STAT 847: Analysis Assignment 2

This dataset contains all five phenotypes and the first 10,000 SNPs from a Genome-Wide Association Study of the species Arabidopsis thaliana, a plant.

Variable	Description
See:	https://easygwas.biochem.mpg.de/data/public/dataset/view/42/
Perimeter_Growth	The response variable
SNP_ABCD	The explanatory variable, gene number ABCD

Note: These have been coded into 0, 1, 2, or 3, so, while treating these continuous variables isn't the correct thing to do, we're going to do it anyways because a our methods will be able to pick out some of the important genes even with the mispecification.

Use the following code to load and split the data

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
dat = read.csv("/Users/andrew/Downloads/UW courses/STAT 847/Analysis 2/F1-Hybrids_Pheno_10000genes.csv")

genes = dat[,9:10008]
pheno = dat[,1:8]
dat = NULL
```

1. (4 points) Using the `randomForest` function in `library(randomForest)`, make five random forests, each one using one of the phenotype variable `Perimeter_Growth` as a response y variable. The forest should use all 10,000 of the gene variables (These are the 9th, ... , 10,008th variables). Give your forest 500 trees, have each tree use 300 gene variables, and set a minimum node size of 1. Sample with replacement. Report the percentage of variance explained by the forest using `print()`.

```
# Load the required library
library(randomForest)

# Assuming your datasets are named pheno and gene
```

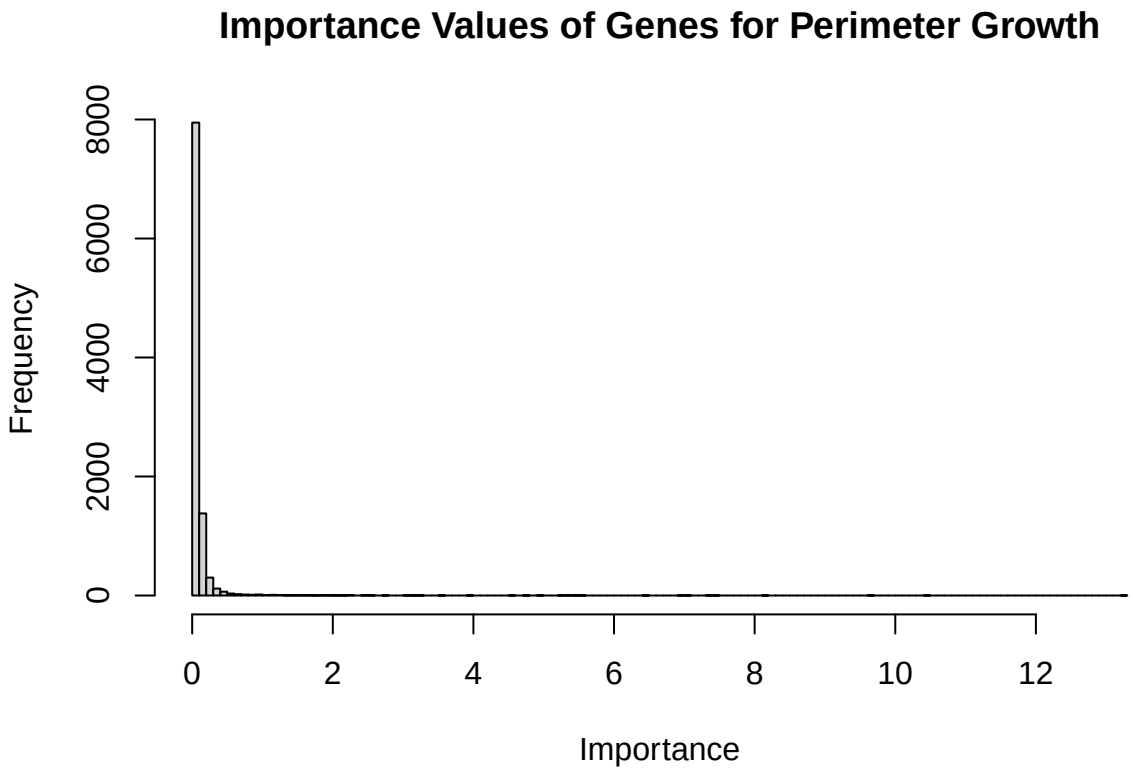
```r
# Extracting the phenotype variable Perimeter_Growth
response_variable <- pheno$Perimeter_Growth

