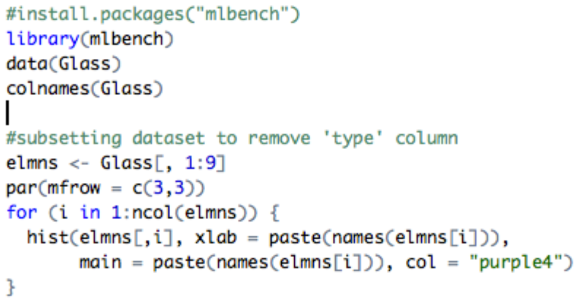
Alec Gray Jr.
SYST 568
Homework 1 – Preprocessing
09/20/18

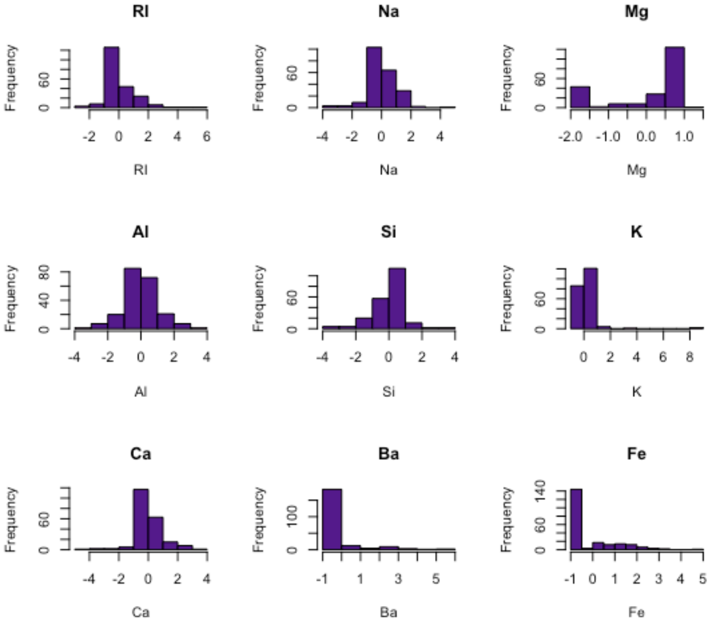
Question 3.1

A. Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.



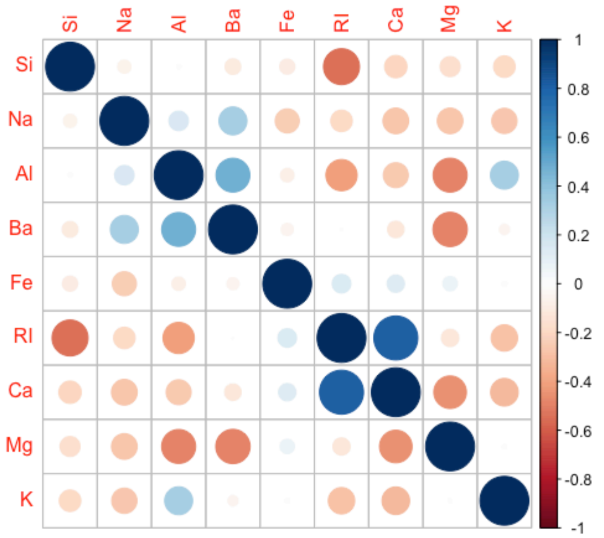
```
#install.packages("mlbench")
library(mlbench)
data(Glass)
colnames(Glass)

#subsetting dataset to remove 'type' column
elmns <- Glass[, 1:9]
par(mfrow = c(3,3))
for (i in 1:ncol(elmns)) {
  hist(elmns[,i], xlab = paste(names(elmns[i])),
       main = paste(names(elmns[i])), col = "purple4")
}
```

