Project Portion A

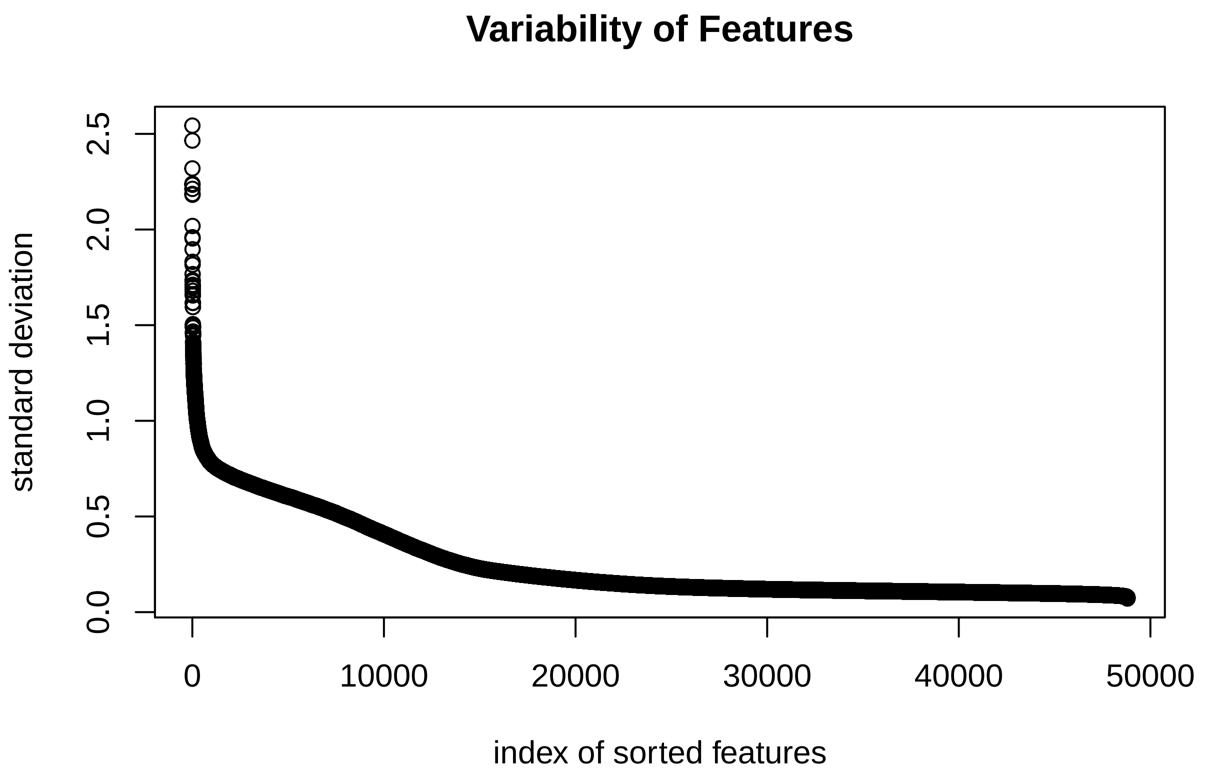
Daniel Kiser

Data

We used a data set (from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE33463) that provided gene expression profiles for 5 different classes of 140 individuals: 30 patients with idiopathic pulmonary arterial hypertension (IPAH); 19 patients with systemic sclerosis (SSc); 42 patients with SSC and IPAH (SSc-PAH), 8 patients with SSc, interstitial lung disease, and pulmonary hypertension (SSc-PH-ILD); and 41 individuals with no disease (controls). The number of gene expressions measured was 48803. Our goal was to find the statistical learning method that could most accurately predict the class of each individual. The methods we compared were Random Forest (RF), k-Nearest-Neighbors (kNN), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), LASSO, and Elastic Net (EN).

