 to perform SVD*
	Spectral (remote)	*Implementation of spectral embedding* [38]	*scikit-learn 0.24*	*Implemented on the server using sklearn.manifold.SpectralEmbedding*
	t-Distributed Stochastic Neighbor	*Local implementation of t-distributed Stochastic Neighbor Embedding* [32] *(t-SNE)*	*Leif Jonsson's Java implementation of t-SNE*	*Ported to clusterMaker2 by clusterMaker authors*
	tSNE (remote)	*Implementation of t-distributed Stochastic Neighbor Embedding* [32] *(t-SNE)*	*scikit-learn 0.24*	*Implemented on the server using sklearn.manifold.TSNE*
	UMAP (remote)	*Implementation of Uniform Manifold Approximation and Projection (UMAP)* [33]	*umap-learn 0.5.1*	*Implemented on the server using umap.UMAP*

Cluster ranking algorithms

MAA and MAM are simple algorithms that through addition or multiplication calculate the average score of a cluster [47]. The remaining methods utilize network ranking algorithms from the Java JUNG library [48]. All of the algorithms except HITS use node and/or edge values to calculate the rank of each cluster. All of these algorithms have been implemented directly in Java.

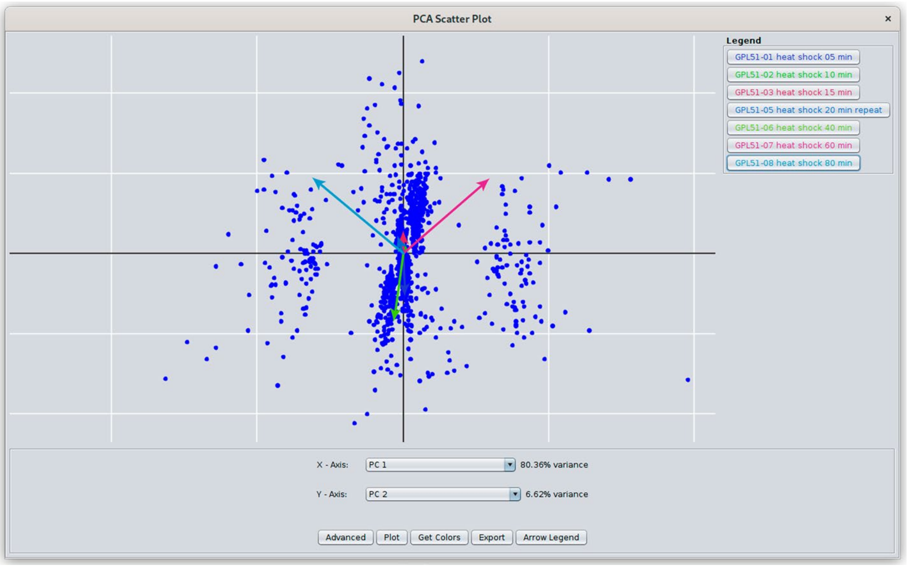
Visualization

The latest version of *clusterMaker2* adds two new visualizations and a new visual tool that augments others, in addition to the three visualization types previously described [46] (see Figure 1 in [46]).

First, we added a traditional PCA plot, complete with loading vectors, as shown in Fig. 1. By adjusting the colors and transparency, users can see the variance and where the major contributions to that variance come from. The dataset in the figure is from an early microarray experiment on the stress response in yeast [55]. The plot allows users to pan and zoom, and to map colors from the Cytoscape network onto the points on the plot.

For other dimensionality reduction techniques and embeddings, a similar scatter plot but without loading vectors is used. For example, in Fig. 2, the same data as in Fig. 1 is embedded using UMAP and shown in a scatter plot. The non-PCA scatter plots provide an additional feature where the 2D coordinates of the points on the plot may be mapped onto the Cytoscape network. This provides a dimensionality reduction-driven layout.

Finally, we added the ability to browse network clustering results in the Cytoscape Results Panel. This includes a small thumbnail of each cluster and allows the user to


PCA Scatter Plot
Legend
GPL51-01 heat shock 05 min
GPL51-02 heat shock 10 min
GPL51-03 heat shock 15 min
GPL51-05 heat shock 20 min repeat
GPL51-06 heat shock 40 min
GPL51-07 heat shock 60 min
GPL51-08 heat shock 80 min
Advanced
Plot
Get Colors
Export
Arrow Legend

X - Axis: PC 1 80.36% variance

Y - Axis: PC 2 6.62% variance

Fig. 1 PCA scatter plot from *clusterMaker2*. This plot shows the two principal components of the yeast heat shock data set (GPL51) discussed in the paper. The arrows represent the loading vectors for each of the attributes. The legend on the right shows the attributes and their loading vector colors, which may be changed by clicking on the button. Selection is bi-directional, so selecting a point in the scatter plot will select the corresponding node in the network. See the Fig. 2 legend for a description of the **Plot**, **Advanced**, and **Get Colors** buttons

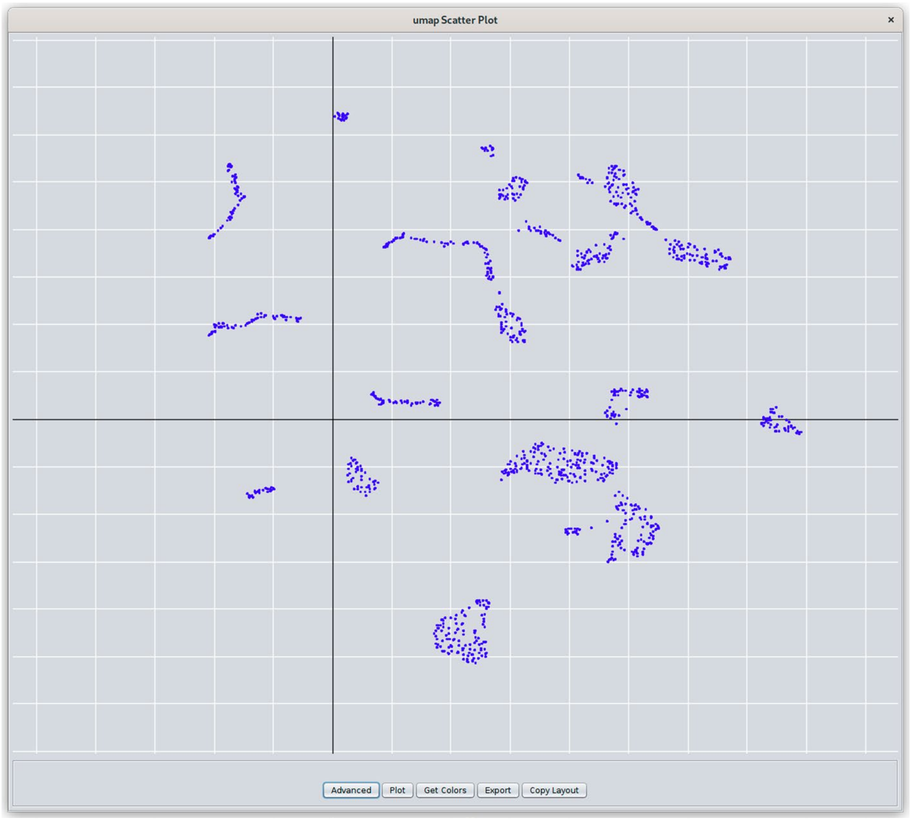


Fig. 2 UMAP scatter plot from *clusterMaker2*. Similar to the PCA plot shown in Fig. 1, UMAP, tSNE, and other dimensionality reduction techniques produce scatter plots with the same functionality except that there are no loading vectors. The plots may be pan and zoomed and selection is also bi-directional. Hovering over a node will show it's name, and the **Get Colors** button can be used to color the points according to the color of the corresponding node in the network. The **Advanced** button allows the user to select the point size and color. The use can **Export** the plot as a PNG, JPG, SVG, or PDF file. The **Copy Layout** button will use the X, Y coordinates of the points in the scatter plot to move the corresponding nodes in the network

select all of the nodes belonging to that cluster by clicking on the row. The Results Panel also provides some summary information and a column for an algorithm-specific measure of cluster quality. In addition, the overall modularity of the clusters is reported as part of the side panel.

Implementation details for all algorithms

Table 5 shows the algorithms, their descriptions and other details, and sources.

