ANSWER ALS VISUALIZATION TOOL

This is the github repository for the Answer ALS Visualization tool. This repository contains all code, images, and references used for building the tool. All data is stored seperately on the SQL Server.

Table of Contents

Dependencies

This project is dependent on the couple of core dependencies. For best results with development, we refer you to our Getting Started Section. For the full list of dependecies, please check out the requirements.txt or the als_viz.yml.

Getting Started

In order to quickly set up your environment to develop for this tool, we have two methods for doing so. We reccommend the first method as it is by far the easiest for setting up your virtual environment. We reccomend the second method if you prefer not to you use conda/mamba, but you will need to use pip then. Regardless, of the method, we still reccommend setting up virtual environments for developmental purposes. Below, we will show you the first method.

First Method: Quick Installation via Virtual Environment Importing

Conda Method:

```
conda env create --file als_viz.yaml
```

Mamba Method:

```
mamba env create --file als_viz.yaml
```

This will auto-install all of the necessary packages needed for developing for this tool and running this tool in its current state.

First Method: Installation via pip

Conda Method:

```
conda create -n <environment_name>
```

Mamba Method:

```
mamba create -n <environment_name>
```

Moving forward for this method, the guide will follow using mamba as our package manager, but the commands will be exactly the same bewteen using conda and mamba. Please look at mamba documentation or conda documentation if you have more questions.

Once you have created your environment, please active the environment so we can install the necessary dependencies as shown below:

```
mamba activate <environment_name>
```

You will know your evironment has changed because the (base) parameter on your terminal screen will change to the name of your environment you just activated. Next we will install the new package manager we need to install the packages we need:

```
mamba install pip
```

From here, we will just install the packages using pip:

```
pip install -r requirements.txt
```

Finally once you have everything installed, you need to install the ODBC Driver 17 for SQL Server provided by Microsoft. You need to make sure you have this installed prior to running as this is how SQL querying works. More details on the SQL querying of the data can be found in the SQL Querying Section.

To test if everything runs correctly, we reccomend quickly running the following command:

```
panel serve multi-page.py
```

It should open you to the local web-page version of the tool on the website. If you reach the tool page, then you have successfully installed the development environment.

Widgets

For the widgets, there are two classes of widgets we use: Veutify and Panel based widgets. ### Veutify Widgets

dataset_button: This widget uses the v.Select() widget. It allows the user to switch between the datasets. It is also a widget that is linked to the graph_button widget via the _data_choices dictionary. When a dataset is selected, it calls the plots that are available for that dataset from the _data_choices dictionary and loads those in as the available options to be selected in the graph_button widget. For more details on the v.Select() widget please see the ipyvuetify documentation.

graph_button: This widget uses the v.Select() widget. It allows the user to switch between the different types of plots available. This widget's input is dependent on the _data_choices dictionary. So whatever dataset is chosen, the list of available visualizations of that dataset are loaded into the widget as the available options. For more detail on each plot, please see the Visualizations section.

user_dropdown: This widget uses the v.Select() widget and allows the user to select which pre-made gene list they wish to explore. The user_dropdown widget's input is dependent on the **_user_dropdown_options dictionary** which contains all of the available gene ID or UniProt ID pathways. Once a pathway is selected, the gene/Uniprot ID's included in that pathway are inserted into the **user_input** widget via using the ID list stored in the **_user_input dictionary** that correspond to that pathway.

multicovariate_selector: This widget uses the v.Select() widget and allows the user to select which covariates they wish to use in their visualization. This widget allows the user to **select multiple covariates at the same time**. It is only used for the Clustermap visualization.

covariate_selector: This widget uses the v.Select() widget and allows the user to select which covariate they wish to use in their visualization. This widget allows the user to select **only one covariates at a time**. It is used in all other visualizations when applicable.

pca_covariate_selector: This widget uses the v.Select() widget and allows the user to select which covariate they wish to use in their visualization. This widget allows the user to select **only one covariates at a time**. It is used specifically for the PCA plots.

norm_selector: This widget uses the v.Select() widget. It allows the user to switch between the different types normalization options that are available which are detailed in the Normalization Section.

