Cancer Survival Genes Random Forest Model

Nathan Harounian

Before we begin, let's load some necessary packages.

```
library(tidyverse)
library(data.table)
library(skimr)
```

Now, let's load both of our clinical and mrna gene data.

```
# change DATA_DIR to wherever the data is located
DATA_DIR <- "../../Stat_154/Discussion"

# fread() is a lot faster than read_csv(), so use it,
# especially for large datasets

mrna_orig <- fread(file.path(DATA_DIR, "data_mrna_agilent_microarray.txt")) %>%
    as_tibble()
clinical_orig <- fread(file.path(DATA_DIR, "brca_metabric_clinical_data.tsv")) %>%
    as_tibble()
```

1) Threshold "Overall Survival (Months)" & Binarizing the Data

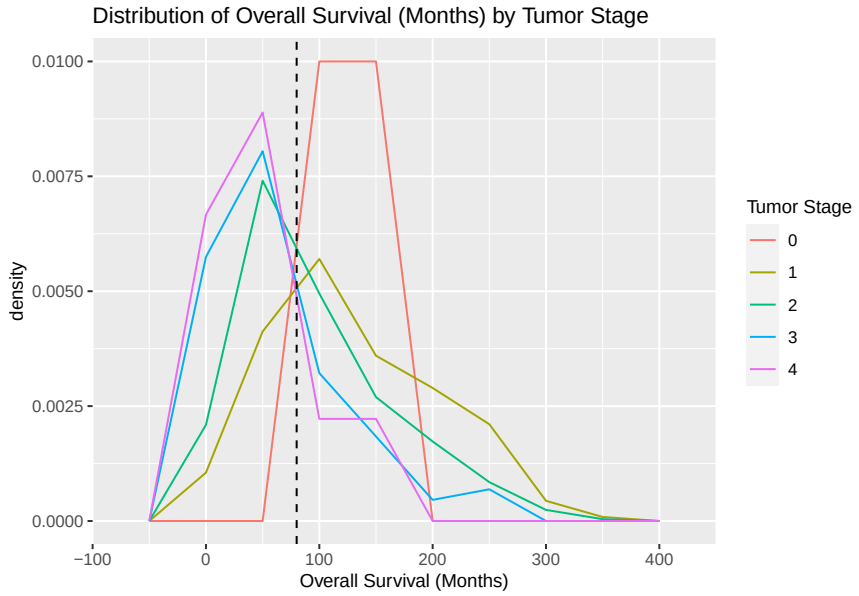
```
LOW_QUANTILE <- 0.2
HIGH_QUANTILE <- 0.8
survival_thresholds <- clinical_orig %>%
    filter(`Overall Survival Status` == "1:DECEASED") %>%
    pull(`Overall Survival (Months)`) %>%
    quantile(c(LOW_QUANTILE, HIGH_QUANTILE), na.rm = TRUE)
survival_thresholds
```

Creating quantiles for survival thresholds

```
##       20%       80%
##  36.60667 163.40000
```

EDA: How will we classify "High" and "Low" survival Now, let's investigate how `Tumor Stage` and `Overall Survival (Months)` are related. First, let's group the data by both variables and aggregate the size of each unique group. Then, let's make a histogram of the relationship.

```r
# grouping data and aggregating counts
clinical_counts <- clinical_orig %>%
    group_by(`Overall Survival (Months)`, `Tumor Stage`) %>%
    summarise(count = n())

# EDA plot
clinical_orig %>%
    drop_na(`Tumor Stage`) %>%
    mutate(`Tumor Stage` = as.factor(`Tumor Stage`)) %>%
    filter(`Overall Survival Status` == "1:DECEASED") %>%
    ggplot(aes(`Overall Survival (Months)`, colour = `Tumor Stage`)) +
    geom_freqpoly(aes(y = after_stat(density)), binwidth = 50) +
    geom_vline(xintercept = 80, linetype = "dashed") + labs(title = "Distribution of Overall Survival (
    theme(title = element_text(hjust = 0.5, size = 10))
```



We define low survival as 50 months or less and high survival as 100 months or more. If we look at the graph above, we see that the distribution peaks (median value) for tumor stages $\{2, 3, 4\}$ are located at 50 months, while the distribution peaks (median value) for tumor stages $\{0, 1\}$ are at 100 months.

```r
clinical_orig <- clinical_orig %>%
    mutate(y = case_when((`Overall Survival (Months)` <= 50) &
        (`Overall Survival Status` == "1:DECEASED") ~ "Low",
        `Overall Survival (Months)` >= 100 ~ "High", TRUE ~ NA_character_  # to be thrown out later
))
table(clinical_orig$y)
```

```
##
## High  Low
## 1144  358
```

2) Data Sculpting & Cleaning

Because `mrna_orig` has gene expressions as the a column `Hugo_Symbol` and patients as columns, we will compute the tranpose of the matrix, and make unique `Hugo_Symbol` values.

```
X = mrna_orig %>%
    dplyr::select(-Hugo_Symbol, -Entrez_Gene_Id) %>%
    t() %>%
    as.data.frame()


# Check duplicated genes
sum(duplicated(mrna_orig$Hugo_Symbol))
```

```
## [1] 194
```

```
# rename duplicated genes
colnames(X) <- make.names(mrna_orig$Hugo_Symbol, unique = TRUE)
X["Patient ID"] = colnames(mrna_orig)[-c(1, 2)]
```

