Incorporating Coding into the Classroom: An Important Component of Modern Bioinformatics Instruction

Nichole Orench-Rivera, April Bednarski, Paul Craig & Austin Talbot

To cite this article: Nichole Orench-Rivera, April Bednarski, Paul Craig & Austin Talbot (01 Oct 2024): Incorporating Coding into the Classroom: An Important Component of Modern Bioinformatics Instruction, Journal of College Science Teaching, DOI: 10.1080/0047231X.2024.2405593

To link to this article: https://doi.org/10.1080/0047231X.2024.2405593

Published online: 01 Oct 2024.

JOURNAL OF COLLEGE SCIENCE TEACHING
https://doi.org/10.1080/0047231X.2024.2405593

Incorporating Coding into the Classroom: An Important Component of Modern Bioinformatics Instruction

Nichole Orench-Rivera[a#], April Bednarski[b], Paul Craig[c], and Austin Talbot[d]

[a]School of Health Sciences, Lasell University, Auburndale, Massachusetts, United States; [b]Washington University in St. Louis, St. Louis, Missouri, United States; [c]School of Chemistry & Materials Science, Rochester Institute of Technology, Rochester, New York, United States; [d]Pillar Diagnostics, Natick, Massachusetts, United States

ABSTRACT
Advancements in computation and machine learning have revolutionized science, enabling researchers to address once insurmountable challenges. Bioinformatics, a field that heavily relies on computer-driven analysis of biological data, has greatly benefited from these developments. However, traditional bioinformatics instruction frequently lacks the necessary coding skills. This article explores the transformation of a bioinformatics course in which feedback from students revealed limitations in traditional web application interfaces and the absence of coding automated pipelines for real-world applications. To address these shortcomings, the authors redesigned the project to incorporate computer programming using Google Colaboratory, where students access databases and websites by coding. The curriculum outlined the integration of modern programming skills with essential bioinformatics concepts. This article evaluates the effectiveness of this redesign by analyzing a self-response survey completed by course participants. Results show a positive impact on students' perception of science and scientific research. Bayesian statistical analysis reveals that the programming component significantly predicts students' career clarity in science and their pursuit of graduate education. Integrating coding exercises in bioinformatics education enhances students' preparedness for real-world applications. The freely available GitHub repository will facilitate adoption. By embracing computational tools, students can become adept researchers capable of tackling complex biological questions.

KEYWORDS
Bioinformatics; Python; Laboratory exercise; Genetics; Google colab; Coding; Biological databases

Introduction

Recent advances in computation and machine learning have revolutionized many scientific disciplines. From simulating complex physical phenomena to identifying patterns in large data, computers have enabled scientists to tackle scientific challenges that were previously out of reach. With their speed, accuracy, and ability to handle massive amounts of data, computers have become indispensable tools for scientists in virtually every field of study. This is particularly true in bioinformatics, which uses computers to collect and analyze biological data such as genomic and proteomic sequences. Traditional bioinformatics instruction focuses on learning to navigate the databases directly on the website and interacting with developed visualizations. While these web applications are useful, this instruction is no longer sufficient for academic or industrial careers. To contribute and be a productive member of this field requires researchers to develop new applications, to combine new datasets, to create new analytical techniques, all using computers. In short, anyone seeking to survive and adapt in this new environment must get started in coding (Craig et al., 2022; Nash et al., 2022).

This need was exposed in the summer of 2021, when we implemented the following project for an online (BIOL3XX level) genetics course at Lasell University: 'Studying the Genetic Basis of Disease Using Web-Based Bioinformatics Tools' (Bednarski et al., n.d.; Bednarski et al., 2005). This project (which we will refer to as Project 1.0) led students through their own research projects, investigating the role of specific proteins in genetic diseases. While accessing databases using their web application interface was informative and useful, student feedback indicated that software compatibility was an issue. Furthermore, many of the analyses required manual manipulation of the data rather than coding automated pipelines

CONTACT Nichole Orench-Rivera nichole26or@gmail.com Lasell University, School of Health Sciences, Auburndale, Massachusetts, United States.
[#]Additional affiliation: Supreme Optimization, United States.

required in real-world applications. These issues were similarly encountered in the Summer offering of the course, prompting us to rectify the shortcomings in the subsequent offering of Genetics.

With permission from the authors above, we have redesigned the project to have students access the databases and websites using computer programming in Google Colaboratory through a series of Notebooks with coding exercises. Google Colaboratory is a coding environment that allows users to combine Python code, text and images in a single document (Bisong, 2019). This coding environment is notebook-based, allowing for the program to be implemented and run with the results and text descriptions functioning as an organized report. This resulting project (which we refer to as Project 2.0) yielded a sequence of lectures and associated notebooks that emphasize teaching modern programming skills as applied to important bioinformatics concepts and problems. Table 1 presents the curriculum outline utilized during the Spring 2022 implementation of the course.

In this work, we analyze the efficacy of the redesigned course using a self-response survey completed by the participants of the full-course offering. We also used Bayesian statistical methods to compare student attitudes about science careers in this setting versus a traditional instruction approach. The remainder of this article describes the specific aspects of the course, along with a detailed statistical analysis of the survey. The notebooks prepared for this project redesign, are freely available online as a GitHub repository: (https://github.com/gencode-png/Genetics_Project_Collaborative) and can be offered over the course of a semester or incorporated individually as part of an existing course.

Materials and Methods

Project 2.0 was designed to correspond to a dry-lab format; instructions and notebooks were prepared ahead of time and provided to the student at the beginning of class. Learning was intended to be a hands-on experience, while the instructor was available to answer questions, students were expected to learn by trial and error as they followed the previously-mentioned notebooks which they downloaded from Google Drive. All notebooks were prepared with code and text to incorporate both coding exercises and instructions. For coding exercises, students were required to fill in missing information in the code and/or generate their own code. This structure is similar to other programming-based classes, such as a recently-developed applied mathematics course series (Humpherys & McQuarrie, n.d.), where the authors used text to describe instructions and steps to complete the exercises. Additionally, students could collect information they obtained on each step by answering questions within each Notebook and later submit the notebooks as their work for each laboratory session. Canvas was used for course management and the Google Colab assignments could be directly submitted through Canvas.

Table 1. Curriculum outline utilized during the Spring 2022 implementation of the Genetics Lab course.

