Title
Databases to Efficiently Manage Medium Sized, Low Velocity, Multidimensional Data in Tissue Engineering.

Journal
Journal of Visualized Experiments, 2019(153)

ISSN
1940-087X

Authors
Ochs, Alexander R
Mehrabi, Mehrsa
Becker, Danielle
et al.

Publication Date
2019

DOI
10.3791/60038

Peer reviewed

HHS Public Access

Author manuscript

J Vis Exp. Author manuscript; available in PMC 2020 November 22.

Databases to Efficiently Manage Medium Sized, Low Velocity, Multidimensional Data in Tissue Engineering

Alexander R. Ochs[1,2], **Mehrsa Mehrabi**[1,2], **Danielle Becker**[1,2], **Mira N. Asad**[1,2], **Jing Zhao**[1,2], **Michael V. Zaragoza**[3,4], **Anna Grosberg**[1,2,5,6,7]

[1]Department of Biomedical Engineering, University of California, Irvine

[2]The Edwards Lifesciences Center for Advanced Cardiovascular Technology, University of California, Irvine

[3]Pediatrics-Genetics & Genomics Division-School of Medicine, University of California, Irvine

[4]Biological Chemistry-School of Medicine, University of California, Irvine

[5]Department of Chemical and Biomolecular Engineering, University of California, Irvine

[6]Center for Complex Biological Systems, University of California, Irvine

[7]The NSF-Simons Center for Multiscale Cell Fate Research (CMCF), University of California, Irvine

Abstract

Science relies on increasingly complex data sets for progress, but common data management methods such as spreadsheet programs are inadequate for the growing scale and complexity of this information. While database management systems have the potential to rectify these issues, they are not commonly utilized outside of business and informatics fields. Yet, many research labs already generate “medium sized”, low velocity, multi-dimensional data that could greatly benefit from implementing similar systems. In this article, we provide a conceptual overview explaining how databases function and the advantages they provide in tissue engineering applications. Structural fibroblast data from individuals with a lamin A/C mutation was used to illustrate examples within a specific experimental context. Examples include visualizing multidimensional data, linking tables in a relational database structure, mapping a semi-automated data pipeline to convert raw data into structured formats, and explaining the underlying syntax of a query. Outcomes from analyzing the data were used to create plots of various arrangements and significance was demonstrated in cell organization in aligned environments between the positive control of Hutchinson-Gilford progeria, a well-known laminopathy, and all other experimental groups. In comparison to spreadsheets, database methods were enormously time efficient, simple to use once set up, allowed for immediate access of original file locations, and increased data rigor. In response to the National Institutes of Health (NIH) emphasis on experimental rigor, it is likely

Correspondence to: Anna Grosberg at grosberg@uci.edu.

Video Link

The video component of this article can be found at https://www.jove.com/video/60038/

Disclosures

The authors have nothing to disclose.

that many scientific fields will eventually adopt databases as common practice due to their strong capability to effectively organize complex data.

Keywords

medium sized data; databases; LMNA; data organization; multidimensional data; tissue engineering

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 underscores 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 remains critical for efficient workflow and output. Bridging the 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 data characterized by huge volumes, large heterogeneity, and high rates of generation, such as audio and video[4,5]. Using automated methods of organization and analysis is mandatory for this field to appropriately handle torrents of data. Many technical terms used in big data are not clearly defined, however, and can be confusing; for instance, "high velocity" data is often associated with millions of new entries per day whereas "low velocity" data might only be hundreds of entries per day, such as in an academic lab setting. Although there are many exciting findings yet to be discovered using big data, most academic labs do not require the scope, power, and complexity of such methods for addressing their own scientific questions[5]. While it is undoubtable that scientific data grows increasingly complex with time[6], many scientists continue to use methods of organization that no longer meet their expanding data needs. For example, convenient spreadsheet programs 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]. Conversely, databases are an effective solution to the problem as they are scalable, relatively cheap, and easy to use in handling varied data sets of ongoing projects.

Immediate concerns that arise when considering schemas of data organization are cost, accessibility, and time investment for training and usage. Frequently used in business settings, database 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 Microsoft (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)[10,11,12]. Like any programming language, learning how to use databases and code using SQL takes time to master, but with the ample resources available the process is straightforward and well worth the effort invested.

Databases can be powerful tools for increasing data accessibility and ease of aggregation, but it is important to discern which data would most benefit from a greater control of organization. Multi-dimensionality refers to the number of conditions that a measurement can be grouped against, and databases are most powerful when managing many different conditions[13]. Conversely, information with low dimensionality is simplest to handle using a spreadsheet program; for example, a data set containing years and a value for each year has only one possible grouping (measurements against years). High dimensional data such as from clinical settings would require a large degree of manual organization in order to effectively maintain, a tedious and error-prone process beyond the scope of spreadsheet programs[13]. Non-relational (NoSQL) databases also fulfill a variety of roles, primarily in applications where data does not organize well into rows and columns[14]. In addition to being frequently open source, these organizational schemas include graphical associations, time series data, or document-based data. NoSQL excels at scalability better than SQL, but cannot create complex queries, so relational databases are better in situations that require consistency, standardization, and infrequent large-scale data changes[15]. Databases are best at effectively grouping and re-updating data into the large array of conformations often needed in scientific settings[13,16].

The main intent of this work, therefore, is to inform the scientific community about the potential of databases as scalable data management systems for “medium sized”, low velocity data as well as to provide a general template using specific examples of patient sourced cell-line experiments. Other similar applications include geospatial data of river beds, questionnaires from longitudinal clinical studies, and microbial growth conditions in growth media[17,18,19]. This work highlights common considerations for and 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 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. This information provides context, commentary, and templates for assisting fellow scientists on the path towards implementing databases for their own experimental needs.

For the purposes of creating a scalable database in a research laboratory setting, data from experiments using human fibroblast cells was collected over the past three years. The primary focus of this protocol is to report on the organization of computer software to enable the user to aggregate, update, and manage data in the most cost- and time-efficient manner possible, but the relevant experimental methods are provided as well for context.

