Assignment 1

Course: MA 5790

Question 3.1

The UC Irvine Machine Learning Repository contains a data set related to glass identification. The data consist of 214 glass samples labeled as one of seven class categories. Nine predictors include 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 and the relationships between predictors.

The dataset has ten variables, of which nine are the predictors, which are all numeric. The response is stored in the column labeled 'Type.'

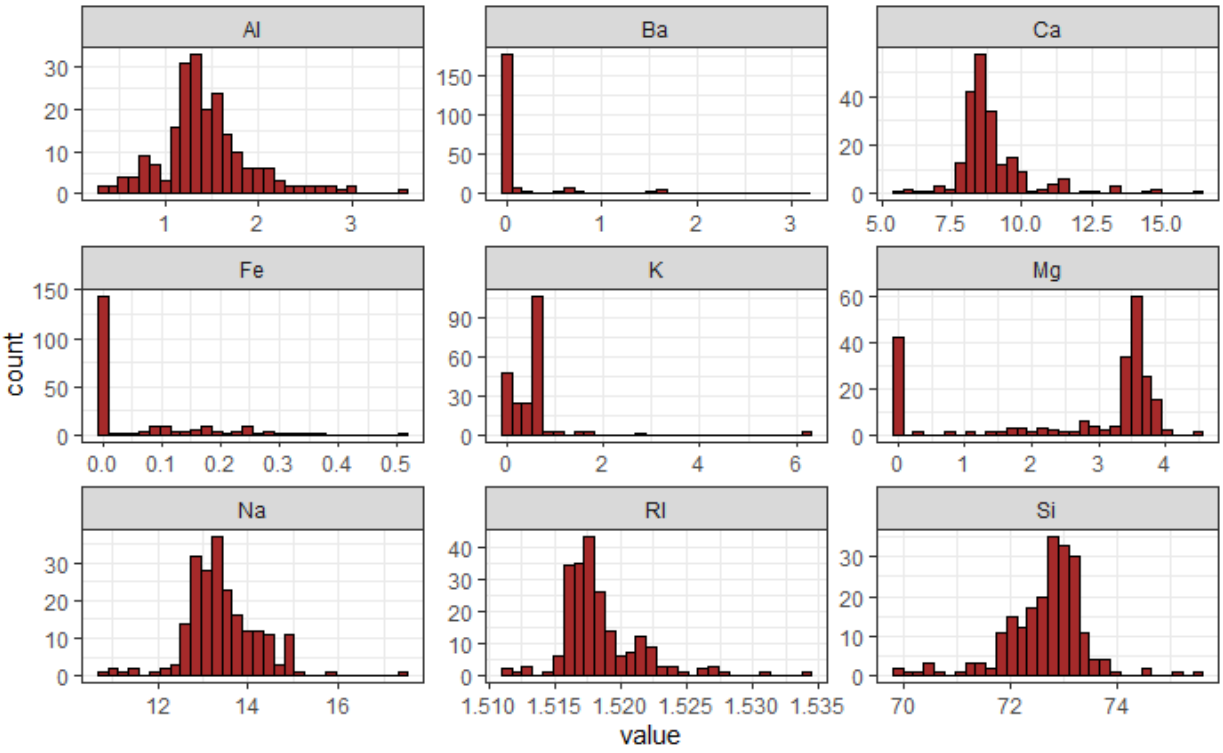

Figure 1. Determining symmetry of predictors using histogram

From Figure 1 above, Na, RI, Si, Al, and Ca are relatively symmetric; hence, the distribution is close to normal. Mg indicates two peaks, while Ba, Fe, and K have singular peaks around the low values of their distribution.

We use the correlation plot to visualize the linear relationships between the predictors. This technique is also known as the Pearson correlation.

The strongest positive correlation is between Ca, and RI shown in Figure 2; weak positive correlations exist between Al and Ba, Ba and Na, and K and Al. The strongest negative correlation is between Si and RI.

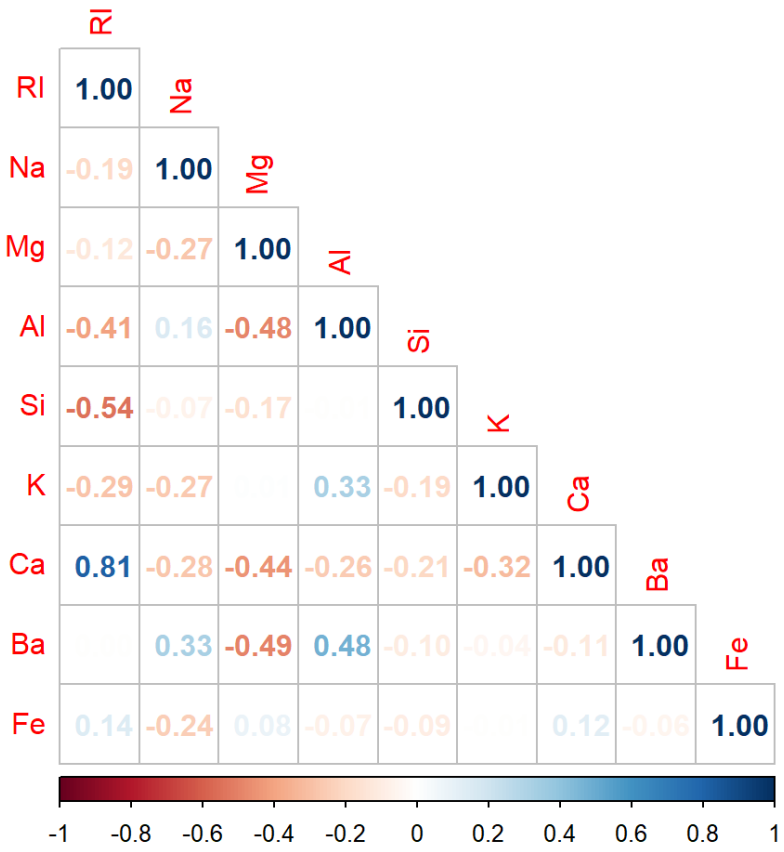

Figure 2. Linear correlation between predictors

b. Do there appear to be any outliers in the data? Are any predictors skewed? (Please calculate the skewness values for the predictors, summarize these values using a table with interpretations).

