Analysis Master

Kate, Maggie, Drew, Ethan

November 27, 2017

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.

Visualize univariate variables in a scatterplot function, by drug dose (Kate)

```
univar_plot <- function(peak_trough, dep_var, data = efficacy_summary) {

  function_data <- data %>%
  filter(level == peak_trough) %>%
  gather(key = independent_var, value = indep_measure, -drug, -dosage, -dose_int, -level, -ELU, -ESP, na
  select(drug, dosage, dose_int, level, dep_var, indep_measure, independent_var)

  if(dep_var=="ELU") {vect <- function_data$ELU}
  if(dep_var=="ESP") {vect <- function_data$ESP}

  scatter_plot <- function_data %>%
  ggplot(aes(x = indep_measure, y = vect, color = dose_int)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = lm, se = FALSE) +
  labs(x = "Independent variable", y = "Dependent Variable") +
  facet_wrap(~independent_var, scales = "free_x")

scatter_plot
}

test <- univar_plot(peak_trough = "Cmax", dep_var = "ELU")
```

The user specifies wether the dependent variable they are interested in is ELU or ESP. They also specify the level (Cmax or Trough). The scatterplot defines the independent variable on the x axis, and dependent variable (ELU or ESP). The faceted plot is colored by dose (which is also tied to the dose interval)

Fitting Linear Models Function (Kate)

```
#Define the linear model function

linear_model <- function(peak_trough, dep_var, data = efficacy_summary) {

function_data <- data %>%
  filter(level == peak_trough) %>%
  gather(key = independent_var, value = indep_measure, -drug, -dosage, -dose_int, -level, -ELU, -ESP, na
  select(drug, dosage, dose_int, level, dep_var, indep_measure, independent_var)

  if(dep_var=="ELU") {function_data$vect <- function_data$ELU}
  if(dep_var=="ESP") {function_data$vect <- function_data$ESP}

  model_function <- function(data) {
    model_results <- lm(vect ~ scale(indep_measure), data = data)
  }

estimate_results <- function_data %>%
  group_by(independent_var, dose_int) %>%
  nest() %>%
  mutate(mod_results = purrr::map(data, model_function)) %>%
  mutate(mod_coefs = purrr::map(mod_results, broom::tidy)) %>%
  select(independent_var, dose_int, mod_results, mod_coefs) %>%
  unnest(mod_coefs) %>%
  filter(term == "scale(indep_measure)")

coef_plot <- estimate_results %>%
  mutate(independent_var = forcats::fct_reorder(independent_var, estimate, fun = max)) %>%
  rename(Dose_Interval = dose_int) %>%
  ggplot(aes(x = estimate, y = independent_var, color = Dose_Interval)) +
  geom_point(aes(size = 1 / std.error)) +
  scale_size_continuous(guide = FALSE) +
  theme_few() +
  ggtitle(label = "Linear model coefficients as function of independent variables, \n by drug dose and r
  geom_vline(xintercept = 0, color = "cornflower blue")

coef_plot
}

test_2 <- linear_model(peak_trough = "Cmax", dep_var = "ELU")
```

