Independent Variable Analyses of Tuberculosis Drug Studies Using R (Group 2)

Alexia Alfano, Rebecca Foos, Colleen Brents, Mackenzie Fry

December 14, 2017

knitr::opts\_chunk$set(echo = TRUE, message = FALSE, warning = FALSE)

### Rationale and Idea Development for Independent Variable Analyses

This project investigates the independent variables from *Mycobacterium tuberculosis* (TB) drug studies, which use mouse species as animal models. The independent variables include variables that were measured both in-vivo and in-vitro. This group chose three different visual analysis methods to look at these independent variables including beeswarm plots, correlation plots, and hierarchical clustering plots.

For each type of plot the goal was to help the researchers understand the connections between the different independent variables based on their data. As a group, we explored three different packages that would assist the researchers in understanding the independent data, these include ggbeeswarm for the beeswarm plots, ggcorplot and ggpolypath for the correlation plots, and ggdend for the dendrograms that represent hierarchical clustering.

Beeswarm plots were made using the R package called ggbeeswarm (Clarke & Sherrill-Mix, 2017). A beeswarm plot, also commonly known as a “violin plot”, allows multiple groups of one dimensional data to be plotted in a way that is more representative of the underlying “distribution” of the data, it also helps prevent overlapping of data points that you see in other types of plots. In this case, it helps to show a pattern of distribution of TB drugs, according to the dosage-interval of the drug given and according to the resulting quantities of drug found in different measurements of mouse/human factors.

Hierarchical clustering plots were made using the ggdendro package in R. (Vries & Ripley, 2016). There are many different variations of hierarchical clustering of independent data. Some of the ways we investigated displaying the clusters included different geom\_edge layouts from the ggraph package (Pederson, 2017). Some of the layouts that were considered were elbow, diagonal, and 2-variant. We determined that the elbow geom\_edge was the best layout, and those graphs inspired us to look into dendrograms which was the final clustering plot we chose. Dendrograms provide a visual using the nodes to identify which variables (either by drug and dose or by measurement) are the most similar by placing them on the same nodes.

**Add quick summaries about other plots here**

This group pushed forward with all the analyses initially proposed from the exploration stage; however, the biggest challenges were in attempting to understand the methodology of the different packages. Many of these packages use techniques that were, at times, more advanced than our R skills. These challenges were addressed using stack overflow and vignettes for the packages to see examples of how they are used ("Stack Overflow - Where Developers Learn, Share, & Build Careers", 2017).

*Describe the different ideas your group explored. What were the biggest challenges in this stage? For any ideas that didn’t pan out, what were the key constraints? Also describe how you would tackle this problem if you were starting over.*

*Key functions: Describe the final functions / app framework you decided on. Explain why you picked these. For functions, include documentation for the functions:*

### Beeswarm

#### Beeswarm Coding Methods

For the beeswarm plots the data was split up between invivo and invitro independent variables. The code for both invivo and invitro is logistically the same and there are several key components in the code which are worth noting. The unite(), gather(), and mutate() have comments next to them describing their importance. Please see comment code describing the key components of this process within the functions.

#### Beeswarm Functions

The beeswarm functions called invivo\_beeswarm\_function() and invitro\_beeswarm\_function() follow suit with the division of the independent variables between invivo and invitro. The functions are built to produce visual outputs. The two functions operate with almost identical logistics in terms of the code, so the following is an explanation of the main concepts used to design and build both functions.

One goal of the functions is to allow the researchers to plot all of the independent variable data quickly and efficiently. Another goal of the functions is to provide the researchers with the power to control which data they want to visualize and/or compare across all of the independent variables measured. There are particular parts of the function code that accomplish these goals. By setting the statements for variables and drugs to NULL, it tells the function to produce plots of all the variables and drugs *unless* otherwise specified. If the researcher inputs invivo\_beeswarm\_function(efficacy\_summary) then all the small multiple plots and all the drugs will be returned.

