UC Irvine
UC Irvine Electronic Theses and Dissertations

Title
Usage of Databases to Manage Multidimensional Data Effectively in Biomedical Engineering Laboratories

Permalink
https://escholarship.org/uc/item/3nt7d0hf

Author
Ochs, Alexander

Publication Date
2018

Copyright Information
This work is made available under the terms of a Creative Commons Attribution License, availalbe at https://creativecommons.org/licenses/by/4.0/

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA,
IRVINE

Usage of Databases to Manage Multidimensional Data Effectively in Biomedical Engineering Laboratories

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTERS OF SCIENCE

In Biomedical Engineering

by

Alexander Richard Ochs

Thesis Committee:
Assistant Professor Anna Grosberg, Chair
Associate Professor Elliot Botvinick
Professor Frithjof Kruggel

2018

© 2018 Alexander Richard Ochs

TABLE OF CONTENTS

	Page
LIST OF FIGURES	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT OF THE THESIS	iv
INTRODUCTION	1
CHAPTER 1: Methods	3
LMNA Mutation Data Set Information	3
Software Usage and Setup	3
Introduction to SQL and Available Resources	4
Statistics	4
Design Requirements & Considerations	4
CHAPTER 2: Results	6
Background	6
Pipeline Setup and Organization	7
Database Structure	8
MS Access SQL & Query Creation Examples	9
Identifying A Novel Relationship in LMNA Mutation Data Set	10
CHAPTER 3: Discussion	12
REFERENCES	14

LIST OF FIGURES

		Page
Figure 2.1	An example visualization of multidimensional data	7
Figure 2.2	An example of common data pipeline needs	8
Figure 2.3	Table and Design View relationships within LMNA Mutation data set	9
Figure 2.4	An example query using SQL syntax	10
Figure 2.5	Comparison between groups for the variable Actin OOP	11

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Anna Grosberg, for her constant mentorship and unwavering support throughout my two years in her lab.

I would also like to thank my two other committee members, Dr. Elliot Botvinick and Dr. Frithjof Kruggel, for their feedback and assistance with the thesis process.

ABSTRACT OF THE THESIS

Usage of Databases to Manage Multidimensional Data Effectively in Biomedical Engineering Laboratories

By

Alexander Richard Ochs

Masters of Science in Biomedical Engineering

University of California, Irvine 2018

Assistant Professor Anna Grosberg, Chair

Data that can be plotted across multiple dimensions of attributes are most often managed using spreadsheet programs, which is both error prone and time inefficient. In comparison, databases are highly scalable and effective solutions, but the perspective and skill sets necessary to implement systems for custom needs are often lacking in biomedical engineering laboratories. In this work, we illustrate the usage of common software in Matlab and Microsoft Access to create a data pipeline to convert raw data into structured formats then aggregate the data in a variety of ways using queries in a database. Methods and visuals for our specific context as well as generalized recommendations are included to provide others with a framework to construct their own custom data pipeline and databases. We also discovered a novel relationship within our own data set using the elucidated methods.

INTRODUCTION

In an era where scientific progress is heavily driven by technology, handling large amounts of data has become an integral facet of research across all disciplines. The emergence of new fields such as computational biology and genomics underscore how critical the proactive utilization of technology has become. These trends are certain to continue due to Moore's law and steady progress gained from technological advances [1, 2]. One consequence, however, is the rising quantities of generated data that exceed the capabilities of previously viable organization methods. Although most academic laboratories have sufficient computational resources for handling complex data sets, many groups lack the technical expertise necessary to construct custom systems suited for developing needs [3]. Having the skills to manage and update such data sets is critical for efficient workflow and output. Bridging this gap between data and expertise is important for efficiently handling, re-updating, and analyzing a broad spectrum of multifaceted data.

Scalability is an essential consideration when handling large data sets. "Big data," for instance, is a flourishing area of research that involves revealing new insights from processing huge volumes of data, of great heterogeneity, and gathered at very high rates [4, 5]. Using automated methods of organization and analysis are mandatory for this field to appropriately handle torrents of data. Although there are many exciting findings yet to be discovered using big data, most academic labs do not need the scope, power, and complexity of those methods for addressing their own scientific questions [5]. It is undoubtable that scientific data grows increasingly complex with time at this point in history [6]. Unfortunately, many scientists continue to use previous methods of organization that no longer meet their expanding data needs. Convenient spreadsheet programs such as Microsoft (MS) Excel are frequently used to organize scientific data, but at the cost of being unscalable, error prone, and time inefficient in the long run [7, 8]. Databases are an effective solution to this problem however, as they are relatively cheap and easy to use in scalably handling varied data sets of ongoing projects.

Immediate concerns that arise when considering databases are cost, accessibility, and time investment for training and usage. Frequently used in business settings, these programs are more economical, being either relatively inexpensive or free, than the funding required to support use of big data systems. In fact, a variety of both commercially available and open source software exists for creating and maintaining databases, such as Oracle Database, MySQL, and MS Access [9]. Many researchers would also be encouraged to learn that several MS Office academic packages come with MS Access included, further minimizing cost considerations. Furthermore, nearly all developers provide extensive documentation online and there is a plethora of free online resources such as Codecademy, W3Schools, and SQLBolt to help researchers understand and utilize structured query language (SQL) [10-12]. Like any programming language, learning how to use databases and code using SQL takes time to master, but between the ample resources available the process is straightforward and well worth the effort invested.

