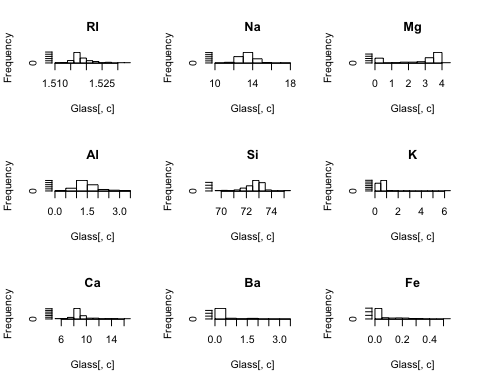
BHao\_HW3

# 3.1

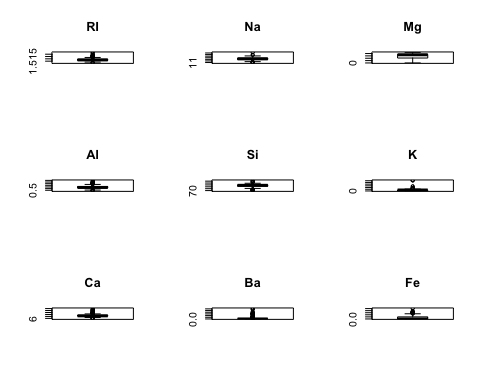
library(mlbench)  
data(Glass)

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

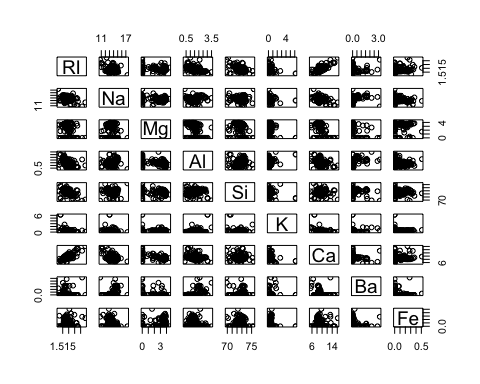
preds = colnames(Glass)  
  
# check histograms for each predictor   
par(mfrow = c(3, 3))  
for (c in 1:(length(preds)-1)){  
 hist(Glass[, c], main = preds[c])  
}



# check boxplots for each predictor   
par(mfrow = c(3, 3))  
for (c in 1:(length(preds)-1)){  
 boxplot(Glass[, c], main = preds[c])  
}



# check scatter plot matrix   
pairs(Glass[, 1:9])



## 3.1.b. Do their appear to be any outliers in the data? Are any predictors skewed?

Based on the histograms and boxplots, the following predictors appear to contain outliers: RI, Na, Al, Si, K, Ca, Ba, Fe. However, K, Ba, Fe are highly skewed to the right, so it's harder to see if the observations in the tails are in fact outliers.

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

Center and scale - since the predictors have significantly different scales, depending on the classification model, centering and scaling may improve classification results.

Log transformation - since predictors K, Ba and Fe are highly skewed, we can consider using a log transformation.

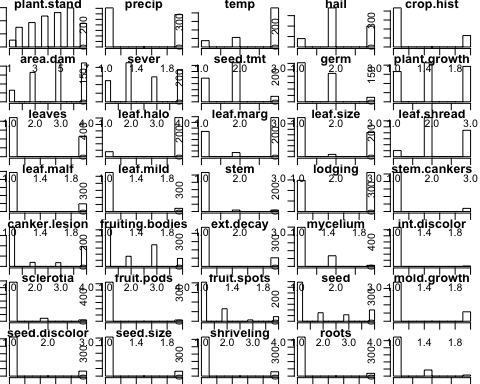
# 3.2

data("Soybean")

## 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?

Although based on the distributions several predictors only have a couple values, only three predictors meet the rules outlined in the chapter for degenerate variables: stem, int.discolor, fruit.pods

preds = colnames(Soybean[, 2:length(colnames(Soybean))])  
  
# check historgrams for each predictor   
par(mfrow = c(7, 5), mar = c(.5, .5, .5, .5))  
for (c in 2:length(colnames(Soybean))){  
 hist(as.numeric(Soybean[, c]),   
 main = preds[c]) # need to change margins to prevent error  
}



# calculate fraction of unique values and ratio of frequency between 1st and 2nd most prevalent value   
df = data.frame(pred = as.character(),  
 deg = as.character())  
for (c in 2:length(colnames((Soybean)))){  
 uniq\_frac = (length(unique(Soybean[, c])) / length(Soybean[, c])) < 0.1   
  
 # determine frequency of top 1 and 2 values   
 var1\_freq = sort(table(Soybean[, c]), decreasing = TRUE, useNA = 'ifany')[1]  
 var2\_freq = sort(table(Soybean[, c]), decreasing = TRUE, useNA = 'ifany')[2]  
 freq\_ratio= (var1\_freq / var2\_freq) > 20   
 uniq\_frac & freq\_ratio  
 df\_temp = data.frame(preds[c], uniq\_frac & freq\_ratio)  
 df = rbind(df, df\_temp)  
}  
colnames(df)= c('pred', 'deg')  
  
# return only those predictors that meet degenerate rules   
df[df['deg'] == TRUE, ]

## pred deg  
## 07 stem TRUE  
## 013 int.discolor TRUE  
## 015 fruit.pods TRUE

## 3.2.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?

Yes, the first table below shows the predictors by number of NAs, most to least.

The second table below shows the number of complete rows (i.e. no NAs) by Class. As shown, only phytophthora-rot has complete and incomplete records. All other classes have records that are all complete or are all missing values.