Results

As described above, *clusterMaker2* provides a variety of clustering, dimensionality reduction, and ranking algorithms to explore biological networks and attributes. In this scenario, we will show how these various algorithms can be used together to explore a dataset, and in addition, we will demonstrate the advantages associated with having *clusterMaker2* as part of the Cytoscape ecosystem and thus able to integrate with other Cytoscape apps and capabilities.

Our scenario begins with data from a yeast heat-shock experiment conducted by Gasch and colleagues, but not reported on in their paper [55]. We then combine this data with yeast protein–protein interaction data from the STRING database. The end result is a network of yeast proteins and their physical interactions annotated with transcriptional changes resulting from heat shock.

To explore this data, we cluster the network to find complexes and other tightly connected proteins, then cluster the expression data to find expression patterns, and group genes together based on those expression patterns. Using the expression patterns to color the nodes in the network allows viewing the transcriptional patterns within the context of the protein complexes and groupings. Given the somewhat manual nature of the approach taken to group the expression patterns, we explore the use of dimensionality reduction techniques to see if we could have grouped the expression data more easily that way.

Next, we explore a subset of the most significant protein clusters using our cluster "fuzzifier" to determine if there might be associations between the clusters that are lost as a result of the discrete nature of the cluster algorithms. Finally, we rank the protein clusters using the transcription data to determine which of these clusters have the most significant response to heat shock at the transcriptional level.

Data sources

Data from two sources are used in this scenario: the physical protein–protein interaction network for *Saccharomyces cerevisiae* from the STRING database, and the same heat shock data from Gasch et al. [55] that we used in our previous paper [46]. We extracted this dataset from GEO series GSE18 platform GPL51 and used the matrix data directly. In particular, we used the following columns: "GPL51-01 heat shock 05 min", "GPL51-02 heat shock 10 min", "GPL51-03 heat shock 15 min", "GPL51-05 heat shock 20 min repeat", "GPL51-06 heat shock 40 min", "GPL51-07 heat shock 60 min", and "GPL51-08 heat shock 80 min."

Importing the protein–protein interaction (PPI) network

We begin by using the stringApp [47] to load the genome-wide *Saccharomyces cerevisiae* physical protein–protein interaction network into Cytoscape. This is accomplished by pulling down the **File** menu and selecting **Import → Network from Public Databases…** Select **STRING: protein query** as the Data Source and ***Saccharomyces cerevisiae*** as the species. Then select "All proteins of this species" and click on "physical subnetwork" under **Network type**. Set the **Confidence (score) cutoff:** to 0.50 to get a network of reasonable size without too many possible false-positive edges. Now, clicking on "Import" will import the entire yeast protein–protein interaction network (Fig. 3).

Importing the expression data

To Import the expression data from GPL51 described above, we dropped any cells from the data matrix with $|\log_2(\text{fold change})| < 1$, and any rows (genes) that did not have more than one significant fold change in any of the heat-shock data columns mentioned above.

Once the dataset was modified as above, we imported it into Cytoscape using **File → Import → Table from File.** To map the data in the expression file to the

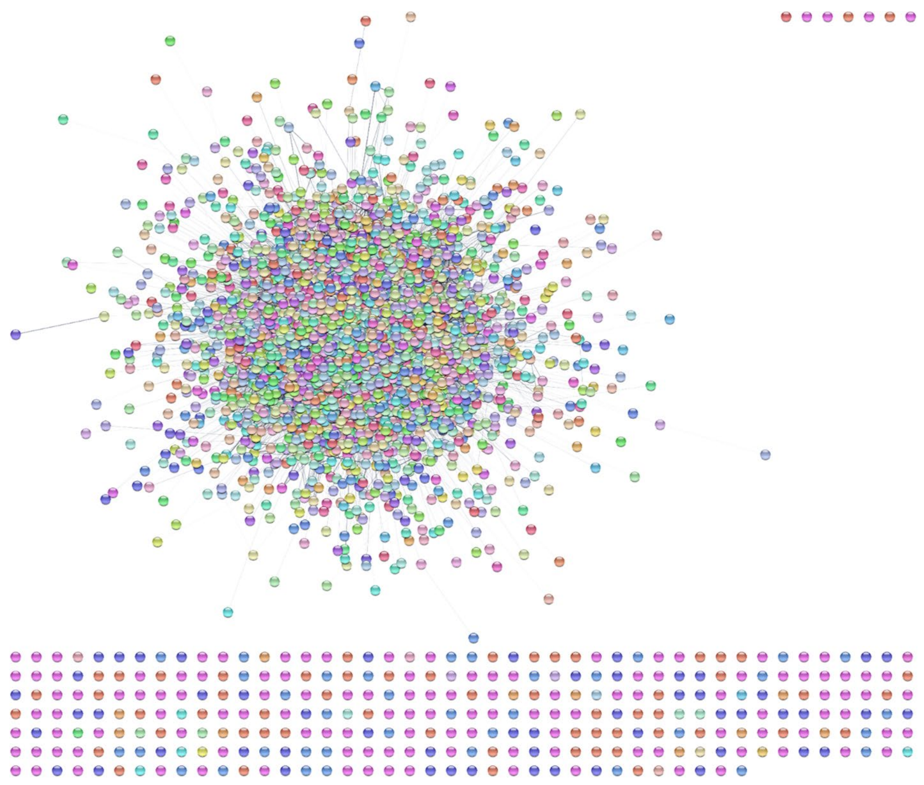


Fig. 3 Cytoscape export of the STRING protein–protein interaction network for Saccharomyces cerevisiae imported using the stringApp. The confidence score used was 0.5 and the standard STRING style is shown

PPI network, we set the “Key column for the network” to **display name** and under “Advanced Options…” we set the delimiter to **Tab** and turn off **COMMA**. Now we have a PPI network where each of the proteins are annotated with the 5-min (GPL51-01), 10-min (GPL51-02), 15-min (GPL51-03), 20-min (repeat) (GPL51-05), 40-min (GPL51-06), 60-min (GPL51-07), and 80-min (GPL51-08) heat-shock expression fold changes, if they passed our cutoffs. We are now set to analyze this dataset.

Clustering the PPI network

The PPI network as shown in Fig. 3 is too dense for easy interpretation. A typical next step is to break the network up into clusters, which represent tightly connected groups of proteins such as complexes. We will use Leiden clustering [19] to do this. The Leiden algorithm is an improvement of the Louvain algorithm, which maximizes the modularity score of each community by comparing how much more connected the nodes are in a community compared to a random network. It has three phases: local moving of nodes, refinement of the partition aggregation of the network based on the refined partition, and using the non-refined partition to create an initial partition for the aggregate network.

Select **Apps → clusterMaker Cluster Network → Leiden Clusterer (remote)** to bring up the Leiden cluster options. After some experimentation (Leiden clustering is relatively quick), we found a resolution parameter of 0.5 and 30 iterations to work well. The resolution parameter correlates with the size of the communities. Higher resolution parameter values lead to smaller communities, while lower values lead to fewer, larger communities. The number of iterations gives the number of times Leiden algorithm is iterated. Each iteration improves the partition further. Adjusting the parameters to be the ones given above results in a network clustered to a biologically meaningful extent. For the **Source for array data**, select "stringdb::score" as the Attribute. This is the edge confidence score assigned by STRING. Select "Create new clustered network" and click "OK". The resulting network should look similar to Fig. 4. Note that we have disabled the "Glass ball effect" and "STRING style labels" in the STRING results panel at the right. A quick exploration of the clustered network confirms that Leiden has done a reasonable job—the first four clusters in the upper left-hand corner are the ribosome, preribosome, large subunit of the preribosome, and the mitochondrial ribosome, respectively. This makes sense, as these are all large complexes. The fifth cluster is RNA polymerase II holoenzyme, and the sixth is the spliceosome. What complex a cluster represents can be determined by selecting a cluster and running the stringApp functional enrichment on that cluster using the entire genome as a background.

Hierarchical clustering of expression data

A classical analysis of an expression dataset would involve performing a hierarchical clustering of the data and viewing it using a heatmap with associated dendrogram.

