ISE 5103 Intelligent Data Analytics
Assignment #2

Name: Jiwon Jeon
Date: 09/19/2016

1 Concordance and Discordance

```
x = c(3,4,2,1,7,6,5)  #a vector x with 7 elements
y = c(4,3,7,6,5,2,1)  #a vector y with 7 elements
tableXY = table(x,y)
ConDisPairs(tableXY)[c("C","D")]
```

Concordance: 6
Discordance: 15

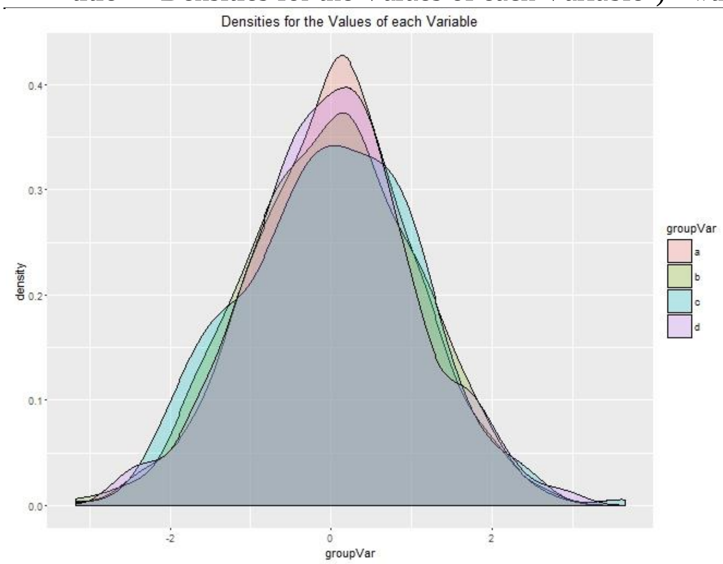
2 Generating data and advanced density plots

(a)

```
a = rnorm(500)  #a vector a with 500 normally distributed random numbers
b = rnorm(500)  #a vector b with 500 normally distributed random numbers
c = rnorm(500)  #a vector c with 500 normally distributed random numbers
d = rnorm(500)  #a vector d with 500 normally distributed random numbers
df = data.frame(a,b,c,d)  #a data frame df with 500 rows and 4 variables
df2 = melt(df, variable.name = "groupVar", value.name = "value")  #a data frame df2 by reforming the data df
                                                                   from a "wide" format to a "long" format
```

(b)

```
ggplot(df2, aes(x=value, fill=groupVar)) +                #identify data & variable
  geom_density(alpha=0.25) +                              #set geometry and transparency
  labs(x = "groupVar",                                    #x-axis label
       title = "Densities for the Values of each Variable")  #title
```



3 Shark Attacks

(a) The data quality might have been affected by the factors such as the availability of dates of accidents especially for early dates, the difference between the dates of attacks happened and the dates of report/record, and the accuracy of dates.

(b)

```
sharkData = read.csv("ISE 5103 GSAF.csv", header = TRUE)  #load the original shark attack data
GSAFdata = sharkData[-(1:4069),]                          #clean the shark attack data
```

(c)

```
GSAFdata$DateR = as.Date(GSAFdata$Date, "%d-%b-%y")   #create a new variable which converts the
                                                        character date type into R date type
for (i in 1:length(GSAFdata$DateR))                   #clean the new date column
  if(GSAFdata$DateR[i] > as.Date("2016-01-01") && !is.na(GSAFdata$DateR[i]))
    GSAFdata$DateR[i] = as.Date(GSAFdata$Date, "%d-%b-%Y")

head_GSAFdata = head(GSAFdata)
```

	Case.Number	Date	Year	Fatal..Y.N.	…	DateR
4070	2000.00.00	2000	2000	N	…	*NA*
4071	2000.01.05	5-Jan-00	2000	Y	…	2000-01-05
4072	2000.01.28.R	Reported 28-Jan-2000	2000	Y	…	*NA*
4073	2000.02.01	1-Feb-00	2000	N	…	2000-02-01
4074	2000.02.03	3-Feb-00	2000	N	…	2000-02-03
4075	2000.02.14	14-Feb-00	2000	N	…	2000-02-14

(d)

```
mean(is.na(GSAFdata$DateR))  #ratio of missing new date field
```

9.399% of new date field is missing. When R converted the character date type into R date type, it did not properly transform some data that were missing day/month information or having descriptive strings in data themselves. Those unconverted dates were indicated as NA in R date type.