Panel Widgets

user_input:

This widget uses the TextAreaInput widget. Its input is free to be customized by the user, meaning the user can input a gene list themselves manually. Its input is directly taken from the **_user_input dictionary** when a gene or protein pathway is selected. **Note**: In the future,we will be adding an upload gene list option.

participant_input:

This widget uses the TextAreaInput widget. Its input is free to be customized by the user, meaning the user can input a participant list on their own if they wish. This widget directly takes input from the Answer ALS Data Portal. All partcipants selected from the data portal are automatically loaded in as the input to this widget. This is done via the requests python package which allows us to parse the json datafile sent through the URL.

p_value_input:

This widget uses the FloatInput widget. It allows the user to choose a p-value from a range of 0.05 to 1. By default, it starts out with 0.05 as a p-value. This widget only appears on certain visualizations like the volcano plot. The value chosen for this widget allows the user to set the threshold for significance.

log2FC_input:

This widget uses the FloatInput widget. It allows the user to choose a log2(Fold Change) from a range of 1.5 to 5. By default, it starts out with 1.5 as the Log2FC value. This widget only appears on certain visualizations like the volcano plot. The value chosen for this widget allows the user to set the threshold for significance.

positive_col:

This widget uses the ColorPicker widget to allow the usee to select which color they wish positive values to appear as on a plot. By default, it is set to visualize positive values as blue (hex:#1e7333). The user is allowed to select any color via the eye dropper tool included in the widget or by manually input the RGB values. This widget is only used in the Clustermap Visualization.

negative_col:

This widget uses the ColorPicker widget to allow the usee to select which color they wish negative values to appear as on a plot. By default, it is set to visualize positive values as blue (hex:#1e7333). The user is allowed to select any color via the eye dropper tool included in the widget or by manually input the RGB values.

Functions

****_update_input:**** This is a updating function that basically tells the computer to keep track of the user_dropdown widget and makes sure that if the user selects a pathway, it updates the user_input widget with the gene/protein pathway stored in the **_user_input dictionary**.

switch_data: This is an update function that does all the backend for switching datasets. It makes sure that all of the widgets update accordingly to each dataset and makes sure that no participants that are not included in the newly selected dataset are carried over.

load_plot: This function is reponsible for making sure the correct graph is loaded onto the main page for the visualization tool. It essentially checks which plot is selected in the graph_button widget, and then grabs the correct visualization and puts that onto the main page of the tool.

****_update_plot:**** This helper function makes sure that all plots are updated with the current options selected by the wigdets. It is specifically dependent on the following widgets: - multicovariate_selector - covariate_selector - pca_covariate_selector - norm_selector

Normalizations

normalize This function is where we formally define all of the normalization options and their calculations associated with each method. Currently,we support the following normalization methods: - **None** - This option allows the user to plot raw counts without any normalization done to the data, besides the base conversion to TPM's. - **Log Norm** - This option allows users to log10(1+x) normalize the data to deal with the 0 values in the data. - Note: In the future, we plan on offering different options for users to choose how they wish to deal with the 0's in the data. - **Z-Score** - This option allows the user to perform the Z-score normalization on the data. This normalization is carried out using the scipy stats package, specifically their zscore function. - **Quantile Norm** - This option allows users to perform a Quantile normalization on the data. This normalization is carrioud out by sorting the data, then calculating the mean, and then using the rank function of dataframes in pandas, with method set to min.

SQL Querying SQL querying works using ODBC Driver 17 for SQL Server provided by Microsoft. You need to make sure you have this installed prior to running, otherwise you will not have access to the data. We use the pyodbc package here to connect the code to the SQL database.

tpm_sql_query: This function queries the TPM data for transcriptomics data from the data sql database.

de_sql_query: This function queries the differentially expressed data for transcriptomics data from the de_data sql database.

cov_sql_query: This function queries the covariates from the covariates sql database.

protein_sql_query: This function queries the proteomics data from the protein_level database.

protein_de_sql_query: This function queries the differentially expressed data for proteomics data from the protein_de sql database.

Visualizations

clustermap: This function is reponsible for carrying out all preprocessing the data and actually plotting the preprocessed data in a clustermap. This plot uses the Clustergrammer2 package

boxplot: This function is reponsible for plotting the boxplot visualization. It uses the Boxplot plotly package.

volcano_plot: This function is reponsible for plotting the volcano visualization. It uses the Volcano plotly package.

pca_plot_2d: This function is reponsible for plotting the PCA 2D visualization. It uses the PCA 2D plotly package.

pca_plot_3d: This function is reponsible for plotting the PCA 3D visualization. It uses the PCA 3D plotly package.

dotplot: This function is reponsible for plotting the dotplot visualization. It uses the Dotplot plotly package.

violinplot: This function is reponsible for plotting the dotplot visualization. It uses the Violin Plot plotly package.

distplot: This function is reponsible for plotting the dotplot visualization. It uses the Distribtion Plot plotly package.

protein_pca_plot_2d: This function is reponsible for plotting the PCA 2D visualization for protein data. It uses the PCA 2D plotly package.

protein_pca_plot_3d: This function is reponsible for plotting the PCA 3D visualization for protein data. It uses the PCA 3D plotly package.