hw#3.data624

Sang Yoon (Andy) Hwang

September 16, 2019

# Exercises

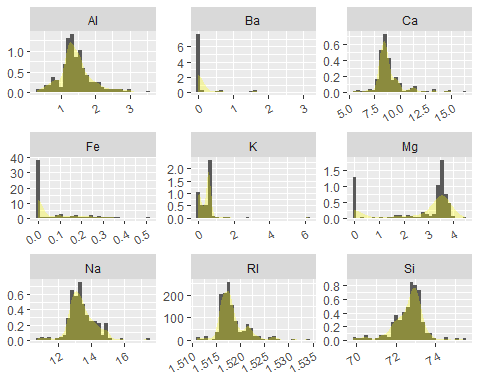
# 3.1. The UC Irvine Machine Learning Repository6 contains a data set related to glass identification. The data consist of 214 glass samples labeled as one of seven class categories. There are nine predictors, including the refractive index and percentages of eight elements: Na, Mg, Al, Si, K, Ca, Ba,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

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

## Univariate distributions

The plot below of a density plot overlayed on a histogram for all variables shows that more variables than those identified using the skewness statistics appear to have a skewed distribution. Some like Ca and RI may benefit from transformation, however other such as Ba and Fe may not benefit even after transformation as they are too far away from a normal distribution.

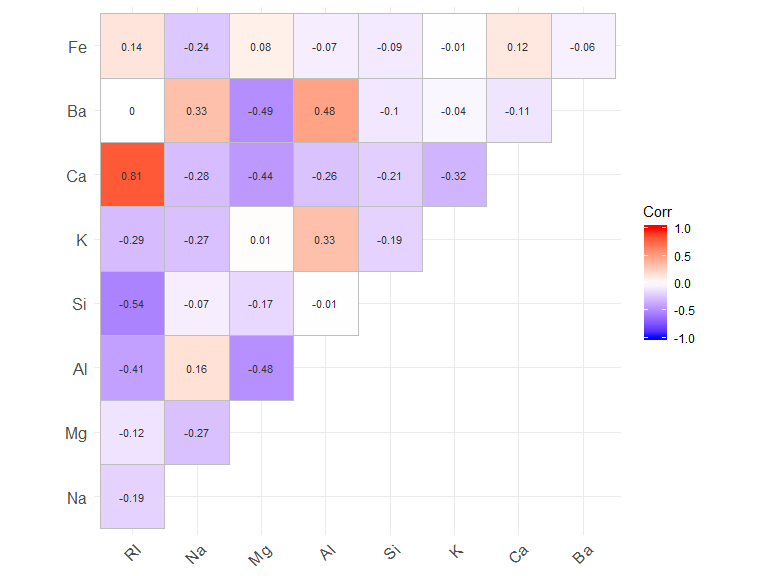
Glass %>%   
 # union(dfEval %>% mutate(TARGET\_WINS = as.numeric(NA))) %>%  
 keep(is.numeric) %>%   
 gather() %>%   
 group\_by(key) %>%   
 ggplot(data = ., aes(value)) +   
 geom\_histogram(bins = 30, aes(y = ..density..)) +  
 geom\_density(alpha = 0.3, color = NA, fill = 'yellow') +   
 #scale\_x\_continuous() +  
 #scale\_y\_continuous() +  
 # scale\_y\_continuous(labels = scales::percent) +  
 facet\_wrap(~key, scales = 'free') +   
 labs(x = NULL, y = NULL) +   
 theme(axis.text.x = element\_text(angle = 30, hjust = 1))



## Correlation matrix

The scatterplot matrix below shows the relationships, Pearson Correlation Coefficient, between all variables in the data. The first column which represents the relationships between the response variable and each predictor further supports the observations made from the scatterplots. The matrix also shows that there are relationships between the some of the predictors as well. The strongest correlation is between Ca and RI.

# Calculate pairwise pearson correlation and display as upper matrix plot  
Glass %>%   
 # union(dfEval %>% mutate(TARGET\_WINS = as.numeric(NA))) %>%  
 keep(is.numeric) %>%   
 cor(method = 'pearson', use = 'pairwise.complete.obs') %>%   
 ggcorrplot(corr = ., method = 'square', type = 'upper'  
 , lab = TRUE, lab\_size = 3, lab\_col = 'grey20')

