Coding Challenge

Jessica Rowell

2022-12-20

Summary

After exploring the data and doing some data cleaning (see Section 1 and Section 3), I normalized the RNA expression data (see Section 2). I extensively explored clustering by HER2 status, *ERBB2* copy number, and other potential explanatory variables such as age, ethnicity, race, menopause status, and hormone receptor status (see Section 4). I employed some modeling to elucidate the predictive power of *ERBB2* copy number and RNA expression data on HER2 status (see [Section 5])(#predict). A random forest model trained on *ERBB2* and the explanatory variables performed with 56% recall and 89% precision (88% accuracy) on the training data, but on the validation data they performed less well. This suggests the model may be overfitted to the data. I trained a random forest model on the RNA expression data (normalized using DESeq2's median-of-ratios normalization), and this model performed with 46% recall and 92% precision on the training data. Additional modeling efforts with a subset of the 1,857 most differentially expressed genes yielded similar results (see Section 6) - random forest model: 53% recall, 92% precision, 88% accuracy; SVM model: 70% recall, 75% precision, 88% accuracy.

In layman's terms, this means that both *ERBB2* copy number and overall RNA expression are not good at predicting HER2+ status. However, they are both unlikely to misclassify an HER2- person as HER2+. From a clinical perspective, there is always a trade-off in this regard. One has to decide what is most important: catching a person who is HER2+ or not misclassifying a person as HER2+ who is actually HER2-. For example, say you have a treatment that improves survival only in HER2+ patients: the consequences of not identifying your target patients are significant (they don't get your treatment). But if the treatment also have potentially severe side effects, the consequences of treating someone who doesn't really need that treatment (i.e., an HER2- patient) are also significant.

To address the overfitting I observed for all random forest models, I could try hypertuning the parameters; especially since I didn't perform any parameter tuning for this exploratory assignment. I would also like to look for additional samples to test. In this dataset only 22% of patients are HER2+, which can cause some issues in modeling. The *ERBB2* copy number levels are also imbalanced. I would like to research *ERBB2* more and try other options for modeling copy number.

In conclusion, I don't think either explanatory variable performs impressively well. Both models can do a great job at predicting HER2- status, with specificity for all models ranging from 98-99%. But none can accurately predict HER2+ status; most of them only get it right a little over 50% of the time. That's as good as a coin toss.

Libraries

```
library(tidyverse)
library(PCAtools)
library(DESeq2)
library(randomForest)
library(ggplot2)
library(ggrepel)
library(RColorBrewer)
library(pheatmap)
library(apeglm)
library(tibble)
library(e1071)
library(caret)
library(knitr)
```

Functions

```
allDup <- function (value)
{
  duplicated(value) | duplicated(value, fromLast = TRUE)
}
```

Section 1.

###. Read in the data & explore

```
cn <- read.csv('brca_tcga_erbb2_copy_number.csv', header = TRUE)
sum(duplicated(cn$patient_id)) # Check for duplicates
table(cn$erbb2_copy_number, exclude=NULL) # Copy number ranges from -2 to 2
```

This dataset includes patient ID, sample ID, and *ERBB2* copy number for 963 patients.

```r
rsem <- read.csv('tcga.brca.rsem.csv', header = TRUE)
dim(rsem) # 1212 rows, 20505 columns
length(unique(rsem$patient_id)) # 1093 unique patients
table(rsem$sample_type) # 3 sample types:
head(rsem[,4:13], 10) # Counts are not integers
any(is.na(rsem)) # No missing values
sum(colSums(rsem[,4:ncol(rsem)]) < 1) # There are 320 non-informative genes
# Get patient ids for metastasis type
m <- unlist(subset(rsem, sample_type == 'Metastasis', select=patient_id))
# Check if those with metastasis sample also have primary sample
table(subset(rsem, patient_id %in% m, select = c(patient_id,sample_type)))
```

This dataset includes the patient ID, the patient barcode, the sample type, and counts for 20,502 genes. It contains 1,093 unique patients, and for some patients there are multiple sample types. Most of the samples are primary tumor (n=1,093) but a few are metastatis type (n=7). There are also some adjacent normal tissue samples (n=112).

Section 2.

QC and normalize the RNA dataset

```
r <- subset(rsem, sample_type == 'Primary Tumor') # Keep only primary tumors
dim(r) # 1093 rows, 20505 columns

length(unique(r$patient_id)) # Double-check there are no duplicate patient IDs
rownames(r) <- r$patient_id # Set rownames to patient ID
r <- as.matrix(subset(r, select = -c(1:3)))  # Get counts as a matrix
r <- r[ ,colSums(r) > 0] # Remove those non-informative genes
r <- t(t(r)/rowSums(t(r))) # Normalize: for each gene, divide by sum (counts)
r <- round(r*10**6, digits = 0) # Multiply by 10^6 & round to reduce computation
```

I will limit this analysis to the primary tumor type. Every patient with a metastasis sample also has a primary tumor sample, so this doesn't decrease the number of observations. Also, it seems that primary tumors and metastases are genomically very similar. [1] Thus, I believe removing the metastasis samples greatly simplifies the analysis (by eliminating correlated data) without impacting or biasing information yield.

Taking a look at the data, I see that the counts are not integers. For this assignment, I will assume these are expected counts derived from ML abundances. [2] I will also assume that no normalization for library size or gene length was applied (i.e. if tximport was used to generate this aggregate RSEM file, no scaling was done).

I removed genes with expected counts less than 1 across all samples. (I use <1 instead of 0 because the data aren't integers). Given the number of genes, I wondered whether these were all protein-coding. [3] Looking closer I see some FISH probes (e.g. HBII.52.27) and maybe some non-standard gene names (e.g. LOC653545, which is listed as *DUX4L5*) so I think this dataset would need a deeper cleaning.

I normalized for read depth only, but in practice I would use DESeq2's normalization function, which normalizes for both sequencing depth and RNA composition. Normalizing for RNA composition is important if some genes because when a gene is is much more abundant than others in a sample, the less abundant genes in that sample will appear under-expressed than those same genes in other samples (because of the skewed scaling factor). For this exploratory analysis, I've only normalized for read length.