(e)

```
GSAFdata = subset(GSAFdata, !GSAFdata$DateR==is.na(GSAFdata$DateR))  #delete rows of missing date
```

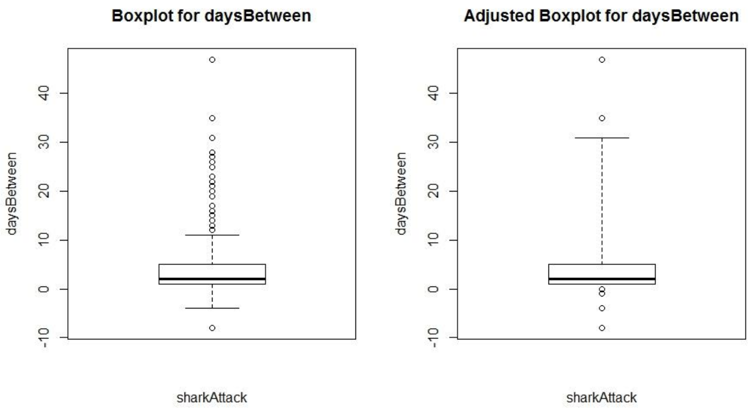
(f) i. create a vector daysBetween

```
daysBetween = as.vector(diff(GSAFdata$DateR))  #create a vector daysBetween
daysBetween = c(NA,daysBetween)              #add a missing value as the first element of daysBetween
GSAFdata$daysBetween = daysBetween           #add the vector daysBetween in GSAFdata
```

ii.

```
boxplot(GSAFdata$daysBetween,              #boxplot of daysBetween
        main = "Boxplot for daysBetween",   #main plot title
        xlab = "sharkAttack",               #x-axis label
        ylab = "daysBetween")               #y-axis label

adjbox(GSAFdata$daysBetween,                         #boxplot of daysBetween
       main = "Adjusted Boxplot for daysBetween",    #main plot title
       xlab = "sharkAttack",                         #x-axis label
       ylab = "daysBetween")                         #y-axis label
```

The boxplot identifies too many upper outliers compared to the adjusted boxplot. This indicates that the standard boxplot outlier rule is not appropriate for highly asymmetric data. The adjusted boxplot in the robustbase package in R (adjbox command) allows more skewed distribution by incorporating a measure of skewness; the nominal distance of 1.5 IQD's is multiplied by a scale factor computed from the medcouple value of data – the scale factor for lower outliers is exp(-3.5 Mc) while that for upper outliers is exp(+4Mc). Therefore, the adjusted boxplot performs better for highly skewed data.



iii.

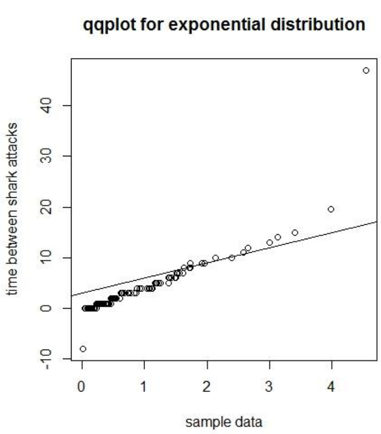
```
grubbs.test(GSAFdata$daysBetween)  #grubb's test for daysBetween
```

The Grubb's test identifies only one highest outlier, 47, where p-value is less than 0.05. Therefore the skewness of the data is not fully analyzed to detect significant ourliers other than 47. The generalized ESD test identifies ten upper outliers, 27 23 21 31 28 26 22 25 35 47 21 21, showing better performance than the Grubb's test. However, the generalized ESD also does not properly cover the skewness. (See R script for generalized ESD)

(g)

