Homework 2 Solution

Brian Lois

1/27/2017

```
library(ggplot2)
library(caret)
library(mlbench)
library(corrplot)
```

1.

```
RMSE <- function(y, y_hat){
  sq_er <- (y - y_hat)^2
  mean_sq_er <- mean(sq_er)
  rmse <- sqrt(mean_sq_er)
  return(rmse)
}
```

2.

```
data(Glass)
```

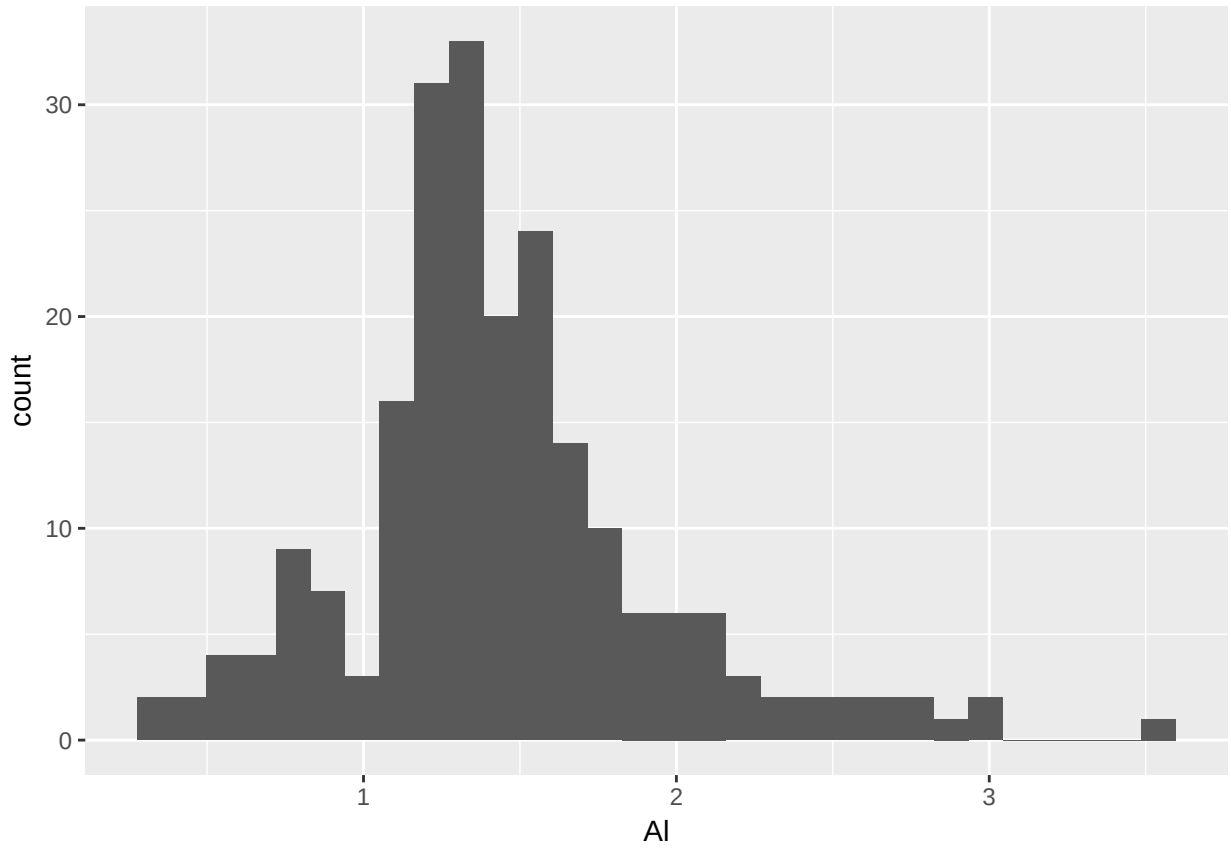
a)