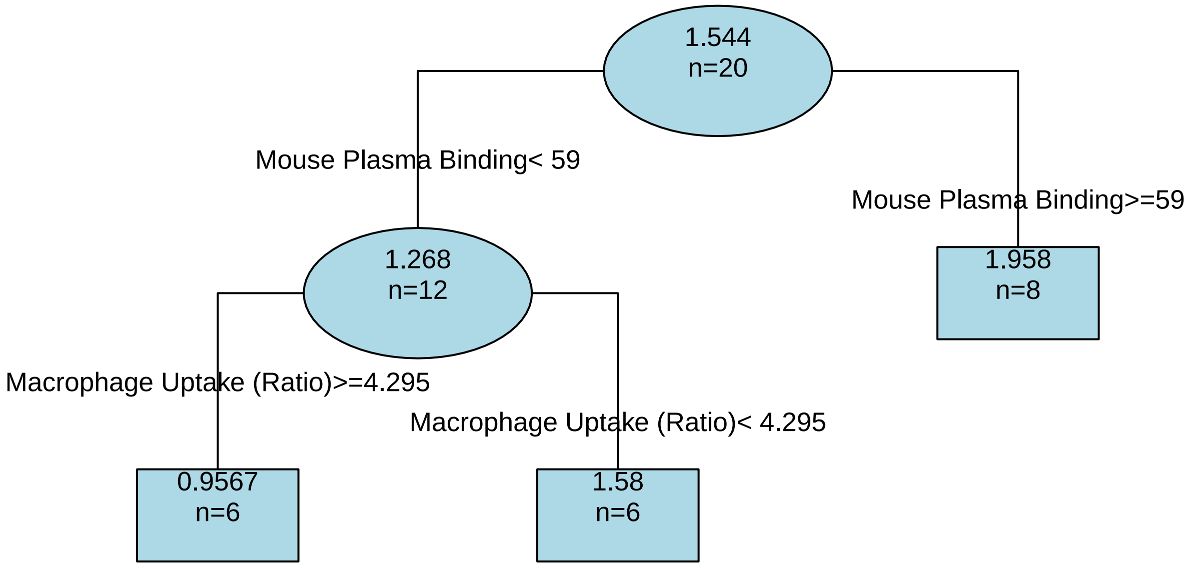
Regression Tree Function (Ethan):

```
regression_tree <- function(dep_var = "ELU", min_split = 8, min_bucket = 6,
                            data = efficacy_summary) {

  if (dep_var == "ELU") {

  function_data <- data %>%
    filter(!is.na(ELU)) %>%
    rename(plasma = PLA, `Uninvolved lung` = ULU,
         `Rim (of Lesion)` = RIM, `Outer Caseum` = OCS, `Inner Caseum` = ICS,
         `Standard Lung` = SLU, `Standard Lesion` = SLE, `Human Plasma Binding` = huPPB,
         `Mouse Plasma Binding` = muPPB, `MIC Erdman Strain` = MIC_Erdman,
         `MIC Erdman Strain with Serum` = MICserumErd, `MIC rv strain` = MIC_Rv,
         `Caseum binding` = Caseum_binding, `Macrophage Uptake (Ratio)` = MacUptake)

  tree <- rpart(ELU ~  drug + dosage + level +
```

```r
                  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))
  }

  if (dep_var == "ESP") {

  function_data <- data %>%
    filter(!is.na(ESP)) %>%
    rename(plasma = PLA, `Uninvolved lung` = ULU,
         `Rim (of Lesion)` = RIM, `Outer Caseum` = OCS, `Inner Caseum` = ICS,
         `Standard Lung` = SLU, `Standard Lesion` = SLE, `Human Plasma Binding` = huPPB,
         `Mouse Plasma Binding` = muPPB, `MIC Erdman Strain` = MIC_Erdman,
         `MIC Erdman Strain with Serum` = MICserumErd, `MIC rv strain` = MIC_Rv,
         `Caseum binding` = Caseum_binding, `Macrophage Uptake (Ratio)` = MacUptake)

  tree <- rpart(ESP ~  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))
  }

  par(mfrow = c(1, 1), xpd = NA, mar = c(5,5,5,5))
  plot(tree, uniform=TRUE)
  text(tree, use.n=TRUE, all=TRUE, cex=.8, minlength = 0, fancy = TRUE,
     bg = "lightblue", fwidth = .8, fheight = .8)
}
```

Testing regression_tree function:

```r
#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)
```



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 (Maggie):

```r
LASSO_model <- function(dep_var, dose, df = efficacy_summary) {
  dataz <- na.omit(df) %>%
  select_if(is.numeric) %>%
  filter(dosage == dose)

response <- dataz %>%
  select(dep_var)

predictors <- dataz %>%
  select(c("PLA", "ULU", "RIM", "OCS", "ICS", "SLU", "SLE", "cLogP", "huPPB", "muPPB", "MIC_Erdman", 'MI

y <- as.numeric(unlist(response))
x <- as.matrix(predictors)

fit = glmnet(x, y)

coeff <- coef(fit,s=0.1)
coeff <- as.data.frame(as.matrix(coeff)) %>%
rownames_to_column()
colnames(coeff) <- c("predictor", "coeff")

coeff <- coeff %>%
  filter(coeff > 0)
coeff %>%
  kable()
}
```

Testing LASSO function:

```
#LASSO_model(dep_var, dose, df)
#dep_var options: "ELU" (lung efficacy) or "ESP" (spleen efficacy)
#dose = dosage
#df = dataframe, default is efficacy_summary

LASSO_model("ELU", 100)
```

predictor	coeff
(Intercept)	1.1545503
cLogP	0.7190423
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 coeff and drops them, resulting in only the coef with the most leverage remaining.

RandomForest Function (Drew)

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).

