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, corrplot for the correlation plots, and ggdend for the dendrograms that represent hierarchical clustering. Another critical function, ggpolypath, was critical for creating visualizations of the biodistribution of each drug within the mice.

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.

Correlation plots were created under the corrplot package (Wei & Simko, 2017). The correlation plots highlighted which in-vivo and in-vitro measures for TB drugs were correlated to each other for all drugs within the dataset. This information is essential to the TB drug study, as it may highlight which assays are highly correlated, potentially reducing the overall number of assays which must be performed.

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.

The final function relied on ggplot2 and ggpolypath packages (Wickham, 2009; Sumner 2016). Combining the in-vivo measures given by the TB study with a dataframe containing coordinates (x,y) for a mouse and lesion visual model, ggplot2 enabled the creation of small multiples for drugs depicting the concentration of drug in each tissue measured. The ggpolypath function was necessary to create the various layers for each tissue region. For example, the mouse model would have appeared solid without the "hole" created by ggpolypath allowing the lung and lesion models to be seen within the mouse body. The result of both packages were summary visualizations of the biodistribution of drug for each drug in the dataset.

**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.

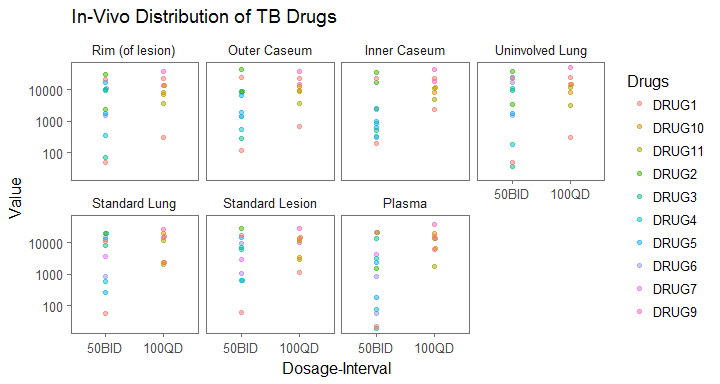
The following *if* statements in the function are what direct the function in that "unless otherwise" specified situation. When the variables statement or drugs statement is *not* null, the function tells it to filter the variables or drugs as requested in the input from the “variable\_filtered” or “Drugs” 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.

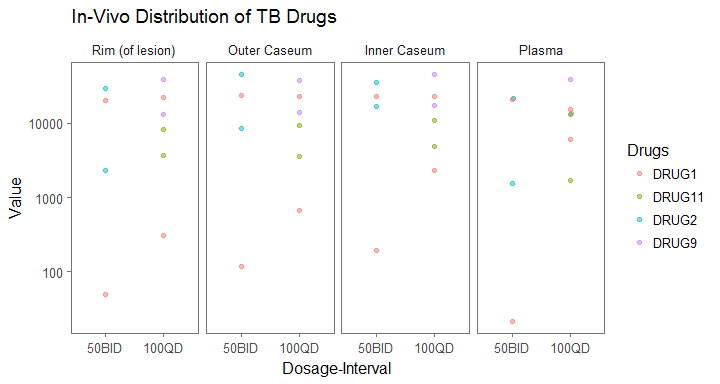
The beeswarm functions employ plotly and ggplotly() in particular to add an interactive element to the plots, however the code lines for this needed to be left out of this write up in order for Markdown to knit this document. This is a key component of the code that was given to the group working on 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)