(EDIT: I actually later normalized using DESeq2 and repeated all the clustering. The PCA plots did not meaningfully change. See Appendix)

Section 3

Identify HER2+ patients

```
clin <- read.csv('brca_tcga_clinical_data.csv', header = TRUE)
sum(duplicated(clin$Patient.ID)) # 7 duplicate patients present
table(clin$IHC.HER2, exclude=NULL) # 567 Neg, 164 Pos, 180 Equiv, 12 Indeterm
# The duplicate patients each have two samples IDs, ending in 01 & 06
cdupes <- clin[allDup(clin$Patient.ID),1:3]
# Group by patient ID, slice the lowest sample ID
clin <- clin %>% group_by(Patient.ID) %>% dplyr::slice_min(Sample.ID, n=1)
dim(clin) # 7 rows removed
# Confirm removal of the duplicate patients' 06 samples
table(subset(clin, Patient.ID %in% cdupes$Patient.ID,
             select=c(Patient.ID, Sample.ID)))
clin <- clin[!is.na(clin$IHC.HER2),] # Remove patients with missing HER2 status
```

The dataset includes study ID, patient ID, sample ID, and 137 variables containing clinical information for 1,108 patients. There are some duplicate patients because a patient can have multiple samples. 7 patients each have two samples (ending in either 01 or 06). I know that the copy number dataset does not contain duplicates, and all of the sample IDs end in 01. In addition, the barcodes in the RSEM data have 01 in the sample ID portion (first 15 digits). So for each duplicate patient I kept the clinical data with sample ID ending in 01 so that the clinical data corresponds to the copy number data. I also removed patients missing HER2 status. In a "real-world" scenario, I would try data imputation methods and compare results.

```
any(colnames(clin) == 'HER2') # Make sure I don't overwrite an existing variable
clin$IHC.HER2 <- factor(clin$IHC.HER2)
# Create a new variable that combines the 2 unknown HER2 statuses
clin <- clin %>% mutate(HER2 = case_when(IHC.HER2 == 'Negative' ~ 'Negative',
            IHC.HER2 == 'Positive' ~ 'Positive',
            IHC.HER2 == 'Equivocal' | IHC.HER2 == 'Indeterminate' ~ 'Unknown'))
table(clin$IHC.HER2, clin$HER2, exclude=NULL) # Check new variable assignment
```

For this exploratory analysis, I am using just the IHC-HER2 column to classify HER2 status (as suggested in the instructions). From initial exploration of the IHC-HER2 column, I can see there are 567 HER2-, 164 HER2+, and 192 HER2 status equivocal/unknown samples. Since only 22% of patients available for modeling (those with HER2+ or HER2- status) are HER2+, this could create some modeling issues. I combined the Equivocal and Indeterminate statuses here, but given more time I would use FISH test results to recategorize some Equivocal patients as Negative/Positive based on American Cancer Society's explanation of HER2 testing.

Section 4

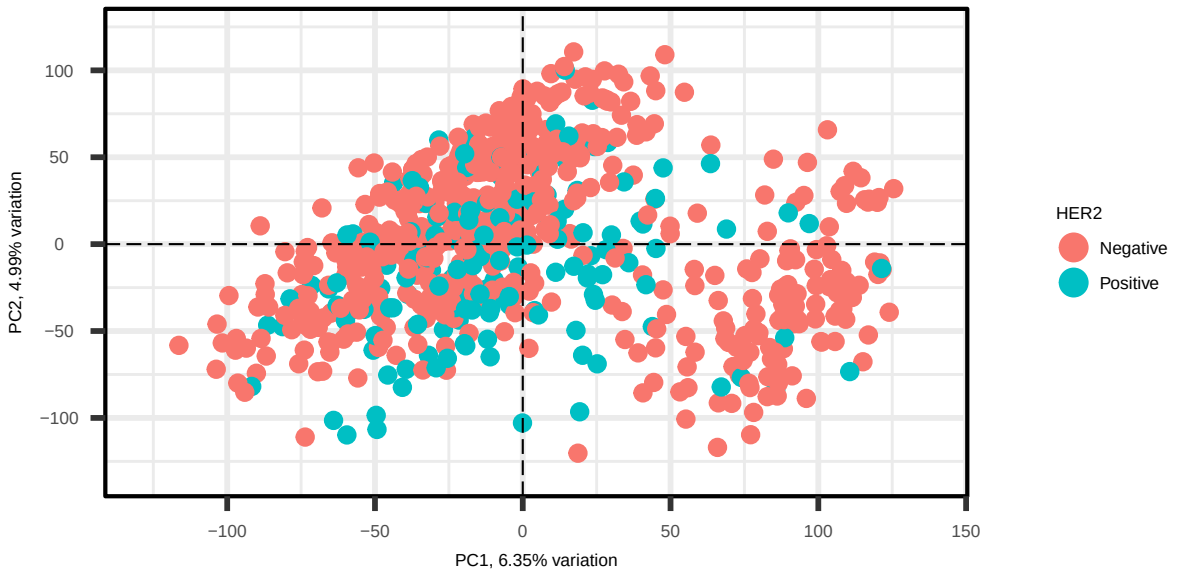
Cluster the cancer RNA data

```
# Get simplified clinical dataset (later will add to this)
c2 <- as.data.frame(clin[,c("Patient.ID", "HER2", "Diagnosis.Age",
                            "Ethnicity.Category", "Race.Category",
                            "Menopause.Status", "ER.Status.By.IHC",
                            "PR.status.by.ihc")])
rownames(c2) <- c2$Patient.ID
r2 <- log(r + 1) # Log transform the normalized counts
# Keep IDs with both RNA data and HER2 status
common_ids <- intersect(rownames(r2), rownames(c2))
r2 <- subset(r2, rownames(r2) %in% common_ids)
c2 <- subset(c2, rownames(c2) %in% common_ids)
# Clean up the clinical variables and create some new ones
colnames(c2) <- c("Patient.ID", "HER2", "Age", "Ethnicity",
                  "Race", "Menopause", "ER", "PR") # Shorten column names
c2$Menopause <- factor(gsub(r"{\s*\([^\)]+\)}","",as.character(c2$Menopause)))
# Create a categorical age variable in case I want to use for visualization
c2 <- c2 %>% mutate(Agecat = case_when(Age <= 50 ~ "<51",
                                       Age > 50 ~ "50+"))
table(c2$Agecat, c2$Age, exclude=NULL) # Check categorical age var
# Add a variable for Triple receptor status (Positive, Negative, or Neither).
c2 <- c2 %>%
 mutate(Triple = case_when(
    is.na(ER) | is.na(PR) |
      ER == 'Indeterminate' | PR == 'Indeterminate' ~ 'Unknown',
    ER == 'Negative' & PR == 'Negative' & HER2 == 'Negative' ~ 'Negative',
    ER == 'Positive' & PR == 'Positive' & HER2 == 'Positive' ~ 'Positive',
    TRUE ~ 'Neither'))
# Add var for hormone receptor status (+ if PR+/ER+, - if ER- & PR-, else Neither)
c2 <- c2 %>%
  mutate(HR = case_when(
    is.na(ER) | is.na(PR) |
      (ER == 'Indeterminate' & PR == 'Indeterminate') ~ 'Unknown',
    ER == 'Negative' & PR == 'Negative' ~ 'Negative',
    ER == 'Positive' | PR == 'Positive' ~ 'Positive',
    TRUE ~ 'Neither'))
ftable(c2$Triple, c2$HR, c2$PR, exclude = NULL) # Check coding of  Triple
table(c2$ER, c2$PR) # Need this to help check coding of the HR variable
ftable(c2$HR, c2$ER, c2$PR, exclude = NULL) # Check HR var coding
c2$Ethnicity <- as.factor(c2$Ethnicity)
c2$Race <- as.factor(c2$Race)
```

```
c2$ER <- as.factor(c2$ER)
c2$PR <- as.factor(c2$PR)
c2$HR <- as.factor(c2$HR)
c2$Triple <- as.factor(c2$Triple)

# Join the ERBB2 copy number info to clinical data to create a complete dataset
c2 <- left_join(
  c2, cn %>% dplyr::select(patient_id, erbb2_copy_number),
  by = c("Patient.ID" = "patient_id"), keep = FALSE)
table(c2$erbb2_copy_number, exclude=NULL) # 110 obs are missing copy number
colnames(c2)[colnames(c2) == 'erbb2_copy_number'] <- 'ERBB2' # Simplify col name
c2$ERBB2 <- as.factor(c2$ERBB2)
rownames(c2) <- c2$Patient.ID

r2 <- t(r2) # Transpose count matrix to do visualizations
```