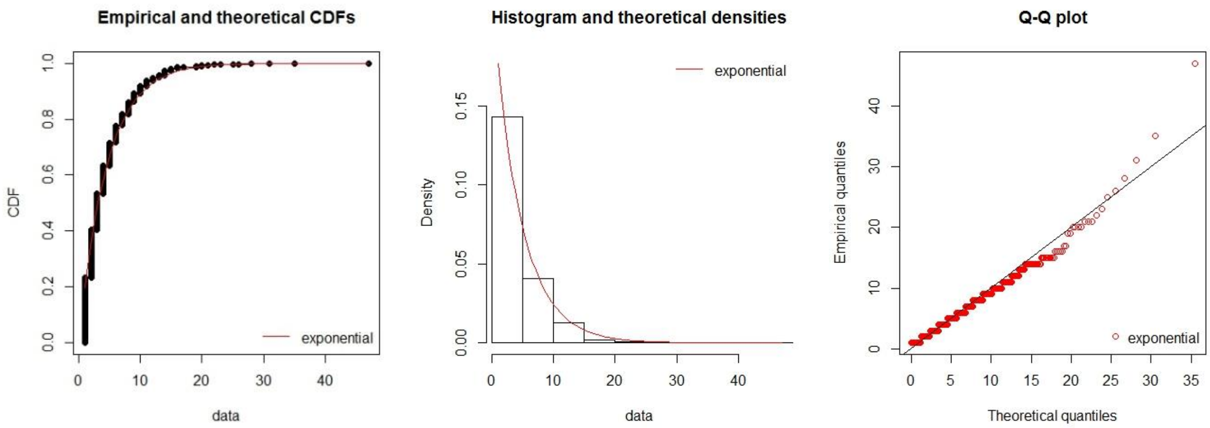
```
sample = rexp(1500)                          #generate sample exponential distribution
qqplot(sample, GSAFdata$daysBetween,         #qqplot to evalute the time between shart attack
      main = "qqplot for exponential distribution",
      xlab = "sample data",
      ylab = "time between shark attacks")
qqline(GSAFdata$daysBetween)                 #add qqline
```

The time between shark attacks shows exponential distribution til 20 days, and deviates from the trend afterwards.



(h)

```
GSAFdata = GSAFdata[GSAFdata$daysBetween>0,]
GSAFdata = GSAFdata[-1,]
fitData = fitdist(GSAFdata$daysBetween, "exp")    #transform data to be a 'fitdist' object using exponential
fitData2 = fitdist(GSAFdata$daysBetween, "pois")  #transform data to be a 'fitdist' object using poisson

cdfcomp((list(fitData)), legendtext = "exponential")
denscomp((list(fitData)), legendtext = "exponential")
qqcomp((list(fitData)), legendtext = "exponential")
```



The GSAFdata$daysBetween fits well to the exponential distribution as shown in the plots of cdfcomp, denscomp, and qqcomp.

```
gofstat((list(fitData, fitData2)), fitnames = c("exponential","pois"))
```

```
Goodness-of-fit statistics
                             exponential       pois
Kolmogorov-Smirnov statistic   0.1980238  0.2337676
Cramer-von Mises statistic     4.6075709 14.0539077
Anderson-Darling statistic    34.8984554        Inf

Goodness-of-fit criteria
                               exponential     pois
Aikake's Information Criterion    6294.712 7694.115
Bayesian Information Criterion    6299.846 7699.248
```

(i) The shark attacks seem to occur as a Poission process because the data fit well to the exponential distribution for the attribute of time between shark attacks. However, it has to be emphasized that the trend does not maintain the exponential relation if the time difference is greater than 20 days.

4 Missing Data

(a)

```
data(freetrade)         #load the data freetrade
md.pairs(freetrade)     #display number of observations per variable pair
md.pattern(freetrade)   #display missing-data patterns
```

Using a command *md.pattern* in package mice, the missing data patterns in freetrade are displayed as below;

```
   year pop gdp.pc usheg polity signed intresmi fiveop tariff country
96    1   1      1     1      1      1        1      1      1       0   1
52    1   1      1     1      1      1        1      1      0       0   2
 2    1   1      1     1      0      1        1      1      1       0   2
 1    1   1      1     1      1      0        1      1      1       0   2
 5    1   1      1     1      1      1        1      0      1       0   2
 2    1   1      1     1      1      0        1      1      0       0   3
 9    1   1      1     1      1      1        0      0      1       0   3
 4    1   1      1     1      1      1        0      0      0       0   4
      0   0      0     0      2      3       13     18     58     171 265
```

(b)

```
tariff_country = freetrade[,c(2,3)]                              #data frame with country and tariff
tariff_country[is.na(tariff_country$tariff),]["tariff"] = 0     #assign 0 to missing values
tariff_country$tariff = ifelse(tariff_country$tariff>=1,1,0)     #assign 1 to the rest of values
table_tc = table(tariff_country$tariff,tariff_country$country)   #table of tariff and country with missing and
                                                                  nonmissing values
```

```
chisq.test(table_tc)                                                    #chiSquare test
removeNepal = tariff_country[!tariff_country$country == "Nepal",]    #remove Nepal from dataset
table_nepal = table(removeNepal$tariff,removeNepal$country)
chisq.test(table_nepal)
removePhilippines = tariff_country[!tariff_country$country == "Philippines",]         #remove Philippines
table_Philippines = table(removePhilippines$tariff,removePhilippines$country)
chisq.test(table_Philippines)
```

From chiSquare test, p-value is 0.003283 which is less than 0.05. Therefore, we reject null hypothesis and the evidence suggest that the variables tariff and country are dependent. When we remove Nepal from the dataset, the p-value of chiSquare test is 0.02666 and still the null hypothesis is rejected. When we remove Philippines, the p-value becomes 0.1188 and is greater than 0.05. The tariff and country are now independent. This is because Philippines does not contain missing values in tariff therefore much affects the relation between tariff and country when it is removed.