Additionally, we decided to only investigate patients who got diagnosed at ages 50 - 74. This decision was partially due to the USPTF (US Preventative Task Force), who recommend women in this age group to be screened for breast cancer via mammography every other year. The other reason we decided to focus in on an age rage is for interpretability; it makes more sense to compare gene and clinical data to predict 'high' or 'low' survival for those close in age rather than say, a teenager and a senior citizen. (source: https://www.uspreventiveservicestaskforce.org/uspstf/recommendation/breast-cancer-screening)

Additionally, I want to recognize that `clinical_orig` and the transposed `mrna_orig` do not have the same number of rows. Upon further investigation, my group concluded that the difference is due to additional patients from the `clinicial_orig` data; however, everyone in `mrna_orig` is also in `clinical_orig`, so we decided to not use the additional data from `clinical_orig`.

```
# make each row corresponds to the same patient
data_full <- left_join(x = X, y = clinical_orig, by = "Patient ID")

data_full = dplyr::filter(data_full, (`Age at Diagnosis` >= 50) &
    (`Age at Diagnosis` <= 74))
```

Here, we confirm what we discussed above.

```
# checking no duplicated values in Patient ID columns of
# clinical_orig and X
sum(duplicated(clinical_orig$`Patient ID`))
```

```
## [1] 0
```

```
sum(duplicated(X$`Patient ID`))
```

```
## [1] 0
```

```r
# what is the difference in length between X and
# clinical_orig (how many people missing)
length(clinical_orig$`Patient ID`) - length(X$`Patient ID`)
```

```
## [1] 605
```

```r
# making sure that everyone in X is also in clinical_orig
# and that this is the sole reason for the difference in
# lengths between the two vectors
sum(X$`Patient ID` %in% clinical_orig$`Patient ID`) == length(X$`Patient ID`)
```

```
## [1] TRUE
```

let's filter out middle portion based on our survival thresholds

```r
keep_samples <- !is.na(data_full$y)

data_full <- data_full[keep_samples, ]

# check NAs
sum(is.na(data_full))
```

```
## [1] 499
```

```r
names(which(colSums(is.na(data_full)) > 0))
```

```
##  [1] "SLC25A19"                      "IDO1"
##  [3] "CSNK2A1"                       "BAMBI"
##  [5] "MRPL24"                        "FAM71A"
##  [7] "Type of Breast Surgery"        "Cellularity"
##  [9] "ER status measured by IHC"     "Neoplasm Histologic Grade"
## [11] "Tumor Other Histologic Subtype" "Primary Tumor Laterality"
## [13] "Mutation Count"                "Relapse Free Status"
## [15] "3-Gene classifier subtype"     "Tumor Size"
## [17] "Tumor Stage"                   "Patient's Vital Status"
```

```r
# check that y_full now has no NA's
sum(is.na(data_full$y))
```

```
## [1] 0
```

```r
row_na_idx <- apply(data_full, 1, function(x) any(is.na(x)))
print(sprintf("# rows with NAs: %s", sum(row_na_idx)))
```

```
## [1] "# rows with NAs: 382"
```

```r
gene_na_idx <- apply(data_full, 2, function(x) any(is.na(x)))
print(sprintf("# columns with NAs: %s", sum(gene_na_idx)))
```

```
## [1] "# columns with NAs: 18"
```

```
print(sprintf("# rows before dealing with NA's in X: %s", nrow(data_full)))
```

```
## [1] "# rows before dealing with NA's in X: 874"
```

```
print(sprintf("# cols before dealing with NA's in X: %s", ncol(data_full)))
```

```
## [1] "# cols before dealing with NA's in X: 24408"
```

The fact that we have 499 total NA values in `data_full` and the above code says we only have 382 rows with NA values in `data_full` tells us that we have multiple NA values in some of our rows. We can therefore get rid of these.

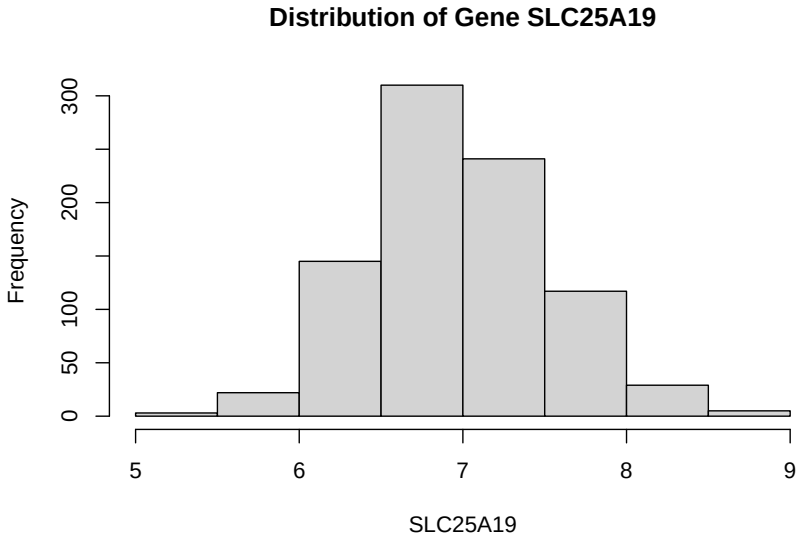
Data Cleaning Now, let's investigate the columns with missing values in `data_full`, so that we can make informed decisions on how to best deal with the NA's.

