The first task I completed during this internship was to build an interface that allows myself and future researchers to easily access the data and format it for analysis. The interface can be accessed at the following link: <https://github.com/AaronKruchten/PSC-EDA> . Below is a table that lists the name of some of the most useful functions and their purposes.

|  |  |
| --- | --- |
| query\_all\_single\_flow | Input: flow\_name, database, save\_location, database username, database\_password  Output: A folder created at the save\_location given by the user with the same name as the given flow name containing separate csv files for each measurement indexed by time.  Further Info: This function creates a useful data directory that form\_dataframe can be called on, |
| form\_dataframe | Input: directory, measurements  Output: An R data frame of the given measurements indexed by time.  Further Info: Some measurements are not recorded at all times and so when this occurs the function will fill in NA for these values. The directory given has to be a directory created by query\_all\_single\_flow |
| query\_by\_flow | Input: flow\_name, measurments, database, database username, database password  Output: An R data frame of the given measurements indexed by time.  Further Info: This function behaves the same as form\_dataframe, however it accesses data directly from the influx database. This is more convenient if you want analyze a flow one time and be done with it. If you are planning on running many different analyses with different measurments on a specific flow, then form\_dataframe and query\_all\_single\_flow should be used because it will be much faster. |
| impute\_frame | Input: dataframe, k number of neighbors  Output: An imputed R data frame  Further Info: This function imputes(infers NA values and fills them in) a dataframe outputted by form\_dataframe. It should work for an arbitrary k but k=10 seems to work the best. Additionally, if a column in a dataframe is entirely NA values this function will do nothing, and those columns should likely be removed.  This function should also work with frames outputted by query\_by\_flow, but I have run into issues before. |

Even though this interface is written in R, the frames outputted by these functions could very easily be converted to Pandas frames with the R library reticulate.

After completing the interface, I began analyzing the data. My first step was to plot many of the measurements and test my knowledge of the TCP/IP networking protocol and see if things that I believed to be correlated were actually correlated. Then, I wrote code to look for flows with very large data transfers as well look at the frequency of commands in our data set. It appears the globus-gridftp commands transfer the most data. I also tried several different causal discovery algorithms and went through the data to test if the assumptions of these algorithms were valid. I finally settled on using the causal discovery algorithm in the rcausal library. I have also attempted to cluster the data in a useful way. I have attempted to cluster the data both by measurement, and by flow. Clustering by measurement did not turn out to be very useful, but I believe clustering by flow has been and I will continue to follow up on that this week. I also attempted to apply the macroscopic model to see what the probability of packet loss was for many of the flows, and to compare which commands were most likely to have data loss. Using the directed acyclic graph (DAG) outputted by rcausal and tweaking it using domain knowledge we could build models to predict causal effects by controlling for other variables according to the DAG.

I have also tried a few other things such as building models to predict congestion, data transfer and other interesting measurements, and many other things that did not appear to be useful and I won’t go into detail here.