Week	Topic/Exercise	Due
1	No lab sessions scheduled the first week of class	
2	Choosing a Project Topic	
	Notebook #1: Colab Programming Tutorial	
	Take home: Notebook #2: Introduction to your Project & Reading Questions #1	
3	NCBI Website Exploration and Practice (or Demonstration)	Notebook #1
	ExPASy Website Exploration and Practice (or Demonstration)	Notebook #2
	Notebook #3: NCBI (National Center for Biotechnology Information) Gene and ExPASy Colab	
	Take Home: Prepare a Poster Template	
4	NCBI BLAST and MSA Websites Exploration and Practice (or Demonstration)	Notebook #3
	Notebook #4: NCBI BLAST and MSA (Multiple Sequence Analysis) Colab	Poster Template
5	Review on amino acids and protein structure	Notebook #4
	Notebook #5: Reading Questions #2	
	PyMOL/Py3Dmol Demonstration and Practice Exercise	
6	Catch up and Revise past work (can schedule Zoom call with instructor)	Notebook #5
7	Notebook #6: 3D visualization and mutagenesis of protein Colab	
8	No Lab Sessions - Spring Break Recess	
9	OMIM Website Exploration and Practice	Notebook #6
	Notebook #7: OMIM (Online Mendelian Inheritance in Man) Colab	
	Take home: Poster Draft #1 with Introduction & Methods	
10	KEGG Website Exploration and Practice	Notebook #7
	Notebook #8: KEGG (Kyoto Encyclopedia of Genes and Genomes) Colab	Poster Draft #1
11	Work on Poster: Results, Figures & Discussion (Poster Draft #2)	Notebook #8
12	Clarifying questions	Poster Draft #2
	Work on Poster: References & Acknowledgements	
	Practice Poster Presentations	
13-14	Poster Presentations	Final Poster
	Final Assessment	

The first lesson (see Table 1) included a notebook tutorial (Blondel, n.d.) for students to get familiar with the Google Colaboratory environment. Time was also taken during the first lab session for students to save all notebooks and required files to their Google Drive. Significant time was dedicated during the first lab session to making sure all students had access to the materials.

Google Colab for Reading Questions and Assignment Submission

The original project included assigned readings and worksheets to contextualize each topic. In Project 2.0 these were converted to Google Colaboratory and students submitted these at specified times. At the end of each lab session, students were asked to submit their Notebooks via link sharing. This allowed for easy access for grading and troubleshooting purposes.

Coding Exercises Included in the Notebooks

Translation of cDNA and Introduction to Python

Students, individually or in pairs, are assigned a 'patient' cDNA sequence coding for a mutated protein that will be studied over the course of the semester. In the first step the mutated amino acid sequence is translated and saved for future use. In the original project the translation took place using bioinformatics.org's 'Translate' suite (Stothard, 2000). However, in our course the students used a modified version of a function by A. Ranjan (Amartya Ranjan Saikia, 2017) from within Google Colab. The students were provided a partially-completed function to convert sequences into amino acids in class. They were then required to complete the function to translate and save the sequence. This also served as an introduction to Python syntax along with standard programming concepts such as variables, functional programming, flow control structures, debugging, and coding environments.

Using NCBI Gene and ExPAsy/Uniprot to Obtain Background Information on Gene of Interest and Protein

In the original project students used the NCBI gene (Brown et al., 2015) and ExPAsy/Uniprot databases (Duvaud et al., 2021) to obtain information about the gene of interest and the protein it encodes. In Project 2.0, students performed the same searches within the Google Colab environment by running code derived from code by Rob Harbert (Rob Harbert, 2018) (Figure 1).

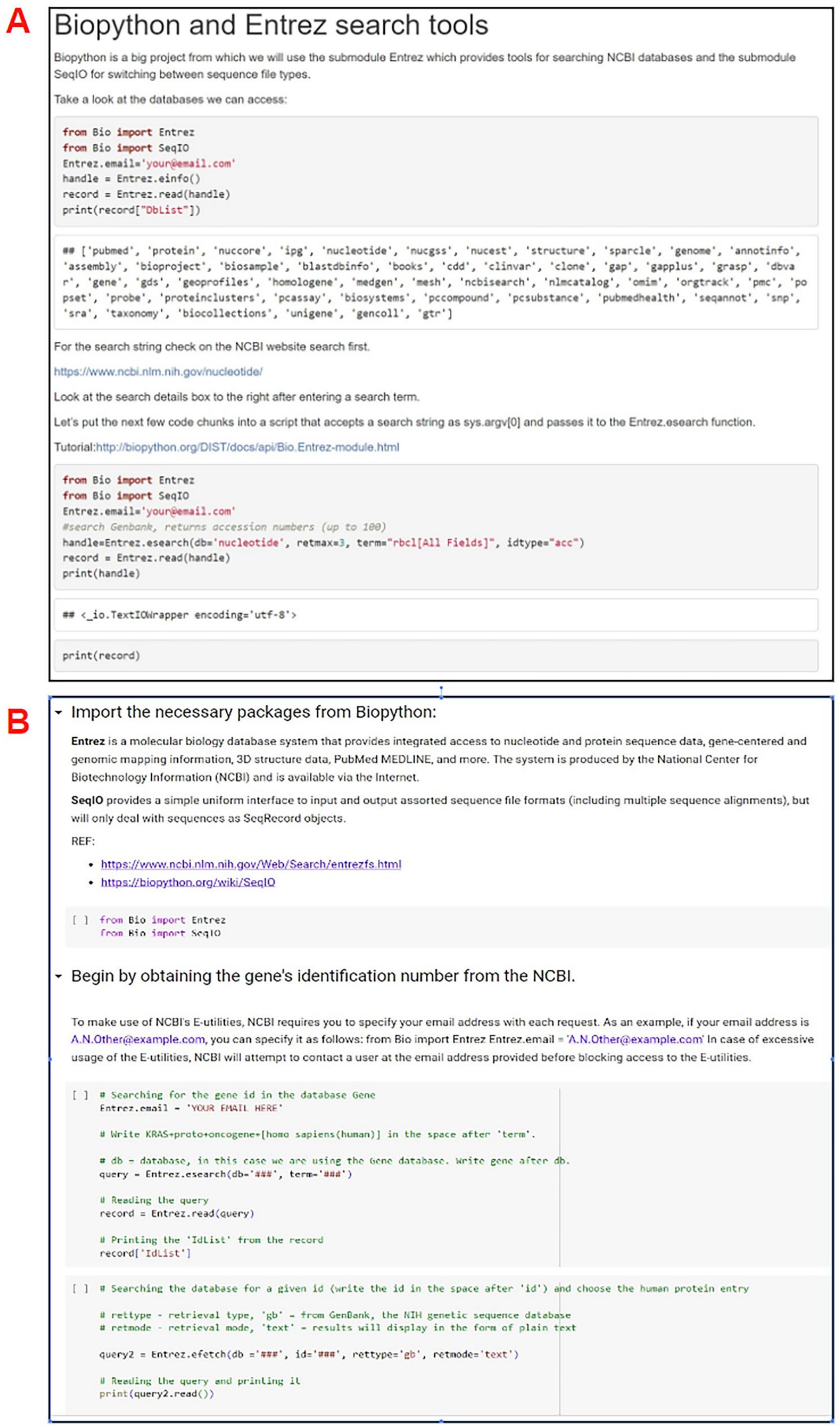
For ExPAsy/Uniprot searches, code was modified from various sources (Cock et al., 2009; Guex & Peitsch, 1997; The UniProt Consortium, 2015). The code in Project 2.0 provided students with opportunities to obtain accession numbers for their proteins, descriptions, sequences (unmutated), comments and features. Aside from obtaining the desired information, the purpose of these exercises was for students to practice basic skills in bioinformatics and machine learning, such as web scraping (using computer programming to extract data from a website), data frames, IO, and data visualization. It also enabled them to learn more about the contents of each database.

