Higher School of Economics

Contemporary Data Analysis

NHL 2014-2015 Personal statistics of the National Hockey League players

Group:

Maxim Demidov

Vasily Mosin

Anastasia Feygina

Contents

Problem 1. Explanation of the choice of the dataset **4**

Problem 2.1 **6**

Problem 2.2 **9**

Problem 2.3 **11**

Problem 2.4 **13**

Problem 3.1 **16**

Problem 3.2 **17**

Problem 3.3 **19**

Problem 3.4 **20**

Problem 3.5 **23**

Problem 3.6 **24**

Results of genetic algorithm: 24

Problem 4.1 **27**

Problem 4.2 **28**

Problem 4.3 **30**

Problem 4.4 **32**

Statistics style (check simple hypothesis) 32

Mirking style 33

Problem 5.1 **34**

Linear regression for normalized data: 34

Problem 5.2 **38**

Determinacy coefficient 38

Confidence interval 39

Problem 5.3 **40**

Problem 6.1 **44**

Problem 6.2 **45**

Problem 6.3 **49**

Problem 7.1 **51**

Problem 7.2 **56**

Problem 1. Explanation of the choice of the dataset

Source:
There are 881 players in our dataset. The dataset was taken from http://war-onice.com

Relevance:
In this project, we consider personal statistics of the National Hockey League players in previous season 2014-2015. In ice hockey, analytics is the analysis of the characteristics of hockey players and teams through the use of statistics and other tools to gain a greater understanding of the effects of their performance. There are several commonly used in ice hockey features in our dataset.

Description of features:

1. Pos - is a position of a player (wing forward, central forward or defenseman). It is a categorical feature.
2. Team - the player's team. It's just additional information. It won't be used in our work.
3. Gm - a number of games a player participated in season.
4. Age - the player's age.
5. Salary - the player's salary ($ a year).
6. G - a number of goals a player scored in season.
7. A - a number of player's assists in season.
8. P - a number of player's points in season (goals + assists).
9. G60 - goals scored per 60 minutes.
10. A60 - assists recorded per 60 minutes.
11. P60 - points recorded per 60 minutes.
12. PenD - Penalty Differential (a differential between number of penalties drawn by the player and number of penalties taken by the player). It's clear that the more this feature the more valuable player is. If this feature is negative then the number of penalties taken by the player is more than the number of penalties drawn by the player. Such player harms to his team.
13. CF% - Corsi for percentage of total. It's very important feature in hockey analytics. CF% = Corsi For / (Corsi For + Corsi Against), where Corsiis the sum of shots on goal, missed shots and blocked shots. "For" means made by player's team, "Against" means made by opponents. Most players have a Corsi For percentage (CF%) between 40 and 60. A player ranked above 55 is often considered "elite".
14. PDO - is the sum of a team's shooting percentage and its save percentage (while player is on the ice). PDO is usually measured at even strength, and based on the theory that most teams will ultimately regress toward a sum of 100, is often viewed as a proxy for how lucky a team is. A player with a PDO over 102 is "probably not as good as they seem", while a player or team below 98 likely is better than they appear.
15. PSh% - the player's personal shooting percentage for goal. In other words, the proportion of goals of all the shoots. It might show how accurate the player is.
16. ZSO%Rel - Percentage of all on-ice non-Neutral Zone Faceoffs in the Offensive Zone minus the Percentage of all off-ice non-Neutral ZoneFaceoffs in the Deffensive Zone. A player who has a high zone start ratio will often have increased Corsi numbers due to starting in the offensive zone, while a player with a low zone start ratio will often have depressedCorsi numbers. Strategically, coaches may give their best offensive players more offensive zone starts to try and create extra scoring chances, while a team's best defensive players will typically have more defensive zone starts.

17. TOI/Gm - Time on ice per game.

Justification of choice: Examples of problems:

- To show correlation between P60 and PDO;
- To find out is the team is playing better with the player on the ice by using CF%;
- To find out which of the features (or combination of the features) attempt to more specifically address individual performance.

Problem 2.1

All information in summary:

```
data = read.csv("data.csv", sep = ",")
summary(data)
```

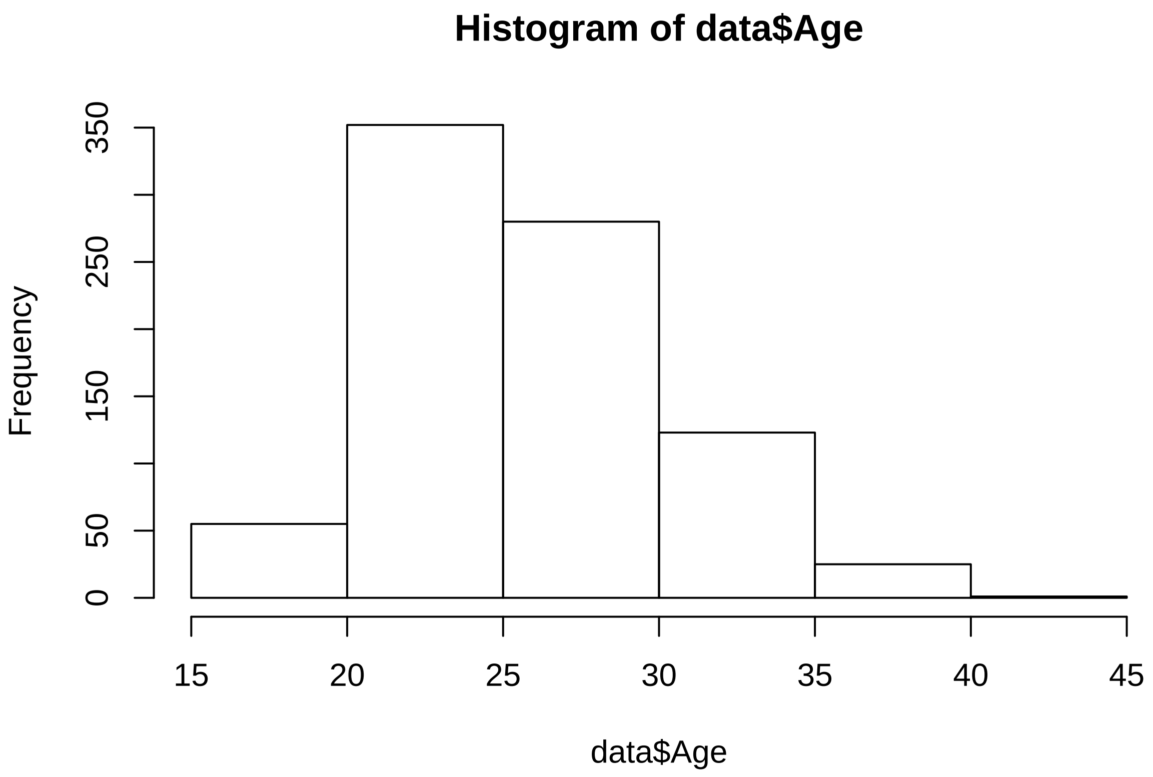
```
##                Name          pos           Team           Gm
##  Aaron.Ekblad    :  1   D      :290   CGY    : 31   Min.   :  1.00
##  Aaron.Volpatti  :  1   C      :160   EDM    : 31   1st Qu.: 28.00
##  Adam.Burish     :  1   L      : 73   N.J    : 29   Median : 65.00
##  Adam.Clendening:  1   R      : 71   CBJ    : 28   Mean   : 55.69
##  Adam.Cracknell  :  1   RL     : 70   COL    : 28   3rd Qu.: 81.00
##  Adam.Henrique   :  1   LR     : 61   NYI    : 28   Max.   :108.00
##  (Other)         :830   (Other):111   (Other):661
##       Age            Salary             G                A
##  Min.   :18.00   Min.   : 0.550   Min.   : 0.000   Min.   : 0.00
##  1st Qu.:23.00   1st Qu.: 0.750   1st Qu.: 1.000   1st Qu.: 3.00
##  Median :26.00   Median : 1.000   Median : 5.000   Median :10.00
##  Mean   :26.24   Mean   : 2.217   Mean   : 8.219   Mean   :14.07
##  3rd Qu.:29.00   3rd Qu.: 3.362   3rd Qu.:13.000   3rd Qu.:21.00
##  Max.   :42.00   Max.   :14.000   Max.   :58.000   Max.   :65.00
##
##        P              G60              A60              P60
##  Min.   : 0.00   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.: 4.00   1st Qu.:0.1000   1st Qu.:0.4200   1st Qu.:0.630
##  Median :15.50   Median :0.3800   Median :0.7300   Median :1.140
##  Mean   :22.28   Mean   :0.4603   Mean   :0.7812   Mean   :1.242
##  3rd Qu.:35.00   3rd Qu.:0.7100   3rd Qu.:1.0800   3rd Qu.:1.780
##  Max.   :95.00   Max.   :5.0000   Max.   :5.5300   Max.   :5.530
##
##       PenD               CF.             PDO              PSh.
##  Min.   :-25.0000   Min.   :20.00   Min.   : 50.00   Min.   :  0.000
##  1st Qu.: -4.0000   1st Qu.:44.44   1st Qu.: 97.77   1st Qu.:  2.840
##  Median :  0.0000   Median :49.72   Median : 99.81   Median :  6.705
##  Mean   : -0.8433   Mean   :49.40   Mean   : 99.12   Mean   :  7.210
##  3rd Qu.:  2.0000   3rd Qu.:54.81   3rd Qu.:101.17   3rd Qu.: 10.530
##  Max.   : 27.0000   Max.   :72.22   Max.   :125.00   Max.   :100.000
##
##     ZSO.Rel            TOI.Gm
##  Min.   :-55.810   Min.   : 4.49
##  1st Qu.: -8.533   1st Qu.:12.23
##  Median :  1.785   Median :15.39
##  Mean   :  1.285   Mean   :15.26
##  3rd Qu.: 12.110   3rd Qu.:18.23
##  Max.   : 58.620   Max.   :28.77
##
```

```
colnames(data)
```

```
##  [1] "Name"    "pos"     "Team"    "Gm"      "Age"     "Salary"  "G"
##  [8] "A"       "P"       "G60"     "A60"     "P60"     "PenD"    "CF."
## [15] "PDO"     "PSh."    "ZSO.Rel" "TOI.Gm"
```

