Final Presentation

Group 3

December 4, 2017

Group 3 Objective: Association between dependent and independent variables

- The objectives were to create summaries and visualizations of how the dependent variable is associated with the different independent variables. Our goal was to develop different models to analyze these associations in the data. Our work included:
- generalized linear models
- scatterplots
- supervised learning methods, including RandomForest, Regression Trees, and Lasso.

Rationale:

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

Here, we will try to discover if there are characteristics of the drugs that are associated with effectiveness against TB. The dependent variables tested included drug concentrations in the lung tissue and spleen. The independent variables included several *in vivo* (mouse model) and *in vitro* tests that were performed by the TB research group.

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.

We explored various models to see if there were concordant associations detected between independent and dependent variables. It was challenging to make strong associations because the dataset provided at this stage was small and did not include many drugs, however, we did discover similar patterns with certain *in vitro* or *in vivo* tests and ELU. We also explored changes in these associations by different drug doseage, and interestingly drug dose appears to be an effect modifier on some of the relationships, so that will be interest to follow up on as more data is provided.

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.

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 3 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):
- Very low or high number of observations in the input dataset
- One thing that we noticed was in the individual drug data, the "NA", or missing data for spleen_efficacy had a space. This type of variation in how missing data is recorded could cause problems for the functions.
- If drug names or codes change, this could create potential problems
- If new independent variables or measurements are added to the dataframe

Room for errors (continued):

- The dataset provided included two dose frequency combinations, 50 BID and 100 QD. If these dose and frequency combinations change it could cause problems with some of the functions.
- The current functions intake a "tidy" form of the dataframe, `efficacy summary`, with column names including: "drug", "dosage", "dose_int", "level", "PLA", "ULU", "RIM", "OCS", "ICS", "SLU", "SLE", "ELU", "ESP", "cLogP", "huPPB", "muPPB", "MIC_Erdman", "MICserumErd", "MIC_Rv", "Caseum_binding", and "MacUptake".

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.

Functions, Get Ready!!!:

Visualize univariate variables in a scatterplot function, by drug dose

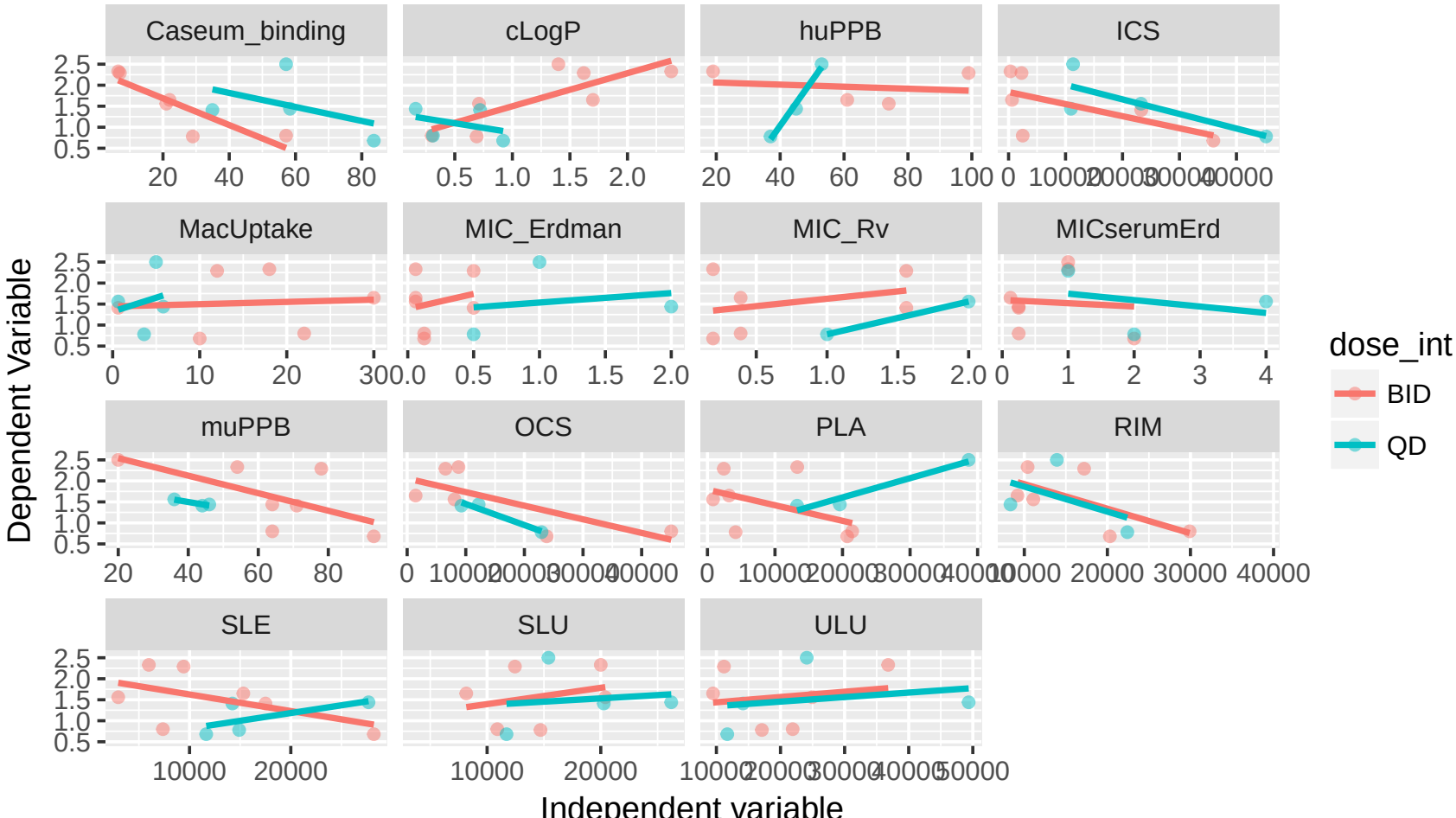
- *Goal*: To visualize univariate measurement variables correlated with the dependent variables, a faceted scatterplot colored by dose was created. Inputs include:
- `peak_trough`, which evaluates the peak or trough, respectively, of drug following injection.
- The `dep_var` options are either ELU or ESP (lung levels and spleen levels, respectively.

