**Manual for operating the software**

This is a manual describing the software and to execute it. One can easily change the source as per the requirement and can implement more on it.

**Creating a project**

From the command line, cd into a directory where you’d like to store your code, then run the following command:

*django-admin startproject frame*

This will create a frame directory in your current directory.

This startproject will create:

frame/

manage.py

frame/

\_\_init\_\_.py

settings.py

urls.py

asgi.py

wsgi.py

Change into the outer frame directory, and run the command:

*python manage.py runserver*

This will start the django development server, a lightweight web server written purely in python. Now that’s the server running, visit <http://127.0.0.1:8000/> with your web browser.

**Creating the software**

Now that the environment is set up, we are start to move forward.

To create the project, make sure you’re in the same directory as manage.py and run the command:

*python manage.py startapp input01*

The directory input01 which is laid out will look like this.

input01/

\_\_init\_\_.py

admin.py

apps.py

migrations/

\_\_init\_\_.py

models.py

tests.py

views.py

This directory structure will house the whole application.

Further create a file input01/urls.py

The algorithm starts with the frame/urls.py and find the path for execution. As an example the /welcome page is redirected to views.welcome function with path “ ”. It will return “welcome”. If this works properly, we can move forward to our main application.

Moving to /takeInput path in the browser will take us to the views.takeInput function. The function redirects us to the input.html page. This is the main user page where the user is asked to provide the family name, .xlsx file containing the id and sequence, patterns of each G boxes, number of mismatches and the spacing between them. When the user will click on submit, we are redirected to form action = “performAlgo”.

The performAlgo function takes the user input and starts processing. All the files are stored in media/ folder inside the root folder. We are working first with similarity function which stores the output in new.xlsx file. This file is taken as an input for further executing script\_SAX, script\_SA\_nomismatch, script\_SA\_mismatch and script\_A\_nomismatch. The output of each scripts are stored in output\_wo\_bias.xlsx, SA\_nomismatch.xlsx, SA\_mismatch.xlsx and A\_nomismatch.xlsx, respectively.

This outputs are further used for script\_X, script\_2X, script\_3X and script\_4X. The outputs as already specified are stored inside the media/ folder.

After executing all those scripts, line 848 till 874 uses the 7mer\_4X\_count\_dict.xlsx as an input and modifies it to provide us with the new G5 box pattern. Then the script\_SAX\_new is being executed with new.xlsx file as an input and producing output\_new.xlsx file as output.

Some modifications are done in the process for letting the user can open the file in the browser. The output\_wo\_bias.xlsx and the output\_new.xlsx files are converted to html pages.

The last portion deals with the negative\_control script taking output\_new.xlsx and new.xlsx as inputs and producing after.csv and before.csv as outputs. The function then transfers the call to view.html page.

The view.html page contains the results of the algorithm. It provides the user with viewing and downloading the file. For each form the action is provided in the page itself. For each call the algorithm moves to the urls.py and find the appropriate function to execute. The downloadfile, downloadfilenew, negative1, and negative2 functions deals with the algorithm for downloading the file in the users OS.

This is followed by a call when the user clicks “Show the plot” button. The redirects the call to negative3 function which deals with the plot for the negative control. This is saved on a buffer from where the user gets to see it. The next call is made automatically to the showimage function which contains the script for G\_plot and spacings\_plot. All the images are directed to get\_concat\_tile\_resize function which combines the images and process it in single image. This image is returned to the web page where the user can view it and can download it.

We are left with one more functionary. In our main page, we have a button to let the user move forward in case he just have a single sequence. This calls the negative\_control function which in turn calls the neg.html page. This page contains a placeholder where the user can enter the sequence. On clicking submit, it gets redirected to the neg\_control function which executes every parameters of each family and produces a file output\_new.xlsx with family most resembling the sequence. Then the user is redirected to view1.html which lets the user to view and download the output file.