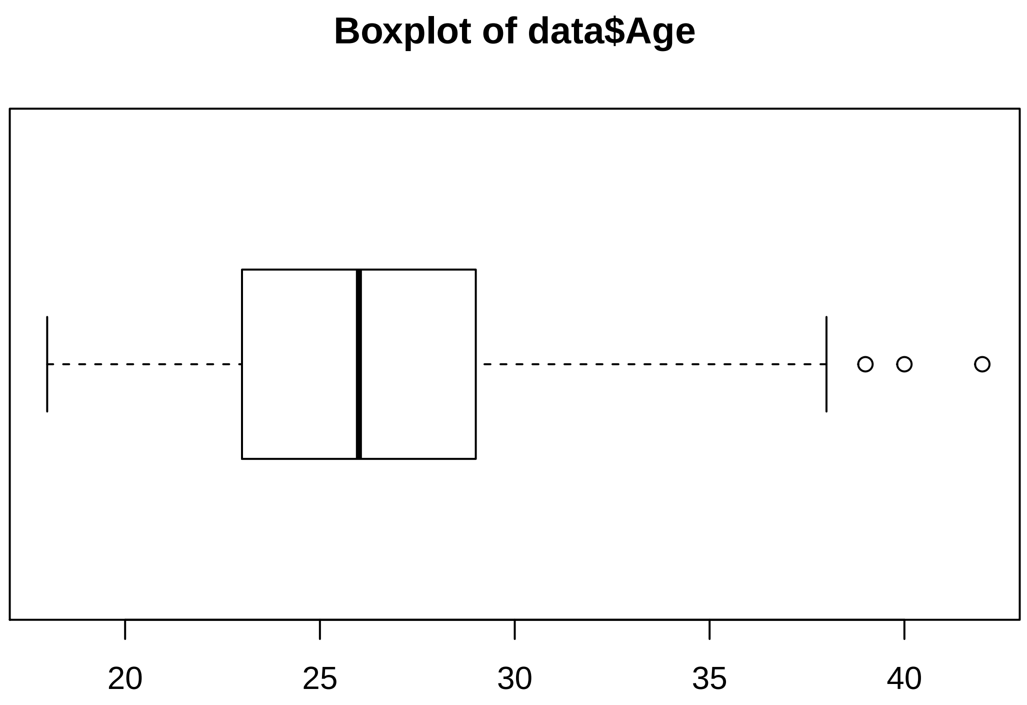
As we know age is one of the most important factors in hockey, a lot of other characteristics depends on it. Also the hockey experts usually compare the average age of teams and then make some assumptions about how the teams play.

Histogram of players Age:



boxplot:

```
boxplot(data$Age, main = 'Boxplot of data$Age', horizontal = TRUE)
```



mean:

```
mean(data$Age)
```

```
## [1] 26.23804
```

median:

```
median(data$Age)
```

```
## [1] 26
```

mode:

```
findMode <- function(arg){
data.t <- table(arg)
return(sort(unique(arg))[which.max(data.t)])
}
findMode(data$Age)
```

```
## [1] 23
```

Hockey clubs select players since 18 years and track them throughout their careers. But if a player has not proved himself up to a certain age, then club parted with him. The mode is 23 years, it shows that basically in this age clubs give players a last chance to show themselves. Later on most of them goes to the lower league. Therefore, there is a so large number of 23-year-old players in the league. A median more than mode just because the maximum players' age is 42 years, and in addition to the 23-year-old players core of the team for the most part consits of the older players.

Problem 2.2

statistical mean:

```
mean(data$Age)
```

```
## [1] 26.23804
```

```
n <- nrow(data)
m <- mean(data$Age)
s <- sd(data$Age)
conf.int <- c(m-1.965*s/sqrt(n), m+1.965*s/sqrt(n))
print(conf.int)
```

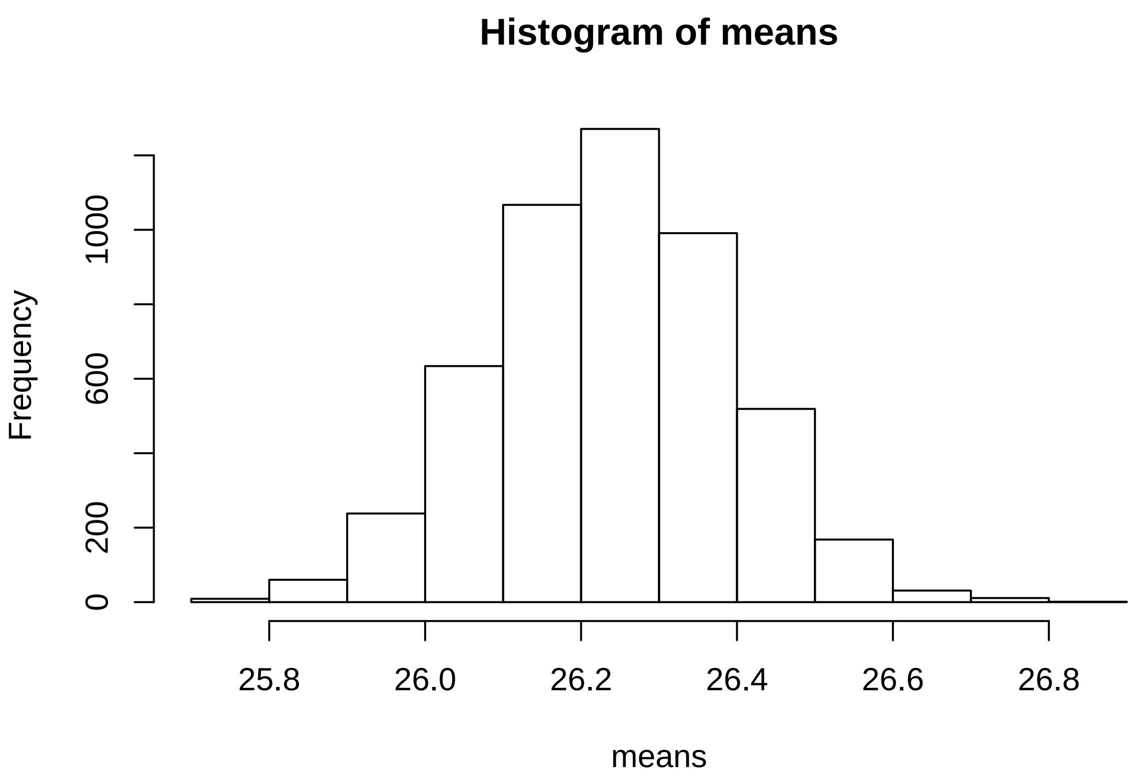
```
## [1] 25.93271 26.54336
```

bootstrap:

```
nn <- nrow(data)
means <- c()
for(i in 1:5000)
  {
  means[i] <- mean(sample(data$Age, nn, replace = TRUE))
  }
```

new histogram:

```
hist(means)
```



Because the histogram is similar to a normal distribution, we apply the technique of pivotal bootstrap.

```r
n <- 5000
m <- mean(means)
s <- sd(means)
pivotal.conf.int <- c(m-1.965*s/sqrt(n), m+1.965*s/sqrt(n))
print(pivotal.conf.int)
```

```
## [1] 26.23434 26.24292
```

non-pivotal bootstrap:

```r
sorted.means  <- sort(means)
nonpivotal.conf.int <- c(sorted.means[0.025*5000], sorted.means[5000-0.025*5000])
print(nonpivotal.conf.int)
```

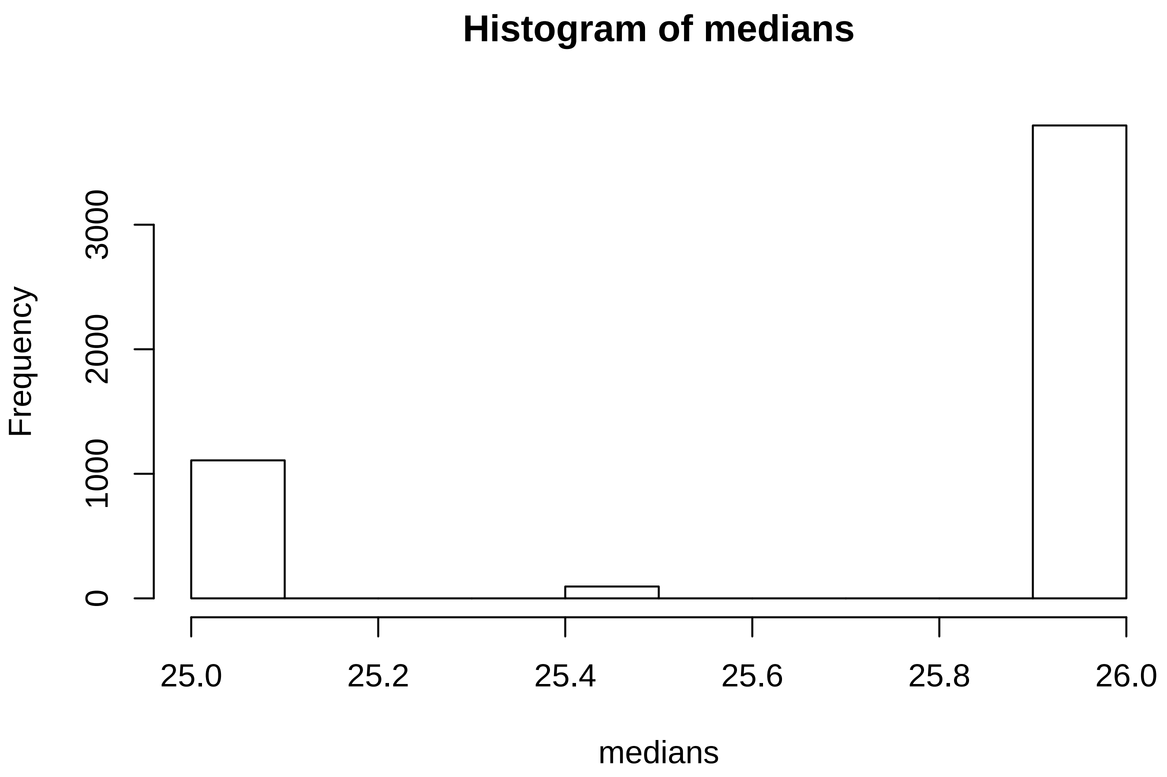
```
## [1] 25.93660 26.53947
```

As the initial distribution is close to the normal the pivotal bootstrap let us get more narrow confidence interval than the statistical method. Non-pivoral bootstrap is not so precise (confidence interval is bigger in this case) as it makes no assupmtion about initial distribution.