The following features have at least 1 data point with a missing value:

1. SLC25A19

- Symmetric
- 2 missing values
- impute mean for the 2 missing values

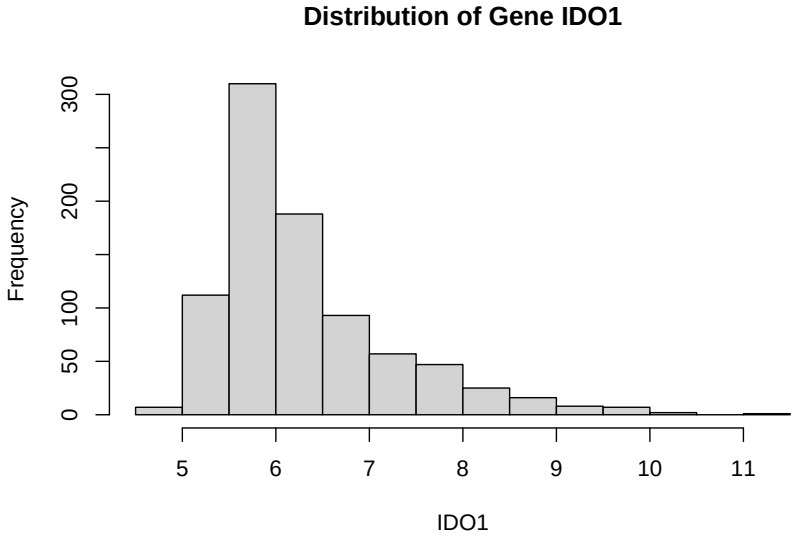
```
hist((data_full$SLC25A19)[!is.na(head(data_full$SLC25A19))],
    main = "Distribution of Gene SLC25A19", xlab = "SLC25A19")
```



2. IDO1

- Right skew
- 1 missing value
- impute median

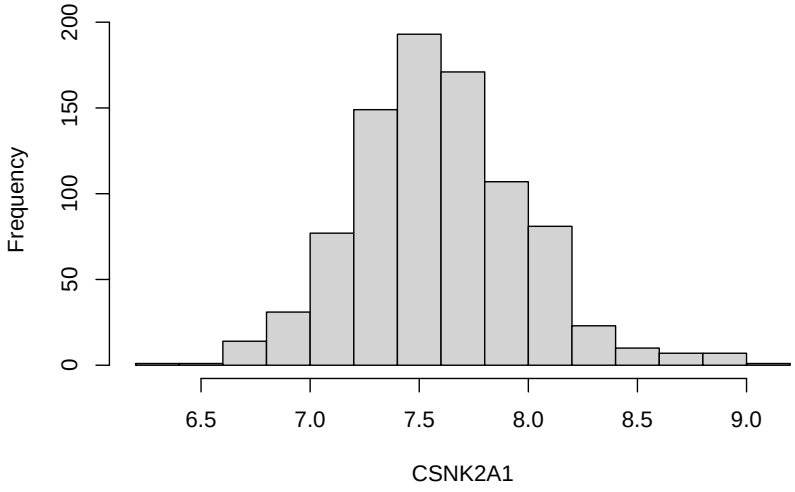
```
hist((data_full$IDO1)[!is.na(head(data_full$IDO1))], main = "Distribution of Gene IDO1",
    xlab = "IDO1")
```



3. CSNK2A1

- Symmetric
- 1 missing value
- impute mean

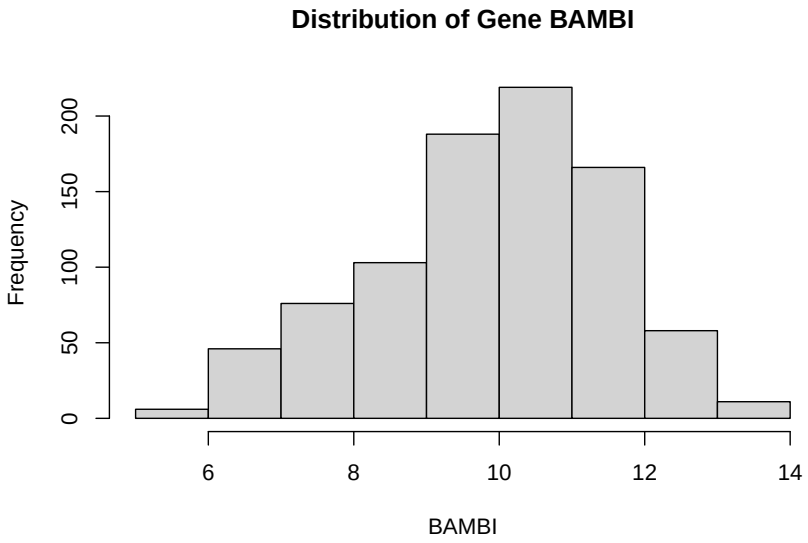
```
hist((data_full$CSNK2A1)[!is.na(head(data_full$CSNK2A1))], main = "Distribution of Gene CSNK2A1",
    xlab = "CSNK2A1")
```