Look at all histograms

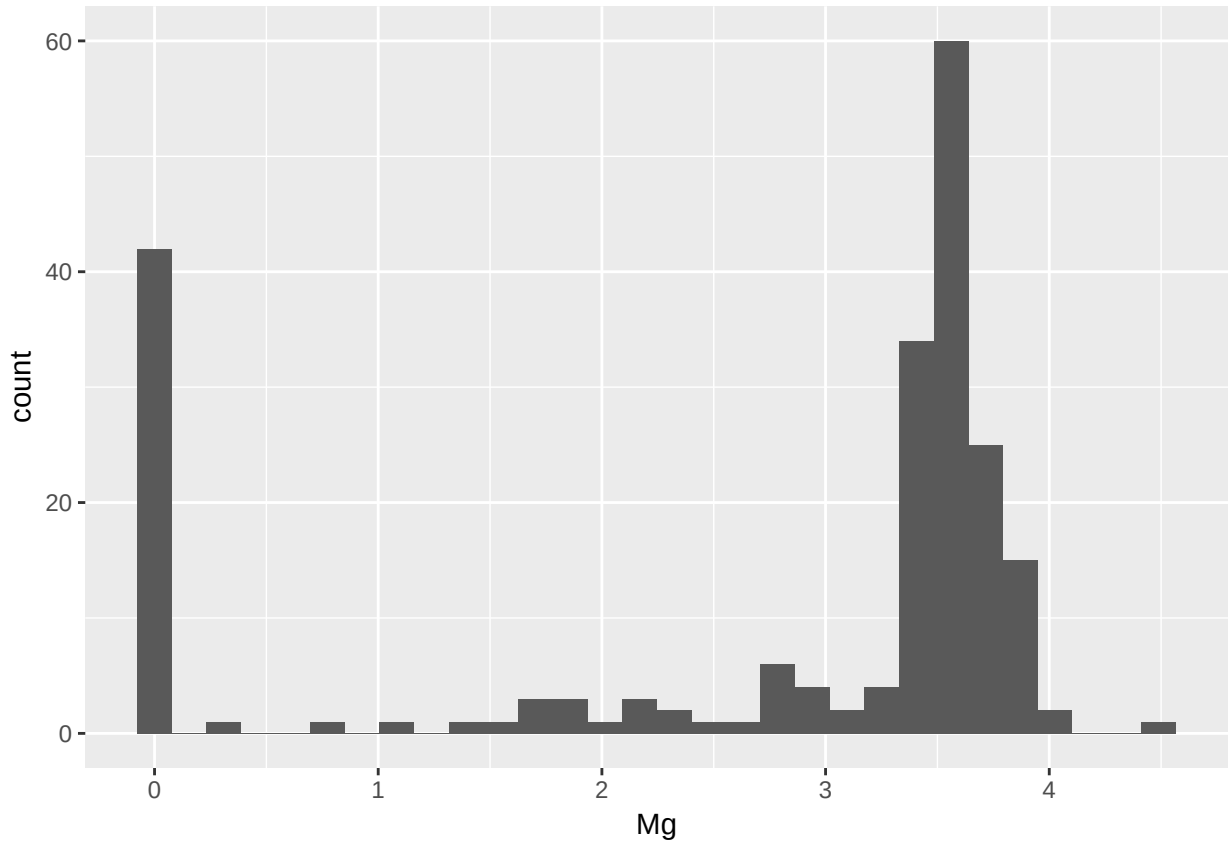
```
for (col in colnames(Glass)){
  print(ggplot(data = Glass) + geom_histogram(mapping = aes_string(x = col)))
}
```

Only plot 2 (any two interesting ones are correct).

```
ggplot(data = Glass) + geom_histogram(mapping = aes(x = Al))
```