Histograms showing distributions of each predictor variable



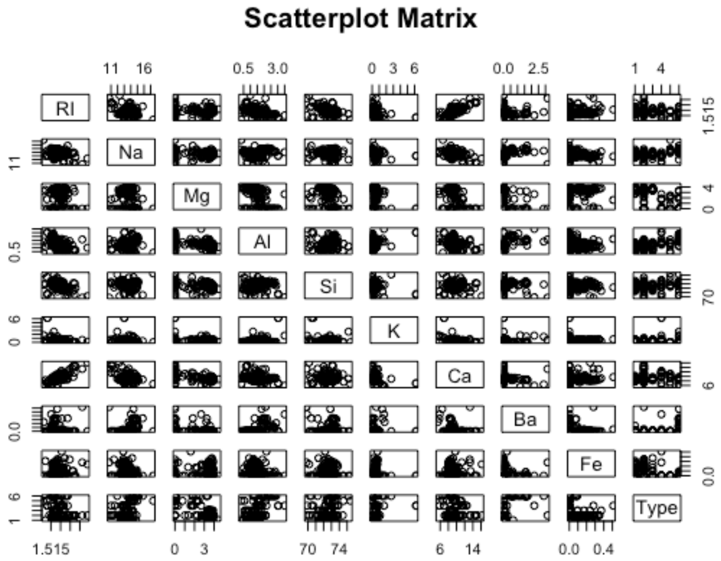
Correlation Plot of the predictors

```r
par(mfrow=c(1,1))
library(corrplot)
corrplot( cor( Glass[,-10] ), order="hclust")
```



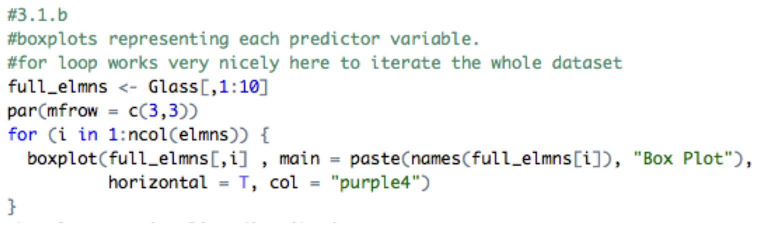
Pairwise scatterplot

```r
##relationship (Correlation) between predictors
par(mfrow=c(1,1))
pairs(Glass, main = "Scatterplot Matrix")
```

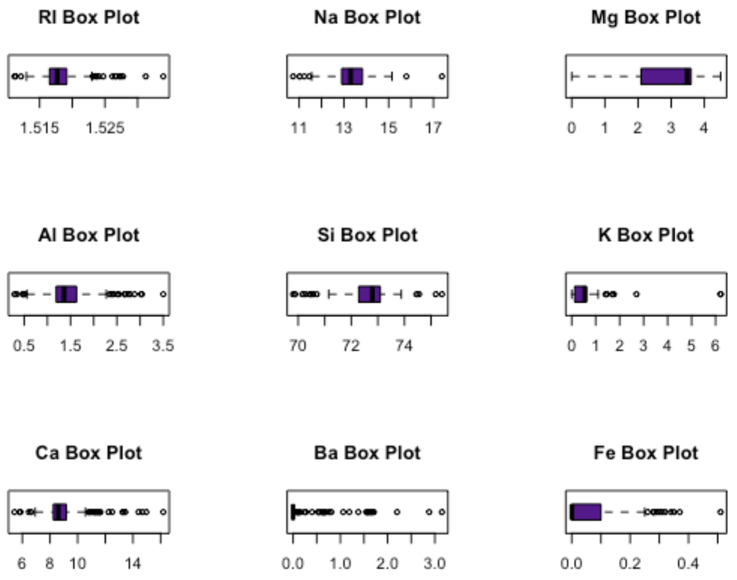


The histograms above indicate that RI, Na, Al, and Si are the only predictors that resemble a normal distribution. The other predictors are asymmetrical. According to the correlation plot, we see that RI possesses a negative correlation with Si and a positive correlation with Ca. This can also be clearly seen by the scatterplot matrix. The correlation plot would suggest that Mg – Al and Mg – Ba are both negatively correlated, but a clear correlation is not as apparent in the scatterplot matrix.