4. BAMBI

- Slight left skew
- 1 missing value
- impute median

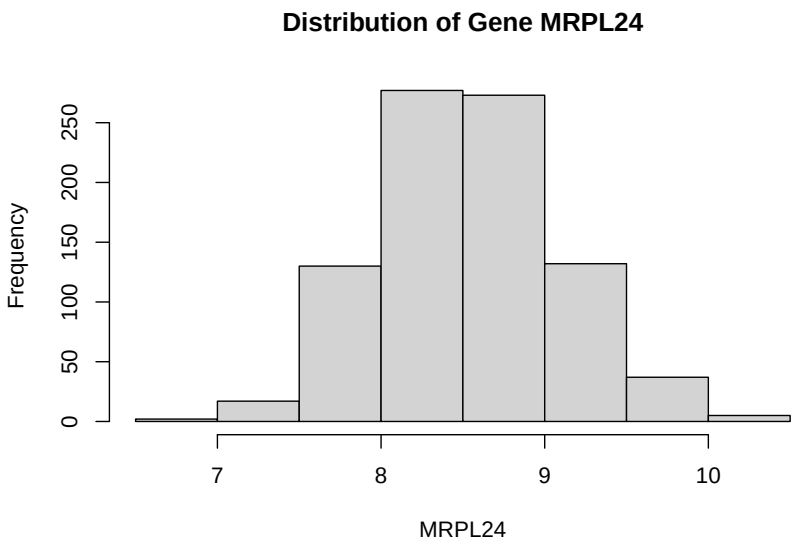
```
hist((data_full$BAMBI)[!is.na(head(data_full$BAMBI))], main = "Distribution of Gene BAMBI",
    xlab = "BAMBI")
```



5. MRPL24

- Symmetric
- 1 missing value
- impute mean

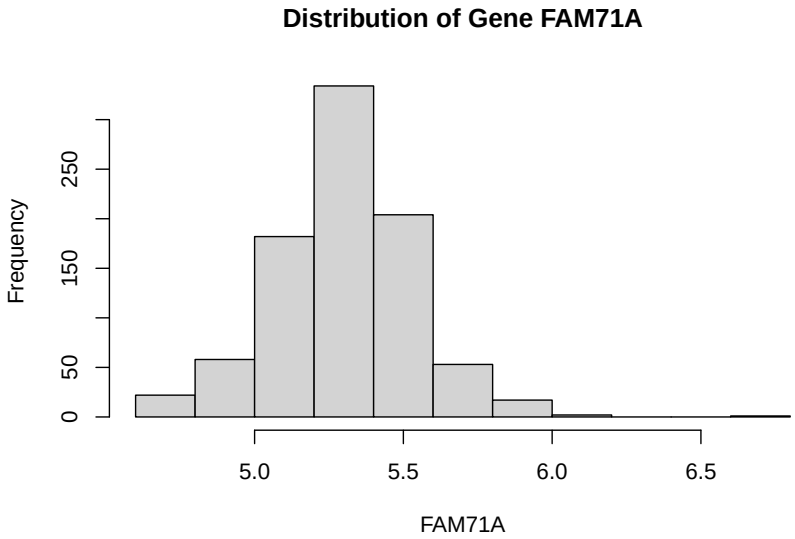
```
hist((data_full$MRPL24)[!is.na(head(data_full$MRPL24))], main = "Distribution of Gene MRPL24",
    xlab = "MRPL24")
```



6. FAM71A

- Right skew
- Has far right outliers
- impute median

```
hist((data_full$FAM71A)[!is.na(head(data_full$FAM71A))], main = "Distribution of Gene FAM71A",
    xlab = "FAM71A")
```



7. "Type of Breast Surgery"

- We would toss out patients with NA values in "Type of Breast Surgery" since there are 5 individuals

- 2 diff values (Breast Conserving, Mastectomy)

```
table(data.frame((data_full$`Type of Breast Surgery`)))
```

```
## X.data_full..Type.of.Breast.Surgery..
## BREAST CONSERVING        MASTECTOMY
##               382               487
```

```
sum(is.na((data_full$`Type of Breast Surgery`)))
```

```
## [1] 5
```

8. Cellularity

- 25 people have NA values
- Values: “High” ; “Moderate” ; “Low”
- Get rid of patients that have NA

```
table(data.frame((data_full$Cellularity)))
```

```
## X.data_full.Cellularity.
##     High      Low Moderate
##      423       89      337
```

```
sum(is.na(data_full$Cellularity))
```

```
## [1] 25
```

9. “ER status measured by IHC”

- 15 people have NA values
- IHC tests reveal more about cancer than standard biopsy tests, so we should probably keep this feature.
- Values: “Positive” ; “Negative”
- get rid of patients that have NA value

```
table(data.frame((data_full$`ER status measured by IHC`)))
```

```
## X.data_full..ER.status.measured.by.IHC..
## Negative  Positve
##      175      684
```

```
sum(is.na((data_full$`ER status measured by IHC`)))
```

```
## [1] 15
```

10. “Neoplasm Histologic Grade”

- 36 people have NA values
- Ordinal categorical data (factors) - 3 diff values
- get rid of patients with NA value

```
table(data.frame((data_full$`Neoplasm Histologic Grade`)))
```

```
## X.data_full..Neoplasm.Histologic.Grade..
##   1   2   3
##  72 337 429
```

```
sum(is.na((data_full$`Neoplasm Histologic Grade`)))
```

```
## [1] 36
```

11. "Tumor Other Histologic Subtype"

- 7 people have NA values
- Nominal with 8 categories
- will not use this feature

```
table(data.frame((data_full$`Tumor Other Histologic Subtype`)))
```

```
## X.data_full..Tumor.Other.Histologic.Subtype..
##          Ductal/NST             Lobular           Medullary         Metaplastic
##                 658                  69                  13                   1
##               Mixed            Mucinous               Other Tubular/ cribriform
##                 100                   7                  11                   8
```

```
sum(is.na((data_full$`Tumor Other Histologic Subtype`)))
```

```
## [1] 7
```

12. "Primary Tumor Laterality"

- 49 people have NA values
- Nominal with 2 categories
- will not use this feature

```
table(data.frame((data_full$`Primary Tumor Laterality`)))
```

```
## X.data_full..Primary.Tumor.Laterality..
##  Left Right
##   447   378
```