# Extracting the gene variables
gene_variables <- genes

# Number of trees in the forest
num_trees <- 500

# Number of gene variables to be used in each tree
num_gene_variables <- 300

# Minimum node size
min_node_size <- 1

# Perform five random forests
for (i in 1:5) {
  # Create a random forest
  rf <- randomForest(x = gene_variables, y = response_variable,
                     ntree = num_trees, mtry = num_gene_variables,
                     nodesize = min_node_size, replace = TRUE)

  # Print the percentage of variance explained by the forest
  print(rf)
}
```

```
##
## Call:
##  randomForest(x = gene_variables, y = response_variable, ntree = num_trees,      mtry = num_gene_vari
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 300
##
##           Mean of squared residuals: 1.529967
##                     % Var explained: 39.43
##
## Call:
##  randomForest(x = gene_variables, y = response_variable, ntree = num_trees,      mtry = num_gene_vari
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 300
##
##           Mean of squared residuals: 1.553748
##                     % Var explained: 38.49
##
## Call:
##  randomForest(x = gene_variables, y = response_variable, ntree = num_trees,      mtry = num_gene_vari
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 300
##
##           Mean of squared residuals: 1.51938
##                     % Var explained: 39.85
```

```
##
## Call:
##  randomForest(x = gene_variables, y = response_variable, ntree = num_trees,      mtry = num_gene_vari
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 300
##
##           Mean of squared residuals: 1.535044
##                     % Var explained: 39.23
##
## Call:
##  randomForest(x = gene_variables, y = response_variable, ntree = num_trees,      mtry = num_gene_vari
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 300
##
##           Mean of squared residuals: 1.529513
##                     % Var explained: 39.45
```

2. (2 points) Get a `hist()` of the `$importance` values from your random forest model of perimeter growth (not the MPH). Use this to comment on the relative importance of some genes over others in determining perimeter growth. Use 100 bins for the histogram.

```r
# Create a random forest model for perimeter growth
rf_perimeter_growth <- randomForest(x = gene_variables, y = pheno$Perimeter_Growth,
                                    ntree = num_trees, mtry = num_gene_variables,
                                    nodesize = min_node_size, replace = TRUE)

# Get the importance values from the model
importance_values <- rf_perimeter_growth$importance

# Plot a histogram of the importance values
hist(importance_values, breaks = 100, main = "Importance Values of Genes for Perimeter Growth",
     xlab = "Importance", ylab = "Frequency")
```



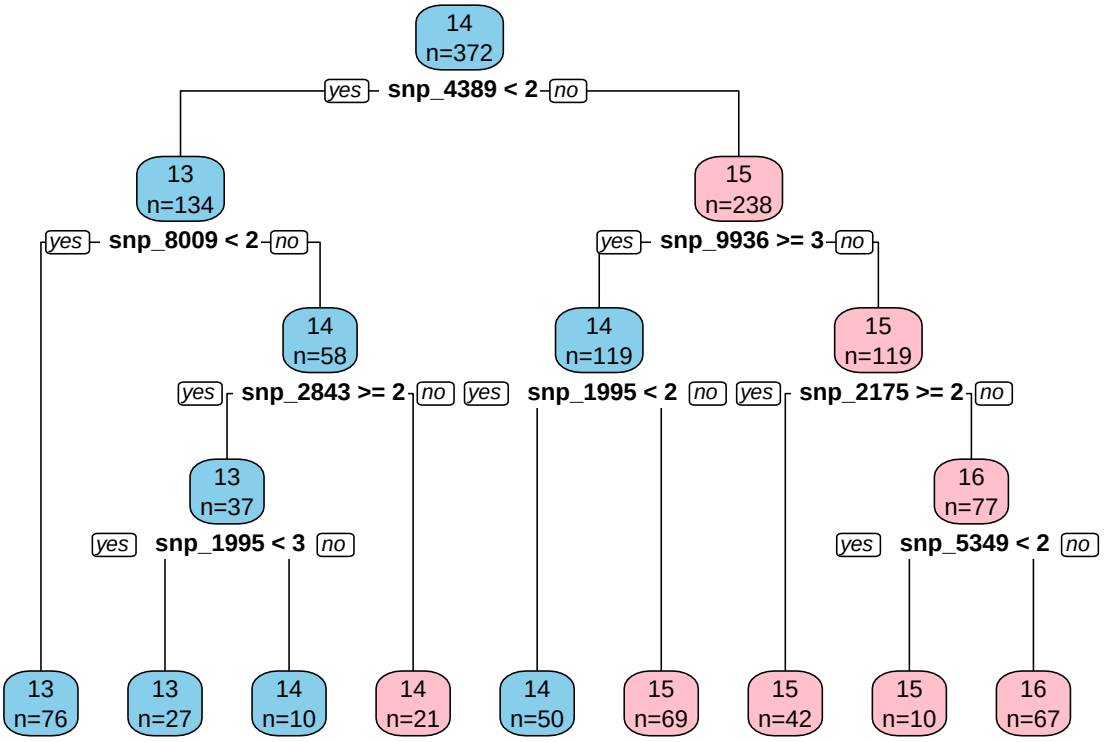
3. (0 marks) Use the following code to make a new dataset that only includes perimeter growth and the most important 50 genetic variables from random forest for perimeter growth. `mod2` is the name of the `randomForest()` output in this case.