5 Principal Component Analysis

(a) i. The correlation matrix of all the attributes of mtcars

```
corMat = as.matrix(cor(mtcars))   #correlation matrix of all the attributes of mtcars
```

ii.

```
eigen(corMat, symmetric = TRUE)   #eigenvalues and eigenvectors of corMat
```

Eigenvalues:

```
[1] 6.60840025 2.65046789 0.62719727 0.26959744 0.22345110 0.21159612
[7] 0.13526199 0.12290143 0.07704665 0.05203544 0.02204441
```

Eigenvectors:

```
            [,1]        [,2]        [,3]         [,4]        [,5]        [,6]
 [1,]  0.3625305 -0.01612440 -0.22574419 -0.022540255 -0.10284468 -0.10879743
 [2,] -0.3739160 -0.04374371 -0.17531118 -0.002591838 -0.05848381  0.16855369
 [3,] -0.3681852  0.04932413 -0.06148414  0.256607885 -0.39399530 -0.33616451
 [4,] -0.3300569 -0.24878402  0.14001476 -0.067676157 -0.54004744  0.07143563
 [5,]  0.2941514 -0.27469408  0.16118879  0.854828743 -0.07732727  0.24449705
 [6,] -0.3461033  0.14303825  0.34181851  0.245899314  0.07502912 -0.46493964
 [7,]  0.2004563  0.46337482  0.40316904  0.068076532  0.16466591 -0.33048032
 [8,]  0.3065113  0.23164699  0.42881517 -0.214848616 -0.59953955  0.19401702
 [9,]  0.2349429 -0.42941765 -0.20576657 -0.030462908 -0.08978128 -0.57081745
[10,]  0.2069162 -0.46234863  0.28977993 -0.264690521 -0.04832960 -0.24356284
[11,] -0.2140177 -0.41357106  0.52854459 -0.126789179  0.36131875  0.18352168
              [,7]         [,8]         [,9]       [,10]        [,11]
 [1,]  0.367723810  0.754091423  0.235701617  0.13928524 -0.124895628
 [2,]  0.057277736  0.230824925  0.054035270 -0.84641949 -0.140695441
 [3,]  0.214303077 -0.001142134  0.198427848  0.04937979  0.660606481
 [4,] -0.001495989  0.222358441 -0.575830072  0.24782351 -0.256492062
 [5,]  0.021119857 -0.032193501 -0.046901228 -0.10149369 -0.039530246
 [6,] -0.020668302  0.008571929  0.359498251  0.09439426 -0.567448697
 [7,]  0.050010522  0.231840021 -0.528377185 -0.27067295  0.181361780
 [8,] -0.265780836 -0.025935128  0.358582624 -0.15903909  0.008414634
 [9,] -0.587305101  0.059746952 -0.047403982 -0.17778541  0.029823537
[10,]  0.605097617 -0.336150240 -0.001735039 -0.21382515 -0.053507085
[11,] -0.174603192  0.395629107  0.170640677  0.07225950  0.319594676
```

