Homework 1

Austin Vanderlyn ajl745

6/24/2022

Exercise 3.1

The UC Irvine Machine Learning Repository6 contains a data set related to glass identification. The data consist of 214 glass samples labeled as one of seven class categories. There are nine predictors, including the refractive index and percentages of eight elements: Na, Mg, Al, Si, K, Ca, Ba, and Fe.

(a) Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors. Access data;

```
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 4.1.3
```

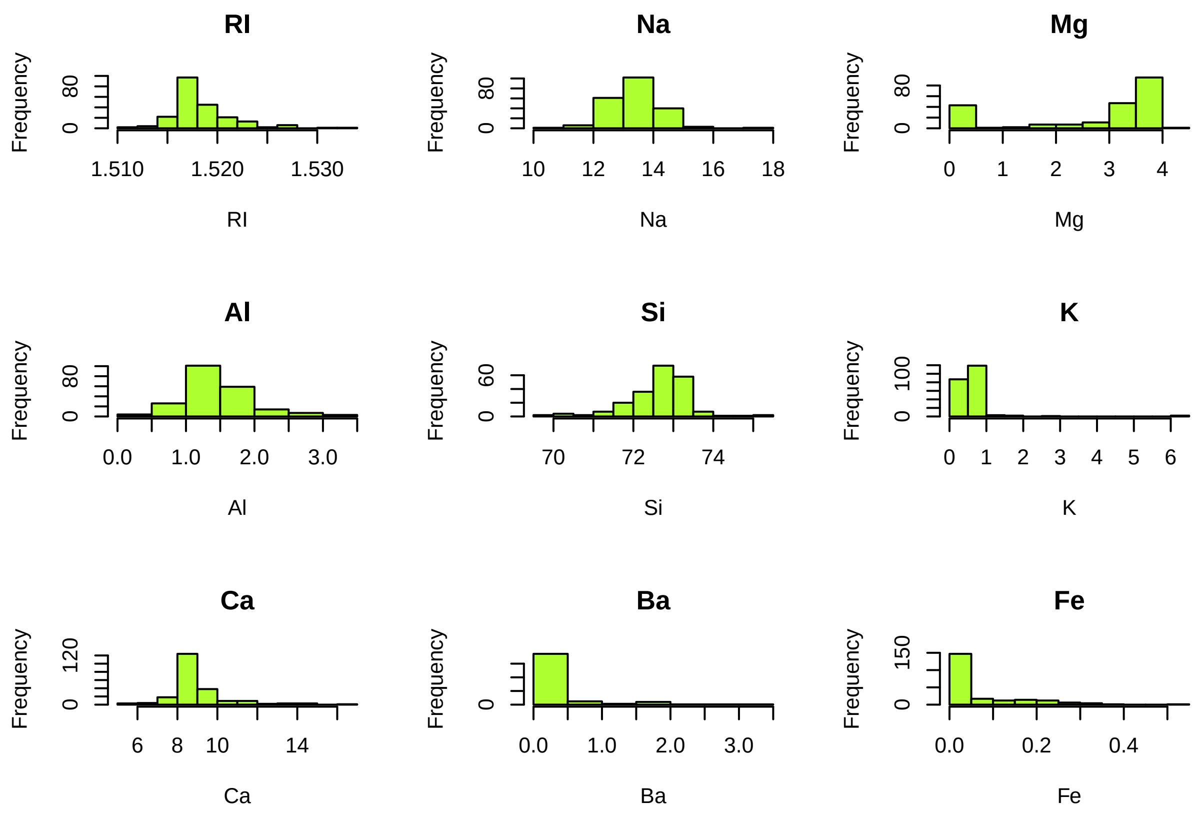
```
data(Glass)
```

Create data set without "Type" column;

```
X = Glass[, -10]
```

For loop of variable histograms;

```
par(mfrow = c(3,3))
for (i in 1:ncol(X)) {
  hist(X[ ,i], xlab = names(X[i]), main = paste(names(X[i])), col = "greenyellow")
}
```

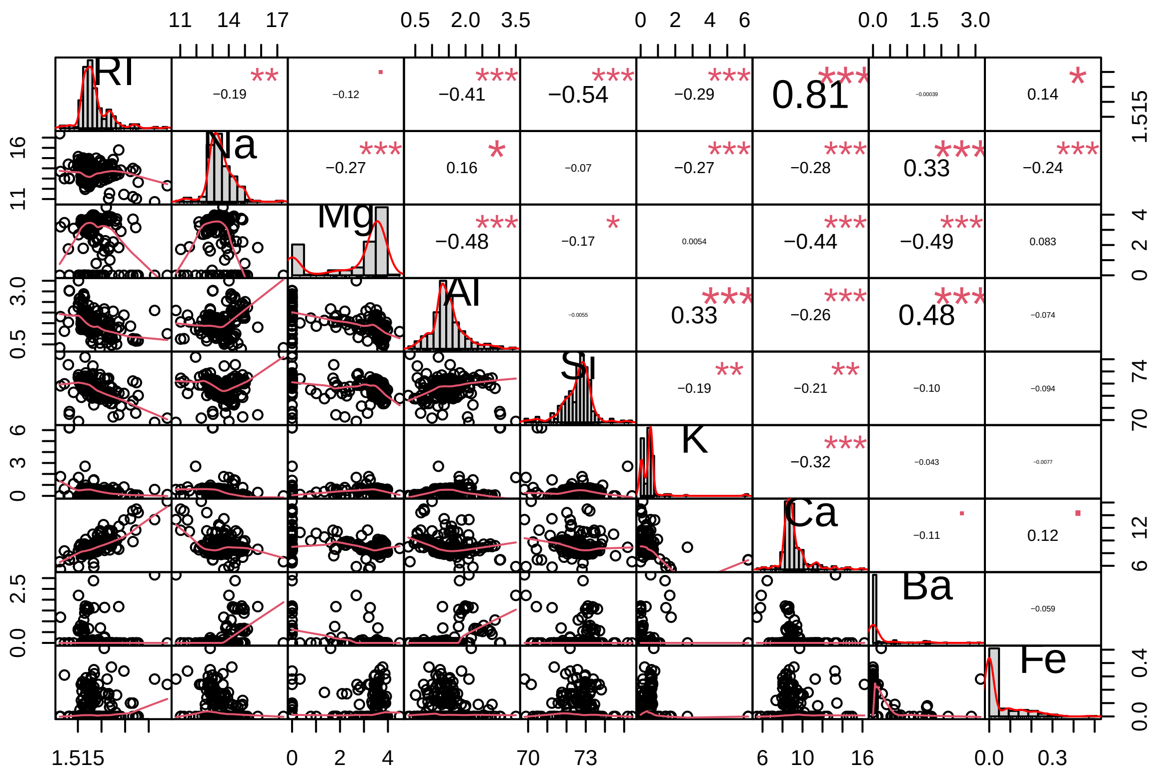


Check correlation matrix;

```
library(PerformanceAnalytics)
```

```
## Warning: package 'PerformanceAnalytics' was built under R version 4.1.3

## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

