BACS-hw13-107070004

Question 1) Let's revisit the issue of multicollinearity of main effects (between cylinders, displacement, horsepower, and weight) we saw in the cars dataset, and try to apply principal components to it. Start by recreating the cars_log dataset, which log-transforms all variables except model year and origin.

Important: remove any rows that have missing values.

```
cars <- read.table("auto-data.txt", header=FALSE, na.strings = "?")
names(cars) <- c("mpg", "cylinders", "displacement", "horsepower", "weight",
                 "acceleration", "model_year", "origin", "car_name")
cars <- na.omit(cars)
cars_log <- with(cars, data.frame(log(mpg), log(cylinders), log(displacement), log(acceleration),
                log(horsepower), log(weight), model_year, factor(origin)))
head(cars_log, 5)
```

```
##   log.mpg. log.cylinders. log.displacement. log.acceleration. log.horsepower.
## 1 2.890372       2.079442          5.726848          2.484907        4.867534
## 2 2.708050       2.079442          5.857933          2.442347        5.105945
## 3 2.890372       2.079442          5.762051          2.397895        5.010635
## 4 2.772589       2.079442          5.717028          2.484907        5.010635
## 5 2.833213       2.079442          5.710427          2.351375        4.941642
##   log.weight. model_year factor.origin.
## 1    8.161660         70              1
## 2    8.214194         70              1
## 3    8.142063         70              1
## 4    8.141190         70              1
## 5    8.145840         70              1
```

(a) Let's analyze the principal components of the four collinear variables

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

(Give this smaller data frame an appropriate name – what might they jointly mean?)

```
df <- data.frame(round(cor(cars_log[,c(2,3,5,6)]), 2))
df
```

```
##                    log.cylinders. log.displacement. log.horsepower. log.weight.
## log.cylinders.               1.00              0.95            0.83        0.88
## log.displacement.            0.95              1.00            0.87        0.94
## log.horsepower.              0.83              0.87            1.00        0.87
## log.weight.                  0.88              0.94            0.87        1.00
```

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

(a summary of the prcomp() shows it, but try computing this from the eigenvalues alone)

```
df_eigen <- eigen(cov(df))
df_eigen$vectors
```

```
##            [,1]       [,2]        [,3]       [,4]
## [1,] -0.6164180  0.4441313 -0.01965006  0.6499155
## [2,] -0.4217856 -0.1674419 -0.83511625 -0.3108714
## [3,]  0.6449840  0.0957147 -0.54221713  0.5299386
## [4,] -0.1616211 -0.8749568  0.09052798  0.4473633
```

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

(i.e., think what concept the first principal component captures or represents)

- PC1 doesn't capture much variance of the original data
- PC2 captures high log.horsepower.
- PC3 captures high log.weight.
- PC4 doesn't capture much variance of the original data

(b) Let's revisit our regression analysis on cars_log:

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

cars_log$new_column_name <- …scores of PC1… Give this new column a name suitable for what it captures (see 1.a.i.)

```
pca <- prcomp(cars_log[,c(2,3,5,6)], scale. = TRUE)
cars_log$PC1 <- pca$x[,1]
head(cars_log)
```

```
##   log.mpg. log.cylinders. log.displacement. log.acceleration. log.horsepower.
## 1 2.890372       2.079442          5.726848          2.484907        4.867534
## 2 2.708050       2.079442          5.857933          2.442347        5.105945
## 3 2.890372       2.079442          5.762051          2.397895        5.010635
## 4 2.772589       2.079442          5.717028          2.484907        5.010635
## 5 2.833213       2.079442          5.710427          2.351375        4.941642
## 6 2.708050       2.079442          6.061457          2.302585        5.288267
##   log.weight. model_year factor.origin.       PC1
## 1    8.161660         70              1 -2.036645
## 2    8.214194         70              1 -2.593998
## 3    8.142063         70              1 -2.237767
## 4    8.141190         70              1 -2.192902
## 5    8.145840         70              1 -2.097313
## 6    8.375860         70              1 -3.337215
```

(ii) Regress mpg over the column with PC1 scores (replacing cylinders, displacement, horsepower, and weight), as well as acceleration, model_year and origin.

```
summary(lm(log.mpg. ~ PC1 + log.acceleration. +
             model_year + factor.origin., data = cars_log))
```

```
##
## Call:
## lm(formula = log.mpg. ~ PC1 + log.acceleration. + model_year +
##     factor.origin., data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.51137 -0.06050 -0.00183  0.06322  0.46792
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.398114   0.166554   8.394 8.99e-16 ***
## PC1                0.145663   0.005057  28.804  < 2e-16 ***
## log.acceleration. -0.191482   0.041722  -4.589 6.02e-06 ***
## model_year         0.029180   0.001810  16.122  < 2e-16 ***
## factor.origin.2    0.008272   0.019636   0.421    0.674
## factor.origin.3    0.019687   0.019395   1.015    0.311
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1199 on 386 degrees of freedom
## Multiple R-squared:  0.8772, Adjusted R-squared:  0.8756
## F-statistic: 551.6 on 5 and 386 DF,  p-value: < 2.2e-16
```