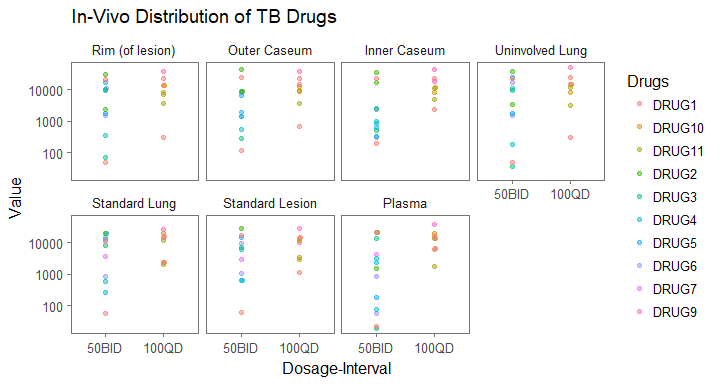
The following *if* statements in the function are what direct the function in that "unless otherwise" specified situation. When the variables statement or drugs statement is *not* null, the function tells it to filter the variables or drugs as requested in the input from the “variable\_filtered” or “Drugs” columns of data. As an example, if the researcher types specific variables or drugs they want to plot together, then the function will plot only what they ask it to. If you are wondering how the “variable\_filtered” differs exactly from the “variables”, this aspect is explained in detail below, under the topic “Problems We Faced”.

The beeswarm functions were designed with these capabilities following discussion with the researchers who provided/and are familiar with the independent variable data. Because all of the data together can be quite overwhelming to look at, it became clear that the researchers were interested in the ability to separate out the plots as much as possible so comparisons of specific variables and drugs can be explored upon their discretion.

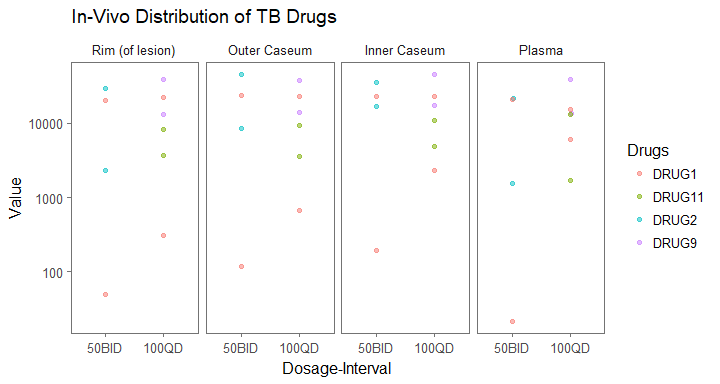
The beeswarm functions employ plotly and ggplotly() in particular to add an interactive element to the plots, however the code lines for this needed to be left out of this write up in order for Markdown to knit this document. This is a key component of the code that was given to the group working on building the Shiny App, which allowed the final plots built in the app to have detailed descriptions appear when the mouse is placed over any point in any of the small multiple plots. The Shiny App group also contributed to the creation of radio (check) buttons to go along with these functions, which ultimately allow the researchers to make selections of specific variables and drugs to compare within the Shiny interface.

#call the following libraries prior to running code   
library(dplyr)  
library(readr)  
library(ggplot2)  
library(ggbeeswarm)  
library(ggthemes)  
library(tidyr)  
library(scales)  
library(plotly)

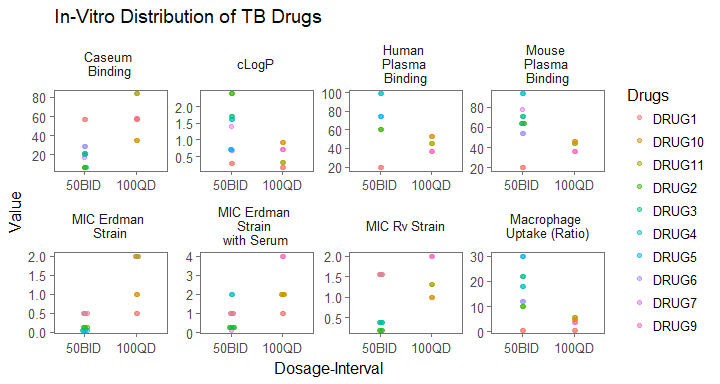
#invivo function  
  
efficacy\_summary <- paste0("https://raw.githubusercontent.com/KatieKey/input\_output\_shiny\_group/",  
 "master/CSV\_Files/efficacy\_summary.csv")  
  