The main intent of this work, therefore, is to enlighten the general scientific community about the potential of databases as scalable data management systems as well as to provide a general

template using specific examples of patient sourced cell-line experiments. It highlights common considerations for and the utility of constructing a database coupled with a data pipeline necessary to convert raw data into structured formats. The basics of database interfaces and coding for databases in structured query language (SQL) are provided and illustrated with examples to allow others to gain the knowledge applicable to building basic frameworks. Finally, a sample, experimental data set demonstrates how easily and effectively databases can be designed to aggregate multifaceted data in a variety of ways. All this compiled information provides context, commentary, and templates for assisting fellow scientists on the path to implementing databases for their own experimental needs.

CHAPTER 1: Methods

For the purposes of creating a scalable database in a research laboratory setting, experimental data was collected over the course of many months to a year. While the nature of the data is not as important for the purposes of this discussion, in our research laboratory the collection came from experiments conducted using human fibroblast cells. The primary focus of this manuscript was to report on the organization of computer software to enable us to aggregate, update, and manage these data in the most cost- and time-efficient manner possible, but the relevant experimental methods used to collect example data are provided below for context purposes.

The experiments were conducted by spin-coating circular glass coverslips with a 10:1 mixture of polydimethylsiloxane (PDMS, Ellsworth Adhesives) and curing agent, then applying fibronectin, an extracellular matrix protein, in either unorganized (Isotropic) or specific micropatterned arrangements (Lines). Fibroblast cells were seeded onto these coverslips at optimal densities and left to grow for 48 hours before being fixed. After fixation and several washes, the coverslips were immunostained for cell nuclei, actin, and fibronectin and re-washed. Clear nail polish was used to seal coverslips to microscope slides and left to dry for 24hr. Fluorescence images were obtained for ten randomly selected fields of view for each coverslip. Custom-written Matlab codes were used to quantify different variables from the images about the nuclei, actin filaments, and fibronectin. These processes are elaborated in more depth in a prior published article by our group [13].

LMNA Mutation Data Set Information

For the sample data used in developing this methodology, data collection was approved and informed consent was performed in accordance with UC Irvine Institutional Review Board (IRB # 2014-1253). Human fibroblast cells were collected from three families of different variations of the same gene mutation: heterozygous *LMNA* splice-site mutation (c.357-2A>G) [14] (Family A); *LMNA* nonsense mutation (c.736 C>T, PQ246X) in exon 4 [15] (Family B); and *LMNA* missense mutation (c.1003C>T, pR335W) in exon 6 [16] (Family C). Fibroblast cells were also collected in other individuals in each family as related mutation-negative controls and others were purchased as unrelated mutation-negative controls. As a positive control, fibroblast cells from an individual with Hutchinson-Gliford Progeria (HGPS) were purchased and grown from a skin biopsy taken from an 8-year-old female patient with HGPS possessing a *LMNA* G608G point mutation [17]. In total, fibroblasts from 22 individuals were tested and used as data in this work.

Software Usage and Setup

Matlab (Mathworks) was used as a coding software to create a data pipeline and MS Access (Microsoft) was used as a database software. There are myriad alternative coding languages and database management systems (DBMS) that in combination could be used for the same data pipeline purposes; the most appropriate choices are highly dependent on the data and context being used. Some excel most in scalability, flexibility, reliability, and other factors [9]. Since the experimental data here was already generated and stored using Matlab, continued usage of this program was an intuitive choice. Similarly, MS Access is one of the most user-friendly DBMS and was already available for our lab through pre-existing packages. A series of codes were

subsequently created to link key identifying information with data file locations in a spreadsheet format. The file locations were also used to pull corresponding data values from their source folders and generate a new spreadsheet to import into the database. The advantage of storing file locations is the ability to quickly refer back to source files if needed, allowing for greater accessibility and replicability of the data.

Setting up a database in Access involved creating a series of tables to link together and queries to pull the data from. By associating specific outputs in certain fields with other tables of linked information, queries have the ability to merge the same data in a variety of different ways. Only one table contains the spreadsheet created by the data pipeline, but the other tables link information such as how Patient A1 is from Family A and has an Age of 38. Queries are automatically reupdated according to the data in the main spreadsheet, and are incredibly useful to use to check preliminary data during ongoing projects.

Introduction to SQL and Available Resources

SQL is the universal database programming language for writing queries. Queries are written requests to the database for specific data or information, and although there are other ways to write queries than using SQL, coding in SQL directly provides users the most stringent control over specifying desired outputs. All common database software developers additionally provide extensive documentation online. Our development of methodology is focused primarily on how users can learn to code in SQL themselves, instead of how to use the alternative workarounds. SQL is universal among all common database programs and, subsequently, there are a variety of different websites available at present that help teach its syntax and logic structures. Nearly all major providers of SQL tutorials are free and available online, including but not limited to Codecademy, W3Schools, and SQLBolt [10-12]. Some alternatives that require subscriptions do exist however, such as the program teaching website Lynda [18].

Statistics

For our sample experimental data, the Analysis of Variance (ANOVA) test with Tukey's method was used for mean comparisons between various conditions.