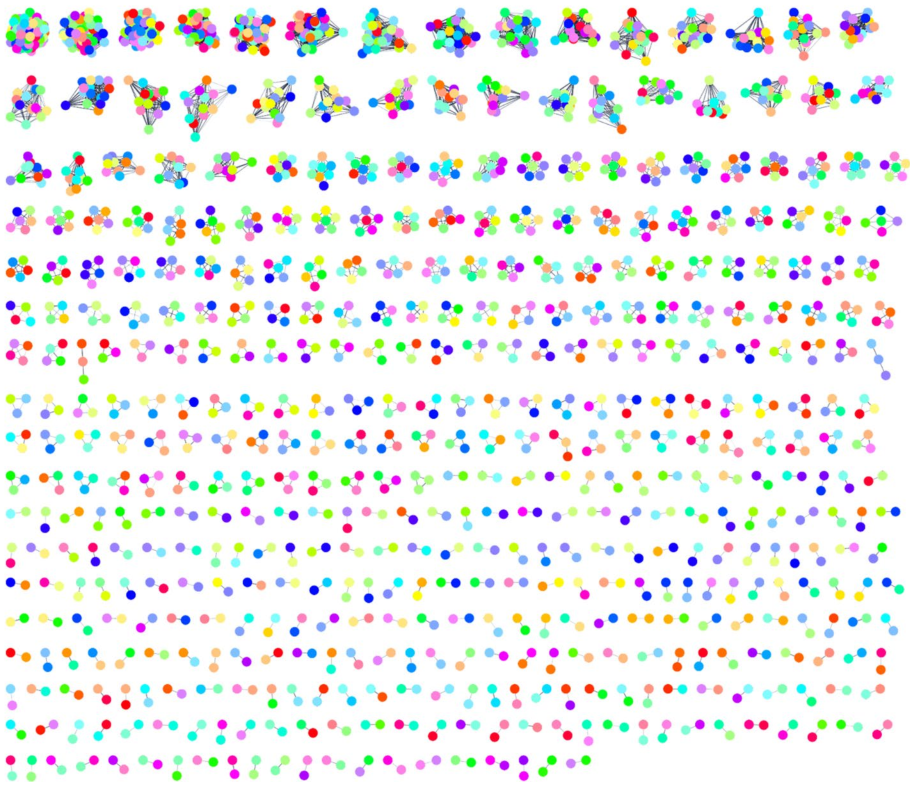


Fig. 4 Export of the *Saccharomyces cerevisieae* protein–protein interaction network after clustering with Leiden clusterer using a resolution of 0.5 and 30 iterations. Node colors are random (preserved from the initial STRING network), and we have disabled the "glass ball" effect as well as the STRING-style labels using the stringApp results panel

In hierarchical clustering, at each iteration of the proximity matrix the similar clusters merge with other clusters until one cluster is formed. The nodes the most similar to each other are grouped together earlier.

To do this using *clusterMaker2*, we select **Apps → clusterMaker Cluster Attributes → Hierarchical cluster**. We then select all of the heat shock columns (GPL51-01 heat shock 5 min, GPL51-02 heat shock 10 min, GPL51-03 heat shock 15 min, GPL51-05 heat shock 20 min repeat, GPL51-06 heat shock 40 min, GPL51-07 heat shock 60 min, and GPL51-08 heat shock 80 min), then click on "Show TreeView when complete" and finally click OK. This will bring up a heat map with the associated dendrogram for the dataset (Fig. 5). By selecting branches of the dendrogram, we can select groups of genes in the heatmap and simultaneously select the corresponding proteins in the PPI view.

Coloring the PPI network

To help understand the biological significance of these transcriptional changes at the protein level, we would like to find a mapping from the hierarchical clustering onto the proteins in the PPI network. This could be useful, for example, to see if any complexes are particularly affected by transcriptional changes.

There are two ways to map expression dataset onto the network. The simplest approach is to use the **Map Colors Onto Network** capability in TreeView: select all of the rows and click on "Create HeatStrips." This will add bar charts showing the expression fold changes at the various times post-heat-shock on the nodes (as in the Fig. 6 inset). Unfortunately, this is extremely hard to see when looking at the entire network. To explore this dataset more fully, we used the ability to select branches of the dendrogram, which selected the corresponding nodes in our PPI. We created a new column in the Node Table named **Color** and assigned values from -10 to 10 depending on the level of upregulation or downregulation across time points we saw in the corresponding dendrogram branch. This was done quite crudely based only on visual inspection of the dendrogram branches and heatmap colors. For completeness, we did try to calculate a cut of the tree using external tools, but the dendrogram is extremely dense and it was difficult to achieve satisfactory results that provided anything close to a usable distribution on the tree.

To map the colors, we created a continuous mapping in Cytoscape using the ColorBrewer [56] diverging Red-Blue palette. The combination of the approximate color and heatstrips provides a general overview of the expression changes for each cluster, and when we zoom in, the corresponding details (Fig. 6). Note that SSA1, a member of the HSP70 family, is initially upregulated for the first 10 min, but shows no expression change after that. This can be seen easily in the close-up view (or the corresponding row in the heatmap), but is not visible in the PPI overview.

UMAP analysis of expression data

The manual process we used above to choose colors was relatively straightforward, albeit somewhat time-consuming. As the number of attributes increases, however, it can be extremely difficult to group the genes together in meaningful ways. Dimensionality reduction techniques have become an increasingly valuable tool to group data based on

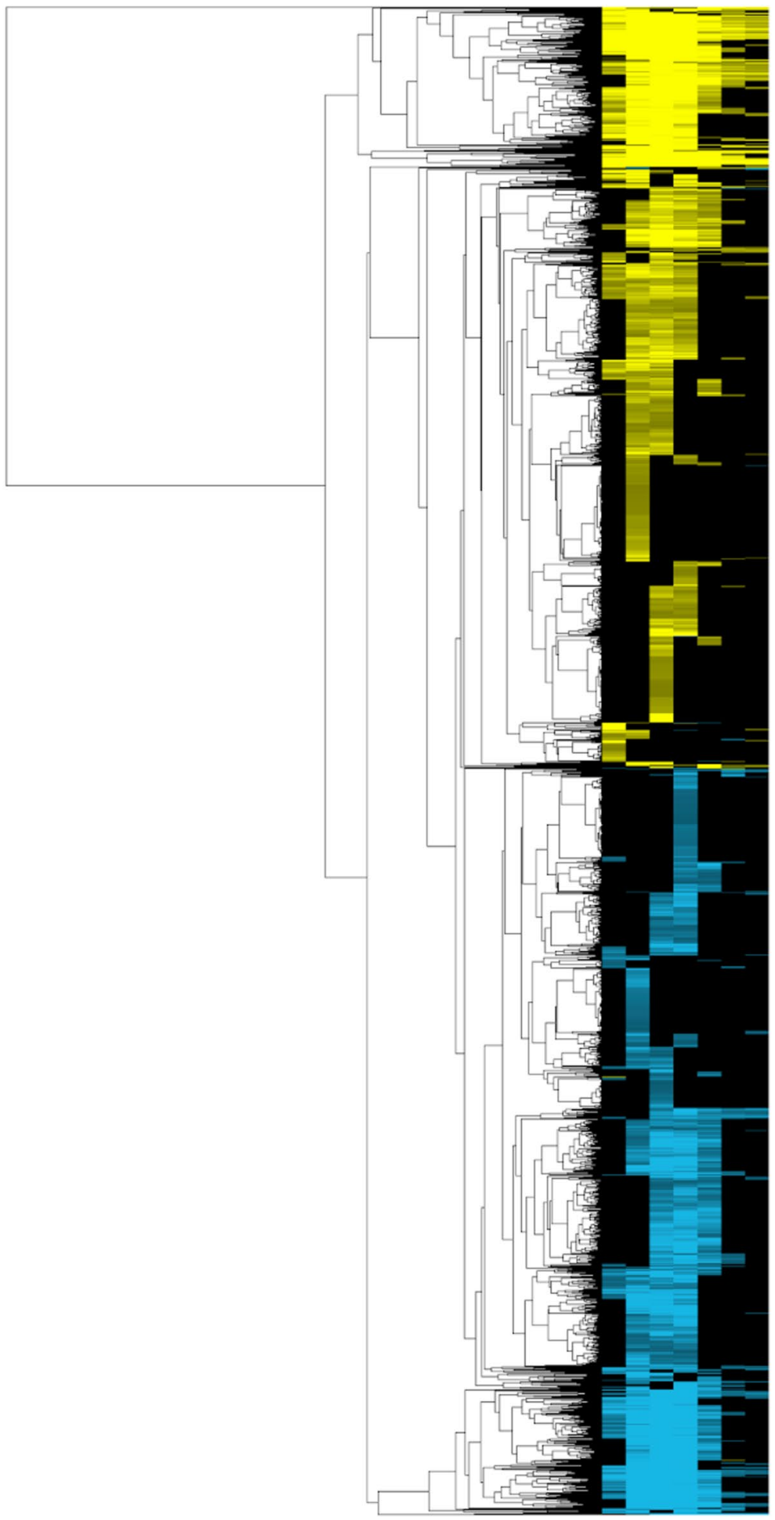


Fig. 5 Export from the clusterMaker2 Jtree HeatMap view of the hierarchical clustering of heat shock data from GPL51. The image uses a standard yellow-cyan color scheme where yellow grandient indicates the degree of over-expression and cyan gradient indicates the degree of under-expression