Scatterplot Input Parameters:

- function name: `univar_plot`
- default dataframe: `efficacy_summary`
- `dep_var` options: “ELU” (lung efficacy) or “ESP” (spleen efficacy)
- `peak_trough` options: “Cmax”" or “Trough”

Scatterplot- Visualize independent and dependent variable correlations

```
#Sample code for function, univar_plot
scatterplot <- univar_plot(peak_trough = "Cmax", dep_var =
scatterplot
```



Scatterplot Interpretation

- There appear to be stronger linear relationships with variables, including Caseum_binding ICS and RIM, compared to others. Some relationships are positively correlated (QD, huPPB), whereas others are negatively correlated (Caseum_binding, ICS)
- It appears that the relationship between independent and dependent variables are dependent on dose for variables cLogP, huPPB, PLA, and SLE. The others are not dependent on dose.
- Dose may be an important factor to consider when evaluating the relationship between independent and dependent variables.

Fitting Linear Models Function

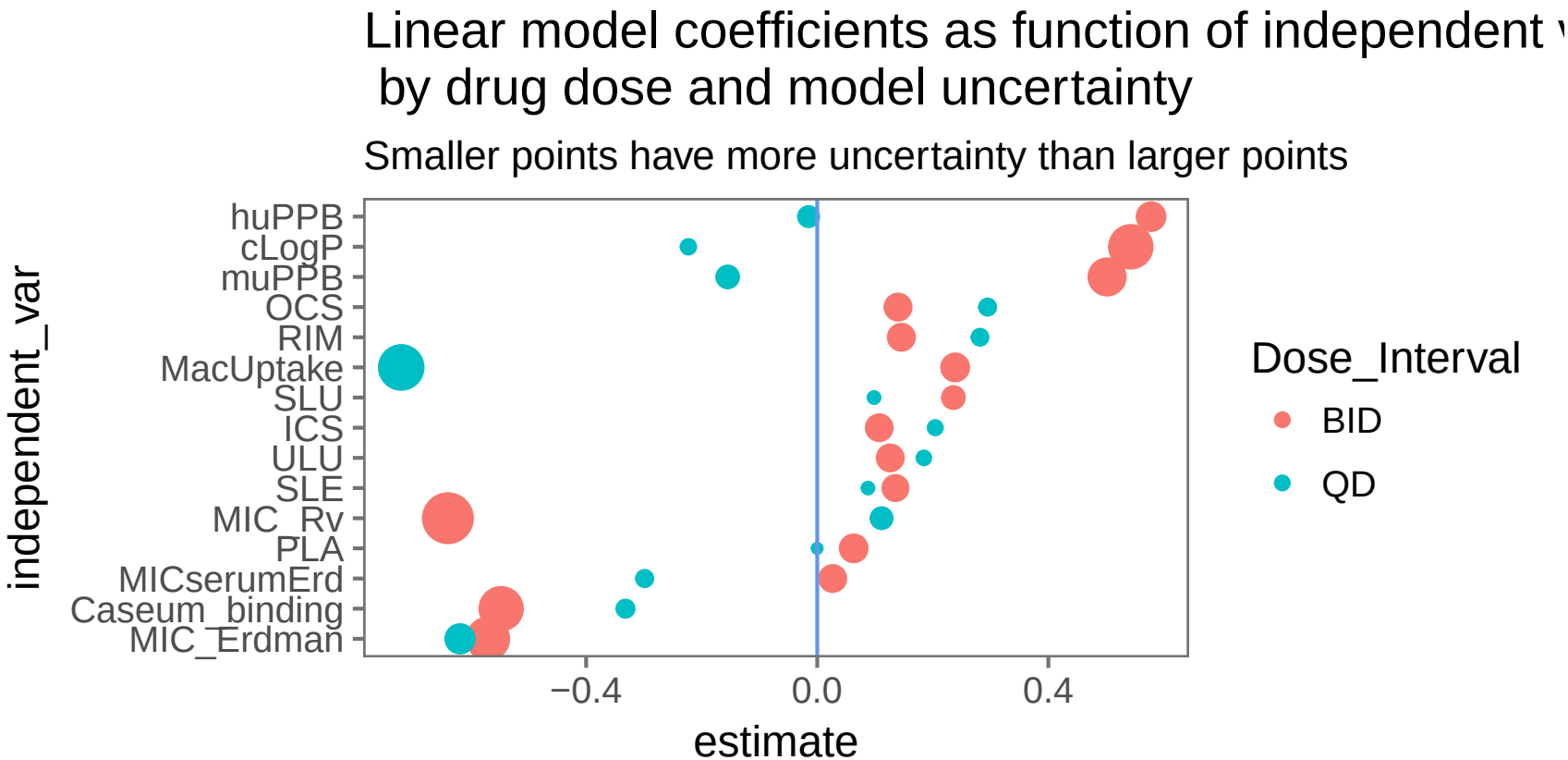
- *Goal*: to fit a linear model regressing dependent on independent variables to assess the relationships between them.
- Assumptions are the observations are idependent, and identically distributed, the error has a normal distribution, and there is a linear relationship between independent and dependent variables
- Inputs to this function include peak_trough, or the dep_var (options are ELU or ESP).
- Output from this function provides a plot of the normalized model coefficients by each independent variable
- *note*: The units of scale for each independent variable were normalized so that we could compare across coeficients.

Linear Model Input Parameters:

- function name: `linear_model`
- default dataframe: `efficacy_summary`
- `dep_var` options: “ELU” (lung efficacy) or “ESP” (spleen efficacy)
- `peak_trough` options: “Cmax”" or “Trough”