Exploratory Analysis

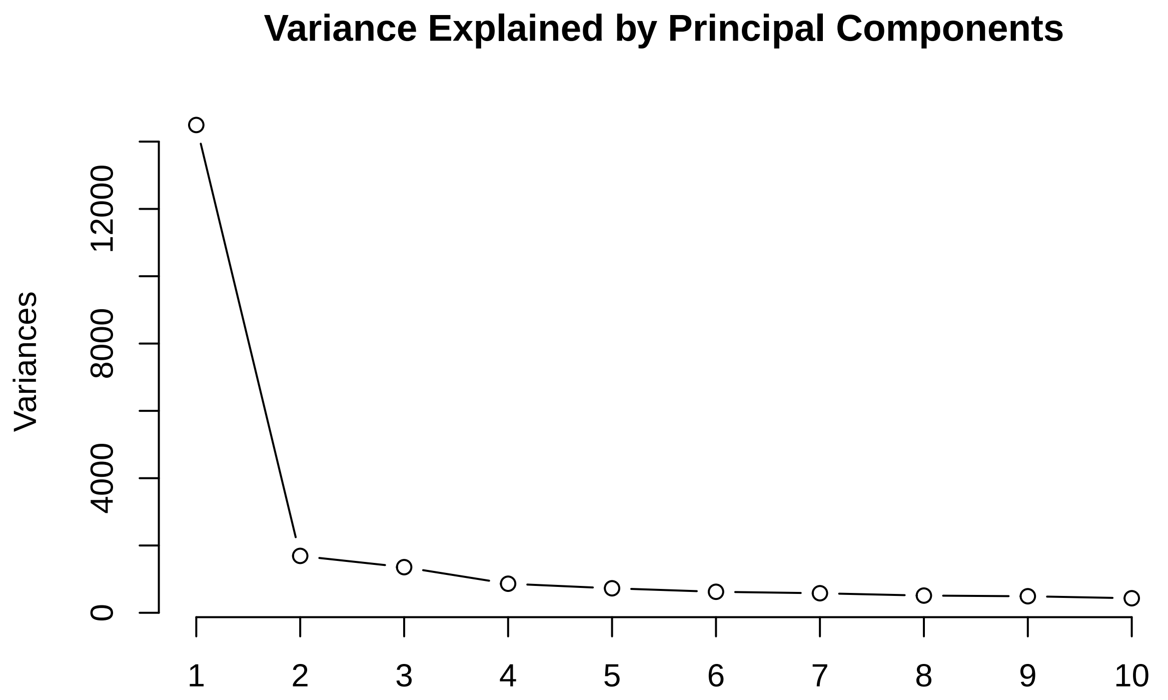
Because the dataset is so large (140 observation and 48803 features), we began our analysis by looking for ways to extract the most meaningful features. We sorted the features based on standard deviation and found that a significant portion of the features varied a very small amount across observations, and hence were unlikely to be important for distinguishing between disease states:



There appear to be two elbows in the above graph: one at about 50 features, and the other at about 15000 features. Since we lacked the computational power to analyze all 48803 features, we limited our prediction models to the 15000 most variable features.

