Nikki Tirrell

Advanced Practical Computer Concepts for Bioinformatics

Final Project Narrative

Introduction

The Computational Modeling Variant Reference (CMVR) is a tool that may be used in computational biology and genomic modeling to keep track of the genomic variants that patients have which are being modeled. In my experience as a computational biologist modeling biological processes in personalized medicine, I found it difficult to keep track of the genomic variants that the patient had. I would come across genomic variants of interest in literature, and search through pages of pdf reports excel sheets to see if this were a variant that the patient also had. If he did have this variant, then I would use this as a guide to model the biological processes that include that gene. It was important to know whether the patient had certain genomic variants, but always a time-consuming effort to search through his reports and determine if he had the variant.

The CMVR shortens this process. Rather than having a multitude of reports to sift through, all patient genomic data is saved in the MySQL schema. The user simply opens a webpage, enters the variant of interest, and my program will search to see if that variant is in the patient's record. This is the basic implementation of the CMVR tool.

I further enhanced the tool by providing additional information to aid in analysis. First, I store basic information on each patient: date of birth, sex, and ethnicity. This information can be helpful when making the probability functions for certain mutations or downstream effects. The user may opt to see this information by clicking the "Patient Data" option. I also added information on the genomic variant itself, which are its frequency and the nucleotide mutation that occurs which are from the NCBI SNP database. This option also displays as the gene ID, name and product which are accessed through the "Variant Data" option which comes automatically checked. Finally, I added information on whether that variant is currently part of a model. This includes information on the overall model as well as the specific process that the variant is involved in, and is displayed when the "Model Data" option is chosen. Say that I search for the variant because I found literature linking it to one process, but this search reveals that it is already part of a different model in a different process. This may reveal new links between these processes, providing alternative networks to explore.

The image below shows the CMVR webpage.

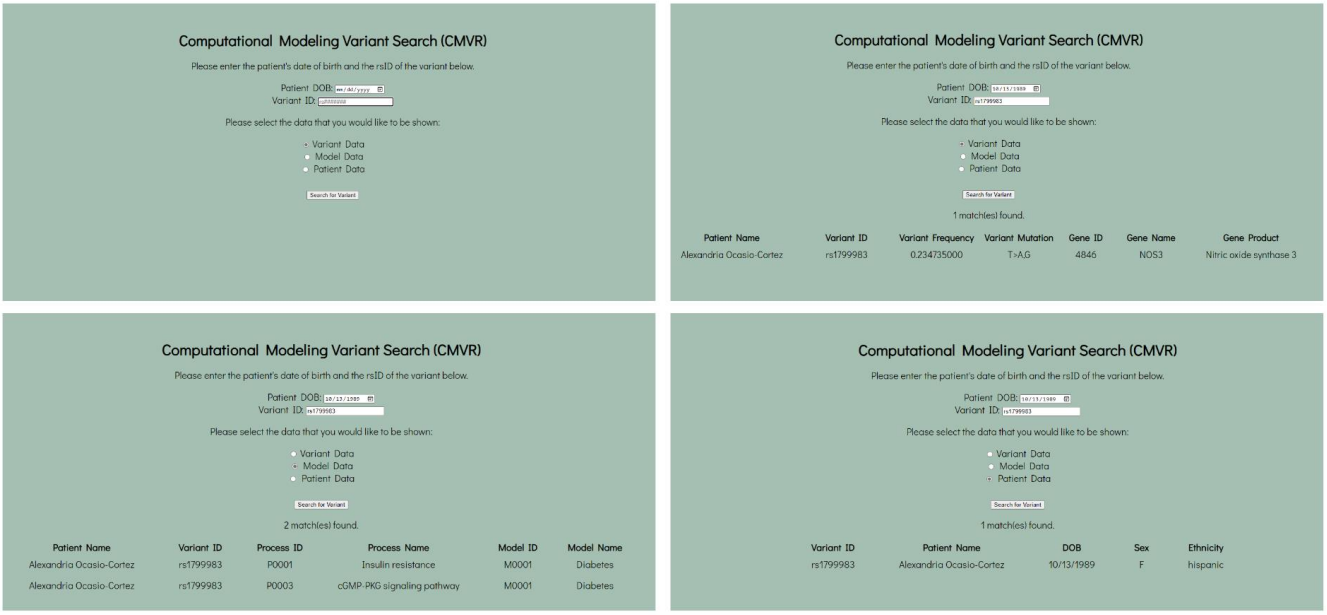

Figure 1: CMVR webpage display before and after search

Preparation

Before building the CMVR, I built a MySQL database of sample patient and variant data. The variant data is accurate from literature and GenBank. Genomic variants involved in diabetes and glycolysis were used as the sample variant data. The variant data was found in GenBank, including the information about the variant and about the gene that is mutated. Planning, creating, and populating the MySQL database took a considerable amount of the project development time, but this was a valuable learning experience and resulted in a thorough schema that I could navigate easily. The figure below shows the MySQL schema for the project.

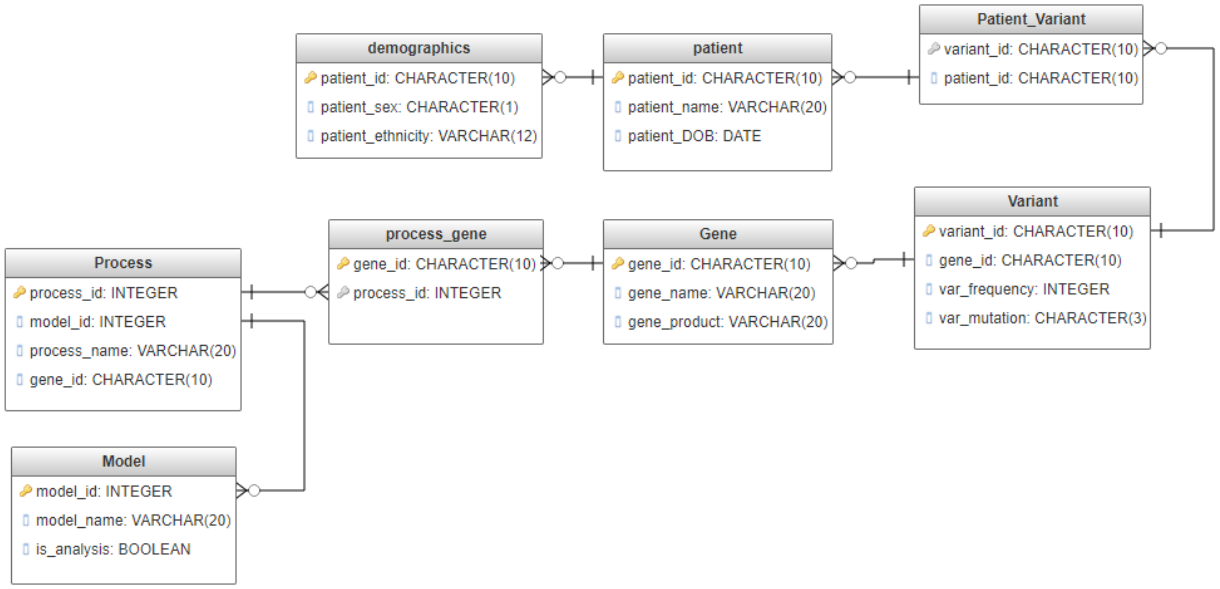

Figure 2: MySQL Database Schema

Additionally, I also spent some time looking into more CSS styling that we had covered in the lecture. I wanted my tool to look professional and usable, so I took the extra time and effort to format it nicely with a soothing background color and font.

Implementation

This project took a lot of work, but was a fantastic learning experience as I navigated several obstacles. First, I learned how to design my own relational database schema. The design went through several iterations as I added more information relevant to computational modeling. I made the schema more modular, and changed some associations between tables. In the end, I am proud of the database schema and feel comfortable searching and joining tables.

