**Creates GUI Component**

\_\_init\_\_

* Initialize component and store app\_state
* Declares all the GUI component variables
* Calls **create\_components**
* Calls **setup\_layout**

create\_widgets

* Initializes all of the GUI component variables
* Adds bindings to required GUI components
* Adds the styling arguments to GUI components

setup\_layout

* Creates a row and column configuration for the GUI
* Adds all of the GUI components to itself

**Creates Plots During Runtime**

**Scree Plot**

create\_scree\_plot

* Shows an error if the df has not been cleaned
* Calls **main.run\_analysis**
* Calls **main.create\_blank\_fig(grid=false)**
* Adds a title and an x and y label to the ax
* Gets the explained variance from pca\_results in the app\_state
* Uses a try block to add the bar and step chart to the ax
* Calls **main.update\_figure**

**Biplot**

create\_biplot

* Calls **validate\_biplot\_data**  to get PCA result data
* Calls **init\_biplot\_fig**
* Calls **get\_color\_mapping**
* Adds legend box to the ax using color map
* Adds a confidence ellipse to the ax
* Scales Loadings
* Calls **set\_biplot\_axis\_limits**
* Calls **add\_biplot\_arrows**
* Adds the scatter plot points to ax
* Calls **main.update\_figure**

init\_biplot\_fig

* Calls **main.create\_blank\_fig**
* Sets title and x and y label on the ax
* Sets the grid appearance and aspect ratio of the ax
* Sets the background color of the ax

set\_biplot\_axis\_limits

* Calculates the minimum and maximum x and y values from the scaled\_loadings
* Calculates a margin based on the max x and y range
* Sets the ax axis limits using the x and y min and max values and the margin

add\_biplot\_arrows

* Creates a list for holding text objects
* Calls **get\_color\_mapping** to get the feature map and color
* Loops through the indexes in top\_idx
  + Gets the feature and magnitude
  + Skips the feature if the magnitude is less than 0.2
  + Gets the group and color
  + Generates and arrow on the ax
  + Gets the text distance from the app\_state
  + Generates a text object and adds it to ax
  + Store the text object in the text object list
* Adjusts the text objects to attempt to minimize overlap

**Interactive Biplot**

create\_interactive\_biplot

* Calls **validate\_biplot\_data** to get the PCA results data
* Initialize function internal figure as a Go figure
* Scales Loadings
* Calls **add\_interactive\_biplot\_groups**
* Updates the figure layout with a menu and sliders
* Updates the figure layout with the top 2 PCA components
* Calls **save\_interactive\_plot** from file\_operations
* Shows a success message if the plot saved properly

add\_interactive\_biplot\_groups

* Calls **get\_color\_mapping** to get the feature map and color
* Creates a set for groups on the legend
* Loops through the indexes in top\_idx
  + Gets the feature and magnitude
  + Skips if the magnitude is less than 0.2
  + Gets the group and color of the feature
  + Determines if the group is already in the legend set
  + Adds the group to the legend set
  + Adds a (arrow) trace to the figure and adds group to the legend if it hasn’t already

**Top Feature Plot**

create\_top\_n\_feat\_plot

* Shows an error if the df has not been cleaned
* Calls **main.run\_analysis**
* Gets top\_n\_feat from app\_state
* Gets pca\_comp\_num from app\_state
* Calls **init\_top\_feat\_plot**
* Gets the loadings and feat\_names from pca\_results in app\_state
* Sorts the loading and feature names
* Adds a bar chart to the ax
* Calls **main.update\_figure**
* Shows a message that the plot was generated successfully

init\_top\_feat\_plot

* Calls **main.create\_blank\_fig(grid=False)**
* Sets the title and the x and y labels on the ax
* Adjusts figure tick label size and rotation to fit on the ax

**Data Functions**

validate\_biplot\_data

* Shows an error if the df has not been cleaned
* Shows an error if custom feature groups are enabled, but have not been uploaded
* Calls **main.run\_analysis**
* Gets the pca\_results from app\_state
* Gets scores, loadings, explained\_varience, and feature names from pca\_results
* Gets the user input for number of features
* Calculates eigenvalues from variance
* Calculates magnitudes from loadings
* Gets the indexes and features with the top magnitudes
* Returns scores, loadings, variance, eigenvalues, feature names, top indexes, magnitudes, and number of features.

get\_color\_mapping

* Gets the current color palette selected by the user
* If feature grouping is enabled
  + Shows and raises an error if no feature group has been loaded
  + Gets the groups from the feature group map
  + Splits the groups into predefined and undefined groups. Where predefined is groups that have color assignments from the selected color palette and undefined groups do not.
  + Creates a dictionary and adds the predefined groups with the colors from the color palette
  + Calls **map\_generic\_colors(undefined\_groups)**
  + Adds the results to the dictionary and returns the dictionary
* else
  + Checks that features have been provided
  + Calls and Returns **map\_generic\_colors(feat)**

map\_generic\_colors

* Checks that 20 or less features have been provided
* Collects a color blind friendly palette, plt ‘tab10’ if 10 or less features are provided
* Otherwise collects a non-color blind friendly palette, plt ‘tab20
* Creates a dictionary mapping each of the features to its own color
* Returns the dictionary

upload\_mapping

* Opens a filedialog asking the user to select a csv file
* Shows an error message and returns if the user doesn’t select a file
* Shows and error message and returns if the user selected file is not a csv file
* Reads the csv file and stores it in a df with lowercase column names
* Shows an error message if the df doesn’t contain a ‘feature’ and ‘group’ column
* Generates a feature mapping using the df and saves it in app\_state
* Gets the unique groups from the df
* Creates a group color map using tab20 colors and the unique groups
* Stores the group color map to app\_state
* Shows a success message