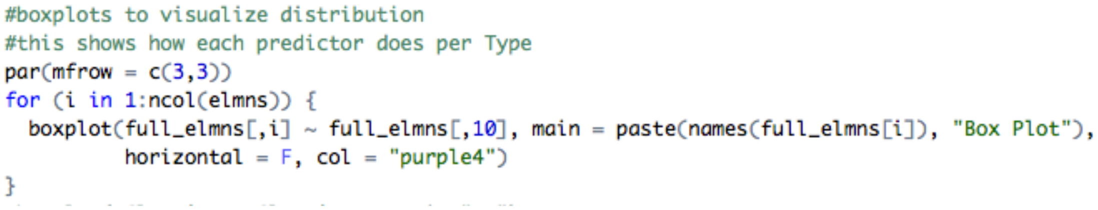
B. Do there appear to be any outliers in the data? Are any predictors skewed?
Box Plot Visualization to display outliers



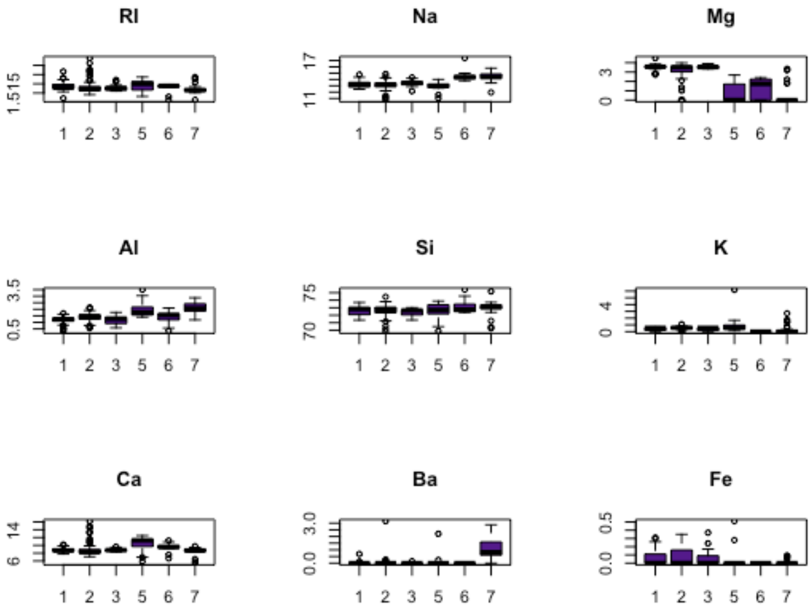
```
#3.1.b
#boxplots representing each predictor variable.
#for loop works very nicely here to iterate the whole dataset
full_elmns <- Glass[,1:10]
par(mfrow = c(3,3))
for (i in 1:ncol(elmns)) {
  boxplot(full_elmns[,i] , main = paste(names(full_elmns[i]), "Box Plot"),
          horizontal = T, col = "purple4")
}
```



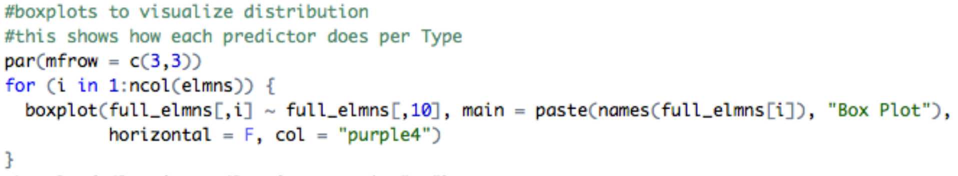
Box Plot Visualization that shows predictor dependency on Glass type



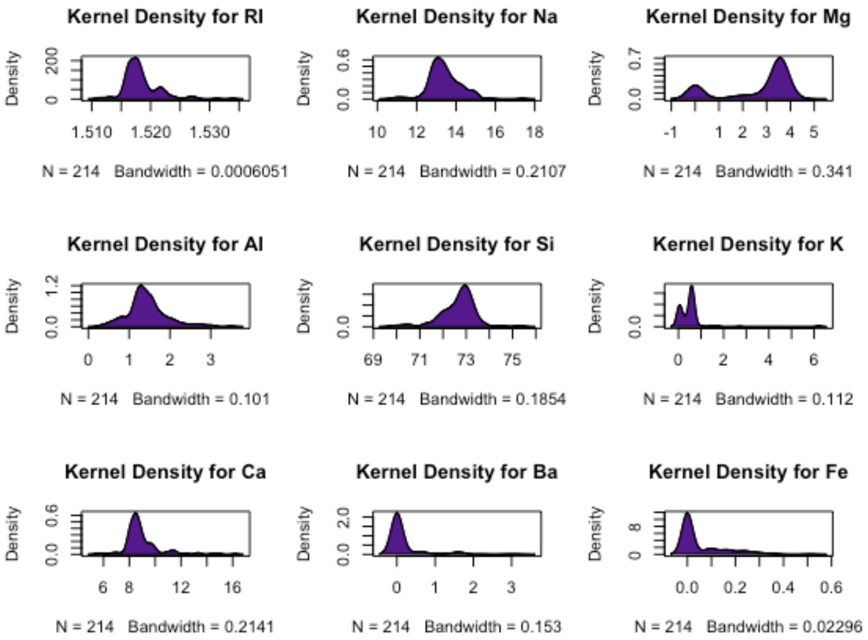
```
#boxplots to visualize distribution
#this shows how each predictor does per Type
par(mfrow = c(3,3))
for (i in 1:ncol(elmns)) {
  boxplot(full_elmns[,i] ~ full_elmns[,10], main = paste(names(full_elmns[i]), "Box Plot"),
          horizontal = F, col = "purple4")
}
```



