BACS HW12

Student ID: 110065508

Due: 2022/05/15

Special thanks to 108071001

Question 1. First, reuse the code of HW12 to read the dataset.

```r
# Question 1
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_log <- with(cars, data.frame(log(mpg), log(cylinders), log(displacement),
                                  log(horsepower), log(weight),
                                  log(acceleration), model_year, origin))
cars_log <- na.omit(cars_log)
```

(a) Construct a linear regression model to explore the multicollinearity.

```r
# Question 1 (a-i)
mpg_all <- lm(log.mpg. ~ log.cylinders. + log.displacement. + log.horsepower. +
                log.weight. + log.acceleration. + model_year + factor(origin),
              data=cars_log)
summary(mpg_all)
car::vif(mpg_all)

keeps <- c("log.cylinders.", "log.displacement.", "log.horsepower.",
           "log.weight.")
cars_mc <- cars_log[keeps] # mc for multicollinearity
```

(i) By the code

`> car::vif(mpg_all)`,

we found that `log.cylinders.`, `log.displacement.`, `log.horsepower.`, and `log.weight.` are the four variables with high multicollinearity, as the result shows:

```
> car::vif(mpg_all)
                       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
>
```

The new `data.frame` item is stored in `cars_mc`.

(ii)(iii) We knows that the first eigenvalue λ_1 of the correlation matrix is the amount of variance explained by the first principal component, i.e., the first eigenvector $v_1 = \text{PC}_1$.

```r
# Question 1 (a-ii)
# Question 1 (a-ii)
cars_mc_eigen <- eigen(cor(cars_mc)) # eigenvalues
cars_mc_eigenvalues <- cars_mc_eigen$values
# The 1st eigenvalue is the proportion of variance of first PC
cars_mc_eigenvalues[1]

# Question 1 (a-iii)
cars_mc_eigenvectors <- cars_mc_eigen$vectors # PCA
cars_mc_eigenvectors[,1] # PC1
```

```
> cars_mc_eigenvalues[1]
[1] 3.674259
> cars_mc_eigenvectors[,1]
[1] -0.4979145 -0.5122968 -0.4856159 -0.5037960
>
```

Hence, we have `cars_mc_eigenvectors[,1]` $= \lambda_1 = 3.674259$. Also, since the value of each component in $\texttt{PC}_1$ is close, we may say that $\texttt{PC}_1$ captures all the information of these four variables.

(b) By the following code, compare the summary of `pca_regr` and `pca_regr_std`

```r
# Question 1 (b-i)
cars_mc_pca <- prcomp(cars_mc)
scores = cars_mc_pca$x
cars_log$composite_score <- scores[,"PC1"]

# Question 1 (b-ii)
pca_regr <- lm(log.mpg.~ composite_score + log.acceleration. + model_year +
                        factor(origin), data=cars_log)

# Question 1 (b-iii)
pca_regr_std <- lm(log.mpg.~ scale(composite_score) + scale(log.acceleration.) +
                      scale(model_year) + factor(origin), data=cars_log)
```

```
> summary(pca_regr)

Call:
lm(formula = log.mpg. ~ composite_score + log.acceleration. +
    model_year + factor(origin), data = cars_log)

Residuals:
     Min       1Q   Median       3Q      Max
-0.53593 -0.06148  0.00149  0.06293  0.50928

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        1.395518   0.172873   8.073 8.84e-15 ***
composite_score    0.387073   0.014110  27.433  < 2e-16 ***
log.acceleration. -0.189830   0.043246  -4.390 1.47e-05 ***
model_year         0.029244   0.001871  15.628  < 2e-16 ***
factor(origin)2   -0.010840   0.020738  -0.523    0.601
factor(origin)3    0.002243   0.020517   0.109    0.913
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1239 on 386 degrees of freedom
Multiple R-squared:  0.8689,	Adjusted R-squared:  0.8672
F-statistic: 511.7 on 5 and 386 DF,  p-value: < 2.2e-16

> |
```

```
> summary(pca_regr_std)

Call:
lm(formula = log.mpg. ~ scale(composite_score) + scale(log.acceleration.) +
    scale(model_year) + factor(origin), data = cars_log)

Residuals:
     Min       1Q   Median       3Q      Max
-0.53593 -0.06148  0.00149  0.06293  0.50928

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)               3.099741   0.009151 338.727  < 2e-16 ***
scale(composite_score)    0.283038   0.010317  27.433  < 2e-16 ***
scale(log.acceleration.) -0.034351   0.007826  -4.390 1.47e-05 ***
scale(model_year)         0.107729   0.006893  15.628  < 2e-16 ***
factor(origin)2          -0.010840   0.020738  -0.523    0.601
factor(origin)3           0.002243   0.020517   0.109    0.913
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1239 on 386 degrees of freedom
Multiple R-squared:  0.8689,	Adjusted R-squared:  0.8672
F-statistic: 511.7 on 5 and 386 DF,  p-value: < 2.2e-16

> |
```