iii. Principal components of the mtcars attributes

```
pca = prcomp(mtcars, scale. = TRUE)     #principal components of mtcars
```

```
            PC1         PC2         PC3          PC4         PC5         PC6
mpg  -0.3625305  0.01612440 -0.22574419 -0.022540255  0.10284468 -0.10879743
cyl   0.3739160  0.04374371 -0.17531118 -0.002591838  0.05848381  0.16855369
disp  0.3681852 -0.04932413 -0.06148414  0.256607885  0.39399530 -0.33616451
hp    0.3300569  0.24878402  0.14001476 -0.067676157  0.54004744  0.07143563
```

```
drat -0.2941514  0.27469408  0.16118879  0.854828743  0.07732727  0.24449705
wt    0.3461033 -0.14303825  0.34181851  0.245899314 -0.07502912 -0.46493964
qsec -0.2004563 -0.46337482  0.40316904  0.068076532 -0.16466591 -0.33048032
vs   -0.3065113 -0.23164699  0.42881517 -0.214848616  0.59953955  0.19401702
am   -0.2349429  0.42941765 -0.20576657 -0.030462908  0.08978128 -0.57081745
gear -0.2069162  0.46234863  0.28977993 -0.264690521  0.04832960 -0.24356284
carb  0.2140177  0.41357106  0.52854459 -0.126789179 -0.36131875  0.18352168
              PC7          PC8          PC9        PC10         PC11
mpg   0.367723810 -0.754091423  0.235701617  0.13928524 -0.124895628
cyl   0.057277736 -0.230824925  0.054035270 -0.84641949 -0.140695441
disp  0.214303077  0.001142134  0.198427848  0.04937979  0.660606481
hp   -0.001495989 -0.222358441 -0.575830072  0.24782351 -0.256492062
drat  0.021119857  0.032193501 -0.046901228 -0.10149369 -0.039530246
wt   -0.020668302 -0.008571929  0.359498251  0.09439426 -0.567448697
qsec  0.050010522 -0.231840021 -0.528377185 -0.27067295  0.181361780
vs   -0.265780836  0.025935128  0.358582624 -0.15903909  0.008414634
am   -0.587305101 -0.059746952 -0.047403982 -0.17778541  0.029823537
gear  0.605097617  0.336150240 -0.001735039 -0.21382515 -0.053507085
carb -0.174603192 -0.395629107  0.170640677  0.07225950  0.319594676
```

iv.

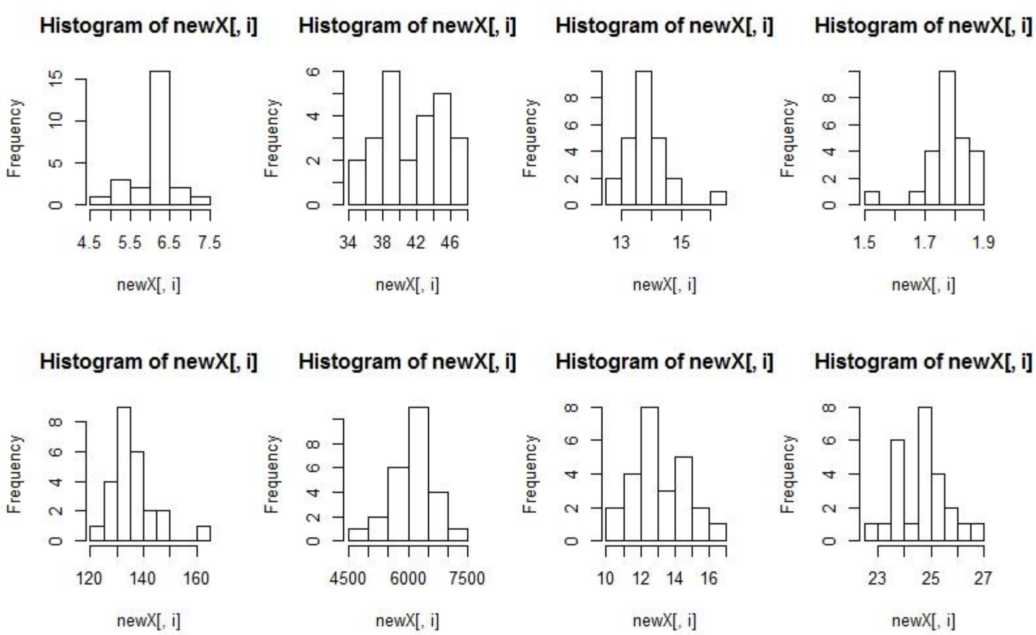
```
PCs = as.matrix(pca$rotation)      #principal components of mtcars
PC1 = PCs[,1]                      #principal component 1
PC2 = PCs[,2]                      #principal component 2
innerProduct = PC1 %*% PC2         #inner product (dot product) of PC1 and PC2
```

The absolute values of the results from ii and iii are same. This is because *prcomp* computes the principal components by scaling the original data in mtcars using z-score, which is represented by the eigenvectors of the correlation matrix corMat. The signs in results ii and iii are opposite because the principal components must have been defined to lie in the opposite direction of the eigenvectors.

v. Inner product of PC1 and PC2 is calculated as -2.775558e-17, nearly 0. Therefore the two vectors PC1 and PC2 are orthogonal.