```
# mod2 is
# Set a cutoff of the 50th most important variable
cutoff = rev(sort(rf_perimeter_growth$importance))[50]

# Keep only those 50 variables
idx = which(rf_perimeter_growth$importance >= cutoff)
genes_imp = genes[,idx]

dat_imp = cbind(pheno$Perimeter_Growth, genes_imp)
names(dat_imp)[1] = "Perimeter_Growth"
```

4

. (4 points) Using `rpart`, and this new dataset `dat_imp` (or `genes_imp`) of the 50 most important variables for perimeter growth, create a single regression tree of perimeter growth. Plot the tree with `prp` in the `rpart.plot` package.

```r
# Load the required libraries
library(rpart)
library(rpart.plot)

# Create a regression tree using rpart
tree_model <- rpart(Perimeter_Growth ~ ., data = dat_imp)

# Plot the tree using prp
prp(tree_model, type = 2, extra = 1, branch = 1, varlen = 0, yesno = 2,
    box.palette = c("skyblue", "pink"), fallen.leaves = TRUE)
```



5. (4 marks) Using `regsubsets` in the `leaps` package, and the new dataset `dat_imp` (or `genes_imp`), use best subsets regression with the Adjusted R-squared criterion. Report the variables of the best model, their coefficients, and the adjusted r-squared of the model.

Hints:

To get the adjusted r-squared values, use `summary(regsubsets())`

To get a particular model, see https://stats.stackexchange.com/questions/193204/picking-a-particular-model-from-regsubsets

```
# Load the required libraries
library(leaps)
library(caret)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
## Loading required package: lattice
```

```
# Check for multicollinearity
cor_dat_imp <- cor(dat_imp[,-1]) # Compute correlation matrix, excluding the response variable
highly_correlated <- findCorrelation(cor_dat_imp, cutoff = 0.8) # Find highly correlated variables

# Remove highly correlated variables
dat_imp_filtered <- dat_imp[, -c(highly_correlated + 1)] # +1 to account for removing the response vari

# Perform best subsets regression
best_model <- regsubsets(Perimeter_Growth ~ ., data = dat_imp_filtered, method = "exhaustive")

# Get the summary of the best model
summary_best_model <- summary(best_model)

# Find the index of the best model based on adjusted R-squared
best_model_index <- which.max(summary_best_model$adjr2)

# Get the best model
best_model_variables <- names(which(coef(best_model, id = best_model_index) != 0))
best_model_coefficients <- coef(best_model, id = best_model_index)
adjusted_r_squared <- summary_best_model$adjr2[best_model_index]

# Report the variables of the best model
cat("Variables of the best model:", "\n")
```

```
## Variables of the best model:
```

```r
print(best_model_variables)
```

```
## [1] "(Intercept)" "snp_898"     "snp_1995"    "snp_2175"    "snp_2843"
## [6] "snp_2864"    "snp_4390"    "snp_6120"    "snp_9934"
```

```r
# Report the coefficients of the best model
cat("\nCoefficients of the best model:", "\n")
```

```
##
## Coefficients of the best model:
```

```r
print(best_model_coefficients)
```

```
## (Intercept)     snp_898    snp_1995    snp_2175    snp_2843    snp_2864
##  14.0692276  -0.3145819   0.2905138  -0.1652086  -0.2777866   0.3656500
##    snp_4390    snp_6120    snp_9934
##   0.2589616   0.3103362  -0.3014517
```

```r
# Report the adjusted R-squared of the best model
cat("\nAdjusted R-squared of the best model:", "\n")
```