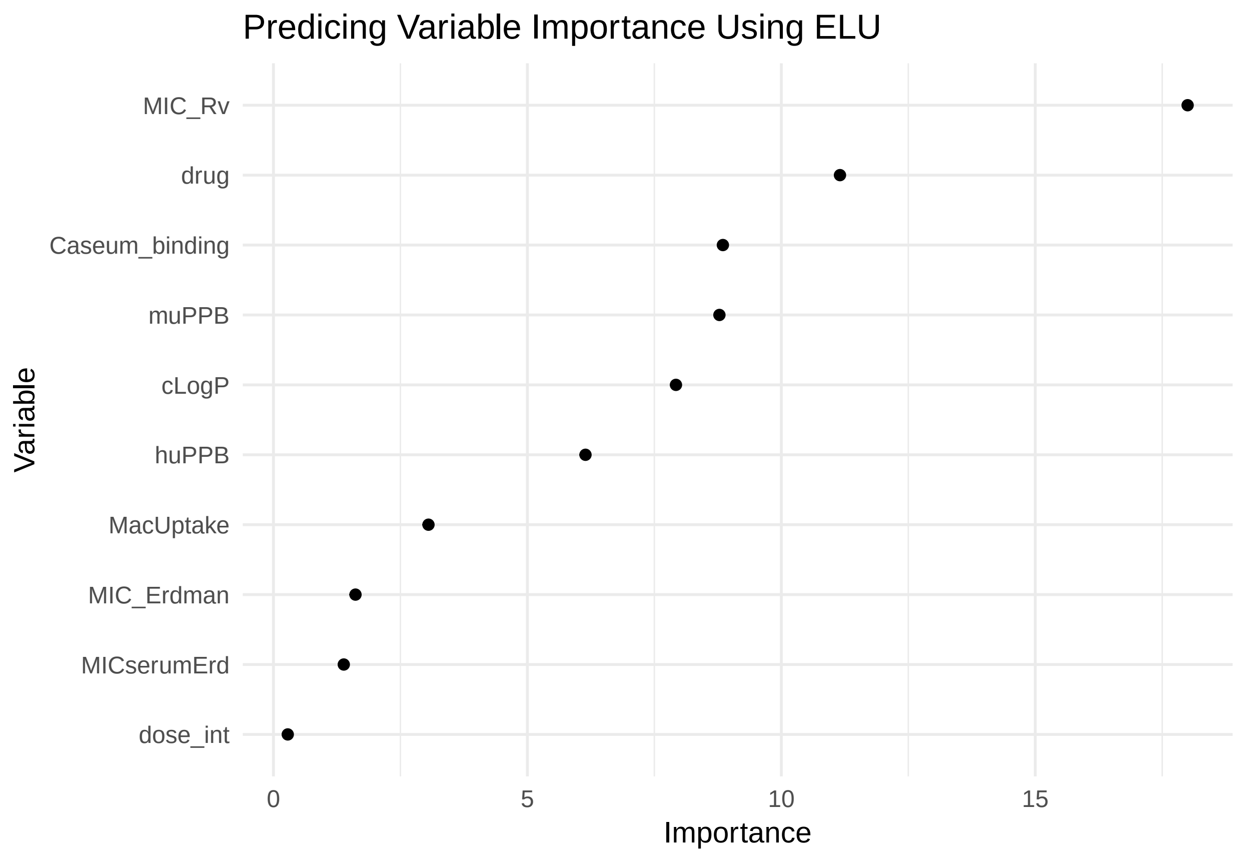
```
best_variables <- function(dep_var, df = efficacy_summary){


  title <- paste0 ("Predicing Variable Importance Using ", as.character(dep_var))

  if(dep_var == "ELU"){
  dataset <- df %>%
    select(-ESP) %>%
    mutate(huPPB = as.numeric(huPPB),
         muPPB = as.numeric(muPPB),
         dosage = as.factor(dosage),
         dose_int = as.factor(dose_int),
         level = as.factor(level),
         drug = as.factor(drug))

efficacy.rf <- randomForest( ELU~ ., data =dataset,
              na.action = na.roughfix,
                        ntree= 1000,
                        importance = TRUE)
  }
```

```r
  if (dep_var == "ESP"){
    dataset <- df %>%
      select(-ELU) %>%
    mutate(huPPB = as.numeric(huPPB),
         muPPB = as.numeric(muPPB),
         dosage = as.factor(dosage),
         dose_int = as.factor(dose_int),
         level = as.factor(level),
         drug = as.factor(drug))

efficacy.rf <- randomForest( ESP ~ ., data =dataset,
              na.action = na.roughfix,
                        ntree= 1000,
                        importance = TRUE)
  }

graph <-importance(efficacy.rf, type = 1) %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  rename(variable = rowname,
         mse = `%IncMSE`)



graph %>%
  filter(mse > 0) %>%
  ggplot()+
  geom_point(aes(x = mse, y = reorder(variable, mse)))+
  theme_minimal()+
  ggtitle(title)+
  labs(y = "Variable",
       x = "Importance")
}
```