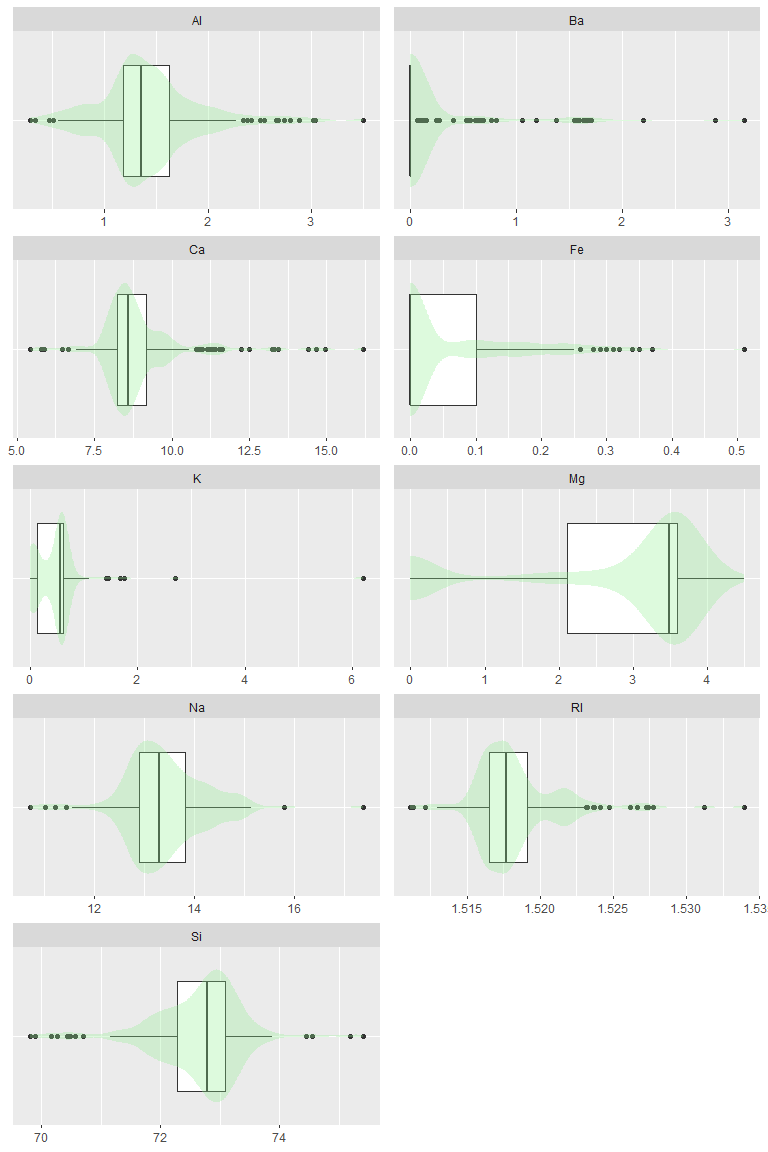


# (b) Do there appear to be any outliers in the data? Are any predictors skewed?

## Boxplot

The histogram and density show that some of the predictors have long tails indicative of outliers. The plot below shows a boxplot and violin plot for each variable. Many of the variables appear to have outliers with predictors like Ba, Fe and K appear to have the most potential outliers. How these outliers will be handled will be dependent on a mix of reference data and statistical techniques.

# boxplot with violin plot overlaid for all variables  
Glass %>%   
 # union(dfEval %>% mutate(TARGET\_WINS = as.numeric(NA))) %>%  
 keep(is.numeric) %>%   
 gather() %>%   
 group\_by(key) %>%   
 ggplot(data = ., aes(x = '', y = value)) +   
 geom\_boxplot() +   
 geom\_violin(alpha = 0.3, color = NA, fill = 'lightgreen') +   
 labs(x = NULL, y = NULL) +   
 theme(axis.ticks.y=element\_blank()) +   
 facet\_wrap(~key, scales = 'free', ncol = 2) +   
 coord\_flip()



## Summary statistics

The following tables are the summary statistics for all of the numeric variables in the dataset. They have been broken up into two tables for ease of reading. While some aspects of the distribution of the variables are easier seen in visualization. Two notable statistics are that K has a strongly positive skew and kurtosis (thus expecting lots of outliers for this variable) which may need to be addressed before developing any type of model; other predictors show skew and kurtosis, but to a lesser degree.

From <https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm>, we know Normal Distirubtion would have a skewness near 0 and a kurtosis higher than 3.

statFuns <-   
 funs(missing = sum(is.na(.))  
 , min = min(., na.rm = TRUE)  
 , Q1 = quantile(., .25, na.rm = TRUE)  
 , mean = mean(., na.rm = TRUE)  
 , median = median(., na.rm = TRUE)  
 , Q3 = quantile(., .75, na.rm = TRUE)  
 , max = max(., na.rm = TRUE)  
 , sd = sd(., na.rm = TRUE)  
 , mad = mad(., na.rm = TRUE)  
 , skewness = skewness(., na.rm = TRUE)  
 , kurtosis = kurtosis(., na.rm = TRUE)  
 )  
  
# Create data frame of basic summary stats  
dfsum <-   
 Glass %>%   
 # union(dfEval %>% mutate(TARGET\_WINS = as.numeric(NA))) %>%   
 keep(is.numeric) %>%   
 summarise\_all(statFuns) %>%   
 gather() %>%   
 separate(key, c('metric', 'stat'), sep = '(\_)(?!.\*\_)') %>%   
 spread(stat, value) %>%   
 dplyr::select(metric, names(statFuns))  
  
# print summary stats as two tables for easier reading  
  
# stats for table 1  
vTbl1Stats <- c('metric', 'min', 'Q1', 'mean', 'median', 'Q3', 'max')  
  
pandoc.table(dfsum[, vTbl1Stats], missing = '-')