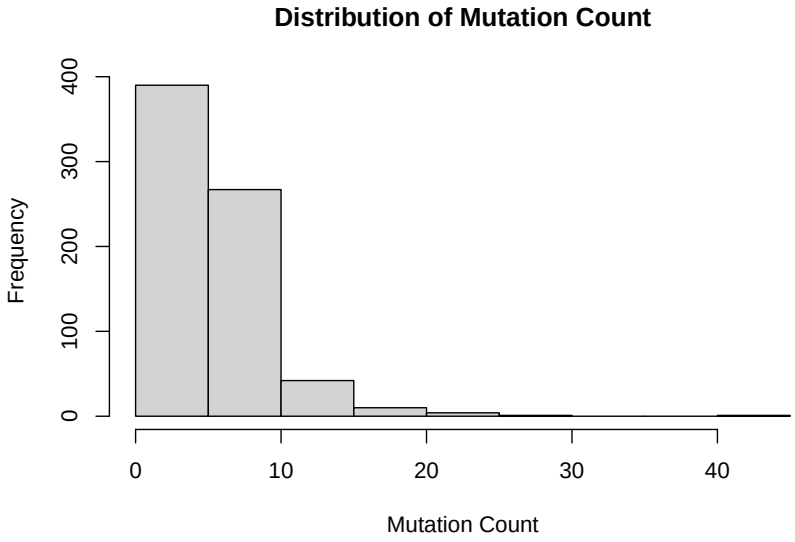
```
sum(is.na((data_full$`Primary Tumor Laterality`)))
```

```
## [1] 49
```

13. "Mutation Count"

- 20 people have NA values
- Quantitative data
- skew right
- use median to impute

```
hist((data_full$"Mutation Count")[!is.na(head(data_full$"Mutation Count"))],
    main = "Distribution of Mutation Count", xlab = "Mutation Count")
```



14. "Relapse Free Status"

- 1 missing value
- Values: "0: Recurred" ; "1: Not Recurred"
- get rid of the one patient

```
table(data_full$`Relapse Free Status`)
```

```
##
## 0:Not Recurred     1:Recurred
##            537            336
```

```
sum(is.na(data_full))
```

```
## [1] 499
```

15.3-gene classifer subtype * 103 missing values * Get rid of this feature due to too many missing values

```
table(data_full$`3-Gene classifier subtype`)
```

```
##
##             ER-/HER2- ER+/HER2- High Prolif  ER+/HER2- Low Prolif
##                   123                   281                   283
##                HER2+
##                    84
```

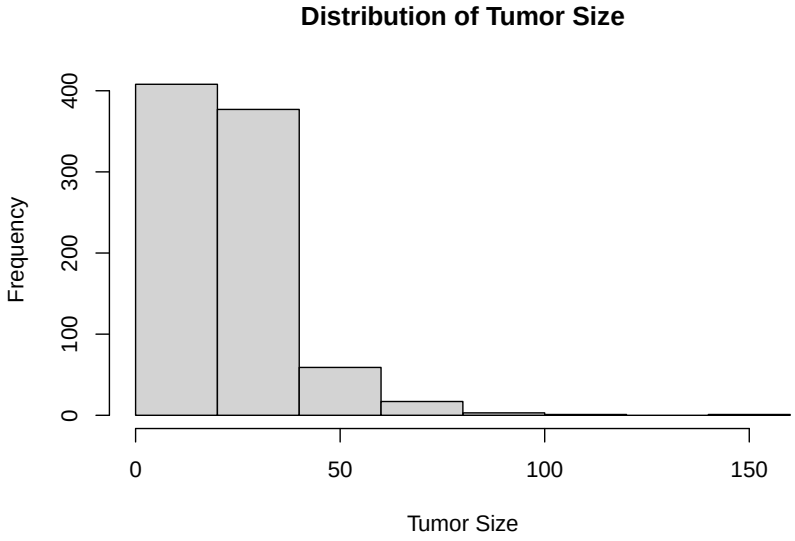
```
sum(is.na(data_full$`3-Gene classifier subtype`))
```

```
## [1] 103
```

16. Tumor Size

- 8 missing values
- numerical
- right skew
- use median to impute

```
hist(data_full$`Tumor Size`, main = "Distribution of Tumor Size",
    xlab = "Tumor Size")
```




```
sum(is.na(data_full$`Tumor Size`))
```

```
## [1] 8
```

17. Tumor Stage

- Has 306 missing values
- Ordinal categorical variable
- get rid of feature

```
sum(is.na(data_full$`Tumor Stage`))
```

```
## [1] 222
```

18. Patient's Vital Status

- Has 1 missing value
- Values: “Died of Disease”, “Living”, “Died of Other Causes”
- get rid of patient who has missing value

```
table((data_full$`Patient's Vital Status`))
```

```
##
##      Died of Disease Died of Other Causes               Living
##                  252                  221                  400
```

```
sum(is.na((data_full$`Patient's Vital Status`)))
```

```
## [1] 1
```

To summarize, our decisions were as follows:

The following features do not have many missing values, thus we will impute the following features with either median or mean (based on distribution of the feature). - Mutation Count - SLC25A19 - IDO1 - CSNK2A1 - BAMBI - MRPL24 - FAM71A - Tumor Size

For the following features, if a row of data does have an NA value for the feature, then we will toss the patient (row) out of the data set. We believe these features are important to building our model, yet these features lack the ability to impute a value for missing data, and/or not many patients (rows) have missing data, so tossing out patients that do are only a tiny amount of the data. - Type of Breast Surgery - Cellularity - ER status measured by IHC - Neoplasm Histologic Grade - Relapse Free Status - Patient’s Vital Status