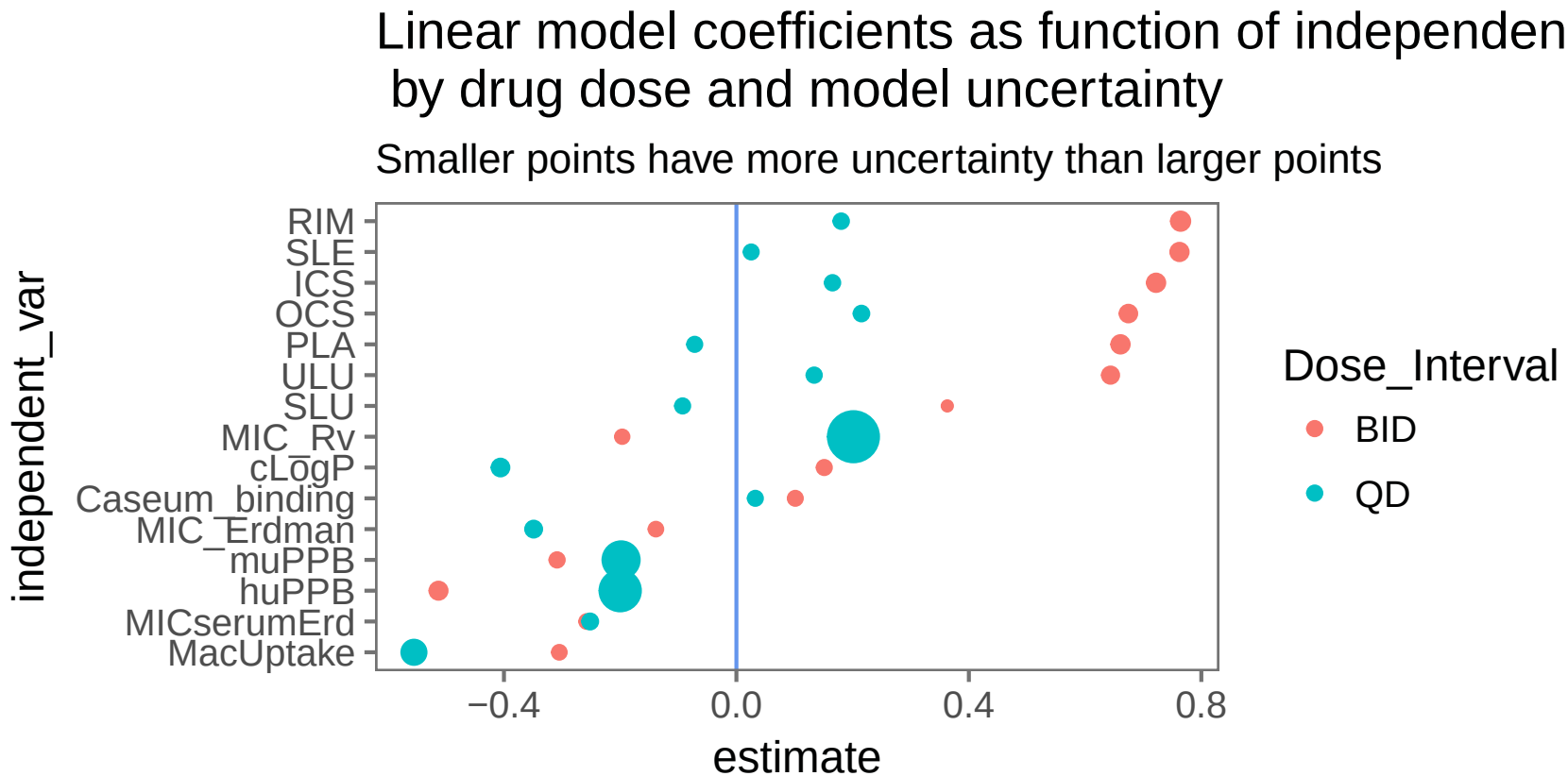
Linear Model- Visualize independent variable coefficients - ELU

```
#Sample code for function, linear_model (Cmax and ELU)
linearmodel_ELU <- linear_model(peak_trough = "Cmax", dep_v
linearmodel_ELU
```



Linear Model- Visualize independent variable coefficients - ESP

```
#Sample code for function, linear_model (Cmax and ESP)
linearmodel_ESP <- linear_model(peak_trough = "Cmax", dep_v
linearmodel_ESP
```



Linear Model Interpretation

- The coefficient plot generated by this function shows each independent variable on the y axis and the respective model coefficient on the x axis, for either ELU or ESP.
- If the coefficient is negative, for example, as it is with *MacUptake* in the ELU model linear regression model, an interpretation would be for every unit of change in the MacUptake, the ELU will decrease by 0.5 Units. Therefore, MacUptake has a negative relationship with ELU, decreasing the ELU. The diameter of the point represents the level of certainty of the coeficient in this model. This may change as more data is collected for each drug.