```
chart.Correlation(X)
```



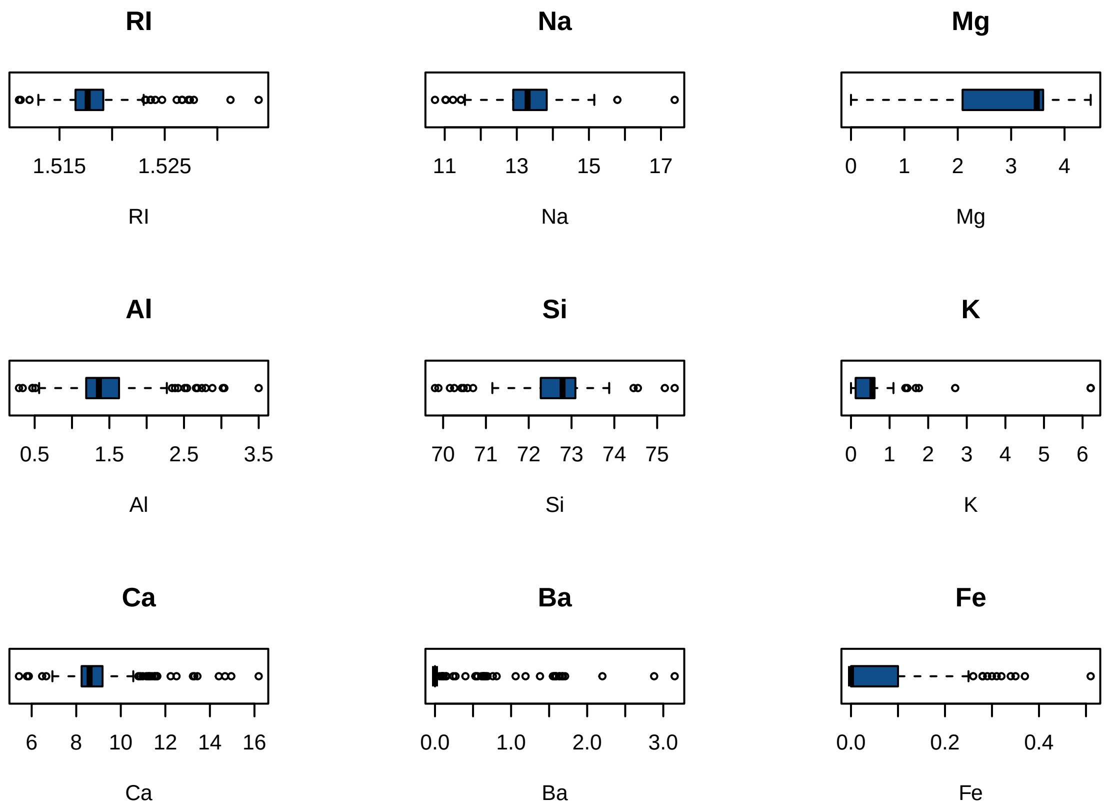
Analysis Looking at the histograms of the distribution for each of the variables, 4 of them seem to be approximately normally distributed; RI, Na, Al and Si, and the rest appear more skewed. K, Ca, Ba, and Fe are right skewed, and Mg is left skewed.

When looking at the correlation matrix, Ri and Ca look to be correlated to a significant level to cause problems. There is some slight correlation between RI and Si, Al and Ba, and Mg and Ba as well, but probably not enought to cause issues.

(b) Do there appear to be any outliers in the data? Are any predictors skewed? The skewed question was answered above, when looking at the histograms. K, Ca, Ba, and Fe are right skewed, and Mg is left skewed.

We can check the outliers visually via boxplots;

```
par(mfrow = c(3,3))
library(lattice)
for (i in 1:ncol(X)) {
  boxplot(X[ ,i],
  xlab = names(X[i]),
  main = paste(names(X[i])),
  horizontal = TRUE,
  col = "dodgerblue4")
}
```



Looking at the boxplots for each of the different variables, most don't seem to have outliers that are too bad, with the exceptions of Ba and K, which have some extreme ones, and Fe and Ca, which have some moderately extreme outliers.

(c) Are there any relevant transformations of one or more predictors that might improve the classification model? We can check the quantitative skewness of each predictor with the skewness function;

```
skewness(X)
```

```
##                 RI        Na        Mg        Al         Si        K       Ca
## Skewness 1.614015 0.4509917 -1.144465 0.9009179 -0.7253173 6.505636 2.032677
##                 Ba       Fe
## Skewness 3.392431 1.742007
```

RI, Mg, K, Ca, Ba and Fe are all outside the range of -1 to 1, so transformations would be in order.

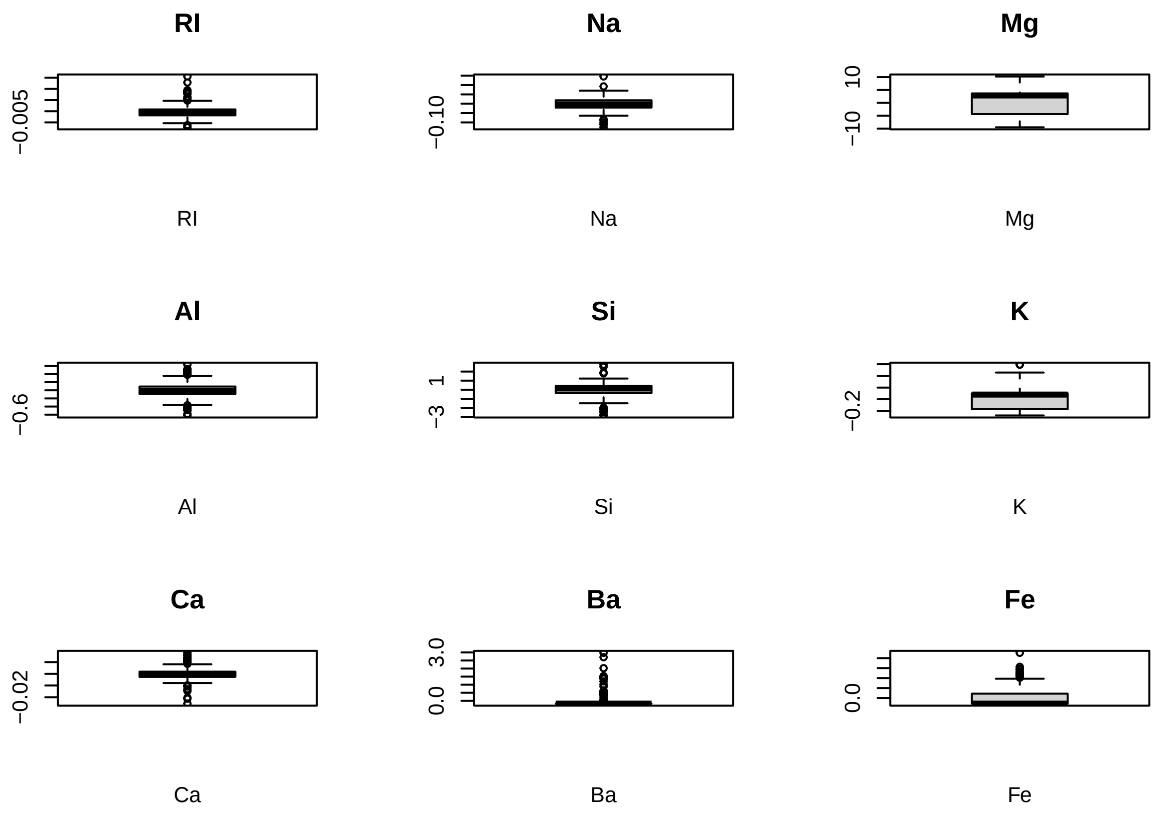
I already went through this process trying different transformations in exercise 2, and found YeoJohnson and center to do the best job of transforming the variables.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.2
```

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

```
GlassPP = preProcess(Glass, method = c("YeoJohnson", "center"))
GlassTrans = predict(GlassPP, Glass)
par(mfrow = c(3,3))
library(lattice)
a = GlassTrans[, 1:9]
for (i in 1:ncol(a)) {
  boxplot(a[ ,i],
  xlab = names(a[i]),
  main = paste(names(a[i])))
}
```



The transformations did normalize most of the skewed variables, except for Ba, which seems to be hopelessly skewed.