Problem 2.3

pivotal bootstrap for median:

```r
medians <- c()
for(i in 1:5000)
  {
  medians[i] <- median(sample(data$Age, nn, replace = TRUE))
  }
hist(medians)
```



Because the histogram is not similar to a normal distribution, we can't apply the technique of pivotal bootstrap here.

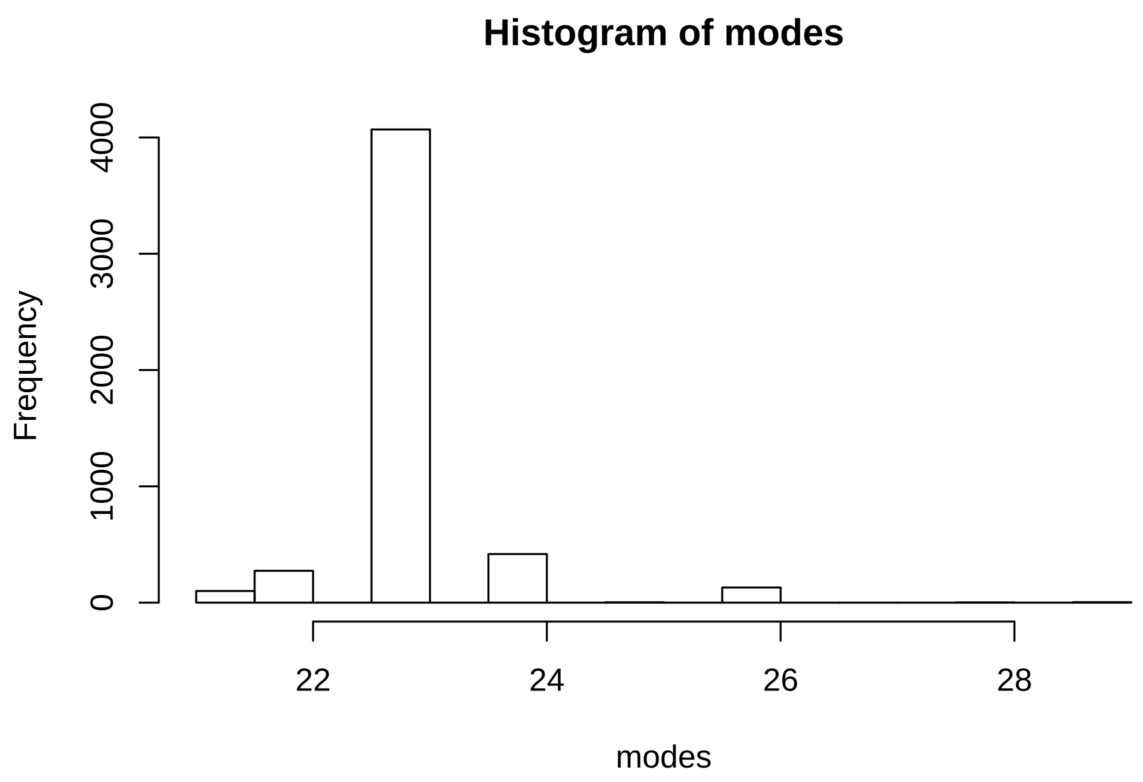
Non-pivotal bootstrap for median:

```r
sorted.medians  <- sort(medians)
nonpivotal.conf.int <- c(sorted.medians[0.025*5000], sorted.medians[5000-0.025*5000])
print(nonpivotal.conf.int)
```

```
## [1] 25 26
```

pivotal bootstrap for mode:

```r
modes <- c()
for(i in 1:5000)
  {
  modes[i] <- findMode(sample(data$Age,nn, replace = TRUE))
  }
hist(modes)
```



Because the histogram is not similar to a normal distribution, we can't apply the technique of pivotal bootstrap here.

Non-pivotal bootstrap for mode:

```
sorted.modes  <- sort(modes)
nonpivotal.conf.int <- c(sorted.modes[0.025*5000], sorted.modes[5000-0.025*5000])
print(nonpivotal.conf.int)
```

```
## [1] 22 26
```

Conclusions: The histogram for the median does not have a polymodal distribution and hasn't heavy tails, so the bootstrap in this case can be applied. For mode-case it is not quite so, because in the histogram we can see heavy tail on the right, this greatly affects the confidence interval. We can use non-pivotal bootstrap here but you need to be careful.

Problem 2.4

Devide players into two groups. Name players who has a salary higher than the median of all salaries in the league high-salary players, and who has a salary lower than the median - low-salary players. Consider their ages.

```
high.salary.players  <- data[data$Salary >= median(data$Salary),]
low.salary.players  <- data[data$Salary < median(data$Salary),]
print (mean(high.salary.players$Age))
```

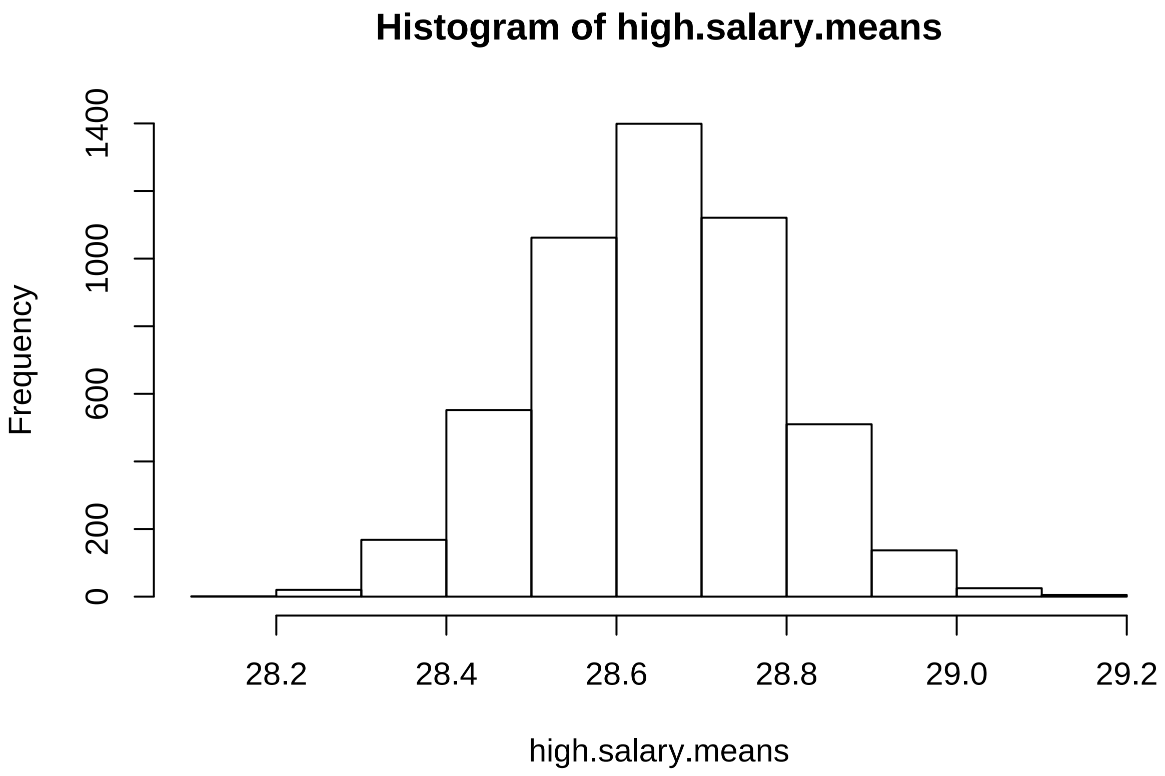
```
## [1] 28.64706
```

```
print (mean(low.salary.players$Age))
```

```
## [1] 23.74696
```

bootstrape for high.salary.means:

```
high.salary.means <- c()
for(i in 1:5000)
  {
  high.salary.means[i] <- mean(sample(high.salary.players$Age, nn, replace = TRUE))
  }
hist(high.salary.means)
```



Because the histogram is similar to a normal distribution, we apply the technique of pivotal bootstrap.

```r
n <- 5000
m <- mean(high.salary.means)
s <- sd(high.salary.means)
pivotal.conf.int <- c(m-1.965*s/sqrt(n), m+1.965*s/sqrt(n))
print(pivotal.conf.int)
```

```
## [1] 28.64509 28.65281
```

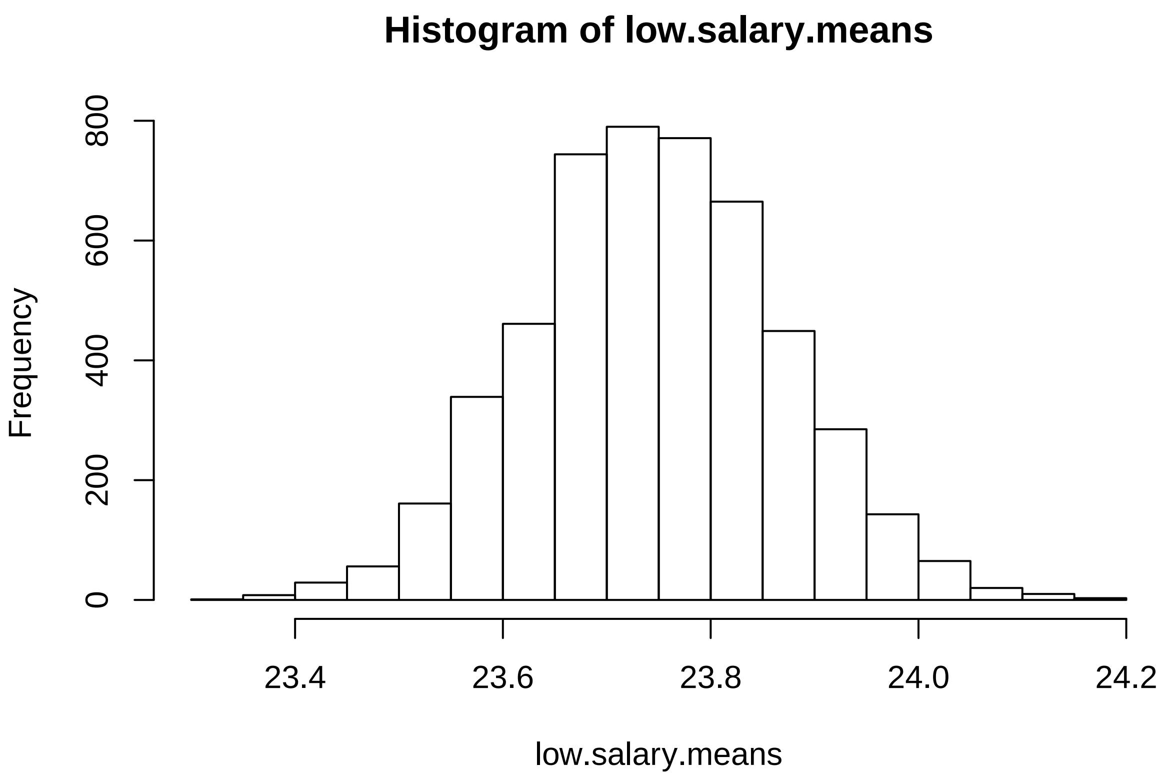
Non-pivotal bootstrap for high.salary.means:

```r
sorted.high.salary.means  <- sort(high.salary.means)
nonpivotal.conf.int <- c(sorted.high.salary.means[0.025*5000],
                         sorted.high.salary.means[5000-0.025*5000])
print(nonpivotal.conf.int)
```

```
## [1] 28.37679 28.91866
```

bootstrap for low.salary.means:

```r
low.salary.means <- c()
for(i in 1:5000)
  {
  low.salary.means[i] <- mean(sample(low.salary.players$Age, nn, replace = TRUE))
  }
hist(low.salary.means)
```