We can determine the presence of outliers in our data by plotting boxplots of the predictors.

From Figure 3, most predictors have outliers, especially for 'Ba'. All the values fall within an acceptable range for Mg, indicating no outliers.

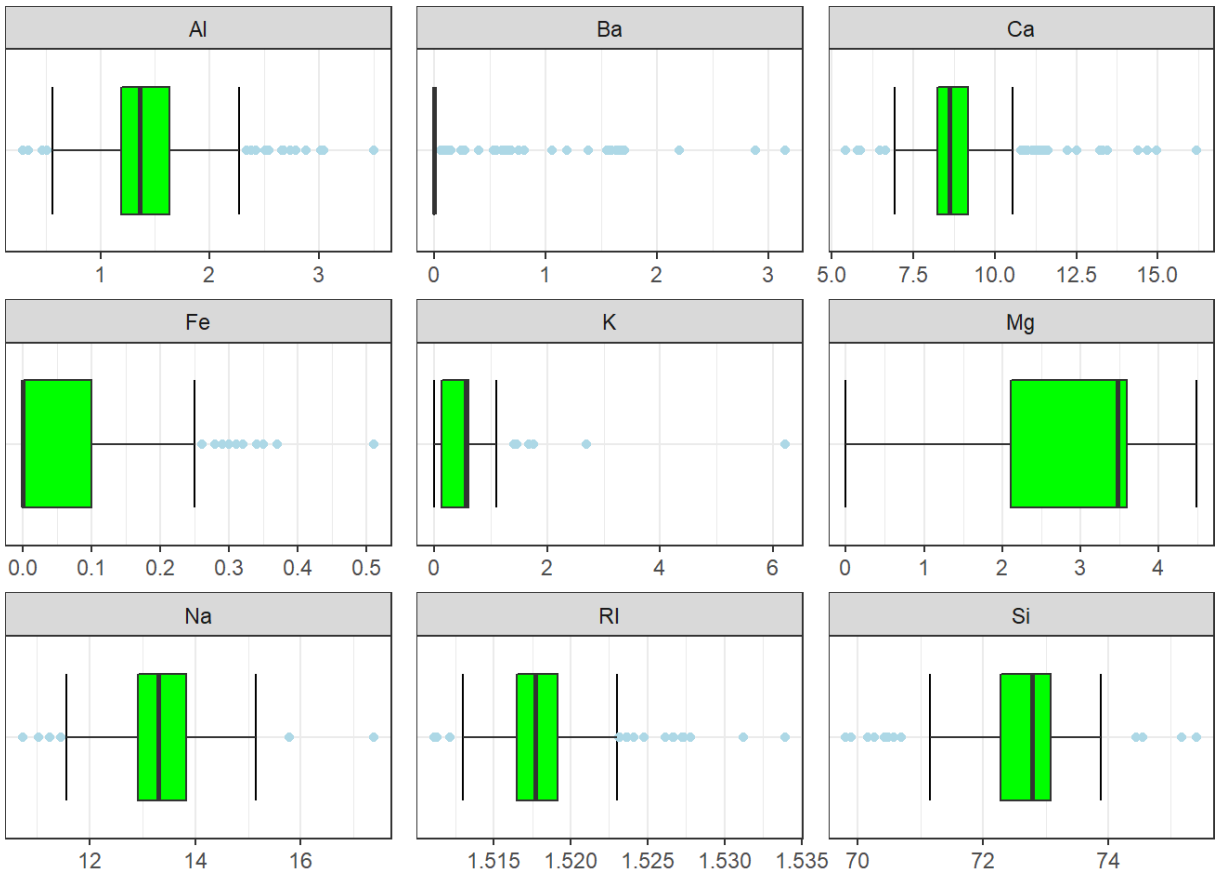

Figure 3. Box plots of the predictors showing the presence of outliers in the data

Table 1. Summary symmetry statistics of predictors

Predictor	skew	Interpretation of skewness	kurtosis
RI	1.603	Highly right-skewed	4.717
Na	0.448	Approximately symmetric	2.898
Mg	-1.136	Highly left-skewed	-0.453
Al	0.895	Highly Right skewed	1.938
Si	-0.72	Highly Left skewed	2.816
K	6.46	Highly right-skewed	52.867
Ca	2.018	Highly right-skewed	6.41
Ba	3.369	Highly right-skewed	12.08
Fe	1.73	Highly right-skewed	2.52

From the statistics calculated above, Table 1, all the predictors are skewed with varying degrees of symmetry. The expected value for perfect symmetry (kurtosis) is 3.

c. Are there any relevant transformations of one or more predictors that might improve the classification model? (Please perform at least two transformations based on your observations of the predictors; use visualizations of before and after the transformations and make comments).

First, we could scale and center all our predictors to reduce the effects of the high magnitudes found in some predictors. For example, relatively high magnitudes are found in K, Si, and Na. Also, using BoxCox, we can reduce the skewness of our predictors, especially highly skewed predictors.

