BACS - HW 13 106073401

Question 1)

a. Create a *new data.frame* of the four log-transformed variables with high multicollinearity

```
cars <- read.table("auto-data.txt", header = FALSE, na.strings = "?")
names(cars) <- c("mpg", "cylinders", "displacement", "horsepower", "weight", "accele
ration","model_year", "origin", "car_name")
cars<-na.omit(cars)
cars_log <- with(cars, data.frame(log(mpg), log(cylinders),log(displacement), log(ho
rsepower), log(weight), log(acceleration),model_year, origin))
```

i. Give this smaller data frame an appropriate name (think what they might jointly mean)

ii. Check the correlation table of these four variables to confirm they are indeed collinear

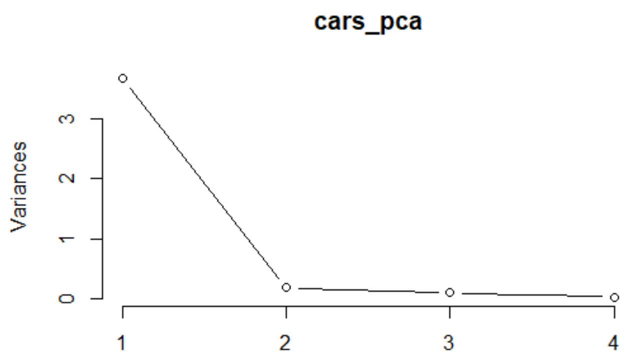
```
new_cars_log <- cars_log[2:5]
cor(new_cars_log)
                  log.cylinders. log.displacement. log.horsepower. log.weight.
log.cylinders.         1.0000000         0.9469109       0.8265831   0.8833950
log.displacement.      0.9469109         1.0000000       0.8721494   0.9428497
log.horsepower.        0.8265831         0.8721494       1.0000000   0.8739558
log.weight.            0.8833950         0.9428497       0.8739558   1.0000000
```

b. Let's analyze the principal components of the four collinear variables

i. How many principal components are needed to summarize these four variables?

```
eigen(cor(new_cars_log))$values
[1] 3.67425879 0.18762771 0.10392787 0.03418563

cars_pca <- prcomp(new_cars_log, scale. = TRUE)
screeplot(cars_pca, type = "lines")
```



ii. How much variance of the four variables is explained by their first principal component?

```
eigen(cor(new_cars_log))$values[1]/sum(eigen(cor(new_cars_log))$values)
[1] 0.9185647
```

iii. Looking at the values and valence (positive/negative) of the first principal component's eigenvector, what would you call the information captured by this component?

```
eigen(cor(new_cars_log))$vectors
           [,1]        [,2]        [,3]       [,4]
[1,] -0.4979145  0.53580374 -0.52633608 -0.4335503
[2,] -0.5122968  0.25665246  0.07354139  0.8162556
[3,] -0.4856159 -0.80424467 -0.34193949 -0.0210980
[4,] -0.5037960 -0.01530917  0.77500928 -0.3812031
```

c. Let's reduce the four collinear variables into one new variable!

i. Store the scores of the first principal component as a new column of cars_log

`cars_log$new_column_name <- ...scores of PC1...`

ii. Name this column appropriately based on the meaning of this first principal component

```
cars_log$pca <- cars_pca$x[,1]
```

d. Let's revisit our regression analysis on cars_log:

(HINT: to compare variables across models, it helps to standardize many of the variables)

i. Regress mpg over the four correlates (*cylinders*, *displacement*, *horsepower*, and *weight*), as well as acceleration, model_year and origin

```
regr <- lm(log.mpg. ~ log.cylinders.+log.displacement.+log.horsepower.+log.weight.
+log.acceleration.+model_year+factor(origin), data=cars_log)
```

ii. Repeat the regression, but replace the four highly collinear variables with a single variable: the scores of their 1st principal component stored in the new column

```
regr_pca <- lm(log.mpg. ~ pca+log.acceleration.+model_year+factor(origin), data=ca
rs_log)
```

iii. Check the VIF values (use vif function of car package) of both models to compare their multicollinearity characteristics

```
library('car')
vif(regr)
                       GVIF Df GVIF^(1/(2*Df))
log.cylinders.    10.456738  1        3.233688
log.displacement. 29.625732  1        5.442952
log.horsepower.   12.132057  1        3.483110
log.weight.       17.575117  1        4.192269
log.acceleration.  3.570357  1        1.889539
model_year         1.303738  1        1.141814
factor(origin)     2.656795  2        1.276702

vif(regr_pca)
                      GVIF Df GVIF^(1/(2*Df))
pca               2.555002  1        1.598437
log.acceleration. 1.549953  1        1.244971
model_year        1.208800  1        1.099454
factor(origin)    1.845979  2        1.165619
```

iv. Comparing the two regressions, how have significances, fit or other characteristics changed?

From the VIF, we can see that principal component can well-substituted the four highly collinear predictors.