NCBI BLAST and MSA Exercise

In the original project, students performed a BLAST search to compare their mutated protein sequence against the unmutated/original sequence of the protein (obtained from NCBI) and against homologous proteins from other organisms. Performing BLAST and later Multiple Sequence Alignment (MSA) allowed students to find the mutated amino acid within the protein sequence of their 'patient'. In Project 2.0 we sourced code from biopython.org to perform the BLAST search within the Google Colab environment and implemented the MSA viewer by D. Farrell (Farrell, n.d.). In this notebook exercise, students had the opportunity to practice retrieving FASTA files from NCBI using computer programming and to examine MSA results using an MSA viewer.

3D Visualization of Protein and Mutagenesis

The coding version of this exercise was still under development during this project and thus was offered in two different ways during two lab periods. In the first section of 3D visualization lab periods, students were instructed and guided through the installation of the PyMOL software (Schrödinger, LLC, 2015) and performed various instructor-led exercises that allowed students to explore the 3D structure of their mutated protein and extract various images for their final project. In the second session, students had the opportunity to practice examining the 3D structure of their protein within the Google environment. While a limited exercise, the purpose of the Google Colab notebook exercise was for students to explore how a visualization exercise could be run using computer programming. It provides a comparison of execution of software commands using the guided user interface to the coding environment. For the coding exercises,



Biopython and Entrez search tools

Biopython is a big project from which we will use the submodule Entrez which provides tools for searching NCBI databases and the submodule SeqIO for switching between sequence file types.

Take a look at the databases we can access:

```
from Bio import Entrez
from Bio import SeqIO
Entrez.email='your@email.com'
handle = Entrez.einfo()
record = Entrez.read(handle)
print(record["DbList"])
```

```
## ['pubmed', 'protein', 'nuccore', 'ipg', 'nucleotide', 'nucgss', 'nucest', 'structure', 'sparcle', 'genome', 'annotinfo',
'assembly', 'bioproject', 'biosample', 'blastdbinfo', 'books', 'cdd', 'clinvar', 'clone', 'gap', 'gapplus', 'grasp', 'dbva
r', 'gene', 'gds', 'geoprofiles', 'homologene', 'medgen', 'mesh', 'ncbisearch', 'nlmcatalog', 'omim', 'orgtrack', 'pmc', 'po
pset', 'probe', 'proteinclusters', 'pcassay', 'biosystems', 'pccompound', 'pcsubstance', 'pubmedhealth', 'seqannot', 'snp',
'sra', 'taxonomy', 'biocollections', 'unigene', 'gencoll', 'gtr']
```

For the search string check on the NCBI website search first.

https://www.ncbi.nlm.nih.gov/nucleotide/

Look at the search details box to the right after entering a search term.

Let's put the next few code chunks into a script that accepts a search string as sys.argv[0] and passes it to the Entrez.esearch function.

Tutorial:http://biopython.org/DIST/docs/api/Bio.Entrez-module.html

```
from Bio import Entrez
from Bio import SeqIO
Entrez.email='your@email.com'
#search Genbank, returns accession numbers (up to 100)
handle=Entrez.esearch(db='nucleotide', retmax=3, term="rbcl[All Fields]", idtype="acc")
record = Entrez.read(handle)
print(handle)
```

```
## <_io.TextIOWrapper encoding='utf-8'>
```

```
print(record)
```

Import the necessary packages from Biopython:

Entrez is a molecular biology database system that provides integrated access to nucleotide and protein sequence data, gene-centered and genomic mapping information, 3D structure data, PubMed MEDLINE, and more. The system is produced by the National Center for Biotechnology Information (NCBI) and is available via the Internet.

SeqIO provides a simple uniform interface to input and output assorted sequence file formats (including multiple sequence alignments), but will only deal with sequences as SeqRecord objects.

REF:

- https://www.ncbi.nlm.nih.gov/Web/Search/entrezfs.html
- https://biopython.org/wiki/SeqIO

```
[ ]  from Bio import Entrez
     from Bio import SeqIO
```

Begin by obtaining the gene's identification number from the NCBI.