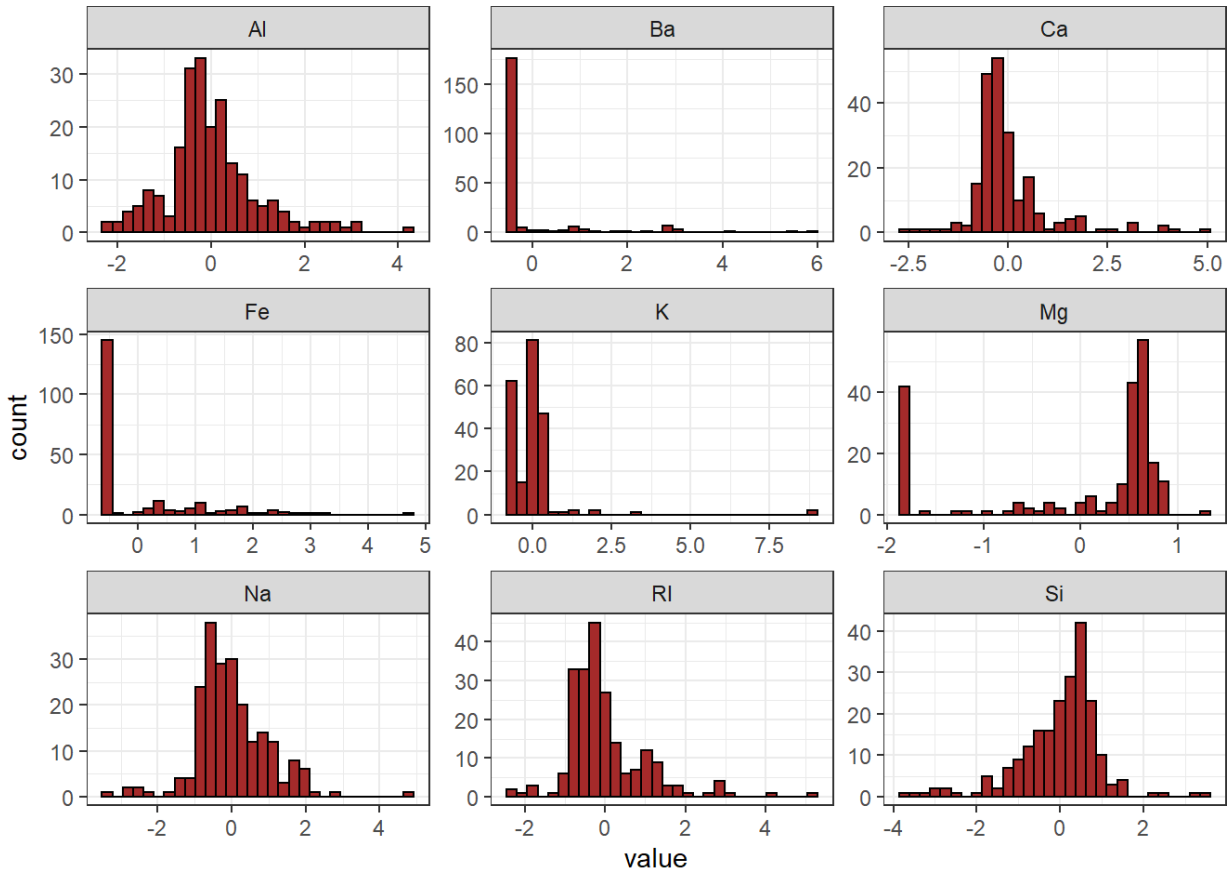

Figure 4. Predictors symmetry after centering and scaling

Centering and scaling, Figure 4 did not improve some of the predictors, especially predictors with bimodal peaks.

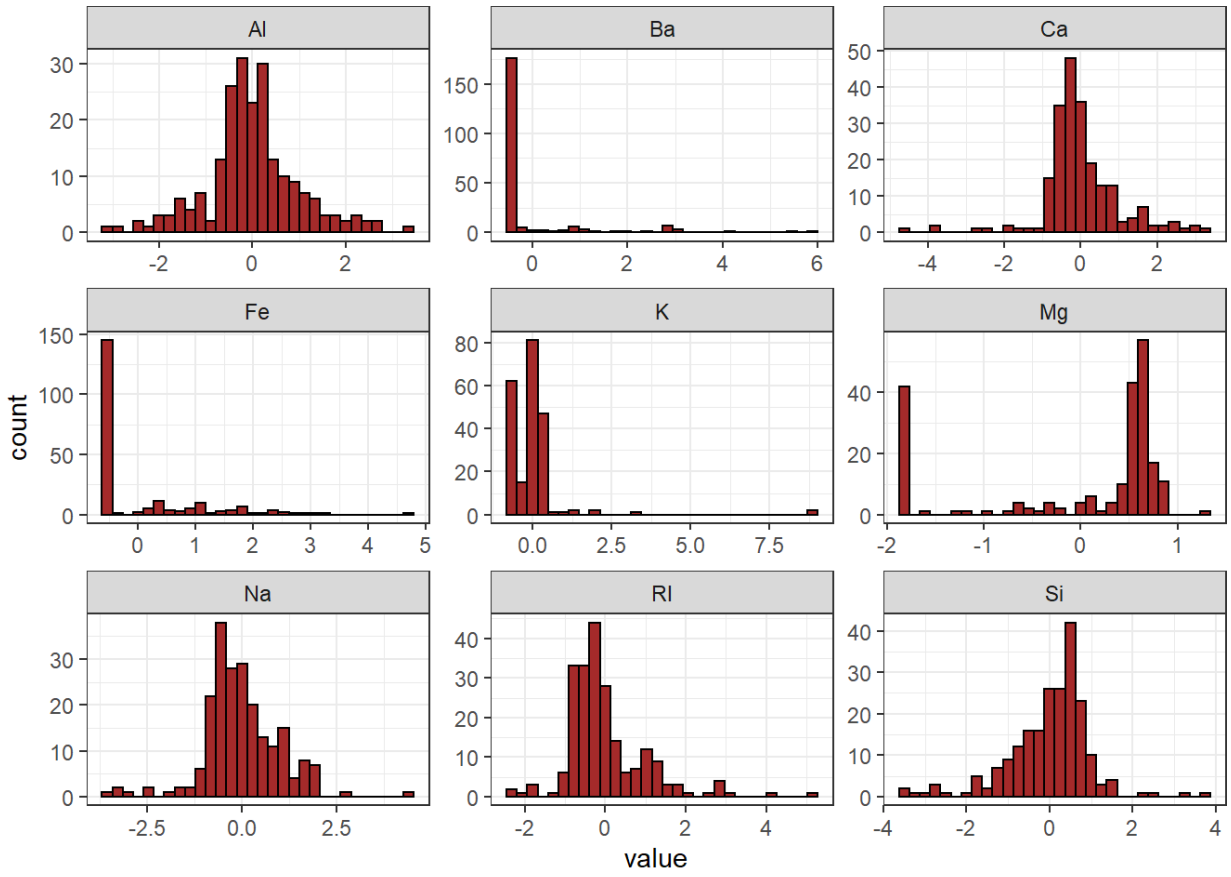

Figure 5. The symmetry of predictors after BoxCox transformation

There are still predictors with bimodal peaks in our data. The BoxCox transformation did not improve the distribution of some of the predictors since the lambda could not be found for some of the predictors. Another transformation worth trying will be the spatial sign for the outliers and PCA for skewness.