(iii) Try running the regression again over the same independent variables, but this time with everything standardized. How important is this new column relative to other columns?

```
cars_log_std <- data.frame(scale(cars_log[,c(1:7,9)]))
summary(lm(log.mpg. ~ PC1 + log.acceleration. + model_year + cars_log$factor.origin.
        , data = cars_log_std))
```

```
##
## Call:
## lm(formula = log.mpg. ~ PC1 + log.acceleration. + model_year +
##     cars_log$factor.origin., data = cars_log_std)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.50385 -0.17791 -0.00538  0.18591  1.37608
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -0.01589    0.02563  -0.620    0.536
## PC1                       0.82112    0.02851  28.804  < 2e-16 ***
```

```
## log.acceleration.        -0.10190    0.02220  -4.589 6.02e-06 ***
## model_year                0.31611    0.01961  16.122  < 2e-16 ***
## cars_log$factor.origin.2  0.02433    0.05775   0.421    0.674
## cars_log$factor.origin.3  0.05790    0.05704   1.015    0.311
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3526 on 386 degrees of freedom
## Multiple R-squared:  0.8772, Adjusted R-squared:  0.8756
## F-statistic: 551.6 on 5 and 386 DF,  p-value: < 2.2e-16
```

Question 2) Please download the Excel data file security_questions.xlsx from Canvas. In your analysis, you can either try to read the data sheet from the Excel file directly from R (there might be a package for that!) or you can try to export the data sheet to a CSV file before reading it into R.

An online marketing firm is studying how customers who shop on e-commerce websites over the winter holiday season perceive the security of e-commerce sites. Based on feedback from experts, the company has created eighteen questions (see 'questions' tab of excel file) regarding security considerations at e-commerce websites. Over 400 customers responded to these questions (see 'data' tab of Excel file). Respondents were asked to consider a shopping site they were familiar with when answering questions (site was chosen randomly from those each subject has recently visited).

The company now wants to use the results of these eighteen questions to reveal if there are some underlying dimensions of people's perception of online security that effectively capture the variance of these eighteen questions. Let's analyze the principal components of the eighteen items.

(a) How much variance did each extracted factor explain?

```
security_questions <- read.csv("./security_questions.csv")
sq_pca <- prcomp(security_questions)
sq_eigen <- eigen(cor(security_questions))
sq_eigen$values/sum(sq_eigen$values)
```

```
##  [1] 0.51727518 0.08868511 0.06386435 0.04233199 0.03750784 0.03398131
##  [7] 0.02794364 0.02601549 0.02510951 0.02139980 0.01971565 0.01673928
## [13] 0.01623763 0.01456354 0.01303216 0.01280357 0.01159706 0.01119690
```

(b) How many dimensions would you retain, according to the two criteria we discussed?