Exercise 3.2

```
library(mlbench)
data(Soybean)
str(Soybean)
```

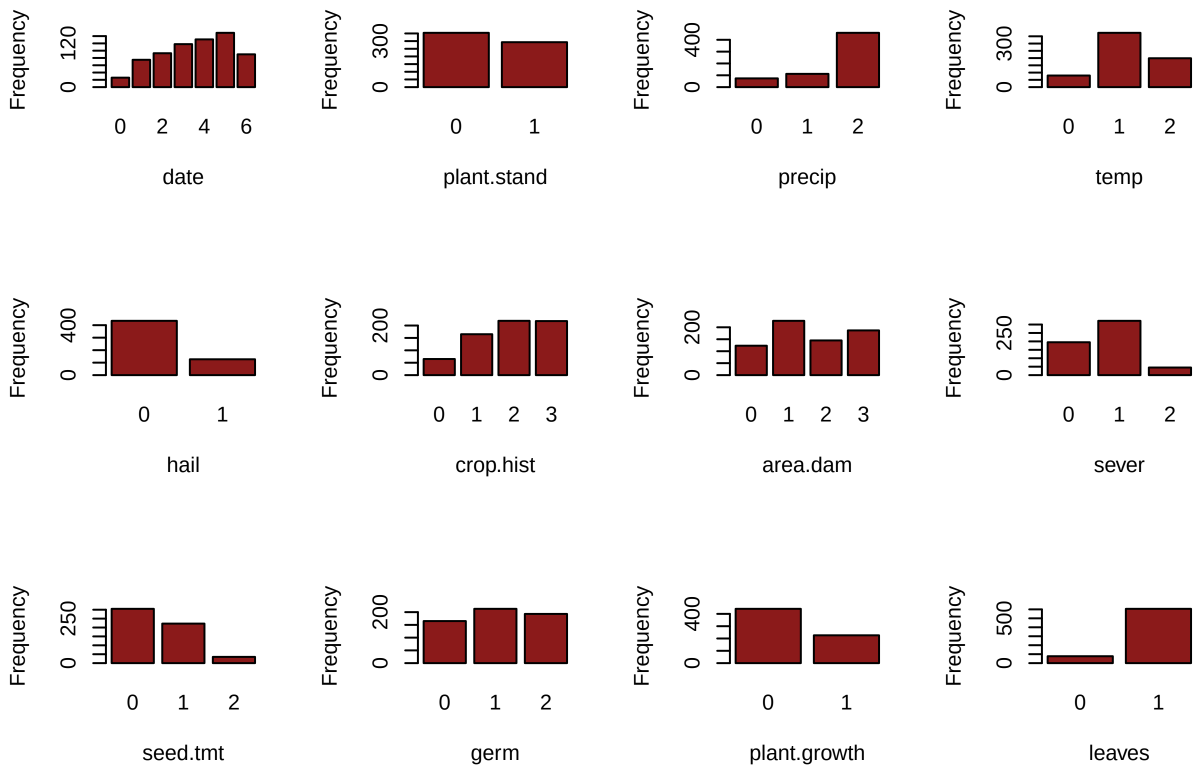
The soybean data can also be found at the UC Irvine Machine Learning Repository. Data were collected to predict disease in 683 soybeans. The 35 predictors are mostly categorical

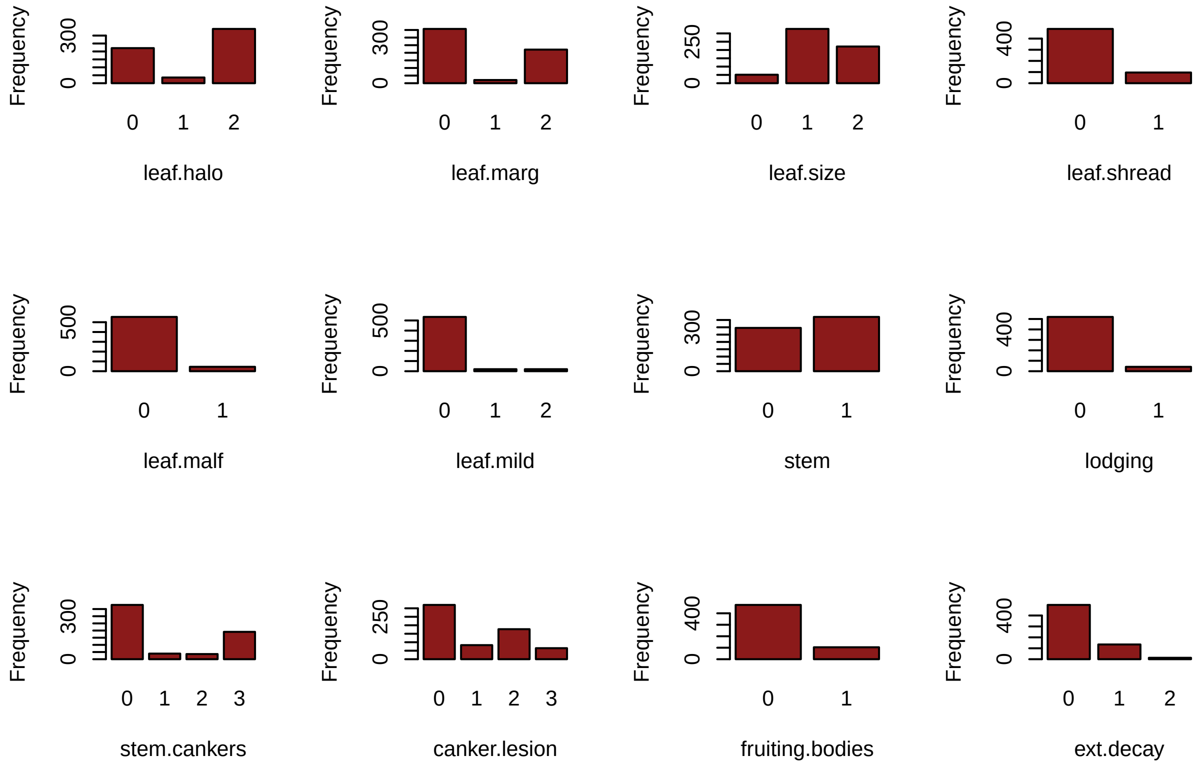
and include information on the environmental conditions (e.g., temperature, precipitation) and plant conditions (e.g., left spots, mold growth). The outcome labels consist of 19 distinct classes.