Experimental setup

The experimental protocol for preparing samples has been described previously[20,21], and is presented briefly here. Constructs were prepared by spin-coating rectangular glass coverslips with a 10:1 mixture of polydimethylsiloxane (PDMS) and curing agent, then applying 0.05 mg/mL fibronectin, in either unorganized (isotropic) or 20 μm lines with 5 μm gap micropatterned arrangements (lines). Fibroblast cells were seeded at passage 7 (or passage 16 for positive controls) onto the coverslips at optimal densities and left to grow for 48 h with media being changed after 24 h. The cells were then fixed using 4% paraformaldehyde (PFA) solution and 0.0005% nonionic surfactant, followed by the coverslips being immunostained for cell nuclei (4',6'-diaminodino-2-phenylinodole [DAPI]), actin (Alexa Fluor 488 phalloidin), and fibronectin (polycloncal rabbit anti-human fibronectin). A secondary stain for fibronectin using goat anti-rabbit IgG antibodies (Alexa Fluor 750 goat anti-rabbit) was applied and preservation agent was mounted onto all coverslips to prevent fluorescent fading. Nail polish was used to seal coverslips onto microscope slides then left to dry for 24 h.

Fluorescence images were obtained as described previously[20] using a 40x oil immersion objective coupled with a digital charge coupled device (CCD) camera mounted on an inverted motorized microscope. Ten randomly selected fields of view were imaged for each coverslip at 40x magnification, corresponding to a 6.22 pixels/μm resolution. Custom-written codes were used to quantify different variables from the images describing the nuclei, actin filaments, and fibronectin; corresponding values, as well as organization and geometry parameters, were automatically saved in data files.

Cell lines

More extensive documentation on all sample data cell lines can be found in prior publications[20]. To describe briefly, the 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 lamin A/C (*LMNA*) gene mutation: heterozygous *LMNA* splice-site mutation (c.357–2A>G)[22] (family A); *LMNA* nonsense mutation (c.736 C>T, pQ246X) in exon 4[23] (family B); and *LMNA* missense mutation (c.1003C>T, pR335W) in exon 6[24] (family C). Fibroblast cells were also collected from other individuals in each family as related mutation-negative controls, referred to as "Controls", and others were purchased as unrelated mutation-negative controls, referred to as "Donors". 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[25]. In total, fibroblasts from 22 individuals were tested and used as data in this work.

Data types

Fibroblast data fell into one of two categories: cellular nuclei variables (i.e., percentage of dysmorphic nuclei, area of nuclei, nuclei eccentricity)[20] or structural variables stemming from the orientational order parameter (OOP)[21,26,27] (i.e., actin OOP, fibronectin OOP, nuclei OOP). This parameter is equal to the maximum eigenvalue of the mean order tensor

of all the orientation vectors, and it is defined in detail in previous publications[26,28]. These values are aggregated into a variety of possible conformations, such as values against age, gender, disease status, presence of certain symptoms, etc. Examples of how these variables are used can be found in the results section.

Example codes and files

The example codes and other files based on the data above can be downloaded with this paper, and their names and types are summarized in Table 1.

Protocol

NOTE: See Table of Materials for the software versions used in this protocol.

1. Evaluate if the data would benefit from a database organization scheme

1. Download the example codes and databases (see Supplemental Coding Files, which are summarized in Table 1).
2. Use Figure 1 to evaluate if the data set of interest is “multi-dimensional”.

   NOTE: Figure 1 is a graphical representation of a multi-dimensional database provided for the example data set.
3. If the data can be visualized in a “multi-dimensional” form like the example and if the ability to relate a specific experimental outcome to any of the dimensions (i.e., conditions) would allow for greater scientific insight into the available data, proceed to construct a relational database.

2. Organize the database structure

NOTE: Relational databases store information in the form of tables. Tables are organized in schema of rows and columns, similar to spreadsheets, and can be used to link identifying information within the database.

1. Organize the data files, so they have well thought out unique names. Good practice with file naming conventions and folder-subfolder structures, when done well, allow for broad database scalability without compromising the readability of accessing files manually. Add date files in a consistent format, such as “20XX-YY-ZZ”, and name subfolders according to metadata is one such example.
2. As the data-base structure is designed, draw relationships between the fields in different tables. Thus, multi-dimensionality is handled by relating different fields (i.e., columns in the tables) in individual tables to each other.
3. Create readme documentation that describes the database and relationships that were created in step 2.2. Once an entry between different tables is linked, all associated information is related to that entry and can be used to call complex queries to filter down to the desired information.

NOTE: Readme documents are a common solution for providing supplemental information and database structural information about a project without adding non-uniform data to the structure.

4. Following steps 2.1–2.3, make the end result similar to this example where the differing characteristics of individuals (Figure 2A) are related to associated experimental data of those individuals (Figure 2B). The same was done through relating columns of pattern types (Figure 2C) and data types (Figure 2D) to matching entries in the main data values table to explain various shorthand notations (Figure 2B).

5. Determine all the essential and merely helpful data points that need to be recorded for long range data collection.

   NOTE: A key advantage of using databases over spreadsheet programs, as mentioned earlier, is scalability: additional data points can be trivially added at any point and calculations, such as averages, are instantly updated to reflect newly added data points.

   1. Identify the necessary information for creating distinct data points prior to beginning. Leave raw data untouched, instead of modifying or saving over it, so that reanalysis is possible and accessible.

      NOTE: For the given example (Figure 2), the “Designator” corresponding to an individual, “Pattern type”, “Coverslip #”, and “Variable type” were all vital fields for distinctness of the associated value.

   2. If desired, add other helpful, non-vital information such as the “Total # of Coverslips” to indicate the number of repetitions conducted and help determine if data points are missing in this example.

3. Set up and organize the pipeline

1. Identify all the various experiments and data analysis methods that might lead to data collection along with the normal data storage practices for each data type. Work with open source version control software such as GitHub to ensure necessary consistency and version control while minimizing user burden.

2. If possible, create procedure for consistent naming and storing of data to allow for an automated pipeline.

   NOTE: In the example, outputs were all consistently named, thus creating a data-pipeline that looked for specific attributes was straightforward once the files were selected. If consistent naming is not possible, the tables in the database will need to be populated manually, which is not recommended.

3. Use any convenient programming language to generate new data entries for the database.

   1. Create small “helper” tables (files #8–#10 in Table 1) in separate files that can guide automated selection of data. These files serve as a

template of possibilities for the pipeline to operate under and are easy to edit.

2. To generate new data entries for the data-pipeline (Figure 3D), program the code (LocationPointer.m, file #1 in Table 1) to use the helper tables as inputs to be selected by the user (files #8–#10 in Table 1).

3. From here, assemble a new spreadsheet of file locations by combining the new entries with the previous entries (Figure 3E). Create a code to automate this step as shown in LocationPointerCompile.m (file #2 in Table 1).

4. Afterwards, check this merged spreadsheet for duplicates, which should be automatically removed. Create a code to automate this step as shown in LocationPointer_Remove_Duplicates.m (file #3 in Table 1).