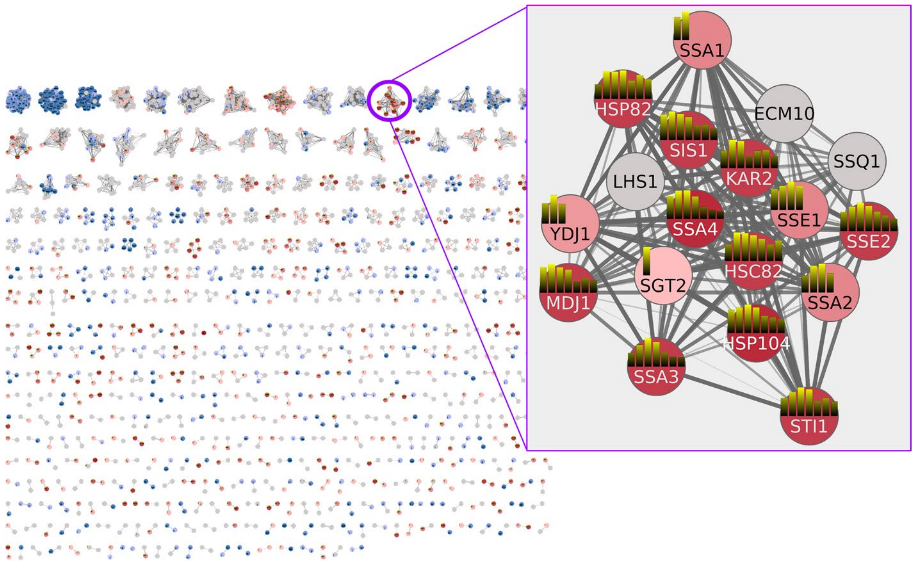


Fig. 6 Using colors and heatstrips to explore PPI and expression data. Nodes are colored using a BrewerColor Red-Blue palette derived from the **Color** column as described in the text. Red indicates overexpression and blue indicates underexpression. The inset shows proteins involved in protein folding and refolding are overexpressed during heat stress response. The heatstrips (small bar charts on the nodes) show the expression changes at the individual time points and use the yellow-cyan gradient described in Fig. 5

numerous attributes by providing a visualization that groups similar items together even when the number of attributes is very large.

We will now look at the same heat shock expression data used for the hierarchical clustering Fig. 5, but this time using the Uniform Manifold Approximation and Projection [57] (UMAP) approach to explore a 2D embedding of this multidimensional data. UMAP can be used for visualization of high-dimensional datasets similarly to t-SNE, but also for general nonlinear dimension reduction. In mathematical terms, it is a manifold learning technique constructed from the theoretical framework based on Riemannian geometry and algebraic topology. It can use labels for supervised dimensionality reduction and transform new data into a pretrained embedding space.

Because there are a number of nodes with no data, first select all of the nodes with a color value or a "Non-Zero Count" 1 or greater (you can sort the column or use the Cytoscape Filter tab). Once they are all selected, use **Apps → clusterMaker Dimensionality Reduction → UMAP (remote)** to bring up the UMAP options. Select all six of the heat shock columns, select "Only use data from selected nodes" and set the "Number of neighbors" to 20 and the "Minimum distance" to 0.5. The number of neighbors controls how UMAP balances local versus global structures. It constraints the size of the local neighborhood the algorithm looks at when learning the data. Low number of neighbors makes UMAP concentrate on a local structure and potentially lose the big picture, whereas larger values of number of neighbors results in UMAP looking at larger neighborhoods with the potential cost of losing fine details. The minimum distance parameter controls how tightly UMAP packs points together. It provides the minimum distance that points are allowed to be apart. Low values of minimum distance will result in clumpier embeddings, which serve the interest in finer details better. Larger values prevent UMAP from packing points together and puts the focus on the preservation of the topological structure. Make sure to also click "Show scatter plot with results" so we can explore the resulting embedding. Click OK to send the request to the server to perform the UMAP. Once the UMAP scatter plot comes up, click on "Get Colors" to apply the red-blue coloring from the nodes to the UMAP. It is easy to see how well the UMAP segregates the data (Fig. 7). Highlighting the red group in the center and performing an enrichment on the selected nodes indicates significant enrichment for protein folding. Indeed, many of the nodes in this cluster are part of the highlighted group in Fig. 6.

Fuzzy clustering

Fuzzy clustering can help to reveal the relationships between clusters to find instances where proteins are shared between clusters. As described above, we have implemented a novel approach to calculating fuzzy clusters in *clusterMaker2*, which we call a "fuzzifier."

To apply fuzzy clustering, we must start with the fully connected network depicted in Fig. 3 or a clustered network with the inter-cluster edges added. We could perform a fuzzy clustering on the entire network, but the result would again be too dense to facilitate exploration. Another approach is to select the nodes from several clusters that are of interest—for example, clusters with high ranking or that show consistently high over-expression or under-expression. In this case, we'll choose the nodes from nine clusters

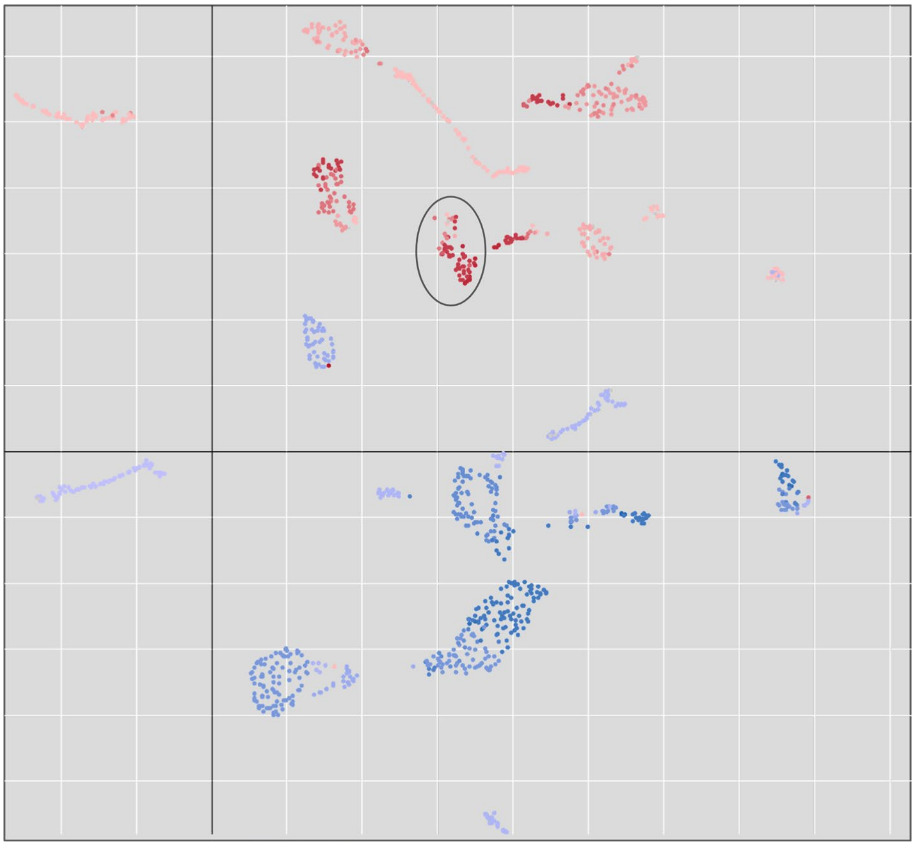


Fig. 7 UMAP embedding of the heat shock expression data exported from clusterMaker2's scatter plot (see Fig. 2). The colors correspond to the manually created **Color** column where fold change values are mapped onto the Red–Blue ColorBrewer gradient (red = degree of overexpression, blue = degree of underexpression). The circled group is enriched in protein folding, unfolded protein binding, and protein refolding

that contain many over-expressed genes: clusters 13, 19, 20, 25, 82, 197, 242, 259, and 668. We selected the nodes in those clusters by using the *clusterMaker2* “link selection across networks” function and then selecting the clusters in the clustered view (Fig. 6). This selected the corresponding nodes in the full network (Fig. 3). Then we could perform fuzzy clustering by selecting **Apps→ clusterMaker Cluster Network→ Cluster Fuzzifier**. We chose *stringdb::score* as the **Array Source**, *1/value* as the **Edge weight conversion** (to convert it to a distance) and selected **Cluster only selected nodes** and selected **Create new clustered network** to see the result. To show the relationships between the clusters, we redid the layout using the score as an edge weight and changed the shape of the fuzzy cluster centroids (Fig. 8).