The third table below shows the number of NAs for each predictor for each Class. Here, we see that typically when a Class is missing data for a predictor, all observations in that Class are missing data for that particular predictor.

library(dplyr)  
library(tidyr)  
  
Soybean %>% select(-one\_of(c('Class'))) %>%   
 summarise\_each(funs(sum(is.na(.)))) %>%  
 gather(pred, n\_na) %>%   
 arrange(desc(n\_na))

## pred n\_na  
## 1 hail 121  
## 2 sever 121  
## 3 seed.tmt 121  
## 4 lodging 121  
## 5 germ 112  
## 6 leaf.mild 108  
## 7 fruiting.bodies 106  
## 8 fruit.spots 106  
## 9 seed.discolor 106  
## 10 shriveling 106  
## 11 leaf.shread 100  
## 12 seed 92  
## 13 mold.growth 92  
## 14 seed.size 92  
## 15 leaf.halo 84  
## 16 leaf.marg 84  
## 17 leaf.size 84  
## 18 leaf.malf 84  
## 19 fruit.pods 84  
## 20 precip 38  
## 21 stem.cankers 38  
## 22 canker.lesion 38  
## 23 ext.decay 38  
## 24 mycelium 38  
## 25 int.discolor 38  
## 26 sclerotia 38  
## 27 plant.stand 36  
## 28 roots 31  
## 29 temp 30  
## 30 crop.hist 16  
## 31 plant.growth 16  
## 32 stem 16  
## 33 date 1  
## 34 area.dam 1  
## 35 leaves 0

any\_na = Soybean %>% complete.cases()  
Soybean %>% mutate(any\_na = any\_na) %>%   
 group\_by(Class, any\_na) %>%   
 summarise(n = n()) %>%   
 spread(any\_na, n) %>%   
 arrange(`TRUE`, `FALSE`, Class) %>%  
 rename('has\_NAs' = `FALSE`, 'no\_NAs' = `TRUE`)

## Source: local data frame [19 x 3]  
## Groups: Class [19]  
##   
## Class has\_NAs no\_NAs  
## <fctr> <int> <int>  
## 1 phytophthora-rot 68 20  
## 2 bacterial-blight NA 20  
## 3 bacterial-pustule NA 20  
## 4 charcoal-rot NA 20  
## 5 diaporthe-stem-canker NA 20  
## 6 downy-mildew NA 20  
## 7 phyllosticta-leaf-spot NA 20  
## 8 powdery-mildew NA 20  
## 9 purple-seed-stain NA 20  
## 10 rhizoctonia-root-rot NA 20  
## 11 anthracnose NA 44  
## 12 brown-stem-rot NA 44  
## 13 alternarialeaf-spot NA 91  
## 14 frog-eye-leaf-spot NA 91  
## 15 brown-spot NA 92  
## 16 herbicide-injury 8 NA  
## 17 cyst-nematode 14 NA  
## 18 diaporthe-pod-&-stem-blight 15 NA  
## 19 2-4-d-injury 16 NA

Soybean %>% mutate(n = NaN) %>%  
 group\_by(Class) %>%  
 summarise\_each(funs(sum(is.na(.))))

## # A tibble: 19 × 37  
## Class date plant.stand precip temp hail  
## <fctr> <int> <int> <int> <int> <int>  
## 1 2-4-d-injury 1 16 16 16 16  
## 2 alternarialeaf-spot 0 0 0 0 0  
## 3 anthracnose 0 0 0 0 0  
## 4 bacterial-blight 0 0 0 0 0  
## 5 bacterial-pustule 0 0 0 0 0  
## 6 brown-spot 0 0 0 0 0  
## 7 brown-stem-rot 0 0 0 0 0  
## 8 charcoal-rot 0 0 0 0 0  
## 9 cyst-nematode 0 14 14 14 14  
## 10 diaporthe-pod-&-stem-blight 0 6 0 0 15  
## 11 diaporthe-stem-canker 0 0 0 0 0  
## 12 downy-mildew 0 0 0 0 0  
## 13 frog-eye-leaf-spot 0 0 0 0 0  
## 14 herbicide-injury 0 0 8 0 8  
## 15 phyllosticta-leaf-spot 0 0 0 0 0  
## 16 phytophthora-rot 0 0 0 0 68  
## 17 powdery-mildew 0 0 0 0 0  
## 18 purple-seed-stain 0 0 0 0 0  
## 19 rhizoctonia-root-rot 0 0 0 0 0  
## # ... with 31 more variables: crop.hist <int>, area.dam <int>,  
## # sever <int>, seed.tmt <int>, germ <int>, plant.growth <int>,  
## # leaves <int>, leaf.halo <int>, leaf.marg <int>, leaf.size <int>,  
## # leaf.shread <int>, leaf.malf <int>, leaf.mild <int>, stem <int>,  
## # lodging <int>, stem.cankers <int>, canker.lesion <int>,  
## # fruiting.bodies <int>, ext.decay <int>, mycelium <int>,  
## # int.discolor <int>, sclerotia <int>, fruit.pods <int>,  
## # fruit.spots <int>, seed <int>, mold.growth <int>, seed.discolor <int>,  
## # seed.size <int>, shriveling <int>, roots <int>, n <int>

## 3.2.c. Develop a strategy for handling missing data, either by eliminating predictors or imputation.

Since so many Classes consistently have missing data, one possibility is to create a dummy variable that indicates such a class. Another approach might be to use two different models, one for those with complete data and another for ones with incomplete data (and drop the empty predictors from the latter model).

I don't think imputation makes sense here because the missing values seem to be missing completely most of the time for a given class, so imputing based on other classes would just create a static (zero variance) value for that particular class and predictor combination.