(b) i. The distributions of each variable are fairly normal.

```
par(mfrow = c(2,2))
apply(heptathlon[,1:8],2,hist)  #quick inspection of heptathlon histograms
dev.off()                       #cancel par(mfrow = c(2,2))
```



ii.

```
grubbs.test(heptathlon$hurdles)       #outlier for hurdles: Launa (PNG)
grubbs.test(heptathlon$highjump)      #outlier for highjump: Launa (PNG)
grubbs.test(heptathlon$shot)          #outlier for shot: Hui-Ing (TAI)
grubbs.test(heptathlon$run200m)       #outlier for run200m: Joyner-Kersee (USA)
grubbs.test(heptathlon$longjump)      #outlier for longjump: Launa (PNG)
grubbs.test(heptathlon$javelin)       #outlier for javelin: Scheider (SWI)
grubbs.test(heptathlon$run800m)   #outlier for run800m: Launa (PNG)
heptathlon = heptathlon[-25,]    #remove the outlier; Launa (PNG)
```

From Grubb's test, Launa (PNG) is found to be an outlier. She was an outlier for four events; hurdles, highjump, longjump and run800m.

iii.

```
heptathlon$hurdles = max(heptathlon$hurdles) - heptathlon$hurdles  #transform hurdles to make large values to be good
heptathlon$run200m = max(heptathlon$run200m) - heptathlon$run200m  #transform run200m to make large values to be good
heptathlon$run800m = max(heptathlon$run800m) - heptathlon$run800m  #transform run800m to make large values to be good
```

iv. Principal components of Hpca

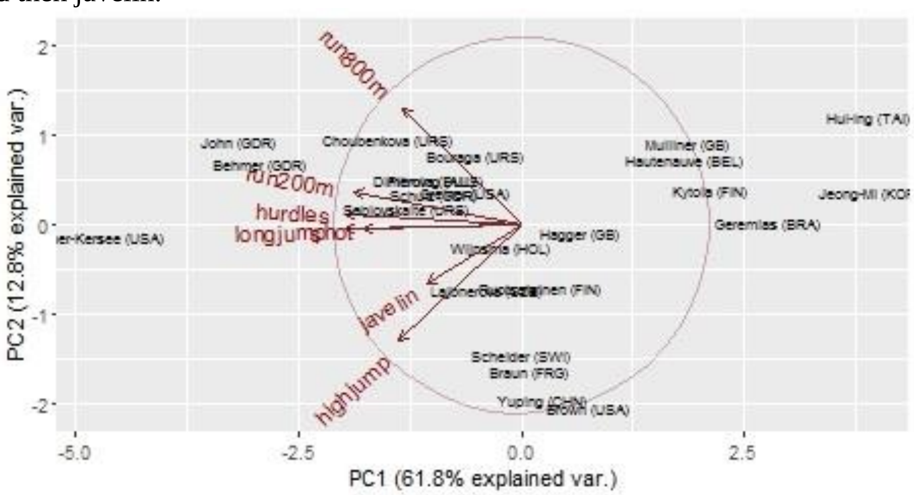
```
Hpca = prcomp(heptathlon[,1:7], scale. = TRUE)  #principal component analysis on the 7 events
```

```
               PC1         PC2        PC3         PC4         PC5         PC6         PC7
hurdles  -0.4503876  0.05772161 -0.1739345  0.04840598 -0.19889364  0.84665086 -0.06961672
highjump -0.3145115 -0.65133162 -0.2088272 -0.55694554  0.07076358 -0.09007544  0.33155910
shot     -0.4024884 -0.02202088 -0.1534709  0.54826705  0.67166466 -0.09886359  0.22904298
run200m  -0.4270860  0.18502783  0.1301287  0.23095946 -0.61781764 -0.33279359  0.46971934
longjump -0.4509639 -0.02492486 -0.2697589 -0.01468275 -0.12151793 -0.38294411 -0.74940781
javelin  -0.2423079 -0.32572229  0.8806995  0.06024757  0.07874396  0.07193437 -0.21108138
run800m  -0.3029068  0.65650503  0.1930020 -0.57418128  0.31880178 -0.05217664  0.07718616
```

v.

```
ggbiplot(Hpca, obs.scale = 1, var.scale = 1,
        varname.size = 4, labels.size = 2.5,
        circle = TRUE, labels = rownames(heptathlon))
```

PC1 is mainly explained by longjump, hurdles, shot, and run200m. PC2 is largely affected by highjump, run800m and then javelin.