#IMPORTANT NOTES: #created as reference for Shiny App creators   
  
#OPTIONS FOR VARIABLES STATEMENT  
#"RIM", "OCS","ICS","ULU","SLU","SLE","PLA"  
  
#OPTIONS FOR DRUGS STATEMENT  
#"DRUG1", "DRUG2", "DRUG3", "DRUG4", "DRUG5", "DRUG6",   
#"DRUG7", "DRUG8", "DRUG9", "DRUG10", DRUG11"   
  
#if default is NULL (i.e. no input for variables and drugs statements)   
#then it will plot ALL variables and drugs!   
  
  
invivo\_beeswarm\_function <- function(efficacy\_summary, variables = NULL, drugs = NULL) {  
   
 efficacy\_summary <- read\_csv(efficacy\_summary)   
   
 in\_vivo <- efficacy\_summary %>%  
 select(drug, dosage, dose\_int, PLA, ULU, RIM, OCS, ICS, SLU, SLE) %>%   
 rename(Drugs = "drug") %>%   
 unite(dosage\_interval, dosage:dose\_int, sep = "") #brings together dosage (50 & 100) with intervals (BID & QD)  
   
 in\_vivo\_SM <- in\_vivo %>%   
 gather(key = variable, value = value, -Drugs, -dosage\_interval) %>% #allows us to make small multiples by combining all the values into one column and the variable they belong to in another column  
 mutate(variable\_filtered = variable) %>% #explained in more detail in "Problems We Faced"  
 mutate(variable = factor(variable, levels = c("RIM", "OCS","ICS","ULU","SLU","SLE","PLA"),  
 labels = c("Rim (of lesion)","Outer Caseum","Inner Caseum","Uninvolved Lung",  
 "Standard Lung", "Standard Lesion", "Plasma"))) %>%   
 mutate(dosage\_interval = factor(dosage\_interval, levels = c("50BID","100QD"))) #prevents the numeric scaling of the x axis values, so it spaces it like two factors in the order given  
   
   
 if(!is.null(variables)) {  
 in\_vivo\_SM <- in\_vivo\_SM %>%   
 dplyr::filter(variable\_filtered %in% variables)  
 }  
   
 if(!is.null(drugs)) {  
 in\_vivo\_SM <- in\_vivo\_SM %>%  
 dplyr::filter(Drugs %in% drugs)  
 }  
   
 in\_vivo\_SMplot <- in\_vivo\_SM %>%   
 ggplot(aes(x = dosage\_interval, y = value, color = Drugs))+ #main structure of x & y axis  
 geom\_beeswarm(alpha = 0.5, size = 1.5)+ #incorporates beeswarm plot style  
 scale\_y\_log10()+ #log scale, used for the invivo group to cover wide range of values  
 labs(x = 'Dosage-Interval', y = 'Value')+  
 ggtitle('In-Vivo Distribution of TB Drugs')+  
 theme\_few()+  
 facet\_wrap(~ variable, ncol = 4) #creates small multiples by variable  
   
 return(in\_vivo\_SMplot)  
   
}  
  
invivo\_beeswarm\_function(efficacy\_summary) #example if statements left null, everything is plotted



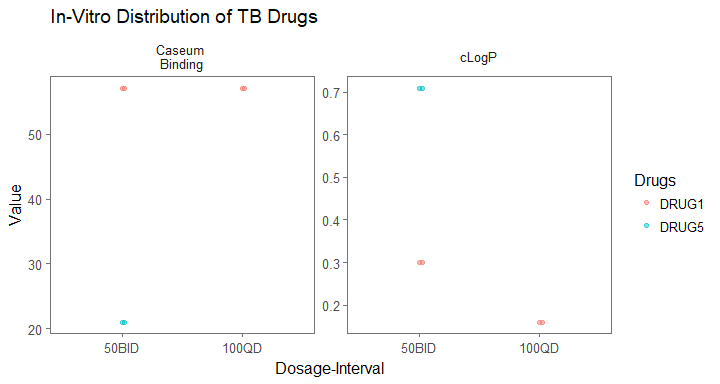
#example plot  
invivo\_beeswarm\_function(efficacy\_summary, variables = c("RIM", "OCS", "ICS", "PLA"),  
 drugs = c("DRUG1", "DRUG2","DRUG11", "DRUG9"))