(see Question 2 on next page)

Question 2).

a. How much variance did each extracted factor explain?

```
security <- read.csv("security_questions.csv", header = TRUE, na.strings = "?")
security_pca <- prcomp(security, scale. = TRUE)
summary(security_pca)
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
Standard deviation     3.0514 1.26346 1.07217 0.87291 0.82167 0.78209 0.70921 0.68431 0.67229
Proportion of Variance 0.5173 0.08869 0.06386 0.04233 0.03751 0.03398 0.02794 0.02602 0.02511
Cumulative Proportion  0.5173 0.60596 0.66982 0.71216 0.74966 0.78365 0.81159 0.83760 0.86271

                         PC10    PC11    PC12    PC13    PC14    PC15   PC16   PC17   PC18
Standard deviation     0.6206 0.59572 0.54891 0.54063 0.51200 0.48433 0.4801 0.4569 0.4489
Proportion of Variance 0.0214 0.01972 0.01674 0.01624 0.01456 0.01303 0.0128 0.0116 0.0112
Cumulative Proportion  0.8841 0.90383 0.92057 0.93681 0.95137 0.96440 0.9772 0.9888 1.0000
```

b. How many dimensions would you retain, according to the criteria we discussed?

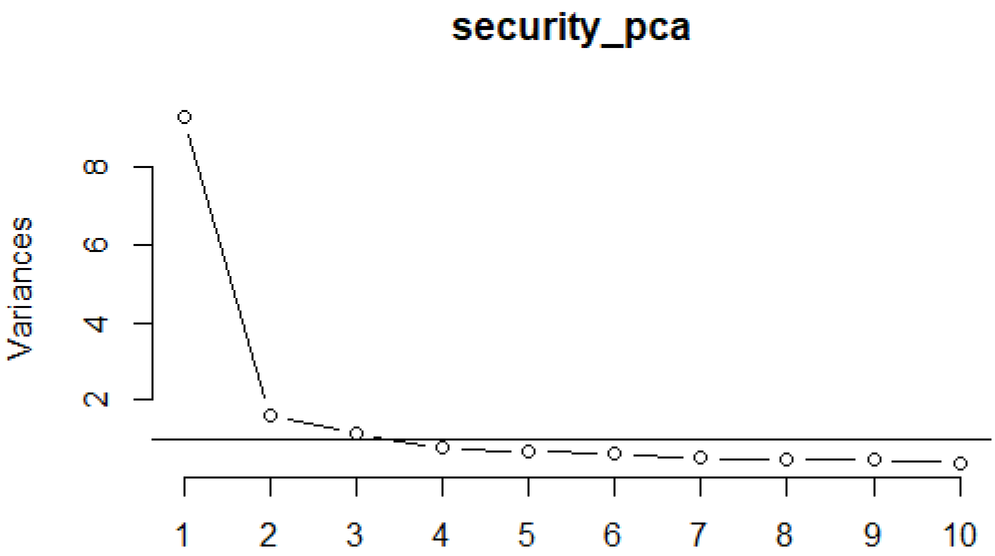
(show a single visualization with scree plot of data, scree plot of noise, eigenvalue = 1 cutoff)

i. Eigenvalues ≥ 1

```
eigen(cor(security))$values
[1] 9.3109533 1.5963320 1.1495582 0.7619759 0.6751412 0.6116636 0.5029855 0.4682788 0.4519711
[10] 0.3851964 0.3548816 0.3013071 0.2922773 0.2621437 0.2345788 0.2304642 0.2087471 0.2015441
```

ii. Scree plot

```
screeplot(security_pca, type = "lines")
abline(h=1)
```

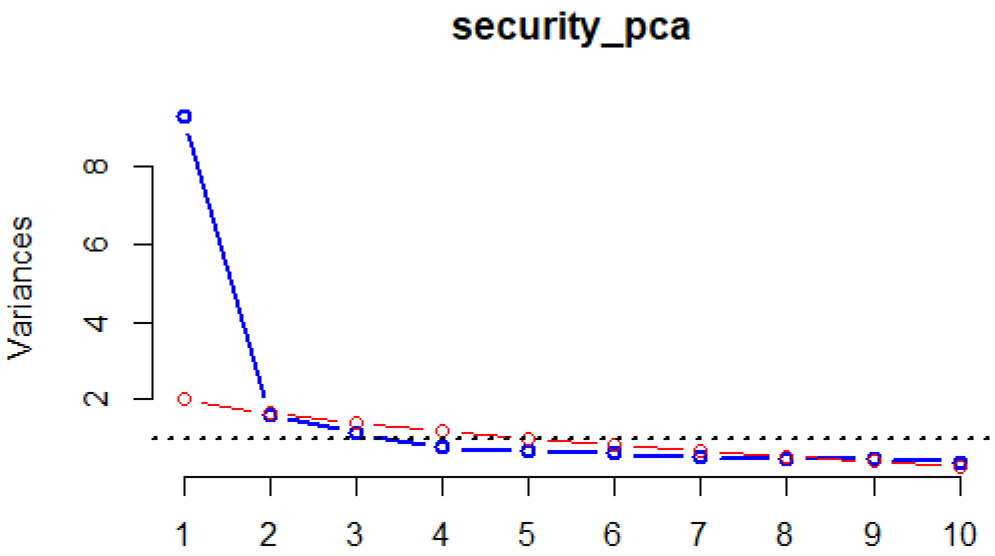


iii. Parallel Analysis

```
noise <- data.frame(replicate(10, rnorm(33)))
sim_noise <- function(n, p) {
    noise <- data.frame(replicate(p, runif(n)))
    return( eigen(cor(noise))$values )
}

set.seed(42)
evalues_noise <- replicate(100, sim_noise(33, 10))
evalues_mean <- apply(evalues_noise, 1, mean)

screeplot(security_pca, type = "lines", col = "blue", lwd = 2)
lines(evalues_mean, type="b", col = "red")
abline(h=1, lty="dotted", lwd = 2)
```



Overall, we should retain 3 dimensions.