Design Requirements & Considerations

For design requirements and considerations for database management systems, this setup takes pre-existing experimental information stored in variables and output files and organizes them into a spreadsheet format to then be imported into a database for aggregation purposes. Since there are many factors that the data can be analyzed or grouped by, benefits of our methods include updating pre-existing data sets and running statistical analysis across a variety of conformations quickly. This is extremely helpful for ongoing projects, where even a single new data entry requires new statistical analysis, and streamlines easy access of data across 10+ different variables in the sample data set. For instance, the setup in its current implementation is used by all researchers working with a sample data set from cell culture experiments of fibroblasts across individuals with and without a LMNA/C mutation, for re-aggregation purposes that streamline ease of new statistical analysis. My contribution to this project has been usability and functionality changes to

the existing pipeline system, creation of the fibroblast database and all accompanying queries, bug fixes, and regular management of the data set as experiments were ongoing.

An example use case of using the database is if a user had several new experiments of data that had been imaged and quantified using lab-specific Matlab codes. The user is able to note the file locations of interest, use the data pipeline method to create new entries corresponding to these files, combine new entries with the sum of all previously generated entries, check for errors, then create a spreadsheet of values matched to entries for database importation. The new data set is then imported into the database, and pre-written queries run and return new means, standard deviations, and counts to be analyzed in a separate statistical software.

CHAPTER 2: Results

Background

Databases can be powerful tools for increasing data accessibility and ease of aggregation, but it's important to discern the differences between data that would benefit and not benefit from greater control of organization. Multidimensionality in data refers to the number of dimensions that a measurement can be grouped against, and databases are most powerful when handling higher dimensions of data. For example, a data set containing years and the number of measurements per year is two dimensional since only one possible grouping exists (measurements against years). It is simplest to use a spreadsheet program such as MS Excel to handle 2D information instead of databases. On the opposite end of the spectrum is data that can be grouped in many possible conformations, such as data from clinical settings. Data involving clinical patients could have an graphable outcome against age, gender, disease status, presence of certain symptoms, and so on. Databases are effective at quickly aggregating the data in the large number of conformations needed here; organizing the data points manually would be a very tedious and nontrivial process due to the large variety of possible groupings.

In the LMNA/C mutation context, **Figure 2.1** demonstrates how the example data set has high dimensionality and can be grouped in a variety of ways. The distinctness of a data point here is a combination of the variable being measured (Variable type), the organized or disorganized patterned arrangement of fibronectin for fibroblasts (Pattern type), and which iteration of the experiment was conducted (Coverslip #). One cube corresponds to one data set individual, who could further be grouped by Family, Mutation Status, etc. or plotted against continuous variables like Age. Other categories could also have been plotted, as this data set has up to ten dimensions of possible conformations.

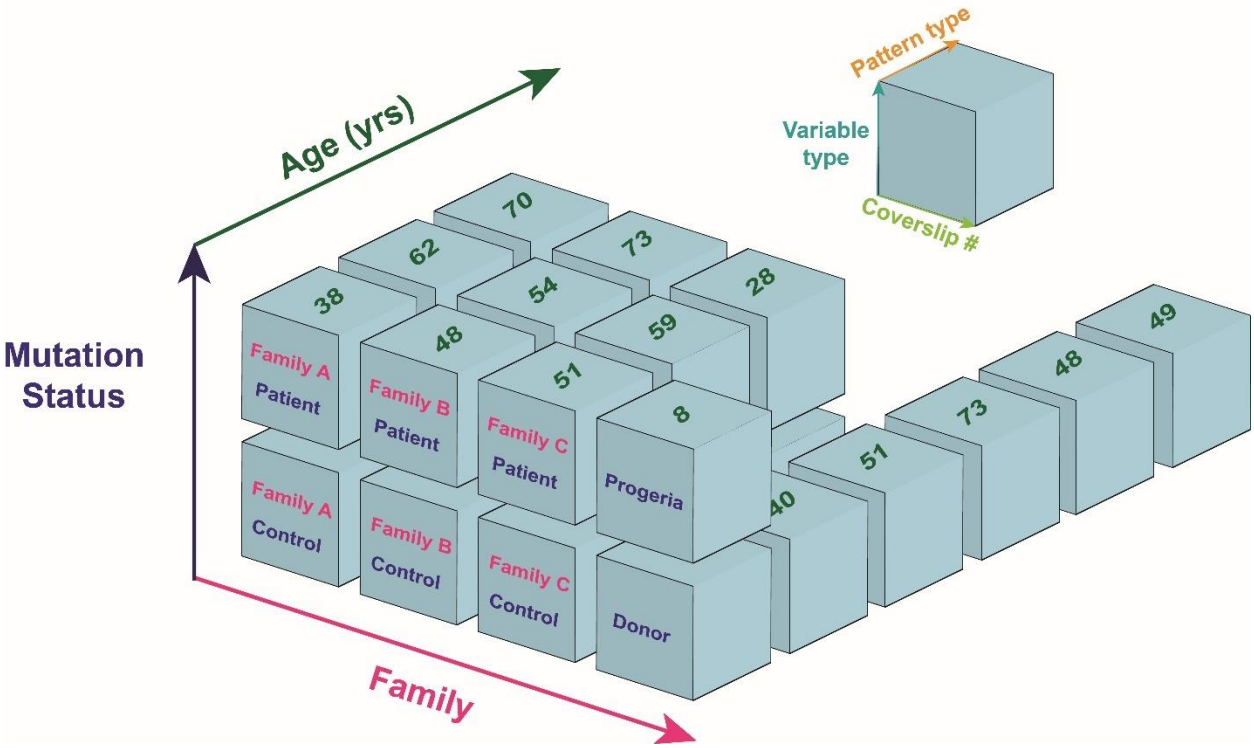

Figure 2.1: An example visualization of multidimensional data. Uniqueness of a data point in this context is defined by a combination of Variable Type, Pattern Type, and Coverslip Number, which will cover a multitude of variables and conditions from multiple experiments. Data is be further separated by Mutation Status, Family status, or Age additionally. This data set has up to ten dimensions but only six are shown for viewing simplicity.