Kernel Density Visualization to determine Skewedness

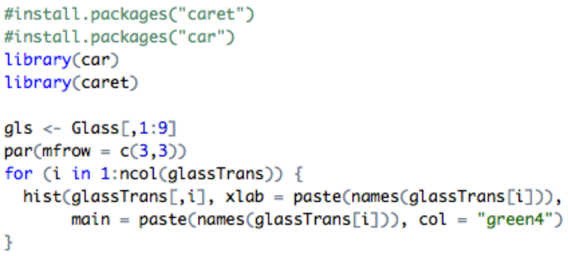


```
#boxplots to visualize distribution
#this shows how each predictor does per Type
par(mfrow = c(3,3))
for (i in 1:ncol(elmns)) {
  boxplot(full_elmns[,i] ~ full_elmns[,10], main = paste(names(full_elmns[i]), "Box Plot"),
          horizontal = F, col = "purple4")
}
```

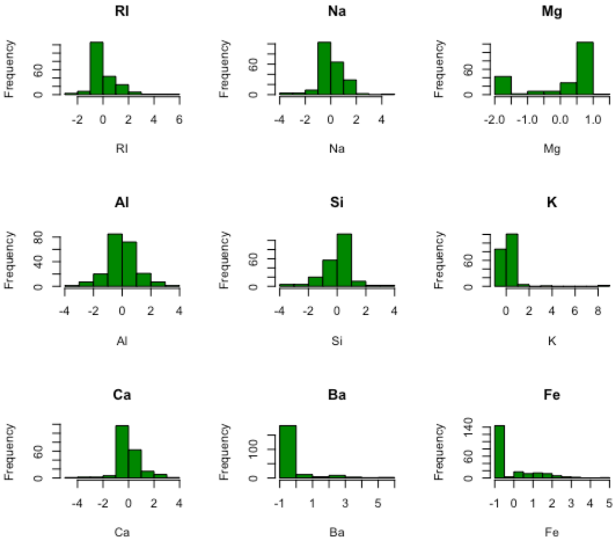


We see outliers in each predictor except for Mg, which doesn't have any values outside of either whisker. All the rest have some to many outliers, especially predictors RI, K, CA, and Ba. According to the density graphs, we see that Mg is left skewed, while Fe, Ba are hugely right skewed.

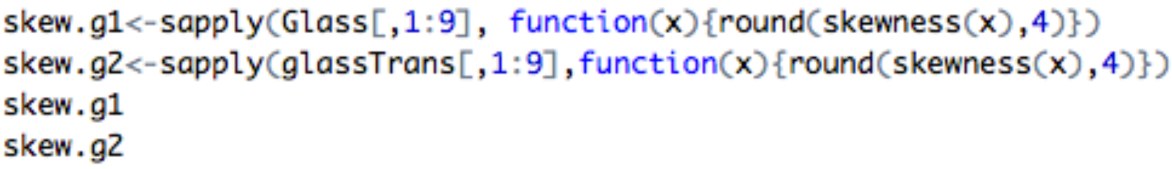
C. Are there any relevant transformations of one or more predictors that might improve the classification model?