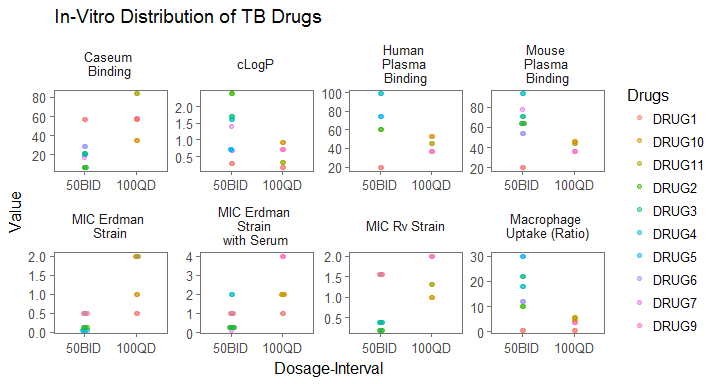
#invivo function  
  
efficacy\_summary <- paste0("https://raw.githubusercontent.com/KatieKey/input\_output\_shiny\_group/",  
 "master/CSV\_Files/efficacy\_summary.csv")  
  
#IMPORTANT NOTES: #created as reference for Shiny App creators   
  
#OPTIONS FOR VARIABLES STATEMENT  
#"RIM", "OCS","ICS","ULU","SLU","SLE","PLA"  
  
#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



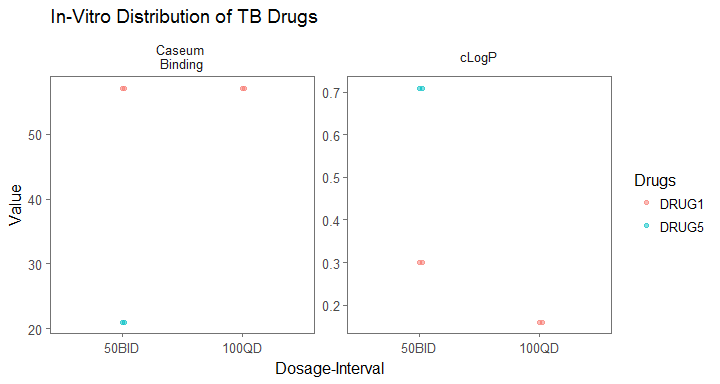
#example plot  
invivo\_beeswarm\_function(efficacy\_summary, variables = c("RIM", "OCS", "ICS", "PLA"),  
 drugs = c("DRUG1", "DRUG2","DRUG11", "DRUG9"))



#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"))