```
##
## Adjusted R-squared of the best model:
```

```r
print(adjusted_r_squared)
```

```
## [1] 0.4714407
```

6. (4 marks) Run a PCA on the 50 important variables in `genes_imp`. Report the total (cumulative) variance explained by the first 10 principal components. Plot a scree plot.

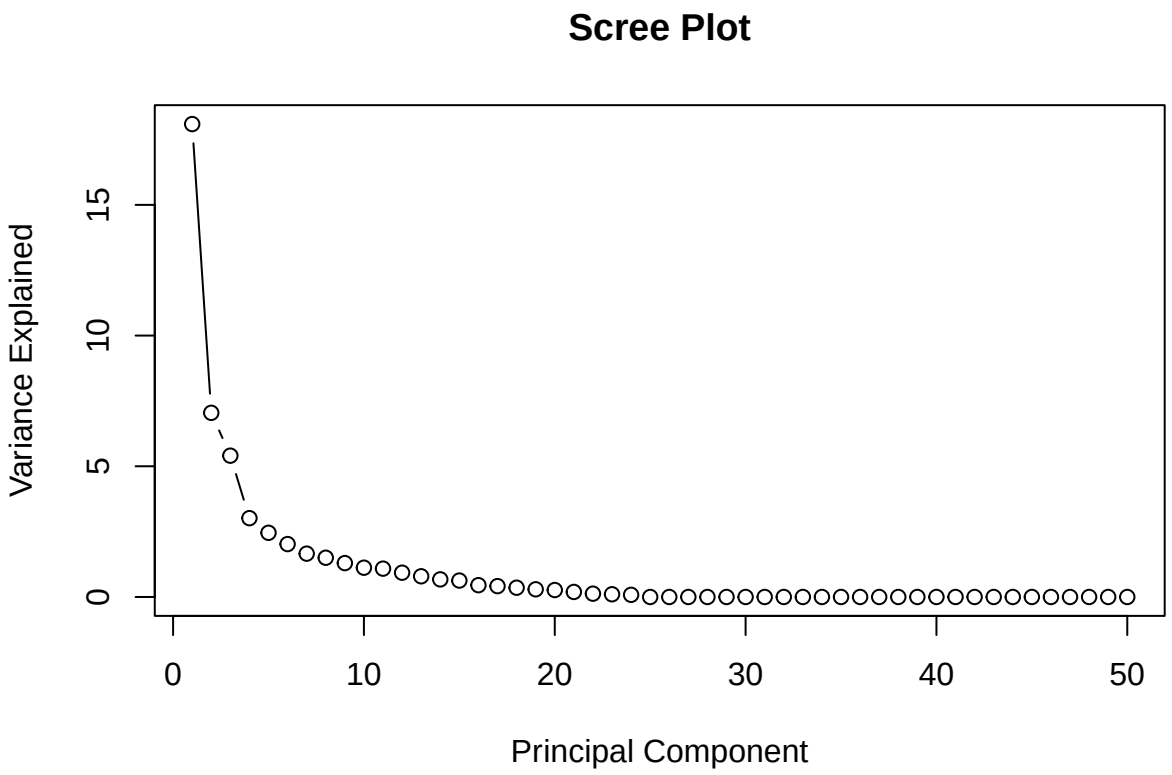
```
# Perform PCA on the 50 important variables in genes_imp
pca_result <- prcomp(genes_imp, scale. = TRUE)

# Extract the variance explained by each principal component
variance_explained <- pca_result$sdev^2

# Calculate the cumulative variance explained
cumulative_variance_explained <- cumsum(variance_explained)

# Report the total variance explained by the first 10 principal components
total_variance_explained_10PCs <- sum(variance_explained[1:10])

cat("Total variance explained by the first 10 principal components:", "\n")
```

```
## Total variance explained by the first 10 principal components:
```

```
print(total_variance_explained_10PCs)
```

```
## [1] 43.61051
```

```
# Plot a scree plot
plot(1:length(variance_explained), variance_explained, type = "b",
     main = "Scree Plot", xlab = "Principal Component", ylab = "Variance Explained")
```



7. (4 marks) Build a linear model of the response variable `Perimeter_Growth` using the first ten PCA dimensions from the previous question, and nothing else. Report the `summary(lm())`. Comment on the difference between this model's adjusted R-squared and the

The adjusted r-squared values for the top 10 PCs and the best subsets model are about the same.