To make use of NCBI's E-utilities, NCBI requires you to specify your email address with each request. As an example, if your email address is A.N.Other@example.com, you can specify it as follows: from Bio import Entrez Entrez.email = 'A.N.Other@example.com' In case of excessive usage of the E-utilities, NCBI will attempt to contact a user at the email address provided before blocking access to the E-utilities.

```
[ ]  # Searching for the gene id in the database Gene
     Entrez.email = 'YOUR EMAIL HERE'

     # Write KRAS+proto+oncogene+[homo sapiens(human)] in the space after 'term'.

     # db = database, in this case we are using the Gene database. Write gene after db.
     query = Entrez.esearch(db='###', term='###')

     # Reading the query
     record = Entrez.read(query)

     # Printing the 'IdList' from the record
     record['IdList']
```

```
[ ]  # Searching the database for a given id (write the id in the space after 'id') and choose the human protein entry

     # rettype - retrieval type, 'gb' - from GenBank, the NIH genetic sequence database
     # retmode - retrieval mode, 'text' - results will display in the form of plain text

     query2 = Entrez.efetch(db ='###', id='###', rettype='gb', retmode='text')

     # Reading the query and printing it
     print(query2.read())
```

Figure 1. A. Code from Rob Harbert [10] in (A), incorporated into a Google Colab as a coding exercise. (B).

we used Py3Dmol (Rego & Koes, 2015), The ProDy Project (Bakan et al., 2011, 2014; Zhang et al., 2021) and PyMOL API code modified from Engelberger et al. (Engelberger et al., 2021).

OMIM Database Search to Obtain Background on Diseases Related to the Protein

As with many of the databases mentioned in this article, in the original project, students accessed OMIM

(omim.org) (Hamosh et al., 2005) and extracted information about the disease related to their protein of study. In Project 2.0, students used an R package by D. Tang (David Tang, 2015) implemented in the Google Colab environment. This notebook, as with others, can be used year after year, however, yearly, a key for API access to the OMIM database must be requested at omim.org/api. The purpose of this exercise was for students to use coding to extract information about the disease related to their protein from the OMIM database and apply skills learned in previous notebooks to explore abstracts of articles found in this database. In addition, students got to practice coding in R, another programming language.

KEGG Database Search to Obtain Information about Molecular Pathways

In the original project, the KEGG database (genome.jp/kegg/) (Kanehisa & Goto, 2000) was used mainly to extract information and diagrams about the molecular pathways or processes that involve the protein of interest. In Project 2.0 the purpose remained the same - to become familiar with molecular pathways, to investigate the role of their protein of interest within that particular pathway, and to hypothesize about the ways in which a mutation within the protein might affect its molecular pathway. However, implementing this exercise in Google Colab is fairly limited since the diagrams that are obtained using code are not interactive like they are within the KEGG website. Although there may not be an immediate benefit to exclusively retrieving information from the KEGG database for a single pathway, students who undertake this exercise can acquire knowledge on how to navigate the database for exploring larger datasets. The code used to extract information from KEGG using computer programming was adapted from the Bio.KEGG package from BioPython (Cock et al., 2009). The ReportLab Toolkit was employed for viewing the diagrams in pdf form (Robinson & Becker, n.d.).

Curriculum

Table 1 describes the implementation of Project 2.0 over a full semester. Each lab session (a 3-hour period) required completing one, or sometimes two, Colab Notebooks and students worked on these in-person, with instructor supervision to further clarify instructions and help with coding errors.

Prior to each notebook exercise students were briefly introduced to the websites they would be parsing using coding. We demonstrated the use of the website with a simple query. Students usually went directly to the website when coding errors arose (Figure 2), which also exposed them to the traditional use of the databases.

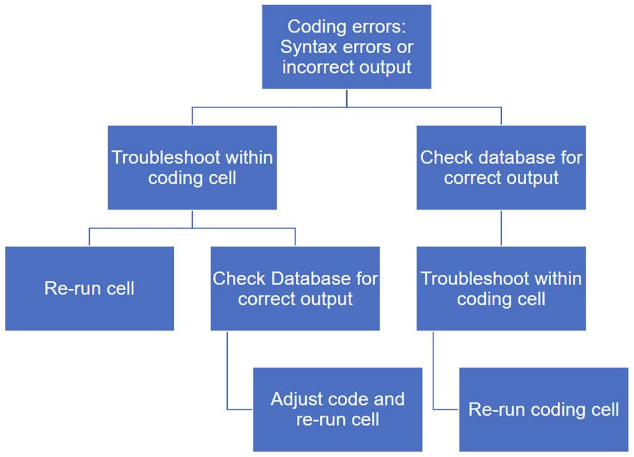

Coding errors: Syntax errors or incorrect output
Troubleshoot within coding cell
Check database for correct output
Re-run cell
Check Database for correct output
Troubleshoot within coding cell
Adjust code and re-run cell
Re-run coding cell

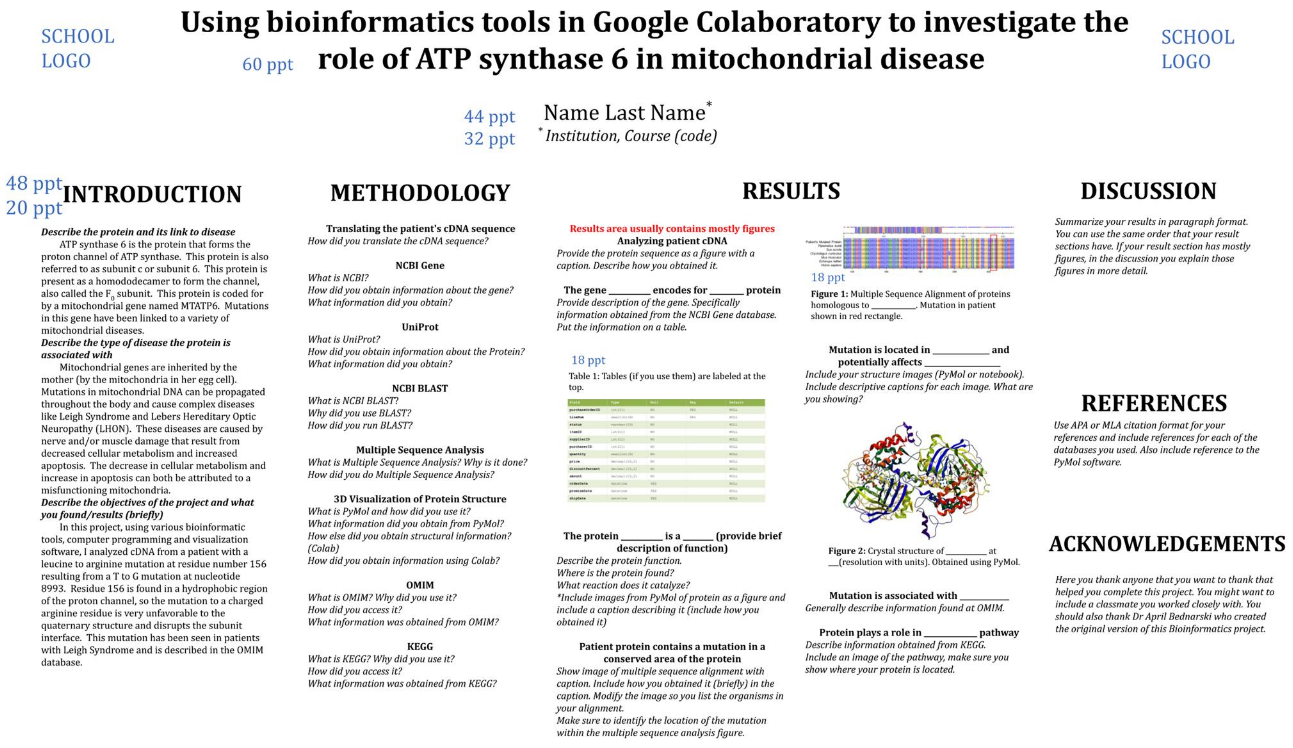
Figure 2. Typical workflow for working through coding errors in the classroom.