5. Additionally, check the spreadsheet for errors, and notify the user of their reason and location (Figure 3F). Create a code to automate this step as shown in BadPointerCheck.m (file #4 in Table 1). Alternatively, write a code that will check the compiled database and identify duplicates in one step as shown in LocationPointer_Check.m (file #5 in Table 1).

6. Create a code to let the user manually remove bad points without losing the integrity of the database as shown in Manual_Pointer_Removal.m (file #6 in Table 1).

7. Then use the file locations to generate a data value spreadsheet (Figure 3G, file #12 in Table 1) as well as to create a most updated list of entries that can be accessed to identify file locations or merged with future entries (Figure 3H). Create a code to automate this step as shown in Database_Generate.m (file #7 in Table 1).

4. Double check that the pipeline adds to the experimental rigor by checking for inclusion of rigorous naming conventions, automated file assembly codes, and automated error checks as previously described.

4. Create the database and queries

NOTE: If tables store information in databases, then queries are requests to the database for information given specific criteria. There are two methods to create the database: starting from a blank document or starting from the existing files. Figure 4 shows a sample query using SQL syntax that is designed to run using the database relationships shown in Figure 2.

1. Method 1: Starting from scratch in creating the database and queries

    1. Create a blank database document.

    2. Load the helper tables (files #8–#10 in Table 1) by selecting **External Data** | **Text File Import** | **Choose File** (files #8–#10) | **Delimited** | **First Row Contains Headers, Comma** | **leave default** | **Choose My**

Own Primary Key (Designator for Cell Lines File #8, Variable Name for Data Types File #9, Pat Name for Pattern Type File #10) | **leave default** | **Finish**.

3. Load the Data value table (file #12 in Table 1) by selecting **External Data** | **Text File Import** | **Choose File** (file #12) | **Delimited** | **First Row Contains Headers, Comma** | **leave default** | **Let Access Add primary key** | **Import to Table: DataValues** | **Finish**.

4. Create the relationships by selecting **Database Tools** | **Relationships** | **Drag all Tables to the board** | **Edit Relationships** | **Create New** | **Match the DataValue fields with Helper Tables Designators** | **Joint Type 3**.

5. Select **Create** | **Query Design**.

6. Select or drag all relevant tables into the top window. In this example ‘Cell Lines’, ‘Data Values’, ‘Data Types’, and ‘Pattern Type’. The relationships should automatically set up based on the previous Relationship design.

7. Fill out the query columns for desired results, for example:

    1. Click on **Show** | **Totals**.

    2. Fill out the first column (Table: DataValues, Field: DataVar, Total: GroupBy, Criteria: “Act_OOP”), the second column (Table: DataValues, Field: PatVar, Total: GroupBy, Criteria: “Lines”), and the third column (Table: Cell_Lines, Field: Designator, Total: GroupBy, Sort: Ascending).

    3. Fill out the fourth column (Table: DataValues, Field: Parameter, Total: Ave), the fifth column (Table: DataValues, Field: Parameter, Total: StDev), and the sixth column (Table: DataValues, Field: Parameter, Total: Count).

8. Run the query.

2. Alternatively, use the provided example database as a basis for examples. Open the database file Database_Queries.accdb (file #13 in Table 1) that was downloaded earlier. Use it as a template by replacing existing tables with the data of interest.

5. Move the output tables to a statistical software for significance analysis

1. For this sample experimental data, use the one-way analysis of variance (ANOVA) using Tukey’s test for mean comparisons between various conditions.

    NOTE: Values of $p < 0.05$ were considered statistically significant.

Representative Results

Multi-dimensionality of the data

In the context of the example data-set presented here, the subjects, described in the Methods section, were divided into groups of individuals from the three families with the heart disease-causing *LMNA* mutation ("Patients"), related non-mutation negative controls ("Controls"), unrelated non-mutation negative controls ("Donors"), and an individual with Hutchinson-Gilford progeria syndrome (HGPS) as a positive control[20]. Results from Controls and Donors could be further grouped together as an overall Negative Control (N.C.) group, given their collective lack of *LMNA* mutations. Every subject's cell line had a "Mutation Status" associated with it, based on their condition group (Figure 1 – dark blue axis). For each experiment, fibroblast cells from the subjects were cultured on arrangements of either unorganized (Isotropic) or micropatterned (Lines) fibronectin, creating the condition of "Pattern type" (Figure 1 – orange axis). After the cells were fixed, immunostained, and imaged, the "Coverslip #" was transcribed, since multiple experiments (i.e., technical replicates) would occur using the same individual's cells (Figure 1 – light green axis). Custom MATLAB codes[20,21] were then used to quantify different aspects of cell nuclei or tissue organization variables as "Variable type" (Figure 1 – teal green axis). The three factors were associated with the cells' human source and consequently linked to the "Family" (Figure 1 – dark pink axis) and "Age at time of biopsy" (Figure 1 – dark green axis) in addition to "Mutation Status." Other dimensions not included in Figure 1 were the "Age of presentation," "Symptoms," "Designator," and "Gender" of the individual in question. The example provided here results in at least ten possible dimensions for data aggregation. Thus this example data is a prime candidate for organization by relational databases.

Organizing the pipeline

Up to an estimated 95% of all digital data is unstructured[4], but structured formats are required for databases. Still, creating a good automated method for the data-pipeline is highly context dependent.

For this example, the images collected from each experiment were stored in folders named by date and initial of the lab member responsible, with sub-folders listing the subject and coverslip number. Pipeline files are provided in the Supplemental Materials section, as well as summarized in a flow chart illustration (Figure 3). Different metrics from various experimental conditions across a variety of subjects were quantified from these fluorescent images (Figure 3A) using custom codes (Figure 3B)[20,21]. For example, actin orientational order parameter[21] was extracted from tissues stained with phalloidin (Figure 3A) and used to compare the organization of fibroblasts from different individuals. The code outputs were saved in the same folder as the source images (Figure 3C).

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 this specific context, we were

interested in comparing the organization of subcellular actin filaments across multiple conditions, measured using the OOP[27].