Because the histogram is similar to a normal distribution, we apply the technique of pivotal bootstrap.

```
n <- 5000
m <- mean(low.salary.means)
s <- sd(low.salary.means)
pivotal.conf.int <- c(m-1.965*s/sqrt(n), m+1.965*s/sqrt(n))
print(pivotal.conf.int)
```

```
## [1] 23.74243 23.74920
```

Non-pivotal bootstrap for high.salary.means:

```
sorted.low.salary.means  <- sort(low.salary.means)
nonpivotal.conf.int <- c(sorted.low.salary.means[0.025*5000],
                         sorted.low.salary.means[5000-0.025*5000])
print(nonpivotal.conf.int)
```

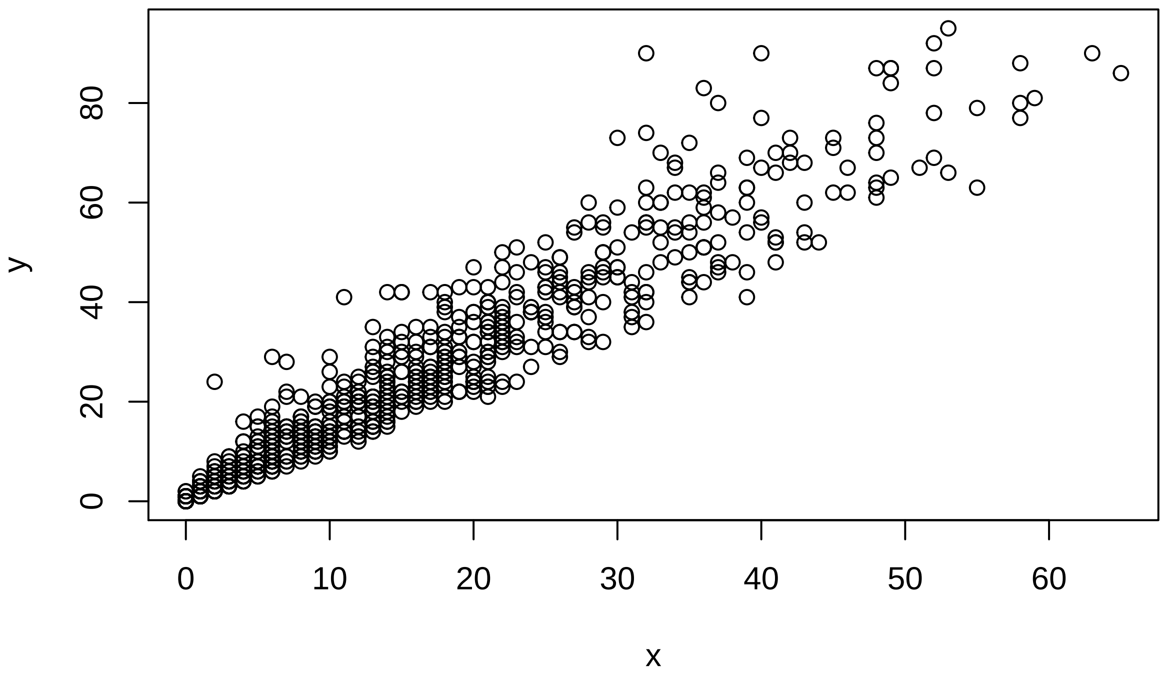
```
## [1] 23.51196 23.98684
```

Confidence intervals differs considerably. It is clear, that high-salary players are older than low-salary players on the average, because older players has more experience and, hense, are more valuable.

Problem 3.1

I used plot(data) function to analyse which features to use. More "linear-like" scatterplot:

```
data = read.csv("data.csv", sep = ",")
y <- data$P
x <- data$A
plot(x,y)
```



Problem 3.2

```
m <- lm(y ~ x)
slope <- m$coefficients[2]
intercept <- m$coefficients[1]
```

Slope:

```
print(slope)
```

```
##        x
## 1.513472
```

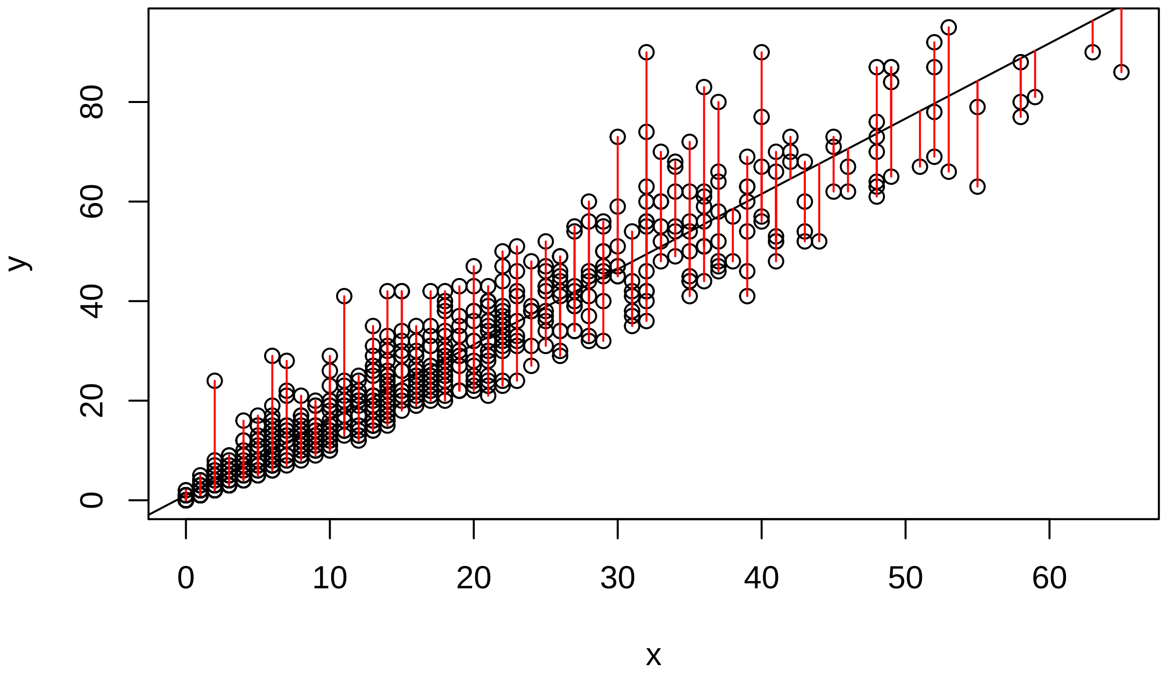
The slope is about 1.51 means that the player, who makes one more assist, gets 1.51 points more on the average. From the one side it seems to be meaningless. It can be only explained by the fact that the players, who makes more assists, makes more goals also (as usual), hence gets more points.

Intercept:

```
print(intercept)
```

```
## (Intercept)
##   0.9965111
```

```
plot(x,y)
abline(m)
# calculate residuals and predicted values
res <- signif(residuals(m), 5)
pre <- predict(m) # plot distances between points and the regression line
segments(x, y, x, pre, col="red")
```



```r
summary(m)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.237  -2.591  -0.997   1.923  40.572
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.99651    0.30204   3.299  0.00101 **
## x            1.51347    0.01549  97.733  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.05 on 834 degrees of freedom
## Multiple R-squared:  0.9197, Adjusted R-squared:  0.9196
## F-statistic:  9552 on 1 and 834 DF,  p-value: < 2.2e-16
```

Problem 3.3

Correlation:

```
cor(x,y)
```

```
## [1] 0.9590089
```

Determinacy coefficient:

```
cor(x,y)*cor(x,y)
```

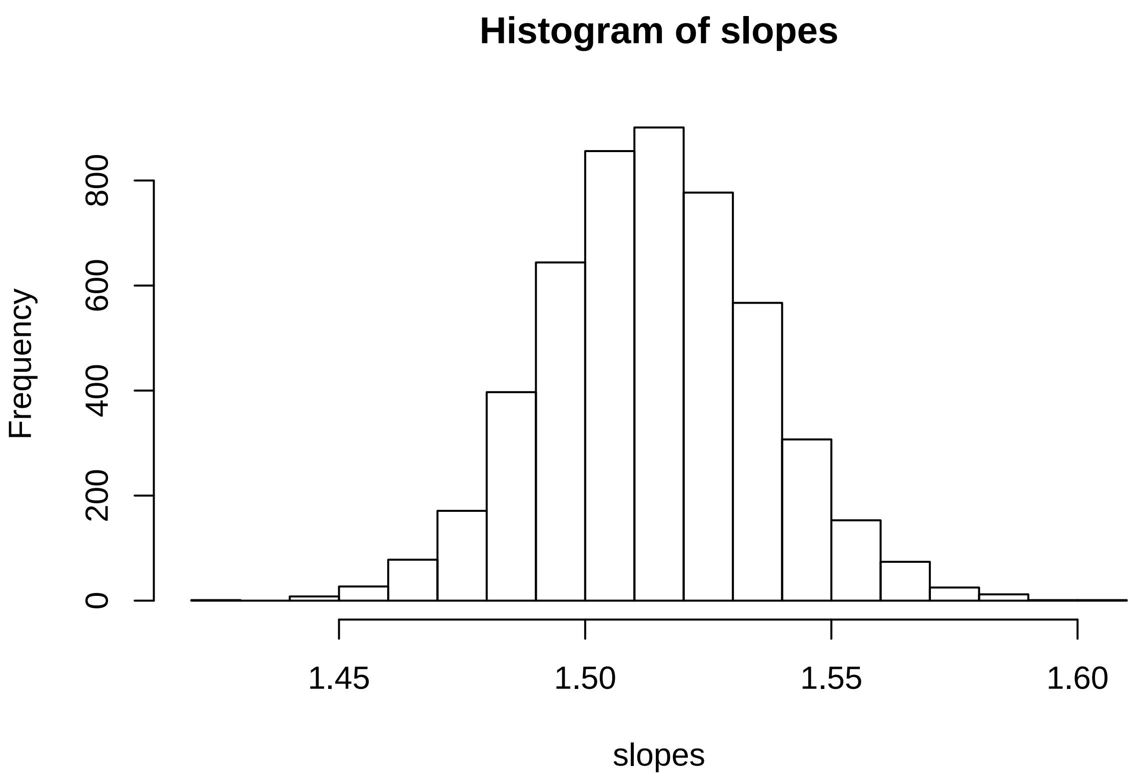
```
## [1] 0.9196981
```

Determinacy coefficient signifies the rate of the explained deviation in relation to total data scatter. Higher Coefficient values suggest that the chosen model fits data quite well. It does not imply causality in general, however, just represents the relation between chosen features of the data.