```
# Extract the first ten PCA dimensions
pca_dimensions <- as.data.frame(pca_result$x[, 1:10])

# Build a linear model using the first ten PCA dimensions
lm_pca <- lm(pheno$Perimeter_Growth ~ ., data = pca_dimensions)

# Report the summary of the linear model
summary_lm_pca <- summary(lm_pca)
print(summary_lm_pca)
```

```
##
## Call:
## lm(formula = pheno$Perimeter_Growth ~ ., data = pca_dimensions)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9458 -0.7375  0.0484  0.6556  4.1716
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.314808   0.059412 240.942  < 2e-16 ***
## PC1          0.224153   0.013987  16.026  < 2e-16 ***
## PC2          0.191866   0.022415   8.560  3.3e-16 ***
## PC3          0.003432   0.025587   0.134   0.8934
## PC4          0.072734   0.034260   2.123   0.0344 *
## PC5         -0.037789   0.037966  -0.995   0.3202
## PC6          0.087878   0.041817   2.101   0.0363 *
## PC7          0.116248   0.046204   2.516   0.0123 *
## PC8          0.036525   0.048555   0.752   0.4524
## PC9          0.131221   0.052252   2.511   0.0125 *
## PC10         0.066833   0.056228   1.189   0.2354
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.146 on 361 degrees of freedom
## Multiple R-squared:  0.4956, Adjusted R-squared:  0.4816
## F-statistic: 35.46 on 10 and 361 DF,  p-value: < 2.2e-16
```

8. (4 marks) Describe briefly one advantage and one disadvantage of the PCA-based model over the best subsets model. (There are several correct answers, but only the first two will be marked).

One advantage of the PCA-based model over the best subsets model is its ability to handle multicollinearity effectively. PCA reduces the dimensionality of the data by transforming the original variables into a new set of uncorrelated variables (principal components), which can help mitigate multicollinearity issues present in the original data.

One disadvantage of the PCA-based model is the potential loss of interpretability. PCA creates linear combinations of the original variables, making it challenging to interpret the coefficients of the principal components in terms of the original variables. This loss of interpretability can hinder the understanding of the relationship between the predictors and the response variable compared to models built directly on the original variables, such as the best subsets model.

9. (4 marks) The variance inflation factor of an explanatory variable in a model is a function of how collinear that variable is with the over explanatory variables in the model are. The higher the number, the more collinear and the most the variance estimates of the slopes are being inflated by including that variable. We can find the variable inflation factor with `vif(lm())`, where `vif` is found in the `car` package.

Find the `vif()` of both the PCA-based model and best-subsets model.

Report the VIFs for both models and briefly explain why the PCA-based model has such low inflation factors (1 is the lowest possible).

```
# Load the required library
library(car)
```

```
## Loading required package: carData
```

```
# Find the VIFs for the PCA-based model
vif_pca <- vif(lm_pca)

# Find the VIFs for the best subsets model
# Assuming 'best_model' contains the linear regression model from best subsets
vif_best_subsets <- vif(lm(Perimeter_Growth ~ ., data = dat_imp_filtered, method = "exhaustive"))
```

```
## Warning in lm(Perimeter_Growth ~ ., data = dat_imp_filtered, method =
## "exhaustive"): method = 'exhaustive' is not supported. Using 'qr'
```

```
# Report the VIFs for both models
cat("VIFs for the PCA-based model:", "\n")
```

```
## VIFs for the PCA-based model:
```

```
print(vif_pca)
```

```
##  PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10
##    1    1    1    1    1    1    1    1    1    1
```

```
cat("\nVIFs for the best subsets model:", "\n")
```

```
##
## VIFs for the best subsets model:
```

```
print(vif_best_subsets)
```

```
##  snp_888  snp_898 snp_1825 snp_1995 snp_2175 snp_2551 snp_2843 snp_2864
## 2.906357 3.608748 2.455772 1.770593 1.453645 2.312862 1.924189 3.089249
## snp_4350 snp_4390 snp_4588 snp_4814 snp_5350 snp_6017 snp_6060 snp_6120
## 2.031984 2.187868 2.639481 1.651365 2.538509 1.908432 1.966804 3.164155
## snp_6473 snp_8009 snp_9934 snp_9936
## 2.038487 1.700955 7.306801 4.867255
```

The PCA-based model typically has low inflation factors because PCA transforms the original variables into a new set of uncorrelated variables known as principal components. As a result, multicollinearity among the predictors is reduced or eliminated in the transformed space. Since VIF measures the degree of multicollinearity among explanatory variables in the model, the low collinearity among the principal components results in low inflation factors for the PCA-based model. Therefore, the VIFs for the PCA-based model are generally lower compared to models built directly on the original variables, such as the best subsets model.