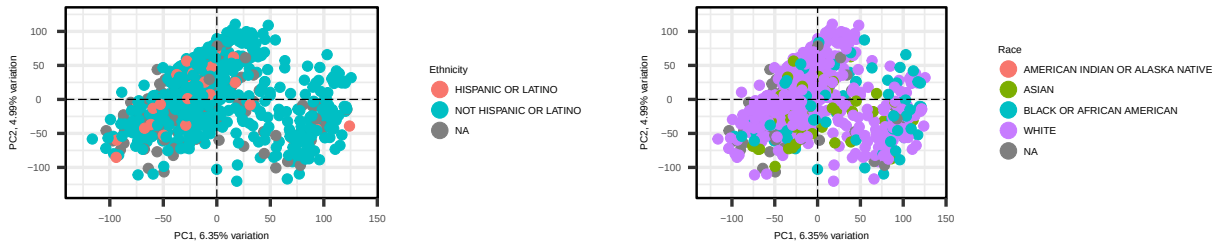
I log-transformed the data for visualization purposes, adding a 1 to every cell to eliminate 0's (since after log-transforming they will result in -infinity values that disrupt further analysis and visualization). Another way to log-transform would be to change all the zero's to a very small number (e.g. 0.0001) and I would compare both methods in a "real-world" setting because log transformations should be done carefully. I also subset the clinical variables to keep the ones I will explore and potentially use in analysis.

```
labels <- subset(c2, c2$HER2 != "Unknown") # Remove the HER2 status Unknowns
labels$HER2 <- factor(labels$HER2) # Reset factor levels
common_ids <- intersect(colnames(r2), rownames(labels))
data <- subset(r2, select = common_ids)
p <- pca(data, metadata = labels, removeVar = 0.1) # Run the PCA
biplot(p, lab = NULL, colby = 'HER2',
       hline = 0, vline = 0,
       labSize = 7, axisLabSize = 7,
       legendLabSize = 7, legendTitleSize = 7,
       legendPosition = 'right')
```

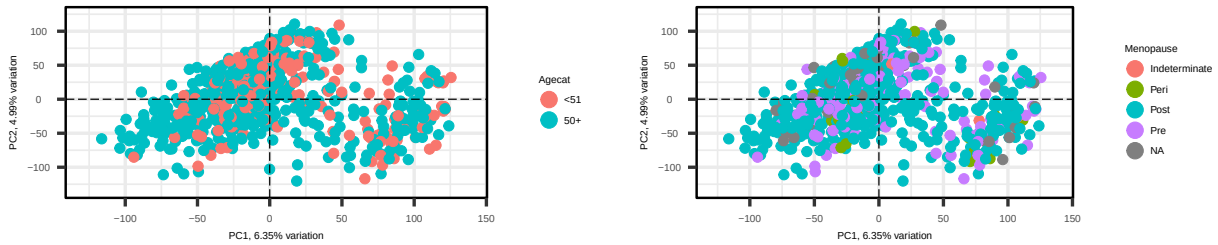


My first observation is that the dataset is noisy (PC1 and PC2 account for very little variation), but two clusters are visible and HER2+ patients seem to mostly cluster together. The HER2- patients are spread throughout both visible clusters. I can see that while HER2+ observations do cluster together, they cluster with HER2- observations.

```
biplot(p, lab = NULL,
       colby = 'Ethnicity', hline = 0, vline = 0,
       labSize = 7, axisLabSize = 7,
       legendLabSize = 7, legendTitleSize = 7, legendPosition = 'right')
biplot(p, lab = NULL,
       colby = 'Race', hline = 0, vline = 0,
       labSize = 7, axisLabSize = 7,
       legendLabSize = 7, legendTitleSize = 7, legendPosition = 'right')
```



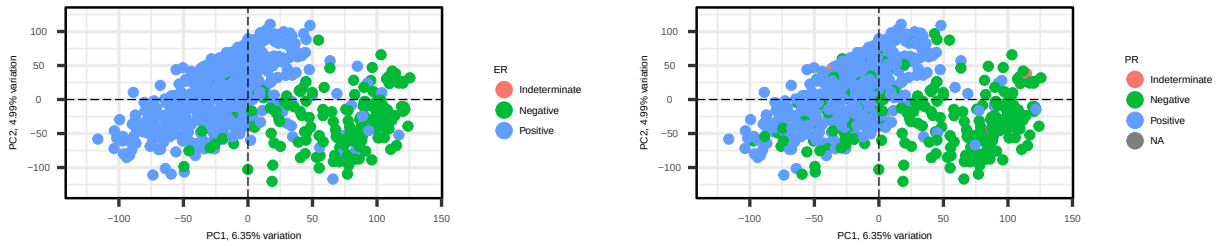
```
biplot(p, lab = NULL,
       colby = 'Agecat', hline = 0, vline = 0,
       labSize = 7, axisLabSize = 7,
       legendLabSize = 7, legendTitleSize = 7, legendPosition = 'right')
biplot(p, lab = NULL,
       colby = 'Menopause', hline = 0, vline = 0,
       labSize = 7, axisLabSize = 7,
       legendLabSize = 7, legendTitleSize = 7, legendPosition = 'right')
```



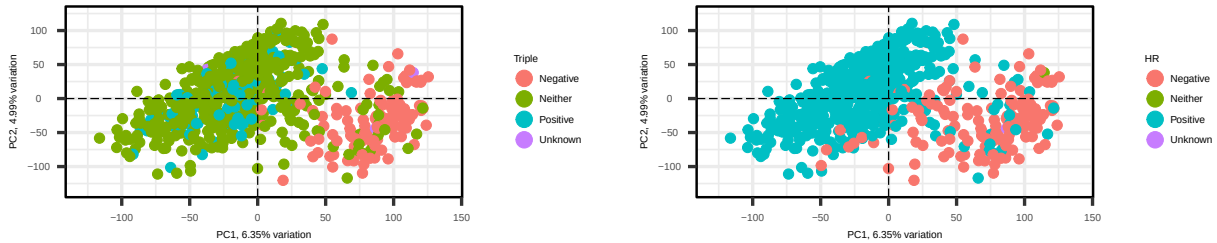
Quickly clustering on some other variables just to get initial insights, I see that some clustering effects with ethnicity and possibly menopause (peri-menipausal patients might be concentrating in the larger of the two clusters). It's too difficult to tell from these plots in this case, but it can be a way of identifying potential confounders for modeling later. Both ethnicity and menopause status might be correlated with HER2, so I need to consider that in modeling. I should also consider age and race in this way, even though the clustering is not obvious here, because the number of data points may obscure some effects. I also examined pair plots to look for clustering in other PCs, but I did not observe any (not surprising because PC1 and PC2 around for very little variation).