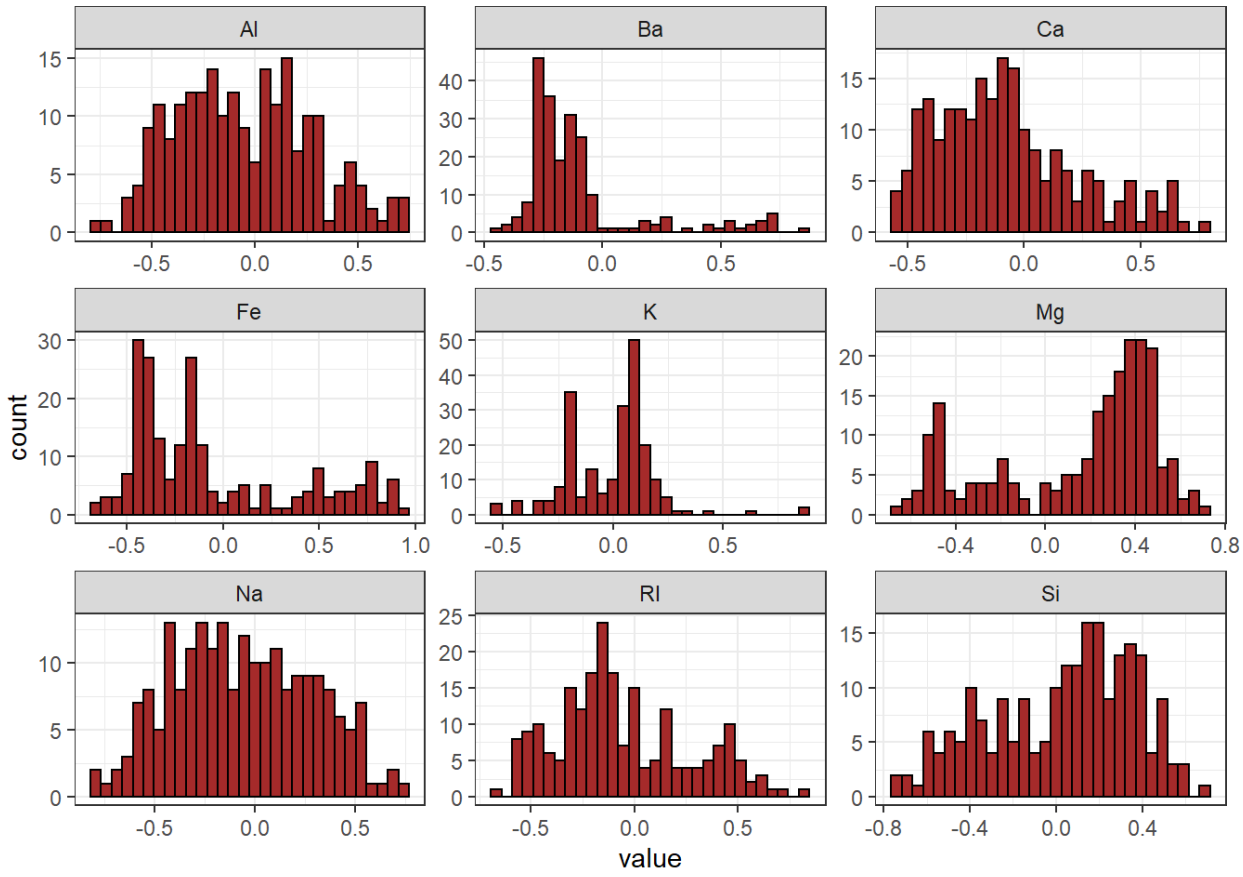

Figure 6. Results for symmetry after Spatial Sign transformation

After the spatial sign transformation, generally, there is an improvement in the distribution of each predictor comparing Figure 6 (after transformation) against Figure 1 (before transformation). Individual transformations could also be done for the predictors before the spatial sign transformation.

All code for question 3.1 can be found in Appendix A.

Question 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 distributions degenerate in the ways discussed earlier in this chapter? (Please provide figures/tables as necessary to support your conclusions)

Predictors with a single value for most of the samples include 'sclerotia,' 'roots', 'fruiting. bodies', 'mycelium', and others. Distributions of these predictors are regarded as degenerate since they have unique values that account for most of the frequency of the samples. In addition, detecting near-zero variance predictors, two conditions can be used to select these predictors. First, the fraction of unique values over the sample size is low, generally about ten percent. Second, the frequency of the most prevalent value to the frequency of the second most prevalent value is large.