We will not be using the following features in our model due to a large amount of missing values which we cannot impute and/or don’t believe will be useful in our model. - Tumor Other Histologic Subtype - Primary Tumor Laterality - 3-Gene classifier subtype - Tumor Stage

Feature Engineering Now that we have decided how to deal with each feature that has missing values, we will create a feature engineering function that takes in a data frame `x` very similar to `data_full` and makes all of these changes.

```
feature_eng = function(x) {
    ## imputing NA values for following columns
    x["Mutation Count"][is.na(x["Mutation Count"])] = median(x$"Mutation Count",
        na.rm = TRUE)
    x["SLC25A19"][is.na(x["SLC25A19"])] = mean(x$SLC25A19, na.rm = TRUE)
    x["IDO1"][is.na(x["IDO1"])] = median(x$IDO1, na.rm = TRUE)
    x["CSNK2A1"][is.na(x["CSNK2A1"])] = mean(x$CSNK2A1, na.rm = TRUE)
    x["BAMBI"][is.na(x["BAMBI"])] = median(x$BAMBI, na.rm = TRUE)
    x["MRPL24"][is.na(x["MRPL24"])] = mean(x$MRPL24, na.rm = TRUE)
    x["FAM71A"][is.na(x["FAM71A"])] = median(x$FAM71A, na.rm = TRUE)
    x["Tumor Size"][is.na(x["Tumor Size"])] = median(x$"Tumor Size",
        na.rm = TRUE)

    ## getting rid of rows with NA values in select columns
    ## we believe will be useful that also have relatively
    ## low missing values
    x = dplyr::filter(x, !is.na(`Type of Breast Surgery`) & !is.na(Cellularity) &
        !is.na(`ER status measured by IHC`) & !is.na(`Neoplasm Histologic Grade`) &
        !is.na(`Relapse Free Status`) & !is.na(`Patient's Vital Status`))
```

```
    ## getting rid of features that have too many missing
    ## values which we can't impute and/or we don't think
    ## would be that useful.
    x = dplyr::select(x, -c(`Tumor Other Histologic Subtype`,
        `Primary Tumor Laterality`, `3-Gene classifier subtype`,
        `Tumor Stage`))

    return(x)
}
```

Let's implement our function and see how many data points we lost!

```
data_final = feature_eng(data_full)

print(sprintf("# rows AFTER dealing with NA's in data_full: %s",
    nrow(data_final)))
```

```
## [1] "# rows AFTER dealing with NA's in data_full: 795"
```

```
print(sprintf("# cols AFTER dealing with NA's in data_full: %s",
    ncol(data_final)))
```

```
## [1] "# cols AFTER dealing with NA's in data_full: 24404"
```

```
sum(is.na(data_final))
```

```
## [1] 0
```

Great! We have gotten to know our data through EDA, data sculpting, and cleaning! Now, it's time to build our model.

3) Test/Train Split, Cross Validation

```
set.seed(121200)

spec = c(train = 0.8, test = 0.2)

g = sample(cut(seq(nrow(data_final)), nrow(data_final) * cumsum(c(0,
    spec)), labels = names(spec)))

res = split(data_final, g)

addmargins(prop.table(table(g)))
```

Test/Train Split

```
## g
## train  test   Sum
##   0.8   0.2   1.0
```

```
data_train = res$train
data_test = res$test
```

Here, we will load more necessary packages.

```
library("randomForest")
library("ranger")
library("devtools")
library("reprtree")
library("caret")
library("partykit")
library("multcomp")
library("party")
```

Cross Validation We will do k-fold cross validation on our training data with k = 10. We believe creating 10 folds is enough to stabilize our model given the data and our predictors.

```
train.control <- trainControl(method = "cv", number = 10)
```

4) Random Forest Model on Data

For our model, we will randomly select 150 genes and then include the variables discussed in our data cleaning and sculpting step; as a collection, these variables will be our features.

```
set.seed(121200)
X_train_without_select = dplyr::select(data_train, -c(y, `Type of Breast Surgery`,
    Cellularity, `ER status measured by IHC`, `Neoplasm Histologic Grade`,
    `Mutation Count`, `Relapse Free Status`, `Tumor Size`, `Patient's Vital Status`))
columns = colnames(X_train_without_select)

random_features_new = dplyr::select(X_train_without_select, sample(columns,
    size = 150, replace = F))
random_features_new["y"] = as.factor(data_train$y)
random_features_new["Type of Breast Surgery"] = data_train$`Type of Breast Surgery`
random_features_new["Cellularity"] = data_train$Cellularity
## there's also er status just by itself
random_features_new["ER status measured by IHC"] = data_train$`ER status measured by IHC`
random_features_new["Neoplasm Histologic Grade"] = data_train$`Neoplasm Histologic Grade`
random_features_new["Mutation Count"] = data_train$`Mutation Count`
random_features_new["Relapse Free Status"] = data_train$`Relapse Free Status`
random_features_new["Tumor Size"] = data_train$`Tumor Size`
## one for patient's vital status in months
random_features_new["Patient's Vital Status"] = data_train$`Patient's Vital Status`

random_features_new_final = data.frame(unclass(random_features_new))

# making col names of unclassed data same as other data
```

```
colnames(random_features_new_final) = colnames(random_features_new)

model <- train(y ~ ., data = random_features_new_final, method = "rf",
    trControl = train.control)
# Summarize the results
print(model)
```

```
## Random Forest
##
## 636 samples
## 158 predictors
##   2 classes: 'High', 'Low'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 572, 573, 572, 572, 573, 572, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.7798611  0.0000000
##    81   0.8646081  0.5918195
##   160   0.8646081  0.6001482
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 81.
```

Great! Our model has a very impressive training accuracy of around 86.5% with a Kappa Statistic of around 59.18%. Our Kappa Statistic tells us that our raters are in moderate to substantial agreement. This is good!