Principle Component Analysis

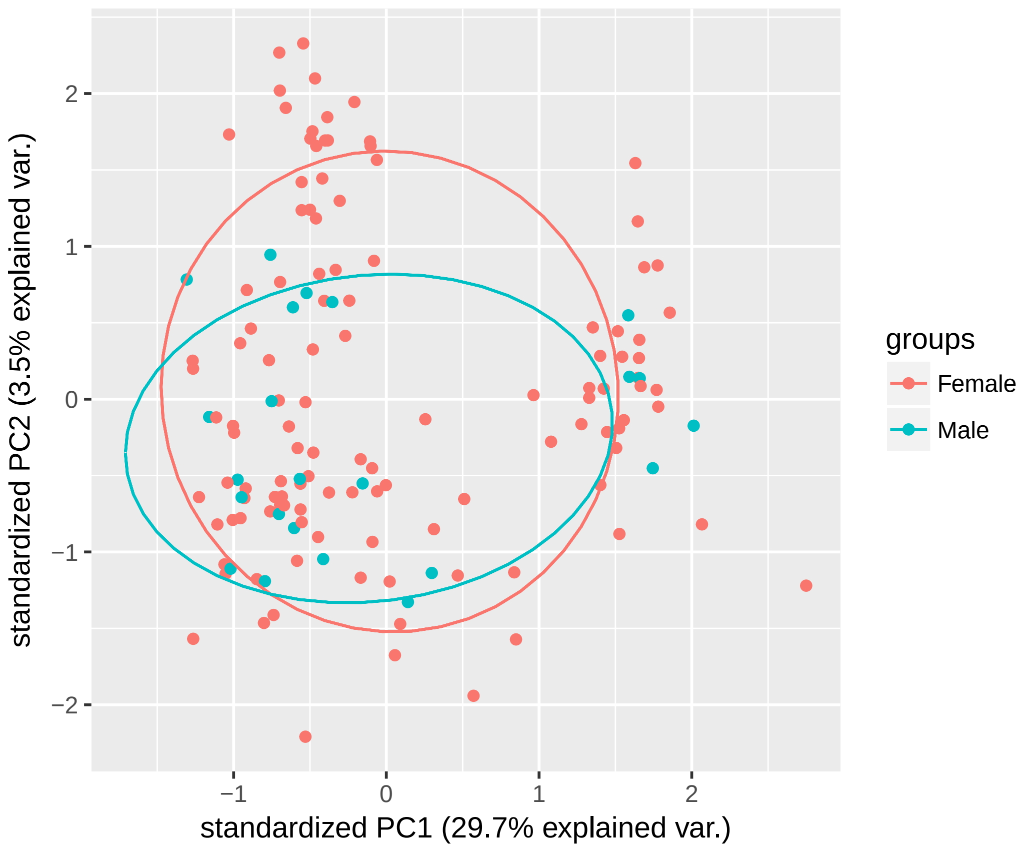
To gain an understanding of the sources of variability in the data, we standardized the data and performed Principle Component Analysis (PCA). Below is a plot showing the amount of variance accounted for by each principal component:

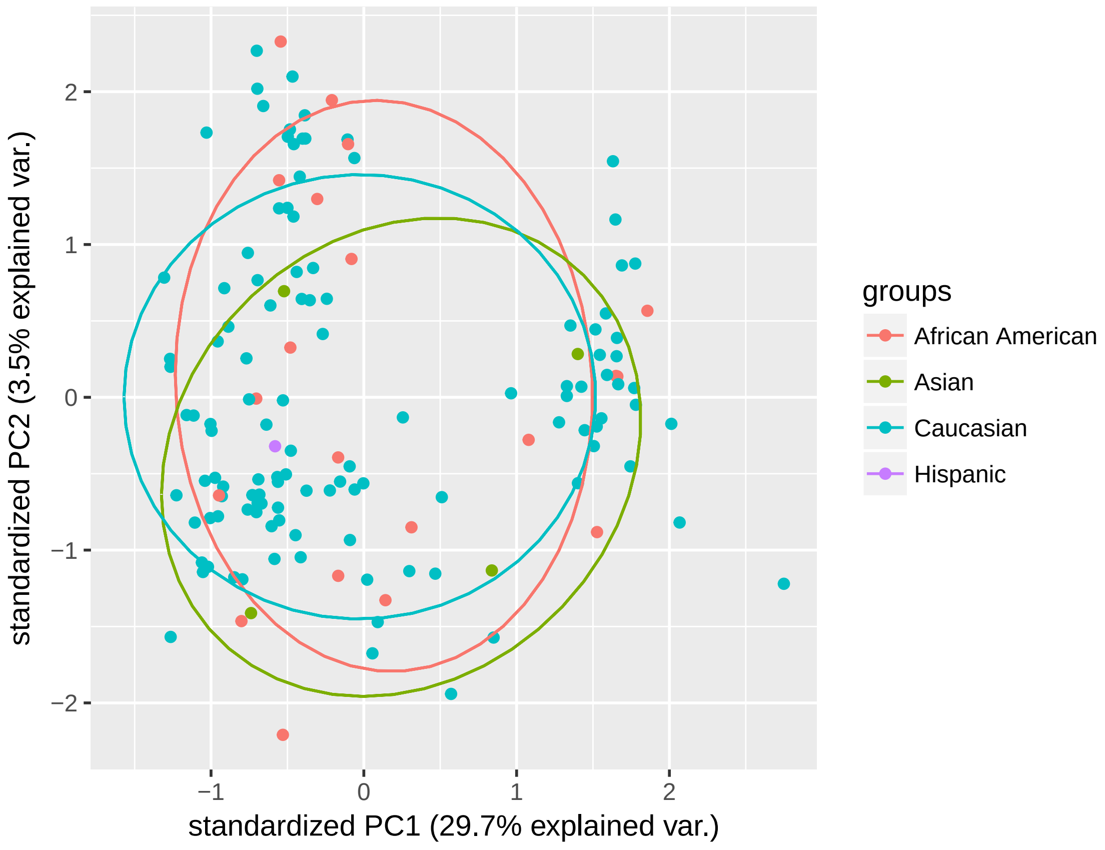


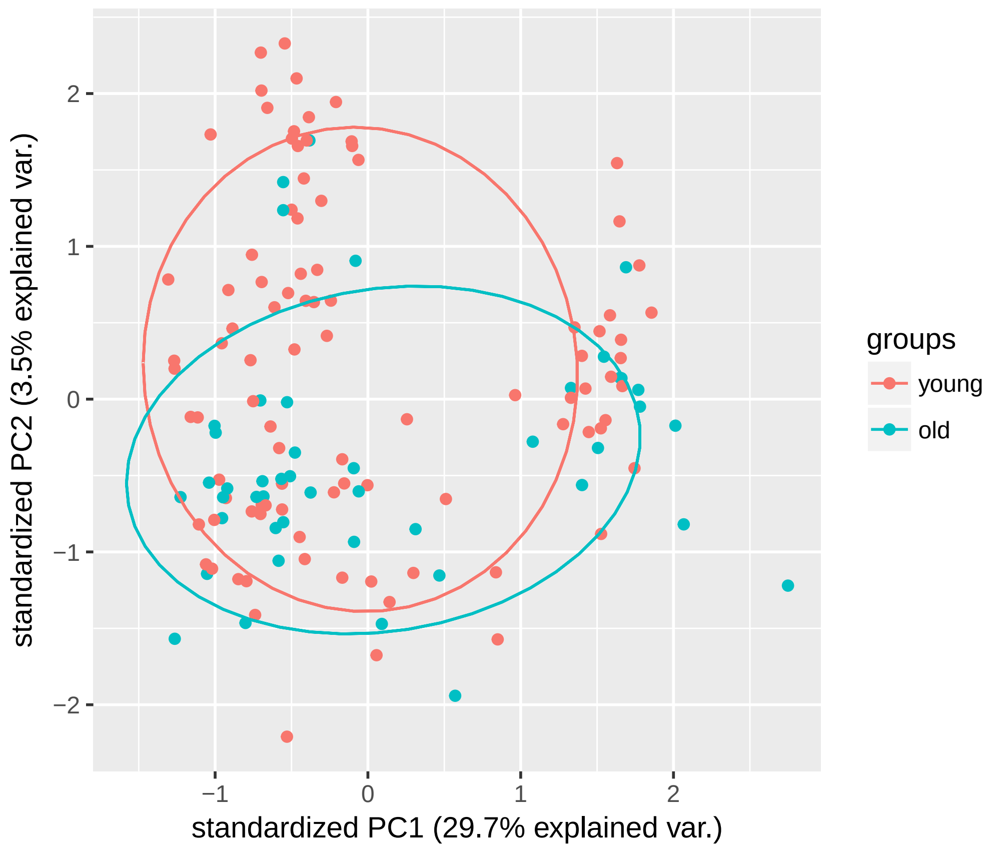
It is clear from the plot that the first principal component explains a much larger degree of variance than any of the other principal components. To better interpret the principal components, we examined the biplot of the first two principal components (with the variable vectors removed for clarity). When we grouped the observations by disease state, we saw that the second principal component contains much of the variance between diseased individuals and healthy individuals:

Healthy individuals (pink) are clearly separated from individuals with a disease. However, it is interesting to note that the first principal component appears to divide the observations into two distinct groups. In an effort to understand the source of variation in the first principal component, we pulled some demographic data associated with the orignal data set and grouped the observations by gender, race, and age (older or younger than 60 years).