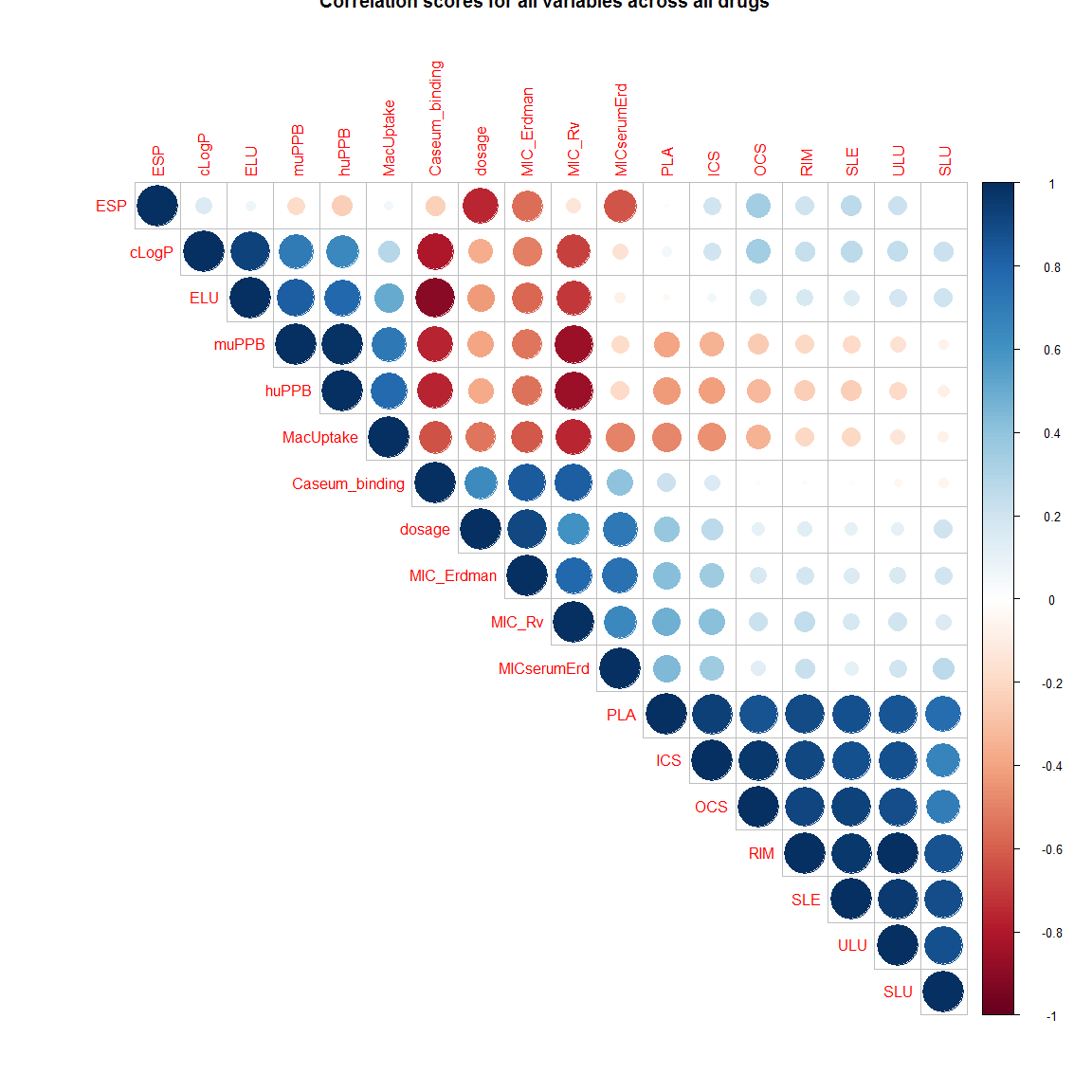


### Correlation Plots

The correlation plots were simple compared to the other functions written for the shiny app. This is mainly due to the fact that correlation plots usually explore all correlations within the dataset, as is the case here. All in-vivo and in-vitro data were crossed in order to give correlation scores for all markers across all drug groups. The function correlation\_function depended only on reading in the correct data file, as prepared by shiny app team, and removing non-numerical measures. After applying a correlation function cor, which was limited only to complete observations (complete.obs) due to the significant proportion of NA data, the correlation values were plotted using corrplot. Type upper and order AOE were selected in order to most clearly view the correlations without mirroring the information and with strong positive and negative correlations grouped within the display.

While this function yeilds a plot which may look data-heavy, filtering out low correlations to increase the value of the displayed data was not allowed. The researchers expressed a desire to see all correlations, so that they could select which assays to continue performing based on either low or high correlation. Therefore, the full correlation plot was the only one given.

library(dplyr)  
library(readr)  
library(corrplot)  
  
#Give address for raw data  
efficacy\_drugs <- (paste0("https://raw.githubusercontent.com/KatieKey/input\_output\_shiny\_group/", "master/CSV\_Files/efficacy\_summary.csv"))  
  
correlation\_function <- function(efficacy\_drugs) {  
efficacy\_drugs\_corr <- read\_csv(efficacy\_drugs)  
efficacy\_drugs\_corr <- efficacy\_drugs\_corr %>%   
 select(dosage:MacUptake, -dose\_int, -level)   
  
corrplot\_drugs <- cor(efficacy\_drugs\_corr, use = "complete.obs") %>%   
 corrplot(type = "upper", order = "AOE", title = "Correlation scores for all variables across all drugs")  
   
  
  
return(corrplot\_drugs)  
}  
  
correlation\_function(efficacy\_drugs)