The results indicate that clusters 20, 82, and 197 are closely intertwined. We used the stringApp enrichment analysis to explore the grouping by looking at enrichment of all three clusters, then cluster 20 and 197 together and cluster 82 alone. All three of these clusters represent genes in the mitochondrion exclusively. All of the genes are enriched in the GO Biological Process “Cellular respiration.” Clusters 20 and 197 represent proteins in the TCA cycle, and all of the proteins in cluster 82 are part of the oxidation–reduction process in the mitochondrion, including the mitochondrial respiratory chain complexes III and and IV. It makes perfect sense that genes responsible for cellular respiration would be upregulated in response to heat stress. RIP1, in particular, has been shown to have an important role in the selective export of heat shock RNAs [58]. The close association of these three clusters would not have been apparent without performing a subsequent fuzzy clustering, yet, there is little doubt that cluster 82 is logically separate from clusters 20 and 197, so this is not in any way an indictment of the Leiden cluster results. Furthermore, cluster 20 represents two complexes, the mitochondrial succinyl-CoA synthetase complex with LSC1 and LSC2 and the respiratory chain complex II with SDH1, SDH2, SDH3, SDH4, and YJ045W (SDH9), which is a paralog of

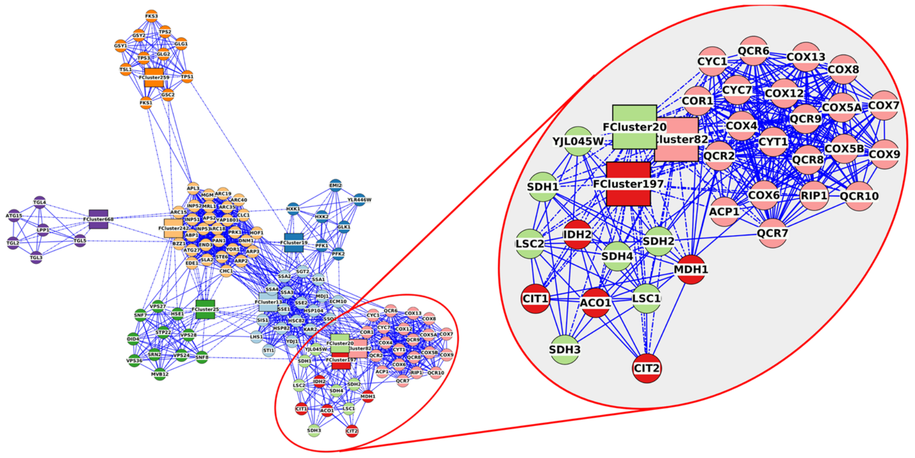


Fig. 8 Fuzzy clustering of 9 selected Leiden clusters. Inset shows three intertwined clusters that represent mitochondrial proteins. Solid lines are from the original STRING protein–protein interaction network and dashed lines are the membership edges between proteins and their fuzzy-cluster centroid. Node colors are assigned based on cluster number, and each fuzzy cluster is represented by a FClusterNNN node where the NNN is replaced by the cluster number of the dominant cluster. The FCluster node is put at the centroid of all nodes in that fuzzy cluster

SDH1. The separation between cluster 20 and 197 is also explained by the STRING evidence, which shows that while LSC2 has a strong confidence score for interaction with IDH2, it does not bind with any confidence to the rest of the proteins. On the other hand, LSC1 has a strong confidence score for interaction with SDH2, and both LSC1 and LSC2 have moderate confidence scores for interaction with the rest of the proteins in respiratory chain complex II.

Overall, the fuzzy clustering analysis provides us with a much more nuanced view of the relationships between these clusters, allowing a more detailed analysis of the molecular processes of the heat shock response in yeast. The incremental approach—discrete clustering, then fuzzy clustering of the groups of interest—allows us to avoid overinterpreting the initial clustering while not increasing the complexity that would result from fuzzy clustering of the entire network.

Cluster ranking

The goal of ranking is to order the clusters based on some criteria (typically node attributes) to determine the most relevant or important clusters [39].

After clustering the network with Leiden clustering as shown in Fig. 6, MAA ranking was applied from **Apps → clusterMaker Ranking → Create rank from multiple nodes and edges (additive sum)**. To focus on up-regulated genes, we will choose the same node attributes (GPL51-01–GPL51-08) and select **Basic** normalization, but **Only positive values** for the Two-tailed values normalization. After running the ranking algorithm, the ranking panel (Fig. 9) can be opened from **Apps → clusterMaker Visualizations → Show results from ranking clusters**. By calculating a ranking score for each cluster, we can analyze the relevance of the clusters in terms of the research question. In this case, higher ranking score would imply that the cluster is more associated with yeast heat shock. The genes grouped in the cluster 13 with a ranking score 1.0 were examined more closely to assess the biological relevance of the ranking results. Cluster 13 was chosen for closer examination because its ranking score was the highest of all clusters.

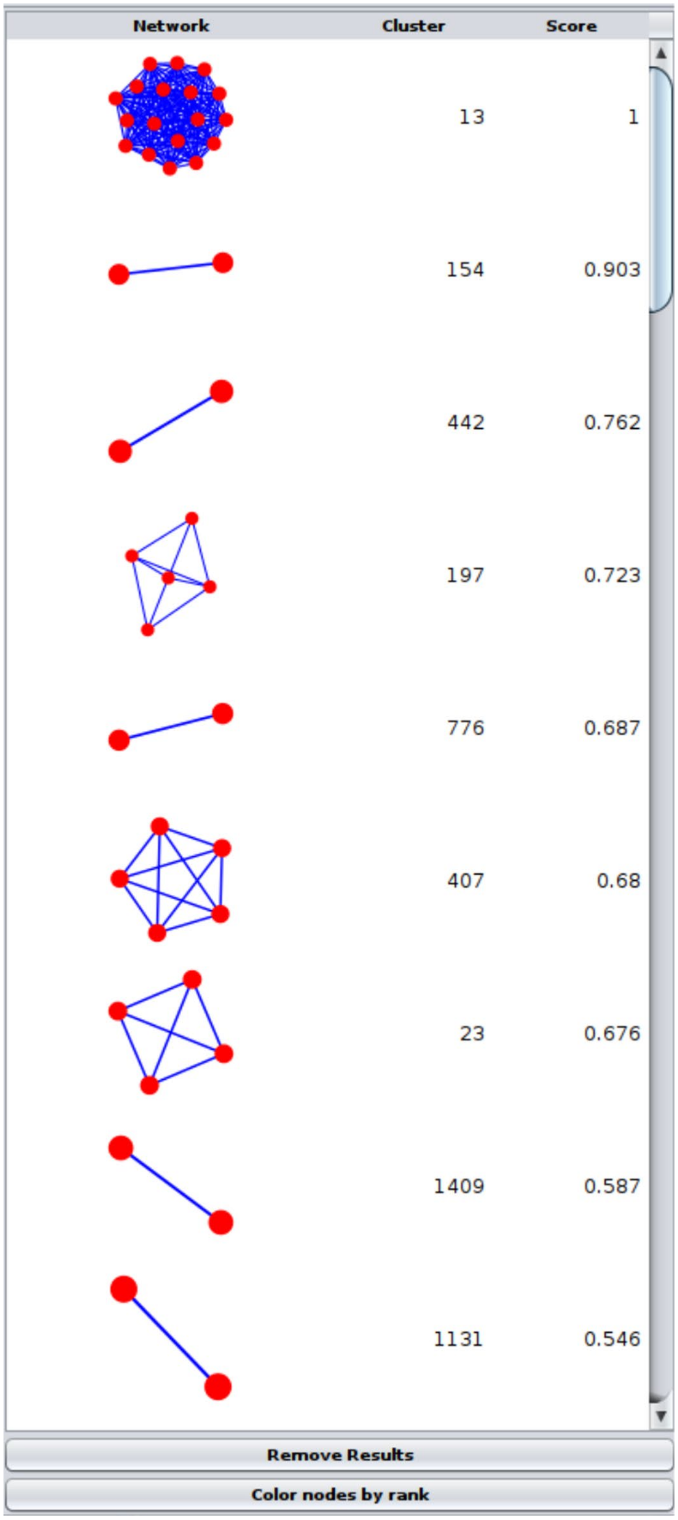
The biological relevance of cluster 13 was checked in two ways. First, the proteins in the cluster were manually looked up in the protein database UniProt [59]. Furthermore, functional enrichment analysis was performed on the cluster.