OOP is a mathematical construct quantifying the degree of order in anisotropic environments, normalized to zero corresponding to completely isotropic tissue and one corresponding to completely aligned tissue. The data set was first split up by pattern type as lines (Figure 5A) and isotropic (Figure 5B) conditions, which were expected to have vastly different OOPs since fibronectin micropatterning heavily influences tissue organization. There were no significant differences between conditions when comparing isotropic tissues (Figure 5B). Conversely, the patterned tissues were statistically less organized in the positive control cell line (HGPS) (Figure 5A), and this relationship held even when the data was aggregated into different groups (Figure 5C). Actin OOP was additionally plotted against individuals' age at time of biopsy (Figure 5D), separated by mutation status and family, to illustrate aggregation against a clinical variable. Unlike with nuclear defects[20], there is no correlation between actin organization and an individual's age (Figure 5D). Ultimately, the plots shown in Figure 5 illustrate how the same data can be analyzed in different combinations and how easily the normally difficult task of aggregating data that falls under multiple classes can be accomplished using databases.

For this article, data from patient sourced fibroblasts were compared between conditions to determine mutation consequences. Although both HGPS and the three families in this study have *LMNA*-linked diseases that potentially disrupt the nuclear envelope, the patients exhibit symptoms primarily associated with heart dysfunction whereas HGPS individuals have multiple organ systems affected[22,23,24]. Indeed, despite the micropatterned environment cells originating from an HGPS patient had a statistically lower actin OOP value than any of the other cell lines considered (Figure 5A,C). This dovetails with HGPS patients being the only ones in the study with any skin abnormalities caused by the mutation. Viewing the same data in different conformations is also helpful for providing additional insight and avenues into scientific inquiry in a varied data set (Figure 5).

Discussion

Technical discussion of the protocol

The first step when considering the use of databases is to evaluate if the data would benefit from such an organization.

The next essential step is to create an automated code that will ask the minimum input from the user and generate the table data structure. In the example, the user entered the category of data type (cell nuclei or structural measurements), cell lines' subject designator, and number of files being selected. The relevant files were then selected by the user (Table 2, column 1), with the row entries being automatically created and populated with all variables contained within the file (Table 2, column 2). Furthermore, it is important the code is flexible so that if another experimental entry needs to be added, the user can select to continue the loop; if not, the files are saved and the loop ends. The basic functions of adding new entries, checking for errors, and assembling the spreadsheet from file locations described in this step are all critical for an efficient data-pipeline setup.

It is imperative to note that using file locations when creating the data-pipeline increases experimental rigor. Specifically, having a corresponding spreadsheet listing all file locations for the data values allows a user to backtrack any data point back to the lab notebook of the researcher who collected the raw data. When dealing with hundreds to tens of thousands of data points, greater transparency and accessibility is invaluable over the lifetime of a project. It is highly recommended that users consider saving file locations first and later compiling values for data instead of only storing the data values.

Once the database is created, the simplest way to get started is by programming the queries through the design view. The user will find it useful to download the provided template (file #13 in Table 1) as a starting point. Alternatively, these can be programed directly through SQL language (Figure 4).

Scientific discussion

The purpose of this article was to disseminate methods involving a data-pipeline and database that elucidated data set scalability and transparency. These methods are not widely used outside of informatics and business, but have enormous potential for those working in biological contexts. As science continues to rely on computers more heavily, the importance of effective management systems also rises[6,29]. Databases are frequently used for high volume and/or high velocity applications and are well cited in the literature, especially regarding their usage for clinical patient populations[8,30,31]. Several have already been constructed for specific fields such as the Rat Genome Database curation tools or REDCap for clinical and translational research[32,33]. Thus, the use of databases has been adopted in the clinical domain[8] or large genomic databases[32], but has not become common in other scientific disciplines such as tissue engineering.

The issues of handling increasingly complex data using spreadsheet programs have long been acknowledged within the scientific community[34]. One study reported that around 20% of genomic journal papers with supplemental files had gene names that were erroneously converted to dates[35]. These mistakes increased at an average of 15% per year from 2010 to 2015, far outpacing the annual increase of genomics papers at 4% per year. It is often nearly impossible to identify individual errors within a large volume of data, as by nature spreadsheet programs are unsuited for easy validation of results or formula calculations. Published articles even exist for educating scientists on better spreadsheet practices in an attempt to reduce the frequency of errors[7]. One of the strongest benefits of databases is the reduction of error through automated methods and ability to validate potentially questionable data (Figure 3).

A significant outcome of this methodology is the increased rigor of data analysis. The importance of increasing the reproducibility of data has been highlighted by the NIH as well as by other scientists and institutions[36,37]. By having a spreadsheet of file locations corresponding to every database, it is easy to trace a data point back to the lab notebook of the experiment in question (Figure 3). Individual data points can also be quickly identified and found electronically using the corresponding file locations, which is invaluable at times, even when coupled with automatic error screening during the data-pipeline process. Even as the data set is amended over time, best practice involves keeping all past files in case issues

occur or older versions need to be checked. Working non-destructively and keeping old versions within the data-pipeline creates security through redundancy and allows for better troubleshooting.

There are myriad relational database management systems in combination of coding languages that can be used for the same data-pipeline needs. The most appropriate choices are highly dependent on the data and context being used; some applications excel best at scalability, flexibility, reliability, and other priorities[9]. Although databases are still technically finite in scale, reaching memory limits remains beyond the scope of most scientific labs. For instance, an MS Access database has a memory size limit of 2 GB, which would be a data set on the order of hundreds of thousands to millions of entries depending on the data and number of fields. Most labs will never have experimental needs of this magnitude, but if they did then spreadsheet software would be far beyond their effective limits anyway. In comparison, business-level relational database management systems can handle data sets of larger magnitudes while processing millions of transactions simultaneously[29]. Part of the reason databases are not commonly used in scientific laboratories is that past experiments rarely crest needs of such data magnitudes, so easy-to-use spreadsheet software became widespread instead. A significant investment required to make these methods function, however, is the time needed to plan the data-pipeline and learn SQL for using databases (Figure 3 and Figure 4). Although coding experience greatly hastens the process, most will need to learn SQL from scratch. A wealth of documentation is available online through extensive documentation by developers, as well as free SQL tutorials such as at Codecademy, W3Schools, and SQLBolt[10,11,12]. Some alternatives that require subscriptions do exist, however, such as the program teaching website Lynda[38]; further reading about database basics can be found online. In an academic setting, good lab buy-in and robust systems can outlast their creators and help facilitate many years of projects across multiple students. This can be accomplished through the creation of guidelines and implementation steps during setup. Indeed, there is high value for all researchers in having a well-functioning joint data-pipeline and database system.

Other benefits of this methodology 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 (Figure 3). 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. In the context shown, commonly available and economical software was used 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.