Pipeline Setup and Organization

Up to an estimated 95% of all digital data is unstructured, so it is very important to use automated methods when converting raw data into structured formats [4]. Manual data entry would defeat the purpose of using the scalability of databases to save time and reduce error. Creating a good automated method is highly context dependent though. In the context discussed in this work, different variables from identical experimental conditions across a variety of subjects were quantified from fluorescent images using custom Matlab codes. These variables were located in different files in a plethora of folders under the same location and the files were not intrinsically linked with the subject whose images were being analyzed, but the file names, variable names, and file outputs were consistent across all data points.

Under the given conditions, the data pipeline needed a method that would efficiently convert scattered data output files into a structured format. Identifying all the details that determine the uniqueness of a data point is critical before beginning the process in earnest however. For instance, in the example studies, an experiment here could vary by the Subject whose cells were used, the Pattern Type that was microcontact printed, and the specific Coverslip # of which iteration the experiment was. In addition, within an experiment there are different Variable Types that are quantified. These four categories of information (called “fields” in database terms) are all required for data points to be distinct from one another. Other helpful, non-vital information can

also be added such as the Total # of Coverslips to indicate the number of repetitions conducted and if data points are missing. Implementing all necessary fields into the pipeline, along with their associated value per data point, is dependent on the coding language and file types being worked with however.

In order to increase the rigor of data used, we chose to input file locations instead of the values initially, so that a corresponding spreadsheet copy was always available with the locations of every data point. This increases the transparency and accessibility of the data, and we highly recommend readers consider implementing similar measures instead of only storing the data values alone.

Figure 2.2 shows a generalized example of common pipeline needs. The ability to add new entries, combine them with older entries, check for errors, and output the desired values from their corresponding file locations are all important elements in an efficient data pipeline.

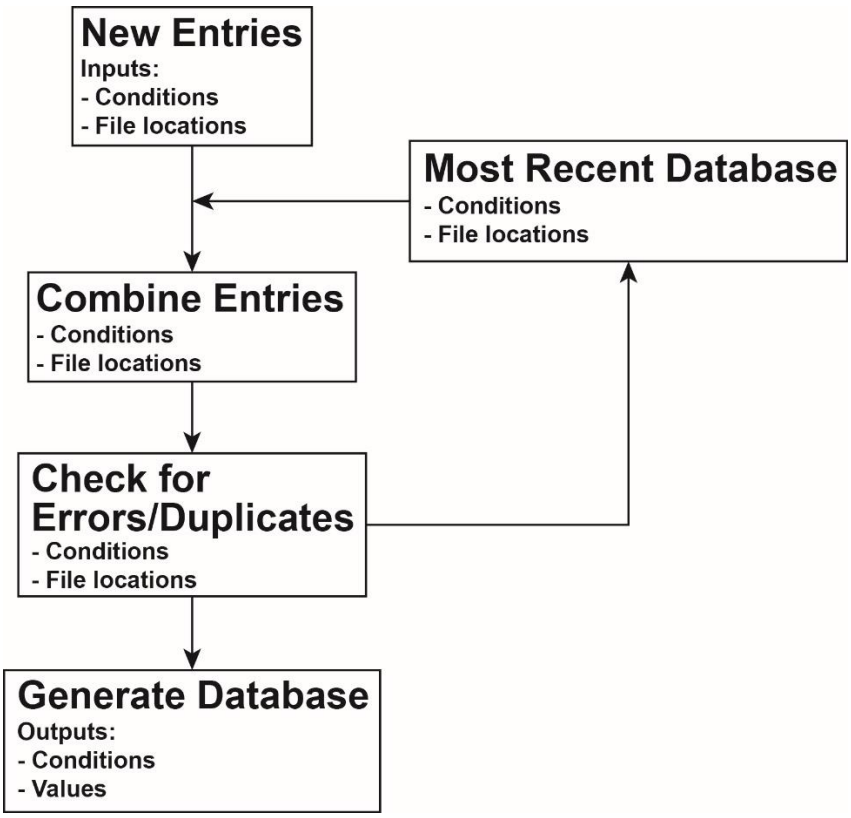

Figure 2.2: An example of common data pipeline needs. New entries are created via user inputs and selecting file locations, formatting important information into a spreadsheet format. These entries are combined with the most recent set of file location entries, checked for errors, then stored as both a spreadsheet of file locations as well as an usable spreadsheet of corresponding values to entry inputs.