Looking at Uniprot, most of the genes in cluster 13 are associated with yeast heat shock, which supports the results of the ranking algorithm. For example, SSA4 (https://www.uniprot.org/uniprot/P22202) and HSP104 (https://www.uniprot.org/uniprot/P31539) are known heat-shock proteins.

Functional enrichment analysis was performed on cluster 13 using the STRING Functional Enrichment function [47] (Fig. 10). Most genes in the cluster are associated with GO [60] Molecular function "Unfolded Protein Binding" and GO Biological Process "Protein Folding." Furthermore, Reactome [61] Pathways "Cellular responses to Stress" and "Cellular Responses to Heat Stress" included most of the proteins involved.

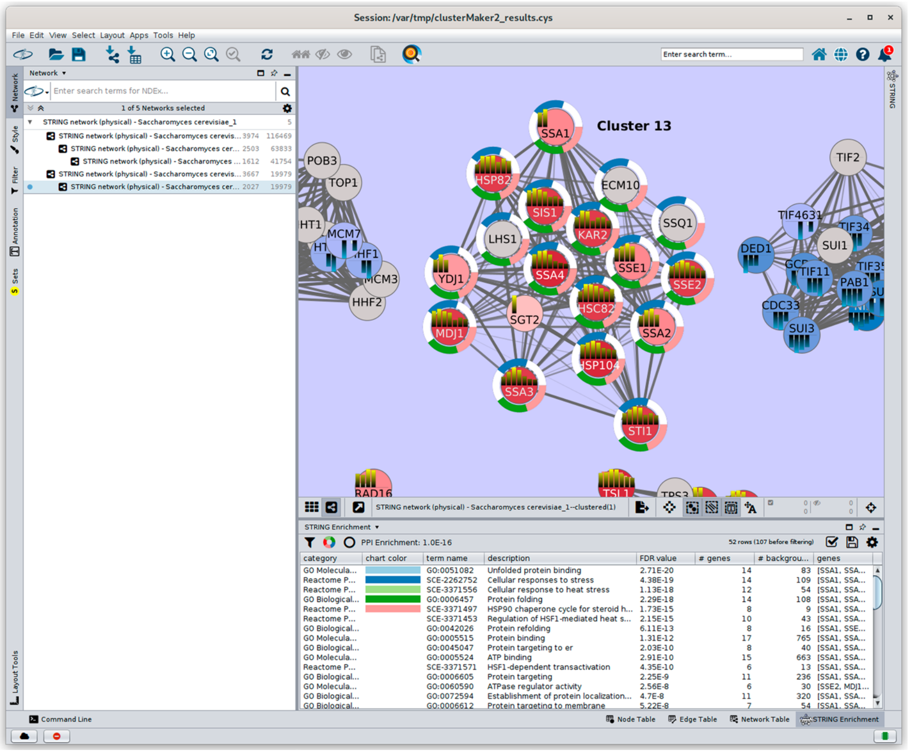
Assessing the biological relevance of other clusters with high, mid and low scores

Cluster 13 is not the only one with a high ranking score. Cluster 154 has a score of 0.903, which indicates a high relevance as well. The cluster consists of two proteins, which are both strongly upregulated. One of these is a heat shock protein, HSP42, involved



Network	Cluster	Score
	13	1
	154	0.903
	442	0.762
	197	0.723
	776	0.687
	407	0.68
	23	0.676
	1409	0.587
	1131	0.546

Fig. 9 clusterMaker2 ranking panel with ranking results from the multiple nodes and edges (additive sum) ranking algorithm. As discussed in the test, the highest cluster with score 1.0 is cluster 13



STRING network (physical) - Saccharomyces cerevisiae_1		5
STRING network (physical) - Saccharomyces cerevis...	3974	116469
STRING network (physical) - Saccharomyces cer...	2503	63833
STRING network (physical) - Saccharomyces ...	1612	41754
STRING network (physical) - Saccharomyces cerevis...	3667	19979
STRING network (physical) - Saccharomyces cer...	2027	19979

STRING network (physical) - Saccharomyces cerevisiae_1--clustered(1)

STRING Enrichment

PPI Enrichment: 1.0E-16

52 rows (107 before filtering)

category	chart color	term name	description	FDR value	# genes	# backgrou...	genes
GO Molecula...	🟦	GO:0051082	Unfolded protein binding	2.71E-20	14	83	[SSA1, SSA...
Reactome P...	🟦	SCE-2262752	Cellular responses to stress	4.38E-19	14	109	[SSA1, SSA...
Reactome P...	🟩	SCE-3371556	Cellular response to heat stress	1.13E-18	12	54	[SSA1, SSA...
GO Biological...	🟩	GO:0006457	Protein folding	2.29E-18	14	108	[SSA1, SSA...
Reactome P...	🟥	SCE-3371497	HSP90 chaperone cycle for steroid h...	1.73E-15	8	9	[SSA1, SSA...
Reactome P...		SCE-3371453	Regulation of HSF1-mediated heat s...	2.15E-15	10	43	[SSA1, SSA...
GO Biological...		GO:0042026	Protein refolding	6.11E-13	8	16	[SSA1, SSE...
GO Molecula...		GO:0005515	Protein binding	1.31E-12	17	765	[SSA1, SSA...
GO Biological...		GO:0045047	Protein targeting to er	2.03E-10	8	40	[SSA1, SSA...
GO Molecula...		GO:0005524	ATP binding	2.91E-10	15	663	[SSA1, SSA...
Reactome P...		SCE-3371571	HSF1-dependent transactivation	4.35E-10	6	13	[SSA1, SSA...
GO Biological...		GO:0006605	Protein targeting	2.25E-9	11	236	[SSA1, SSA...
GO Molecula...		GO:0060590	ATPase regulator activity	2.56E-8	6	30	[SSE2, MDJ1...
GO Biological...		GO:0072594	Establishment of protein localization...	4.7E-8	11	320	[SSA1, SSA...
GO Biological...		GO:0006612	Protein targeting to membrane	5.22E-8	7	54	[SSA1, SSA...

Fig. 10 Screenshot of Cytoscape showing functional enrichment results for cluster 13. The top 5 results are shown, which include unfolded protein binding (cyan), cellular response to stress (blue), cellular response to heat stress (light green), protein folding (green), and HSP90 chaperone cycle for steroid hormone receptors (SHR) (pink). The second, third, and fifth terms are Reactome pathways; the first term is from GO molecular function; and the fourth term is from GO biological process. As shown in Fig. 6, the heatstrips represent the individual expression fold change for each time point. The enrichment results strongly correspond to the consistent up-regulation across the entire time spectrum for most of these genes

in cytoskeleton reorganization after heat shock (https://www.uniprot.org/uniprotkb/Q12329/entry). Cluster 447, with a relatively high score of 0.762, is also a two-protein cluster. Both proteins are upregulated, indicating an association with heat shock. Cluster 259 has a low score of 0.469 and consists of four genes, of which some are up- and some downregulated, and one protein is neutral.

Discussion

There is little doubt or debate about the importance of grouping data based on some similarity or distance metric. Within a network context, whether the nodes represent genes, proteins, cells, or any other entities linked by some form of relationship, the metric can be expressed as the connectivity or weighted connectivity between the linked entities. Independent of a network context, the metric can be expressed as the similarity of values or features between the entities. However, as demonstrated above, combining grouping based on network connectivity with that based on entity features can be valuable. *clusterMaker2* provides exactly that capability.

However, it is also clear that *clusterMaker2* does not cover the entire space of community detection, clustering, and unsupervised classification. It might be argued that some favored conventional clustering algorithm or dimensionality reduction technique is missing. While this may be true, it is also the case that no package could provide access to all clustering and dimensionality reduction techniques. The new web service framework added in the latest releases of *clusterMaker2* should provide us with the ability to respond much more quickly to user requests and we will continue to add new algorithms over time based on user feedback.