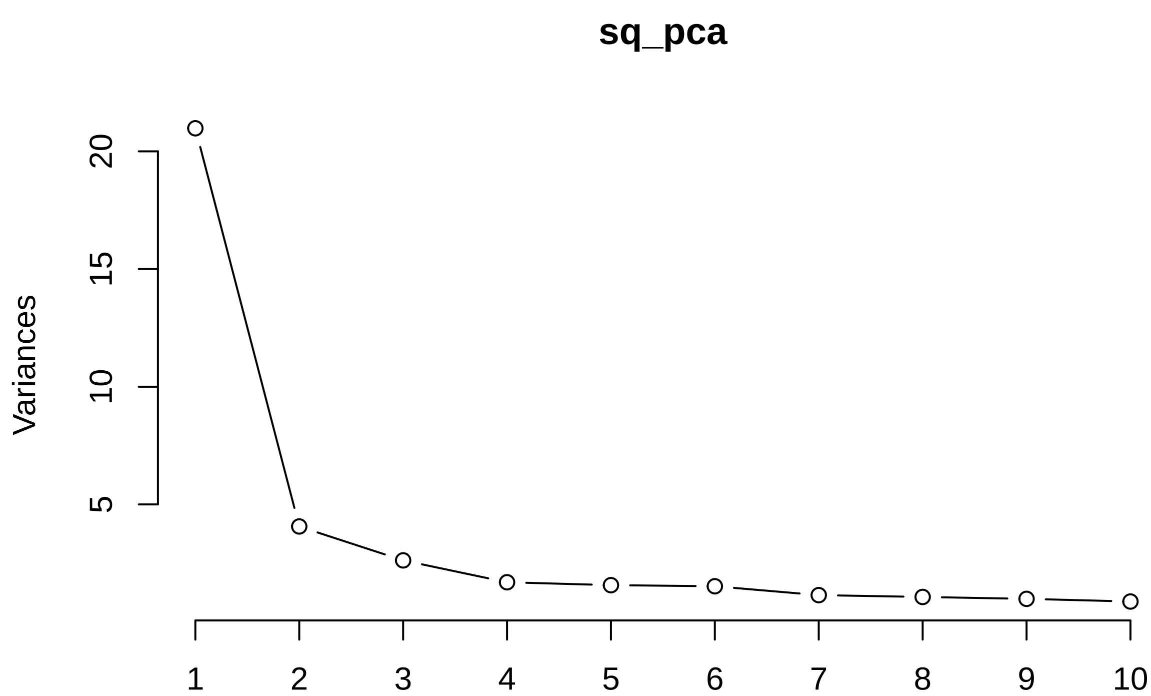
(Eigenvalue >= 1 and Scree Plot – can you show the screeplot with eigenvalue=1 threshhold?)

```
sq_eigen$values
```

```
##  [1] 9.3109533 1.5963320 1.1495582 0.7619759 0.6751412 0.6116636 0.5029855
##  [8] 0.4682788 0.4519711 0.3851964 0.3548816 0.3013071 0.2922773 0.2621437
## [15] 0.2345788 0.2304642 0.2087471 0.2015441
```

Only consider PCs with eigenvalues >= 1.

```r
screeplot(sq_pca, type="lines")
```



Choose only first one principal components.

Question 3) Let's simulate how principal components behave interactively.

- Install the package devtools in RStudio
- Run devtools::install_github("soumyaray/compstatslib")
- Load the new library: library(compstatslib)
- Run the interactive simulation using: interactive_pca()

(a) Create an oval shaped scatter plot of points that stretches in two directions – you should find that the principal component vectors point in the major and minor directions of variance (dispersion). Show this visualization.

install.packages("devtools") library(devtools) devtools::install_github("soumyaray/compstatslib") library(compstatslib) interactive_pca()