##   
## ------------------------------------------------------------  
## metric min Q1 mean median Q3 max   
## -------- ------- -------- --------- -------- ------- -------  
## Al 0.29 1.19 1.445 1.36 1.63 3.5   
##   
## Ba 0 0 0.175 0 0 3.15   
##   
## Ca 5.43 8.24 8.957 8.6 9.172 16.19   
##   
## Fe 0 0 0.05701 0 0.1 0.51   
##   
## K 0 0.1225 0.4971 0.555 0.61 6.21   
##   
## Mg 0 2.115 2.685 3.48 3.6 4.49   
##   
## Na 10.73 12.91 13.41 13.3 13.83 17.38   
##   
## RI 1.511 1.517 1.518 1.518 1.519 1.534   
##   
## Si 69.81 72.28 72.65 72.79 73.09 75.41   
## ------------------------------------------------------------

pandoc.table(dfsum[  
 , c('metric', setdiff(colnames(dfsum), vTbl1Stats))], missing = '-')

##   
## --------------------------------------------------------------  
## metric missing sd mad skewness kurtosis   
## -------- --------- ---------- ---------- ---------- ----------  
## Al 0 0.4993 0.3113 0.8946 1.938   
##   
## Ba 0 0.4972 0 3.369 12.08   
##   
## Ca 0 1.423 0.6598 2.018 6.41   
##   
## Fe 0 0.09744 0 1.73 2.52   
##   
## K 0 0.6522 0.1705 6.46 52.87   
##   
## Mg 0 1.442 0.3039 -1.136 -0.4527   
##   
## Na 0 0.8166 0.6449 0.4478 2.898   
##   
## RI 0 0.003037 0.001875 1.603 4.717   
##   
## Si 0 0.7745 0.5708 -0.7202 2.816   
## --------------------------------------------------------------

### Statistical approach - removing outliers

Unknown outliers will be determined using the Median Absolute Deviation (MAD); as a metric of central tendency, median is less effected by extreme values than mean. Anything that exceeds three MAD of the median will be considered an outlier and dropped. Outlier removed data will be stored in a seperate dataframe. 59% of data are removed after MAD removal and it is a quite huge loss. It is not reasonable to consider adressing 59% of loss through outlier removal - we will keep using original dataframe for the next question.

# create function to evaluate whether a value is an outlier  
  
Glass\_num <- Glass %>% keep(is.numeric)  
Glass\_num$index <- c(1:nrow(Glass\_num))  
  
madOutlier <- function(x, cutoff = 3){  
 madrange <- mad(x)\*c(-1,1)\*cutoff  
 lower <- x < median(x) + madrange[1]  
 upper <- x > median(x) + madrange[2]  
 return(lower | upper)  
}  
  
# store outliers in a dataframe  
dfOutlier <- as.data.frame(apply(Glass\_num, 2, madOutlier))  
  
# retain INDEX of outliers to remove  
outlierIndex <-   
 Glass\_num$index[  
 apply(  
 dfOutlier  
 , 1, any)]  
  
# df after outliers are removed  
  
Glass\_num\_mad <- Glass\_num[!(Glass\_num$index %in% outlierIndex),]  
#Glass\_num\_mad  
round( length(outlierIndex) / nrow(Glass\_num) \* 100, 2)

## [1] 59.35

# potential outliers?  
# length(outlierIndex)

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

## Transformations - BoxCox & Yeo-Johnson

The data contains enough observations that some deviation from normal distributions are acceptable.

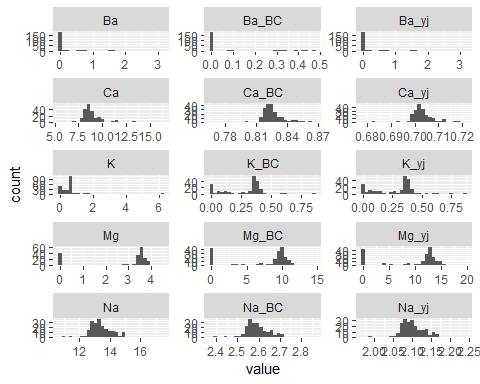
It is known that Box-Cox does not allow zero values so that a constant = 1 was added for zero\_inflated variables such as Ba, K, Mg and Fe before applying the method. It seems like except for Al, Ca and Na, most of variables did not get normalized from Box-Cox.

Yeo-Johnson works like Box-Cox but the difference is that it allows zero/negative values.

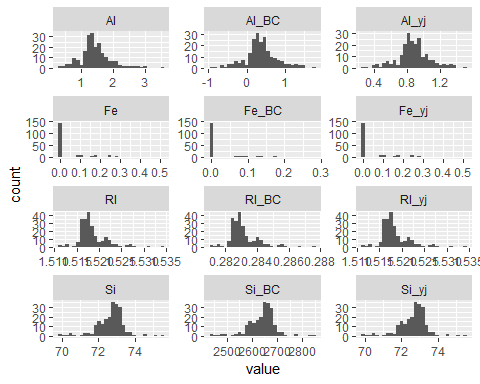
From below stat summary, we know that Yeo-Johnson did not perform any transformation for zero-inflated variables such as Ba, Fe and already fairly normalized variables such as Si, Ri.

In general, Yeo-Johnson works better in transformation for most variables than Box-Cox.