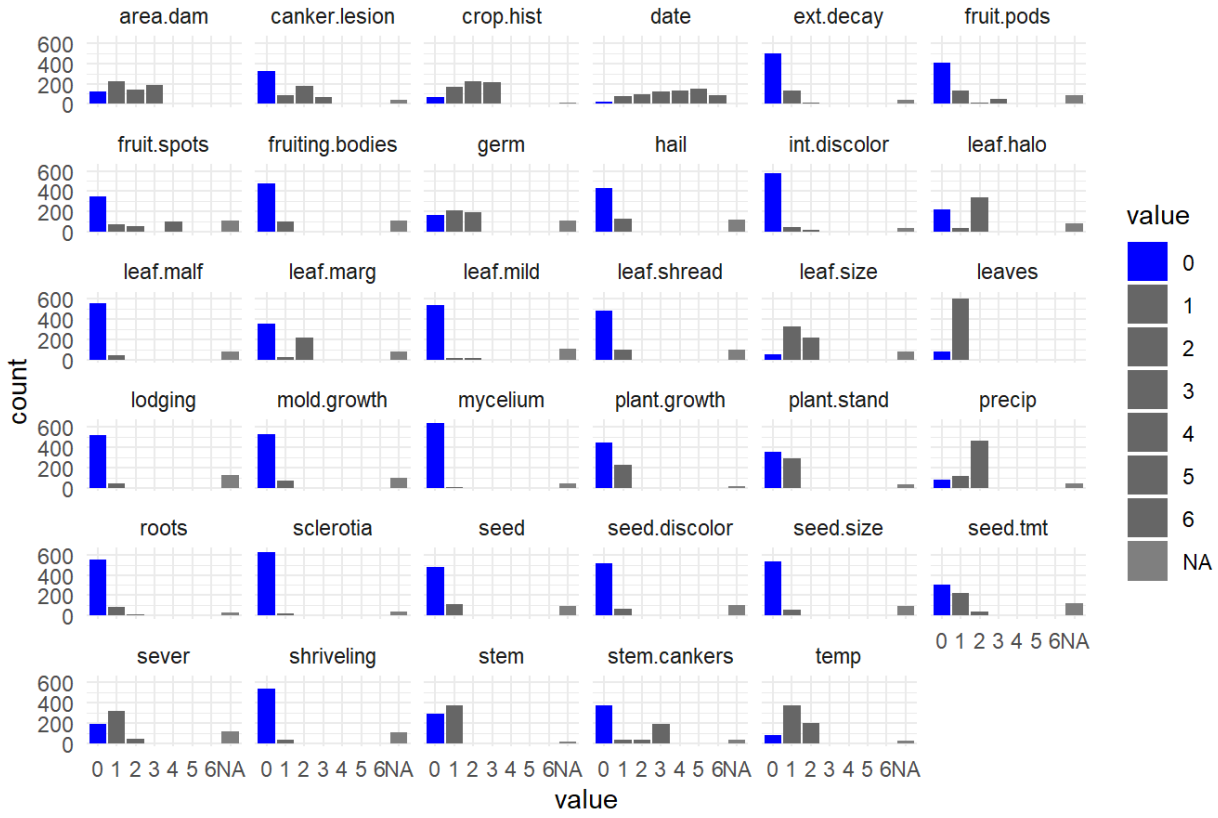

Figure 7. Distributions by predictor showing the degenerate feature in some distributions

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?

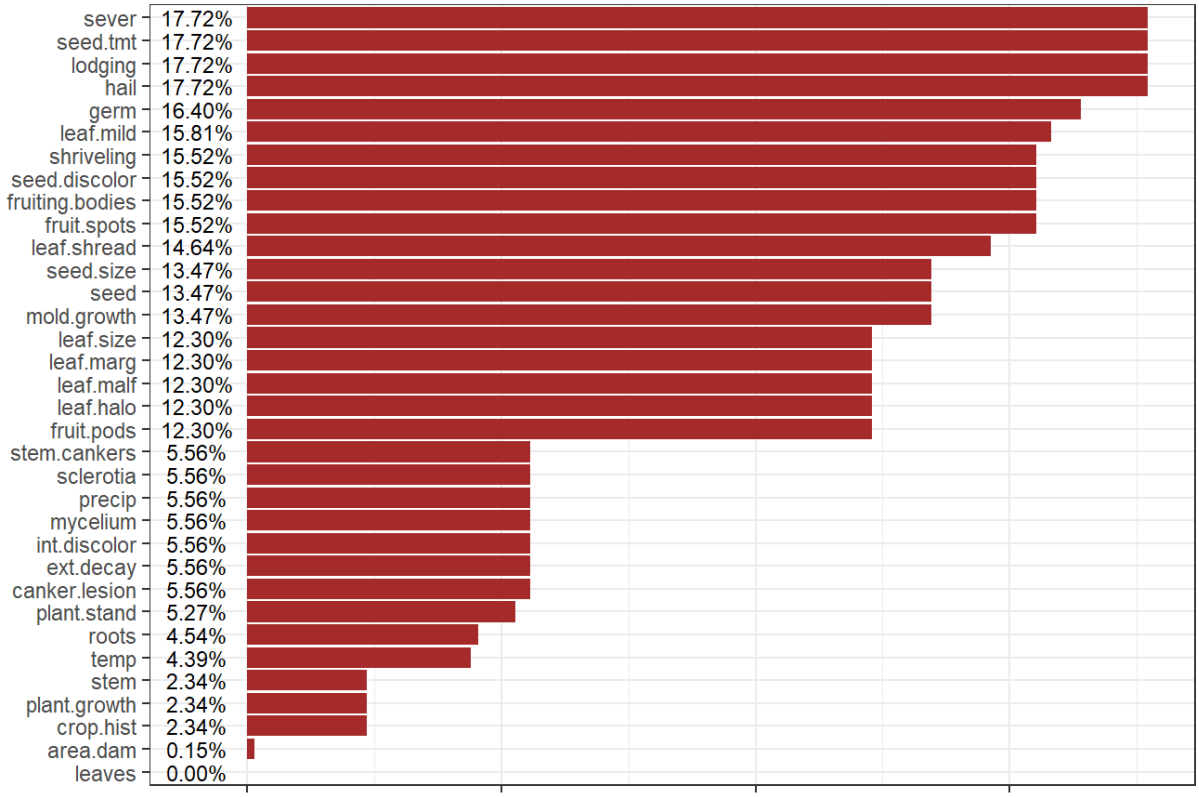

Figure 8. Percentage of missing data by the predictor

From Figure 8, 'sever,' 'seed.tmt', 'lodging', and 'hail' variables are missing in most of the classes. 'leaves' is the only other predictor with entries for all cases.

Table 2. Classes with missing data in proportion to all other classes

Class	Count	Percentage Proportion
phytophthora-rot	88	12.884
diaporthe-pod-&-stem-blight	15	2.196
cyst-nematode	14	2.05
2-4-d-injury	16	2.343
herbicide-injury	8	1.171

From Table 2, it is evident that these five classes are responsible for all the missing data: `phytophthora-rot`, `2-4-d-injury`, `diaporthe-pod-&-stem-blight`, `cyst-nematode`, and `herbicide-injury`.

c. Develop a strategy for handling missing data, either by eliminating predictors or imputation. (You only need to provide the strategy, you do not need to implement the strategy).