Some laboratory sessions were dedicated for students to collect their data and prepare a poster with their findings. We provided a short presentation on how to present posters modified from Scientifica (n.d.), and provided a poster template (available under Supplementary Resources) which helped guide students through the preparation of their own poster (Figure 3). Students often referred to the template during the poster preparation process. Students prepared two drafts of the poster and submitted both for feedback before submitting a final version for grading. Students presented to their peers informally in the lab and a total of 6 out of 15 students chose to present their posters at the end-of-the-year University-wide symposium (Lasell University, n.d.). Students who did not present at the symposium, were given the opportunity to gain extra credit by visiting their peer's poster presentations and filling out a reflection questionnaire on their poster-visiting experience. The project was well-received within the Lasell community with positive feedback from Science faculty.

Results

Statistical Analysis of Student Assessment

During the final lab session students were asked to answer a set of questions based on Grinnell College's Classroom Undergraduate Research Experience (CURE) survey as a form of assessment (Lopatto, n.d.). 15 students out of 20 in both class sections filled the questionnaire (See Supplementary Resources for questions



SCHOOL LOGO

60 ppt **Using bioinformatics tools in Google Colaboratory to investigate the role of ATP synthase 6 in mitochondrial disease**

SCHOOL LOGO

44 ppt Name Last Name*

32 ppt *Institution, Course (code)*

48 ppt **INTRODUCTION**

20 ppt

Describe the protein and its link to disease

ATP synthase 6 is the protein that forms the proton channel of ATP synthase. This protein is also referred to as subunit c or subunit 6. This protein is present as a homododecamer to form the channel, also called the F_0 subunit. This protein is coded for by a mitochondrial gene named MTATP6. Mutations in this gene have been linked to a variety of mitochondrial diseases.

Describe the type of disease the protein is associated with

Mitochondrial genes are inherited by the mother (by the mitochondria in her egg cell). Mutations in mitochondrial DNA can be propagated throughout the body and cause complex diseases like Leigh Syndrome and Lebers Hereditary Optic Neuropathy (LHON). These diseases are caused by nerve and/or muscle damage that result from decreased cellular metabolism and increased apoptosis. The decrease in cellular metabolism and increase in apoptosis can both be attributed to a misfunctioning mitochondria.

Describe the objectives of the project and what you found/results (briefly)

In this project, using various bioinformatic tools, computer programming and visualization software, I analyzed cDNA from a patient with a leucine to arginine mutation at residue number 156 resulting from a T to G mutation at nucleotide 8993. Residue 156 is found in a hydrophobic region of the proton channel, so the mutation to a charged arginine residue is very unfavorable to the quaternary structure and disrupts the subunit interface. This mutation has been seen in patients with Leigh Syndrome and is described in the OMIM database.

METHODOLOGY

Translating the patient's cDNA sequence
How did you translate the cDNA sequence?

NCBI Gene
What is NCBI?
How did you obtain information about the gene?
What information did you obtain?

UniProt
What is UniProt?
How did you obtain information about the Protein?
What information did you obtain?

NCBI BLAST
What is NCBI BLAST?
Why did you use BLAST?
How did you run BLAST?

Multiple Sequence Analysis
What is Multiple Sequence Analysis? Why is it done?
How did you do Multiple Sequence Analysis?

3D Visualization of Protein Structure
What is PyMol and how did you use it?
What information did you obtain from PyMol?
How else did you obtain structural information? (Colab)
How did you obtain information using Colab?

OMIM
What is OMIM? Why did you use it?
How did you access it?
What information was obtained from OMIM?

KEGG
What is KEGG? Why did you use it?
How did you access it?
What information was obtained from KEGG?

RESULTS

Results area usually contains mostly figures

Analyzing patient cDNA
Provide the protein sequence as a figure with a caption. Describe how you obtained it.

The gene ________ encodes for _______ protein
Provide description of the gene. Specifically information obtained from the NCBI Gene database. Put the information on a table.

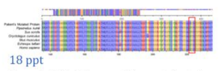
18 ppt

Table 1: Tables (if you use them) are labeled at the top.

Field	Type	Null	Key	Default
purchaseOrderID	int(11)	NO	PRI	NULL
lineNum	smallint(6)	NO	PRI	NULL
status	varchar(25)	NO		NULL
itemID	int(11)	NO		NULL
supplierID	int(11)	NO		NULL
purchaserID	int(11)	NO		NULL
quantity	smallint(6)	NO		NULL
price	decimal(10,2)	NO		NULL
discountPercent	decimal(10,2)	NO		NULL
amount	decimal(10,2)	NO		NULL
orderDate	datetime	YES		NULL
promiseDate	datetime	YES		NULL
shipDate	datetime	YES		NULL

The protein ________ is a ______ (provide brief description of function)
Describe the protein function.
Where is the protein found?
What reaction does it catalyze?
**Include images from PyMol of protein as a figure and include a caption describing it (include how you obtained it)*

Patient protein contains a mutation in a conserved area of the protein
Show image of multiple sequence alignment with caption. Include how you obtained it (briefly) in the caption. Modify the image so you list the organisms in your alignment.
Make sure to identify the location of the mutation within the multiple sequence analysis figure.



