EDAT Documentation v.1

**Background**

**Objectives**

* The overall objective of the project was to create an app using R-Shiny in order to allow a person to explore gene expression data without extensive computational knowledge and rigor. The app will incorporate two dimensionality reduction algorithms, PCA and t-SNE, to help aid in data visualization. The user then may further visualize the data by selecting only specific genes, time points, and donors they wish to visualize.
* In order to accomplish this, some processing is needed to convert the raw fold change data into a usable format. Currently the program only accepts pre-processed gene expression data using the ration of change over the original expression level, but we hope to have the program process raw data in the future. Once the data has been processed, the user is able to subset the data using the UI which calls the subset() function in the program which extracts only the selected data. See the image in results for more information.
* In regards to the PCA performed by the program, we are using the prcomp() function which accepts the scale and center parameters which will soon be modifiable by the user. The variance of each dimension is also computer with the PCA and is displayed in a corresponding scree plot.
* For the t-SNE performed we are using the function Rtsne() which is part of the RTsne package. We will implement the function such that the user can adjust the dimension, initial\_dimension, perplexity, pca, and max\_iter parameters.

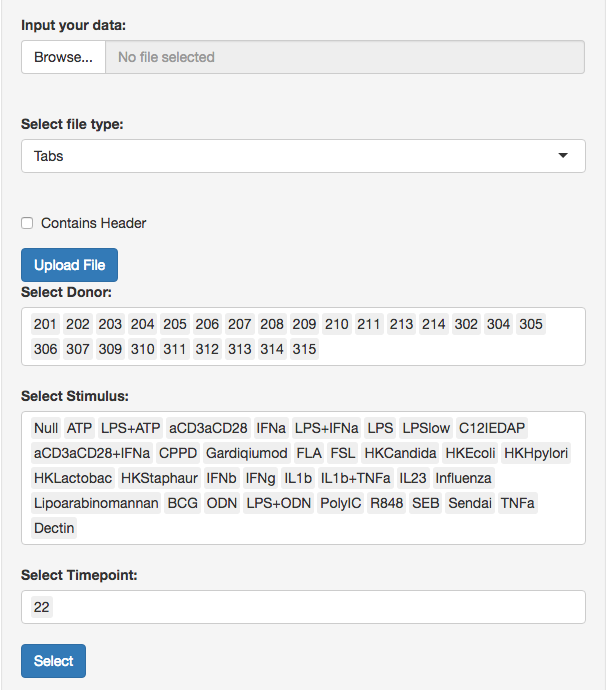
**Results**

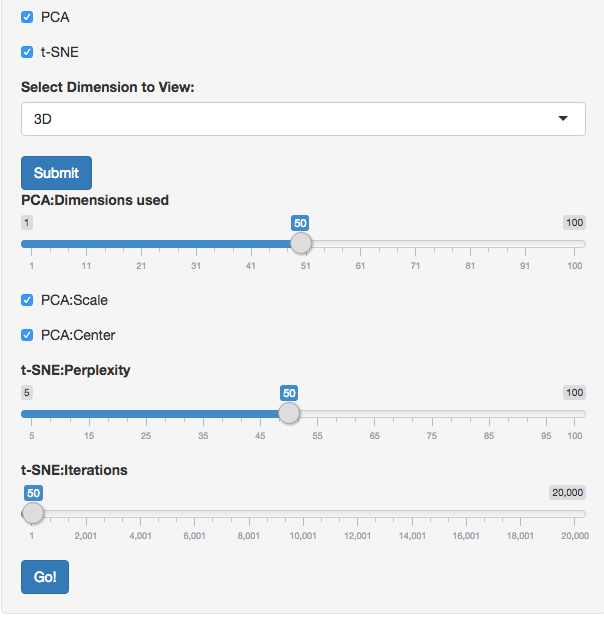
Pictured to the left is the sidebar for the application. It allows for the user to upload a file they wish to be parsed, select what the deliminator is and whether there is a header and begin upload. After the file is uploaded the selection menu is presented to the user.