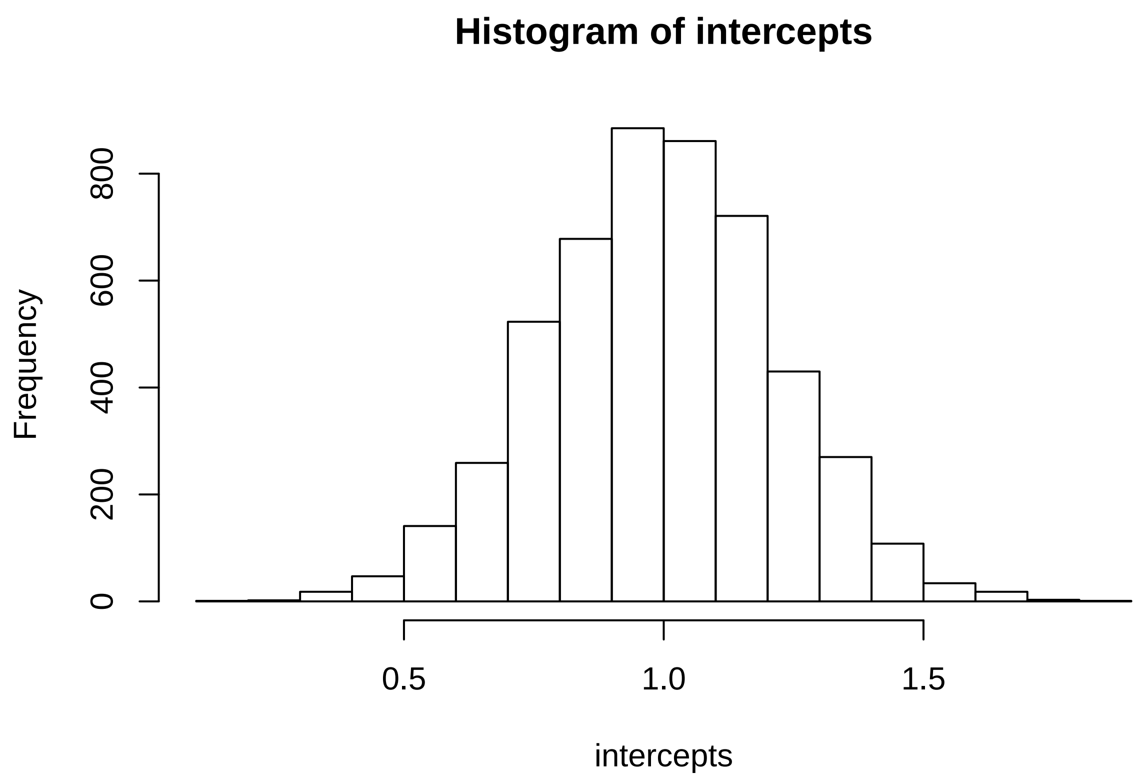
Problem 3.4

bootstrap:

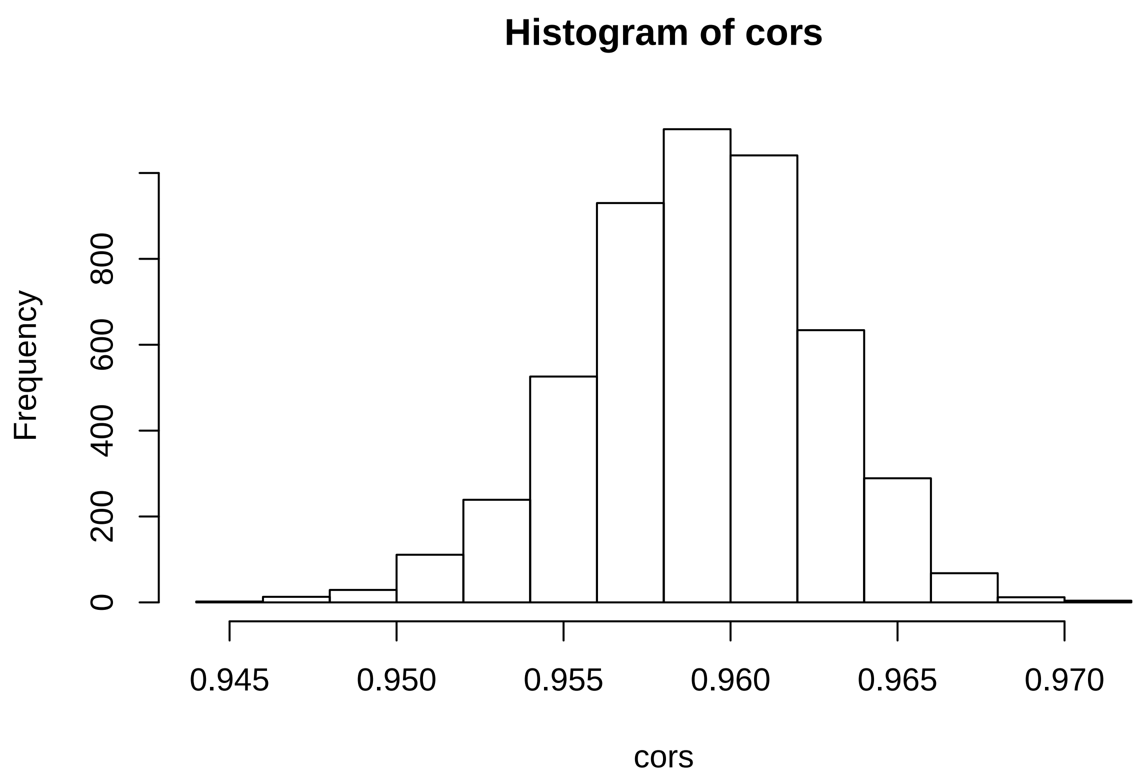
```r
n <- nrow(data)
test_rows <- c()
slopes <- c()
intercepts <- c()
cors <- c()
for(i in 1:5000)
  {
  test_rows <- sample(n, n, replace = TRUE)
  x <- c()
  y <- c()
  for(j in 1:n)
  {
    x[j] <- data$A[test_rows[j]]
    y[j] <- data$P[test_rows[j]]
  }
  newdata <- data.frame(x, y)
  m <- lm(y ~ x)
  slopes[i] <- m$coefficients[2]
  intercepts[i] <- m$coefficients[1]
  cors[i] <- cor(x,y)
}
hist(slopes)
```



```r
hist(intercepts)
```



```r
hist(cors)
```



Because the histogram is similar to a normal distribution, we can apply the technique of pivotal bootstrap.

Slope:

```r
n <- 5000
m <- mean(slopes)
s <- sd(slopes)
pivotal.conf.int <- c(m-1.96*s/sqrt(n), m+1.96*s/sqrt(n))
print(pivotal.conf.int)
```

```
## [1] 1.513177 1.514404
```

Intercept:

```r
n <- 5000
m <- mean(intercepts)
s <- sd(intercepts)
pivotal.conf.int <- c(m-1.96*s/sqrt(n), m+1.96*s/sqrt(n))
print(pivotal.conf.int)
```

```
## [1] 0.9866943 0.9990844
```

Cor:

```r
n <- 5000
m <- mean(cors)
s <- sd(cors)
pivotal.conf.int <- c(m-1.96*s/sqrt(n), m+1.96*s/sqrt(n))
print(pivotal.conf.int)
```

```
## [1] 0.9589660 0.9591611
```

As we see, bootstrap confirms the results of the linear regression

Problem 3.5

```r
index <- which(data$P > 0)
print(mean((abs((data$P[index] - slope*data$A[index] - intercept)/data$P[index]))*100))
```

```
## [1] 28.95489
```

Although determinacy coefficient value is very high (about 92%), the average relative error is also high (about 29%). Linear regression minimizes the average squared difference. However, the average relative error can be made lower by using a nature inspired optimization approach.