After further reading, I decided to look at a few other clinical variables and their relationship with HER2 status. I looked at ER positivity status and PR positivity status because both factors influence patient treatment decisions.

```
biplot(p, lab = NULL,
       colby = 'ER', hline = 0, vline = 0,
       labSize = 7, axisLabSize = 7,
       legendLabSize = 7, legendTitleSize = 7, legendPosition = 'right')
biplot(p, lab = NULL,
       colby = 'PR', hline = 0, vline = 0,
       labSize = 7, axisLabSize = 7,
       legendLabSize = 7, legendTitleSize = 7, legendPosition = 'right')
```

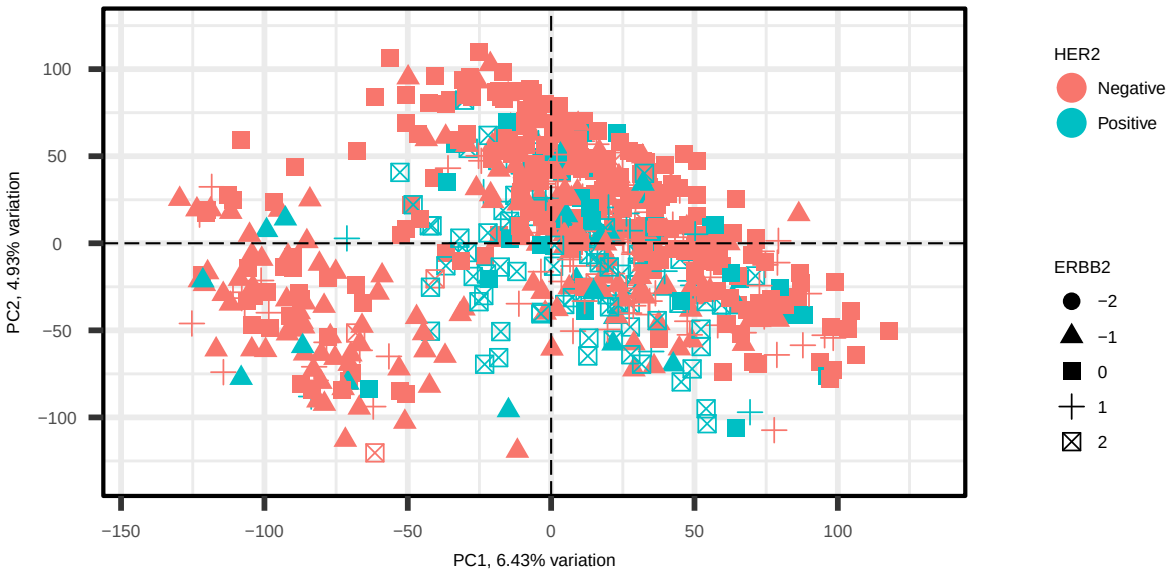



```
biplot(p, lab = NULL,
       colby = 'Triple', hline = 0, vline = 0,
       labSize = 7, axisLabSize = 7,
       legendLabSize = 7, legendTitleSize = 7, legendPosition = 'right')
biplot(p, lab = NULL,
       colby = 'HR', hline = 0, vline = 0,
       labSize = 7, axisLabSize = 7,
       legendLabSize = 7, legendTitleSize = 7, legendPosition = 'right')
```



Here I can see that hormone receptor status clearly segregates into clusters. I can also see that the Triple-negative group forms a cluster, whereas Triple-positive patients cluster with those who have either HER2+, ER+, or PR+ tumors (or some combination of two of those). Triple-negative breast cancers are an important subgroup because they grow and spread faster than other types of breast cancer, according to the American Cancer Society. From this exploration, my preliminary hypothesis is that HER2 status doesn't impact clustering as much as hormone receptor status, but also HER2- status (n=558) is 3.4 times more prevalent in this dataset than HER2+ status (n=163). There is a limit to what I can guess just from visualizations.

```
labels <- labels[!is.na(labels$ERBB2),] # Remove obs with missing ERBB2 status
common_ids <- intersect(colnames(r2), rownames(labels))
data <- subset(r2, select = common_ids)
p <- pca(data, metadata = labels, removeVar = 0.1) # Run the PCA
biplot(p, lab = NULL,
       colby = 'HER2', shape = 'ERBB2', hline = 0, vline = 0,
       labSize = 7, axisLabSize = 7,
       legendLabSize = 7, legendTitleSize = 7, legendPosition = 'right')
```



Finally, I wanted to try to visualize *ERBB2* copy number and HER2 status in a PCA plot. This graph is somewhat busy, but I can see clearly that those with 2 copies of *ERBB2* cluster together. For this analysis, I removed the observations with missing ERBB2 status. In a "real-world" scenario, I might also try data imputation methods and compare results.

Section 5.

Analyze how RNA, copy number, and clinical status are correlated.

How predictive is DNA of clinical status?

```
sapply(labels,function(x) sum(is.na(x))) # Assess amount of missing data
# Impute missing data
labels.imputed <- rfImpute(
  HER2 ~ ERBB2 + Age + Ethnicity + Race + Menopause + ER + PR + Triple + HR,
  data = labels, iter=5)

n.test = ceiling(nrow(labels) * 0.3) # Test set will have 194 obs (30%)
ind = sample(nrow(labels), n.test, FALSE) # Randomly sample 194 rows from counts

# Train and test RF model using all variables (except the ones I created)
r.train <- labels.imputed[-ind,]
r.train <- subset(r.train, select = -c(HER2, HR, Triple)) # Train
l.train <- data.frame(HER2 = labels.imputed[-ind, "HER2"]) # Outcome
r.test <- labels.imputed[ind,]
r.test <- subset(r.test, select = -c(HER2, HR, Triple)) # Test data
l.test <- data.frame(HER2 = labels.imputed[ind, "HER2"]) # Outcome
model <- randomForest(x = r.train, y = l.train[,1],
                      xtest = r.test, ytest = l.test[,1], proximity = TRUE)
model

# How well do the other variables predict HER2 (not ERBB2)?
r.train <- labels.imputed[-ind,]
r.train <- subset(r.train, select = -c(HER2, ERBB2, HR, Triple))
l.train <- data.frame(HER2 = labels.imputed[-ind, "HER2"]) # Outcome
r.test <- labels.imputed[ind,]
r.test <- subset(r.test, select = -c(HER2, ERBB2, HR, Triple)) # Test
l.test <- data.frame(HER2 = labels.imputed[ind, "HER2"]) # Outcome
model <- randomForest(x = r.train, y = l.train[,1],
                      xtest = r.test, ytest = l.test[,1], proximity = TRUE)
model

# Train and test using ERBB2 only
r.train <- labels[-ind,]
r.train <- subset(r.train, select = ERBB2) # Training data
l.train <- data.frame(HER2 = labels[-ind, "HER2"]) # Outcome

r.test <- labels[ind,]
r.test <- subset(r.test, select = ERBB2) # Test data
```

```
l.test <- data.frame(HER2 = labels[ind, "HER2"]) # Outcome
model <- randomForest(x = r.train, y = l.train[,1],
                      xtest = r.test, ytest = l.test[,1], proximity = TRUE)
model
```