# apply box-cox power transformations to heavily skewed variables  
# TODO revise to use either lapply & function or purrr map commands  
  
# apply yeo-johnson -> it works for negative or zero values  
pre <- preProcess(Glass, method = 'YeoJohnson')  
yj <- predict(pre,Glass)  
  
Glass$Ba\_yj <- yj$Ba  
Glass$K\_yj <- yj$K  
Glass$Mg\_yj <- yj$Mg  
Glass$Fe\_yj <- yj$Fe  
Glass$Al\_yj <- yj$Al  
Glass$Ca\_yj <- yj$Ca  
Glass$Na\_yj <- yj$Na  
Glass$RI\_yj <- yj$RI  
Glass$Si\_yj <- yj$Si  
  
  
# Since these variables have zero value adding 1 to allow box-cox transform  
   
 Ba = BoxCoxTrans(Glass$Ba + 1) #as.numeric(powerTransform(Glass$Ba + 1)$lambda)  
 K = BoxCoxTrans(Glass$K + 1) #as.numeric(powerTransform(Glass$K + 1)$lambda)  
 Mg = BoxCoxTrans(Glass$Mg + 1) #as.numeric(powerTransform(Glass$Mg + 1)$lambda)  
 Fe = BoxCoxTrans(Glass$Fe + 1) #as.numeric(powerTransform(Glass$Fe + 1)$lambda)  
   
 Al = BoxCoxTrans(Glass$Al) #as.numeric(powerTransform(Glass$Al)$lambda)  
 Ca = BoxCoxTrans(Glass$Ca) #as.numeric(powerTransform(Glass$Ca)$lambda)  
 Na = BoxCoxTrans(Glass$Na) #as.numeric(powerTransform(Glass$Na)$lambda)  
 RI = BoxCoxTrans(Glass$RI) #as.numeric(powerTransform(Glass$RI)$lambda)  
 Si = BoxCoxTrans(Glass$Si) #as.numeric(powerTransform(Glass$Si)$lambda)  
  
  
Glass$Ba\_BC <-   
 predict(Ba,Glass$Ba+1)  
  
Glass$Ca\_BC <-   
 predict(Ca,Glass$Ca)  
  
Glass$K\_BC <-   
 predict(K,Glass$K+1)  
  
Glass$Mg\_BC <-   
 predict(Mg,Glass$Mg+1)  
  
Glass$Na\_BC <-   
 predict(Na,Glass$Na)  
  
Glass$RI\_BC <-   
 predict(RI,Glass$RI)  
  
Glass$Si\_BC <-   
 predict(Si,Glass$Si)  
  
Glass$Fe\_BC <-   
 predict(Fe,Glass$Fe+1)  
  
Glass$Al\_BC <-   
 predict(Al,Glass$Al)  
  
  
  
# plot histogram to show impact of box-cox & YeoJohnson  
Glass %>%   
 dplyr::select(matches('Ba|Ca|K|Mg|Na')) %>%   
 gather() %>%   
 ggplot(data = ., aes(value)) +   
 geom\_histogram(bins = 30) +   
 facet\_wrap(~key, ncol = 3, scales = 'free')



Glass %>%   
 dplyr::select(matches('RI|Si|Fe|Al')) %>%   
 gather() %>%   
 ggplot(data = ., aes(value)) +   
 geom\_histogram(bins = 30) +   
 facet\_wrap(~key, ncol = 3, scales = 'free')



# Let's re-run stat summary gain  
  
# Create data frame of basic summary stats  
dfsum <-   
 Glass %>%   
 # union(dfEval %>% mutate(TARGET\_WINS = as.numeric(NA))) %>%   
 keep(is.numeric) %>%   
 summarise\_all(statFuns) %>%   
 gather() %>%   
 separate(key, c('metric', 'stat'), sep = '(\_)(?!.\*\_)') %>%   
 spread(stat, value) %>%   
 dplyr::select(metric, names(statFuns))  
  
# print summary stats as two tables for easier reading  
  
statFuns <-   
 funs(missing = sum(is.na(.))  
 , min = min(., na.rm = TRUE)  
 , Q1 = quantile(., .25, na.rm = TRUE)  
 , mean = mean(., na.rm = TRUE)  
 , median = median(., na.rm = TRUE)  
 , Q3 = quantile(., .75, na.rm = TRUE)  
 , max = max(., na.rm = TRUE)  
 , sd = sd(., na.rm = TRUE)  
 , mad = mad(., na.rm = TRUE)  
 , skewness = skewness(., na.rm = TRUE)  
 , kurtosis = kurtosis(., na.rm = TRUE)  
 )  
  
# Create data frame of basic summary stats  
dfsum <-   
 Glass %>%   
 # union(dfEval %>% mutate(TARGET\_WINS = as.numeric(NA))) %>%   
 keep(is.numeric) %>%   
 summarise\_all(statFuns) %>%   
 gather() %>%   
 separate(key, c('metric', 'stat'), sep = '(\_)(?!.\*\_)') %>%   
 spread(stat, value) %>%   
 dplyr::select(metric, names(statFuns))  
  
# print summary stats as two tables for easier reading  
  
# stats for table 1  
vTbl1Stats <- c('metric', 'min', 'Q1', 'mean', 'median', 'Q3', 'max')  
  
pandoc.table(dfsum[, vTbl1Stats], missing = '-')