#in vitro function   
  
efficacy\_summary <- paste0("https://raw.githubusercontent.com/KatieKey/input\_output\_shiny\_group/",  
 "master/CSV\_Files/efficacy\_summary.csv")  
  
#IMPORTANT NOTES:  
  
#OPTIONS FOR VARIABLES STATEMENT  
#"Caseum\_binding", "cLogP", "huPPB", "muPPB",  
#"MIC\_Erdman", "MICserumErd", "MIC\_Rv", "MacUptake"  
  
#OPTIONS FOR DRUGS STATEMENT  
#"DRUG1", "DRUG2", "DRUG3", "DRUG4", "DRUG5", "DRUG6",   
#"DRUG7", "DRUG8", "DRUG9", "DRUG10", DRUG11"   
  
#if default is NULL (i.e. no input for variabels and drugs statements)   
#then it will plot ALL variables and drugs!   
  
invitro\_beeswarm\_function <- function(efficacy\_summary, variables = NULL, drugs = NULL) {  
   
 efficacy\_summary <- read\_csv(efficacy\_summary)   
   
 in\_vitro <- efficacy\_summary %>%  
 select(drug, dosage, dose\_int, cLogP, huPPB, muPPB,   
 MIC\_Erdman, MICserumErd, MIC\_Rv, Caseum\_binding, MacUptake) %>%  
 rename(Drugs = "drug") %>%   
 unite(dosage\_interval, dosage:dose\_int, sep = "")  
   
 in\_vitro\_SM <- in\_vitro %>%   
 gather(key = variable, value = value, -Drugs, -dosage\_interval) %>%   
 mutate(variable\_filtered = variable) %>%   
 mutate(variable = factor(variable, levels = c("Caseum\_binding", "cLogP", "huPPB", "muPPB", "MIC\_Erdman",  
 "MICserumErd", "MIC\_Rv", "MacUptake"),  
 labels = c("Caseum \nBinding", "cLogP",   
 "Human \nPlasma \nBinding", "Mouse \nPlasma \nBinding",   
 "MIC Erdman \nStrain", "MIC Erdman \nStrain \nwith Serum", "MIC Rv Strain",  
 "Macrophage \nUptake (Ratio)"))) %>%   
 mutate(dosage\_interval = factor(dosage\_interval, levels = c("50BID", "100QD")))  
   
   
 if(!is.null(variables)) {  
 in\_vitro\_SM <- in\_vitro\_SM %>%   
 dplyr::filter(variable\_filtered %in% variables)  
 }  
   
 if(!is.null(drugs)) {  
 in\_vitro\_SM <- in\_vitro\_SM %>%  
 dplyr::filter(Drugs %in% drugs)  
 }  
   
   
 in\_vitro\_SMplot <- in\_vitro\_SM %>%   
 ggplot(aes(x = dosage\_interval, y = value, color = Drugs))+  
 geom\_beeswarm(alpha = 0.5, size = 1.5)+  
 labs(x = 'Dosage-Interval', y = 'Value')+  
 ggtitle('In-Vitro Distribution of TB Drugs')+  
 theme\_few()+  
 facet\_wrap(~ variable, ncol = 4, scale="free")  
   
 return(in\_vitro\_SMplot)  
   
}  
  
invitro\_beeswarm\_function(efficacy\_summary)



#example plot  
invitro\_beeswarm\_function(efficacy\_summary, variables = c("cLogP", "Caseum\_binding"),  
 drugs = c("DRUG1", "DRUG5"))



### Correlation Plots

key function stuff

### Hierarchial Clustering Function

We named the dendrogram function we created dendro\_cluster so that users would be able to easily remember what the function does which is create dendrograms based on clustering methods. The function is created to quickly create dendrograms using the independent variables to understand which variables are similar and clustered together. This function has two different categories or levels that the dendrogram can evaluate by\_test which will display a cluster dendrogram that shows the similarities among the test or measurements within the tuberculosis data. We further designed the function to show invitro variables as red text, and invivo variables as blue text. This could help the researchers understand which test types are more effective and could be helpful in determining which tests are more cost-effective. The function also allows the user to evaluate clustering patterns among by drug type with the code by\_test. The output of the dendro\_cluster function is either a plot with clustering by drug or by test whichever the user selects.