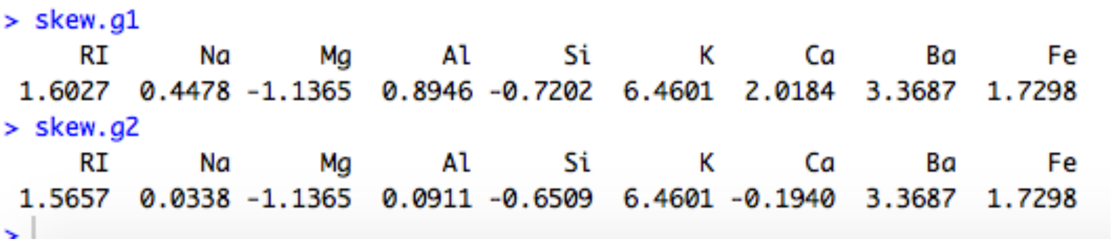
```
#install.packages("caret")
#install.packages("car")
library(car)
library(caret)

gls <- Glass[,1:9]
par(mfrow = c(3,3))
for (i in 1:ncol(glassTrans)) {
  hist(glassTrans[,i], xlab = paste(names(glassTrans[i])),
       main = paste(names(glassTrans[i])), col = "green4")
}
```



We use BoxCox to transform the distributions of each variable in order to remove skewness. In addition to this visual, I ran the skewness values of the original dataset as well as the transformed dataset to compare and see which skewness levels were smaller.



```
skew.g1<-sapply(Glass[,1:9], function(x){round(skewness(x),4)})
skew.g2<-sapply(glassTrans[,1:9],function(x){round(skewness(x),4)})
skew.g1
skew.g2
```



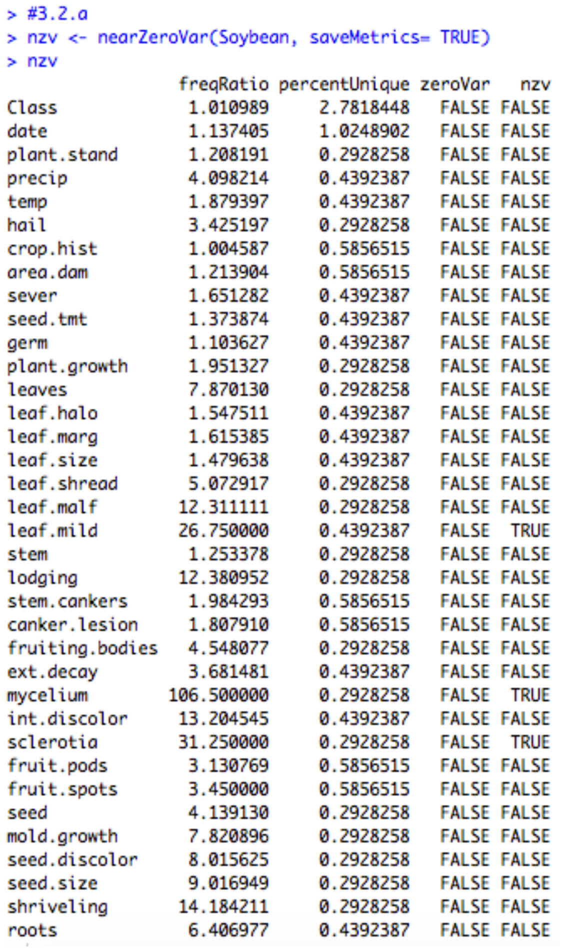
```
> skew.g1
     RI      Na      Mg      Al      Si       K      Ca      Ba      Fe 
 1.6027  0.4478 -1.1365  0.8946 -0.7202  6.4601  2.0184  3.3687  1.7298 
> skew.g2
     RI      Na      Mg      Al      Si       K      Ca      Ba      Fe 
 1.5657  0.0338 -1.1365  0.0911 -0.6509  6.4601 -0.1940  3.3687  1.7298 
> 
```

From the results, there are a few predictors that would benefit from a transformation. RI, NA, Al, Si, and Ca show less skewness in the transformed dataset than in the original Dataset. Of those predictors, transforming Ca, Al, and Na present the most beneficial change as the difference in their skewness values is 1.824, 0.8035, and 0.414 respectively.

Question 3.2

A. Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?