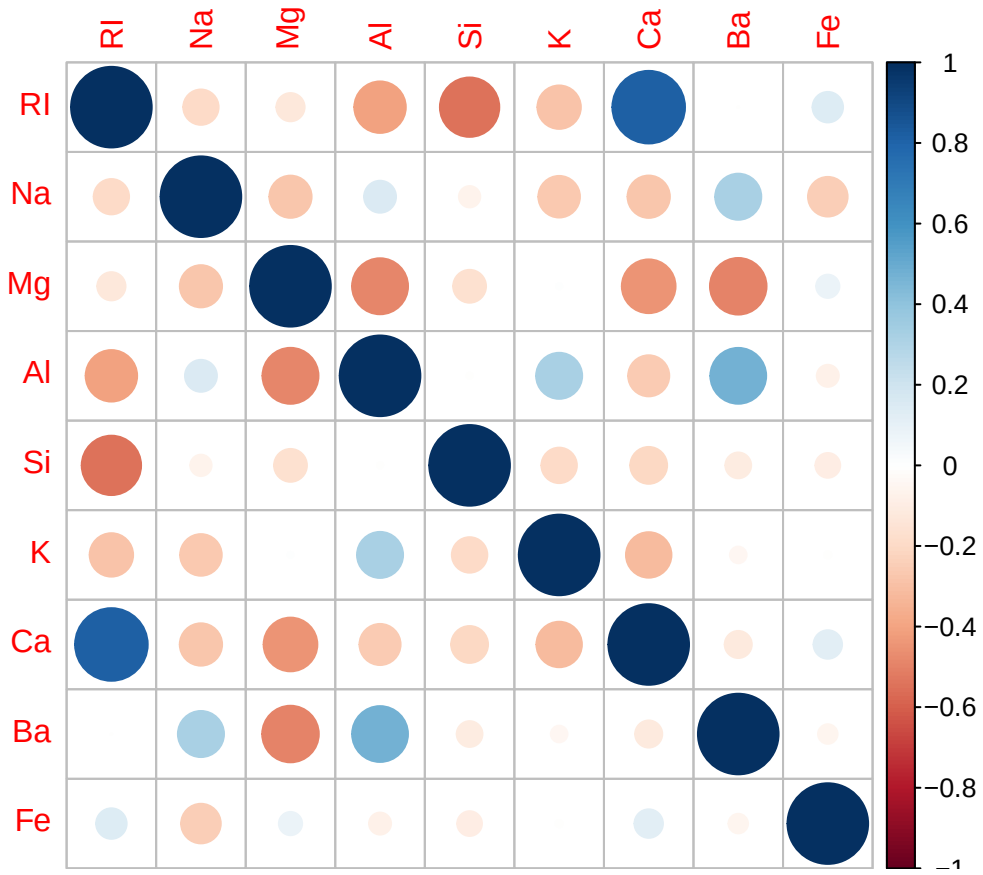



```
ggplot(data = Glass) + geom_histogram(mapping = aes(x = Mg))
```



Correlation matrix:

```
correlation_matrix <- cor(subset(Glass, select = -c(Type)))
corrplot(correlation_matrix)
```



b)