18 ppt

Figure 1: Multiple Sequence Alignment of proteins homologous to _________. Mutation in patient shown in red rectangle.

Mutation is located in __________ and potentially affects _____________
Include your structure images (PyMol or notebook). Include descriptive captions for each image. What are you showing?

Figure 2: Crystal structure of ________ at __(resolution with units). Obtained using PyMol.

Mutation is associated with _________
Generally describe information found at OMIM.

Protein plays a role in _________ pathway
Describe information obtained from KEGG. Include an image of the pathway, make sure you show where your protein is located.

DISCUSSION

Summarize your results in paragraph format. You can use the same order that your result sections have. If your result section has mostly figures, in the discussion you explain those figures in more detail.

REFERENCES

Use APA or MLA citation format for your references and include references for each of the databases you used. Also include reference to the PyMol software.

ACKNOWLEDGEMENTS

Here you thank anyone that you want to thank that helped you complete this project. You might want to include a classmate you worked closely with. You should also thank Dr April Bednarski who created the original version of this Bioinformatics project.

Figure 3. Poster template provided to students.

included in the survey and answers provided by the students). Among the 15 students, 4 were in their second year, 8 were in their third year, and 3 were in their fourth year. The students' majors included Forensic Science (6), Forensic Science/Biology (1), Applied Forensic Science (2), Health Science (4), Health Science with a Psychology Minor (1), and Biology (5), with one student on the Pre-Med track. It is important to note that at the beginning of the semester, the instructor asked the students if any of them had prior programming experience, and only one student mentioned having done some programming in high school. Broadly speaking, we can infer that most students had limited or no coding experience before engaging in this lab. From the survey, we observed an almost universal experience gained in working individually (p-value = 0.007) and universal agreement that they can do well in science courses (p-value = 0.001). Most students also agreed that experiments confirm information studied in class (p-value = 0.007) and only one student viewed negative results as a failure (with several abstentions in this question). The majority of students surveyed found the course to be a good way to learn about the subject matter (p-value = 0.001) and all indicated they will still be able to use the thinking skills they learn in science even if they forget the facts. Unfortunately, this course did not eliminate the misconception that all theories are equally valid with only 7 out of 15 students agreeing and 3 in disagreement.

Due to the limited number of respondents, we were unable to test a large number of statistical questions due to concerns associated with multiple hypothesis testing (Sedgwick, 2014). As a result, we limited ourselves to testing a single question; what teaching methods, if any, were predictive of influencing career plans? To test this hypothesis, we predicted the student response to career clarification using their evaluations of a variety of instruction methods, namely group work, reading literature, reading the textbook, programming experience, peer critique, and lecture.

We used logistic regression as a statistical model (Christensen, 2011) combined with Bayesian inference (Gelman et al., 2015) to reduce the noise in coefficient estimates by incorporating prior information on the coefficients in the form of a prior distribution. We placed a t-distribution prior on the intercept term and Gaussian priors on the coefficients centered at 0 with variance 10. This corresponds to a weakly-informative prior belief that the instruction methods have no effect on career clarification, but allows for these beliefs to be altered via moderate data observations. The model was fit using the BRMS package in R (Bürkner, 2018), which generates posterior samples using Hamiltonian Monte Carlo, with further details provided in the supplement (Bürkner, 2018). In our methods, we used N(0,10) priors for all the coefficients. We ran four chains of Markov Chain Monte

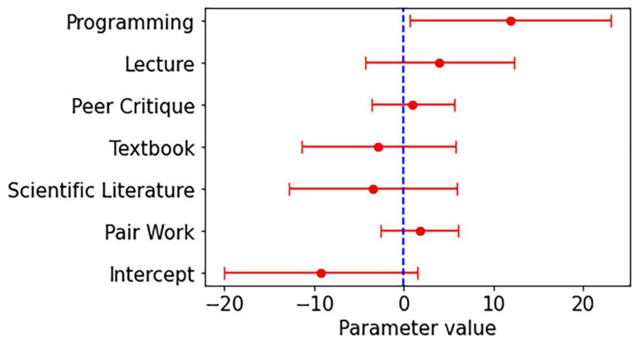


Figure 4. The posterior means and 95% credible intervals of the different instruction methods. The posterior mass for programming experience is almost exclusively positive, indicating a high probability of positive correlation and by extension statistical significance. On the other hand, the probability mass of textbook readings is on both sides of 0, so we have little evidence of any effect.

Carlo (MCMC) sampling, generating 30,000 samples for each chain, with a burn-in phase of 2,000 initial samples and 4 chains. This approach ensures reliable convergence and thorough exploration of the parameter space. We used the Gelman-Rubin statistic (Rhat) values to evaluate the quality of mixing. Heuristically, this measures the extent to which the chain is insensitive to the starting point, suggesting stationarity. Values close to 1 indicate a good fit while values substantially larger than 1 indicate poor sampling. All Rhat values were below 1.001, showing solid convergence and consistency across chains, a crucial aspect for reliable inference in Bayesian analysis.

We then evaluated the significance of the instruction methods using the 95% credible interval of the posterior (the Bayesian version of a confidence interval, but with the more intuitive interpretation that with 95% probability the parameter estimate lies within the credible interval as opposed to a frequentist interpretation of 95% of confidence intervals containing the true value). Significance was then evaluated by determining whether the credible interval contained 0 (no effect). We found that the only instruction method that had a significant impact on clarification of career plans was programming experience (estimate $= 11.9, CI = [1.9, 24.3]$ (Figure 4). While the credible interval is large, further samples would allow for more precise quantification of the effect. The remaining instruction methods lacked any significance. In conclusion, the project positively influenced the perception of science and scientific research suggesting that programming experience is absolutely critical in clarifying student career paths in science and continuing with graduate education, in comparison to traditional instruction methods such as group-work, lecture, and readings.