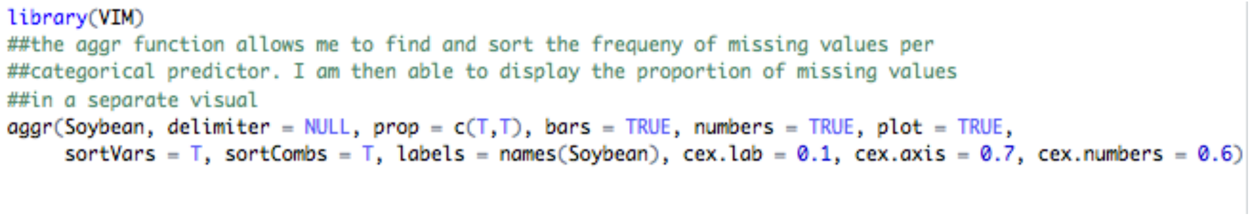
In order to determine degenerate distributions, we have to find predictors that have zero variance, meaning that predictor only has one single value. There is a function called 'nearZeroVar,' which calculates this with ease.



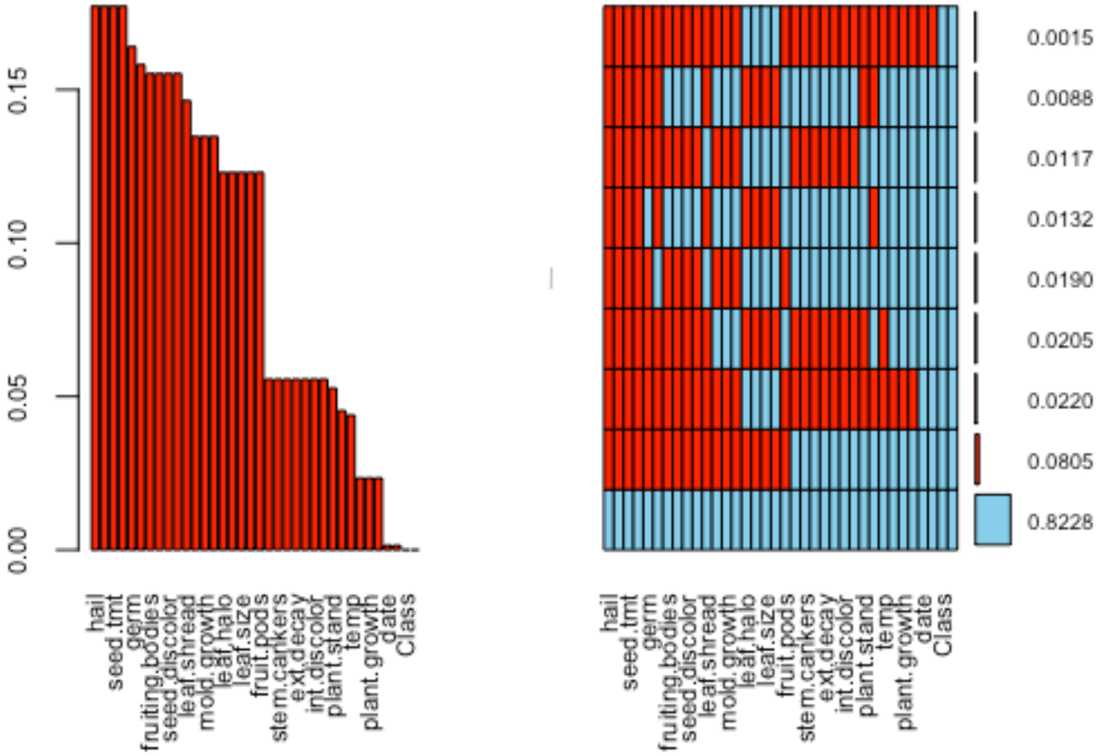
```
> #3.2.a
> nzv <- nearZeroVar(Soybean, saveMetrics= TRUE)
> nzv
                 freqRatio percentUnique zeroVar   nzv
Class             1.010989     2.7818448   FALSE FALSE
date              1.137405     1.0248902   FALSE FALSE
plant.stand       1.208191     0.2928258   FALSE FALSE
precip            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
```