Problem 3.6

```r
data = read.csv("data.csv", sep = ",")
yy <- data$P
xx <- data$A
```

We used library("GA")

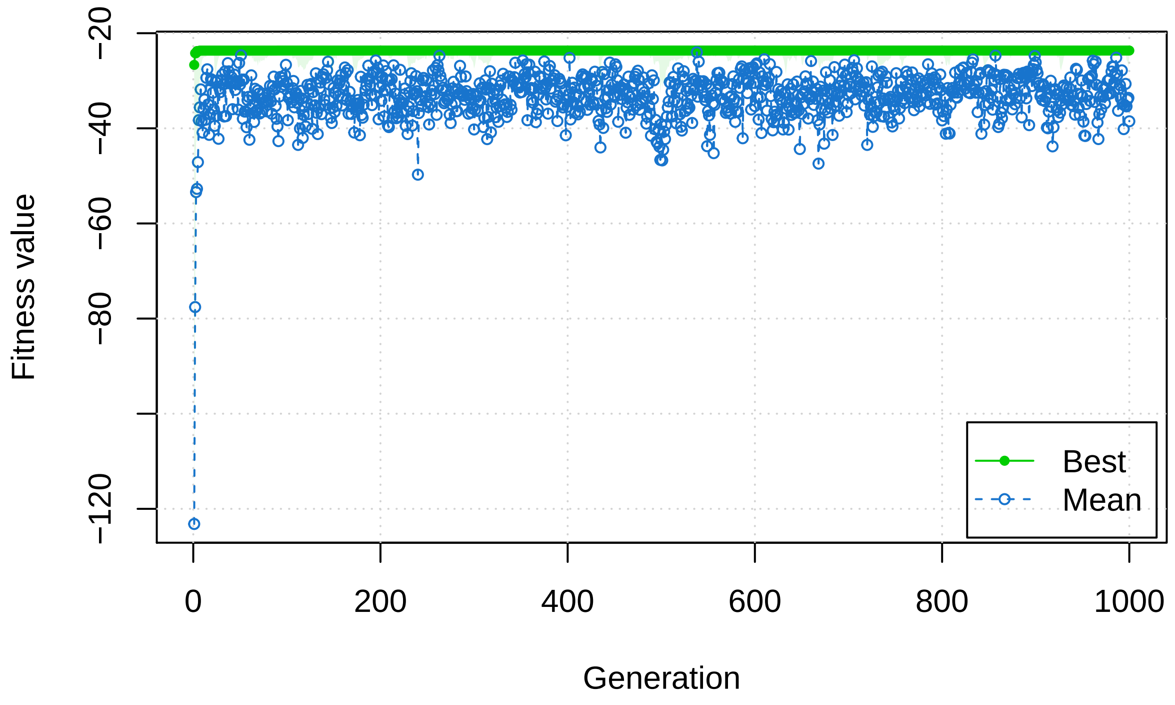
```r
error_function <- function(x,y) (mean(abs((yy[index] - x*xx[index] - y)/yy[index])*100))
GA <- ga(type = "real-valued",
fitness = function(x) -error_function(x[1], x[2]),
min = c(0, -10), max = c(5, 2), popSize = 50,
maxiter = 1000)
```

Results of genetic algorithm:

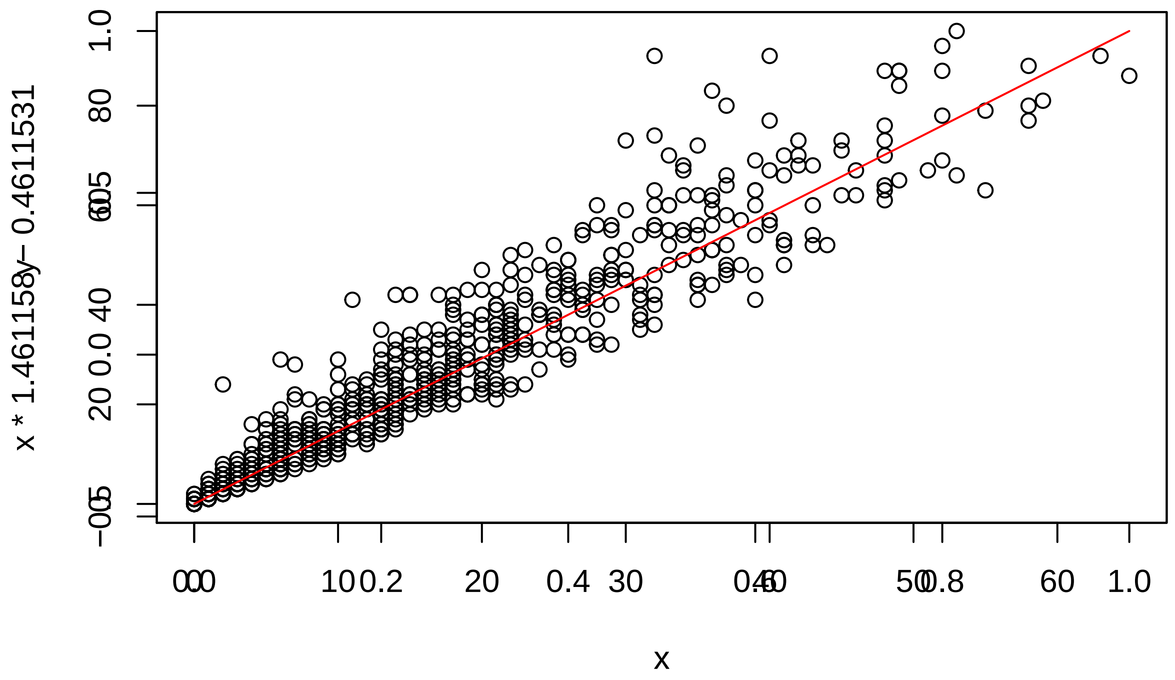
```r
summary(GA)
```

```
## +-----------------------------------+
## |         Genetic Algorithm         |
## +-----------------------------------+
##
## GA settings:
## Type                  =  real-valued
## Population size       =  50
## Number of generations =  1000
## Elitism               =  2
## Crossover probability =  0.8
## Mutation probability  =  0.1
## Search domain
##     x1  x2
## Min  0 -10
## Max  5   2
##
## GA results:
## Iterations             = 1000
## Fitness function value = -23.65106
## Solution               =
##             x1         x2
## [1,] 1.461321 -0.4612051
```

```r
plot(GA)
```



```
y <- data$P
x <- data$A
plot(x,y)
par(new = TRUE)
curve(x*1.461158-0.4611531, col="red")
```



Average relative error:

```r
index <- which(data$P > 0)
print(mean((abs((data$P[index] - 1.461059*data$A[index] +0.4610588)/data$P[index]))*100))
```

```
## [1] 23.65106
```

Use of Nature-Inspired optimization algorithm allows us to receive better value of average relative error, which is expected, given that regression optimizes different measure of accuracy. The actual choice of optimization is dependent on accuracy metric viable for the model. In analyzing sport data the squared distance does not seem like a very accurate metric, so optimizing average relative square may be better

Problem 4.1

Little reminder of used here features:

CF - Corsi for percentage of total. It's very important feature in hockey analytics. TOI.Gm - Time on ice per game. Salary - the player's salary ($ a year).

```
#developing TOI.Gm feature(time on ice)
breaks <- c(4,10,15,20,25,30)
data$TOI.Gm_parts <- sapply(data$TOI.Gm, function(x) which.min(x > breaks))
```

```
#developing Salary feature
breaks <- c(0,1,3,5,14)
data$Salary_parts <- sapply(data$Salary, function(x) which.min(x > breaks))
```

```
#developing CF feature
breaks <- c(20,40,50,60,73)
data$CF_parts <- sapply(data$CF, function(x) which.min(x > breaks))
```

Problem 4.2

```r
#contingency table (conditional frequency table) for Salary and TOI.Gm
cont1.table  <- table( data$Salary_parts, data$TOI.Gm_parts)
norm.cont1.table  <-  100*cont1.table/nrow(data)
```

Conditional frequency table for Salary and TOI.Gm:

```r
print(norm.cont1.table)
```

```
##
##                huge_time large_time   med_time small_time vsmall_time
##   high_salary 0.8373206  2.9904306  6.1004785  0.2392344   0.0000000
##   low_salary  0.1196172  2.3923445  8.7320574  8.4928230   1.5550239
##   med_salary  0.3588517  4.9043062  9.8086124  2.6315789   0.0000000
##   vlow_salary 0.0000000  2.0334928 14.3540670 23.8038278  10.6459330
```

```r
#Quetelet coefficients table
row1.sums  <- rowSums(cont1.table)
col1.sums  <- colSums(cont1.table)
norm.row1.sums  <- row1.sums/nrow(data)
norm.col1.sums  <- col1.sums/nrow(data)
q1.table <-  norm.cont1.table/(norm.row1.sums%*% t(norm.col1.sums)) - 1
```

Quetelet coefficients table for Salary and TOI.Gm:

```r
print(q1.table)
```

```
##
##                huge_time large_time   med_time small_time vsmall_time
##   high_salary 624.882353 237.720731 152.865031   5.690676   -1.000000
##   low_salary   41.696629  90.196684 104.169918 112.421998   58.859000
##   med_salary  153.054054 223.849121 141.082573  41.268799   -1.000000
##   vlow_salary  -1.000000  31.466019  71.407073 132.144458  170.635525
```

```r
#contingency table (conditional frequency table) for Salary and CF
cont2.table  <- table(data$Salary_parts, data$CF_parts)
norm.cont2.table  <-  100*cont2.table/nrow(data)
```

Conditional frequency table for Salary and CF:

```r
print(norm.cont2.table)
```

```
##
##               high_corsi huge_corsi  med_corsi small_corsi vsmall_corsi
##   high_salary  6.3397129  1.9138756  1.9138756   0.0000000    0.0000000
##   low_salary   6.5789474  0.9569378 11.1244019   2.6315789    0.0000000
##   med_salary   8.8516746  0.9569378  7.0574163   0.8373206    0.0000000
##   vlow_salary 19.6172249  2.2727273 21.7703349   7.0574163    0.1196172
```

```
#Quetelet coefficients table
row2.sums  <- rowSums(cont2.table)
col2.sums  <- colSums(cont2.table)
norm.row2.sums  <- row2.sums/nrow(data)
norm.col2.sums  <- col2.sums/nrow(data)
q2.table <-  norm.cont2.table/(norm.row2.sums%*% t(norm.col2.sums)) - 1
```

Quetelet coefficients table for Salary and CF:

```
print(q2.table)
```

```
##
##              high_corsi huge_corsi med_corsi small_corsi vsmall_corsi
##   high_salary  149.65624  307.55825  43.96134    -1.00000     -1.00000
##   low_salary    73.65740   72.67262 123.79615   116.41573     -1.00000
##   med_salary   119.80925   87.60625  94.22008    43.93243     -1.00000
##   vlow_salary   92.23631   72.28258 101.28706   130.88235    195.70588
```

From the conditional frequency table for Salary and TOI/Gm we can conclude that the number of the lowest paid hockey players with small on-the-ice time prevails in the league. It makes sense. Also, from this table we see that there are no players with the lowest salary and huge on-the-ice time and there are no players with high salary and the lowest on-the-ice time. It is also clear. From the Quetelet coefficients table for Salary and TOI/Gm the main think that we can derive is that the presence of the highest salary increases the probability of the highest on-the-ice time by 625%. It is huge number. However, it is clear that the top-players get the most of the playing time. From the conditional frequency table for Salary and CF we can conclude that among the players with the lowest corsi number there are only the lowest paid players. Also, we have that among the highest paid hockey players there are no players with small and very small corsi number. Remember only one thing about corsi number: the more corsi number is the better player plays. In addition, from this table we see that the most players have medium corsi number. The main idea form the Quetelet coefficients table for Salary and CF: the fact that the hockey player has a high salary increases the corsi number by 308%.

