**Phase I**

Simulating Data

* Packages
  + dpylr
    - Contains the commands select, filter, mutate, summarise, left\_join, group\_by and the %>% notation
  + tidyr
    - Contains the gather command
  + ggplot2
    - The graphing package and notation I chose to use
  + EnvStats
    - This is the package that contains the ebeta function
* Set Parameters
  + Mu is the mean used in the normal distribution
  + Sd is the standard deviation used in the normal distribution
  + Rho is the correlation coefficient used when obtaining xy pairs
  + Num\_of\_sims is the number of correlation coefficients we obtain before we find the beta distribution for that distribution
  + Num \_of\_ab is the number of alpha and beta pairs one would want at the end of the simulation
* Create data frames
  + df\_nt
    - Create a matrix filled with NA’s with a column for each combination of n and theta for this batch. (Label each after the combination)
    - The number of rows should be equal to the num\_of\_sims
  + df\_RV
    - Create a matrix filled with NA’s with two columns for each combination of n and theta for this batch. One column with store the alpha for that combination of n and theta, and the other will store the beta.
    - A stands for Alpha
    - B stands for Beta
* For Loop
  + Inside Loop (first section, for(I in 1:num\_of\_sims))
    - At the specified combination of n and theta
      * Simulate two standard normal distributions with a sample size of n
      * Cbind takes the two vectors and combines them and puts them into a data frame which makes it much easier to work with
      * The next formula uses the specified rho value to obtain xy pairs
      * The next patch of lines which includes the pipe notation (%>%) reads as “Take the xz95 data frame then (%>%) only select the rows (filter) that are less than the value of qnorm(.95). Store all of this as xy95 (🡨).”
        + This is where the censoring occurs
      * The next batch of lines I find the correlation coefficient, and the absolute value of that correlation coefficient
        + Then store that |r| in the df\_nt data frame in its specified column for that combination of n and theta
    - At the end of one loop for this for loop you will have obtained one |r| for each of the combinations of n and theta in this batch
    - Because the loop is for(i in 1:num\_of\_sims) the inside loop will repeat however many times num\_of\_sims specifies, which will give us an output of that many |r|
  + Outer loop
    - For each of the column in the df\_nt data frame find beta distribution to represent those |r|
    - Below the inner loop (separated by })
      * The ebeta function takes a vector and fits a beta distribution to the distribution (in list form)
      * The next two lines in this section pulls the beta distribution parameters and stores them into the correct column given its n value, theta value, and whether it is an alpha or beta value
        + At the end of one loop there is an alpha and beta value for each combination of n and theta.
    - The outer loop says to repeat this entire process num\_of\_ab times
* Wide to Long Data
  + Separate the df\_RV by n and theta
    - Each batch of lines reads “Take df\_RV and only select these two vectors (select) then add a column called (mutate) n which will always be equal to the given n for this combination of n and theta. Then add another column (mutate) called theta which will always be equal to the given theta for this combination of n and theta. After all of that store the new data in the df.long for that combination of n and theta(🡨).”
    - Change the names of the original columns to make more sense
  + Use rbind to stack all of the separate data frames on top of each other
* Summarize the Data
  + First the Beta column is still in a list some reason so I used the unlist function to change it to a vector
  + The next group of lines reads “Take df\_RV and create groups based off n and theta (group\_by). Then for each of these groups find an average alpha (summarise) . Store it as df.alpha (🡨)”
  + Graph
    - Use this data frame
    - But theta on x axis
    - Alpha value on y axis
    - Have n represented by changes in color
    - geom\_point adds points without any additional information. The information in there changes point shape and size
  + Repeat for beta
  + Join the df.alpha and df.beta by n and theta (left\_join)
    - Save the resulting data frame

Combining the Data

* All I do in this script is load in the data frames I created at the end of the simulation process for each batch of n values we studied and combined them with rbind.
* This data frame is now saved and is our master alpha data