Now, let's see how our model does on the test set, and let's plot an ROC curve on those results.

```
data_features_test = data_test[colnames(data_test) %in% colnames(random_features_new_final)]

X_test = dplyr::select(data_features_test, -c("y"))
y_test = data_features_test$y

predictions_test = predict(model, X_test, y = "response")
```

Making Predictions on Test Set and Test Accuracy Let's create a confusion matrix and get our model accuracy on the test set, from which we will build the ROC curve.

```
## confusion matrix
tab = table(predictions_test, y_test)
confusion_matrix = confusionMatrix(tab)
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##                 y_test
## predictions_test High Low
```

```
##             High  124   7
##             Low     8  20
##
##                Accuracy : 0.9057
##                  95% CI : (0.8492, 0.9462)
##     No Information Rate : 0.8302
##     P-Value [Acc > NIR] : 0.005011
##
##                   Kappa : 0.6703
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.9394
##             Specificity : 0.7407
##          Pos Pred Value : 0.9466
##          Neg Pred Value : 0.7143
##              Prevalence : 0.8302
##          Detection Rate : 0.7799
##    Detection Prevalence : 0.8239
##       Balanced Accuracy : 0.8401
##
##        'Positive' Class : High
##
```

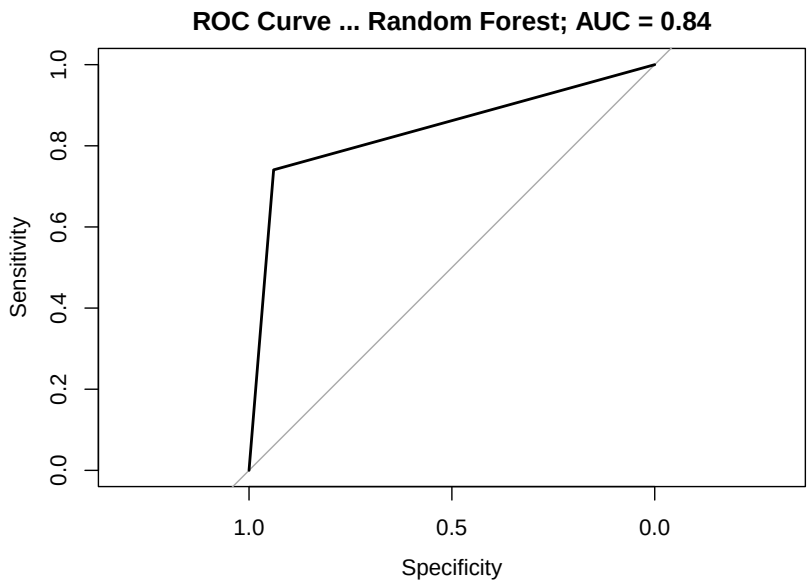
Wow! Our model has about 91% accuracy on the test set (thanks CV), with a 95% Confidence Interval: $[0.8492, 0.9462]$. The Kappa Statistic for model performance on the test set is higher as well at about 67%, meaning our raters are in substantial agreement!

ROC Curve Now, let's move on to the confusion matrix and ROC curve.

```
library(pROC)

tp = confusion_matrix$table["High", "High"]
fp = confusion_matrix$table["High", "Low"]
fn = confusion_matrix$table["Low", "High"]
tn = confusion_matrix$table["Low", "Low"]

tpr = tp/(tp + fn)
fpr = fp/(fp + tn)

roc_curve = roc(y_test, as.ordered(predictions_test))
```

```
## Warning in value[[3L]](cond): Ordered predictor converted to numeric vector.
## Threshold values will not correspond to values in predictor.
```

```
auc = (1 + tpr - fpr)/2
plot(roc_curve, colorize = T, lwd = 2, main = paste0("ROC Curve - Random Forest",
    "; AUC = ", round(auc, 2)))
```



This ROC curve tells us that our model is very close to being an ideal clinical discriminator (sensitivity and specificity at 1). Due to the curve being pulled in the top left corner and away from the $y = x$ line, we know that our model has pretty good predictive value.

5) Interogate the Random Forest

Let's look at our feature importances as calculated by the Gini Impurity metric.