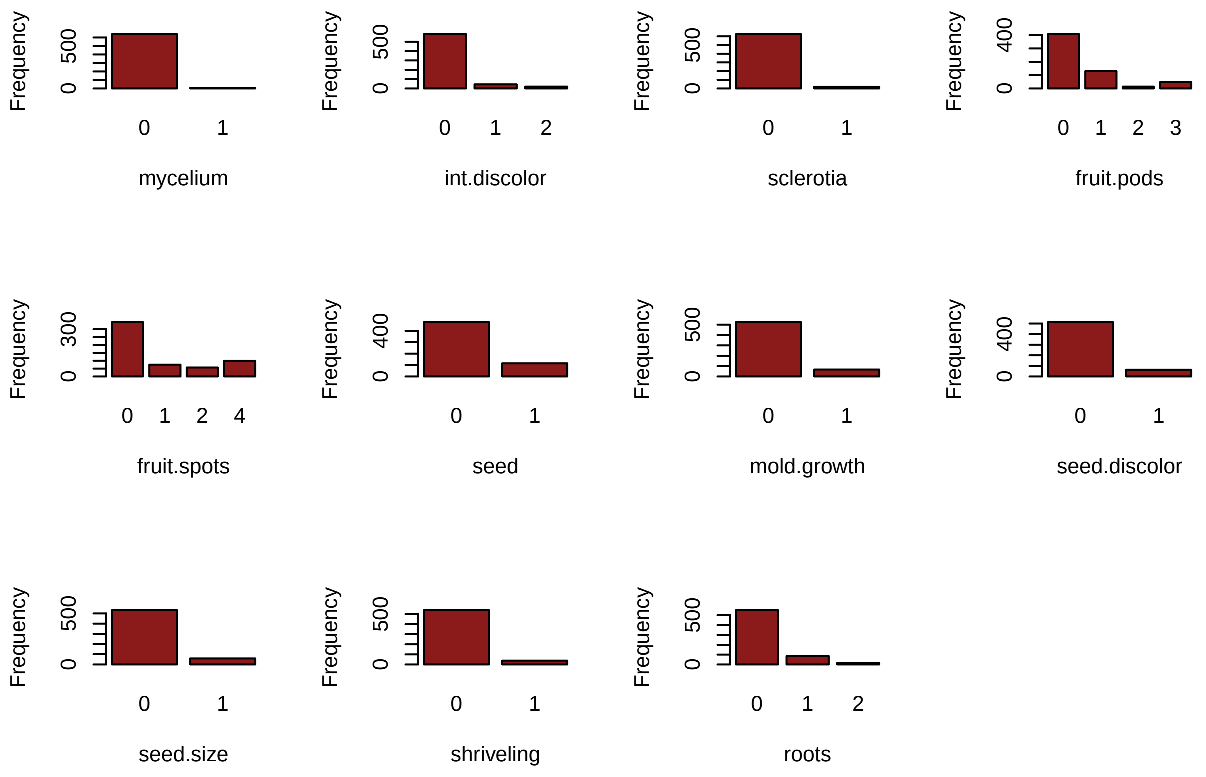
```
## 'data.frame':    683 obs. of  36 variables:
##  $ Class          : Factor w/ 19 levels "2-4-d-injury",..: 11 11 11 11 11 11 11 11 11 11 ...
##  $ date           : Factor w/ 7 levels "0","1","2","3",..: 7 5 4 4 7 6 6 5 7 5 ...
##  $ plant.stand    : Ord.factor w/ 2 levels "0"<"1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ precip         : Ord.factor w/ 3 levels "0"<"1"<"2": 3 3 3 3 3 3 3 3 3 3 ...
##  $ temp           : Ord.factor w/ 3 levels "0"<"1"<"2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ hail           : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
##  $ crop.hist      : Factor w/ 4 levels "0","1","2","3": 2 3 2 2 3 4 3 2 4 3 ...
##  $ area.dam       : Factor w/ 4 levels "0","1","2","3": 2 1 1 1 1 1 1 1 1 1 ...
##  $ sever          : Factor w/ 3 levels "0","1","2": 2 3 3 3 2 2 2 2 2 3 ...
##  $ seed.tmt       : Factor w/ 3 levels "0","1","2": 1 2 2 1 1 1 2 1 2 1 ...
##  $ germ           : Ord.factor w/ 3 levels "0"<"1"<"2": 1 2 3 2 3 2 1 3 2 3 ...
##  $ plant.growth   : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ leaves         : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ leaf.halo      : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ leaf.marg      : Factor w/ 3 levels "0","1","2": 3 3 3 3 3 3 3 3 3 3 ...
##  $ leaf.size      : Ord.factor w/ 3 levels "0"<"1"<"2": 3 3 3 3 3 3 3 3 3 3 ...
##  $ leaf.shread    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ leaf.malf      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ leaf.mild      : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ stem           : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ lodging        : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 2 1 1 1 ...
##  $ stem.cankers   : Factor w/ 4 levels "0","1","2","3": 4 4 4 4 4 4 4 4 4 4 ...
##  $ canker.lesion  : Factor w/ 4 levels "0","1","2","3": 2 2 1 1 2 1 2 2 2 2 ...
##  $ fruiting.bodies: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ext.decay      : Factor w/ 3 levels "0","1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ mycelium       : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ int.discolor   : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sclerotia      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ fruit.pods     : Factor w/ 4 levels "0","1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ fruit.spots    : Factor w/ 4 levels "0","1","2","4": 4 4 4 4 4 4 4 4 4 4 ...
##  $ seed           : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ mold.growth    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ seed.discolor  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ seed.size      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ shriveling     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ roots          : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
```

(a) Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter? Check frequency distributions for each variable;

```r
Y = Soybean[, 2:36]
par(mfrow = c(3, 4))
for (i in 1:ncol(Y)) {
  plot(Y[, i],
  xlab = names(Y[i]),
  ylab = "Frequency",
  col = "firebrick4")
}
```


Frequency
0
120
0 2 4 6
date
300
0 1
plant.stand
400
0 1 2
precip
300
0 1 2
temp
400
0 1
hail
200
0 1 2 3
crop.hist
200
0 1 2 3
area.dam
250
0 1 2
sever
250
0 1 2
seed.tmt
200
0 1 2
germ
400
0 1
plant.growth
500
0 1
leaves


Frequency
0
300
0 1 2
leaf.halo
Frequency
0
300
0 1 2
leaf.marg
Frequency
0
250
0 1 2
leaf.size
Frequency
0
400
0 1
leaf.shread
Frequency
0
500
0 1
leaf.malf
Frequency
0
500
0 1 2
leaf.mild
Frequency
0
300
0 1
stem
Frequency
0
400
0 1
lodging
Frequency
0
300
0 1 2 3
stem.cankers
Frequency
0
250
0 1 2 3
canker.lesion
Frequency
0
400
0 1
fruiting.bodies
Frequency
0
400
0 1 2
ext.decay



So based on the discussion in the chapter, degenerate variables are ones that have near zero variance or zero variance, i.e. a frequency distribution where almost all of the results are one single result.

It looks like there's a couple variables that might be close, like mycellium, scierotia, leaf.mild, leaf.malf, but it's a little hard to tell for sure just by the graphs.

We can check the variance quantitatively to see if any are close to zero;

```r
nearZeroVar(Y, names = TRUE, saveMetrics = TRUE)
```

```
##                  freqRatio percentUnique zeroVar   nzv
## date              1.137405     1.0248902   FALSE FALSE
## plant.stand       1.208191     0.2928258   FALSE FALSE
## precip            4.098214     0.4392387   FALSE FALSE
## temp              1.879397     0.4392387   FALSE FALSE
## hail              3.425197     0.2928258   FALSE FALSE
## crop.hist         1.004587     0.5856515   FALSE FALSE
## area.dam          1.213904     0.5856515   FALSE FALSE
## sever             1.651282     0.4392387   FALSE FALSE
## seed.tmt          1.373874     0.4392387   FALSE FALSE
## germ              1.103627     0.4392387   FALSE FALSE
## plant.growth      1.951327     0.2928258   FALSE FALSE
## leaves            7.870130     0.2928258   FALSE FALSE
## leaf.halo         1.547511     0.4392387   FALSE FALSE
## leaf.marg         1.615385     0.4392387   FALSE FALSE
## leaf.size         1.479638     0.4392387   FALSE FALSE
## leaf.shread       5.072917     0.2928258   FALSE FALSE
## leaf.malf        12.311111     0.2928258   FALSE FALSE
## leaf.mild        26.750000     0.4392387   FALSE  TRUE
## stem              1.253378     0.2928258   FALSE FALSE
## lodging          12.380952     0.2928258   FALSE FALSE
## stem.cankers      1.984293     0.5856515   FALSE FALSE
## canker.lesion     1.807910     0.5856515   FALSE FALSE
## fruiting.bodies   4.548077     0.2928258   FALSE FALSE
## ext.decay         3.681481     0.4392387   FALSE FALSE
## mycelium        106.500000     0.2928258   FALSE  TRUE
## int.discolor     13.204545     0.4392387   FALSE FALSE
## sclerotia        31.250000     0.2928258   FALSE  TRUE
## fruit.pods        3.130769     0.5856515   FALSE FALSE
## fruit.spots       3.450000     0.5856515   FALSE FALSE
## seed              4.139130     0.2928258   FALSE FALSE
## mold.growth       7.820896     0.2928258   FALSE FALSE
## seed.discolor     8.015625     0.2928258   FALSE FALSE
## seed.size         9.016949     0.2928258   FALSE FALSE
## shriveling       14.184211     0.2928258   FALSE FALSE
## roots             6.406977     0.4392387   FALSE FALSE
```

It looks like none have zero variance, but three of the variables do have near zero variance;

```
nearZeroVar(Y, names = TRUE)
```

```
## [1] "leaf.mild" "mycelium"  "sclerotia"
```

Leaf.mild, Mycelium, and Sclerotia are all degenerate.