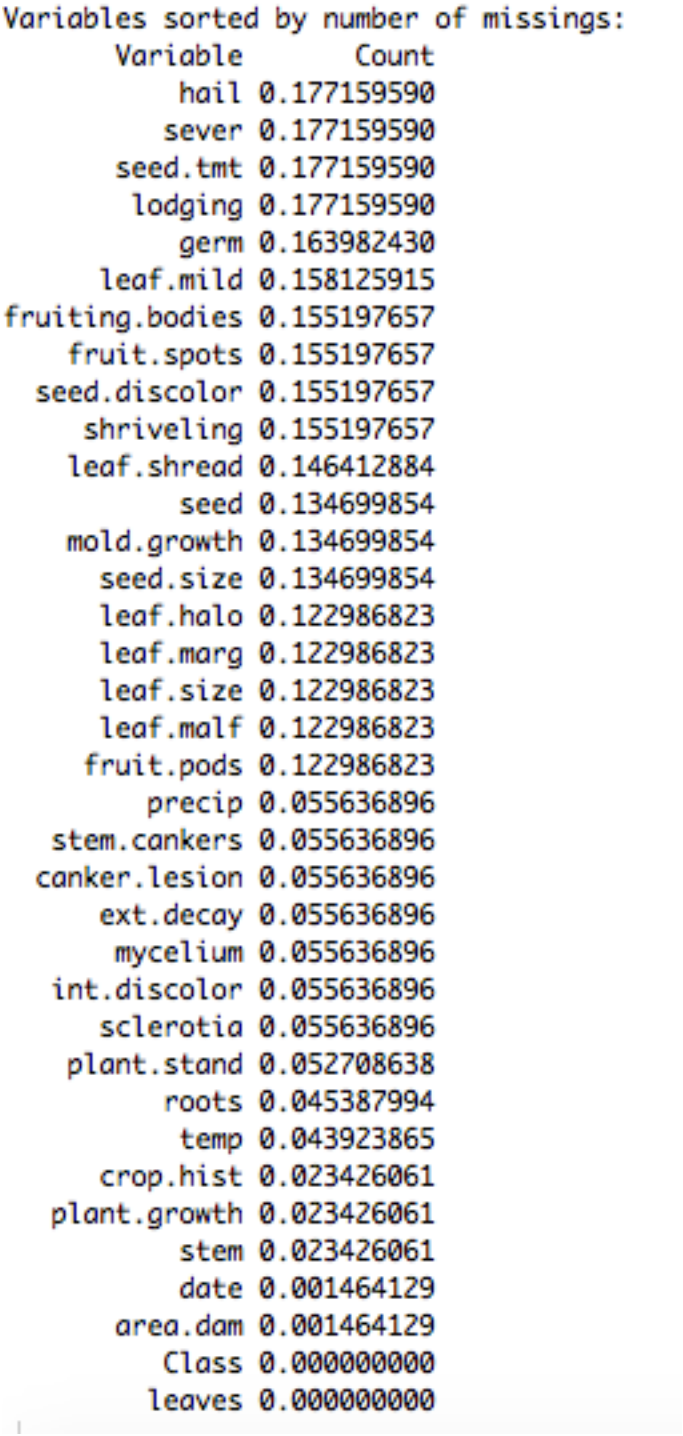
The findings results in No zero variance predictors, but there are three predictors that have very low variance (or near-zero variance): **leaf.mild**, **mycelium**, and **sclerotia**.

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?



```
library(VIM)
##the aggr function allows me to find and sort the frequeny of missing values per
##categorical predictor. I am then able to display the proportion of missing values
##in a separate visual
aggr(Soybean, delimiter = NULL, prop = c(T,T), bars = TRUE, numbers = TRUE, plot = TRUE,
     sortVars = T, sortCombs = T, labels = names(Soybean), cex.lab = 0.1, cex.axis = 0.7, cex.numbers = 0.6)
```