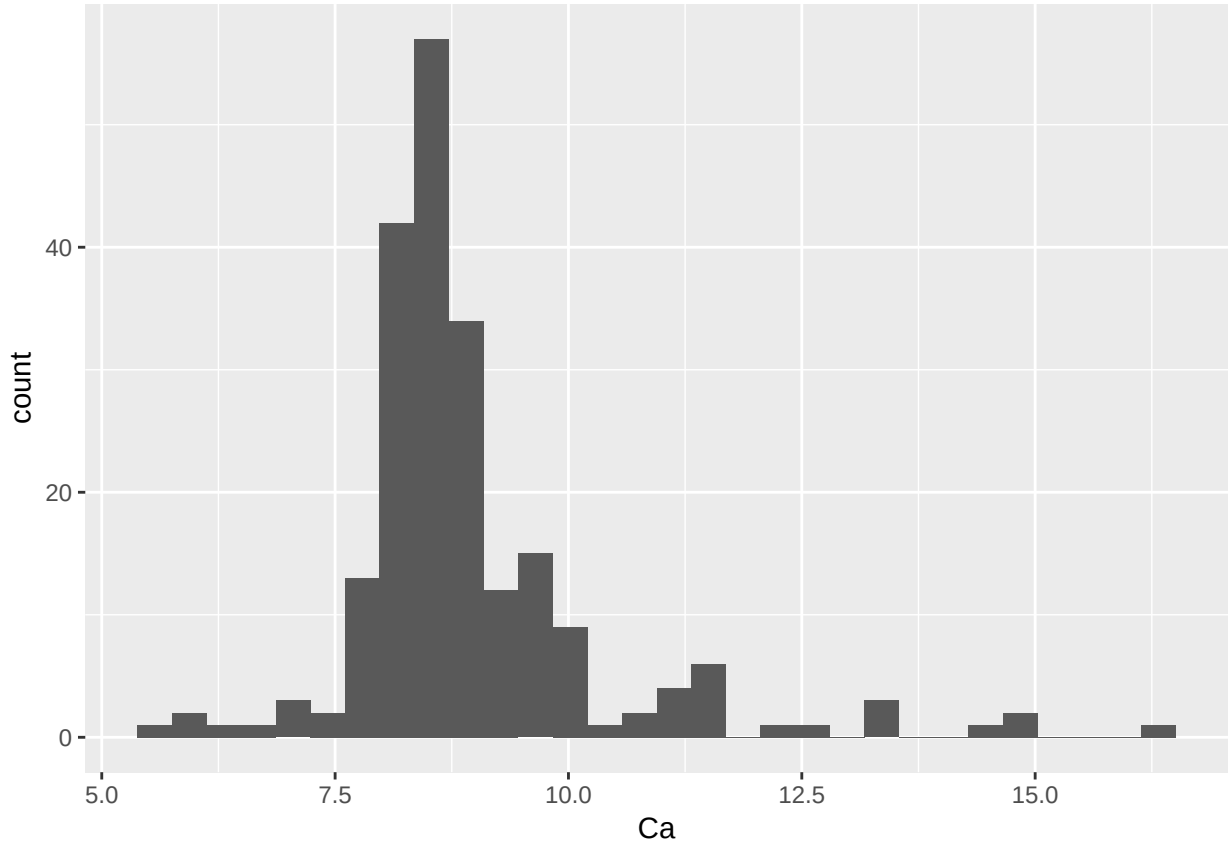
```
skewness_list <- apply(Glass[,1:9], 2, BoxCoxTrans)
for (col in colnames(Glass)[1:9]){
  print(paste(col, skewness_list[[col]]$skewness))
}
```

```
## [1] "RI 1.60271508274373"
## [1] "Na 0.447834258917133"
## [1] "Mg -1.13645227846653"
## [1] "Al 0.89461041611312"
## [1] "Si -0.720239210805621"
## [1] "K 6.46008889572281"
## [1] "Ca 2.01844629445302"
## [1] "Ba 3.36867996880571"
## [1] "Fe 1.7298107095598"
```

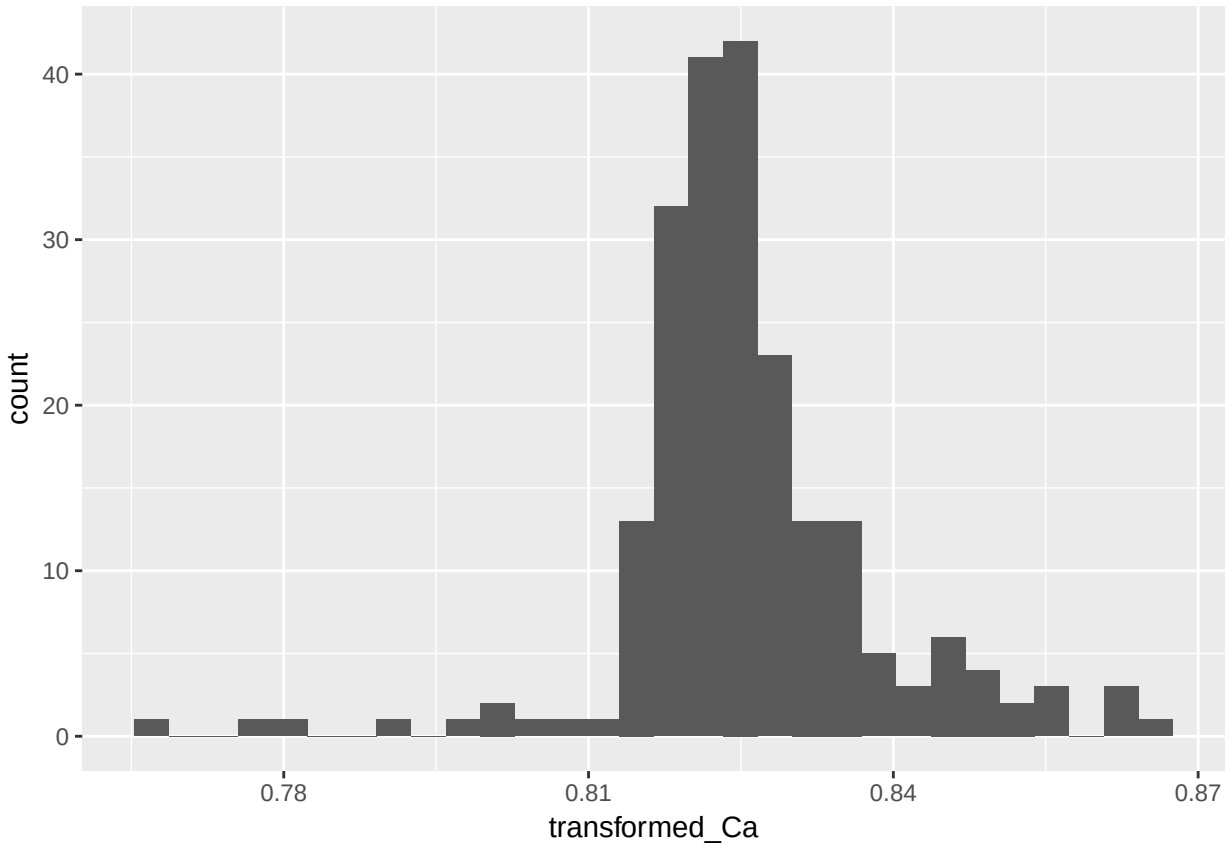
c)