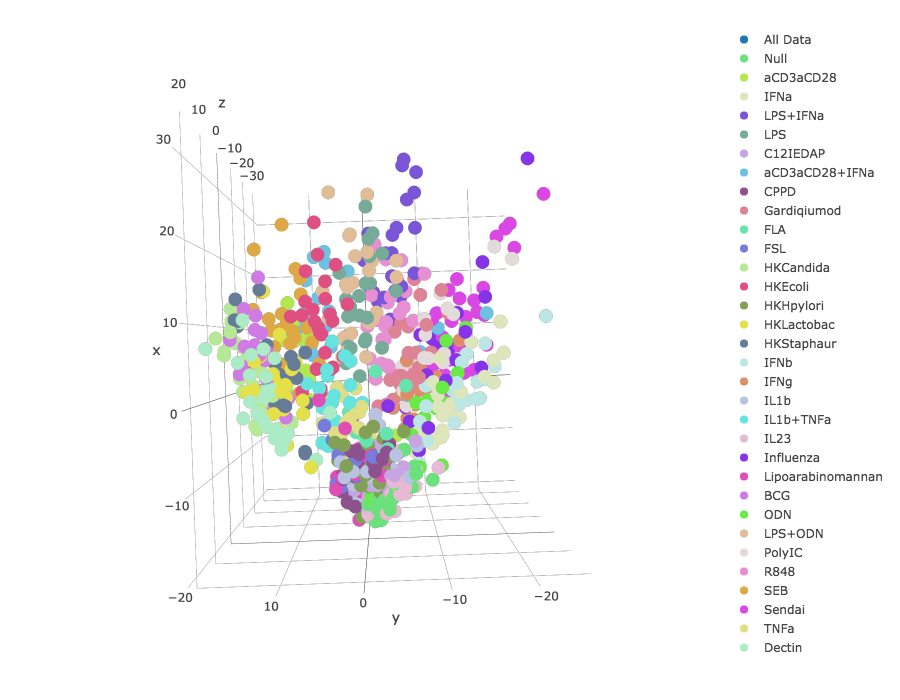
This menu is seen directly underneath the upload menu and allows for the user to select which donors, stimuli, and time points from the original file they want to include in the analysis.

After pressing select, the user is then prompt to choose the dimension and algorithm they wish to use. They are given a choice between Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE) or both.

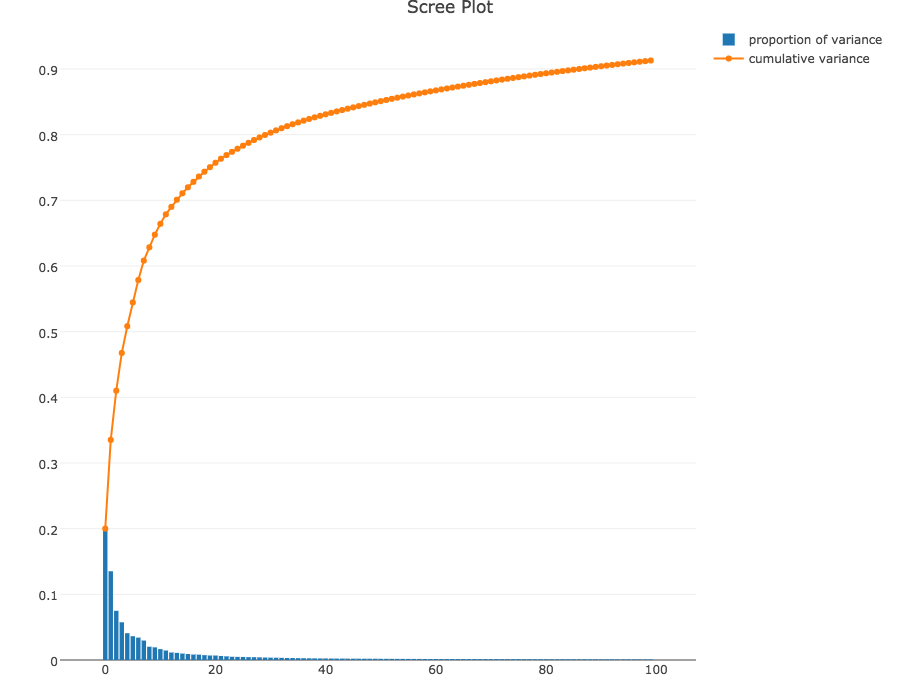
Once those two settings are submitted, the user is able to tune parameters relevant to the algorithm they chose and then produce the plot by pressing Go.