Visualizing individual drug efficacy by mouse

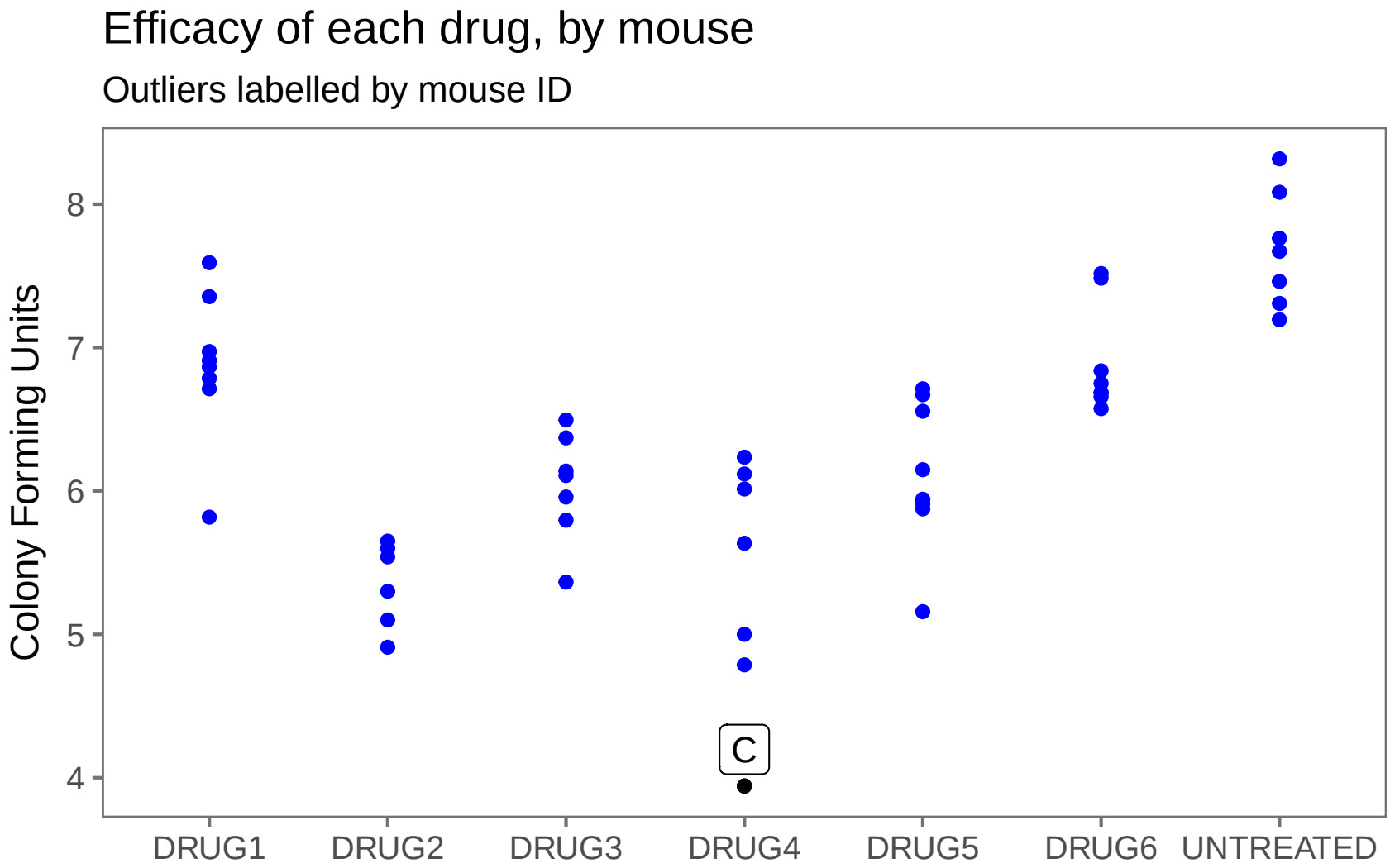
- *Goal*: using individual drug data, visualize efficacy in the lung or spleen and detect mouse data outliers in the dataset.
- Inputs to this function include lung_efficacy or spleen_efficacy
- Output from this function provides a plot of the efficacy by individual mouse.

Individual drug plot input parameters:

- function name: `drug_mouse`
- default dataframe: `cleaned_2_combined`
- `dep_var` options: “lung_efficacy” or “spleen_efficacy