We will do the Box-Cox transformation on `Ca`.

```
ggplot(data = Glass) + geom_histogram(mapping = aes(x = Ca))
```




```
Glass['transformed_Ca'] <- predict(skewness_list$Ca, Glass$Ca)
ggplot(data = Glass) + geom_histogram(mapping = aes(x = transformed_Ca))
```



3.

a)

```
data("Soybean")
P <- ncol(Soybean)
n <- nrow(Soybean)
degenerate <- function(dist){
  frac_unique <- length(dist) / sum(dist)
  ratio <- sort(dist, decreasing = TRUE)[1] / sort(dist, decreasing = TRUE)[2]
  if ((frac_unique < .1) & (ratio > 20)){
    return(TRUE)
  } else {
    return(FALSE)
  }
}

var_tables <- apply(Soybean, 2, table)
degenerate_features <- sapply(var_tables, degenerate)

Soybean <- Soybean[ ,!degenerate_features]
P <- ncol(Soybean)
```

b)

```
percent_missing <- function(x){
  p_m <- sum(is.na(x)) / length(x)
  return(p_m)
}
```

Percent missing by feature:

```
apply(Soybean, 2, percent_missing)
```

```
##           Class            date     plant.stand          precip
##     0.000000000     0.001464129     0.052708638     0.055636896
##            temp            hail       crop.hist        area.dam
##     0.043923865     0.177159590     0.023426061     0.001464129
##           sever        seed.tmt            germ    plant.growth
##     0.177159590     0.177159590     0.163982430     0.023426061
##          leaves       leaf.halo       leaf.marg       leaf.size
##     0.000000000     0.122986823     0.122986823     0.122986823
##     leaf.shread       leaf.malf            stem         lodging
##     0.146412884     0.122986823     0.023426061     0.177159590
##    stem.cankers   canker.lesion fruiting.bodies       ext.decay
##     0.055636896     0.055636896     0.155197657     0.055636896
##    int.discolor      fruit.pods     fruit.spots            seed
##     0.055636896     0.122986823     0.155197657     0.134699854
##     mold.growth   seed.discolor       seed.size      shriveling
##     0.134699854     0.155197657     0.134699854     0.155197657
##           roots
##     0.045387994
```

Yes, some features are much more likely to be missing.