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 even more effective than current widespread spreadsheet usage for data storage. Issues with data transparency and replicability in science will only continue to expand in the future as data sets continue to 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.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

This work is supported by the National Heart, Lung, and Blood Institute at the National Institutes of Health, grant number R01 HL129008. The authors especially thank the *LMNA* gene mutation family members for their participation in the study. We also would like to thank Linda McCarthy for her assistance with cell culture and maintaining the lab spaces, Nasam Chokr for her participation in cell imaging and the nuclei data analysis, and Michael A. Grosberg for his pertinent advice with setting up our initial Microsoft Access database as well as answering other technical questions.

References

1. Cavin RK, Lugli P, Zhirnov VV Science and engineering beyond Moore's law. Proceedings of the IEEE. 100 (Special Centennial Issue), 1720–1749 (2012).
2. Mast FD, Ratushny AV, Aitchison JD Systems cell biology. The Journal of Cell Biology. 206 (6), 695–706 (2014). [PubMed: 25225336]
3. Barone L, Williams J, Micklos D Unmet needs for analyzing biological big data: A survey of 704 NSF principal investigators. PLoS Computational Biology. 13 (10), e1005755 (2017). [PubMed: 29049281]
4. Gandomi A, Haider M Beyond the hype: Big data concepts, methods, and analytics. International Journal of Information Management. 35 (2), 137–144 (2015).
5. Siddiqa A et al. A survey of big data management: Taxonomy and state-of-the-art. Journal of Network and Computer Applications. 71, 151–166 (2016).
6. Anderson C The End of Theory: The Data Deluge Makes the Scientific Method Obsolete. Wired Magazine. (2008).
7. Broman KW, Woo KH Data Organization in Spreadsheets. The American Statistician. 72 (1), 2–10 (2018).
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. 22 (4), 527–533 (2015). [PubMed: 25641319]
9. Bassil Y A comparative study on the performance of the Top DBMS systems. Journal of Computer Science, Research. 1 (1), 20–31 (2012).
10. Learn SQL - Codecademy. https://www.codecademy.com/learn/learn-sql (2018).
11. SQL Tutorial - w3schools.com https://www.w3schools.com/sql/ (2018).
12. Introduction to SQL - SQLBolt. https://sqlbolt.com/ (2018).
13. Pedersen TB, Jensen CS Multidimensional database technology. Computer. 34 (12), 40–46 (2001).
14. Győrödi C, Gyorodi R, Sotoc R A Comparative Study of Relational and Non-Relational Database Models in a Web- Based Application. International Journal of Advanced Computer Science and Applications. 6 (11), 78–83 (2015).
15. Nayak A, Poriya A, Poojary D Type of NOSQL databases and its comparison with relational databases. International Journal of Applied Information Systems. 5 (4), 16–19 (2013).
16. Lei C, Feng D, Wei C, Ai-xin Z, Zhen-hu C The application of multidimensional data analysis in the EIA database of electric industry. Procedia Environmental Sciences. 10, 1210–1215 (2011).
17. Soranno PA et al. Building a multi-scaled geospatial temporal ecology database from disparate data sources: fostering open science and data reuse. GigaScience. 4, 28 (2015). [PubMed: 26140212]
18. Edwards P Questionnaires in clinical trials: guidelines for optimal design and administration. Trials. 11, 2 (2010). [PubMed: 20064225]

19. Richards MA et al. MediaDB: A Database of Microbial Growth Conditions in Defined Media. PLoS ONE. 9 (8), e103548 (2014). [PubMed: 25098325]
20. Core JQ et al. Age of heart disease presentation and dysmorphic nuclei in patients with LMNA mutations. PLoS ONE. 12 (11), e0188256 (2017). [PubMed: 29149195]
21. Drew NK, Johnsen NE, Core JQ, Grosberg A Multiscale Characterization of Engineered Cardiac Tissue Architecture. Journal of Biomechanical Engineering. 138 (11), 111003 (2016).
22. Zaragoza MV 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. 11 (5), e0155421 (2016). [PubMed: 27182706]
23. Zaragoza M, Nguyen C, Widyastuti H, McCarthy L, Grosberg A Dupuytren's and Ledderhose Diseases in a Family with LMNA-Related Cardiomyopathy and a Novel Variant in the ASTE1 Gene. Cells. 6 (4), 40 (2017).
24. Zaragoza MV, Hakim SA, Hoang V, Elliott AM Heart-hand syndrome IV: a second family with LMNA-related cardiomyopathy and brachydactyly. Clinical Genetics. 91 (3), 499–500 (2017). [PubMed: 27723096]
25. Eriksson M et al. Recurrent de novo point mutations in lamin A cause Hutchinson-Gilford progeria syndrome. Nature. 423 (6937), 293–298 (2003). [PubMed: 12714972]
26. Drew NK, Eagleson MA, Baldo DB Jr, Parker KK, Grosberg A Metrics for Assessing Cytoskeletal Orientational Correlations and Consistency. PLoS Computational Biology. 11 (4), e1004190 (2015). [PubMed: 25849553]
27. Hamley IW Introduction to Soft Matter: Synthetic and Biological Self-Assembling Materials. John Wiley, Sons Hoboken, NJ (2013).
28. Grosberg A, Alford PW, McCain ML, Parker KK Ensembles of engineered cardiac tissues for physiological and pharmacological study: Heart on a chip. Lab Chip. 11 (24), 4165–4173 (2011). [PubMed: 22072288]
29. Hey T, Trefethen A The Data Deluge: An e-Science Perspective In Grid Computing: Making the Global Infrastructure a Reality. Edited by Berman F, Fox G, Hey AJG, Ch. 36, John Wiley, Sons Hoboken, NJ (2003).
30. Wardle M, Sadler M How to set up a clinical database. Practical Neurology. 16 (1), 70–74 (2016). [PubMed: 26537840]
31. Kerr WT, Lau EP, Owens GE, Trefler A The future of medical diagnostics: large digitized databases. The Yale Journal of Biology and Medicine. 85 (3), 363 (2012). [PubMed: 23012584]
32. Laulederkind SJ 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, bar002 (2011). [PubMed: 21321022]
33. Harris PA et al. Research electronic data capture (REDCap)--a metadata-driven methodology and workflow process for providing translational research informatics support. Journal of Biomedical Informatics. 42 (2), 377–381 (2009). [PubMed: 18929686]
34. Panko RR What we know about spreadsheet errors. Journal of Organizational and End User Computing (JOEUC). 10 (2), 15–21 (1998).
35. Ziemann M, Eren Y, El-Osta A Gene name errors are widespread in the scientific literature. Genome Biology. 17 (1), 177 (2016). [PubMed: 27552985]
36. NIH. Enhancing Reproducibility through Rigor and Transparency. https://grants.nih.gov/reproducibility/index.htm (2018).
37. Hofseth LJ Getting rigorous with scientific rigor. Carcinogenesis. 39 (1), 21–25 (2017).
38. SQL Training and Tutorials - Lynda.com https://www.lynda.com/SQL-training-tutorials/446-0.html (2018).