##   
## ----------------------------------------------------------------  
## metric min Q1 mean median Q3 max   
## -------- -------- -------- --------- -------- --------- --------  
## Al 0.29 1.19 1.445 1.36 1.63 3.5   
##   
## Al\_BC -0.923 0.1817 0.3685 0.3324 0.5534 1.742   
##   
## Al\_yj 0.2547 0.7849 0.8753 0.8598 0.9685 1.508   
##   
## Ba 0 0 0.175 0 0 3.15   
##   
## Ba\_BC 0 0 0.05545 0 0 0.471   
##   
## Ba\_yj 0 0 0.175 0 0 3.15   
##   
## Ca 5.43 8.24 8.957 8.6 9.172 16.19   
##   
## Ca\_BC 0.7677 0.8197 0.8256 0.8238 0.8297 0.8666   
##   
## Ca\_yj 0.6778 0.7007 0.7033 0.7025 0.7051 0.7212   
##   
## Fe 0 0 0.05701 0 0.1 0.51   
##   
## Fe\_BC 0 0 0.04289 0 0.08678 0.2807   
##   
## Fe\_yj 0 0 0.05701 0 0.1 0.51   
##   
## K 0 0.1225 0.4971 0.555 0.61 6.21   
##   
## K\_BC 0 0.1091 0.2771 0.3569 0.3789 0.8613   
##   
## K\_yj 0 0.1092 0.2781 0.3581 0.3802 0.8709   
##   
## Mg 0 2.115 2.685 3.48 3.6 4.49   
##   
## Mg\_BC 0 4.353 7.323 9.535 10.08 14.57   
##   
## Mg\_yj 0 5.231 9.486 12.35 13.13 19.72   
##   
## Na 10.73 12.91 13.41 13.3 13.83 17.38   
##   
## Na\_BC 2.373 2.558 2.594 2.588 2.626 2.855   
##   
## Na\_yj 1.973 2.079 2.1 2.096 2.118 2.245   
##   
## RI 1.511 1.517 1.518 1.518 1.519 1.534   
##   
## RI\_BC 0.281 0.2826 0.2831 0.2829 0.2833 0.2875   
##   
## RI\_yj 1.511 1.517 1.518 1.518 1.519 1.534   
##   
## Si 69.81 72.28 72.65 72.79 73.09 75.41   
##   
## Si\_BC 2436 2612 2639 2649 2670 2843   
##   
## Si\_yj 69.81 72.28 72.65 72.79 73.09 75.41   
## ----------------------------------------------------------------

pandoc.table(dfsum[  
 , c('metric', setdiff(colnames(dfsum), vTbl1Stats))], missing = '-')

##   
## -----------------------------------------------------------------  
## metric missing sd mad skewness kurtosis   
## -------- --------- ----------- ----------- ----------- ----------  
## Al 0 0.4993 0.3113 0.8946 1.938   
##   
## Al\_BC 0 0.4134 0.2644 0.09106 1.168   
##   
## Al\_yj 0 0.2005 0.1316 0.0002128 0.9978   
##   
## Ba 0 0.4972 0 3.369 12.08   
##   
## Ba\_BC 0 0.131 0 2.136 2.92   
##   
## Ba\_yj 0 0.4972 0 3.369 12.08   
##   
## Ca 0 1.423 0.6598 2.018 6.41   
##   
## Ca\_BC 0 0.01254 0.00717 -0.194 4.291   
##   
## Ca\_yj 0 0.00553 0.003179 -0.2064 4.247   
##   
## Fe 0 0.09744 0 1.73 2.52   
##   
## Fe\_BC 0 0.06896 0 1.327 0.453   
##   
## Fe\_yj 0 0.09744 0 1.73 2.52   
##   
## K 0 0.6522 0.1705 6.46 52.87   
##   
## K\_BC 0 0.1711 0.06567 -0.08698 0.1301   
##   
## K\_yj 0 0.1721 0.06618 -0.07082 0.1747   
##   
## Mg 0 1.442 0.3039 -1.136 -0.4527   
##   
## Mg\_BC 0 4.161 1.392 -0.9274 -0.7869   
##   
## Mg\_yj 0 5.467 1.984 -0.8771 -0.8481   
##   
## Na 0 0.8166 0.6449 0.4478 2.898   
##   
## Na\_BC 0 0.06062 0.04903 0.03385 2.5   
##   
## Na\_yj 0 0.03426 0.02774 -0.008848 2.49   
##   
## RI 0 0.003037 0.001875 1.603 4.717   
##   
## RI\_BC 0 0.0008634 0.0005372 1.566 4.553   
##   
## RI\_yj 0 0.003037 0.001875 1.603 4.717   
##   
## Si 0 0.7745 0.5708 -0.7202 2.816   
##   
## Si\_BC 0 56.06 41.55 -0.6509 2.781   
##   
## Si\_yj 0 0.7745 0.5708 -0.7202 2.816   
## -----------------------------------------------------------------

# 3.2. 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.

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

## Frequency distributions

The variables with degenrate distributions are variables with “zero variance” - in fact, variables that have one unique value with lots of zeroes.