(b) Roughly 18 % of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes? The aggr() function can count the number of missing data points for each predictor;

```
library(VIM)
```

```
## Warning: package 'VIM' was built under R version 4.1.3
```

```
## Loading required package: colorspace
```

```
## Loading required package: grid
```

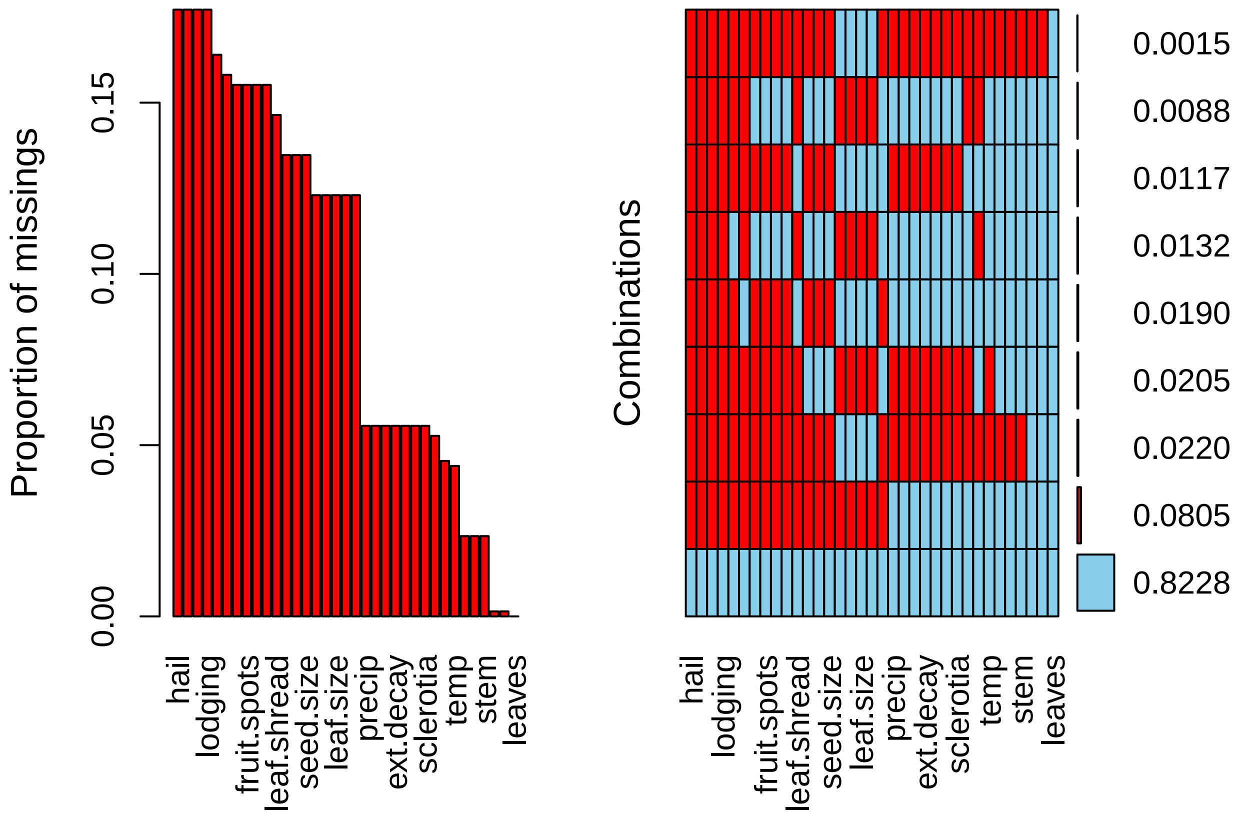
```
## VIM is ready to use.
```

```
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues
```

```
##
## Attaching package: 'VIM'
```

```
## The following object is masked from 'package:datasets':
##
##     sleep
```

```r
aggr(Y,
     sortVars = TRUE,
     numbers = TRUE,
     bars = TRUE)
```



```
##
##  Variables sorted by number of missings:
##         Variable       Count
##             hail 0.177159590
##            sever 0.177159590
##         seed.tmt 0.177159590
##          lodging 0.177159590
##             germ 0.163982430
##        leaf.mild 0.158125915
##  fruiting.bodies 0.155197657
##      fruit.spots 0.155197657
##    seed.discolor 0.155197657
##       shriveling 0.155197657
##      leaf.shread 0.146412884
##             seed 0.134699854
##      mold.growth 0.134699854
##        seed.size 0.134699854
##        leaf.halo 0.122986823
##        leaf.marg 0.122986823
##        leaf.size 0.122986823
##        leaf.malf 0.122986823
##       fruit.pods 0.122986823
##           precip 0.055636896
##     stem.cankers 0.055636896
##    canker.lesion 0.055636896
##        ext.decay 0.055636896
##         mycelium 0.055636896
##     int.discolor 0.055636896
##        sclerotia 0.055636896
##      plant.stand 0.052708638
##            roots 0.045387994
##             temp 0.043923865
##        crop.hist 0.023426061
##     plant.growth 0.023426061
##             stem 0.023426061
##             date 0.001464129
##         area.dam 0.001464129
##           leaves 0.000000000
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:xts':
##
##     first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
Soybean %>%
  mutate(Total = n()) %>%
  filter(!complete.cases(.)) %>%
  group_by(Class) %>%
  mutate(Missing = n(), Proportion=Missing/(68+15+14+16+8)) %>%
  select(Class, Missing, Proportion) %>%
  unique()
```

```
## # A tibble: 5 x 3
## # Groups:   Class [5]
##   Class                       Missing Proportion
##   <fct>                         <int>      <dbl>
## 1 phytophthora-rot                 68     0.562
## 2 diaporthe-pod-&-stem-blight      15     0.124
## 3 cyst-nematode                    14     0.116
## 4 2-4-d-injury                     16     0.132
## 5 herbicide-injury                  8     0.0661
```

Yes, there are various rates of missing values for the different predictors; hail, sever, seed.tmt, lodging are the top 4 that have elevated rates of missing values.

As for higher proportions of missing values by class, "phytophthora-rot" has the overwhelming majority of missing values at 56%, with the next highest class at only 12%.

(c) Develop a strategy for handling missing data, either by eliminating predictors or imputation. Removing the entries with missing values is one of the more straightforward options, but since the distribution of missings is not equal across classes or predictors, I'm going to impute them.