No matter doing the standardization or not, the new column is the most important. ∎

Question 2. I am too lazy to read `.xlsx` file in R. Hence I just save the second worksheet as a new `.csv` file.

```r
# Question 2
security <- read.csv("security_questions.csv")
sec_eigen <- eigen(cor(security))
sec_pca <- prcomp(security, scale. = TRUE)
summary(sec_pca)

png(filename = "2.png")
screeplot(sec_pca, type="lines")
dev.off()
```

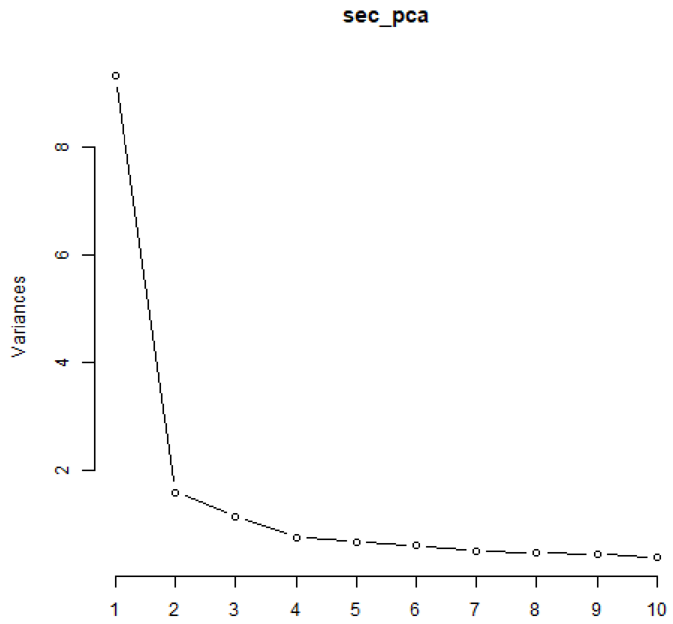
(a) The amount of variance each extracted factor explain is actually the eigenvalues. Hence, using the command

`> sec_eigen$values`

We have

```
> sec_eigen$values
 [1] 9.3109533 1.5963320 1.1495582 0.7619759 0.6751412 0.6116636 0.5029855 0.4682788
 [9] 0.4519711 0.3851964 0.3548816 0.3013071 0.2922773 0.2621437 0.2345788 0.2304642
[17] 0.2087471 0.2015441
```

(b) According to the eigenvalue $\geqslant 1$ and the criteria screeplot criteria, It looks like I should choose the first 3 components. However, I choose only 2 of them at last. I'll say why later.



(c) Let's view the values of PCs.