## ESP cLogP ELU muPPB huPPB  
## ESP 1.000000000 0.15993853 0.06403427 -0.1899850 -0.24131664  
## cLogP 0.159938525 1.00000000 0.92364555 0.7094042 0.64340564  
## ELU 0.064034271 0.92364555 1.00000000 0.8201688 0.78924417  
## muPPB -0.189985011 0.70940423 0.82016879 1.0000000 0.98973544  
## huPPB -0.241316642 0.64340564 0.78924417 0.9897354 1.00000000  
## MacUptake 0.055026288 0.28613467 0.50546615 0.7181964 0.77247671  
## Caseum\_binding -0.234379539 -0.80705525 -0.90639003 -0.7602935 -0.76543894  
## dosage -0.758250177 -0.36293964 -0.42305277 -0.3933826 -0.37140381  
## MIC\_Erdman -0.554702611 -0.50734134 -0.57736203 -0.5347788 -0.54533960  
## MIC\_Rv -0.131950707 -0.68371315 -0.70504380 -0.8586332 -0.85991139  
## MICserumErd -0.623046637 -0.15846515 -0.07985250 -0.1859022 -0.19985965  
## PLA 0.021099006 0.05836969 -0.03511890 -0.3908102 -0.42820434  
## ICS 0.191950225 0.19395391 0.05677903 -0.3493280 -0.41347243  
## OCS 0.341946071 0.34554069 0.17043538 -0.2525520 -0.32170561  
## RIM 0.208998709 0.23918162 0.17437678 -0.2000267 -0.24050570  
## SLE 0.260108290 0.26694832 0.14684228 -0.1934217 -0.24256415  
## ULU 0.210323699 0.24115158 0.18957028 -0.1567129 -0.19465293  
## SLU -0.002519917 0.21916009 0.20743317 -0.0652205 -0.08150374  
## MacUptake Caseum\_binding dosage MIC\_Erdman MIC\_Rv  
## ESP 0.05502629 -0.23437954 -0.7582502 -0.5547026 -0.1319507  
## cLogP 0.28613467 -0.80705525 -0.3629396 -0.5073413 -0.6837131  
## ELU 0.50546615 -0.90639003 -0.4230528 -0.5773620 -0.7050438  
## muPPB 0.71819640 -0.76029351 -0.3933826 -0.5347788 -0.8586332  
## huPPB 0.77247671 -0.76543894 -0.3714038 -0.5453396 -0.8599114  
## MacUptake 1.00000000 -0.63578303 -0.5314000 -0.6100712 -0.7560399  
## Caseum\_binding -0.63578303 1.00000000 0.6340132 0.8395243 0.8271186  
## dosage -0.53139995 0.63401316 1.0000000 0.9023081 0.6077666  
## MIC\_Erdman -0.61007122 0.83952430 0.9023081 1.0000000 0.7880096  
## MIC\_Rv -0.75603988 0.82711859 0.6077666 0.7880096 1.0000000  
## MICserumErd -0.49353005 0.40736374 0.7138321 0.7490033 0.6413995  
## PLA -0.48978892 0.21704796 0.3835997 0.4217515 0.4810514  
## ICS -0.45778249 0.15471346 0.2679982 0.3596708 0.4132690  
## OCS -0.34018263 -0.01173408 0.1036268 0.1647908 0.2129522  
## RIM -0.20902050 -0.02125726 0.1312576 0.1837130 0.2413494  
## SLE -0.20853853 -0.01154383 0.1029815 0.1546977 0.1713114  
## ULU -0.13803296 -0.04015126 0.1024318 0.1612083 0.1986912  
## SLU -0.07635262 -0.05970619 0.2098297 0.1953112 0.1517020  
## MICserumErd PLA ICS OCS RIM  
## ESP -0.6230466 0.02109901 0.19195023 0.34194607 0.20899871  
## cLogP -0.1584652 0.05836969 0.19395391 0.34554069 0.23918162  
## ELU -0.0798525 -0.03511890 0.05677903 0.17043538 0.17437678  
## muPPB -0.1859022 -0.39081021 -0.34932798 -0.25255204 -0.20002666  
## huPPB -0.1998597 -0.42820434 -0.41347243 -0.32170561 -0.24050570  
## MacUptake -0.4935301 -0.48978892 -0.45778249 -0.34018263 -0.20902050  
## Caseum\_binding 0.4073637 0.21704796 0.15471346 -0.01173408 -0.02125726  
## dosage 0.7138321 0.38359968 0.26799819 0.10362681 0.13125755  
## MIC\_Erdman 0.7490033 0.42175153 0.35967085 0.16479084 0.18371301  
## MIC\_Rv 0.6413995 0.48105144 0.41326898 0.21295222 0.24134945  
## MICserumErd 1.0000000 0.44079960 0.35577140 0.12736883 0.22914534  
## PLA 0.4407996 1.00000000 0.93849825 0.86664221 0.89413812  
## ICS 0.3557714 0.93849825 1.00000000 0.96146261 0.90831751  
## OCS 0.1273688 0.86664221 0.96146261 1.00000000 0.91676841  
## RIM 0.2291453 0.89413812 0.90831751 0.91676841 1.00000000  
## SLE 0.1093091 0.87047180 0.87687677 0.92452825 0.96704399  
## ULU 0.1966693 0.85479954 0.87433100 0.88903752 0.99220828  
## SLU 0.2628076 0.76540967 0.66900503 0.69061968 0.86929713  
## SLE ULU SLU  
## ESP 0.26010829 0.21032370 -0.002519917  
## cLogP 0.26694832 0.24115158 0.219160086  
## ELU 0.14684228 0.18957028 0.207433165  
## muPPB -0.19342170 -0.15671290 -0.065220504  
## huPPB -0.24256415 -0.19465293 -0.081503737  
## MacUptake -0.20853853 -0.13803296 -0.076352616  
## Caseum\_binding -0.01154383 -0.04015126 -0.059706191  
## dosage 0.10298154 0.10243179 0.209829657  
## MIC\_Erdman 0.15469768 0.16120834 0.195311188  
## MIC\_Rv 0.17131140 0.19869118 0.151702002  
## MICserumErd 0.10930906 0.19666930 0.262807552  
## PLA 0.87047180 0.85479954 0.765409672  
## ICS 0.87687677 0.87433100 0.669005026  
## OCS 0.92452825 0.88903752 0.690619684  
## RIM 0.96704399 0.99220828 0.869297127  
## SLE 1.00000000 0.95977176 0.889331813  
## ULU 0.95977176 1.00000000 0.873117938  
## SLU 0.88933181 0.87311794 1.000000000