As can be seen in the biplots below, none of the groupings seems to explain the existence of the two distinct clusters:


standardized PC2 (3.5% explained var.)
2
1
0
−1
−2
−1
0
1
2
standardized PC1 (29.7% explained var.)
groups
Female
Male


standardized PC2 (3.5% explained var.)
2
1
0
−1
−2
−1
0
1
2
standardized PC1 (29.7% explained var.)
groups
African American
Asian
Caucasian
Hispanic



Thus, it appears that the first principle component contains a large amount of variance that is unrelated to the disease state or other known characteristics of the individual observations. Possibly the variance is due to differences in data collection techniques or to experimental error, or there is some underlying biological factor that was unrecorded in the study. For the purposes of our analysis, where the goal is to predict disease state, the fact that the first principal component appears to contain no information about disease state indicates that we may not be able to rely on the features with the most variance to determine the class of each observation.

Random Forest

One of the methods we chose to compare was Random Forest (RF), since adaptations of RF are often used for feature selection in gene expression data sets. Since the principal components are expected to contain a large portion of the information of the original data set, we first attempted to train the RF on the principal components:

```
library(randomForest)
set.seed(1234)
class <- as.factor(c(rep("control", 41), rep("IPAH", 30), rep("SSc-PAH", 42), rep("SSc", 19),
            rep("SSc-PH-ILD", 8)))
d_pc <- cbind(class, pc$x)
d_pc <- as.data.frame(d_pc)
d_pc[,1] <- factor(d_pc[,1])
fit <- randomForest(class~., data = d_pc[,-1], importance=TRUE)
```

According to the model, the 10 most important principal components were:

```
##       PC2     PC113       PC6     PC115       PC7     PC127      PC96      PC32
## 8.825501 1.669935 1.244442 1.182075 1.105520 1.094060 1.090586 1.068732
##      PC8     PC64
## 1.031155 1.027961
```

As expected, the second principal component is important for predicting disease state while the first principal component is not used at all. In fact, many of the top-10 principal components contain a very small proportion of the variability in the original dataset. Below is the confusion matrix for this model:

```
##
## Call:
##  randomForest(formula = class ~ ., data = d_pc[, -1], importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
##         OOB estimate of  error rate: 45.71%
## Confusion matrix:
##            control IPAH SSc SSc-PAH SSc-PH-ILD class.error
## control         37    1   0       3          0  0.09756098
## IPAH             3    7   0      20          0  0.76666667
## SSc              4    6   0       9          0  1.00000000
## SSc-PAH          4    6   0      32          0  0.23809524
## SSc-PH-ILD       1    0   0       7          0  1.00000000
```

We note that the RF failed to predict any SSc or SSc-PH-ILD patients. The model seemed to primarily choose patients from the three largest groups, and it seems likely that 140 observations are two few to accurately predict 5 groups. Thus, we lumped the patients with some form of disease together into a “disease” group, so that the problem is now one of binary prediction.

```
class <- as.factor(c(rep("control", 41), rep("disease", 99)))
d_pc <- as.data.frame(cbind(rep(0, 140), pc$x))
d_pc[,1] <- class
fit <- randomForest(class~., data = d_pc[,-1], importance=TRUE)
fit
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = d_pc[, -1], importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
##         OOB estimate of  error rate: 20%
## Confusion matrix:
##         control disease class.error
## control      13      28   0.6829268
## disease       0      99   0.0000000
```

With only two classes to choose between, the model performed much better. However, it seemed to heavily favor the larger group (disease), wrongly predicting that 28 of the healthy patients had a disease but correctly identifying all of the disease patients. The order of the importance of the principal components changed somewhat:

```
##        PC2      PC113       PC96       PC64      PC117      PC111
## 10.0356566  2.2566525  0.8651949  0.7870079  0.7755052  0.6622651
##      PC133      PC112        PC6       PC18
##  0.6610371  0.6520989  0.6432009  0.6131921
```