	PC1	PC2	PC3	PC4	PC5	PC6
Q1	-0.2677422	0.110341691	-0.001973491	0.126220668	-0.048468417	0.1826730451
Q2	-0.2204272	0.010886972	0.083171536	0.258122218	0.093887919	0.7972988590
Q3	-0.2508767	0.025878543	0.083648794	-0.399268076	-0.061766335	0.1343170710
Q4	-0.2042919	-0.508981768	0.100759585	0.040690031	-0.072913141	-0.0683434170
Q5	-0.2261544	0.024745268	-0.505845415	0.052574743	-0.193207848	0.1493338250
Q6	-0.2237681	0.082805088	0.193281966	-0.004209098	0.611348765	0.0551361412
Q7	-0.2151891	0.251398450	0.302354487	0.327318232	0.008596733	-0.0562329401
Q8	-0.2576225	-0.033526840	-0.320109219	0.076017162	0.209097752	-0.2005009349
Q9	-0.2369512	0.183342667	0.189853454	-0.124795087	0.025138160	-0.2696485391
Q10	-0.2248660	0.078103267	-0.496820932	-0.034236123	-0.249119125	0.0232597277
Q11	-0.2467645	0.206580870	0.160903091	0.264607608	-0.210724202	-0.1928970917
Q12	-0.2065785	-0.504591429	0.113342400	0.060346524	0.052819352	-0.0454546580
Q13	-0.2333066	0.051159791	0.078658760	-0.602543012	-0.030357718	0.0949114194
Q14	-0.2659342	0.078910404	0.146232765	-0.362581586	-0.086718158	-0.0006735609
Q15	-0.2307289	-0.008373326	-0.310161141	0.069411508	0.513508897	-0.2572918341
Q16	-0.2482681	0.160524168	0.170839887	0.204337585	-0.342722070	-0.2189544787
Q17	-0.2023781	-0.525747030	0.102652280	0.080754652	-0.157376900	-0.0527365890
Q18	-0.2643810	0.089915229	-0.060800871	0.051492827	-0.024214541	-0.0327588454

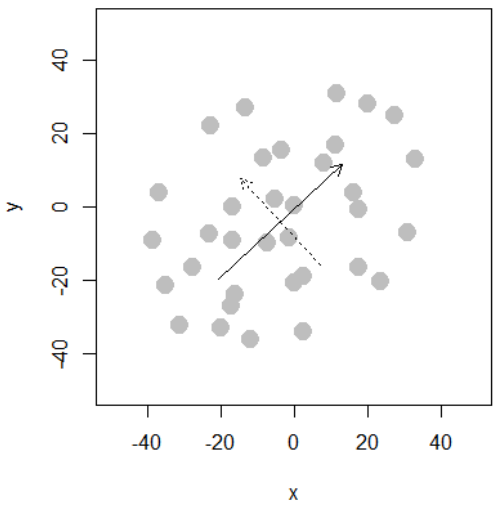
By checking the values of each component, I decide to choose 2 only since it looks like 2 PCs are enough, as the above figure shows. Also, the investigate of problems support my ideas. Let's group the problems according to first two PC's.

- Group 1: Q1, Q2, Q3, Q5, Q6, Q8, Q9, Q10, Q11, Q13, Q14, Q15, Q16, Q18.
- Group 2: Q4, Q7, Q12, Q17.

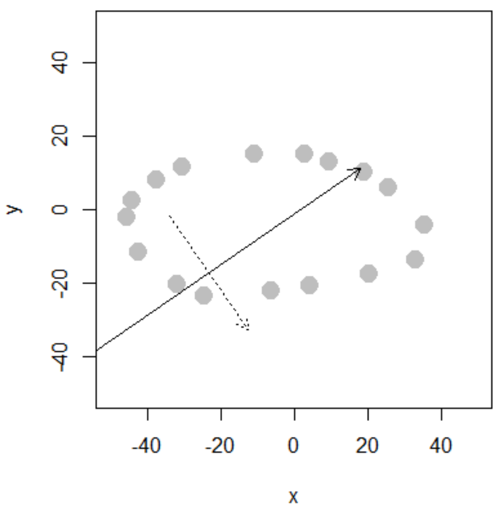
For group 1, the questions may sum up as "information security". For group 2, the questions ask whether the site is convenient and protect the users from denial of transaction.

■

Question 3. (a) The oval shaped scatter plot:



(b) I create such a scatter plot by placing some crazy outliers at the place (lower left) that the plot cannot display.



∎