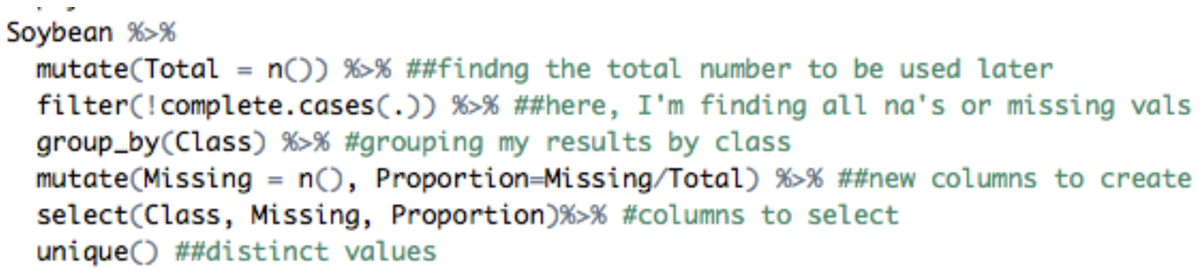


```
Variables sorted by number of missings:
       Variable       Count
           hail 0.177159590
          sever 0.177159590
       seed.tmt 0.177159590
        lodging 0.177159590
           germ 0.163982430
      leaf.mild 0.158125915
fruiting.bodies 0.155197657
    fruit.spots 0.155197657
  seed.discolor 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
          Class 0.000000000
         leaves 0.000000000
```

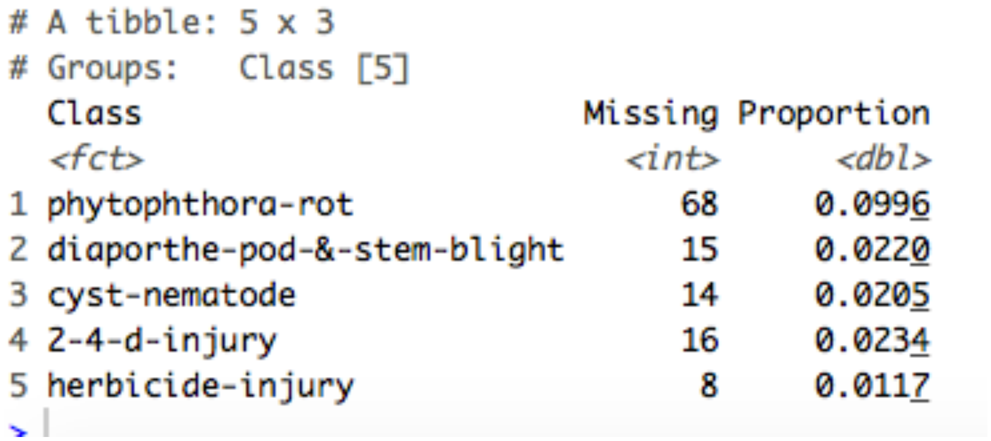
Using the VIM package in R for handling missing data points, I am able to use the aggr function to bin the frequencies of missing data points per categorical variable. The first visual presents a sorted bar chart of the proportion of missing data per categorical variable; it gives the percentage of missing data points per categorical variable. The second visual includes the

combined present and missing percentages for each categorical variable, and it shows the total amount of available data (82%, which is correct since there are 18% of missing data).
We can see that hail, server, and seed.tmt are the top three categorical variables that present the most amount of missing data.

In order to find potential patterns of missing data by class type, we have to use a data manipulation package like dplyr (pronounced dee-plyer). This package allows us to customize and manipulate the return information from a dataset to meet our requirements. It is basically SQL for R.



```
Soybean %>%
  mutate(Total = n()) %>% ##findng the total number to be used later
  filter(!complete.cases(.)) %>% ##here, I'm finding all na's or missing vals
  group_by(Class) %>% #grouping my results by class
  mutate(Missing = n(), Proportion=Missing/Total) %>% ##new columns to create
  select(Class, Missing, Proportion)%>% #columns to select
  unique() ##distinct values
```



```
# A tibble: 5 x 3
# Groups:   Class [5]
  Class                       Missing Proportion
  <fct>                         <int>      <dbl>
1 phytophthora-rot                 68     0.0996
2 diaporthe-pod-&-stem-blight      15     0.0220
3 cyst-nematode                    14     0.0205
4 2-4-d-injury                     16     0.0234
5 herbicide-injury                  8     0.0117
>
```

From the data above, we see that the phytophthora-rot class shows up missing 68 times, which is almost 10% of the missing data. The remaining ~18% of missing data shows up in classes 2-5 above. We conclude that out of the 19 classes, the 5 classes above comprise 100% of the missing data in this dataset.