The inputs of the function are the function name dendro\_cluster and then df= data frame which was for the purpose of the project was the efficacy summary which is titled input\_data . The user also must enter the category choice of either by\_test or by\_drug. If the user does not choose a category in the function the function will not run and will tell the user that argument category is missing from the function call.

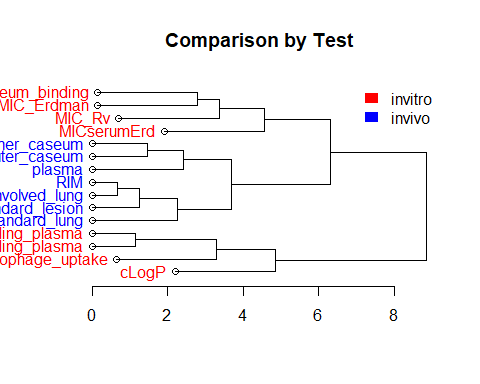
The function is set up as the first part being by\_test, the function has an else statement so that if the user is not interested in the by\_test plot the function goes to the ELSE statement and then identifies the by\_drug and follows the code for that selection. There is also a code within the by\_test that evaluates if the test is invitro or invivo and colors the text accordingly if the test is invitro the text will be red if else it would be blue, with the else being invivo

In the shiny app, there will be a tab for dendrogram clustering or hierarchical clustering and the user can then choose which category they want to display either by test or by drug. The output for the shiny app is the selected plot.

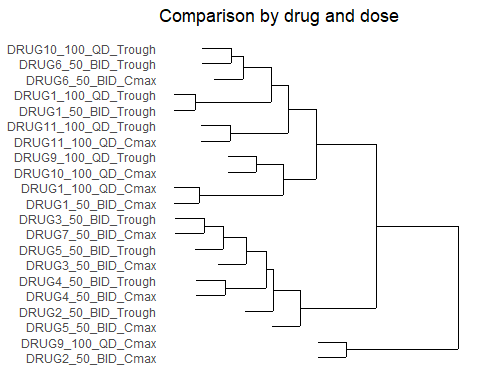
See comment code below to understand in each line what is occurring at each step.

library(readr)  
library(dplyr)  
library(ggraph)  
library(ggthemes)  
library(tibble)  
library(dendextend)  
library(ggplot2)  
library(ggdendro)  
  