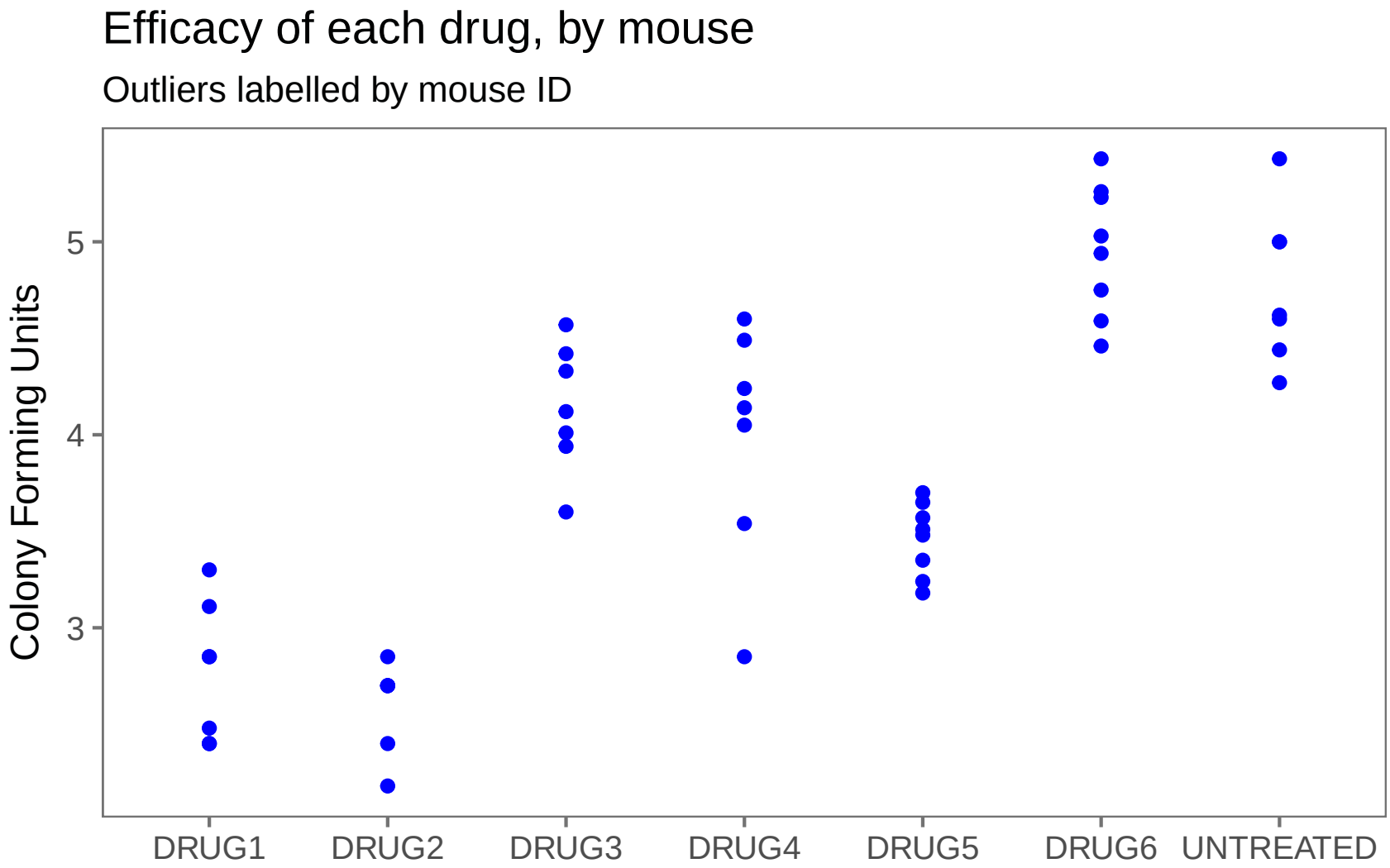
Individual drug plot

```
#Sample code for function, drug_mouse
drugmouse_lung <- drug_mouse(dep_var = "lung_efficacy")
drugmouse_lung
```



Individual drug plot

```
#Sample code for function, drug_mouse
drugmouse_spleen <- drug_mouse(dep_var = "spleen_efficacy")
drugmouse_spleen
```



Individual drug plot interpretation

-There is variation in the efficacy of each drug on lung efficacy in the mouse model. This plot shows the individual mouse variation in efficacy for each drug. Outliers (calculated using the 1.5 X IQR rule), are automatically labelled with the mouse ID. In this dataset for `lung_efficacy`, mouse C for drug 4 was an outlier. There were no outliers present in the `spleen_efficacy` dataset. You can see that untreated mice were had the least efficacy to decrease bacteria in the lung or spleen, respectively.

Regression Tree Function

- This function will input all of the efficacy summary variables as independent variables and let you choose the dependent variable (lung efficacy or spleen efficacy).

```
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)`,
      data = function_data,
      control = rpart.control(cp = -1, minsplit = min_split
                              minbucket = min_bucket))
```