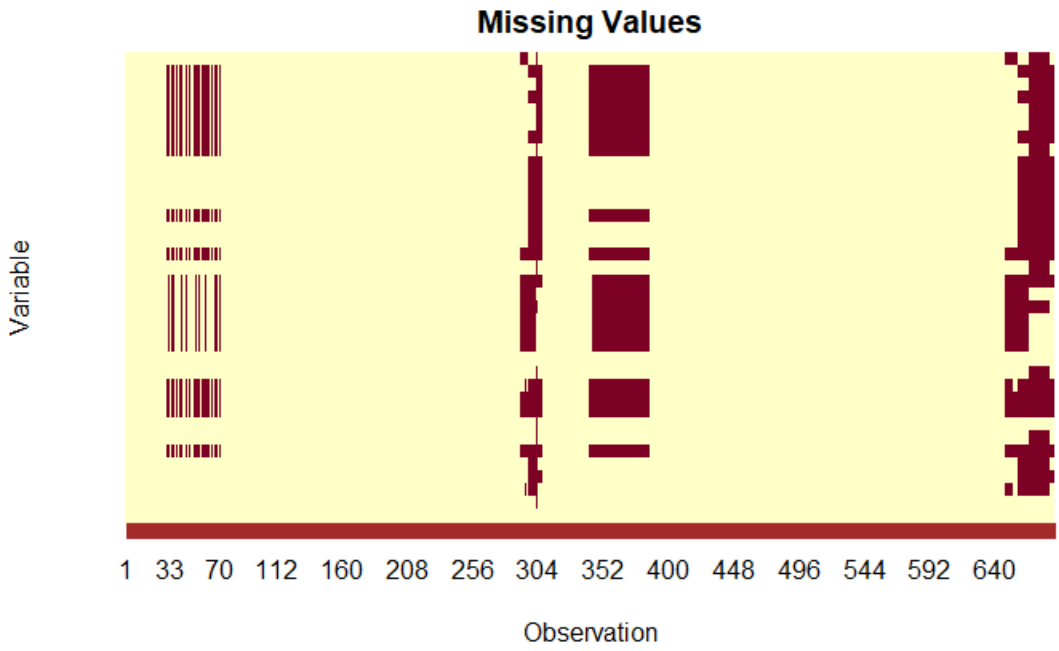

Figure 9. The plot of variable against Observation using missing data

Due to the size of our data, the elimination of predictors would not be ideal. Each variable has less than 18% of data missing, which could be handled by KNN or mode imputation approach. It may also be helpful to reduce the dimensions by extracting the most variance through PCA. However, imputation may have to be done for all the predictors in a few cases. Another strategy will be to eliminate the classes with missing data altogether. Models such as naive Bayes and tree-based that are less sensitive to missing data will be suitable.

All code for question 3.2 can be found in Appendix B.

Question 3.3

Chapter 5 introduces Quantitative Structure-Activity Relationship (QSAR) modeling where the characteristics of a chemical compound are used to predict other chemical properties. The caret package contains a QSAR data set from Mente and Lombardo (2005). Here, the ability of a chemical to permeate the blood-brain barrier was experimentally determined for 208 compounds. One hundred thirty-four descriptors were measured for each compound.

a. Start R and use these commands to load the data: Check Appendix C for code.
b. Generally speaking, are there strong relationships between the predictor data? If so, how could correlations in the predictor set be reduced? Does this have a dramatic effect on the number of predictors available for modeling? (Please provide figures/tables as necessary to support your conclusions)

First, we can find the correlation of the raw data without any transformations.

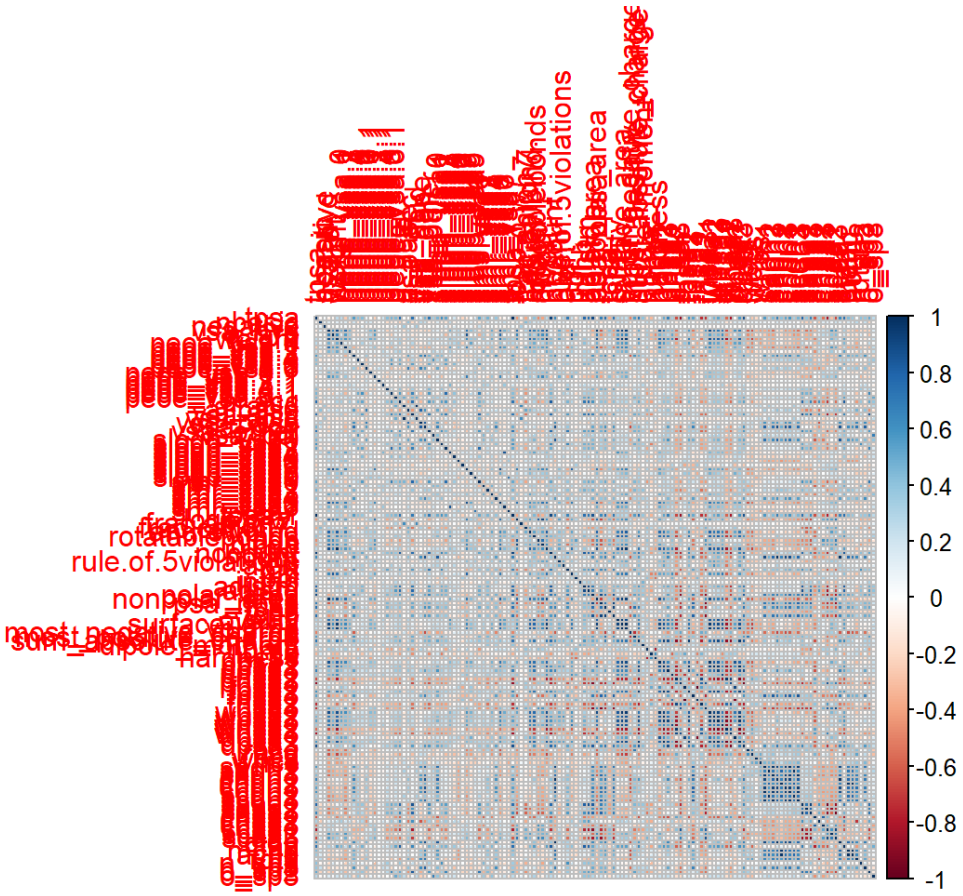

Figure 10. Correlations for all the predictors before any preprocessing

There are high correlations between certain predictors. A stepwise procedure would be to use the nearzerovar function to diagnose predictors with near-zero variance and filter those predictors to reduce correlation among the predictors. We can also check for skewness for the filtered predictors. Once this preprocessing is carried out, we can center and scale the