Here I tried some data imputation on some of the independent explanatory variables: ethnicity (137 missing), race (89 missing), menopause (44 missing), and PR (1 missing). Here I used one imputation method and 5 iterations, but in practice I would try different options and compare the imputed values for consistency and robustness of the imputation process.

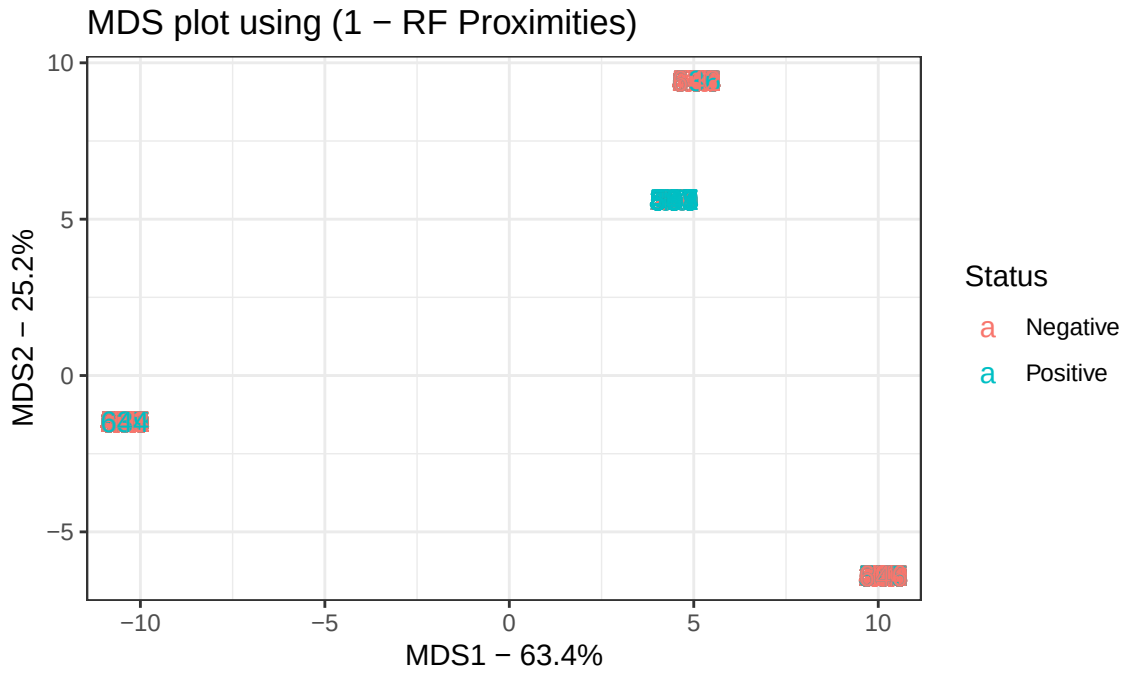
I built a random forest model from the imputed data to classify HER2 status using *ERBB2* copy number, age, ethnicity, race, menopause status, ER status, and PR status. I also built a simple RF model on the original dataset including only *ERBB2* copy number. For both models, the training data data out-of-bag (OOB) error rate was 12% - meaning that for samples not included in the bootstrapping data used to build the RF, the model correctly classified HER2 status for 88% of them. The model's recall is 56% (proportion of HER2+ patients correctly classified as HER2+) and precision is 89% (proportion of patients classified as HER2+ who are truly HER2+). If we use *ERBB2* copy number to identify HER2+ patients, we will miss a lot of HER2+ people but we are unlikely to misclassify an HER2- patient as HER2+. *ERBB2* copy number is not great at capturing HER2+ patients, but it's good at predicting HER2- status. I saw evidence of this in the clustering - HER2+ patients seemed to possibly cluster in the middle of HER2- patients. The model's accuracy is 89%. When I leave *ERBB2* out of the model and predict HER2 using the other variables, the error rate increases to 23%, and the accuracy drops to 77%. It appears that of all the predictors I have selected to explore, *ERBB2* copy number has the most predictive power on HER2 status and the additional of other variables doesn't improve classification.

However, the test data OOB error rate was consistently 1-3% higher than that of the training data; this suggests that the RF model may be overfitted to the data and may not perform as well on a different dataset. Thus, it is difficult to say whether the result is true or whether it is a characteristic of this dataset. For example, HER2 status is relatively unbalanced, with only 22% of patients HER2+. We also have some imbalance in *ERBB2* copy number: only 1 patient has -2 copies, 33% of patients have 1 or 2 copies, and 67% have 0 or -1 copy. I would want to explore other datasets before making a conclusion.

(In a real-world scenario, for my first model I would vary the number of variables chosen to make a decision tree and see how it affects OOB error rate, but here I've used the default, which is to choose 3 variables at a time. Also, I would try some GLM models and try different ways to model *ERBB2* copy number. It can be treated as an ordinal variable, or categorized into 2 or 3 groups, etc. I would explore and research this gene more.)

```
# Create MDS plot with all data
model <- randomForest(HER2 ~ ERBB2, data = labels, proximity = TRUE)
d.mat <- dist(1-model$proximity) # Get a distance matrix
mds <- cmdscale(d.mat, eig=TRUE, x.ret=TRUE) # Run MDS scaling
mds.pv <- round(mds$eig/sum(mds$eig)*100, 1) # Calc percent variation
```

```
# Format data for plotting
mds.vals <- mds$points
mds.d <- data.frame(Sample=seq.int(1,nrow(mds.vals)),
                    X=mds.vals[,1],
                    Y=mds.vals[,2],
                    Status=labels$HER2)
ggplot(data=mds.d, aes(x=X, y=Y, label=Sample)) +
  geom_text(aes(color=Status)) +
  theme_bw() +
  xlab(paste("MDS1 - ", mds.pv[1], "%", sep="")) +
  ylab(paste("MDS2 - ", mds.pv[2], "%", sep="")) +
  ggtitle("MDS plot using (1 - RF Proximities)")
```



The MDS plot shows that the samples mostly segregate and cluster tightly together, but there is one cluster in which HER2+ and HER2- samples cluster tightly together. This agrees with the results so far.