### 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.

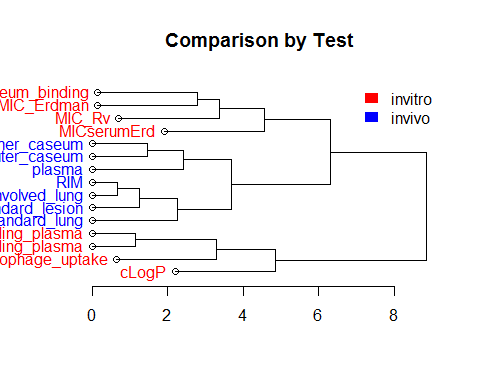
The function is set up as the first part being by\_test, the function has an else statement so that if the user is not interested in the by\_test plot the function goes to the ELSE statement and then identifies the by\_drug and follows the code for that selection. There is also a code within the by\_test 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.

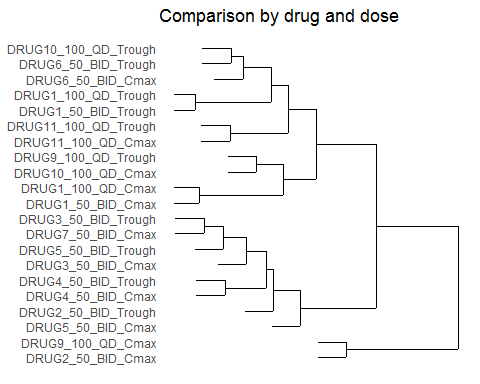
See comment code below to understand in each line what is occurring at each step.