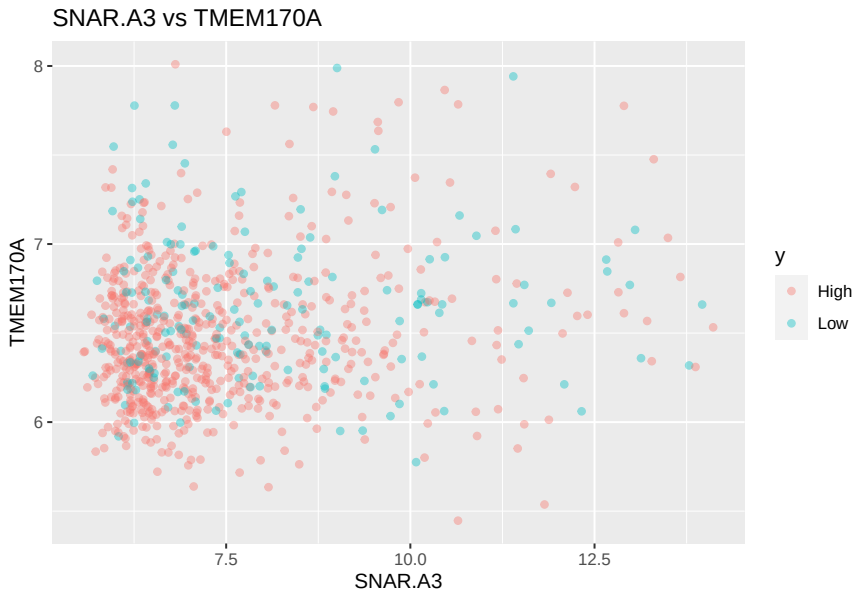
```
varImp(model)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 160)
##
##                                             Overall
## ‘Relapse Free Status‘1:Recurred             100.000
## ‘Patient’s Vital Status‘Living               74.892
## TMEM170A                                     21.446
## ‘Patient’s Vital Status‘Died of Other Causes 18.292
## ‘Tumor Size‘                                 13.339
## SNAR.A3                                       9.812
## CDK6                                          7.921
## G6PC3                                         7.888
## MRPL46                                        5.965
## C19orf70                                      5.841
## MIS18BP1                                      5.170
## C8orf40                                       5.144
## PIH1D2                                        5.100
## BC045810                                      5.010
## SPRY4                                         4.263
## ‘ER status measured by IHC‘Positve            3.962
## ‘Neoplasm Histologic Grade‘                   3.879
```

```
## ACD                                    3.835
## MYD88                                  3.750
## UCHL3                                  3.495
```

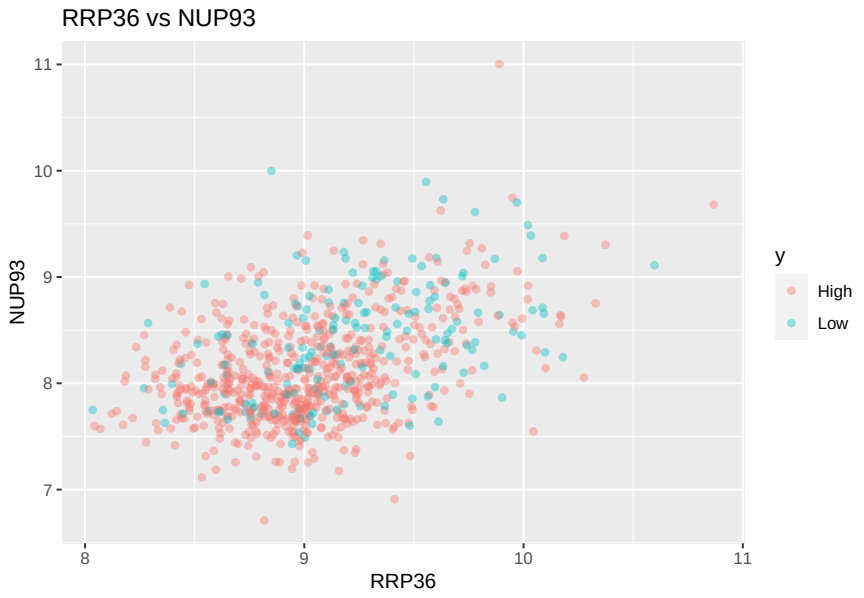
Let's look at the relationship between the two highest scoring genes: `TMEM170A` and `SNAR.A3`.

```
ggplot(data_final) + geom_point(aes(x = SNAR.A3, y = TMEM170A,
    colour = y), alpha = 0.4) + labs(title = "SNAR.A3 vs TMEM170A")
```



Wow! Although there is much overlap, it's clear that lower values of both `SNAR.A3` and `TMEM170A` are associated with 'High' cancer survival time and larger values of both (but specifically `SNAR.A3`) are associated with 'Low' cancer survival time. Also, it's interesting that SNAR.A3 has higher observed values overall in comparison to 7MEM170A, and they both clearly have lots of variability.

```
ggplot(data_final) + geom_point(aes(x = RRP36, y = NUP93, colour = y),
    alpha = 0.4) + labs(title = "RRP36 vs NUP93")
```



Here, we once again see very similar behavior as the previous two cells. We see a large cluster of values between $[8.5, 9.5] \in$ `RRP36` and $[7.5, 8.5] \in$ `NUP93` that is associated with 'High' cancer survival time. Additionally, we see that very high values of both are associated with 'Low' cancer survival time.

Conclusion From our analysis of breast cancer data of women aged 50-74, we saw a number of genes whose presence seemed linked to survival. In particular, the gene cluster between NUP93, PPP2R5D, NOP16 and RRP36 stood out as well as the pairing between SNAR.A3 and TMEM170A. When looking at the gene RRP36, we saw that according to UniProt's database, RRP36 is frequently predicted to cause consequences no matter where it's located in the DNA, either that or its effect is uncertain in some small cases. If this is generally the case, then it makes sense that the presence of such a gene would decrease the chances of high survival for people.

The NUP39 gene is known to be responsible for programmed cell death and it's awfully convenient that cancer cells are known for not dying when they're meant to. If there is an issue with this gene, it would mean that a cancer would be able to spread without control and take over and once again decrease the likelihood of survival. The other gene, NOP16 is a known marker of low breast cancer survival according to the NIH.

There are similar stories for all of these genes, and it was incredible to find these results with the random forests we learnt about in class. It goes to show how powerful such data analysis methods are even on such a large number of variables. Our group had a lot of fun working with the data and fitting a Random Forest model. We hope you enjoy our report.