We next tried to train the RF using the most variable features in the data set. Using only the first 100 most variable features to predict 5 classes, we obtained the following confusion matrix:

```
class <- as.factor(c(rep("control", 41), rep("IPAH", 30), rep("SSc-PAH", 42), rep("SSc", 19),
            rep("SSc-PH-ILD", 8)))
d[,1] <- class
M <- 100
index <- numeric(M)
for(i in 1:M) {
  index[i] <- match(max(stan_dev), stan_dev)
  stan_dev[index] <- 0
}
index <- index+1
d_lim <- d[,c(1, index)]
fit <- randomForest(class~., data = d_lim, importance=TRUE)
fit
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = d_lim, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 47.86%
## Confusion matrix:
##            control IPAH SSc SSc-PAH SSc-PH-ILD class.error
## control         36    2   2       1          0   0.1219512
## IPAH             2    8   3      17          0   0.7333333
## SSc              4    3   3       9          0   0.8421053
## SSc-PAH          4    9   3      26          0   0.3809524
## SSc-PH-ILD       1    1   0       6          0   1.0000000
```

The error rate we obtained is about the same as the error rate we obtained training the RF on the principal components. However, when we lumped the different disease states into one disease class, we obtained much better results than we did using principal components:

```
class <- as.factor(c(rep("control", 41), rep("disease", 99)))
d_lim[,1] <- class
fit <- randomForest(class~., data = d_lim, importance=TRUE)
fit
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = d_lim, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 11.43%
## Confusion matrix:
##         control disease class.error
## control      31      10  0.24390244
## disease       6      93  0.06060606
```

Thus, we concluded that a significant amount of information is lost if we train our model on the principal components rather than on the original features, even if they are only 100 of the most variable features. To determine if the addition of more features would increase the prediction accuracy, we attempted to predict 5 classes and 2 classes using 100, 1000, and 15000 features. For each set of classes and features, we trained an RF twenty times using different random seeds so that we could also assess the variability in the accuracy of the predictions (which we measured using out-of-bag (OOB) error).