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")



### Biodistribution modelling function

The biodistribution modelling for both mouse and lesion were combined into one display of small multiples, arranged by drug and dose. The function was named efficacy\_summary\_modelling\_function to clearly illustrate the data source as being summary data. The models for mouse and lesion were created within Excel. The mouse model was drawn point-by-point using a live excel scatterplot with a representative mouse in the background. The lesion was based on the formula for a circle, with a x,y point for each degree and iterated for 4 concentric circles representing the three regions of the lesion. The fibrous outer edge of the rim of the lesion was modelled by varying the radius for that particular circle. All tissue regions, representing plasma, standard lung, standard lesion, uninvolved lung, lesion rim, lesion outer caseum, and lesion inner caseum, were graphed for the model and filled according to reported maximum concentration as given by the efficacy\_summary file.

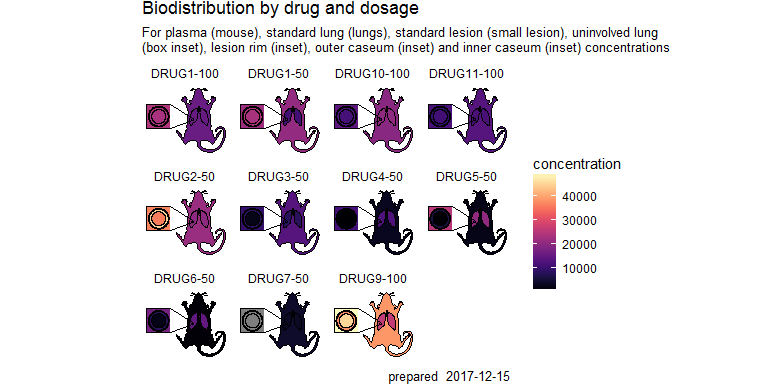
The function inputs were two dataframes, one set within the function (ModelCoord.xlsx) and the other called by the function (efficacy\_summary). The efficacy\_summary file was called by default, though another file with the same column names could be named within the function call (efficacy\_summary\_modelling\_function(efficacy\_summary = df\_address)). Missing data in any dataframe will simply show as a grey color within the mapping, as the concentrations are unavailable to use as fill values.

Two challenging elements were addressed within this function. The first was the joining of the model coordinates, some 3,000 points, to the fill concentrations for each region by drug. The end product simply joined the two dataframes by ELEMENT, or tissue label. This created a tibble of dimension (5 plus the number of drugs by unique dosage) x 3,000 columns and rows. This solution is noted within the join function, where the dataframes were merged. The second challenge for the function was the creation of holes within the layers of each plot region. The ggpolypath package addresses this issue, but required duplication of the coordinates for the model. Essentially, using geom\_polypath each point was plotted twice: once as the outline for the element, and once as the hole in the surrounding element (next layer out). For the mouse body, lungs, and lesion, this meant that the mouse body was drawn as an object with a lung-shaped hole, and then the lungs were drawn in the lung-shaped hole, with a lesion-shaped hole for the lesion to fill. Additional geoms were added to the ggplot2 object, mouse\_plot, in order to outline each region in black and to add lines for the lesion inset.

Additional functions from tidyr were also employed in the function to select the drug data and gather it into the proper format (long data instead of wide) prior to merging with the model coordinates. See the chunk notations for further details.

The overall impact of this function was a visual representation of biodistribution for each drug and dosage level. Trough and Cmax versions were both made available in the shiny app. This function did not address sample size or variations with maximum concentration for each drug, but could be added in the future by incorporation of summary statistics within plotly.

library(dplyr)  
library(readr)  
library(ggplot2)  
library(ggthemes)  
library(tidyr)  
library(readxl)  
library(ggpolypath)  
library(viridis)  
  
#Give the location for the source data file  
efficacy\_summary\_modelling <- (paste0("https://raw.githubusercontent.com/KatieKey/input\_output\_shiny\_group/", "master/CSV\_Files/efficacy\_summary.csv"))  
  