Database Structure

Databases store information in the form of tables. Tables are organized identically to spreadsheets in the form of rows and columns and can be used to link identifying information with specific values or other associated information. A key advantage of using databases compared to

spreadsheet programs is scalability: additional data points can be trivially added at any point and calculations such as means will be instantly updated to reflect newly added data points. It is typical to store most data into a single table and pull needed information from there, but there are situations where multiple tables of data are more effective, such as when experiments are conducted using both fibroblasts and cardiomyocytes. Another column can be included in the data pipeline listing which cell line the data value comes from, or alternatively store all fibroblast data in one table and all cardiomyocyte data in another table. The same principle applies when having a single database with multiple tables or multiple databases with single tables. Keeping different classes of information separate from each other is critical for ensuring data purity and avoiding confusion when using queries. Further reading about database basics can be found online [19]. Based on our experience, it is highly recommended that databases to be built off of existing examples instead of being constructed from scratch.

Figure 2.3 illustrates how databases can use relationships between data to pull associated information from other tables. In this figure, both the Table View (what database interfaces look like) and the Design View (logic relationships between information) of MS Access are shown.

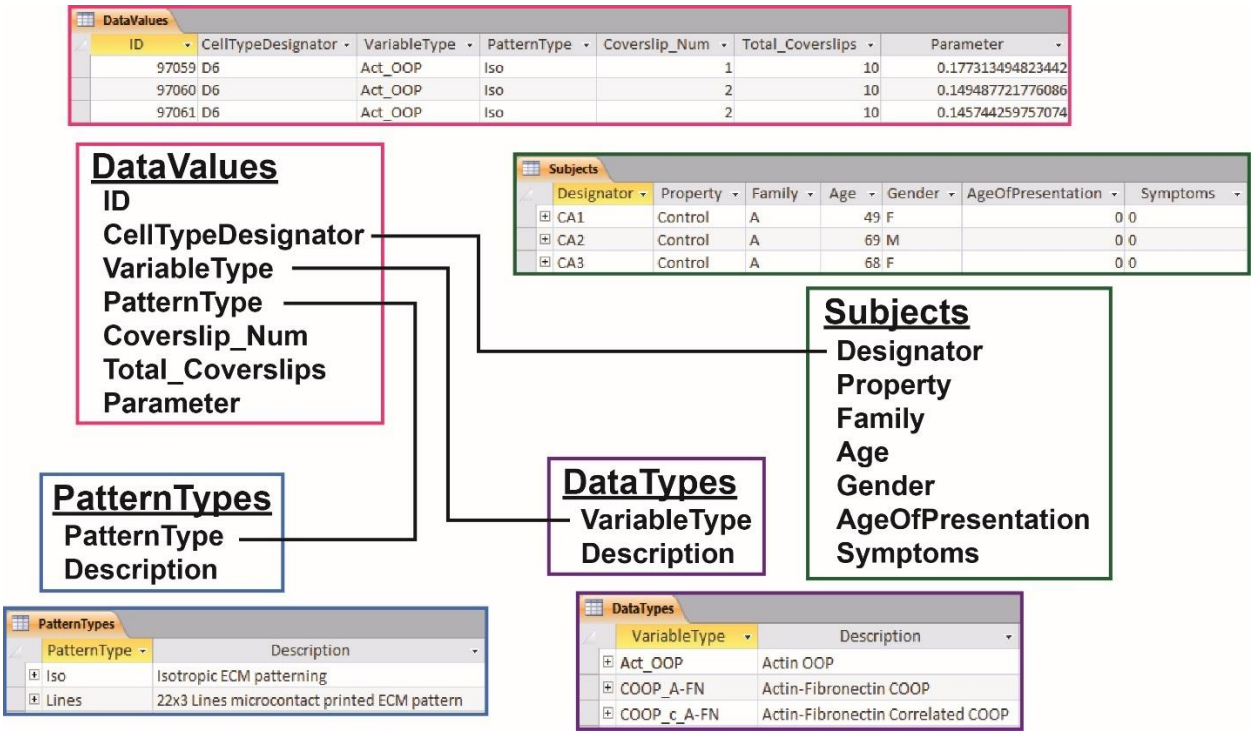

Figure 2.3: Table and Design View relationships within LMNA Mutation data set. Relational databases have the advantage of linking fields in one table with information in another table, which allows for immediate interchangeability of aggregation. The example here visually demonstrate how differing information can be linked.

MS Access SQL & Query Creation Examples

If tables store information in databases, then queries are requests to the database for information given specific criteria. These requests can be coded using structured query language (SQL)

although in MS Access is also possible to use a form-like view to request the information as well. Figure 2.4 shows a sample query using SQL syntax. This syntax underlies the data shown Figure 5b & 5d.

```
SELECT Cell_Lines.Property, DataValues.DataVar, Avg(DataValues.Parameter) AS Average,
StDev(DataValues.Parameter) AS StDev, Count(DataValues.Parameter) AS Count
```

SELECT calls designated fields from different tables. Calculations such as Avg happen here, and AS can rename the query's fields

```
FROM Pattern_type RIGHT JOIN (Data_Types RIGHT JOIN (Cell_Lines RIGHT JOIN DataValues
ON Cell_Lines.Designator = DataValues.CellTypeDesignator) ON Data_Types.Variable_Name =
DataValues.DataVar) ON Pattern_Type.Pat_Name = DataValues.PatVar
```

FROM selects the tables to call fields from. Nested loops using JOIN and ON statements can be used to require entry matches between tables.

```
GROUP BY Cell_Lines.Property, DataValues.DataVar, Pattern_Type.Pat_Name
```

GROUP BY lists criteria for how the data points should be grouped together, by order of priority.

```
HAVING ((Cell_Lines.Property = "Patient") OR (Cell_Lines.Property = "Control) OR
(Cell_Lines.Property = "Donor") OR (Cell_Lines.Property = "Progeria")) AND
(DataValues.DataVar = "Act_OOP")
```

HAVING sets requirements on data points. WHERE is instead used for non-aggregating queries (no calculations or GROUP BY criteria).

```
ORDER BY Cell_Lines.Property;
```

ORDER BY organizes the query output by order of priority.