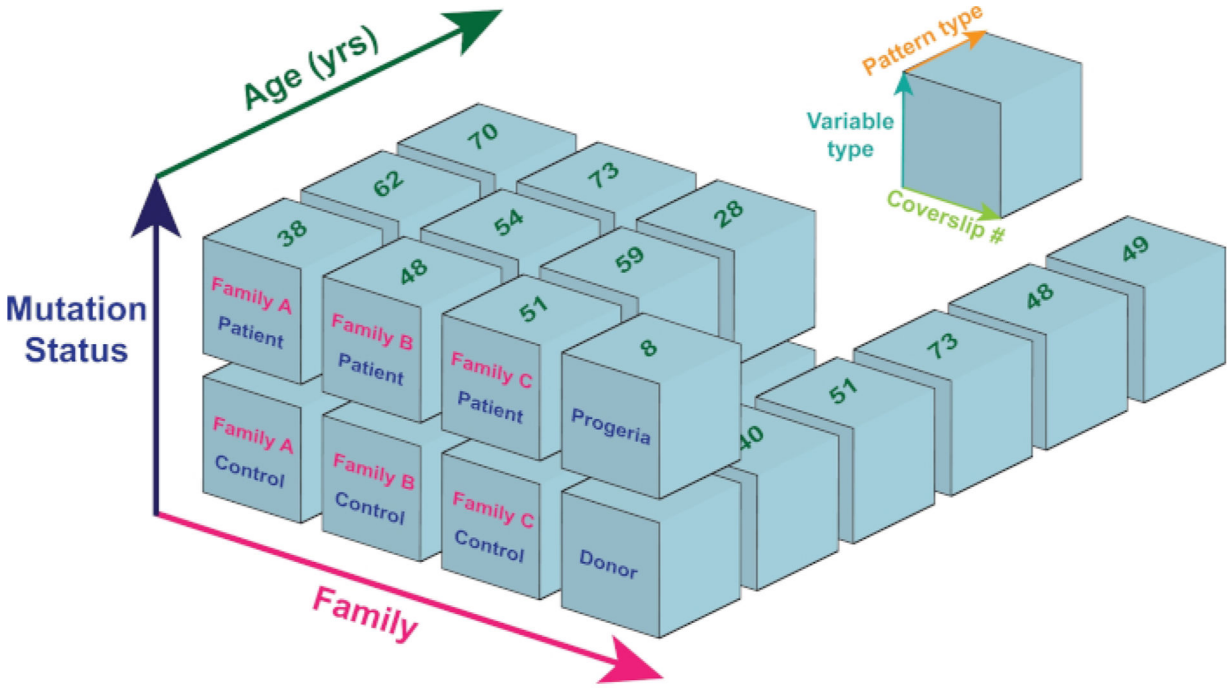

Figure 1: A visualization of multi-dimensional data from the *LMNA* mutation data set.

A single cube is defined by the three dimensions of "Variable type," "Pattern type," and "Coverslip #." Further dimensions are shown as the axes of "Mutation Status," "Age of biopsy" (yrs), and "Family." Colored labels correspond to the different axes shown, such as the age of biopsy (green numbers) for each individual's cube. Here, six of the ten possible dimensions are used to illustrate the multi-dimensionality of experimental data points.

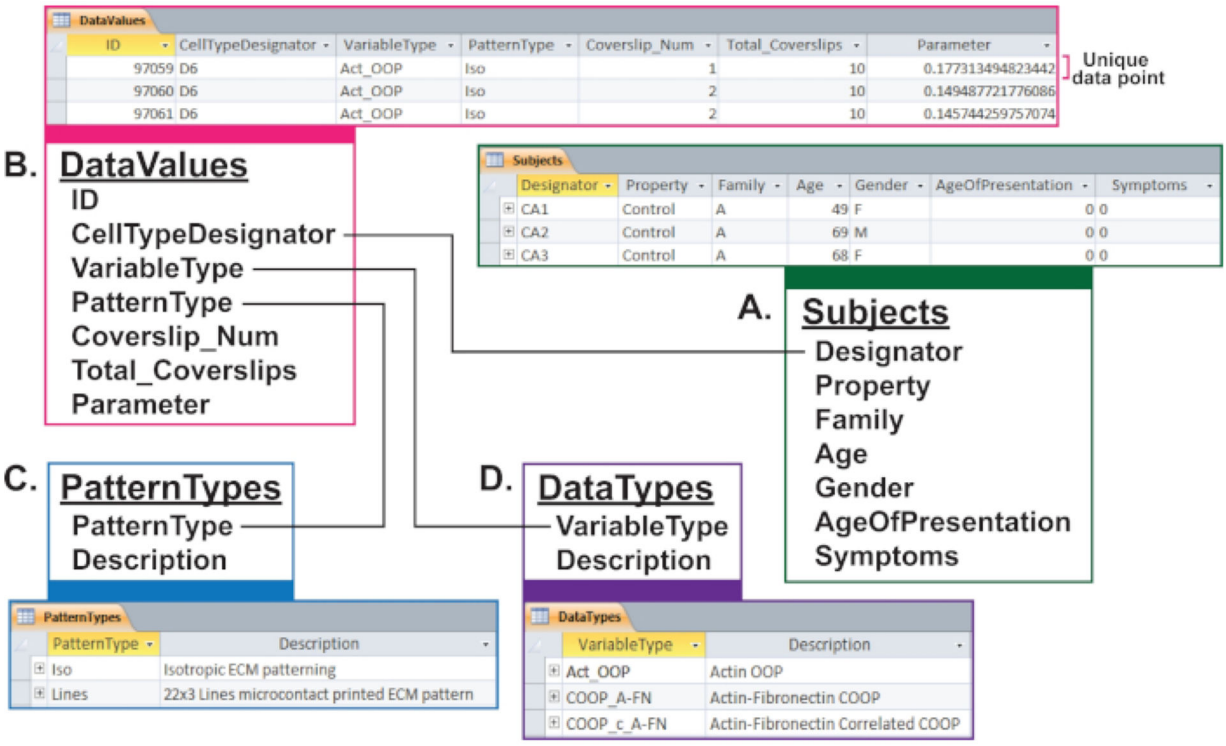

Figure 2: Table and design view relationships within the *LMNA* mutation data set.

