Plotting

***Main script*: `plotting.py`**

***Webapp folder location*: `PhosQuest\service_scripts`**

***Technologies and tools*:**

`Python, Pandas, Plotly, Matplotlib, WordCloud, Flask`

***Strategy*:**

This plotting documentation outlines the methods used to analyse user input data once uploaded via the "User data analysis" section on the main website.

The plotting functionality utilizes a number of python modules. The **`Pandas`** module was utilized to allow handling of data structures. **`Plotly`** and a number of related modules and libraries were utilized to generate volcano plots and pie charts. **`Matplotlib`** was also utilized to help plot the most active kinase list. The script also utilizes some generic imports such as the session module from flask and the timeframe module from timeframe which are utilized within the script.

***Functions*:**

1) **`create_userfilename()`** : Function to create userdata_temp user id and store in session cookie.

The script has some generic code such as:-

1) **`create_userfilename()`** : Function to create userdata_temp user id and store in session cookie.

2) **`read_html_to_variable()`** : Function to open savedfile and read lines into variable.

3) **`pie_chart()`** : pie charts of "Metrics" data, summarising various distributions within the user data. The generic function **`pie_chart()`** is called from the **`userdata_display.plotall()`** function on the **`phos_enrich_pie`** , **`phos_enrich`** and **`multi_phos_res_freq`** dataframes created in the **`run_all()`** function. Each **`pie_chart()`** function has four specific input variables:-

- **Input**:
  1: A pandas dataframe to be analysed (**`phos_enrich_pie`** , **`phos_enrich`** or **`multi_phos_res_freq`**)
  2: A header for the respective column in string format.
  3: A designated name for the pie chart.

4: In integer denoting if a specifc row is to be removed from the analysis.

- The output of this function produces three pie charts:-
  i): % enrichment.
  ii): Phosphorylated AA residue frequency distribution.
  iii): Multiple phosphorylation frequency distribution.

- Generating pie charts involves two steps. Initially the row headings of the dataframes are imported as objects and stored as a series. Specific values are then imported as a list. Any data not intended to be included in the analysis will be removed at this point.

- The layout of the pie chart is defined along with the parameters of colour, line width and size.

- Variables and layout are then passed to the `plotly` function `pie()` .

- **Output**: 3 interactive pie charts as html.

4) `style_df()` : Styled tables for analysed user table and relative kinase activities. * **Input**:
1: significant phospho-site hits dataframe of `table_sort_parse()` function.
2: kinase activities dataframe of `kinase_analysis()` function. * Subset of the significant phospho-site hits dataframe is passed to a new variable. * Kinase activities dataframe not processed further. * CSS styles and auxiliary functions for passing extra styling to the tables are defined. * Dataframes, CSS styling and auxiliary functions passed into the `Pandas` function `style` . * Over-layed heat-map applied to intensity columns (significant hits table) * Over-layed barplots applied to log2 fold change and kinase activity columns of significant phospho-site hits and kinase activity dataframes respectively. * **Output**: styled html tables.

5) `user_data_volcano_plot()` : plot for displaying distribution of significantly differentially regulated phospho-sites. * **Input**: full phospho-sites dataframe of `table_sort_parse()` function. * Subset of the of the input dataframe, corresponding to phosphosites detected in both conditions, is passed to new variable. * Log2 fold changes and -log10(corrected p-values) of this dataframe is then passed to the `plotly` function `scatter()` . * Dimensions as follows: x-axis as the log2 fold change and the y-axis as the -log10(corrected p-value). * CSS styling and interactivity options passed to the scatter object. * **Output**: html.

6) `wordcloud_freq_charts()` : Wordcloud and barplots for kinase and substrate/sites frequency analysis.

- **Input**: dataframes and wordlist outputs of `kinase_analysis()` function.
  1: dataframe 1 - Kinase frequency distribution analysis (top 30 most active kinases).
  2: dataframe 2 - Substrate/site frequency distribution analysis (top 30 most targeted substrate/sites).
  3: word list 1 - kinases (top 30 most active kinases).
  4: word list 2 - substrate/site (top 30 most targeted substrate/sites).
- Dataframes fed into `matplotlib` plotting function `bar()` . Barplot objects returned with various

styling.

- Word lists fed into `wordcloud` function `WordCloud()`. WordCloud objects returned with various styling.
- **Output**: pngs of wordclouds and barplots.

***Further work*:**

1. The volcano plot has the potential for a susbtantial expansion of interaction options and visuals: * User input fields could be generated that allow the user to shift the dashed thresholds that are currently hard-coded. * Another useful option would be to enable user selection of points, such that they are labeled for image export. An alternative approach would be to have a function that displays selected hits as a table on the webapp volcano plot tab. This has been implemented in a scatter here: https://plot.ly/python/selection-events/ * The volcano plot tab could also implement extra visualisation in the form of density distributions of log10 intensity data. This may inform the user as to whether the underlying assumption of t-test calculations, i.e. data is normally distributed, can be corroborated.