Figure 2.4: An example query using SQL syntax. SELECT and FROM statements are the only requirements to generate a query, but often include additional commands and criteria. GROUP BY provides clarification on how aggregate the data, HAVING or WHERE statements limit the output to data that meets specific criteria, and ORDER BY indicates the order the outputs should be arranged by.

SQL requires SELECT and FROM statements to denote which columns ("fields") and tables to use in a query. Calculations such as averages, standard deviations, and counts can also be performed on the data within the SELECT statement. Additional criteria can be added in WHERE, HAVING, and GROUP BY statements to allow for selection from the data. ORDER BY in comparison simply lists the query outputs in a fashion better organized to the user's needs.

Identifying A Novel Relationship in LMNA Mutation Data Set

When given multitude of possible conformations, it can be difficult to identify where novel relationships exist using manual data aggregation methods. In our specific context, we were interested in determining whether the organization of subcellular actin filaments, measured using a mathematical construct called Orientational Order Parameter (OOP), varied between the different condition groups of LMNA Mutation by different Family groups, Negative Controls, and positive control of Hutchinson-Gilford Progeria (HGPS) [20]. Higher OOP values denote a higher degree of organization within the tissue. Through using our database, Figure 2.5 was easily

constructed in addition to dozens of other plots exploring different combinations of groupings against measurements.

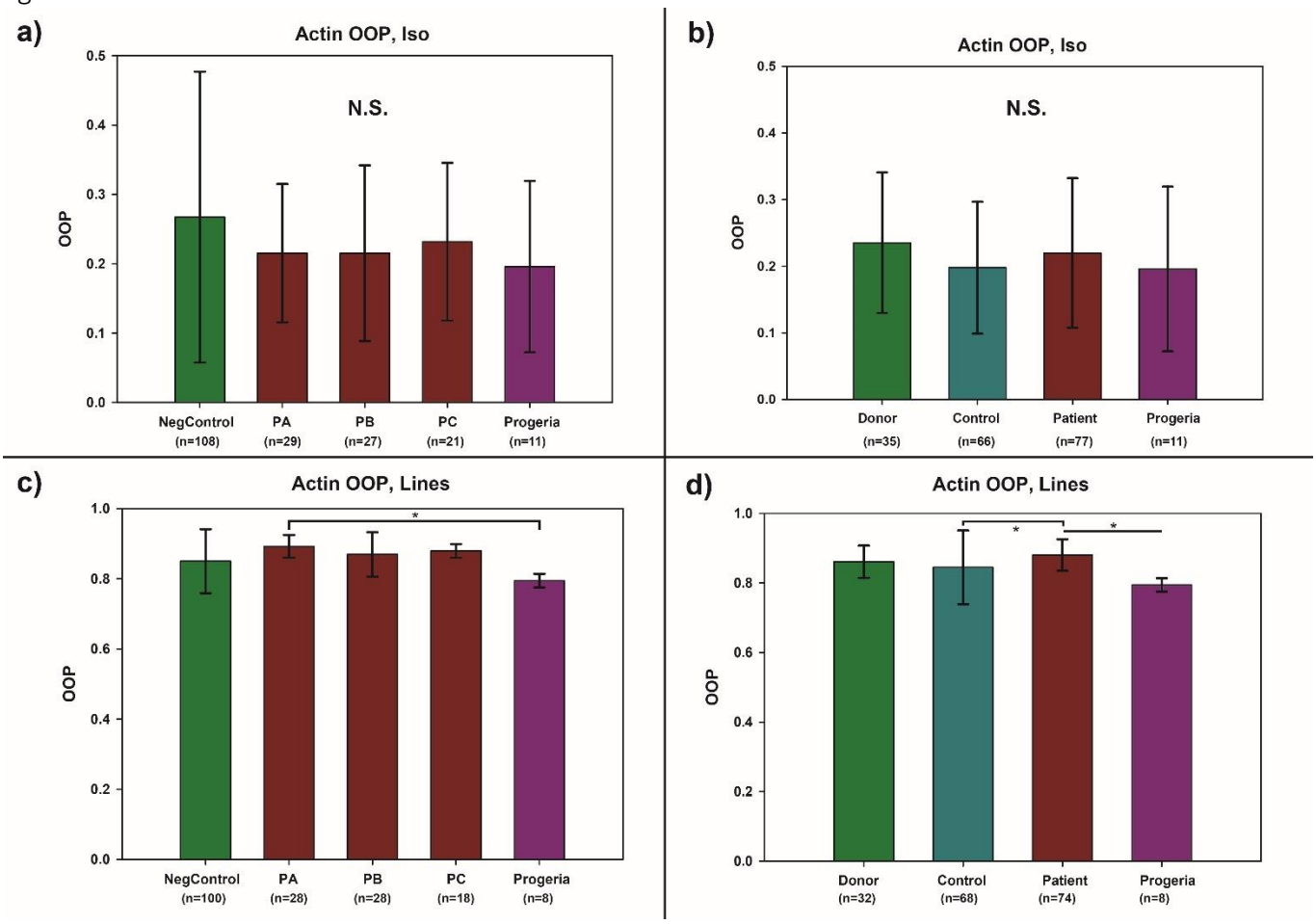

Figure 2.5: Comparison between groups for the variable Actin OOP. In a) and c) the groups are NegControl (a combination of related, "Control", and unrelated, "Donor" non-mutation individuals), HGPS (progeria as a positive control), PA (LMNA mutation patients from Family A), PB (Family B), and PC (Family C). In b) and d), the groups are NegControl split into its two components ("Donor" and "Control"), all Patients merged into the same group, and HGPS. Statistical significance is only found in the Actin OOP Lines condition.