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 demonstrates how differing information can be linked.

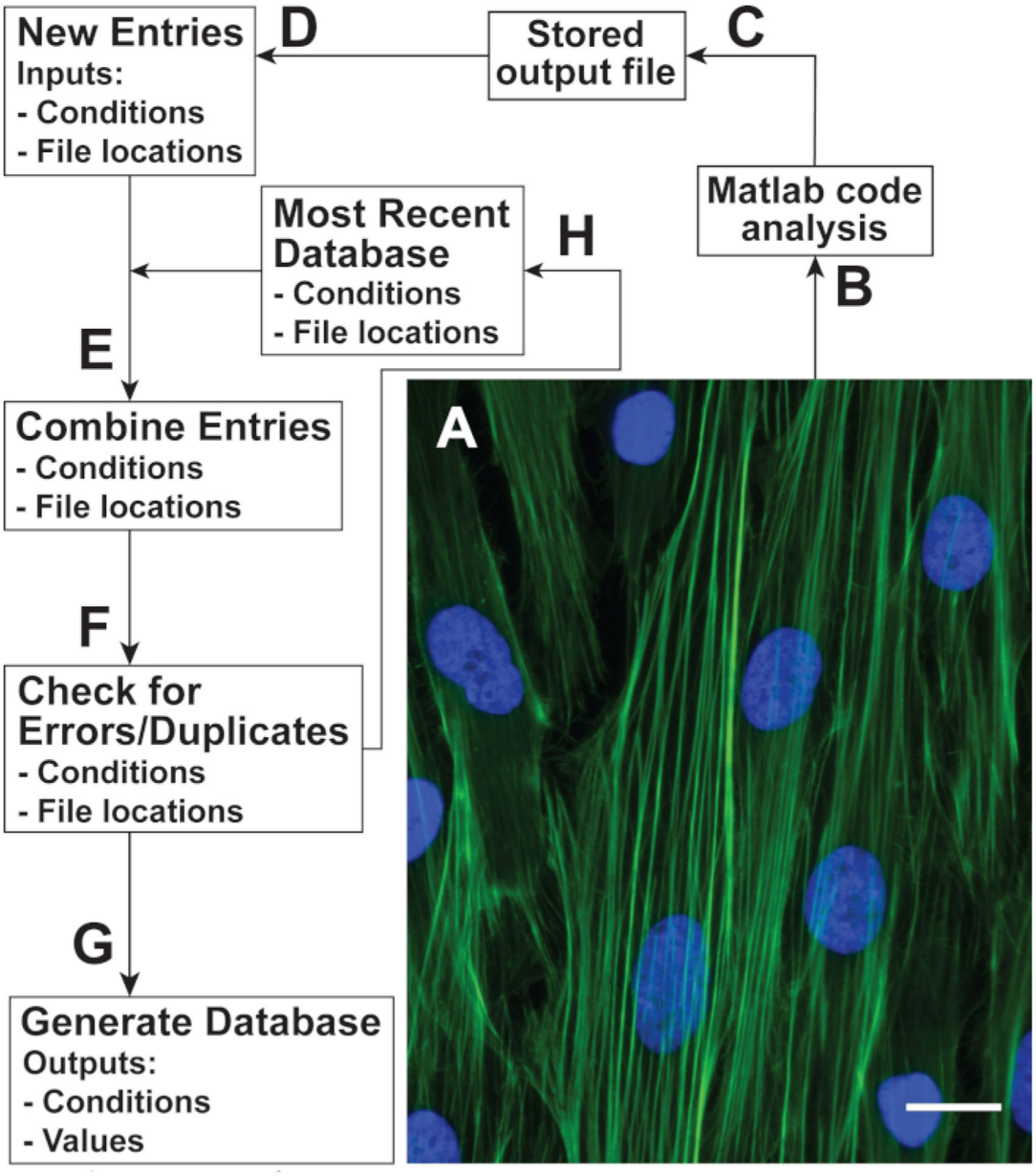

Figure 3: An example of common data-pipeline needs in a generalized context.

New entries were created using user inputs and automated codes, formatting important information into a spreadsheet format. These entries were combined with the most recent set of file location entries, checked for errors, then stored as both a spreadsheet of file locations and a spreadsheet of data values. Scale bar = 20 μm.

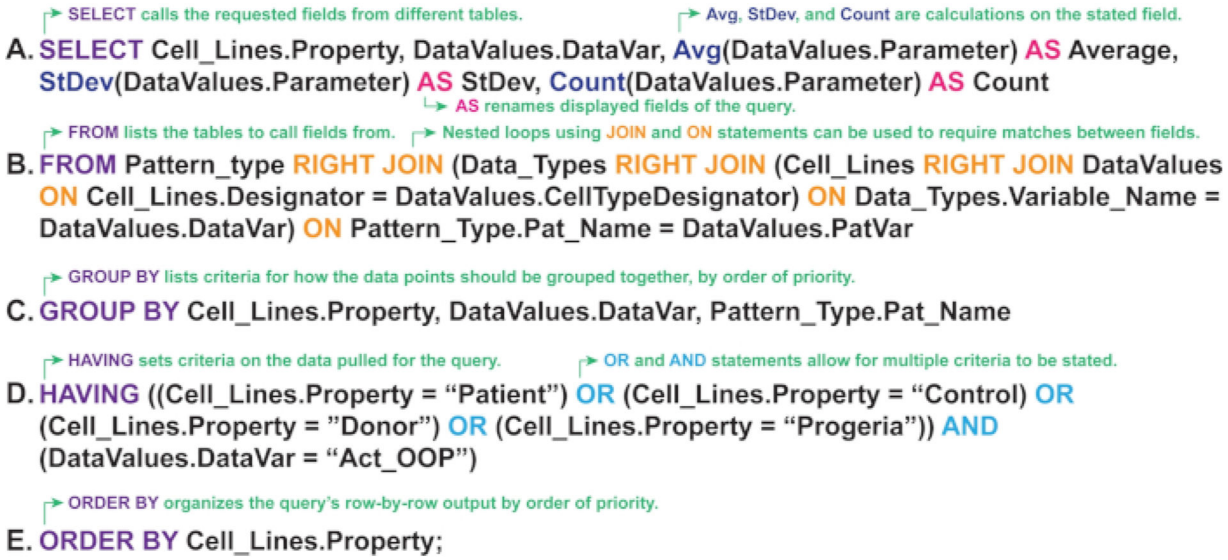

Figure 4: An example query using SQL syntax.