data to improve the symmetry of each predictor, then further transform our filtered data using spatial sign to make our filtered predictors uncorrelated. Using the nearzerovar function reduces the number of predictors from 134 to 127.

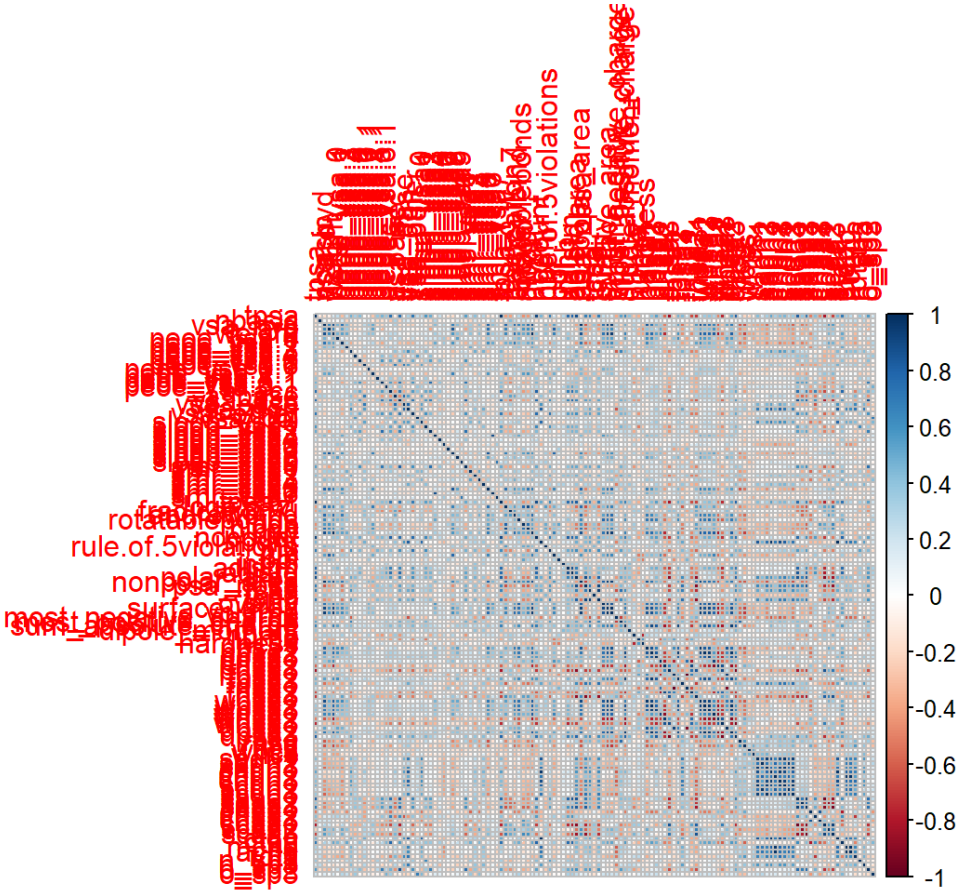

Figure 11. New correlation plot after transformations

Another method will be to use the 'findCorrelation' function on the raw predictors to reduce the number of predictors. Depending on the cutoff (threshold) specified, the number of predictors can be dramatically reduced. For example, using a cutoff of 0.50 the number of predictors is reduced to thirty-four as given in the table below.

Cutoff used in findCorrelation function	Number of Predictors removed	Number of predictors retained
.50	100	34
.65	80	54
.75	66	68
.85	49	85

In addition, feature extraction methods such as principal component analysis can resolve the high correlation between predictors. However, this may result in a complex relationship between the predictors and the response.

All code for question 3.3 can be found in Appendix C.