How predictive is RNA of clinical status?

```
# DESeq2 requires non-normalized counts
r <- subset(rsem, sample_type == 'Primary Tumor') # Keep only primary tumors
dim(r) # 1093 rows, 20505 columns
rownames(r) <- r$patient_id # Set rownames to patient ID
r <- as.matrix(subset(r, select = -c(1:3)))  # Get counts as a matrix
```

```
z.cols <- which(apply(r, 2, function(x) all(x < 1))) # <1 b/c not integers yet
r <- r[, -z.cols] # Remove those non-informative genes (now n=20,079)
# Later I decided to remove all genes with >50% zeros among the samples
half.z.cols <- which(apply(r, 2, function(x) mean(x == 0) >= 0.5))
r <- r[, -half.z.cols]
r <- subset(r, rownames(r) %in% rownames(labels)) # Subset new count matrix
r <- t(round(r, digits = 0)) # DESeq2 ob must contains integers
nrow(labels) == ncol(r) # Check that datasets match up
dds <- DESeqDataSetFromMatrix(countData = r,
                              colData = labels, design = ~ HER2)
keep <- rowSums(counts(dds)) >= 10 # Apply stricter pre-filter
dds <- dds[keep,] # New count is same (after adding the half.z.cols filter)

# Try to build a RF model (just for comparison)
dds <- estimateSizeFactors(dds) # Normalize for seq depth
rld <- vst(dds, blind=TRUE) # Variance-stabilizing transform
rld.mat <- t(assay(rld))

identical(rownames(rld.mat), rownames(labels)) # Check order
n.test = ceiling(nrow(rld.mat) * 0.3) # Test set contains 194 obs (30%)
ind = sample(nrow(rld.mat), n.test, FALSE) # Randomly sample 194 rows from counts
r.train <- as.matrix(rld.mat[-ind,]) # Training data
l.train <- data.frame(HER2 = labels[-ind, "HER2"]) # Outcome

r.test <- as.matrix(rld.mat[ind,]) # Testing data
l.test <- data.frame(HER2 = labels[ind, "HER2"])
model <- randomForest(x = r.train, y = l.train[,1],
                      xtest = r.test, ytest = l.test[,1],
                      importance = TRUE, proximity = TRUE)
model
```

I prepared the data for use with DESeq2, normalized the data and applied DESeq2's variance-stabilizing transform. I also tried to apply log regularization, but it took too long because of the number of samples. DESeq2 documentation suggests that VST works as well and with improved performance for >100 samples. I trained a random forest model on 70% of the data and achieved an out-of-bag error rate of 13%. The model recall was 46% and the precision was 92%. Overall accuracy was 87%, so by using the RNA expression data we've sacrificed a lot of our ability to identify HER2+ people while reducing our changes of misclassifying someone as HER2+. Here, again, I observe that the OOB errors increase in the test data so the same issues I noted after modeling DNA also apply here. In a real-world scenario, I would use the feature importance matrix to identify the genes that are most predictive and do further investigations. Later (see Section 6) I will construct an SVM model for this data.

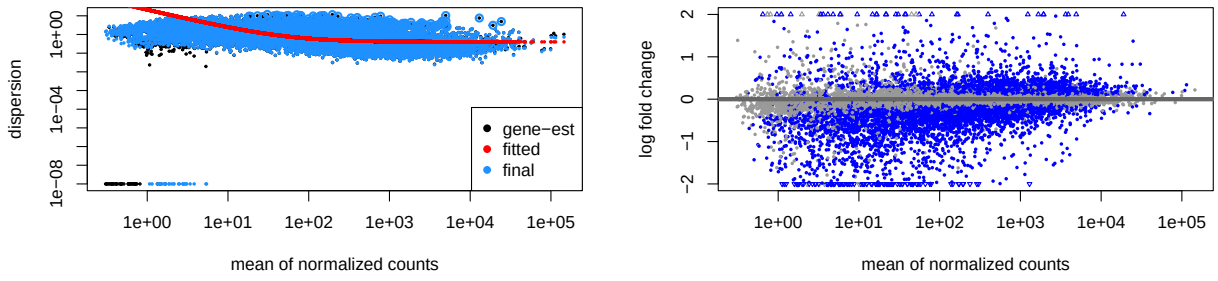
```
identical(rownames(t(r)), rownames(labels)) # Check order
n.test = ceiling(ncol(r) * 0.3) # Test set contains 194 obs (30%)
ind = sample(ncol(r), n.test, FALSE) # Randomly sample 194 rows from counts
r.train <- as.matrix(r[,-ind]) # Training data
l.train <- data.frame(HER2 = labels[-ind, "HER2"]) # Outcome

r.test <- as.matrix(r[,ind]) # Testing data
l.test <- data.frame(HER2 = labels[ind, "HER2"])

dseq.train <- DESeqDataSetFromMatrix(countData = r.train,
            colData = l.train, design = ~ HER2)
dseq.train <- DESeq(dseq.train, fitType = "local")
# T
dseq.test <- DESeqDataSetFromMatrix(countData = r.test,
            colData = l.test, design = ~ HER2)
dseq.test <- DESeq(dseq.test, fitType = "local")

# Takes too long to run
#rf <- classify(data = dseq.train, method = "rf",
#               preProcessing = "deseq-vst", ref = "T",
#               control = trainControl(method = "repeatedcv", number = 5,
#               repeats = 2, classProbs = TRUE))
```

I also tried to train a random model model (and a SVM model, not shown) from DESeq2 objects. I wanted to see if the model improved by leveraging DESeq2's negative binomial model. [4 I only recently found this method, and I haven't explored it much. In any case, this model took too long (>1 hr) to run so I abandoned it.