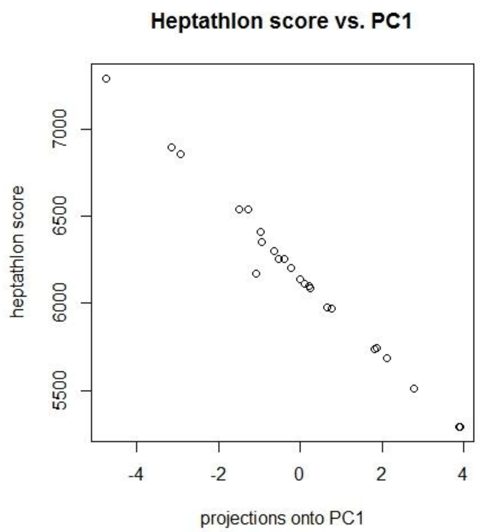
vi. The heptathlon score vs. projections on PC1

```
Hpca$x[,1]                              #projections onto PC1
plot(Hpca$x[,1], heptathlon$score,      #plot of the heptathlon score vs. projections on PC1
    main = "Heptathlon score vs. PC1",  #main title
    xlab = "projections onto PC1",      #x-axis label
    ylab = "heptathlon score")          #y-axis label
as.vector(Hpca$x[,1])
```

The projections on PC1 for each competitor are calculated as below;

```
[1] -4.757530189 -3.147943402 -2.926184760 -1.288135516 -1.503450994
[6] -0.958467101 -0.953445060 -0.633239267 -0.381571974 -0.522322004
[11] -0.217701500 -1.075984276  0.003014986  0.109183759  0.208868056
[16]  0.232507119  0.659520046  0.756854602  1.880932819  1.828170404
[21]  2.118203163  2.770706272  3.901166920  3.896847898
```

The proportion of PC1 variance takes up 61.77%. When plotting the heptathlon score vs. these projections, the score is well aligned with the projections on PC1 showing (negative) linear relationship. This means that the heptathlon score is highly affected by PC 1.

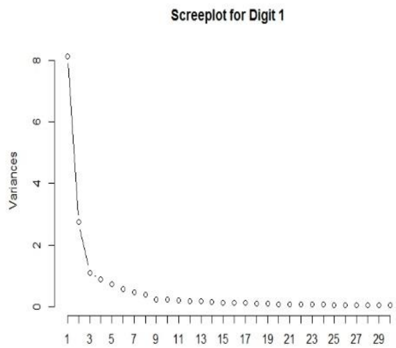


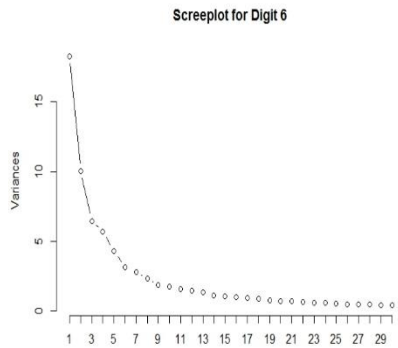
(c) i. PCA analysis on digit 1, digit 6 and digit 7

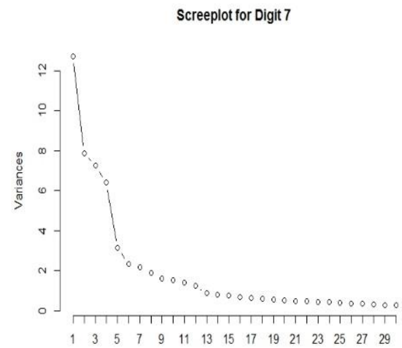
```
digit1 = read.csv("train.1", header = FALSE)   #load the data digit 1
digit6 = read.csv("train.6", header = FALSE)   #load the data digit 6
digit7 = read.csv("train.7", header = FALSE)   #load the data digit 7

pca1 = prcomp(digit1)  #PCA analysis on digit 1 without scaling
pca6 = prcomp(digit6)  #PCA analysis on digit 6 without scaling
pca7 = prcomp(digit7)  #PCA analysis on digit 7 without scaling

head(pca1$rotation[,1:5])  #rotation matrix of PCA digit 1
head(pca6$rotation[,1:5])  #rotation matrix of PCA digit 6
head(pca6$rotation[,1:5])  #rotation matrix of PCA digit 7

head(pca1$x[,1:5])  #projections of digit 1 on PC's
head(pca6$x[,1:5])  #projections of digit 6 on PC's
head(pca7$x[,1:5])  #projections of digit 7 on PC's
```

```
summary(pca1)  #proportion of variance of each PC for digit 1
summary(pca6)  #proportion of variance of each PC for digit 6
summary(pca7)  #proportion of variance of each PC for digit 7

screeplot(pca1, npcs = 30, type = "lines", main = "Screeplot for Digit 1")  #screeplot for PCA digit 1
screeplot(pca6, npcs = 30, type = "lines", main = "Screeplot for Digit 6")  #screeplot for PCA digit 6
screeplot(pca7, npcs = 30, type = "lines", main = "Screeplot for Digit 7")  #screeplot for PCA digit 7
```