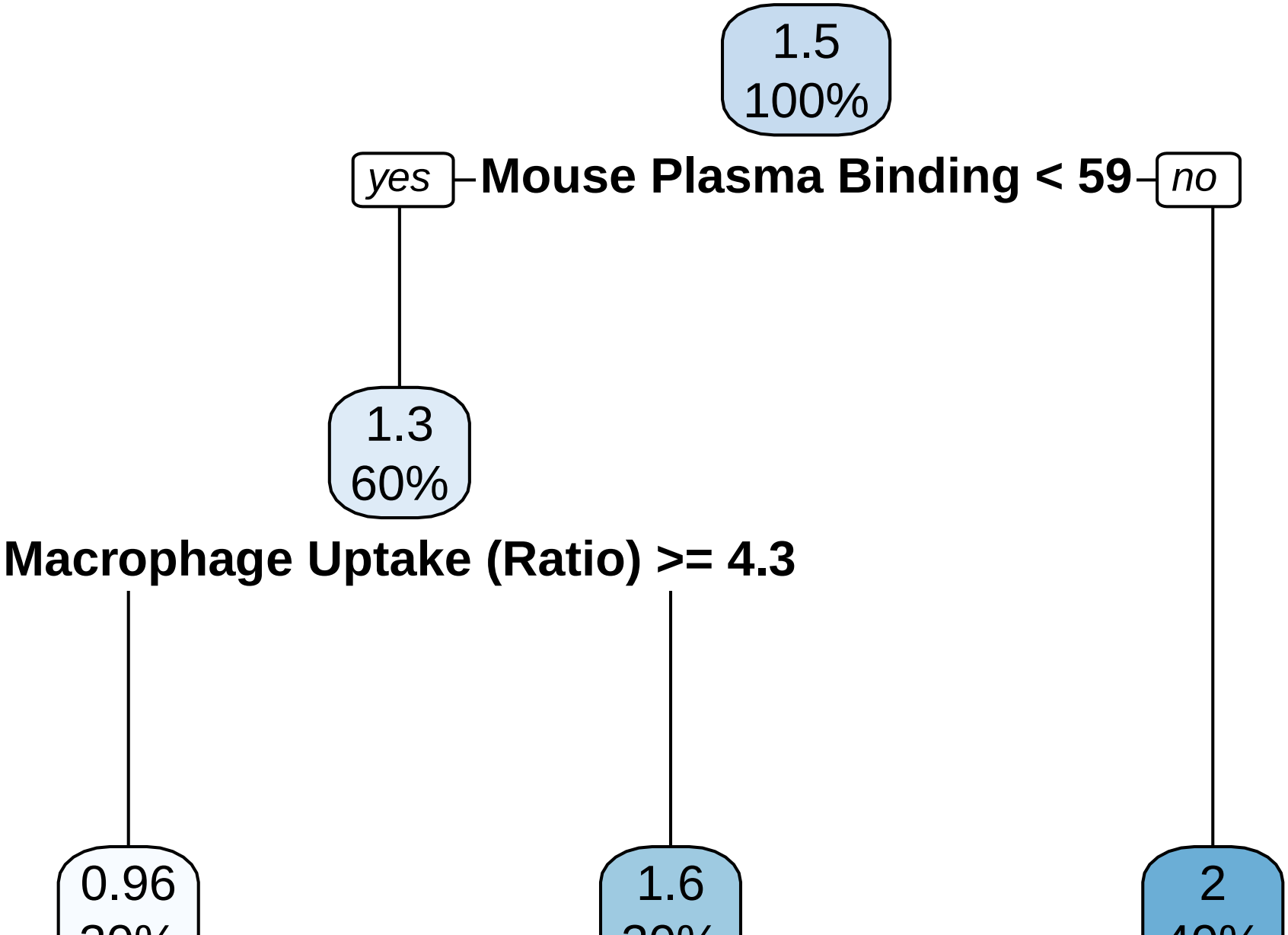
Regression Tree Function

- Parameters:
- dep_var options: “ELU” (lung efficacy) or “ESP” (spleen efficacy)
- min_split: numeric input indicating minimum # observations for a split to be attempted
- min_bucket: numeric input indicating minimum # observations in a terminal node

```
regression_tree(dep_var = "ELU", min_split = 8,
                min_bucket = 6)
```

Regression Tree Function

- Output:



Interpretation

- The number at the top of each node is indicating the mean of the outcome variable for the observations in that node (mean of 1.5 for node 1). Below each node is indicating what each split was based on. Splits are chosen based on a complexity parameter. Starting from node 1, the first split is made so that it leads to the greatest possible reduction in RSS. Node 3 is a terminal node because it only has 4 observations, which was the minimum number of observations a node can have to be considered (set in our function parameters). Given the 16 observations in node 2, another split is made that again gives the greatest possible reduction in RSS. This process continues until either the min_split or the min_bucket parameters are fulfilled for each node from the preset function parameters.

LASSO Function

The function for LASSO required the following inputs:

- Dependant varaible
- Dose
- Dataframe, though the dataframe default is efficacy_summary

The output in the end is coefficients with smaller coefficients being resitricted to zero.