Percent missing by class and feature: (Output is suppressed to save space)

```
missing_mat <- list()
for (level in levels(Soybean$Class)){
  for (col in colnames(Soybean)[2:P]){
    vec <- Soybean[[col]][Soybean$Class == level]
    missing_mat[[level]][[col]] <- percent_missing(vec)
  }
}
print(missing_mat)
```

Many classes have no missing values, and some classes have a lot of missing values. Some classes may even have all of the features missing.

c)

Possible strategies could include any combination of:

- Removing whole classes such as `herbicide-injury` and `cyst-nematode`.

- Mean, median, or knn imputation
- I don't see any columns that I would entirely remove based on missingness, but I won't take off points for removing features with more than 10% missing.
- Adding missing indicator columns.

The best approach here is unclear without knowing the ultimate goal of the project.

4.

a)

```
data(BloodBrain)
```

b)

Reuse our function from before

```
bbb_var_tables <- apply(bbbDescr, 2, table)
degenerate_vec <- sapply(bbb_var_tables, degenerate)
sum(degenerate_vec)
```

```
## [1] 7
```

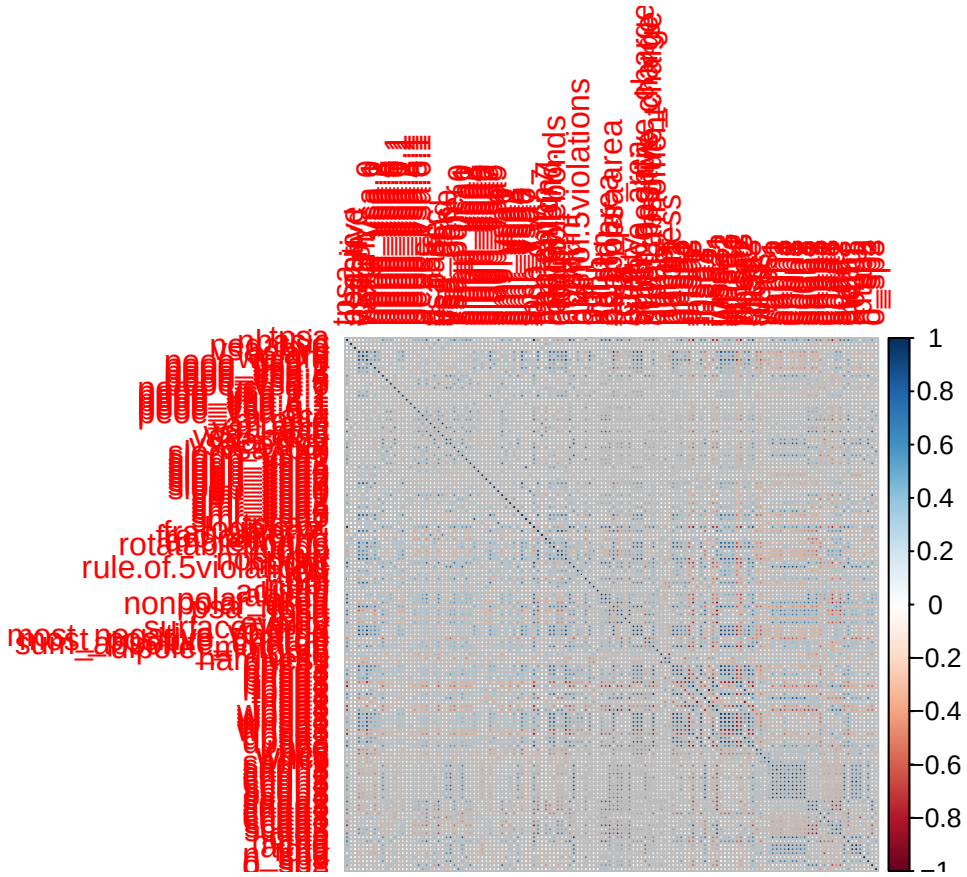
```
var_vec <- apply(bbbDescr, 2, var)
head(sort(var_vec), 10)
```

```
##        chaa3        chdh3         tcsa         tcnp        fpsa3
## 7.512226e-07 8.830284e-07 9.064777e-06 4.618282e-05 3.793002e-04
##        scdh3        scaa3        fnsa3        sadh3         rpcg
## 4.298434e-04 8.640507e-04 1.341075e-03 2.342703e-03 2.958051e-03
```

None are degenerate as defined in the chapter, but some have **very** near-zero variance.