#Set up the beginning of the function, where in-vivo concentrations are read in from excel  
efficacy\_summary\_modelling\_function <- function(efficacy\_summary\_modelling) {  
  
efficacy\_summary\_modelling <- read\_csv(efficacy\_summary\_modelling)   
  
#Select the in-vivo elements of the efficacy summary, create a new column for drug and dosage together, filter for the maximum concetrations, and gather the in-vivo elements to prepare the dataframe for merging  
example\_data <- efficacy\_summary\_modelling %>%   
 select(drug, dosage, level, PLA, SLU, SLE, ULU, RIM, OCS, ICS) %>%   
 unite(drug\_dosing, drug, dosage, sep = "-") %>%   
 filter(level == "Cmax") %>%   
 gather(PLA:ICS, key = "ELEMENT", value = concentration) %>%   
 mutate(ELEMENT = as.character(ELEMENT))  
  
#Join the mouse/lesion coordinate dataframe to the efficacy summary modelling dataframe  
mouse <- read\_excel('../sample\_data/ModelCoord.xlsx') %>%   
 left\_join(example\_data, by = "ELEMENT")  
  
#Plot drug distribution, facetted by drug\_dosing  
mouse\_plot <- mouse %>%   
 ggplot(aes(mapping = TRUE, x = X, y = Y, group = HOLE, fill = concentration)) +  
 geom\_polypath(rule = "evenodd") +  
 geom\_path(colour = "black", size = .25) +  
 geom\_segment(x=-6, y=-2, xend=-28, yend=12) + #create inset lines for lesion zoom  
 geom\_segment(x=-6, y=-4, xend=-28, yend=-12) +  
 theme\_void() +  
 theme(legend.position = 'right') +  
 labs(title = "Biodistribution by drug and dosage",   
 subtitle = "For plasma (mouse), standard lung (lungs), standard lesion (small lesion), uninvolved lung \n(box inset), lesion rim (inset), outer caseum (inset) and inner caseum (inset) concentrations",   
 caption = paste("prepared ", Sys.Date() )) +   
 coord\_fixed() + #keep models from skew when scaled  
 scale\_fill\_viridis(option = "magma") +   
 facet\_wrap(~ drug\_dosing, ncol=4)  
  
return((mouse\_plot))  
}  
  
efficacy\_summary\_modelling\_function(efficacy\_summary\_modelling)



### Problems We Faced

#### Mouse small plots probs/Correlation Plots

The main problem within the correlation plots was formatting for the best visual communication of data. The correlation plots could be expanded by incorporation of unique correlation plots by drug, again in small multiple format. As most of the data available was in summary format, there was not enough unique information by drug to create correlation matrices by drug. In future, a cleaner full dataset with all variables by drug iteration, i.e. mouse, would make this possible. Additionally, the strength of correlations as displayed were undermined by the high prevalence of missing values, or NAs.

The modelling by small multiples of biodistribution was difficult mainly because it relied on two unrelated dataframes - one with data results and the other with the visual modelling coordinates. A third dataframe would likely be required to incorporate summary statistics for each concentration region by plotly. Calling these various datasets uniquely within a function is not easily feasible within R, hence the merge to a massive dataframe for the efficacy\_summary and ModelCoord dataframes. In future, nesting functions might address the problem. An alternate solution specifically for the current model would be to create a geom for the mouse/lesion model itself, with the input parameter being fill only, perhaps by region. Learning how to create custom geoms for this type of application is a newly developing functionality of R. Though possible, it lay outside the scope of this project.

#### 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>

Hadley Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2009.

Michael D. Sumner (2016). ggpolypath: Polygons with Holes for the Grammar of Graphics. R package version 0.1.0. <https://CRAN.R-project.org/package=ggpolypath>

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

Taiyun Wei and Viliam Simko (2017). R package "corrplot": Visualization of a Correlation Matrix (Version 0.84). Available from <https://github.com/taiyun/corrplot>

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>