This wasn't in the demonstration for data processing but in doing some R research it looks like the mice() function is fairly standard for imputing missing values, and I'm going to use predictive mean matching, which works for either numeric or categorical variables as the method;

```
library(mice)
```

```
## Warning: package 'mice' was built under R version 4.1.3
```

```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

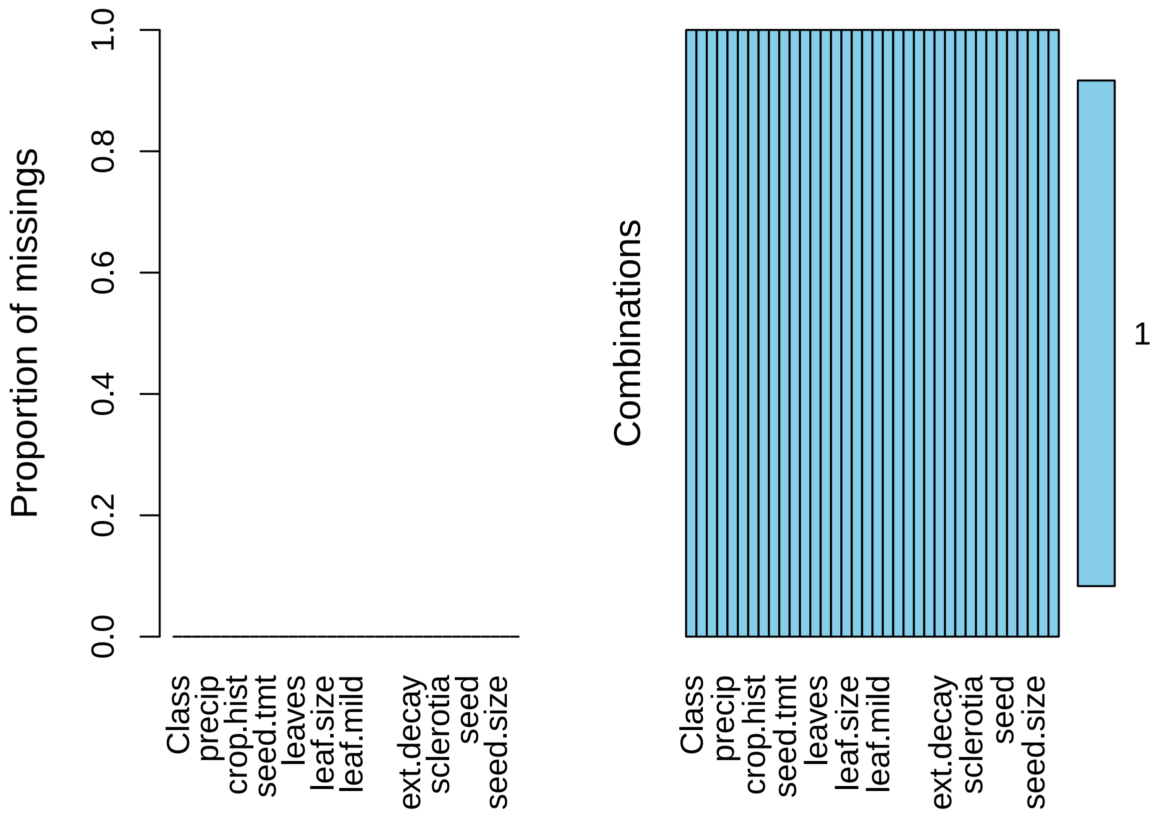
```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
soy2 = mice(Soybean,
            m = 5,
            method = "pmm",
            printFlag = FALSE)
```

```
## Warning: Number of logged events: 1671
```

Can now run the aggr() function again to make sure that the missing values were imputed;

```
aggr(complete(soy2),
     sortVars = TRUE,
     numbers = TRUE,
     bars = TRUE)
```



```
##
##  Variables sorted by number of missings:
##         Variable Count
##            Class     0
##             date     0
##      plant.stand     0
##           precip     0
##             temp     0
##             hail     0
##        crop.hist     0
##         area.dam     0
##            sever     0
##         seed.tmt     0
##             germ     0
##     plant.growth     0
##           leaves     0
##        leaf.halo     0
##        leaf.marg     0
##        leaf.size     0
##      leaf.shread     0
##        leaf.malf     0
##        leaf.mild     0
##             stem     0
##          lodging     0
##     stem.cankers     0
##    canker.lesion     0
## fruiting.bodies     0
##       ext.decay     0
##        mycelium     0
##    int.discolor     0
##       sclerotia     0
##      fruit.pods     0
##     fruit.spots     0
##            seed     0
##     mold.growth     0
##   seed.discolor     0
##       seed.size     0
##      shriveling     0
##           roots     0
```

Exercise 4.1

Consider the music genre data set described in Sect. 1.4. The objective for these data is to use the predictors to classify music samples into the appropriate music genre.

(a) What data splitting method(s) would you use for these data? Explain. The distribution of data for the music genre dataset is summarized in a figure from the text that I'm having a hard time figuring out how to add to this rmd file, so I will have to just describe the dataset.

According to the text, there are 12,495 samples and 191 predictors. There are far more samples than predictors, so we are safe to divide it into a training and testing set on that front.

The distribution of class is not equal, for instance, there are far more samples of classical than metal, but there's enough of each class that we should be able to split the data using resampling and cross validation, and 10 fold cross-validation should be adequate and not too computationally taxing.

(b) Using tools described in this chapter, provide code for implementing your approach(es). I will use the createDataPartition() function and a train/test split of 80/20, and since I don't have the actual music dataset, I will write the code using a fake data file of "music";

```
# set.seed(1)
# music.split = createDataPartition(music, p = .80, k = 10, list = FALSE)
```

Exercise 4.4

```


data(oil)
str(oilType)
```

. Brodnjak-Vonina et al. (2005) develop a methodology for food laboratories to determine the type of oil from a sample. In their procedure, they used a gas chromatograph (an instrument that separate chemicals in a sample) to measure seven different fatty acids in an oil. These measurements would then be used to predict the type of oil in a food samples. To create their model, they used 96 samples2 of seven types of oils.

```
##  Factor w/ 7 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
set.seed(123)
n = 10
oilSamples = vector(mode = "list", length = 10)
for (i in seq(along = oilSamples)) oilSamples[[i]] = table(sample(oilType, size = n))
head(oilSamples)
```

(a) Use the sample function in base R to create a completely random sample of 60 oils. How closely do the frequencies of the random sample match the original samples? Repeat this procedure several times of understand the variation in the sampling process.

```
## [[1]]
##
## A B C D E F G
## 2 2 0 2 3 1 0
##
## [[2]]
##
## A B C D E F G
## 4 0 0 1 1 4 0
##
## [[3]]
##
## A B C D E F G
## 4 5 0 1 0 0 0
##
## [[4]]
##
## A B C D E F G
## 4 1 0 1 0 2 2
##
## [[5]]
##
## A B C D E F G
## 5 4 0 0 1 0 0
##
## [[6]]
##
## A B C D E F G
## 4 2 0 0 2 2 0
```

It looks like there's a great deal of variation just using a simple random sample

```r
oilSamples = do.call("rbind", oilSamples)
summary(oilSamples)
```

```
##        A               B              C            D              E
##  Min.   :2.00   Min.   :0.0   Min.   :0   Min.   :0.0   Min.   :0.00
##  1st Qu.:3.25   1st Qu.:2.0   1st Qu.:0   1st Qu.:0.0   1st Qu.:1.00
##  Median :4.00   Median :4.0   Median :0   Median :0.5   Median :1.00
##  Mean   :3.70   Mean   :3.1   Mean   :0   Mean   :0.6   Mean   :1.20
##  3rd Qu.:4.00   3rd Qu.:4.0   3rd Qu.:0   3rd Qu.:1.0   3rd Qu.:1.75
##  Max.   :5.00   Max.   :5.0   Max.   :0   Max.   :2.0   Max.   :3.00
##        F              G
##  Min.   :0.00   Min.   :0.0
##  1st Qu.:0.00   1st Qu.:0.0
##  Median :1.00   Median :0.0
##  Mean   :1.10   Mean   :0.3
##  3rd Qu.:1.75   3rd Qu.:0.0
##  Max.   :4.00   Max.   :2.0
```

(b) Use the caret package function createDataPartition to create a stratified random sample. How does this compare to the completely random samples? Create stratified random sample;