#dendro\_cluster <- function(category, variables =c("measurement"))  
#dendro\_cluster, by\_test or by\_category   
input\_data<- read\_csv(paste0("https://raw.githubusercontent.com/KatieKey/input\_output\_shiny\_group/",  
 "master/CSV\_Files/efficacy\_summary.csv"))   
#dendro\_cluster(df= input\_data, category = "by\_test" OR "drug")  
df = input\_data  
category= "by\_test"  
dendro\_cluster <- function(df, category) {  
 #test\_type <- ifelse(colnames(input\_data) %in%   
 #c("cLogP", "huPPB","muPPB", "MIC\_Erdman",   
 #"MICserumErd", "MIC\_Rv","Caseum\_binding", "MacUptake"), 1, 2)   
 #makes vecotr #could assign color 1 is invivo   
 #how do we change the inputted data file? (raw data)  
 #1.5 choose what you want to do, then clean the data accordngly  
 if(category == "by\_test"){  
 by\_test <- df %>%   
 select(PLA:SLE,cLogP:MacUptake) %>%   
 mutate\_all(funs(scale(.))) %>%  
 rename(plasma = PLA,  
 uninvolved\_lung = ULU,  
 outer\_caseum = OCS,  
 inner\_caseum = ICS,  
 standard\_lung = SLU,  
 standard\_lesion = SLE,  
 macrophage\_uptake = MacUptake,  
 human\_binding\_plasma =huPPB,  
 mouse\_binding\_plasma = muPPB) %>%   
 as.matrix() %>%   
 t() %>%   
 dist() %>%   
 hclust() %>%   
 as.dendrogram(horiz = TRUE, hang = .3) #%>%   
 #new function within function to plot colors red = invivo; blue = in vitro   
 labelCol <- function(by\_test) {  
 if (is.leaf(by\_test)) {  
 ## fetch label  
 label <- attr(by\_test, "label")   
 ## set label color to red for A and B, to blue otherwise  
 attr(by\_test, "nodePar") <- list(lab.col=ifelse(label %in%   
 c("macrophage\_uptake","cLogP", "MIC\_Erdman", "MICserumErd",   
 "MIC\_Rv","Caseum\_binding",  
 "human\_binding\_plasma",  
 "mouse\_binding\_plasma"), "red", "blue")) # red is invitro  
 }  
 return(by\_test)  
 }  
 d <- dendrapply(as.dendrogram(by\_test), labelCol)  
 plot\_horiz.dendrogram(d, side = TRUE, main = "Comparison by Test")  
 #plot(d, horiz = TRUE, main = "by test", sub="color coded by test type", xlab = "")  
 cols <- c("red","blue")  
 legend("topright", legend = c("invitro","invivo"),  
 fill = cols, border = cols, bty = "n")  
 par(cex = 0.6, mar=c(9,11,10,10)) %>% #cex magnifies text; mar does axis  
 par(cex = 0.6)   
 #base plot oldpar<- par(mar xxxx, oma xxx) run at start  
 # par(oldpar) rest at begining and end of function ; side effect of function  
 # try ggplot or ggdend with colors   
 } else {  
 by\_drug <- df %>%   
 tidyr::unite(drugdetail, drug:level, sep = "\_") %>% #combine identifying data into one column,   
 mutate\_at(funs(scale(.) %>% as.vector),  
 .vars = c("PLA", "ULU", "RIM", "OCS", "ICS", "SLU", "SLE", "cLogP", "huPPB","muPPB",  
 "MIC\_Erdman", "MICserumErd", "MIC\_Rv", "Caseum\_binding", "MacUptake")) %>% #scales  
 select(drugdetail, PLA:MacUptake, -ELU, -ESP) %>% #remove efficacy   
 column\_to\_rownames (var = "drugdetail") %>% #make drugdetail leaf name! ignore warning  
 dist() %>%   
 hclust() %>% #can change method   
 as.dendrogram(horiz = TRUE, hang = .1)   
ggdendrogram(by\_drug, rotate = TRUE, theme\_dendro = TRUE) +  
 labs(title = "Comparison by drug and dose") +  
 theme(axis.title.x = element\_blank(),  
 axis.text.x=element\_blank(),  
 axis.ticks.x=element\_blank())  
   
 } }

dendro\_cluster(df = input\_data, category = "by\_test")



dendro\_cluster(df = input\_data, category = "by\_drug")



### Problems We Faced

#### Mouse small plots probs/Correlation Plots

#### Beeswarm

For the beeswarm plots and function, there were some problems that we faced that are important to note. It was not possible to use the data, as it was upon importing, to create small multiples. The gather() step in the function is an important step in particular, which modifies the data frame to a format that allows for the creation of small multiples.

Labeling in general became problematic because in many instances labels for the actual plots needed to be different or more specific than the variable names, which were minimized to an extent that would not be suitable for plot titles (i.e. PLA, SLU, huPPB, etc). This became a particular issue in the functions when markup language was used to format the spacing on the proper plot titles and was then required, as per the function code, to be typed as such in the variables statement input. The resolution to this problem was met by applying the minimized variable names to the variables statement and this is the reason why “variable\_filtered” is found as a reference in the function instead of “variable". It can be noted that in both functions, a new column “variable\_filtered” is created. By creating this new column, we can conserve the formal plot titles without having to reference them exactly in the function statement. It should be recognized that a set up like this might only be ideal for those who have had time to develop an understanding of both the shorthand (data frame) and longhand (plot title) versions of these independent variables.