Another potential criticism of *clusterMaker2* is that all of the algorithms implemented in *clusterMaker2* are already available in R, or Python. This is certainly true, and we have taken advantage of that by using Python packages extensively for our web service implementation, so we are extremely grateful and supportive of the R and Python bioinformatics communities. On the other hand, *clusterMaker2* is integrated into Cytoscape and is part of the overall Cytoscape ecosystem. This provides the opportunity to leverage these clustering packages from within Cytoscape without programming, and to integrate those results into Cytoscape networks. As mentioned above, *clusterMaker2* is already used by other Cytoscape apps to add clustering capabilities to specific workflows. Integration into Cytoscape also provides significant advantages in terms of the integration of the visualizations provided with the network, as demonstrated in the figures above. *clusterMaker2* specifically links the visualizations, providing a brushing and linking [62, 63] facility between the various visualizations and the Cytoscape network.

One other criticism is that *clusterMaker2* provides very limited support for cluster evaluation techniques. Currently, only a modularity score is produced for network cluster algorithms and silhouette is provided for algorithms that require a cluster number as an input parameter. However, there are several other cluster evaluation techniques that provide internal or external measures of the clustering. External cluster evaluation techniques such as the Rand Index [64] and Mutual Information [65] require some type of ground truth, which makes them somewhat unsuitable for the typical *clusterMaker2* use cases where ground truth is almost never known. Internal cluster evaluation techniques, on the other hand, do not require a ground truth, and would be worthwhile to additions to a future version of *clusterMaker2.* For example, both the Calinski-Harabasz index [66] and the Davies-Bouldin index [67] could be added to *clusterMaker2* in a new **Cluster Evaluation** category. Until that point, however, the Python package sklearn provides these functions (sklearn.metrics.calinski_harabasz_score and sklearn.metrics. davies_bouldin_score and a good tutorial covering these methods is available on the analyticsindiamag.com web site [68]. It should be noted, however, that while cluster evaluation algorithms provide an assessment of the clustering based on various measures of "goodness", the best clustering for most *clusterMaker2* users is one that reflects the underlying biology. A good way to inspect that is shown in the workflow above, where over-representation analysis was performed on each cluster to determine the extent to which the underlying biology was reflected in the cluster. This in no way detracts from the use of cluster evaluation algorithms, but they should be used in conjunction with biological meaningful measures. Hopefully, a future version of *clusterMaker2* will facilitate that.

The contributions of supervised and semi-supervised machine learning techniques to biology are also important to mention here. In particular, modern unsupervised learning techniques such as autoencoders have been used effectively in a biological context (c.f. [69–73]), and *clusterMaker2* currently does not provide any support for these algorithms. While it is clear that these techniques are achieving great success, at this point, it would be difficult to incorporate them into *clusterMaker2*'s architecture. However, it should be noted that applications such as AlphaFold [74] have been extremely successful by distributing pre-trained neural networks, skipping the training step for end users. One could imagine similar approaches that would pre-train a series of neural networks for common types of biological networks. Pre-trained networks would be much easier to integrate into *clusterMaker2*.

Finally, there are several areas to explore in future versions of *clusterMaker2*. First, as mentioned above, we continue to respond to user requests for additional clustering algorithms that are currently not included in *clusterMaker2*. Second, we are exploring unsupervised machine learning algorithms such as variational autoencoders [75] for inclusion. Finally, the usability of the scatterplot implementation could be improved to provide support for better selection modes, panning, and zooming.

Conclusions

clusterMaker2 represents a significant advance over the previously published version of *clusterMaker* and is the culmination of a large number of improvements, including the addition of dimensionality reduction techniques, fuzzy clustering methods, and the use of web services to improve the ease of implementation of new algorithms and the performance of algorithms on large networks. All of these additional features have been integrated into the Cytoscape ecosystem and have been made accessible via CyREST commands to allow integration into bioinformatic pipelines using R or Python. Most importantly, *clusterMaker2* provides an easy-to-use tool to perform clustering and to visualize clusters within the Cytoscape network context.

Availability and requirements

Project name: clusterMaker2

Project home page: https://www.rbvi.ucsf.edu/cytoscape/clusterMaker2/

Project source: https://www.github.com/RBVI/clusterMaker2/

Operating system(s): Platform independent

Programming language: Java

Other requirements: Java 11, Cytoscape 3.9 or higher

License: GNU LGPL

Any restrictions to use by non-academics: None.

Acknowledgements
The authors are grateful for the excellent comments from Lars Juhl Jensen and Elaine Meng on this manuscript

Author contributions
JHM and MU were both responsible for coding and testing the application; MU implemented the ***jobs*** API integration and JHM implemented the backend. Both JHM and MU participated in the development of the manuscript. Both authors read and approved the final manuscript.

Funding
JHM was funded by National Institute of General Medical Sciences Grant P41 GM103504. MU was funded by the Google Summer of Code program in 2020 and 2021.

Availability of data and materials
The datasets generated and/or analysed during the current study are available publicly as a Cytoscape session file at: https://doi.org/10.5281/zenodo.7366164

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 16 November 2022 Accepted: 11 March 2023
Published online: 05 April 2023

References

1. Eisen MB, Spellman PT, Brown PO, Botstein D. Cluster analysis and display of genome-wide expression patterns. Proc Natl Acad Sci U S A. 1998;95:14863–8.
2. Krogan NJ, Cagney G, Yu H, et al. Global landscape of protein complexes in the yeast *Saccharomyces cerevisiae*. Nature. 2006;440:637–43.
3. Collins SR, Kemmeren P, Zhao XC, Greenblatt JF, Spencer F, Holstege FC, Weissman JS, Krogan NJ. Toward a comprehensive atlas of the physical interactome of *Saccharomyces cerevisiae*. Mol Cell Proteomics. 2007;6:439–50.
4. Grün D, Lyubimova A, Kester L, Wiebrands K, Basak O, Sasaki N, Clevers H, van Oudenaarden A. Single-cell messenger RNA sequencing reveals rare intestinal cell types. Nature. 2015;525:251–5.
5. Jaitin DA, Kenigsberg E, Keren-Shaul H, et al. Massively parallel single-cell RNA-seq for marker-free decomposition of tissues into cell types. Science. 2014;343:776–9.
6. Mahata B, Zhang X, Kolodziejczyk AA, et al. Single-cell RNA sequencing reveals T helper cells synthesizing steroids de novo to contribute to immune homeostasis. Cell Rep. 2014;7:1130–42.
7. Lloyd Least square quantization in PCM. Bell Telephone Laboratories Paper. Published in journal much later: Lloyd, SP: Least squares quantization in PCM. IEEE Trans Inf Theory.
8. Forgy EW. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. Biometrics. 1965;21:768–9.
9. van der Laan MJ, Pollard KS. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. J Stat Plan Inference. 2003;117:275–303.
10. Kaufman L, Rousseeuw PJ. Partitioning around medoids (program pam). In: Finding groups in data: an introduction to cluster analysis, vol. 344; 1990. p. 68–125.
11. Newman AM, Cooper JB. AutoSOME: a clustering method for identifying gene expression modules without prior knowledge of cluster number. BMC Bioinform. 2010;11:117.
12. Wittkop T, Emig D, Lange S, Rahmann S, Albrecht M, Morris JH, Böcker S, Stoye J, Baumbach J. Partitioning biological data with transitivity clustering. Nat Methods. 2010;7:419–20.
13. Frey BJ, Dueck D. Clustering by passing messages between data points. Science. 2007;315:972–6.
14. Su G, Kuchinsky A, Morris JH, States DJ, Meng F. GLay: community structure analysis of biological networks. Bioinformatics. 2010;26:3135–7.
15. Bader GD, Hogue CW. An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinform. 2003;4:2.
16. vanDongen S (2000) A cluster algorithm for graphs. Information Systems [INS]
17. Enright AJ, Van Dongen S, Ouzounis CA. An efficient algorithm for large-scale detection of protein families. Nucleic Acids Res. 2002;30:1575–84.
18. Nepusz T, Sasidharan R, Paccanaro A. SCPS: a fast implementation of a spectral method for detecting protein families on a genome-wide scale. BMC Bioinform. 2010;11:120.
19. Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep. 2019;9:5233.
20. Rosvall M, Axelsson D, Bergstrom CT. The map equation. Eur Phys J Spec Top. 2009;178:13–23.
21. Clauset A, Newman MEJ, Moore C. Finding community structure in very large networks. Phys Rev E Stat Nonlinear Soft Matter Phys. 2004;70:066111.
22. Newman MEJ. Finding community structure in networks using the eigenvectors of matrices. Phys Rev E Stat Nonlinear Soft Matter Phys. 2006;74:036104.
23. Raghavan UN, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E Stat Nonlinear Soft Matter Phys. 2007;76:036106.
24. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. J Stat Mech Theory Exp. 2008;2008:P10008.
25. Peizhuang W. Pattern recognition with fuzzy objective function algorithms (James C. Bezdek). SIAM Rev. 1983;25:1.
26. Dunn JC. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. J Cybern. 1973;3:32–57.
27. Pearson K. LIII. On lines and planes of closest fit to systems of points in space. Lond Edinb Dublin Philos Mag J Sci. 1901;2:559–72.
28. Gower JC. Principal coordinates analysis. Wiley StatsRef: statistics reference online 1–7; 2015.
29. Borg I, Groenen PJF. Modern multidimensional scaling: theory and applications. Berlin: Springer; 2005.
30. Kruskal JB. Nonmetric multidimensional scaling: a numerical method. Psychometrika. 1964;29:115–29.