```
oilStrat = createDataPartition(oilType,
                               p = 0.8,
                               times = 10)
oilStrat= do.call("rbind", oilStrat)
summary(oilStrat)
```

```
##        V1              V2             V3              V4              V5
##  Min.   :1.00   Min.   :2.0   Min.   :3.00   Min.   :4.00   Min.   :5.00
##  1st Qu.:1.00   1st Qu.:2.0   1st Qu.:3.25   1st Qu.:4.25   1st Qu.:5.25
##  Median :1.00   Median :2.5   Median :4.00   Median :5.00   Median :6.00
##  Mean   :1.30   Mean   :2.7   Mean   :4.00   Mean   :5.20   Mean   :6.20
##  3rd Qu.:1.75   3rd Qu.:3.0   3rd Qu.:4.00   3rd Qu.:6.00   3rd Qu.:7.00
##  Max.   :2.00   Max.   :5.0   Max.   :6.00   Max.   :7.00   Max.   :8.00
##        V6              V7              V8               V9             V10
##  Min.   :6.00   Min.   : 8.0   Min.   : 9.00   Min.   :10.0   Min.   :11.0
##  1st Qu.:7.00   1st Qu.: 8.0   1st Qu.: 9.25   1st Qu.:11.0   1st Qu.:12.0
##  Median :7.00   Median : 8.5   Median :10.00   Median :11.0   Median :12.0
##  Mean   :7.60   Mean   : 8.9   Mean   :10.20   Mean   :11.4   Mean   :12.5
##  3rd Qu.:8.75   3rd Qu.:10.0   3rd Qu.:11.00   3rd Qu.:12.0   3rd Qu.:13.0
##  Max.   :9.00   Max.   :10.0   Max.   :12.00   Max.   :13.0   Max.   :15.0
##       V11              V12              V13              V14             V15
##  Min.   :12.00   Min.   :13.00   Min.   :14.0   Min.   :15.0   Min.   :16.00
##  1st Qu.:13.00   1st Qu.:14.00   1st Qu.:15.0   1st Qu.:16.0   1st Qu.:17.00
##  Median :13.50   Median :15.00   Median :16.0   Median :17.0   Median :18.00
##  Mean   :13.90   Mean   :15.20   Mean   :16.3   Mean   :17.3   Mean   :18.40
##  3rd Qu.:14.75   3rd Qu.:15.75   3rd Qu.:17.5   3rd Qu.:18.5   3rd Qu.:19.75
##  Max.   :17.00   Max.   :18.00   Max.   :19.0   Max.   :20.0   Max.   :21.00
##       V16              V17              V18              V19              V20
##  Min.   :17.00   Min.   :18.00   Min.   :19.0   Min.   :20.00   Min.   :21.00
##  1st Qu.:19.00   1st Qu.:20.00   1st Qu.:21.0   1st Qu.:22.25   1st Qu.:23.25
##  Median :19.50   Median :21.00   Median :22.0   Median :23.50   Median :24.50
##  Mean   :19.90   Mean   :21.00   Mean   :22.1   Mean   :23.30   Mean   :24.50
##  3rd Qu.:20.75   3rd Qu.:21.75   3rd Qu.:23.0   3rd Qu.:24.00   3rd Qu.:25.75
##  Max.   :23.00   Max.   :24.00   Max.   :25.0   Max.   :26.00   Max.   :27.00
##       V21              V22              V23              V24
##  Min.   :22.00   Min.   :23.00   Min.   :24.00   Min.   :25.00
##  1st Qu.:25.00   1st Qu.:26.25   1st Qu.:27.25   1st Qu.:28.25
##  Median :26.00   Median :27.00   Median :28.50   Median :29.50
##  Mean   :25.70   Mean   :26.80   Mean   :28.00   Mean   :29.10
##  3rd Qu.:26.75   3rd Qu.:27.75   3rd Qu.:29.00   3rd Qu.:30.00
##  Max.   :28.00   Max.   :29.00   Max.   :30.00   Max.   :32.00
##       V25             V26             V27             V28
##  Min.   :26.00   Min.   :27.00   Min.   :28.00   Min.   :29.00
##  1st Qu.:29.25   1st Qu.:30.25   1st Qu.:31.25   1st Qu.:32.25
##  Median :30.50   Median :32.00   Median :33.00   Median :34.00
##  Mean   :30.20   Mean   :31.60   Mean   :32.70   Mean   :33.70
##  3rd Qu.:31.00   3rd Qu.:32.75   3rd Qu.:34.50   3rd Qu.:35.50
##  Max.   :34.00   Max.   :35.00   Max.   :36.00   Max.   :37.00
##       V29             V30             V31            V32             V33
##  Min.   :30.00   Min.   :31.00   Min.   :32.0   Min.   :33.00   Min.   :35.0
##  1st Qu.:33.25   1st Qu.:34.25   1st Qu.:36.0   1st Qu.:37.25   1st Qu.:40.0
##  Median :35.00   Median :36.00   Median :37.0   Median :38.00   Median :40.0
##  Mean   :34.70   Mean   :36.00   Mean   :37.2   Mean   :38.30   Mean   :40.1
##  3rd Qu.:36.50   3rd Qu.:38.50   3rd Qu.:39.5   3rd Qu.:40.50   3rd Qu.:41.5
##  Max.   :38.00   Max.   :39.00   Max.   :40.0   Max.   :41.00   Max.   :43.0
##       V34            V35            V36            V37             V38
##  Min.   :37.0   Min.   :38.0   Min.   :39.0   Min.   :41.00   Min.   :42.00
##  1st Qu.:41.0   1st Qu.:42.0   1st Qu.:43.0   1st Qu.:44.25   1st Qu.:45.25
##  Median :41.5   Median :43.0   Median :44.0   Median :45.00   Median :46.50
##  Mean   :41.5   Mean   :42.7   Mean   :43.9   Mean   :45.10   Mean   :46.40
##  3rd Qu.:43.0   3rd Qu.:44.0   3rd Qu.:45.0   3rd Qu.:46.00   3rd Qu.:47.75
##  Max.   :44.0   Max.   :45.0   Max.   :48.0   Max.   :49.00   Max.   :50.00
##       V39             V40             V41             V42             V43
##  Min.   :45.00   Min.   :46.00   Min.   :47.00   Min.   :48.00   Min.   :49.0
##  1st Qu.:47.00   1st Qu.:48.00   1st Qu.:49.00   1st Qu.:50.25   1st Qu.:52.0
##  Median :47.50   Median :48.50   Median :49.50   Median :51.50   Median :53.0
##  Mean   :47.80   Mean   :48.80   Mean   :49.80   Mean   :51.20   Mean   :52.7
##  3rd Qu.:48.75   3rd Qu.:49.75   3rd Qu.:50.75   3rd Qu.:52.00   3rd Qu.:54.0
##  Max.   :51.00   Max.   :52.00   Max.   :53.00   Max.   :54.00   Max.   :55.0
##       V44            V45             V46          V47            V48
##  Min.   :50.0   Min.   :51.00   Min.   :52   Min.   :53.0   Min.   :54.0
##  1st Qu.:53.0   1st Qu.:54.25   1st Qu.:56   1st Qu.:57.0   1st Qu.:58.0
##  Median :54.0   Median :55.00   Median :56   Median :57.0   Median :59.0
##  Mean   :53.7   Mean   :54.90   Mean   :56   Mean   :57.1   Mean   :58.4
##  3rd Qu.:55.0   3rd Qu.:56.00   3rd Qu.:57   3rd Qu.:58.0   3rd Qu.:59.0
##  Max.   :56.0   Max.   :57.00   Max.   :58   Max.   :59.0   Max.   :61.0
##       V49             V50            V51            V52            V53
##  Min.   :55.00   Min.   :56.0   Min.   :57.0   Min.   :58.0   Min.   :59.00
##  1st Qu.:59.25   1st Qu.:61.0   1st Qu.:62.0   1st Qu.:63.0   1st Qu.:64.00
##  Median :60.00   Median :61.0   Median :62.0   Median :63.0   Median :64.00
##  Mean   :59.50   Mean   :60.6   Mean   :61.7   Mean   :62.7   Mean   :64.00
##  3rd Qu.:60.00   3rd Qu.:61.0   3rd Qu.:62.0   3rd Qu.:63.0   3rd Qu.:64.75
##  Max.   :62.00   Max.   :63.0   Max.   :64.0   Max.   :65.0   Max.   :67.00
##       V54            V55            V56             V57             V58
##  Min.   :60.0   Min.   :61.0   Min.   :62.00   Min.   :63.00   Min.   :64.0
##  1st Qu.:65.0   1st Qu.:66.0   1st Qu.:67.25   1st Qu.:68.25   1st Qu.:69.5
##  Median :65.0   Median :66.5   Median :68.00   Median :69.00   Median :71.0
##  Mean   :65.2   Mean   :66.3   Mean   :67.60   Mean   :68.70   Mean   :70.0
##  3rd Qu.:66.0   3rd Qu.:67.0   3rd Qu.:68.75   3rd Qu.:70.00   3rd Qu.:71.0
##  Max.   :68.0   Max.   :69.0   Max.   :70.00   Max.   :71.00   Max.   :72.0
##       V59             V60             V61             V62
##  Min.   :65.00   Min.   :67.00   Min.   :68.00   Min.   :69.00
##  1st Qu.:71.25   1st Qu.:72.25   1st Qu.:73.25   1st Qu.:74.25
##  Median :72.00   Median :73.00   Median :74.00   Median :75.00
##  Mean   :71.10   Mean   :72.20   Mean   :73.20   Mean   :74.20
##  3rd Qu.:72.00   3rd Qu.:73.00   3rd Qu.:74.00   3rd Qu.:75.00  
##  Max.   :73.00   Max.   :74.00   Max.   :75.00   Max.   :76.00  
##       V63             V64            V65            V66             V67       
##  Min.   :72.00   Min.   :73.0   Min.   :75.0   Min.   :76.00   Min.   :77.00  
##  1st Qu.:75.25   1st Qu.:77.0   1st Qu.:78.0   1st Qu.:79.25   1st Qu.:80.25  
##  Median :76.00   Median :77.0   Median :79.0   Median :80.00   Median :81.50  
##  Mean   :75.40   Mean   :77.1   Mean   :78.6   Mean   :79.80   Mean   :81.00  
##  3rd Qu.:76.00   3rd Qu.:78.0   3rd Qu.:79.0   3rd Qu.:80.75   3rd Qu.:82.00  
##  Max.   :77.00   Max.   :80.0   Max.   :81.0   Max.   :82.00   Max.   :83.00  
##       V68            V69             V70             V71            V72      
##  Min.   :78.0   Min.   :80.00   Min.   :81.00   Min.   :82.0   Min.   :83.0  
##  1st Qu.:82.0   1st Qu.:83.00   1st Qu.:84.00   1st Qu.:85.0   1st Qu.:87.0  
##  Median :82.5   Median :83.50   Median :84.50   Median :85.5   Median :88.0  
##  Mean   :82.2   Mean   :83.40   Mean   :84.50   Mean   :85.6   Mean   :87.1  
##  3rd Qu.:83.0   3rd Qu.:84.75   3rd Qu.:85.75   3rd Qu.:87.0   3rd Qu.:88.0  
##  Max.   :84.0   Max.   :85.00   Max.   :87.00   Max.   :88.0   Max.   :89.0  
##       V73             V74            V75            V76            V77      
##  Min.   :84.00   Min.   :86.0   Min.   :89.0   Min.   :90.0   Min.   :91.0  
##  1st Qu.:88.25   1st Qu.:90.0   1st Qu.:91.0   1st Qu.:92.0   1st Qu.:93.0  
##  Median :89.00   Median :90.0   Median :91.0   Median :92.0   Median :93.5  
##  Mean   :88.20   Mean   :89.5   Mean   :91.1   Mean   :92.1   Mean   :93.2  
##  3rd Qu.:89.00   3rd Qu.:90.0   3rd Qu.:92.0   3rd Qu.:93.0   3rd Qu.:94.0  
##  Max.   :90.00   Max.   :91.0   Max.   :92.0   Max.   :93.0   Max.   :94.0  
##       V78            V79      
##  Min.   :92.0   Min.   :94.0  
##  1st Qu.:94.0   1st Qu.:96.0  
##  Median :95.0   Median :96.0  
##  Mean   :94.4   Mean   :95.8  
##  3rd Qu.:95.0   3rd Qu.:96.0  
##  Max.   :95.0   Max.   :96.0
```

I'm not really sure about these results, I might have messed up the code somewhere, but if it's correct, then at least there's much less variability among predictors.

(3) With such a small samples size, what are the options for determiningperformance of the model? Should a test set be used? It's difficult to say, there's a smaller ratio of samples to predictors than in previous problems, but there are still enough that a test/training split is still possible, but it might not be necessary with this dataset.