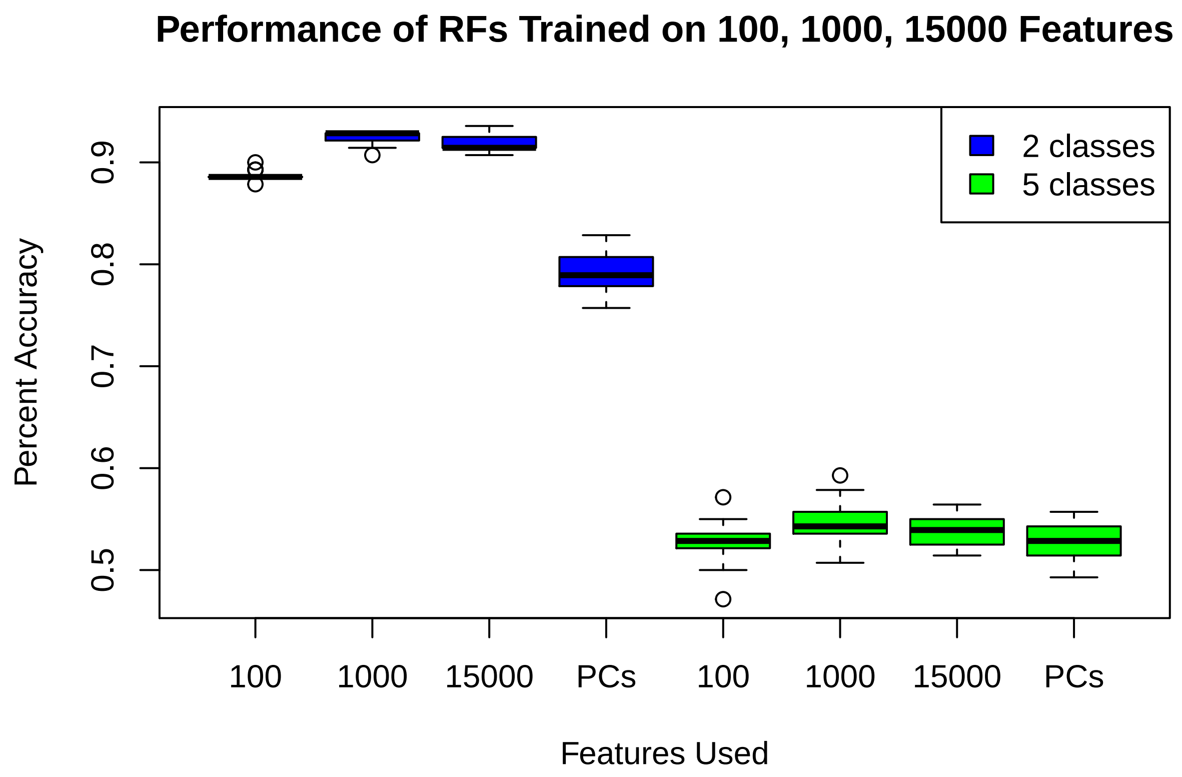
```r
n <- 20
k <- 1
acc <- numeric(160)
# 5 classes
class <- as.factor(c(rep("control", 41), rep("IPAH", 30), rep("SSc-PAH", 42), rep("SSc", 19),
            rep("SSc-PH-ILD", 8)))
d[,1] <- class
d_pc[,1] <- class

# 100 variables
stan_dev <- apply(d[,-1], 2, sd)
M <- 100
index <- numeric(M)
for(i in 1:M) {
  index[i] <- match(max(stan_dev), stan_dev)
  stan_dev[index] <- 0
}
index <- index+1
d_lim <- d[,c(1, index)]
for(i in 1:n) {
  fit <- randomForest(class~., data = d_lim)
  acc[k] <- 1 - fit$err.rate[500,1]
  k = k + 1
}

#1000 variables
stan_dev <- apply(d[,-1], 2, sd)
M <- 1000
index <- numeric(M)
for(i in 1:M) {
  index[i] <- match(max(stan_dev), stan_dev)
  stan_dev[index] <- 0
}
index <- index+1
d_lim <- d[,c(1, index)]

for(i in 1:n) {
  fit <- randomForest(class~., data = d_lim)
  acc[k] <- 1 - fit$err.rate[500,1]
  k <- k + 1
}

#15000 variables - taken from job output on cluster (1:38 run time)

# code used on cluster commented out:

#library(randomForest)
#set.seed(1234)
#d <- read.table("/home/dskiser/GDS5499_labeled.csv", sep = ",", header = T)
```

```
#class <- as.factor(c(rep("control", 41), rep("IPAH", 30), rep("SSc-PAH", 42), rep("SSc", 19),
#             rep("SSc-PH-ILD", 8)))
#d[,1] <- class
#stan_dev <- apply(d[,-1], 2, sd)

#M <- 15000
#index <- numeric(M)
#for(i in 1:M) {
#  index[i] <- match(max(stan_dev), stan_dev)
#  stan_dev[index] <- 0
#}
#d_lim <- d[,c(1, index)]

#n <- 20
#k <- 1
#acc <- numeric(n)
#for(i in 1:n) {
#  fit <- randomForest(class~., data = d_lim, importance=TRUE)
#  acc[k] <- 1 - fit$err.rate[500,1]
#  cat(acc[k], "\n")
#  k <- k + 1
#}

acc[k:(k+19)] <- c(0.5571429,0.5428571,0.5571429,0.55,0.5428571,0.5214286,0.5642857,0.5285714,
                   0.55,0.5428571,0.5285714,0.5214286,0.5142857,0.5285714,0.5285714,0.5642857,
                   0.5142857,0.5357143,0.55,0.5214286)

k <- k + 20

# PCA
for(i in 1:n) {
  fit <- randomForest(class~., data = d_pc[,-1])
  acc[k] <- 1 - fit$err.rate[500,1]
  k = k + 1
}


# 2 classes

class <- as.factor(c(rep("control", 41), rep("disease", 99)))
d[,1] <- class
d_pc[,1] <- class

# 100 variables
stan_dev <- apply(d[,-1], 2, sd)
M <- 100
index <- numeric(M)
for(i in 1:M) {
  index[i] <- match(max(stan_dev), stan_dev)
  stan_dev[index] <- 0
}
index <- index+1
d_lim <- d[,c(1, index)]
```

```r
for(i in 1:n) {
  fit <- randomForest(class~., data = d_lim)
  acc[k] <- 1 - fit$err.rate[500,1]
  k = k + 1
}

#1000 variables
stan_dev <- apply(d[,-1], 2, sd)
M <- 1000
index <- numeric(M)
for(i in 1:M) {
  index[i] <- match(max(stan_dev), stan_dev)
  stan_dev[index] <- 0
}
index <- index+1
d_lim <- d[,c(1, index)]

for(i in 1:n) {
  fit <- randomForest(class~., data = d_lim)
  acc[k] <- 1 - fit$err.rate[500,1]
  k <- k + 1
}

#15000 variables - taken from job output on cluster (1:41 run time)

# code used on cluster commented out:

#library(randomForest)
#set.seed(1234)
#d <- read.table("/home/dskiser/GDS5499_labeled.csv", sep = ",", header = T)
#class <- as.factor(c(rep("control", 41), rep("disease", 99)))
#d[,1] <- class

#stan_dev <- apply(d[,-1], 2, sd)

#M <- 15000
#index <- numeric(M)
#for(i in 1:M) {
#  index[i] <- match(max(stan_dev), stan_dev)
#  stan_dev[index] <- 0
#}
#d_lim <- d[,c(1, index)]


#n <- 20
#acc <- numeric(n)
#k <- 1
#for(i in 1:n) {
#  fit <- randomForest(class~., data = d_lim, importance=TRUE)
#  acc[k] <- 1 - fit$err.rate[500,1]
#  cat(acc[k], "\n")
#  k <- k + 1
#}
```

```r
acc[k:(k+19)] <- c(0.9142857,0.9142857,0.9071429,0.9142857,0.9357143,0.9142857,0.9142857,0.9142857,
                   0.9214286,0.9214286,0.9285714,0.9142857,0.9285714,0.9357143,0.9142857,0.9142857,
                   0.9142857,0.9142857,0.9142857,0.9285714)
k <- k + 20

# PCA
for(i in 1:n) {
  fit <- randomForest(class~., data = d_pc[,-1])
  acc[k] <- 1 - fit$err.rate[500,1]
  k = k + 1
}
```

