Final Report

Group 3

December 4, 2017

The report should be no longer than 5 pages (using default font size, line spacing, etc. for RMarkdown documents; if you include figures or tables that show output from your functions, these do not need to count towards that page limit). Be sure to show examples of using your functions in your Word report.

# Introduction/Background/Objective

## Group 3 Objective: Association between dependent and independent variables

* Create summaries and visualizations of how the dependent variable is associated with different independent variables. Here, we will try to discover if there are characteristics of the drugs that are associated with effectiveness against TB. This group will need to come up with ways (and code) to analyze that in the data. This might include generalized linear models, scatterplots, and possibly other supervised learning methods.

# Rationale:

* Explain what you were hoping to achieve in writing the functions / app framework that your group created.

# Idea development:

* 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:
* Write a brief title for the function (< 8 words) and a brief description (3–4 sentences).
* Define all parameters. For example, if you have a df parameter, explain that this is the dataframe that will be modeled / visualized. If it must have certain column with certain names, specify that.
* Define what the functions will output (e.g., “A ggplot object showing . . . ” or “The model output object from running a . . . ”).
* If you have a reference (e.g., for a model you’re fitting in the function), you can include that
* If you want an extra challenge, try to use the Roxygen2 syntax in writing these descriptions. Otherwise, you can write them in code comments.

## Regression Tree Function:

The regression tree function is used as a way to determine which independent variables are most important for predicting a particular dependent or outcome variable. Using our data, it helps us find which independent variables have the largest influence on our outcomes of lung efficacy (ELU) or spleen efficacy (ESP). Below is an example of code and output for the regression tree function to help explain how the function works and how to interpret the output.

We begin with a tidy dataset of all observations (in this case, 20) called efficacy\_summary. We then choose an independent variable of interest; here we have chosen ELU. To run the function, we input ELU along with all of the independent variables from the dataset into the rpart() function, as seen in the code below. In the function, we also specify the dataset from which to pull the variables, as well as control parameters that determine how the function runs.

#### Regression Tree code:

rpart(ELU ~ drug + dosage + level +   
 plasma + `Uninvolved lung` + `Rim (of Lesion)` +   
 `Outer Caseum` + `Inner Caseum` +   
 `Standard Lung` + `Standard Lesion` + cLogP +   
 `Human Plasma Binding` +   
 `Mouse Plasma Binding` + `MIC Erdman Strain` +   
 `MIC Erdman Strain with Serum` +   
 `MIC rv strain` + `Caseum binding` +   
 `Macrophage Uptake (Ratio)`

After trial and error, the regression tree function was completed and ready to use with the following parameters:

* dep\_var: This is where the user specifies the dependent variable of interest. Options include "ELU" (lung efficacy) or "ESP" (spleen efficacy).
* min\_split: This is a numeric input indicating the minimum number of observations for a split to be attempted in the regression tree.
* min\_bucket: This is another numeric input indicating the minimum number of observations in a terminal node of the regression tree.
* data = efficacy\_summary: This is the default dataset that the function uses. The function will only run with a dataset of this name that has these specific variable names.

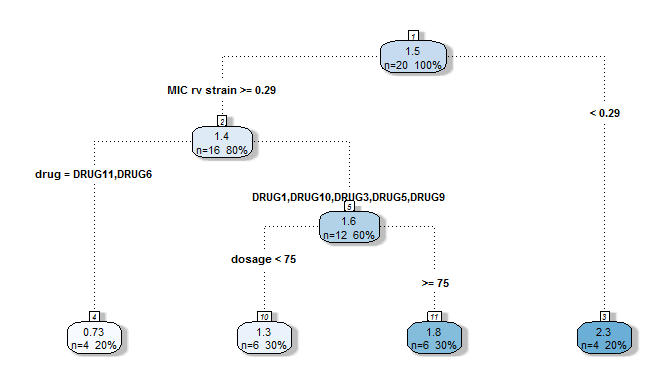
The following code gives an example of using the regression\_tree function with the parameters that were explained above:

#### Regression Tree function input example:

regression\_tree(dep\_var = "ELU", min\_split = 8,   
 min\_bucket = 4)

Now that the code and function have been explained, we will run examples of the function and explain what the output is telling us. Consider the following plot given by the code printed above:

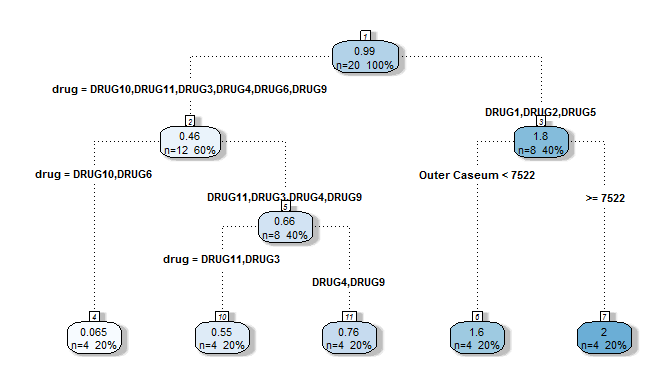
#### Regression Tree ELU Example:



Each "node"" in the regression tree is numbered at it's top. Within each node is a count of how many observations (for node 1, n=20) and the percent of total observations (for node 1, 100%) that are being explained by that node. The top number within each node is indicating the mean of the outcome variable for the observations in that node. In this case, ELU has a mean of 1.5 for node one.

Starting from node one, the first split is made so that it leads to the greatest possible reduction in residual sum of squares (RSS). At node one, the split was made at a level of 0.29 for variable MIC rv strain. This is saying that this variable at this level lead to the greatest reduction in RSS for the outcome of ELU. Continuing on, node three is a terminal node because it only has 4 observations. This happened because in our function parameters we set min\_bucket = 4. Similarly, node two splits again because it has 16 observations. This happened because in our funciton parameters we set min\_split = 8. This process continues until either the min\_split or the min\_bucket parameters are fulfilled for each node. The final regression tree above is showing us a step-by-step selection of the most important variables in determining ELU. Below is a similar regression tree for the outcome of ESP:

#### Regression Tree ESP Example:



# Room for errors:

* So far, we have focused on getting working prototypes, without making sure they’re error-proof and robust to a user doing something non-standard. Identify three things a user could do that could make your functions “break” (i.e., either return an error message or return something other than what you hope they will).

# Next steps:

* Include a section where you describe what you think are interesting next steps, i.e., what you would pursue next if you were continuing work on this project. Lay out explicitly a few ideas (2–3) that you think would be helpful. Be sure, when relevant, to describe how feedback from the project researchers helped in forming these ideas for next steps.