Another issue came from streamlining the formatting of variants from the HTML input page to the MySQL database. The date option originally did not match the same format as the patient birthdays in the database, so no results were returned. Once I realized the formatting issue, I changed the date formatting so that they lined up.

A similar issue was that there were several data types in the MySQL database that were not strings. The patient birthday is a date format, and the variant frequency is a decimal number. These data types are not JSON-serializable, so gave an error when the JSON tried to return the results list that they were appended to. So, I converted those specific variables to strings when appending them to the result array to avoid that issue.

My last obstacle was my goal to return different results depending on the radio button option that the user selects. I was not sure how to make variable output since our projects in the class so far only used one type of output. I made several changes to accommodate my requirement. First, I made separate functions in the CGI script with their own MYSQL queries and result list based on the different information that we want. Then, I removed the hard-coded table headings from the HTML page, so that the entire table would be made in the JavaScript file. Finally, the JavaScript file was changed such that the data to be printed to the html table was not hard-coded but rather informed by the key-value pairs in the results dictionary. The table headers are taken from the keys in the dictionary, and the table rows are populated by the value of that key. This also accounts for a different number of output variables.

Validation

This tool was verified by testing that the CMVR accurately displayed the results of traversing the schema correctly. Accurate pairs of sample patient birthdays and genomic variants were entered into the input form, and the output was confirmed to be correct.

Next Steps

There are several ways that this tool could be expanded in the future to fit the computational biology niche. First, rather than a radio option to display the desired output data, I could add a checklist of each option (gene name, gene product, process name, etc.) where the user selects exactly what information they want returned. Furthermore, I could add the patient ID to the model table so that the models are

filtered by patient to keep track of which patient has which models. These additional implementations would enhance this tool, yet the CMVR is still a strong tool as it is now.

Conclusion

To conclude, this project encompassed the entire development of the CMVR tool: creating and populating the MySQL database, writing, testing, and troubleshooting the HTML, JS, CSS and CGI files, and updating the files in Github using Git. The development of this tool flexed all of the skills that I learned this semester, and I found myself referencing almost every lecture for guidance. This project significantly increased my confidence in myself as a programmer, and I really enjoyed seeing the material I learned this semester directly apply to a tool that I would use.

Supplementary Material

MySQL Code

MySQL code to make schema:

```
CREATE TABLE patient (
    patient_id CHAR(10),
    patient_name VARCHAR(30),
    patient_DOB DATE,
    PRIMARY KEY (patient_id)
);

CREATE TABLE demographics (
    patient_id CHAR(10),
    patient_sex CHAR(1),
    patient_ethnicity VARCHAR(12),
    PRIMARY KEY (patient_id)
);

CREATE TABLE patient_variant (
    patient_id CHAR(10),
    variant_id VARCHAR(12)
);

CREATE TABLE variant (
    variant_id VARCHAR(12),
    gene_id INT,
    var_freq DECIMAL(10, 9),
    var_mutation VARCHAR(8),
    PRIMARY KEY (variant_id)
);

CREATE TABLE gene (
    gene_id INT,
    gene_name VARCHAR(6),
    gene_product VARCHAR(50),
    PRIMARY KEY (gene_id)
);

CREATE TABLE process_gene (
    process_id CHAR(10),
    gene_id INT
```

```
);

CREATE TABLE process (
    model_id CHAR(10),
    process_id CHAR(10),
    process_name varchar(50),
    PRIMARY KEY (process_id)
);

CREATE TABLE model (
    model_id CHAR(10),
    model_name VARCHAR(20),
    is_analysis BOOLEAN,
    PRIMARY KEY (model_id)
);
```

MySQL code to populate schema:

```
INSERT INTO patient (patient_id, patient_name, patient_DOB)
    VALUES
        ('0001', 'Ada Lovelace', '1985-12-10'),
        ('0002', 'Marie Curie', '1967-11-07'),
        ('0003', 'Mark Ruffalo', '1967-11-22'),
        ('0004', 'Alexandria Ocasio-Cortez', '1989-10-13'),
        ('0005', 'Michael Jordan', '1987-02-09')
;

INSERT INTO demographics (patient_id, patient_sex, patient_ethnicity)
    VALUES
        ('0001', 'F', 'white'),
        ('0002', 'F', 'white'),
        ('0003', 'M', 'white'),
        ('0004', 'F', 'hispanic'),
        ('0005', 'M', 'black')
;

INSERT INTO patient_variant
    VALUES
        ('0001', 'rs1800795'),
        ('0001', 'rs1800629'),
        ('0001', 'rs2306302'),
        ('0001', 'rs1132173'),
        ('0002', 'rs1053000'),
        ('0002', 'rs10492080'),
        ('0003', 'rs5186'),
        ('0003', 'rs1132173'),
        ('0003', 'rs608112'),
        ('0003', 'rs1800795'),
        ('0004', 'rs1799983'),
        ('0004', 'rs2274971'),
        ('0004', 'rs11168417'),
        ('0005', 'rs4908519'),
        ('0005', 'rs1803621'),
        ('0005', 'rs3785921')
;
```

```
INSERT INTO variant
    VALUES
        ('rs5186', '185', '0.211712', 'A>C'),
        ('rs1800629', '7124', '0.134935', 'G>A'),
        ('rs1799983', '4846', '0.234735', 'T>A,G'),
        ('rs1800795', '3569', '0.249794', 'C>G,T'),
        ('rs2274971', '2023', '0.279863', 'C>A,T'),
        ('rs4908519', '2023', '0.400257', 'C>A,T'),
        ('rs11168417', '5213', '0.250629', 'C>T'),
        ('rs10492080', '5213', '0.238796', 'A>G,C,T'),
        ('rs2306302', '5214', '0.097374', 'G>A,C,T'),
        ('rs1053000', '5214', '0.207133', 'C>A,G,T'),
        ('rs1132173', '5214', '0.324236', 'C>T'),
        ('rs3785921', '5164', '0.134916', 'C>T'),
        ('rs608112', '3099', '0.235970', 'A>G'),
        ('rs1803621', '2567', '0.328297', 'T>A,C,G')
;se

INSERT INTO gene
    VALUES
        ('185', 'AGTR1', 'Angiotension II receptor type 1'),
        ('7124', 'TNFA', 'Tumor necrosis factor alpha'),
        ('4846', 'NOS3', 'Nitric oxide synthase 3'),
        ('3569', 'IL-6', 'Interleukin-6'),
        ('2023', 'ENO1', 'Enolase 1'),
        ('5213', 'PFKM', 'Phosphofructokinase, Muscle'),
        ('5214', 'PFKP', 'Phosphofructokinase, Platelet'),
        ('5164', 'PDK2', 'Pyruvate dehydrogenase kinase 2'),
        ('3099', 'HK2', 'Hexokinase 2'),
        ('2567', 'GADPH', 'Glyceraldehyde-3-phosphate dehydrogenase')
;

INSERT INTO process_gene
    VALUES
        ('P0001', '7124'),
        ('P0001', '4846'),
        ('P0001', '3569'),
        ('P0002', '185'),
        ('P0003', '185'),
        ('P0003', '4846'),
        ('P0004', '7124'),
        ('P0004', '3569'),
        ('P0005', '5213'),
        ('P0005', '5214'),
        ('P0005', '3099'),
        ('P0006', '2023'),
        ('P0006', '5164'),
        ('P0006', '2567')
;

INSERT INTO process
    VALUES
        ('M0001', 'P0001', 'Insulin resistance'),
        ('M0001', 'P0002', 'Calcium signalling pathway'),
        ('M0001', 'P0003', 'cGMP-PKG signaling pathway'),
        ('M0001', 'P0004', 'Toll-like receptor signaling pathway'),
        ('M0002', 'P0005', 'Upper glycolysis'),
```

```
        ('M0002', 'P0006', 'Lower glycolysis')
;

INSERT INTO model
    VALUE
        ('M0001', 'Diabetes', TRUE),
        ('M0002', 'Glycolysis', TRUE)
;
```