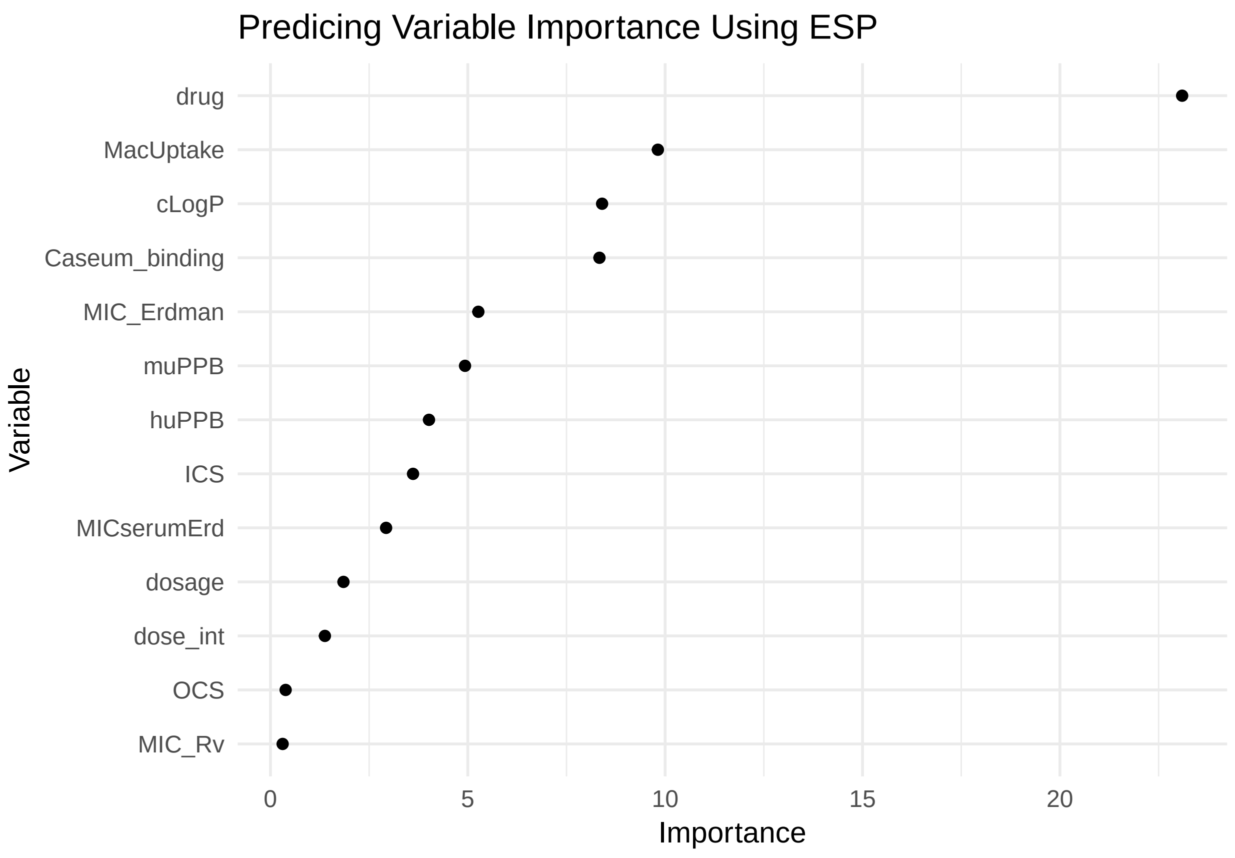
Interpretation

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.

```r
best_variables("ELU")
```



```r
best_variables("ESP")
```


Predicing Variable Importance Using ESP
Variable
drug
MacUptake
cLogP
Caseum_binding
MIC_Erdman
muPPB
huPPB
ICS
MICserumErd
dosage
dose_int
OCS
MIC_Rv
0
5
10
15
20
Importance