Characteristics of “zero variance”:

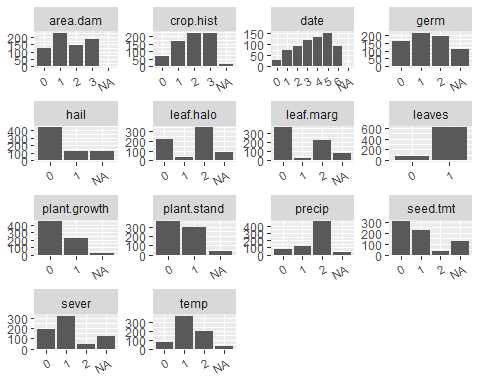
1. The fraction of unique values over the sample size is low (say 10%).
2. The ratio of the frequency of the most prevalent value to the frequency of the second most prevalent value is large (say around 20)

Those variables are leaf.mild, mycelium and sclerotia.

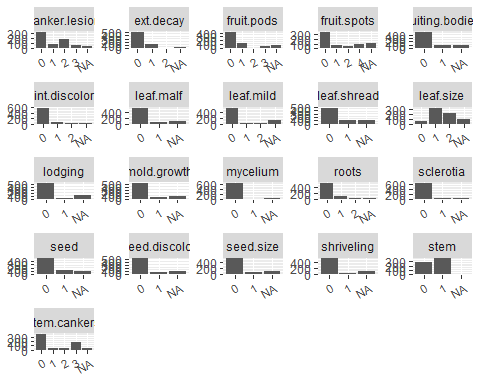
paste("Degenerate variables:", paste(names(Soybean[,nearZeroVar(Soybean)]),collapse = ', '))

## [1] "Degenerate variables: leaf.mild, mycelium, sclerotia"

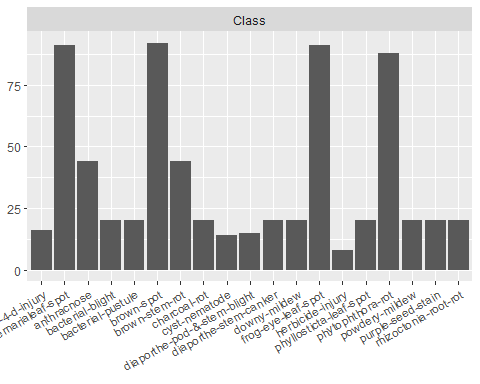
Soybean[2:15] %>%   
 # union(dfEval %>% mutate(TARGET\_WINS = as.numeric(NA))) %>%  
 #keep(is.numeric) %>%   
 #select(-Class) %>%  
 gather() %>%   
 group\_by(key) %>%   
 ggplot(data = ., aes(value)) +   
 stat\_count() +  
 #geom\_density(alpha = 0.3, color = NA, fill = 'yellow') +   
 #scale\_x\_continuous() +  
 #scale\_y\_continuous() +  
 # scale\_y\_continuous(labels = scales::percent) +  
 facet\_wrap(~key, scales = 'free') +   
 labs(x = NULL, y = NULL) +   
 theme(axis.text.x = element\_text(angle = 30, hjust = 1))



Soybean[16:length(Soybean)] %>%   
 # union(dfEval %>% mutate(TARGET\_WINS = as.numeric(NA))) %>%  
 #keep(is.numeric) %>%   
 #select(-Class) %>%  
 gather() %>%   
 group\_by(key) %>%   
 ggplot(data = ., aes(value)) +   
 stat\_count() +  
 #geom\_density(alpha = 0.3, color = NA, fill = 'yellow') +   
 #scale\_x\_continuous() +  
 #scale\_y\_continuous() +  
 # scale\_y\_continuous(labels = scales::percent) +  
 facet\_wrap(~key, scales = 'free') +   
 labs(x = NULL, y = NULL) +   
 theme(axis.text.x = element\_text(angle = 30, hjust = 1))



Soybean[1] %>%   
 # union(dfEval %>% mutate(TARGET\_WINS = as.numeric(NA))) %>%  
 #keep(is.numeric) %>%   
 #select(Class) %>%  
 gather() %>%   
 group\_by(key) %>%   
 ggplot(data = ., aes(value)) +   
 stat\_count() +  
 #geom\_density(alpha = 0.3, color = NA, fill = 'yellow') +   
 #scale\_x\_continuous() +  
 #scale\_y\_continuous() +  
 # scale\_y\_continuous(labels = scales::percent) +  
 facet\_wrap(~key, scales = 'free') +   
 labs(x = NULL, y = NULL) +   
 theme(axis.text.x = element\_text(angle = 30, hjust = 1))

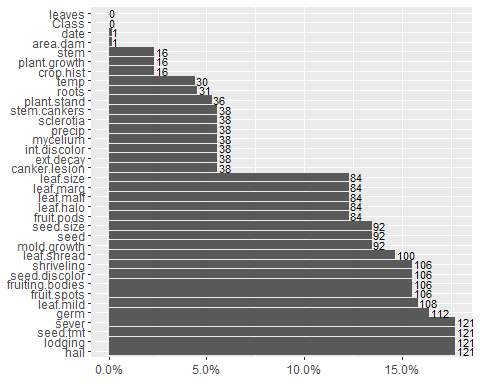


# (b) Roughly 18 % of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?

### Missing data

In review of the missing statistics sever, seed.tmt , lodging and hail have approximately 17.7% of the data missing. It is not immediately apparent whether these missing values are not applicable or actually missing. If it is the latter, the proportion of missing data for these four variables is reasonable to consider addressing through substitution or imputation. (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3701793/>)