Problem 4.3

Summary Quetelet index is the inner product of two tables,that of co-occurrence and Quetelet:

```
norm.cont1.table  <-  cont1.table/nrow(data)
norm.cont2.table  <-  cont2.table/nrow(data)
Q1_table <- norm.cont1.table*q1.table
```

```
print(Q1_table)
```

```
##
##                  huge_time  large_time    med_time  small_time vsmall_time
##   high_salary  5.23226851  7.10887354  9.32549828  0.01361406  0.00000000
##   low_salary   0.04987635  2.15781540  9.09617705  9.54780127  0.91527153
##   med_salary   0.54923704 10.97824636 13.83824284  1.08602104  0.00000000
##   vlow_salary  0.00000000  0.63985925 10.24981913 31.45543911 18.16574367
```

Pirson index for Salary and TOI/Gm table:

```
r.table1 <-  (-norm.row1.sums%*% t(norm.col1.sums)+
                norm.cont1.table)/sqrt(norm.row1.sums%*% t(norm.col1.sums))
print(r.table1)
```

```
##        huge_time  large_time    med_time  small_time vsmall_time
## [1,]  0.19234820  0.15526128  0.10725533 -0.17644197 -0.11137899
## [2,] -0.03033055 -0.01425833  0.01489691  0.03672777 -0.06469817
## [3,]  0.02608851  0.18438631  0.11056958 -0.14404853 -0.14696864
## [4,] -0.08178705 -0.16901652 -0.12285554  0.14014344  0.17840898
```

Pirson index for Salary and CF table:

```
r.table2 <-  (-norm.row2.sums%*% t(norm.col2.sums)+
                norm.cont2.table )/sqrt(norm.row2.sums%*% t(norm.col2.sums))
print(r.table2)
```

```
##       high_corsi  huge_corsi   med_corsi small_corsi vsmall_corsi
## [1,]  0.10391403  0.16425388 -0.11355468 -0.10345334  -0.01102816
## [2,] -0.07523032 -0.03000520  0.07403242  0.02607274  -0.01595893
## [3,]  0.05632731 -0.01184067 -0.01301306 -0.07517299  -0.01455206
## [4,] -0.03102485 -0.04705088  0.01055114  0.07375303   0.02384740
```

Summary Quetelet index for Salary and TOI/Gm table:

```
Q1 <-  sum(norm.cont1.table^2/(norm.row1.sums%*%t(norm.col1.sums)))-1
print(Q1)
```

```
## [1] 0.314098
```

```
Q2 <-  sum(norm.cont2.table^2/(norm.row2.sums%*%t(norm.col2.sums)))-1
```

Summary Quetelet index for Salary and CF table:

```
print(Q2)
```

```
## [1] 0.09311244
```

Q1 = 0.3141 means that if we know category one of the feature (TOI/Gm or Salary) the probability of another feature's category increases by 31.4% on the average. Similarly, Q2 = 0.0931 means that if we know category one of the feature (Corsi or Salary) the probability of another feature's category increases by 9.3% on the average. Also, in practice we can use Pearson???s chi-squared a measure of correlation. In other words, the average degree of correlation between categories of the Salary and categories of TOI/Gm is more than the average degree of correlation between categories of the Salary and categories of Corsi.

Problem 4.4

Statistics style (check simple hypothesis)

```
n <- nrow(data)
#Every new iteration we randomly select sample with size = i:
for (i in 10:n)
{
  test_rows <- c()
  test_rows <- sample(n, i, replace = FALSE)
  x <- y <- c()
  for(j in 1:i)
  {
    x[j] <- data$Salary_parts[test_rows[j]]
    y[j] <- data$TOI.Gm_parts[test_rows[j]]
  }
  newdata <- data.frame(x, y)
    cont1.table  <- table(x, y)
    nrow_table1 <- nrow(cont1.table)
    ncol_table1 <- ncol(cont1.table)
#Count Q for new sample:
    norm.cont1.table <-  100*cont1.table/nrow(newdata)
  row1.sums  <- rowSums(cont1.table)
  col1.sums  <- colSums(cont1.table)
  norm.row1.sums  <- row1.sums/nrow(newdata)
  norm.col1.sums  <- col1.sums/nrow(newdata)
  q1.table <-  norm.cont1.table/(norm.row1.sums%*% t(norm.col1.sums)) - 1
  norm.cont1.table  <-  cont1.table/nrow(newdata)
  Q11 <-  sum(norm.cont1.table^2/(norm.row1.sums%*%t(norm.col1.sums)))-1
#Finding chi square values:
  chi=qchisq(.95, df=(nrow_table1-1)*(ncol_table1-1))
#Check hypothesis:
  if(chi < Q11*nrow(newdata))
  {
    n_min <- i
    break
  }
}
```

Numbers of observations suffice to see the features as associated at 95% confidence level. First table

```
print(n_min)
```

```
## [1] 17
```

Numbers of observations suffice to see the features as associated at 99% confidence level

```
print(n_min)
```

```
## [1] 17
```

Numbers of observations suffice to see the features as associated at 99% confidence level. Second table.

```
print(n_min)
```

```
## [1] 16
```

Numbers of observations suffice to see the features as associated at 95% confidence level. Second table.

```
print(n_min)
```

```
## [1] 23
```

Mirking style

Search of number of observations that suffices to see the features 'Salary' and 'TOI.Gm' as associated at 95% confidence level:

```
chi1 <- qchisq(.95, df=(nrow(norm.cont1.table)-1)*(ncol(norm.cont1.table)-1))
new.n1 <- chi1 / Q1
print(new.n1)
```

```
## [1] 53.86528
```

Search of number of observations that suffices to see the features 'Salary' and 'TOI.Gm' as associated at 95% confidence level:

```
chi2 <- qchisq(.99, df=(nrow(norm.cont1.table)-1)*(ncol(norm.cont1.table)-1))
new.n2 <- chi2 / Q1
print(new.n2)
```

```
## [1] 68.97844
```

Search of number of observations that suffices to see the features 'Salary' and 'CF' as associated at 95% confidence level:

```
chi3 <- qchisq(.95, df=(nrow(norm.cont1.table)-1)*(ncol(norm.cont1.table)-1))
new.n3 <- chi3 / Q2
print(new.n3)
```

```
## [1] 181.7048
```

Search of number of observations that suffices to see the features 'Salary' and 'CF' as associated at 95% confidence level:

```
chi4 <- qchisq(.99, df=(nrow(norm.cont1.table)-1)*(ncol(norm.cont1.table)-1))
new.n4 <- chi4 / Q2
print(new.n4)
```

```
## [1] 232.6864
```

Problem 5.1

Linear regression for normalized data:

Another important problem of multidimensional linear regression is a sign of diversity. If the scales measuring attributes significantly (by several orders of magnitude) are different, then there is DANGER that will take into account only the "large-scale" signs. To avoid this, do the standardization

```
standardize <- function(x) {
    y <- (x - mean(x))/(sd(x))
    y
}
```

Let's see if we have correlation between data that we want to use for linear regression:

```
some.data <- data[,4:7]
cor(some.data)
```