Finally, we performed a rough sensitivity analysis to evaluate the influence of our priors on the model by evaluating the results from using N(0,1) priors (highly informative) and N(0,100) priors (nearly flat, no prior information). We found that the highly informative prior resulted in no significant coefficients, with all credible intervals being approximately $CI = (-1,2)$. This suggests that the prior dominates the resulting posterior, which is due to the limited sample size. Given that the prior is centered at 0, representing skepticism in parameter significance, it is unsurprising that the resulting posterior reflects this skepticism. On the other end of the spectrum, we found that N(0,100) priors resulted in the terms for reading $CI = (-152.5, -3.3)$ and lecture $CI = (1.1, 119.1)$ became significant in addition to the programming component $CI = (23.3, 197.6)$. However, these excessively large credible intervals indicate a lack of certainty in the true parameter value stemming from the minimal amount of prior information and near collinearity of the covariates. Based on this analysis, we found that N(0,10) priors provided a reasonable amount of skepticism and regularization while allowing for the observations to influence the posterior.

Conclusions

Overall, the project resulted in a valuable learning experience for both the students and the instructor. Students were largely satisfied with the course and instruction; by the end students had a good grasp of the purpose of bioinformatics, fundamental concepts of programming and data analysis, and practical programming experience. Furthermore, we found that the coding aspect of the course was highly predictive of career clarification, with half of the class expressing interest in furthering their scientific education. Even the students who were not interested in further learning found the course improved critical thinking and problem-solving skills.

There are multiple straightforward extensions for improving Project 2.0. First, we suggest ensuring that students have access to all files and are working in shareable Notebooks within their Google Drive at the beginning of class. We found a substantial minority of students did not make their working folder within Google Drive shareable. This impeded the instructor from accessing the files for grading. Second, students should be instructed to download all files from GitHub to prevent issues with file storage within Google Drive. Third, during programming exercises many students attempted to troubleshoot their own code. This is inefficient and does not reflect the team-based aspect of modern bioinformatics. Pair programming is a common programming strategy (Begel & Nagappan, 2008), with one student observer reviewing each line of code as the driver types. This is reminiscent of a well-known and effective debugging strategy, known as the rubber duck strategy (Hunt & Thomas, 2000). When a programmer encounters a difficult bug, he or she explains the task out loud, line by line, to a rubber duck (or another classmate). By the end of the explanation both the error and solution are often apparent. Furthermore, students should be encouraged to search for solutions to error messages online. It is very effective to troubleshoot many errors by submitting the problem to comprehensive sites such as Stack Exchange (stackexchange.com) or Stack Overflow (stackoverflow.com), which are often suggested by Google Colab. Self-diagnosis is a critical skill in any professional environment. Finally, poster presentations will be incorporated as mandatory final presentations of the student's projects, as oral and poster presentations are an integral component of professional development.

We encourage other instructors to implement this course and contact us with any issues encountered. The Notebooks are all available to use freely at: https://github.com/gencode-png/Genetics_Project_Collaborative. The notebooks can be used individually within existing lessons and adapted to any Genetics or Biochemistry course.

Respective Contributions

Author 1 adapted the original project to Google Colab and wrote the article. Author 2 led a workshop that provided valuable tools for the development of this project, provided advice, and helped write the article. Author 4 performed the statistical analyses and contributed to the writing of the article. Author 3 developed the initial project that this work was based on and provided input on the redesign and development of student exercises.

Acknowledgments

We would like to thank Walter Novak for providing excellent troubleshooting with code and implementations of molecular visualization software. We would also like to thank MolSSI (molssi.org) for providing an excellent workshop on implementing coding in the classroom, which was supported by NSF CHE-2136142.

Supplementary Resources

- Original Project website: https://community.gep.wustl.edu/gepweb/bio3055/bio3055.html
- Access to all notebooks: https://github.com/gencode-png/Genetics_Project_Collaborative
- Access to Poster template: https://docs.google.com/presentation/d/16ZRfabx0_GuKgBMuRbhLoLmOCXv2QkhHftni_5K9bTQ/edit?usp=sharing
- Assessment and Feedback Form Questions and Responses: https://docs.google.com/spreadsheets/d/1Gbv1PPVPh6qGaJzq-9iAMh4TSbhf7_BIlHWPdaLymm4/edit?usp=sharing

Ethical approval

This article contains student data obtained with the approval of the Institutional Review Board (IRB) of Lasell University. All necessary ethical considerations were followed to ensure the privacy, confidentiality, and consent of the students involved in this research. The data presented in this article adheres to the guidelines and regulations set forth by the IRB, and all identifying information has been appropriately anonymized to protect the participants' identities.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

ORCID

Nichole Orench-Rivera http://orcid.org/0000-0003-0509-1856
April Bednarski http://orcid.org/0009-0000-5265-758X
Paul Craig http://orcid.org/0000-0002-2085-7816
Austin Talbot http://orcid.org/0000-0001-6207-1304

Data Availability Statement

All data supporting the findings of this article are available upon request. Researchers interested in accessing the data may contact author 1 at nichole26or@gmail.com to inquire about the availability and obtain the necessary datasets for further analysis and verification.

References

Bakan, A., Dutta, A., Mao, W., Liu, Y., Chennubhotla, C., Lezon, T. R., & Bahar, I. (2014). Evol and ProDy for bridging protein sequence evolution and structural dynamics. *Bioinformatics*, *30*(18), 2681–2683. https://doi.org/10.1093/bioinformatics/btu336

Bakan, A., Meireles, L. M., & Bahar, I. (2011). ProDy: Protein Dynamics Inferred from Theory and Experiments. *Bioinformatics*, *27*(11), 1575–1577. https://doi.org/10.1093/bioinformatics/btr168

Bednarski, A. E., Elgin, S. C. R., & Pakrasi, H. B. (n.d.). *Biology 3055 laboratory-public site*. Retrieved December 3, 2022, from https://community.gep.wustl.edu/gepweb/bio3055/center.html

Bednarski, A. E., Elgin, S. C. R., & Pakrasi, H. B. (2005). An inquiry into protein structure and genetic disease: Introducing undergraduates to bioinformatics in a large introductory course. *Cell Biology Education*, *4*(3), 207–220. https://doi.org/10.1187/cbe.04-07-0044

Begel, A., & Nagappan, N. (2008). *Pair programming: What's in it for me?* [Paper presentation]. *Proceedings. of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 120–128). https://doi.org/10.1145/1414004.1414026

Bisong, E. (2019). Google Colaboratory. In E. Bisong (Ed.), *Building machine learning and deep learning models on google cloud platform: A comprehensive guide for beginners* (pp. 59–64). Apress. https://doi.org/10.1007/978-1-4842-4470-8_7