# create lists of desired summary stats for calculation  
statFuns <-   
 funs(missing = sum(is.na(.))  
 )  
  
  
# Create data frame of basic summary stats  
dfsum <-   
 Soybean %>%   
 # union(dfEval %>% mutate(TARGET\_WINS = as.numeric(NA))) %>%   
 #keep(is.numeric) %>%   
 summarise\_all(statFuns) %>%   
 gather() %>%   
 separate(key, c('metric', 'stat'), sep = '(\_)(?!.\*\_)') %>%   
 spread(stat, value) %>%   
 dplyr::select(metric, names(statFuns))  
  
  
  
# create bar plot of percent of missing data, add counts as text labels  
dfsum %>%   
 group\_by(metric) %>%   
 mutate(miss\_perc = missing / !!nrow(Soybean)) %>%   
 dplyr::select(metric,missing, miss\_perc) %>%   
 ggplot(data = ., aes(x = reorder(metric, -miss\_perc) , y = miss\_perc)) +   
 geom\_bar(stat = 'identity') +  
 coord\_flip() +   
 geom\_text(aes(label = missing), hjust = -0.1, size = 3) +   
 labs(x = NULL, y = NULL, Title = '% Missing') +   
 theme(legend.position = 'none') +   
 scale\_y\_continuous(labels = scales::percent)



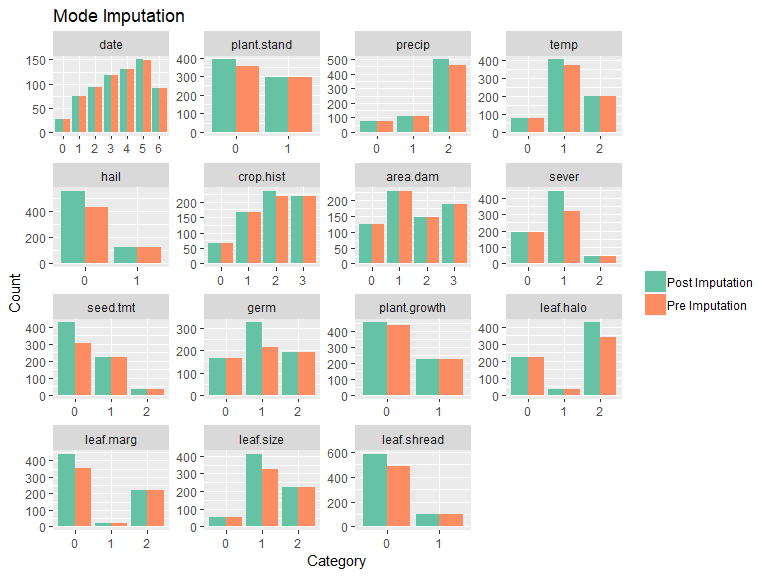
#dfsum[2][which.max(dfsum$missing),]/nrow(Soybean)  
#dfsum[dfsum$missing == 121,]

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

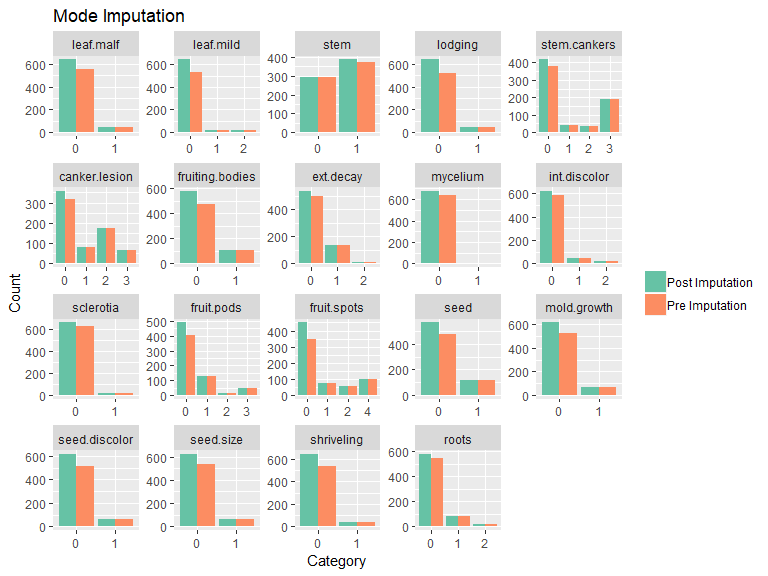
## Imputation - by mode & KNN

Mode and KNN imputation for the missing categorical values would be used. As you can see, when Mode is used, since it replaces NAs with the most occuring value, mode category can be highly over-represented, while all others are underrepresented. In other words, the distribution of imputed data can be highly biased. When KNN is used, however, it minimizes the biase as categories are fairly evenly represented.