****

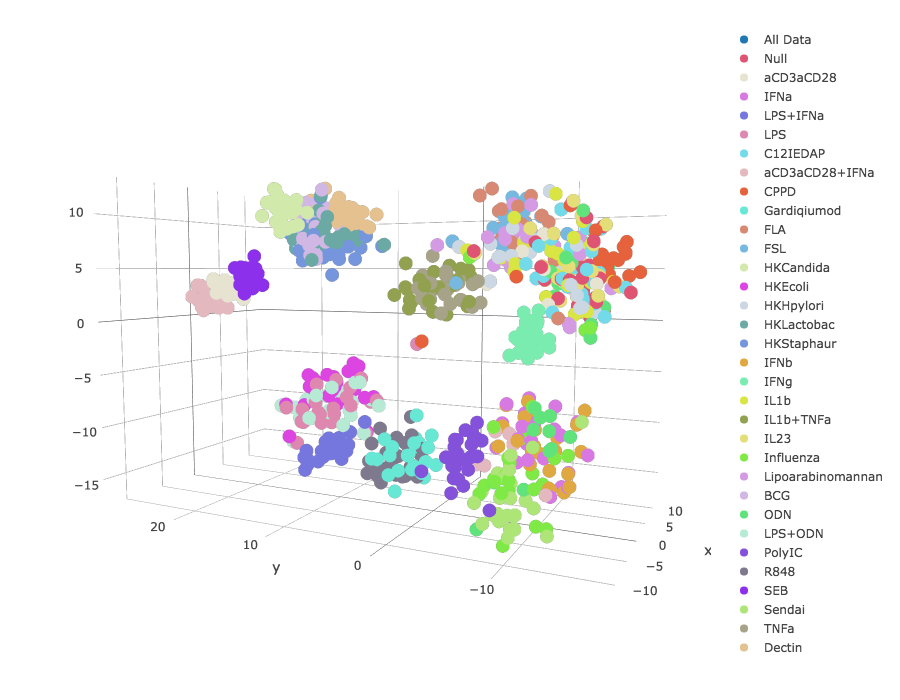
****

****

One of the graphs generated by the app. As you can see the dataset has been reduced from over 500 dimensions to just 3 so we can view it on a 3D plot. With PCA the dataset is reduced to a little over 200 dimensions but the information lost by cutting off the dimensions after 3 is small enough that we can still see some patterns in the data’s 3D plot. The information loss is shown in the scree plot explained below.



For each run of PCA a corresponding scree plot is displayed showing the variance captured by each dimension and the cumulative variance captured over a span of dimensions. This plot allows the user to see how much information is lost with the reduction of the data into the 2 or 3 dimensions seen in the PCA plot.



Above is an example of the PCA+t-SNE plot outputted by the program. Layering t-SNE on top of PCA usually yields promising results as opposed to using just t-SNE so the option was added to the program. The user has the ability to adjust parameters such as:

* PCA dimensions used – number of dimensions retained in original PCA step.
* Scale and center – manipulate the data to center/scale before applying PCA.
* Perplexity - number of nearest neighbors similar to the k nearest neighbors seen in many manifold learning algorithms.
* Iterations – number of iterations for the optimization algorithm.
* Learning rate- the sizes of steps taken when optimizing.

**Difficulties and Areas of Improvement**

* One difficulty we’ve had trouble with is the use of R-Shiny, while it helps create an easy-to-use interface, it also sometimes has some not very intuitive design choices, such as the interaction between plot traces and hovertext which has led us to have a bug where hovertext and traces cannot be used at the same time.
* Another issue we’ve had is the sheer amount of data being processed. In order to get around slow runtimes we have halted the dynamic nature of Shiny by introducing buttons which only allow computations to occur when they are pressed. This way if the user changes 1 parameter but needs to change 3 overall, 3 different computations will not occur in succession, instead only 1 will occur when the users presses the button.
* Another useful feature would be a progress bar or updates from the console as the program runs for the more time-heavy functions. These are easy to implement with functions defined within the program, but the dimensionality reduction functions were found in other packages, yielding it difficult to edit them in order to incorporate the progress updates.

**Conclusion**

* There is work remaining in the UI of the app. Currently the user must push several buttons and go through multiple selection screens in order to generate the plot and it is not the most intuitive thing to navigate.
* Another possible next step is in experimenting with different implementations of the dimension reduction algorithms. Different algorithms use different methods of solving the reduction problem so some are perhaps more useful than other for our dataset.
* We also plan on creating a dynamic display of the t-SNE algorithm being used on the data. This display would show how the plot looks after each iteration of t-SNE, so the user could use this information to optimize their parameters.

**Usage**

With Processed Data:

With Raw Data: