

Documentation

Nayam Chowdhury, DIOnysios Grigoriadis, MichaelA matthews, DANIEL GRANT

2018

Contents

General Information

* FLASK
* MYSQL
* JQUERY

Family Table

Distribution

Translated products

AA relationship

Upload Files

$Fettucine Retrotransposon database project

Features:

* Family Table
* Distribution of retrotransposons
* Translated products
* Amino Acid relationship viewer
* Peptide Sequence identifier
* Mzid/mzTab peptide analyser
* Expression Atlas

Packages / Systems used:

* Flask
* Python
* MySQL
* JQuery / Javascript

**FLASK**

Flask is a micro web framework written in python which is also based on the Werkzeug toolkit and uses the Jinja2 template engine which allows for the use of python-like expressions in HTML document.

FLASK was chosen as the main web development toolkit as it was quick to learn and understand under the time constraints that were given.

Flask was also used as it, by default, protects against the use of cross site scripting (XSS) so malicious scripts that are injected into the website does not compromise the website security.

This micro web framework is also compatible with python 3 so that if the website needs to migrate from python 2.7, it can occur generally without too much hassle.

**MySQL**

MySQL is an open-source database which uses structured query language to be able to access data. The system is usually known to be fast, reliable and stable and works on all major operating systems.

We used MySQL as the database as it is fast and efficient to use and has the ability to query on the fly. It was also very simple to use in conjunction with flask and allowed for multiple different data tables to be created and read easily.

**JQuery**

JQuery is a JavaScript library designed to simplify the client-side scripting of HTML It was used to be able to add JavaScript functionality into the browser. The features that are available with the use of JQuery are endless as it allows you to manipulate and visualise the webpage in many different ways such as adding animations and processing events after certain specifications are met.

**HTML / CSS**

All websites require HTML to be able to run. Combined with CSS and Javascript, the possibilities that can occur are limitless. CSS allows for custom styling to be created to provide a more visually appeasing website.

We opted to go for a minimalistic visual approach to make the website seem more professional and articulate.

This was done is varying ways such as using CSS and bootstrapping. Bootstrapping is a popular free frontend framework which allows for more efficient creation of websites that are visually pleasing, responsive and have the ability to work with devices with smaller screens such as mobile phones.

The use of bootstrapping has reduced the time required to create the website exactly to our specifications by allowing faster and seamless addition of content without having to hand create the CSS manually for every object that is displayed in the website.

**Python Modules and Libraries**

The following libraries and modules were used to be able to create the website:

* Flask (flask, send\_from\_directory, render\_template, request, url\_for, flash)
* OS (operating system module)
* MySQLdb
* Pandas
* Numpy
* Bio (SeqIO and Phylo)
* Re (regex)
* Pygraphviz
* Hashlib
* Json
* Cgi
* Pylab
* matplotlib

The Flask library was used as it was the fundamental micro web framework required to run the whole system and connect the individual processes together.

**Starting of Program**

The application has been setup that it tries to connect to the database first using the try statement (Used for Errors and Exceptions) and failure to connect to a database results in the program still running, but displaying an error message in the console stating that the database has not connected. This would mean that most of the website would not function as intended.

**Family Table**

**Distribution table**

This part of the application allows users to select a chromosome to view the distribution of HERV and LINE1 repeats. This includes a dropdown function which shows the user the exact numbers of each repeat on that given chromosome which is linked to our MySQL database.

For this visualisation, the R libraries karyoploteR and BSgenome.Hapiens.UCSC.hg38 were used. The karyoploteR library was used to create ideograms for each of the chromosomes 1-22, X and Y. The HERV and LINE1 repeats were mapped to the ideogram according to their start and end positions on each chromosome. The BSgenome.Hapiens.UCSC.hg38 library was used to set the sequence lengths of the chosen chromosome to avoid incorrect mapping of the repeats. Two ‘tracks’ were visualised for each Superfamily, one for the mapping of the repeat regions to the appropriate position and the other for mapping the density of those repeats within a window of 100,000 base pairs. From this the user can easily see which part of the chromosome contains the most repeats for HERV and LINE1. This process created an image for each chromosome which could be exported from R.

The karyoploteR library was chosen as it allows users to easily customize how their data is visualised using a variety of different functions.

**Peptide sequence list**

**Relationship Viewer**

This part of the website allows the user to click the 2 main buttons (HERV and LINE1) which displays their respective relationship tree image. The trees were created using a few different programs which allowed for the use of aligning the sequence, creating consensus sequences and finally displaying the relationship between the sequences.

Muscle was used to create an alignment followed by Emboss Consensus to create a consensus sequence. Muscle was then used again to align all the sequences and a tree image file was created using an online resource tool called ITOL (Interactive tree of life).

**Additional User Functionality – Upload User Tree**

The software allows a user to upload a tree file of their own and the software will process the file and output a tree that they can visualise. The only accepted files currently are tree files in the newick format (.nwk, .newick) and .ph files.

Upon uploading a file, a new window displaying the tree file is shown. If the window does not appear, the website also automatically displays another version of the tree file in the website itself which is generated after submission of the file.

The Phylo module from Biopython is used to parse the file and then using matplotlib and phylo module, a tree image is displayed in a new pop up window which can be altered using limited functions such as zooming and panning around.

If an incorrect filetype is uploaded, the software rejects the file and displays an error message.

**Uploading of Files**

Allowing users to upload files of their own allows them to be able to search our database and identify family members that the peptide sequence has been translated from.

**Peptide Sequence Identifier (Fasta format)**

This webpage allows for user input via 2 different methods. The first method allows a user to upload a fasta sequence file to the server in order for it to analyse the file. The second method allows a user to paste a fasta sequence into a text box before pressing search.

The server first waits for a POST request method to engage before doing any further actions. If a GET request method is used, the normal webpage is served to the client with some data (up to 1000 rows) from the database being displayed. This allows the user to be able to navigate part of the database without any input.

*Fasta Checker*

Due to the possible memory constraints on the host machine, the option was taken to disallow entering in more than one fasta sequence and fasta sequences with headers via the text box. Instead the user is able to upload the file as a fasta file (including headers) where it can be parsed more efficiently using SeqIO module from biopython.

The text field allows the user to enter any length of peptide sequence with spaces and tabs in between and is able to concatenate and parse the data together (for ease of copy / paste use for the client).

If a file is uploaded that contains no data, an error message is relayed back to the user stating the file is empty.

*Incorrect Filetypes*

The program is able to differentiate between the correct and incorrect filetypes. If an incorrect filetype is uploaded, an error will be produced stating the user to upload a correct file with the correct extension.

*Parsing correct files*

If the correct requirements have been met, the peptide data is then queried using MYSQL and if a match has been found in the database, it returns the family name in a table.

*The programming*

The website is able to differentiate between input via upload box and the search box and conducts different execution of the commands depending on the route the information has taken to arrive.

When a post method is received via the text box, the program does a fasta check by looking at the presence of headers and rejects the data if it has been detected. This decision was taken as parsing a file with headers would require more memory and time than would be preferred. Also the user is instead able to upload the file (with headers) for it to be parsed quicker.

If a post method is received via the upload function, the server moves to the correct file directory to where the file has been uploaded so that it can be read into memory. A few checks are done on the file to ensure that it is an appropriate filetype of the fasta format before being read in. Failure to meet the correct specifications of the file means that the website can provide different errors from incorrect file types to empty file found.

Once the checks have been passed, the Biopython module known as SeqIO is used to parse the file into memory and a further check is conducted to see the length of the sequences. If the length is equal to one, a simple MySQL query is conducted and returns the family name if a match has been found.

The program can also detect whether the total number of sequences is below or above 5000. If it is equal to or below, the program does a simple join of the sequences to create a long query and submits it to the MySQL server. If the number of sequences exceeds 5000, the program divides the sequences into chunks of 5000 and submits the query.

**This was done to reduce the wait times and stress on the server. Recent optimisation of the code increased the efficiency by up to 10x.**

**Mzident / MzTab**

This webpage allows a user to upload a file that can be either an mzident or mztab formatted file. A user can also select the tissue type and disease progression type before submitting the file if the information is known about where the data has been sourced from.

This part of the software is able to detect the different filetypes that are uploaded and will only accept files with the correct extension.

*Parsing of files*

The type of parsing that occurs is dependent on the type of file extension that is uploaded to the server. The server is able to locate the extension and then use regular expressions to extract the information needed. The type of regex used differs between the type of files. The extracted peptide sequence is then queried in the database via MYSQL and the resulting matches are displayed in a table.

The additional information is also saved and is saved to be used for the expression atlas. The server conducts a hash check of the file to see if it has been uploaded before, if it has not been uploaded previously, the additional information is saved which contributes to the count tracker that is available on the expression atlas.

*Disease / Tissue Types*

Before submitting a file, a user is able to select the tissue and disease type. This is then saved a lot with the family names that is retrieved from the MySQL query and is used for the expression atlas.

*Hashing of files*

To ensure the integrity of the database and the validity of the data (specifically the number of counts / hits of the peptide sequences), files are hashed to create a unique ID which is then stored in a CSV file. When a file is uploaded, a unique ID using the hash checker is created. This ID is then searched for in the CSV file and if no match is found, the ID is then appended to the end and the data (peptide sequences and the resulting family) is saved into the MYSQL DB and the requested data is displayed to the user.

If the unique ID is found in the CSV, the data is not saved into MYSQL DB, however it is still analysed and provides the requested data back to the user.

*Benchmarking Of Regex*

Benchmarking was conducted on the regex function to ensure that this would not cause any problems when submitting large files (over 100mb) and satisfied the loading times that would be expected.

|  |  |  |
| --- | --- | --- |
| MZIDENT FILE | Sequences | Time (seconds) |
| 12MB | 2,458 | 1 – 1.50 |
| 59MB | 44,780 | 3 – 3.50 |
| 111MB | 22,122 | 5.50 – 6.50 |
| 1GB | ?? | ?? |
|  |  |  |
| MZTAB FILE |  |  |
| 16kb | 36 | <1 |
| 112MB | 269,100 | 9.0 – 9.50 |
| 1GB | ?? | ?? |

The results displayed show that the current code that was created to do the job seems adequate for the task required and that the amount of sequences plays a large role in the time taken compared to the size of the file.