Blondel, M. (n.d.). *Python basics—Google colaboratory*. Retrieved April 22, 2023, from https://colab.research.google.com/github/data-psl/lectures2020/blob/master/notebooks/01_python_basics.ipynb#scrollTo=zbJ2fw29OHAX

Brown, G. R., Hem, V., Katz, K. S., Ovetsky, M., Wallin, C., Ermolaeva, O., Tolstoy, I., Tatusova, T., Pruitt, K. D., Maglott, D. R., & Murphy, T. D. (2015). Gene: A gene-centered information resource at NCBI. *Nucleic Acids Research, 43*(Database issue), D36–D42. https://doi.org/10.1093/nar/gku1055

Bürkner, P.-C. (2018). Advanced bayesian multilevel modeling with the R package brms. *The R Journal*, *10*(1), 395–411. https://doi.org/10.32614/RJ-2018-017

Christensen, R. (2011). *Plane answers to complex questions: The theory of linear models*. Springer. https://doi.org/10.1007/978-1-4419-9816-3

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., & de Hoon, M. J. L. (2009). Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163

Craig, P. A., Nash, J. A., & Crawford, T. D. (2022). Python scripting for biochemistry and molecular biology in Jupyter Notebooks. *Biochemistry and Molecular Biology Education: a Bimonthly Publication of the International Union of Biochemistry and Molecular Biology*, *50*(5), 479–482. https://doi.org/10.1002/bmb.21676

Duvaud, S., Gabella, C., Lisacek, F., Stockinger, H., Ioannidis, V., & Durinx, C. (2021). Expasy, the Swiss bioinformatics resource portal, as designed by its users. *Nucleic Acids Research, 49*(W1), W1–W7. https://doi.org/10.1093/nar/gkab225

Engelberger, F., Galaz-Davison, P., Bravo, G., Rivera, M., & Ramírez-Sarmiento, C. A. (2021). Developing and implementing cloud-based tutorials that combine bioinformatics software, interactive coding, and visualization exercises for distance learning on structural bioinformatics. *Journal of Chemical Education*, *98*(5), 1801–1807. https://doi.org/10.1021/acs.jchemed.1c00022

Farrell, D. (n.d.). *Bioinformatics and other bits—A sequence alignment viewer with Bokeh and Panel* [Blog]. Retrieved April 22, 2023, from https://dmnfarrell.github.io/bioinformatics/bokeh-sequence-aligner

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2015). *Bayesian data analysis* (3rd ed.). Chapman and Hall/CRC. https://doi.org/10.1201/b16018

Guex, N., & Peitsch, M. C. (1997). SWISS-MODEL and the Swiss-Pdb Viewer: An environment for comparative protein modeling. *Electrophoresis*, *18*(15), 2714–2723. https://doi.org/10.1002/elps.1150181505

Hamosh, A., Scott, A. F., Amberger, J. S., Bocchini, C. A., & McKusick, V. A. (2005). Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Research*, *33*(Database issue), D514–D517. https://doi.org/10.1093/nar/gki033

Harbert, R. (2018). *Scripting NCBI web-scraping*. https://rsh249.github.io/python_workshop/scripting_NCBI_searches.html

Humpherys, J., McQuarrie, S. A. (n.d.). *Foundations of Applied Mathematics* [Repository]. GitHub. Retrieved April 22, 2023, from https://github.com/Foundations-of-Applied-Mathematics

Hunt, A., & Thomas, D. (2000). *The pragmatic programmer: From journeyman to master*. Addison-Wesley Longman Publishing Co., Inc.

Kanehisa, M., & Goto, S. (2000). KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, *28*(1), 27–30. https://doi.org/10.1093/nar/28.1.27

Lasell University. (n.d.). *Bioinformation in the Genetics Classroom: Spring 2022: Coming of Age*. Retrieved April 22, 2023, from (Lasell University Connected Learning Symposium).

Lopatto, D. (n.d.). *CURE Survey*. Grinnell College. Retrieved April 22, 2023, from https://sure.sites.grinnell.edu/cure-survey/

Nash, J. A., Mostafanejad, M., Crawford, T. D., & McDonald, A. R. (2022). MolSSI education: Empowering the next generation of computational molecular scientists. *Computing in Science & Engineering*, *24*(3), 72–76. https://doi.org/10.1109/MCSE.2022.3165607

Rego, N., & Koes, D. (2015). 3Dmol.js: Molecular visualization with WebGL. *Bioinformatics*, *31*(8), 1322–1324. https://doi.org/10.1093/bioinformatics/btu829

Robinson, A., Becker, R, ReportLab team and the community. (n.d). *reportlab: The reportlab toolkit* (3.6.12) [Python]. Retrieved April 22, 2023, from http://www.reportlab.com/

Saikia, A. R. (2017). DNA to protein in python 3. *GeeksforGeeks*, June 27. https://www.geeksforgeeks.org/dna-protein-python-3/

Scientifica. (n.d). *Tips for presenting your scientific poster at a conference*. Tips for Presenting Your Scientific Poster at a Conference. Retrieved April 22, 2023, from https://www.scientifica.uk.com/neurowire/tips-for-presenting-your-scientific-poster-at-a-conference

Schrödinger, LLC. (2015). *The PyMOL Molecular Graphics System, Version 1.8*.

Sedgwick, P. (2014). Pitfalls of statistical hypothesis testing: Multiple testing. *BMJ*, *349*.

Stothard, P. (2000). The sequence manipulation suite: JavaScript programs for analyzing and formatting protein and DNA sequences. *BioTechniques*, *28*(6), 1102, 1104–1104. https://doi.org/10.2144/00286ir01

Tang, D. (2015). *Getting started with the OMIM API in R*. Dave Tang's Blog. https://davetang.org/muse/2015/03/17/getting-started-with-the-omim-api/

The UniProt Consortium. (2015). UniProt: A hub for protein information. *Nucleic Acids Research*, *43*(Database issue), D204–D212. https://doi.org/10.1093/nar/gku989

Zhang, S., Krieger, J. M., Zhang, Y., Kaya, C., Kaynak, B., Mikulska-Ruminska, K., Doruker, P., Li, H., & Bahar, I. (2021). ProDy 2.0: Increased scale and scope after 10 years of protein dynamics modelling with Python. *Bioinformatics*, *37*(20), 3657–3659. https://doi.org/10.1093/bioinformatics/btab187