The beeswarm plots, when plotted with all of the possible data, can appear pretty busy. This problem was touched upon in the description of the function as one of the reasons why the function was built as it was. These functions play a significant part in mitigating this “busy” problem by incorporating tools that help make the plots more specific and customizable by the researchers and other potential users.

One aspect that might be noticed about the beeswarm plots is that the classic beeswarm or violin shape is not prominent. This might be related to the quantity of data. Analyses using beeswarm across multiple studies over more time, that collect the same independent variable data, might demonstrate a more clear beeswarm shape.

#### ggdendrogram

The biggest challenge faced with the ggdendrogram package was manipulating the aesthetics of the plots to best represent our data. Our first challenge was playing with the axes, we wanted our dendrogram to be displayed horizontally not vertically to make the labels easier to read. This took a little bit of research, but for both output plots we were able to make them display horizontally by using a horiz = TRUE statement. The next challenge took the longest to solve, but helped out plot tell an interesting story. We wanted our by\_test plot to display which tests were invitro or invivo by color coding them. We understood how to group the data using basic R commands. But, we had a difficult time having these groupings display in our plots. We found that most of the color coding done in dendrograms is done by cluster and not by a specific element like test. In the end, we were able to color code the labels of the different nodes of whether they were invitro or invivo.

### Input Parameters/Potential Errors

While writing functions for each of our plots, we considered potential user error for our functions. One consideration, is dealing with data if there is missing data within the columns for each of our plots there may be limitations if there is missing data in an entire column. For example, the beeswarm functions will still run with existing "NA" values (they are excluded from the plot automatically) but if an entire column of data is missing, that would cause an error because the function would not be able to find an object it needs. It is also important to consider that our group used the format of the efficacy\_summary data frame as the basis for the functions created. If another data frame is referenced instead, it could cause a number of errors.

Another issue we thought about is the number of rows within the data sets for future use. We considered if there as just one row versus 500 rows, if our code would run the same as it is now. Another consideration is how the radio buttons might discord with the drug labels. For example, in the beeswarm functions, the functions should still work if the drugs are called different names (as long as the column name containing the drug information stays the same) but the radio buttons in the Shiny App would require an adjustment.

A potential error for the dendrogram clustering specifically, is if the user does not include a category the function will not work. However, this would not be a relevant issue in the shiny app, because they don't have to input a category they just need to chose which category they want.

### Future Considerations

Some of the next steps in general, which touch upon all of our specific analyses, that were considered included being able to customize and manipulate the labels. We might also want to add descriptions of the different tests or measurements using plotly so the user can understand what the different measurements or tests are indicating about the TB drug. We also thought it might be useful to incorporate a one-click export to PDF feature for the plots, in order to save specific, useful outputs. Another thought was that it would be beneficial to include control data, to further clarify the relationship between the independent variables and the drugs. In addition to controls, it would be interesting to consider how to incorporate Cmax and Trough data into these functions and the use of combined-drug data that might be found in some TB drug trials.

A future consideration for the beeswarm functions, in particular, would be including code that would plot blank small multiples for missing columns in a data set. Also, it would be helpful to code a fix in the radio buttons for potential drug-name variation.

### References

Andrie de Vries and Brian D. Ripley (2016). ggdendro: Create Dendrograms and Tree Diagrams Using 'ggplot2'. R package version 0.1-20. <https://CRAN.R-project.org/package=ggdendro>

Erik Clarke and Scott Sherrill-Mix (2017). ggbeeswarm: Categorical Scatter (Violin Point) Plots. R package version 0.6.0. <https://CRAN.R-project.org/package=ggbeeswarm>

Stack Overflow - Where Developers Learn, Share, & Build Careers. (2017). Stackoverflow.com. Retrieved 14 December 2017, from <https://stackoverflow.com/>

Thomas Lin Pedersen (2017). ggraph: An Implementation of Grammar of Graphics for Graphs and Networks. R package version 1.0.0. <https://CRAN.R-project.org/package=ggraph>