Although both HGPS and LMNA are both laminopathies that affect cell nuclear envelope stability, the families exhibit symptoms only in the heart whereas HGPS patients are systemically affected across the body [14-16]. Since fibroblasts are the primary component of human skin, it was expected that there would be decreased tissue organization between at least one of the LMNA families and HGPS in anisotropically patterned experimental conditions.

Chapter 3: Discussion

Expanding database usage to a greater scientific spectrum is important as the need for increased data rigor expands and spreadsheet methods cause more uncaught errors, such as erroneous gene name conversions [21]. Databases are frequently used and cited in the literature, especially regarding their usage for clinical patient populations [8, 22, 23], and several have already been constructed for specific purposes and fields, like the Rat Genome Database curation tools available online [24]. While using databases for information purposes is hardly a novel concept, to our knowledge no prior methods papers have called attention to the benefits of databases and provided generalized examples of how they can be constructed. The published articles that do exist are frequently specific to only the clinical domain [8] or large genomic databases [24], not to specific lab purposes. Thus, we are among the first to create and report on a method that uses a data pipeline to convert raw data files into structured formats and then utilizes databases for scalably aggregating data into a variety of conformations.

The advantages of our methodology are many. They include the ability to employ automated methods for converting raw data into structured formats, ease of use once stored inside the database, and constant re-updating and re-aggregation of datasets. With our system, it is also possible to pull multiple variables' worth of information from a single data file and automate the data pipeline to do so when prompted. We also used commonly available and economical software, Matlab and MS Access, to achieve results, demonstrating that expensive and niche software packages are not mandatory in achieving a functional database. Given the limited reach of most laboratories' research funds, the ability to increase the efficiency of database management is a priceless commodity.

Another strong benefit of this method is the increased rigor of data analysis. The recent importance of increasing the reproducibility of data has been highlighted by the NIH as well as other scientists [25, 26]. Being able to identify individual data points and easily find corresponding source files allows us to be more rigorous with our data, in addition to automatic screening of errors during the data pipeline process. It is good practice to keep all generated structured formats as a data set is expanded by additional experimental data, in case issues occur or older versions want to be checked. Working in non-destructive ways and leaving past versions of the same documents within the data pipeline allow for better troubleshooting.

One disadvantage of our system is that if data needs are changed, updates are often made reactively instead of proactively. Going back to make systemic changes to the pipeline process, where columns and information inputs are changed, can require significant time and testing prior to usage. This issue can be avoided by implementing more effective organization of data from the start of research, making its collection and management more of a proactive process. In addition, initially constructing the data pipeline and learning SQL can be rather time inefficient. For those without a strong coding background, progress is often slow and it is possible to make programming errors or be unsure where to begin. Frequent student turnover in academic labs due to the nature of educational setting can make guaranteeing continuity in the system after its creator leaves more difficult, but can be overcome with creation of guidelines and implementation steps during

setup. Individuals without coding knowledge can use existing systems to fit specific needs, but blindly using codes for new purposes without modification can cause problems without proper mentoring or training on its use. However, good lab buy-in and robust systems can outlast their creators and help facilitate for multiple students and many years of projects, and there is high value for all researchers in having a well-functioning joint data pipeline and database system.

Another consideration is that, although databases are scalable, scalability is a relative concept to the needs being addressed. MS Access is intended for thousands of data points and low-scale data access, with database memory sizes limited to 2 GB, but in comparison some company-level database systems can store millions of data points and process millions of transactions simultaneously [27]. Switching database software or systems to heavy-duty database tools such as Oracle Database or MS SQL Server are potential solutions to these problems, but difficult to switch to once a system has been constructed since it would involve starting over from scratch. Utilizing different data types to store the information can also be further optimized, but inefficient default settings are often used instead [28]. Matching specific project needs to optimal data pipeline structures and database software can also be difficult, as many academic labs often have multiple projects underway simultaneously and different logic structures will be required for different needs. Often projects of related needs can be adapted to fit each other though, building off the same base structure. Using databases still inherently decreases the amount of direct human interaction with the data and consequently possibility for uncaught error though, leaving databases superior to spreadsheet methods for multidimensional uses.

In the specific context given, not much is left to be done on the fibroblast database as the relevant experiments have concluded and data fully analyzed. Other studies being conducted will continue to use slightly modified processes to accomplish similar results, however, from the quantification of cardiomyocyte structural behaviors as well as electrical measurements stored in vectors from contractility assays. Specifics of the current Matlab codes used need to be changed, to reflect new project needs and variables stored. Ultimately these methods will likely be applied to all ongoing projects in our lab that involve quantifying variables then needing to look at relationships between those variables and patient-specific attributes.