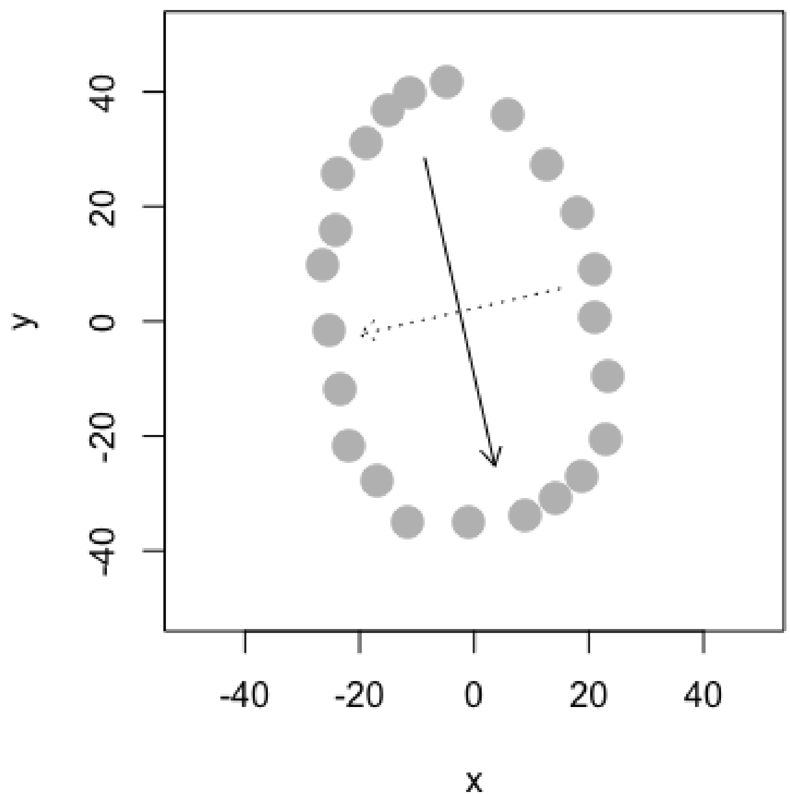


Figure 1: (a) pic

(b) Can you create a scatterplot whose principal component vectors do NOT seem to match the major directions of variance? Show this visualization.

```
> interactive_pca()
Click on the plot to create data points; hit [esc] to stop$points
            x          y
1   -4.823463  41.7400478
2    5.810944  36.0430440
3   12.647348  27.3076382
4   17.964552  18.9520326
5   21.002954   9.0772261
6   21.002954   0.7216205
7   23.281756  -9.5329864
8   22.901955 -20.5471937
9   18.724152 -27.0037980
10  14.166549 -30.8018005
11   8.849346 -33.8402026
12  -1.025461 -34.9796033
13 -11.659868 -34.9796033
14 -16.977071 -27.7633985
15 -21.914475 -21.6865945
16 -23.433676 -11.8117879
17 -25.332677  -1.5571810
18 -26.472078   9.8368266
19 -24.193276  15.9136306
20 -23.813476  25.7884372
21 -18.876073  31.1056407
22 -15.078070  36.8026445
23 -11.280068  39.8410466

$pca
Standard deviations (1, .., p=2):
[1] 27.57679 18.03285

Rotation (n x k) = (2 x 2):
         PC1        PC2
x  0.2228377 -0.9748556
y -0.9748556 -0.2228377
```

Figure 2: (a) data

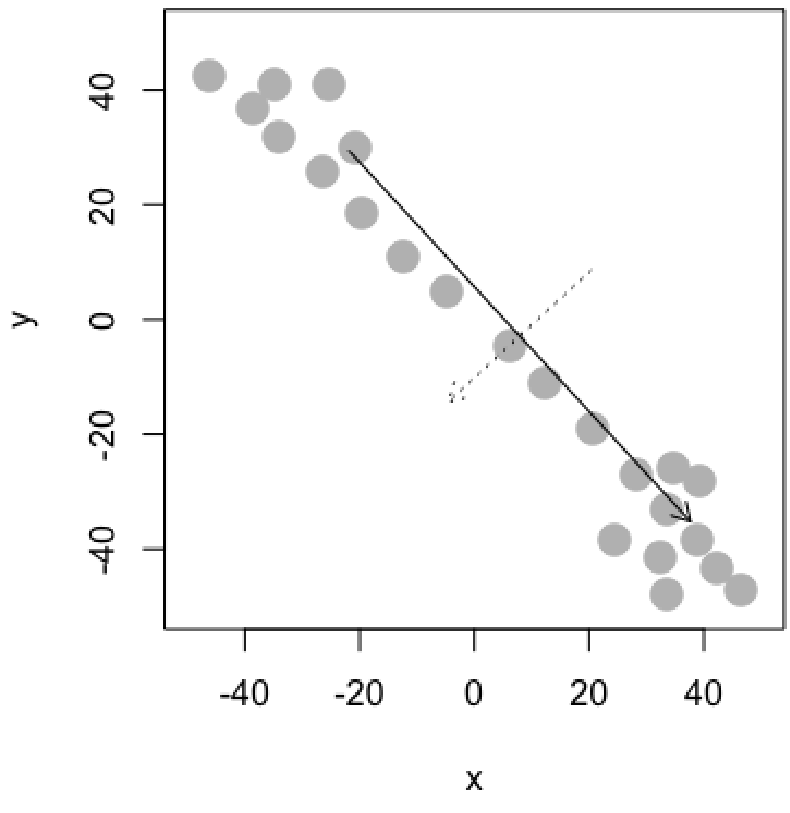


Figure 3: (b) pic

```
> interactive_pca()
Click on the plot to create data points; hit [esc] to stop$points
            x          y
1  -46.221691  42.499648
2  -38.625686  36.802645
3  -34.068083  31.865241
4  -26.472078  25.788437
5  -19.635673  18.572232
6  -12.419468  10.976227
7   -4.823463   4.899423
8    6.190744  -4.595583
9   12.267548 -11.052187
10  20.623154 -19.027993
11  28.219159 -27.003798
12  33.536362 -33.080602
13  42.271768 -43.335209
14  46.449571 -47.133211
15  38.853566 -38.397806
16 -34.827683  40.980447
17 -25.332677  40.980447
18 -20.775074  29.966240
19  24.421156 -38.397806
20  33.536362 -47.892812
21  32.396962 -41.436208
22  39.233366 -28.143199
23  34.675763 -25.864397
24  60.122380  53.134055

$pca
Standard deviations (1, .., p=2):
[1] 43.95132 16.92646

Rotation (n x k) = (2 x 2):
         PC1        PC2
x  0.6774851 -0.7355365
y -0.7355365 -0.6774851
```

Figure 4: (b) data