SELECT and FROM statements are requirements to generate a query, but additional commands and criteria are often included. 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 by which the outputs should be arranged by.

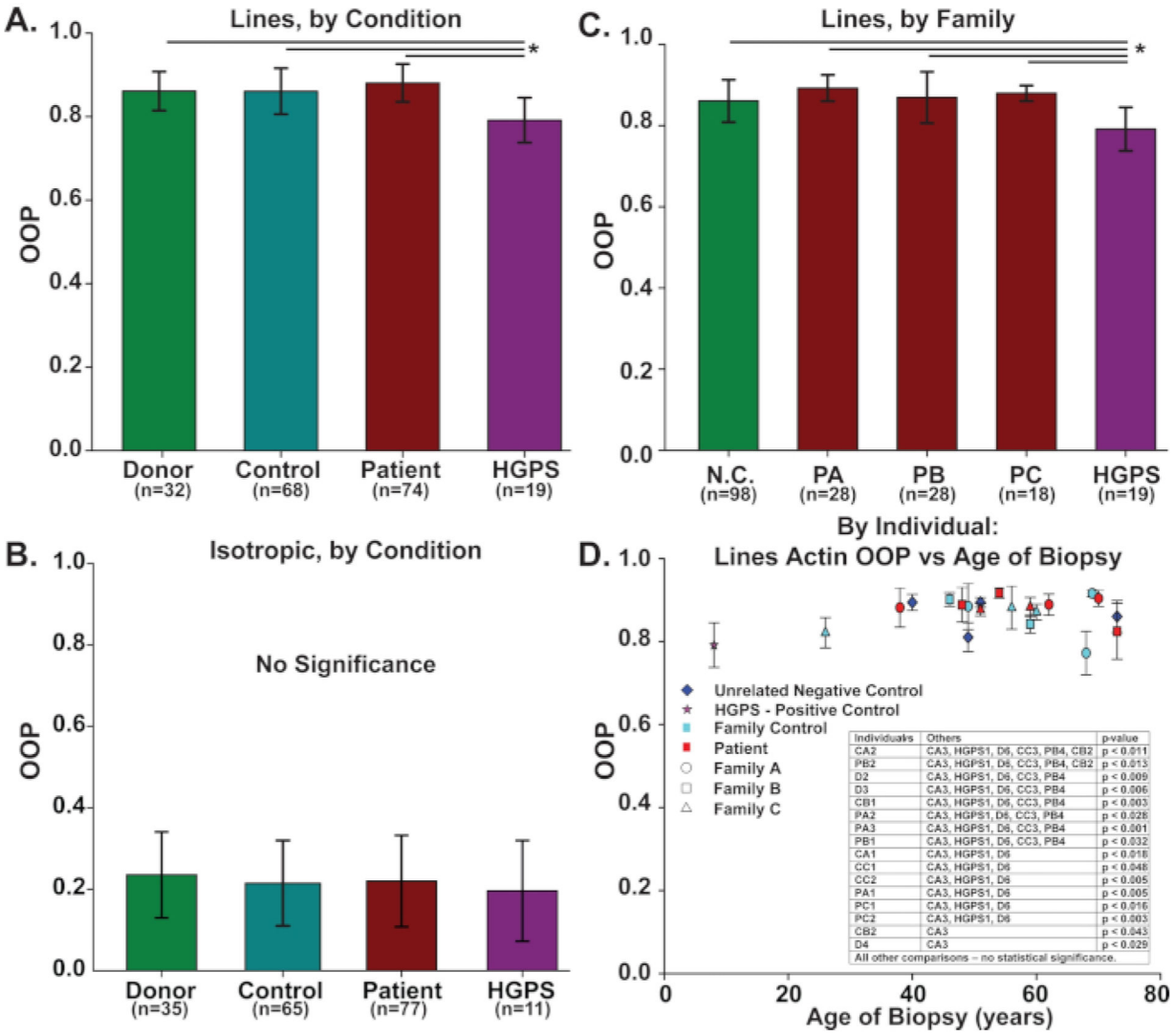

Figure 5: Comparisons between conditions for the actin OOP variable.

(**A**,**B**) groupings correspond to the four primary conditions: non-related negative control Donors, related negative control Controls, *LMNA* mutation Patients from three families, and positive control HGPS. (**C**) all negative controls (N.C.) were combined and patients were separated by family (PA, PB, PC) instead. (**D**) A potential graph of isotropic actin OOP against age at time of biopsy collected for this study, separated by condition and family. Panels A, C, and D are plotted for the tissues micropatterned with a Lines pattern, while panel B is plotted for isotropic tissues. Statistical significance of $p < 0.05$ (*) was found in panels A, C, and D. No significance between any pairs was found in panel B. All error bars represent standard deviations calculated within the database.

Table 1:

List of all the example files that can be uploaded to run the protocol.

Reference Number	File Name	Type
1	LocationPointer.m	Pipe-line Code
2	LocationPointerCompile.m	Pipe-line Code
3	LocationPointer_Remove_Duplicates.m	Pipe-line Code
4	BadPointerCheck.m	Pipe-line Code
5	LocationPointer_Check.m	Pipe-line Code
6	Manual_Pointer_Removal.m	Pipe-line Code
7	Database_Generate.m	Pipe-line Code
8	Cell_Lines.csv	Helper Table
9	Data_Types.csv	Helper Table
10	Pattern_Types.csv	Helper Table
11	DataLocation_Comp_2018_6_26_10_01.csv	Example Data Location File
12	DataValues_2018_6_26_10_02.csv	Example Data Values File
13	Database_Queries.accdb	Example Database

Table 2:

Listed select files that correspond to different variables of either cell nuclei measurements or fibroblast structural (OOP) data.

File Selected	Variable
Summary.mat	Proportion of Defective Nuclei
	All Nuclei Area Average (μm2)
	Defective Nuclei Area Average (μm2)
	Normal Nuclei Area Average (μm2)
	All Nuclei Eccentricity Average
	Defective Nuclei Eccentricity Average
	Normal Nuclei Eccentricity Average
	All Nuclei MNC Average
	Defective Nuclei MNC Average
	Normal Nuclei MNC Average
Act_OOP.mat	Actin OOP
	Actin OOP Director Angle
Fibro_OOP.mat	Fibronectin OOP
	Fibronectin OOP Director Angle
Nuc_OOP.mat	Nuclei OOP
	Nuclei OOP Director Angle