The number of principal components of each digit data to downsize the dimensions is chosen to cover ≥75% of variance. The screeplots and cumulative proportion to the chosen PC's are as below;







Digit 1: Cumulative proportion to **PC7** = 80.78%
Digit 6: Cumulative proportion to **PC14** = 75.15%
Digit 7: Cumulative proportion to **PC12** = 74.74%

ii. Image data usually have high dimensions (attributes) and require classification in analysis. Using PCA, we can reduce the dimensions and simplify the data set to deal with.

6 Kaggle.com

(a) Survival on the Titanic: this data contains the passenger information on Titanic survived and dead in the tragedy.
(b) Analysis on Titanic

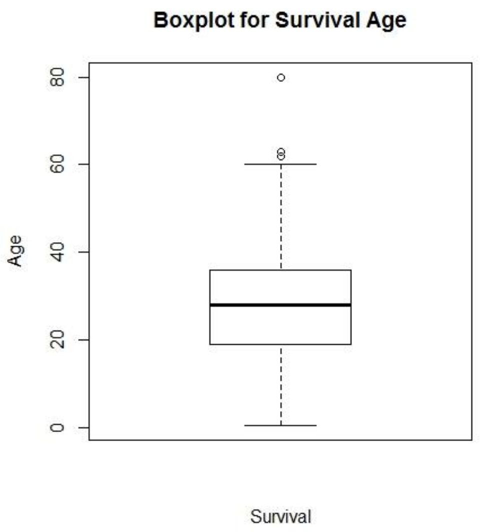
```
titanic = read.csv("titanic.csv", header = TRUE)
attach(titanic)

nrow(titanic)          #number of rows
ncol(titanic)          #number of variables
plot(titanic)          #plot of variables
md.pairs(titanic)      #missingness in data per variable
md.pattern(titanic)    #missingness in data total

titanic = titanic[!is.na(titanic$Age),]                  #clean data by removing missing values in Age
titanic_survival = titanic[titanic$Survived == 1,]   #clean data for survivals

boxplot(titanic_survival$Age,                        #boxplot of Survival Age
        main = "Boxplot for Survival Age",
        xlab = "Survival",
        ylab = "Age")

grubbs.test(titanic_survival$Age)                    #find the strongest outlier
```

```
ggplot(titanic_survival, aes(x=Age, fill=Survived)) +  #density plot for the Survival's age
  geom_density(alpha=0.25) +
  labs(x = "age",
       title = "Survival by Age")
```

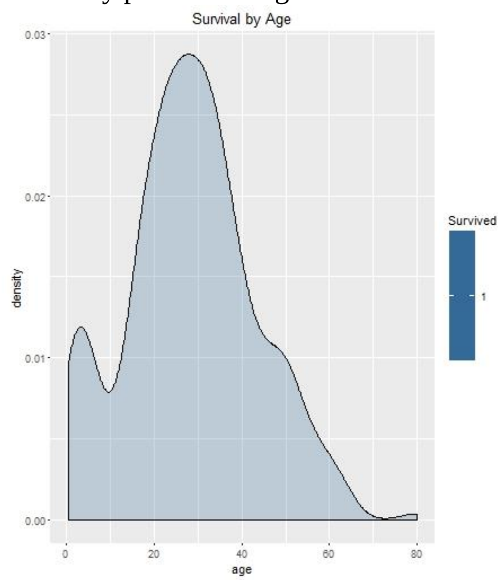
Number of rows: 891
Number of variables: 12
Missingness:

```
     PassengerId Survived Pclass Name Sex Age SibSp Parch Ticket Fare Cabin Embarked
[1,]           1        1      1    1   1   1     1     1      1    1     1        1 0
[2,]           0        0      0    0   0   0     0     0      0    0     0        0 0
```

Boxplot for the age of survivals



Density plot for the age of survivals



Outlier in the Age by the Grubb’s test: 80