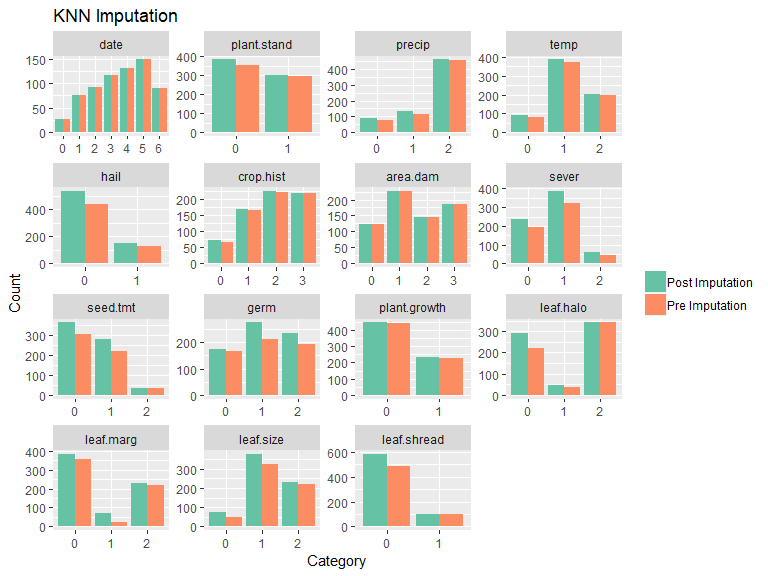
# subset columns for imputation  
dfimpute <- subset(Soybean, select = - c(Class, leaves) )  
df <- dfimpute  
  
# perform mode imputation  
df[, sapply(df, function(x) !is.numeric(x))] <- apply(df[, sapply(df, function(x) !is.numeric(x))], 2, function(x) {x[is.na(x)] <- names(sort(table(x), decreasing = TRUE)[1]); x})  
  
  
# create dataframe for plotting imputation impact  
plotlist <- list()  
  
for (i in 1:length(names(df))){  
   
 t = nrow(unique(df[i]))  
   
 name <- names(df[i])  
 missingness <- c(rep("Pre Imputation", t), rep("Post Imputation", t)) # Pre/post imputation  
 Category <- as.factor(rep(names(table(dfimpute[i])), 2)) # Categories  
 Count <- c(as.numeric(table(dfimpute[i])), as.numeric(table(df[i]))) # Count of categories  
   
 plotlist[[i]] <- data.frame(name, missingness, Category, Count) }  
  
alldata <- do.call(rbind, plotlist)  
  
plot1 = alldata[alldata$name %in% names(df)[1:15],]  
plot2 = alldata[alldata$name %in% names(df)[16:length(df)],]  
  
  
# let's see Pre vs Post imputation  
ggplot(plot1, aes(Category, Count, fill = missingness)) + # Create plot  
 geom\_bar(stat = "identity", position = "dodge") +   
 scale\_fill\_brewer(palette = "Set2") +  
 facet\_wrap(~name, scales = 'free') +  
 theme(legend.title = element\_blank()) +  
 ggtitle("Mode Imputation")



ggplot(plot2, aes(Category, Count, fill = missingness)) + # Create plot  
 geom\_bar(stat = "identity", position = "dodge") +   
 scale\_fill\_brewer(palette = "Set2") +  
 facet\_wrap(~name, scales = 'free') +  
 theme(legend.title = element\_blank()) +  
 ggtitle("Mode Imputation")



# perform KNN imputation  
knnOutput <- knnImputation(dfimpute)  
df <- knnOutput  
  
# create dataframe for plotting imputation impact  
plotlist <- list()  
  
for (i in 1:length(names(df))){  
   
 t = nrow(unique(df[i]))  
   
 name <- names(df[i])  
 missingness <- c(rep("Pre Imputation", t), rep("Post Imputation", t)) # Pre/post imputation  
 Category <- as.factor(rep(names(table(dfimpute[i])), 2)) # Categories  
 Count <- c(as.numeric(table(dfimpute[i])), as.numeric(table(df[i]))) # Count of categories  
   
 plotlist[[i]] <- data.frame(name, missingness, Category, Count) }  
  
alldata <- do.call(rbind, plotlist)  
  
plot1 = alldata[alldata$name %in% names(df)[1:15],]  
plot2 = alldata[alldata$name %in% names(df)[16:length(df)],]  
  
# let's see Pre vs Post imputation  
ggplot(plot1, aes(Category, Count, fill = missingness)) + # Create plot  
 geom\_bar(stat = "identity", position = "dodge") +   
 scale\_fill\_brewer(palette = "Set2") +  
 facet\_wrap(~name, scales = 'free') +  
 theme(legend.title = element\_blank()) +  
 ggtitle("KNN Imputation")



ggplot(plot2, aes(Category, Count, fill = missingness)) + # Create plot  
 geom\_bar(stat = "identity", position = "dodge") +   
 scale\_fill\_brewer(palette = "Set2") +  
 facet\_wrap(~name, scales = 'free') +  
 theme(legend.title = element\_blank()) +  
 ggtitle("KNN Imputation")