31. Kruskal JB. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika. 1964;29:1–27.
32. van der Maaten LJP, Hinton GE. Visualizing high-dimensional data using t-SNE. J Mach Learn Res. 2008;9:2579–605.
33. McInnes L, Healy J, Saul N, Großberger L. UMAP: uniform manifold approximation and projection. J Open Source Softw. 2018;3:861.
34. Paatero P, Tapper U. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. Environmetrics. 1994;5:111–26.
35. Seung, Lee. Algorithms for non-negative matrix factorization. In: Advances in neural information processing systems.
36. Tenenbaum JB, de Silva V, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science. 2000;290:2319–23.
37. Roweis ST. Nonlinear dimensionality reduction by locally linear embedding. Science. 2000;290:2323–6.
38. Belkin M, Niyogi P. Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. 2003;15:1373–96.
39. Lund-Hanssen H. Ranklust: an extension of the Cytoscape clusterMaker2 plugin and its application to prioritize network biomarkers in prostate cancer; 2016.
40. Su G, Morris JH, Demchak B, Bader GD. Biological network exploration with Cytoscape 3. Curr Protoc Bioinform. 2014;47:8.13.1-8.13.24.
41. Yeung N, Cline MS, Kuchinsky A, Smoot ME, Bader GD. Exploring biological networks with Cytoscape software. Curr Protoc Bioinform. 2008;Chapter 8:Unit 8.13.
42. Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T. Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res. 2003;13:2498–504.
43. Lotia S, Montojo J, Dong Y, Bader GD, Pico AR. Cytoscape app store. Bioinformatics. 2013;29:1350–1.
44. Ono K, Muetze T, Kolishovski G, Shannon P, Demchak B. CyREST: turbocharging cytoscape access for external tools via a RESTful API. F1000Res. 2015;4:478.
45. Otasek D, Morris JH, Bouças J, Pico AR, Demchak B. Cytoscape Automation: empowering workflow-based network analysis. Genome Biol. 2019;20:185.
46. Morris JH, Apeltsin L, Newman AM, Baumbach J, Wittkop T, Su G, Bader GD, Ferrin TE. clusterMaker: a multi-algorithm clustering plugin for Cytoscape. BMC Bioinform. 2011;12:436–436.
47. Doncheva NT, Morris JH, Gorodkin J, Jensen LJ. Cytoscape StringApp: network analysis and visualization of proteomics data. J Proteome Res. 2019;18:623–32.
48. Kucera M, Isserlin R, Arkhangorodsky A, Bader GD. AutoAnnotate: a Cytoscape app for summarizing networks with semantic annotations. F1000Res. 2016;5:1717.
49. oj! Algorithms. https://www.ojalgo.org/. Accessed 24 Jan 2022.
50. Wendykier P, Nagy JG. Parallel colt: a high-performance java library for scientific computing and image processing. ACM Trans Math Softw. 2010;37:1–22.
51. Newman MEJ, Girvan M. Finding and evaluating community structure in networks. Phys Rev E Stat Nonlinear Soft Matter Phys. 2004;69:026113.
52. Rosvall M, Bergstrom CT. Maps of information flow reveal community structure in complex networks; 2007. arXiv preprint physics. soc-ph/0707.0609.
53. Wittkop T, Baumbach J, Lobo FP, Rahmann S. Large scale clustering of protein sequences with FORCE—a layout based heuristic for weighted cluster editing. BMC Bioinform. 2007;8:396.
54. Bittinger K, Bittinger MK. Package "usedist."; 2020. https://brieger.esalq.usp.br/CRAN/web/packages/usedist/usedist.pdf. Accessed 16 Jul 2022.
55. Gasch AP, Spellman PT, Kao CM, Carmel-Harel O, Eisen MB, Storz G, Botstein D, Brown PO. Genomic expression programs in the response of yeast cells to environmental changes. Mol Biol Cell. 2000;11:4241–57.
56. Brewer CA. A transition in improving maps: the ColorBrewer example. Cartogr Geogr Inf Sci. 2003;30:159–62.
57. McInnes L, Healy J, Melville J. UMAP: uniform manifold approximation and projection for dimension reduction; 2018. arXiv [stat.ML].
58. Stutz F, Kantor J, Zhang D, McCarthy T, Neville M, Rosbash M. The yeast nucleoporin rip1p contributes to multiple export pathways with no essential role for its FG-repeat region. Genes Dev. 1997;11:2857–68.
59. Pundir S, Martin MJ, O'Donovan C. UniProt protein knowledgebase. Methods Mol Biol. 2017;1558:41–55.
60. Ashburner M, Ball CA, Blake JA, et al. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat Genet. 2000;25:25–9.
61. Fabregat A, Sidiropoulos K, Garapati P, et al. The Reactome pathway Knowledgebase. Nucleic Acids Res. 2016;44:481.
62. Becker RA, Cleveland WS. Brushing scatterplots. Technometrics. 1987;29:127–42.
63. Hearst User interfaces and visualization. Modern information retrieval.
64. Rand WM. Objective criteria for the evaluation of clustering methods. J Am Stat Assoc. 1971;66:846–50.
65. Vinh NX, Epps J, Bailey J. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: Proceedings of the 26th annual international conference on machine learning. Association for Computing Machinery, New York, NY, USA; 2009. p. 1073–80.
66. Caliński T, Harabasz J. A dendrite method for cluster analysis. Commun Stat Simul Comput. 1974;3:1–27.
67. Davies DL, Bouldin DW. A cluster separation measure. IEEE Trans Pattern Anal Mach Intell. 1979;1:224–7.
68. Mehta S. A tutorial on various clustering evaluation metrics. In: Analytics India Magazine; 2022. https://analyticsindiamag.com/a-tutorial-on-various-clustering-evaluation-metrics/. Accessed 7 Mar 2023.
69. Kinalis S, Nielsen FC, Winther O, Bagger FO. Deconvolution of autoencoders to learn biological regulatory modules from single cell mRNA sequencing data. BMC Bioinform. 2019;20:379.
70. Chen H-IH, Chiu Y-C, Zhang T, Zhang S, Huang Y, Chen Y. GSAE: an autoencoder with embedded gene-set nodes for genomics functional characterization. BMC Syst Biol. 2018;12:142.
71. Seninge L, Anastopoulos I, Ding H, Stuart J. VEGA is an interpretable generative model for inferring biological network activity in single-cell transcriptomics. Nat Commun. 2021;12:5684.

72. Pratella D, Ait-El-Mkadem Saadi S, Bannwarth S, Paquis-Fluckinger V, Bottini S. A survey of autoencoder algorithms to pave the diagnosis of rare diseases. Int J Mol Sci. 2021. https://doi.org/10.3390/ijms221910891.
73. Way GP, Greene CS. Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. Pac Symp Biocomput. 2018;23:80–91.
74. Jumper J, Evans R, Pritzel A, et al. Highly accurate protein structure prediction with AlphaFold. Nature. 2021;596:583–9.
75. Kingma DP, Welling M. Auto-encoding variational Bayes; 2013. arXiv [stat.ML].

Publisher's Note