In conclusion, as scientific data sets become more complex, databases become increasingly more important for the scientific community and have great potential to be as commonplace as and more effective than current widespread MS Excel usage for data storage. Issues with data transparency and replicability in science will only continue to expand in the future as data sets grow in size and complexity, highlighting the importance of more widespread adoption of databases and automated data pipeline methods for general scientific needs now and into the future.

REFERENCES

1. Cavin, R.K., P. Lugli, and V.V. Zhirnov, *Science and engineering beyond Moore's law.* Proceedings of the IEEE, 2012. **100**(Special Centennial Issue): p. 1720-1749.
2. Mast, F.D., A.V. Ratushny, and J.D. Aitchison, *Systems cell biology.* J Cell Biol, 2014. **206**(6): p. 695-706.
3. Barone, L., J. Williams, and D. Micklos, *Unmet needs for analyzing biological big data: A survey of 704 NSF principal investigators.* PLoS Computational Biology, 2017. **13**(10): p. e1005755.
4. Gandomi, A. and M. Haider, *Beyond the hype: Big data concepts, methods, and analytics.* International Journal of Information Management, 2015. **35**(2): p. 137-144.
5. Siddiqa, A., et al., *A survey of big data management: Taxonomy and state-of-the-art.* Journal of Network and Computer Applications, 2016. **71**: p. 151-166.
6. Anderson, C., *The End of Theory: The Data Deluge Makes the Scientific Method Obsolete*, in *Wired Magazine*. 2008.
7. Broman, K.W. and K.H. Woo, *Data organization in spreadsheets.* The American Statistician, 2017(just-accepted).
8. Lee, H., et al., *How I do it: a practical database management system to assist clinical research teams with data collection, organization, and reporting.* Academic radiology, 2015. **22**(4): p. 527-533.
9. Bassil, Y., *A comparative study on the performance of the Top DBMS systems.* arXiv preprint arXiv:1205.2889, 2012.
10. *Learn SQL - Codecademy*. [Web Page] 2018 May 14, 2018]; Available from: https://www.codecademy.com/learn/learn-sql.
11. *SQL Tutorial - w3schools.com*. [Web Page] 2018 May 14, 2018]; Available from: https://www.w3schools.com/sql/.
12. *Introduction to SQL - SQLBolt*. [Web Page] 2018 May 14, 2018]; Available from: https://sqlbolt.com/.
13. Core, J.Q., et al., *Age of heart disease presentation and dysmorphic nuclei in patients with LMNA mutations.* PloS one, 2017. **12**(11): p. e0188256.
14. Zaragoza, M.V., et al., *Exome Sequencing Identifies a Novel LMNA Splice-Site Mutation and Multigenic Heterozygosity of Potential Modifiers in a Family with Sick Sinus Syndrome, Dilated Cardiomyopathy, and Sudden Cardiac Death.* PLoS One, 2016. **11**(5): p. e0155421.
15. Zaragoza, M.V., et al., *Dupuytren's and Ledderhose Diseases in a Family with LMNA-Related Cardiomyopathy and a Novel Variant in the ASTE1 Gene.* Cells, 2017. **6**(4): p. 40.
16. Zaragoza, M., et al., *Heart-hand syndrome IV: a second family with LMNA-related cardiomyopathy and brachydactyly.* Clinical genetics, 2017. **91**(3): p. 499-500.
17. Eriksson, M., et al., *Recurrent de novo point mutations in lamin A cause Hutchinson–Gilford progeria syndrome.* Nature, 2003. **423**(6937): p. 293.
18. *SQL Training and Tutorials - Lynda.com*. [Web Page] 2018 May 14, 2018]; Available from: https://www.lynda.com/SQL-training-tutorials/446-0.html.
19. *Definition of database (DB)*. [Web Page] 2018 May 14, 2018]; Available from: https://searchsqlserver.techtarget.com/definition/database.

20. Hamley, I.W., *Introduction to soft matter: synthetic and biological self-assembling materials*. 2013: John Wiley & Sons.
21. Ziemann, M., Y. Eren, and A. El-Osta, *Gene name errors are widespread in the scientific literature.* Genome biology, 2016. **17**(1): p. 177.
22. Wardle, M. and M. Sadler, *How to set up a clinical database.* Pract Neurol, 2016. **16**(1): p. 70-4.
23. Kerr, W.T., et al., *The future of medical diagnostics: large digitized databases.* The Yale journal of biology and medicine, 2012. **85**(3): p. 363.
24. Laulederkind, S.J., et al., *The Rat Genome Database curation tool suite: a set of optimized software tools enabling efficient acquisition, organization, and presentation of biological data.* Database, 2011. **2011**.
25. (NIH), N.I.o.H. *Rigor and Reproducibility*. [Web Page] 2018 May 14, 2018]; Available from: https://grants.nih.gov/reproducibility/index.htm.
26. Hofseth, L.J., *Getting rigorous with scientific rigor.* Carcinogenesis, 2017. **39**(1): p. 21-25.
27. Hey, T.T., A, *The Data Deluge: An e-Science Perspective*, in *Grid Computing: Making the Global Infrastructure a Reality*. 2003, John Wiley & Sons, Ltd.
28. Savage, B. *Designing Databases: Picking The Right Data Types*. 2009 May 14, 2018]; Available from: https://www.brandonsavage.net/designing-databases-picking-the-right-data-types.