Section 6

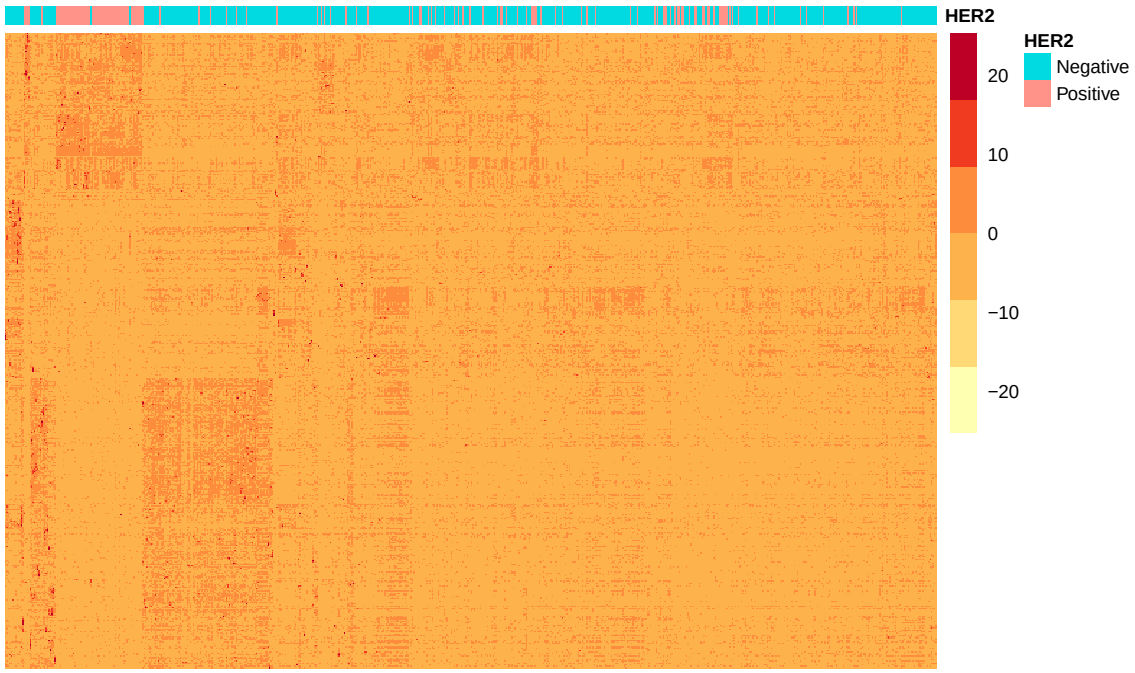
Differential expression analysis and an SVM model using the top DE genes

```
dds <- DESeq(dds) # Run DESeq2 (negative binomial model)
plotDispEsts(dds) # Dispersion estimate
resultsNames(dds) # Get comparison name
res <- results(dds, alpha=0.05,name="HER2_Positive_vs_Negative") # store results
res.lfc <- lfcShrink(dds, coef=2, res=res, type="apeglm") # Apply log2 shrinkage
plotMA(res.lfc, ylim=c(-2,2)) # Sig fold-change across most levels
summary(res.lfc) # Too many significant genes
# Reduce number of significant genes
padj.cutoff <- 0.05 # p val
lfc.cutoff <- 0.58 # fold change threshold of 1.5 (log2 fc of 0.58)
res.tb <- res %>% data.frame() %>%
  rownames_to_column(var="gene") %>%
  as_tibble()
# Get significantly expressed genes with new cutoffs
sig.e <- res.tb %>%
  dplyr::filter(padj < padj.cutoff & abs(log2FoldChange) > lfc.cutoff)
norm.counts <- counts(dds, normalized=TRUE) # Normalized counts (no VST)
norm.counts <- norm.counts %>%
     data.frame() %>%
     rownames_to_column(var="gene") %>%
     as_tibble()
```

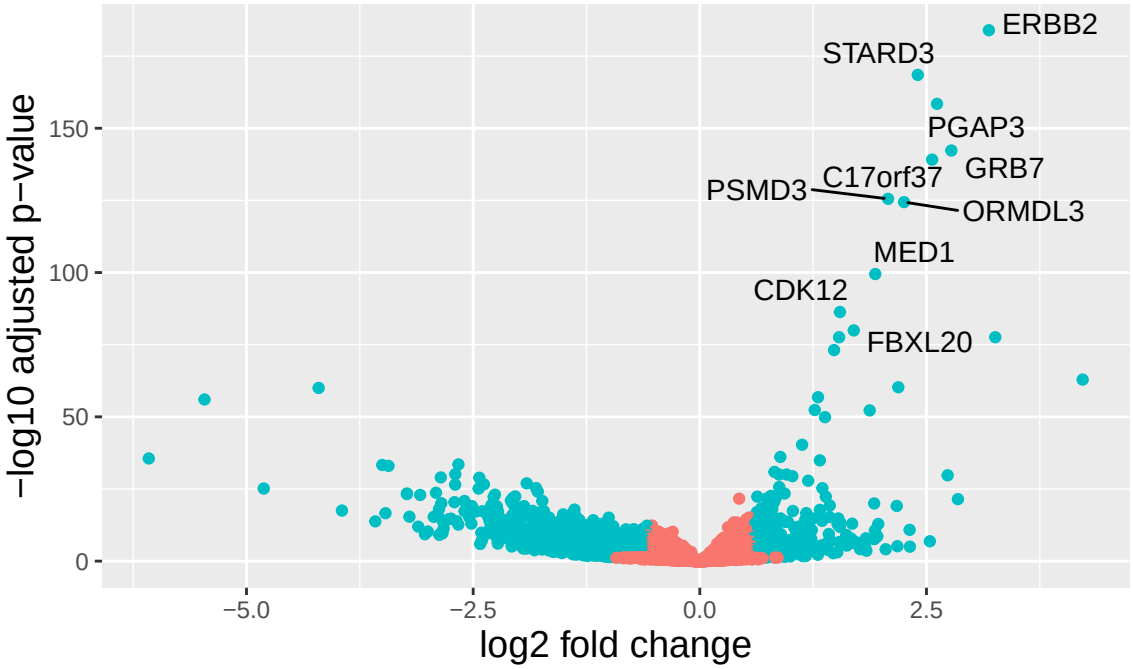


To evaluate the performance of DESeq2, I plotted the dispersion estimates (on left) and the log fold changes (MA plot, on right). The overwhelming majority of genes have low dispersion estimates and were shrunken towards the model across the whole range of means. In the MA plot, we see significant fold changes across the expression levels.

```r
top500 <- sig.e %>% arrange(padj) %>% pull(gene) %>% head(n=500)
# Note for self: dplyr and R different filter() functions are tricky here
norm.top500 <- norm.counts %>% filter(gene %in% top500)
# Get the normalized counts for the significant genes with new cutoffs
norm.sig.e <- norm.top500 %>%
     dplyr::filter(gene %in% sig.e$gene) %>%
     data.frame() %>%
     column_to_rownames(var = "gene")
# Get metadata to annotate heatmap
annotate <- labels %>% select(HER2)
# pheatmap wouldn't run without this; I manually checked the order b4 this
rownames(annotate) <- colnames(norm.sig.e)
heat_colors <- brewer.pal(6, "YlOrRd")
pheatmap(norm.sig.e,
          color = heat_colors, cluster_rows = T,
          treeheight_row = 0, treeheight_col = 0,
          show_rownames = F, show_colnames = F,
          annotation = annotate,
          border_color = NA, fontsize = 7,
          scale = "row", height = 15)
```



```
res.tb.tf <- res.tb %>%
              mutate(threshold_OE = padj < 0.05 & abs(log2FoldChange) >= 0.58)
res.tb.tf <- res.tb.tf %>% arrange(padj) %>% mutate(genelabels = "")
res.tb.tf$genelabels[1:10] <- res.tb.tf$gene[1:10]

ggplot(res.tb.tf, aes(x = log2FoldChange, y = -log10(padj))) +
        geom_point(aes(colour = threshold_OE)) +
        geom_text_repel(aes(label = genelabels)) +
        xlab("log2 fold change") +
        ylab("-log10 adjusted p-value") +
        theme(legend.position = "none",
          plot.title = element_text(size = rel(1.5), hjust = 0.5),
          axis.title = element_text(size = rel(1.25)))
```



I performed differential expression analysis using DESeq2 and extracted the most differently-expressed genes (fold change >1.5, adjusted p values < 0.05). This resulted in 1,857 genes. For curiosity, I visualized these results with a heatmap (using only the top 500 genes). In this heatmap, I can again see the tight cluster of HER2+ patients, but I also see a lot of HER2+ patients who cluster with HER2- patients. This confirms the result observed above, that in this dataset gene expression cannot reliably distinguish HER2 status. Given more time, I would like to also perform DE analysis on hormone receptor status since I saw clear clustering by hormone receptor in the PCA plots.