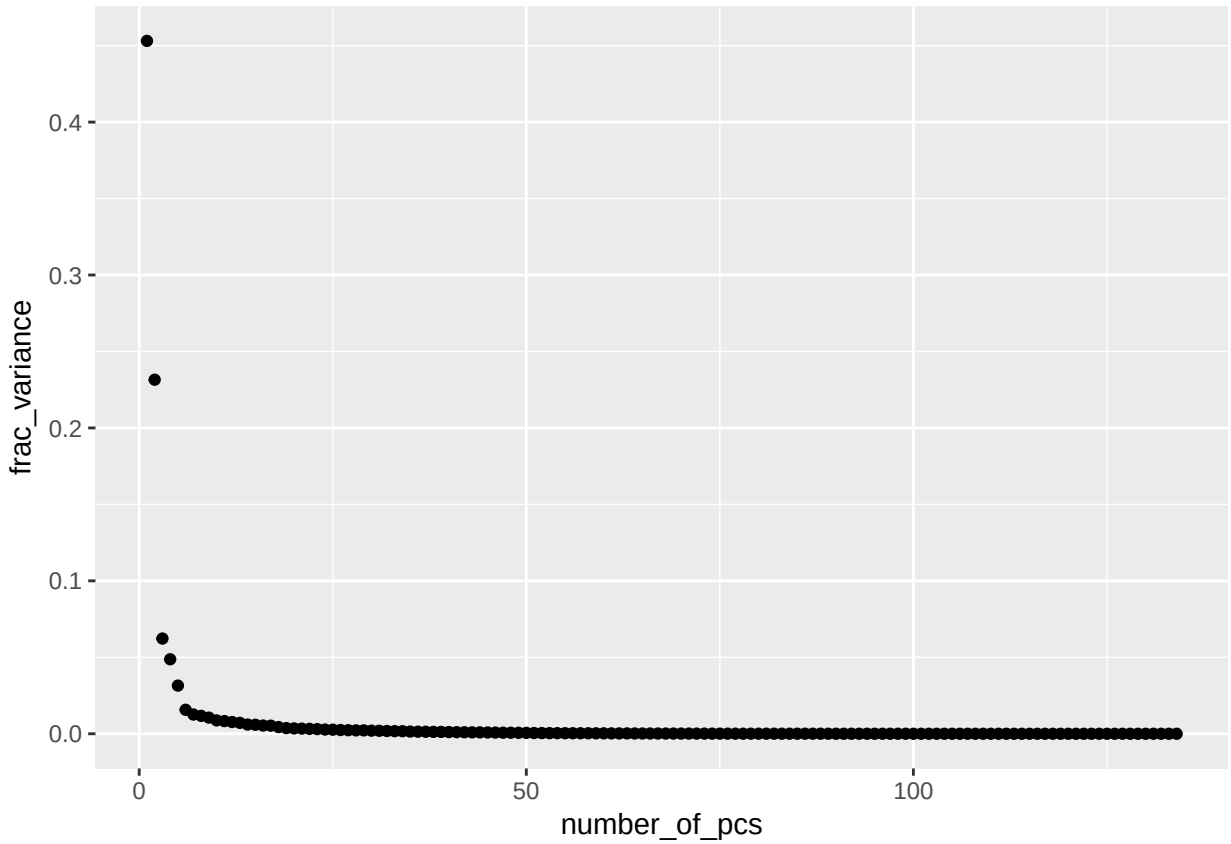
c)

```
cor_mat <- cor(bbbDescr)
corrplot(cor_mat)
```



Yes, there do appear to be strong relationships (dark colors in the corrplot).

```
pc <- prcomp(bbbDescr)
scaled_stdev <- as.data.frame(cbind(pc$sdev / sum(pc$sdev), 1:length(pc$sdev)))
colnames(scaled_stdev) <- c('frac_variance', 'number_of_pcs')
ggplot(data=scaled_stdev) + geom_point(mapping=aes(x=number_of_pcs, y=frac_variance))
```



Based on this, I would keep 5 or 6 principal components.