Testing LASSO function:

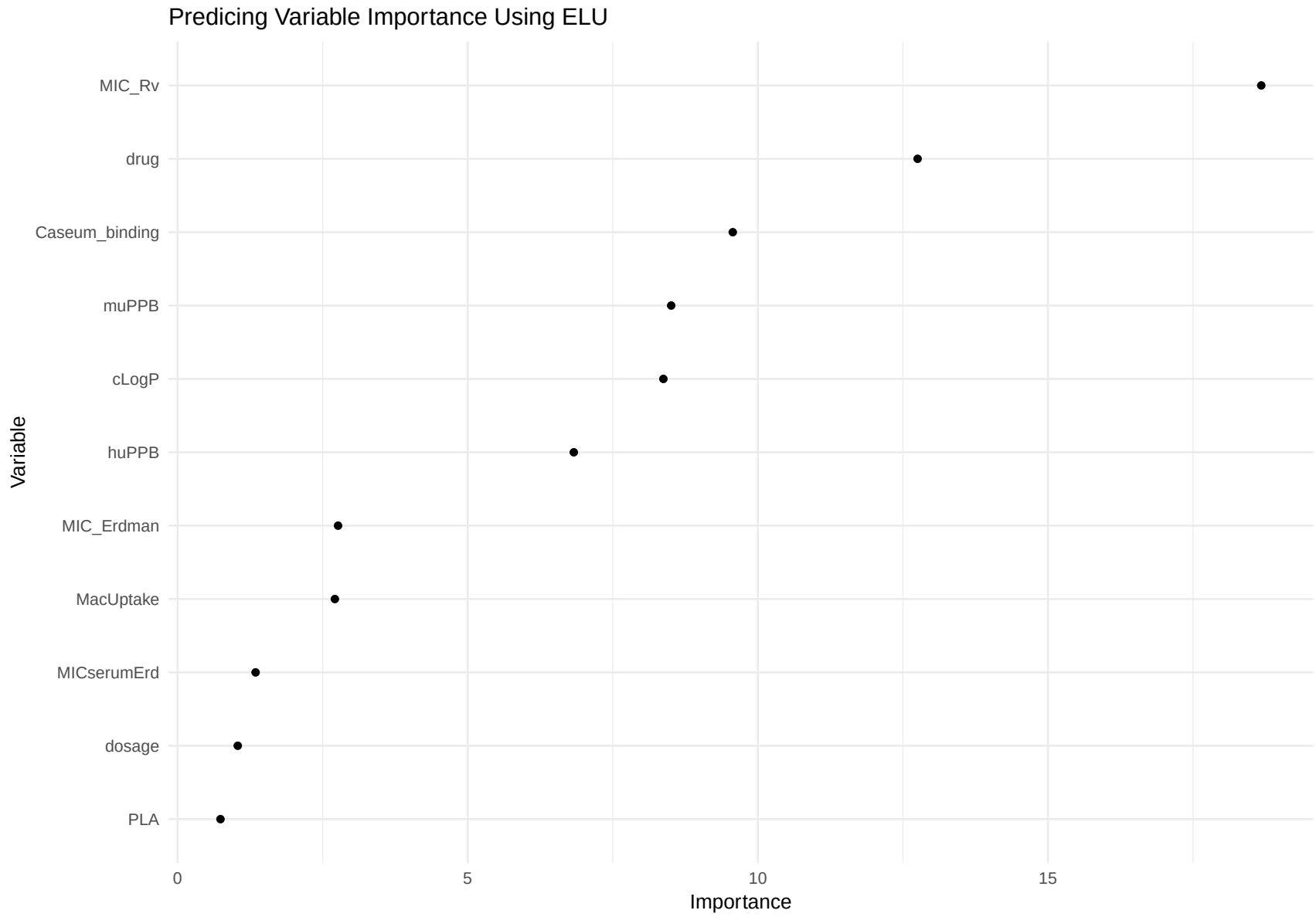
Interpretation:

- This function outputs raw coefficents and an intercept on an penalized maximum likelihood model. Models dropped from the model are represented with zeros. It computes a LASSO penalty for small coefficents and drops them, resulting in only the coef with the most leverage remaining.

RandomForest Function

- The user specifies which dependent variable they would like to use (either ELU or ESP). The user can also specify a dataset they would like to use, if one is not provided then a default dataset is utilized. The function var_importance takes the input and outputs a graph displaying the which variables are the best predictors of the input (either ELU or ESP).

Testing Random Forest



Interpretation

- This function utilizes the function randomForest to predict which variables are the most important predictors of the associated outcome. This model works by randomly creating small data nodes that are split using the best predictor from a subset of predictors randomly chosen at each node. In order to determine variable importance, the algorithm looks at how much the predictive error increases as all variables remain unchanged while one is permuted. The resulting output shows the % increase in the mean standard error for each variable considered individually. The higher the number the more important the variable for model building.