I also constructed a volcano plot just to get an overall picture of the expression by HER2 status, and I labeled the top 10 differentially-expressed genes. Not surprisingly, *ERBB1* is overexpressed in HER2+ patients. Also present are genes known to be involved in breast

cancer: *STARD3* (a gene involved in steroid hormone metabolism), *GRB7* (involved in interleukin signaling pathways), *MED1* (involved in chromatin binding and signaling receptor activity), and *MIEN1* (C17orf37, involved in selenium binding). There are other genes not involved in breast cancer as well, accoding to GeneCards entries. I don't know anything about what treatments these patients are taking and how that segregates by HER2+ status. Some of these differentially-expressed genes might be the result of specific treatments that activate certain metabolic pathways.

```
top.de.mat <- subset(rld.mat, select=colnames(rld.mat) %in% sig.e$gene)

identical(rownames(top.de.mat), rownames(labels)) # Check order
ind = sample(nrow(top.de.mat), n.test, FALSE) # Randomly sample 194 rows
r.train <- as.matrix(top.de.mat[-ind,]) # Training data
l.train <- data.frame(HER2 = labels[-ind, "HER2"]) # Outcome
r.test <- as.matrix(top.de.mat[ind,]) # Testing data
l.test <- data.frame(HER2 = labels[ind, "HER2"])
model <- randomForest(x = r.train, y = l.train[,1],
                      xtest = r.test, ytest = l.test[,1],
                      importance = TRUE, proximity = TRUE)
model

r <- cbind(r.train,l.train)
t <- cbind(r.test,l.test)
model <- svm(HER2 ~ ., data = r, kernel = "linear")
predictions <- predict(model, t)
confusionMatrix(predictions, t$HER2)
```

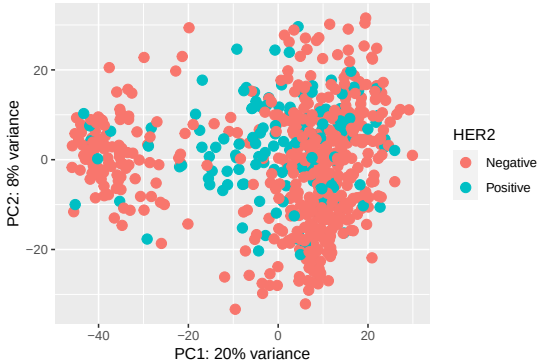
Finally, I wanted to try another classification model, SVM, on the data. I initially tried this on all genes but it was too many. Here, I ran the random forest model again on the subset of 1,857 most differentially-expressed genes as a point of comparison with the complete model of all genes. I wanted to see if I could speed up computation and improve predictive power by reducing the dataset size to only the most predictive genes. I also wanted to compare my results to another model.

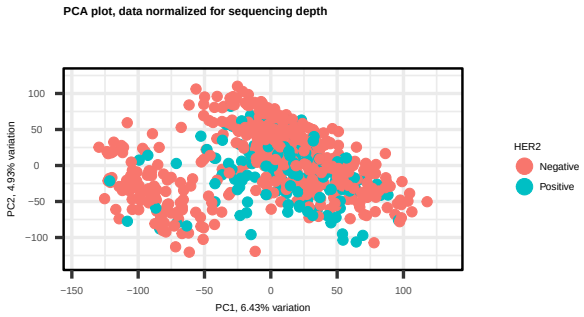
The random forest model with a subset of genes (about 10% of those in the initial RF model) had essentially the same overall accuracy (88% vs. 87%), and the same precision (92%) but slightly improved recall (53% vs. 46%). However, the same overfitting issues of course still exists. The SVM model performed with the same accuracy (88%) but interestingly this model was better at detecting HER2+ patients (70% recall). The improved recall comes at the expense of higher probability of misclassifying HER2- patients as HER2+ (75% precision).

Appendix

Checking clustering for different normalization methods

```
# DESeq2 method: normalize for sequencing depth and library composition
dds <- estimateSizeFactors(dds) # Normalize for seq depth
rld <- vst(dds, blind=TRUE) # Variance-stabilizing transform
pca.plot <- plotPCA(rld, intgroup="HER2", returnData=TRUE)
pv <- round(100 * attr(pca.plot, "percentVar"))
ggplot(pca.plot, aes(PC1, PC2, color=HER2)) +
  geom_point(size=3) +
  xlab(paste0("PC1: ", pv[1], "% variance")) +
  ylab(paste0("PC2: ", pv[2], "% variance")) +
  coord_fixed()
  ggtitle('PCA plot, data normalized for seq depth & lib composition') +
  theme(plot.title = element_text(size=8))

# Quick-and-dirty method: normalize for seq depth only
biplot(p, lab = NULL, colby = 'HER2',
       labSize = 7, axisLabSize = 7,
       legendLabSize = 7, legendTitleSize = 7,
       titleLabSize = 8,
       title = "PCA plot, data normalized for sequencing depth",
       legendPosition = 'right')
```





The PCA plots look different, but the clustering patterns look approximately the same. One important observation is that using DESeq2's normalization method, PC1 explains 20% of the variance vs. 6%, and PC2 explains 8% of the variance vs. 5%. So it's important to use DESeq2's normalization for downstream analyses.

References

Section: “The Genomics of Metastasis”
[1] Perou, C. M. & Børresen-Dale, A.-L. Systems Biology and Genomics of Breast Cancer. Cold Spring Harb Perspect Biol 3, a003293 (2011).

“These days, the span of estimates has shrunk — with most now between 19,000 and 22,000 — but there is still disagreement”
[2] Willyard, C. New human gene tally reignites debate. Nature 558, 354–355 (2018).

Paragraph begins: “The primary output of RSEM consists of two files. . . .”
[3] Li, B. & Dewey, C. N. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC Bioinformatics 12, 323 (2011).

[4] Anders, S. & Huber, W. Differential expression analysis for sequence count data. Genome Biology 11, R106 (2010).