The boxplot below compares our results:



The mean accuracy of the best performing RF model (trained on 1000 features):

```r
mean(acc[101:120])
```

```
## [1] 0.9239286
```

The prediction accuracy improved with the addition of more features only up to 1000 features. When 15000 features were used, the accuracy remained about the same as at 1000 features, or even decreased slightly. Due to computatational limitations, we assume that this trend would hold even if the entire data set were used. Thus, it seems reasonable to use the subset of the data that varies the most for the purposes of prediction. However, too small of a subset (100 features) yields slightly worst predictions, which supports our conclusions from the PCA that the features with the highest variance will not necessarily be the best predictors.

As we noted before, the accuracy of the 2-class predictions when the RF is trained using principal components is markedly lower than the accuracy of the predictions obtained using the original features, even if only 100 of the original features are used. This is interesting, since there are a greater number of principal components (140) than original features, and one would think that the principal components would contain more information. However, it seems that PCA is obscuring the relationships between variables, and thus negatively affecting the predictions of the RF.

Of course, the most obvious differences in the above plot are between the 2-class prediction accuracies and the 5-class prediction accuracies. Not only are the 5-class prediction accuracies much lower, but they also appear to be more variable. It seems that the number of observations available are inadequate for predicting 5-classes using RF, but are sufficient for predicting 2-classes with reasonable accuracy. Of course, even with 5 classes the RF performed better than random guessing, since with 5 classes random guessing would produce an accuracy of about 20%.