Appendix A

#Setting up working directory

```
setwd('C:/Users/John/Desktop/MA4790_MA5790')


# Load libraries and data
library(ggplot2)
library(dplyr)
library(moments)
library(e1071)
library(caret)
library(mlbench)
library(psych)
library(tidyverse)
library(kableExtra)
library(corrplot)
library(knitr)


# Compute and print out the summary statistics for the predictors
# Question 3.1
#a.


data(Glass)
glass_summary <- Glass %>% discard(is.factor) %>% describe()


glass_summary %>%  kable(caption="Summary Statistics of Glass dataset",digits
= 3) %>%
  kable_styling(c("hold_position", "striped"))


print(glass_summary)


# Create a histogram plot for all the predictors


Glass %>%
```

```
  discard(is.factor)%>%
  gather() %>%
  ggplot(aes(value)) +
  geom_histogram(fill="brown", color = "black") +
  facet_wrap(~ key, scales = "free") +
  theme_bw()

#Create correlation plot
glass_pred_Cor <- cor(Glass[-10])
corrplot(glass_pred_Cor,method = "number", type = "lower")

#b.

# Create histogram plots
Glass %>%
  discard(is.factor)%>%
  gather() %>%
  ggplot(aes(x ="", y = value))+
  stat_boxplot(geom ="errorbar") +
  geom_boxplot(outlier.colour = "lightblue", fill="green") +
  facet_wrap(~ key, scales = "free") +
  labs(x = NULL, y = NULL) +
  theme_bw() +
  theme(axis.ticks.y=element_blank())+
  coord_flip()

# Print variable and corresponding skewness
glass_summary$var <- rownames(glass_summary)
glass_skew <-glass_summary %>% select(var,skew)
glass_skew %>%
  arrange(skew) %>%
  kable(digits = 3) %>%
  kable_styling(c("striped", "hover"), full_width = FALSE)
```

```
#c
# Transformation 1
#center and scale predictors
glass2 <- Glass %>%
  discard(is.factor)
glass2 <- as.data.frame(sapply(glass2, scale))

# Visualize transformation
glass2 %>%
  gather()%>%
  ggplot(aes(value)) +
  geom_histogram(fill="brown", color = "black") +
  facet_wrap(~ key, scales = "free") +
  theme_bw()

# Transformation 2
#BoxCox transformation
predictors <- as.vector(glass_summary$var)

for (predictor in predictors){
  print(predictor)
  print(BoxCoxTrans(Glass[,predictor]))
}
head(Glass[-10]) %>% kable(caption="Glasswithout transformations", digits = 3
) %>%
  kable_styling(c("hold_position", "striped"))

# Print first six rows of the transformed predictors
trans <- preProcess(Glass[-10], method = c('center', 'scale', 'BoxCox'))
trans_glass <- predict(trans, Glass[-10])

head(trans_glass) %>% kable(caption="Glass with transformations", digits = 3)
%>%
  kable_styling(c("hold_position", "striped"))
```

```
# Plot the transformed predictors
trans_glass%>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free") +
    geom_histogram(fill ='brown', color ='black')+
    theme_bw()

# Transformation 3
# Spatial sign transformation
trans <- preProcess(Glass, method = c('center', 'scale', 'spatialSign'))
trans_glass <- predict(trans, Glass)
trans_glass%>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free") +
    geom_histogram(fill ='brown', color ='black')+
    theme_bw()
```

Appendix B

```
# Question 3.2
#a.

#Load the data
data(Soybean)
str(Soybean)

# Clean dataset, removing non-numeric variables
subset(Soybean, select= c(-Class)) %>%
  gather() %>%
  ggplot(aes(value, fill = value)) +
  geom_bar() +
  scale_fill_manual(values = c('blue', rep('grey40', 7))) +
  facet_wrap(~ key) +
  theme_minimal()+
  labs(title = 'Soybean: Distributions by Predictor')

b.

Soybean %>%
   select(-Class, -date) %>%
  summarise_all(funs(perc_missing = sum(is.na((.)) / nrow(Soybean)))) %>%
  rename_all(funs(str_replace(., '_perc_missing', ''))) %>%
  gather() %>%
  ggplot(aes(x = reorder(key, value), y = value)) +
  geom_bar(stat = 'identity', fill = 'brown') +
  geom_text(aes(label = scales::percent(value), y = -.01), size = 3,position
= position_dodge(width = 0.9)) +
  coord_flip() +
  labs(title = 'Soybean: Missing Data by Predictor',
         x = '',
         y = '') +
```

```
  theme_bw() +
  theme(axis.text.x = element_blank())


Soybean %>%
  group_by(Class) %>%
  mutate(Count = n(), Proportion=round(Count/nrow(Soybean)*100,3)) %>%
  ungroup() %>%
  filter(!complete.cases(.)) %>%
  select(Class, Count, Proportion) %>% unique() %>%
  kable(caption="Classes with Missing Data in Proportion to All Classes") %>%
  kable_styling(full_width = FALSE)
#c.
#Create a graph of missing values
image(is.na(Soybean), main = "Missing Values", xlab = "Observation", ylab = "
Variable", xaxt = "n", yaxt = "n", bty = "n")
axis(1, seq(0, 1, length.out = nrow(Soybean)), 1:nrow(Soybean), col = "brown"
)
```

Appendix C

```
# Question 3.3
#a. Start R and use these commands to load the data:

data(BloodBrain)
str(logBBB)
str(bbbDescr)

# Print names of the datasets
names(bbbDescr)
names(logBBB)

#b.

raw_Corr<-cor(bbbDescr)
corrplot(raw_Corr,method = "square")

#diagnose for near zero variance and store each predictor's results
predictor_Info <- nearZeroVar(bbbDescr, saveMetrics = TRUE)

#discard predictors with near zero variance metric == True
predictor_filtered <- bbbDescr[,!predictor_Info$nzv]

# Compute and print out the summary statistics for the predictors

filter1_summary <- predictor_filtered %>%  describe()

filter1_summary %>%  kable(caption="Summary Statistics of Filtered dataset",d
igits = 3) %>%
  kable_styling(c("hold_position", "striped"))

print(head(filter1_summary["skew"]))
```

```
trans1 <- preProcess(predictor_filtered, method = c('center', 'scale', 'spati
alSign'))

trans_filter1 <- predict(trans1, predictor_filtered)

trans_filter1_summary <- trans_filter1 %>%  describe()


trans_filter1_summary %>%  kable(caption="Summary Statistics of Filtered data
set",digits = 3) %>%

  kable_styling(c("hold_position", "striped"))


# Number of columns in after transformations

ncol(trans_filter1)


print(head(trans_filter1_summary["skew"]))


# Recalculating the correlations between the transformed predictors

trans_Corr<-cor(trans_filter1)

corrplot(trans_Corr,method = "square")


#Reduce the number of predictors with findCorrelation


highCorr <- findCorrelation(raw_Corr, cutoff = .85)

length(highCorr)

highCorr

filteredPredictors <- bbbDescr[, -highCorr]

length(filteredPredictors)


highCorr <- findCorrelation(raw_Corr, cutoff = .75)

length(highCorr)

highCorr

filteredPredictors <- bbbDescr[, -highCorr]

length(filteredPredictors)


highCorr <- findCorrelation(raw_Corr, cutoff = .65)

length(highCorr)

highCorr
```

```
filteredPredictors <- bbbDescr[, -highCorr]
length(filteredPredictors)

highCorr <- findCorrelation(raw_Corr, cutoff = .5)
length(highCorr)
highCorr
filteredPredictors <- bbbDescr[, -highCorr]
length(filteredPredictors)
```