```r
binom.test(16, 20)
```

(4) Try different samples sizes and accuracy rates to understand the trade-off between the uncertainty in the results, the model performance, and the test set size.

```
## 
##  Exact binomial test
## 
## data:  16 and 20
## number of successes = 16, number of trials = 20, p-value = 0.01182
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.563386 0.942666
## sample estimates:
## probability of success
##                    0.8
```

```
binom.test(24, 30)
```

```
##
##  Exact binomial test
##
## data:  24 and 30
## number of successes = 24, number of trials = 30, p-value = 0.001431
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.6143335 0.9228645
## sample estimates:
## probability of success
##                    0.8
```

```
binom.test(8, 10)
```

```
##
##  Exact binomial test
##
## data:  8 and 10
## number of successes = 8, number of trials = 10, p-value = 0.1094
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.4439045 0.9747893
## sample estimates:
## probability of success
##                    0.8
```

```
binom.test(32, 40)
```

```
##
##  Exact binomial test
##
## data:  32 and 40
## number of successes = 32, number of trials = 40, p-value = 0.0001822
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.6435220 0.9094776
## sample estimates:
## probability of success
##                    0.8
```

```
binom.test(40, 50)
```

```
##
```

```
##  Exact binomial test
##
## data:  40 and 50
## number of successes = 40, number of trials = 50, p-value = 2.386e-05
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.6628169 0.8996978
## sample estimates:
## probability of success
##                    0.8
```

```
binom.test(80, 100)
```

```
##
##  Exact binomial test
##
## data:  80 and 100
## number of successes = 80, number of trials = 100, p-value = 1.116e-09
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.7081573 0.8733444
## sample estimates:
## probability of success
##                    0.8
```

We can see through multiple tests of the binomial test that the lower level of the confidence interval keeps increasing, but at a sample size of around 40, the lower level stops increasing as much and the upper level starts to drop, so our ideal sample size is probably somewhere around 30.