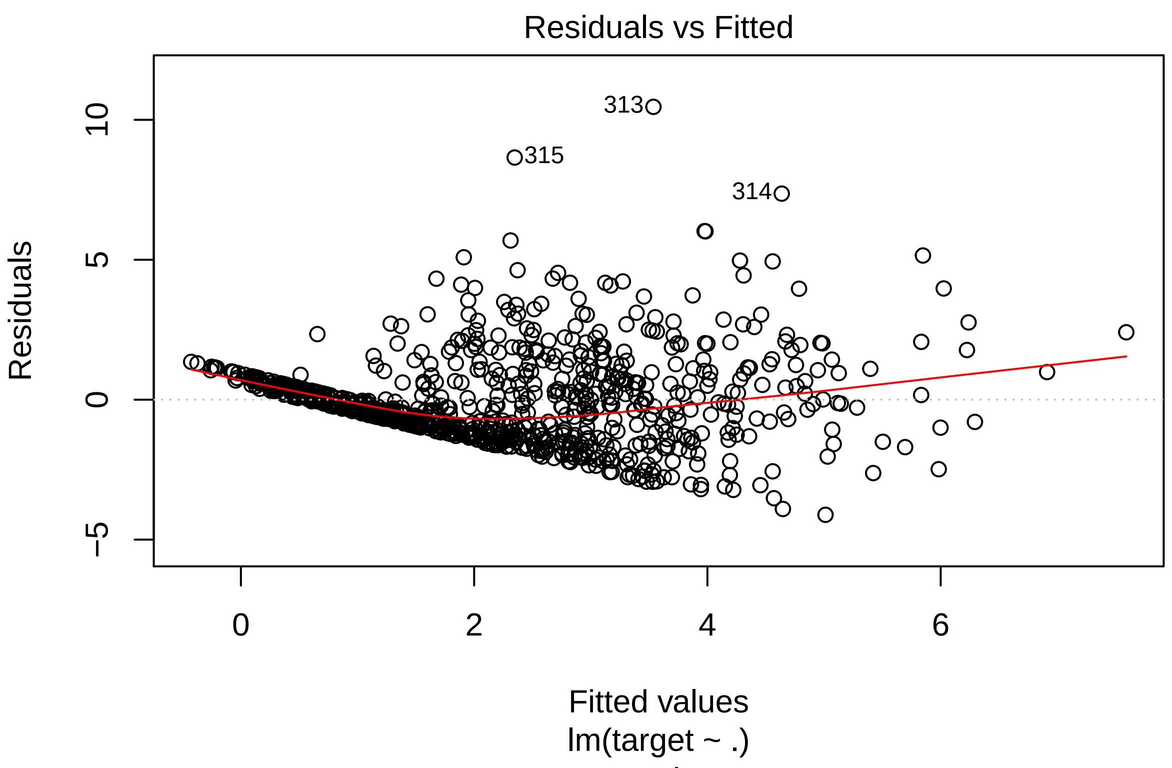
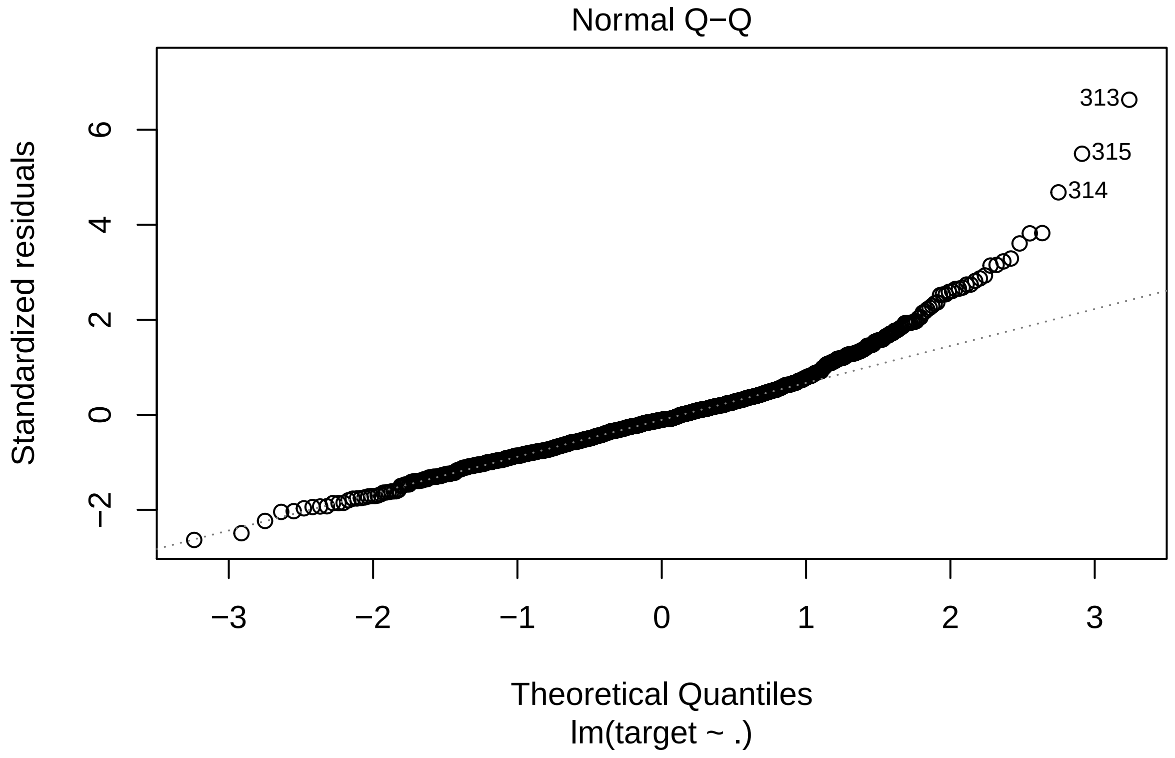
```
##               Gm        Age    Salary          G
## Gm     1.0000000 0.24332508 0.4747229 0.63941427
## Age    0.2433251 1.00000000 0.4273528 0.05525991
## Salary 0.4747229 0.42735276 1.0000000 0.51472597
## G      0.6394143 0.05525991 0.5147260 1.00000000
```

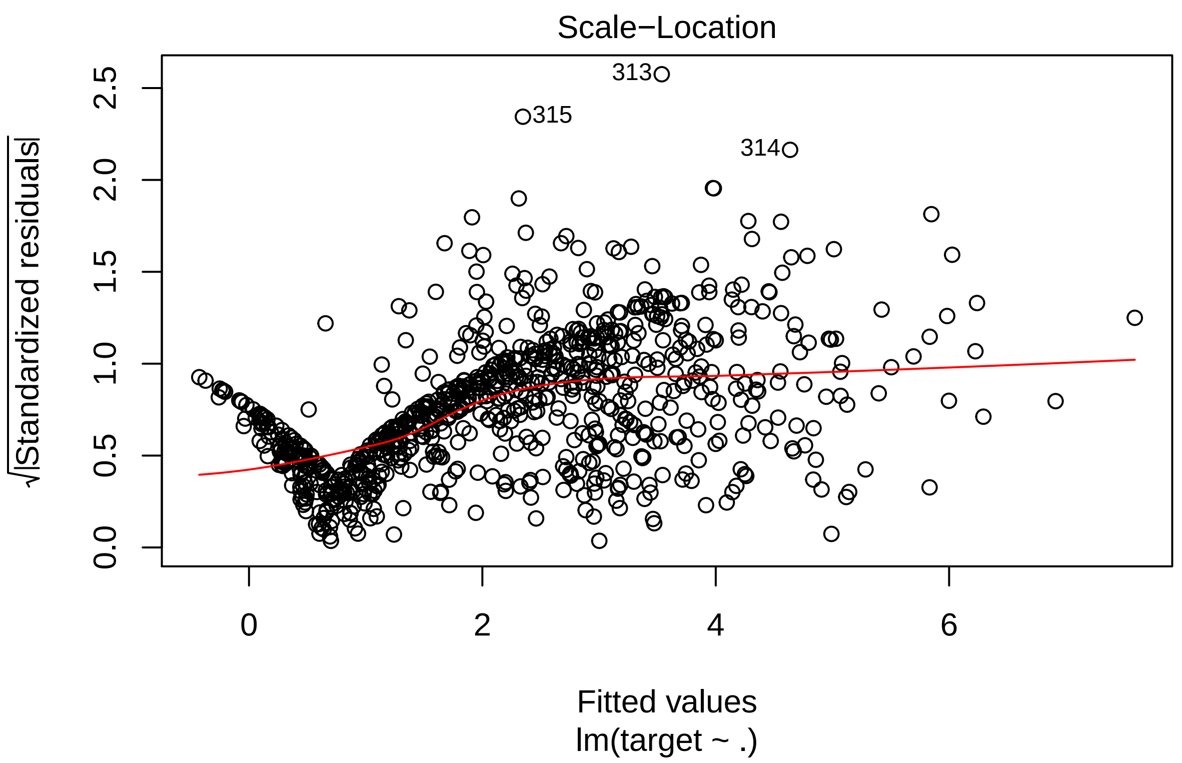
Results of modeling:

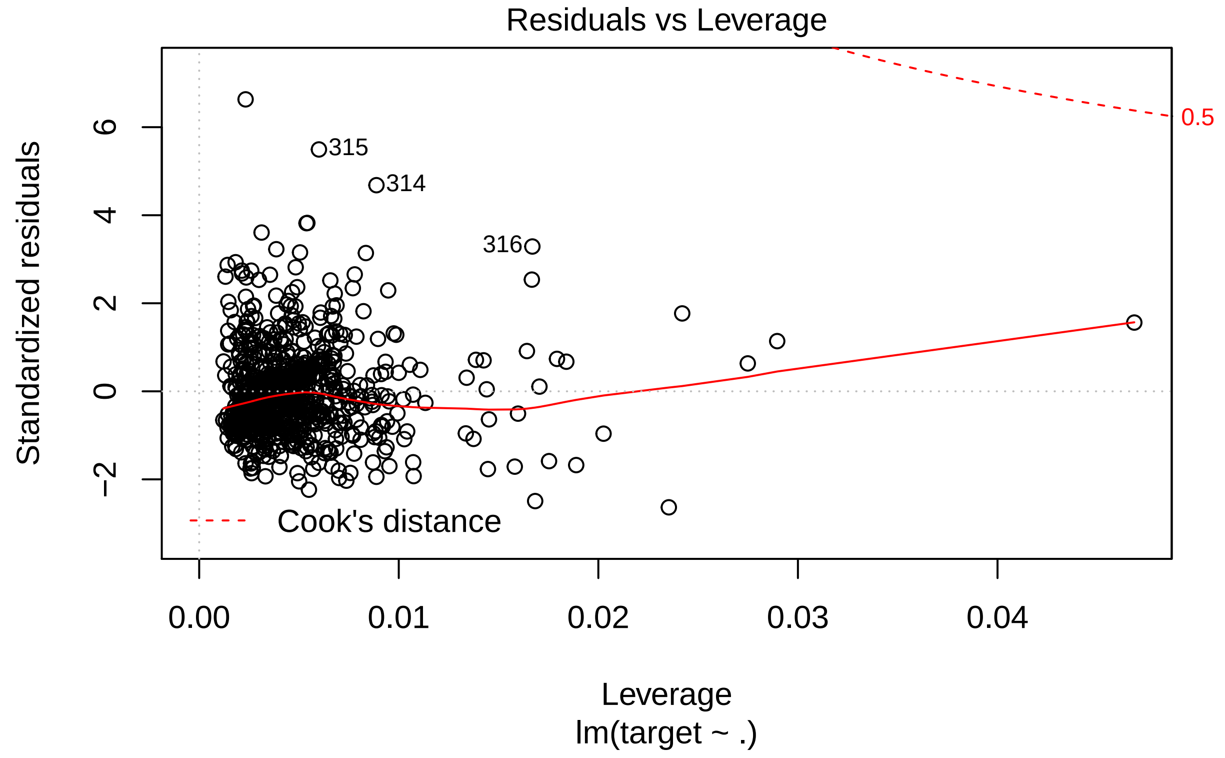
```
target  <- some.data$Salary
some.data$Salary <- NULL
raw.data <- some.data
normal.data <- as.data.frame(lapply(some.data, standardize))
norm.m  <-  lm(target ~., normal.data)
#summary(norm.m)
print(norm.m$coefficients)
```

```
## (Intercept)          Gm         Age           G
##   2.2171483   0.2381681   0.7871758   0.8807499
```

```
plot(norm.m)
```


Residuals vs Fitted
Residuals
Fitted values
lm(target ~ .)
313
315
314
10
5
0
−5
0
2
4
6

Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(target ~ .)
313
315
314
6
4
2
0
−2
−3
−2
−1
0
1
2
3





Linear regression for raw data:

```
raw.m  <-  lm(target ~., raw.data)
print(raw.m$coefficients)
```

```
##   (Intercept)            Gm          Age             G
## -3.595560640  0.007706404  0.175213774  0.095669718
```

We can conclude that the number of Goals has the most impact on the player's Salary. This means that the Salary would change more on the average if the number of Goals is increased, while the others features do not change, rather than if the other features are changed. Also, we see that the data standartization has led us to increased coefficients. It is so because the raw data values is bigger than the standardized data values.

Problem 5.2

Determinacy coefficient

```
print(summary(norm.m)$r.square)
```

```
## [1] 0.4316618
```

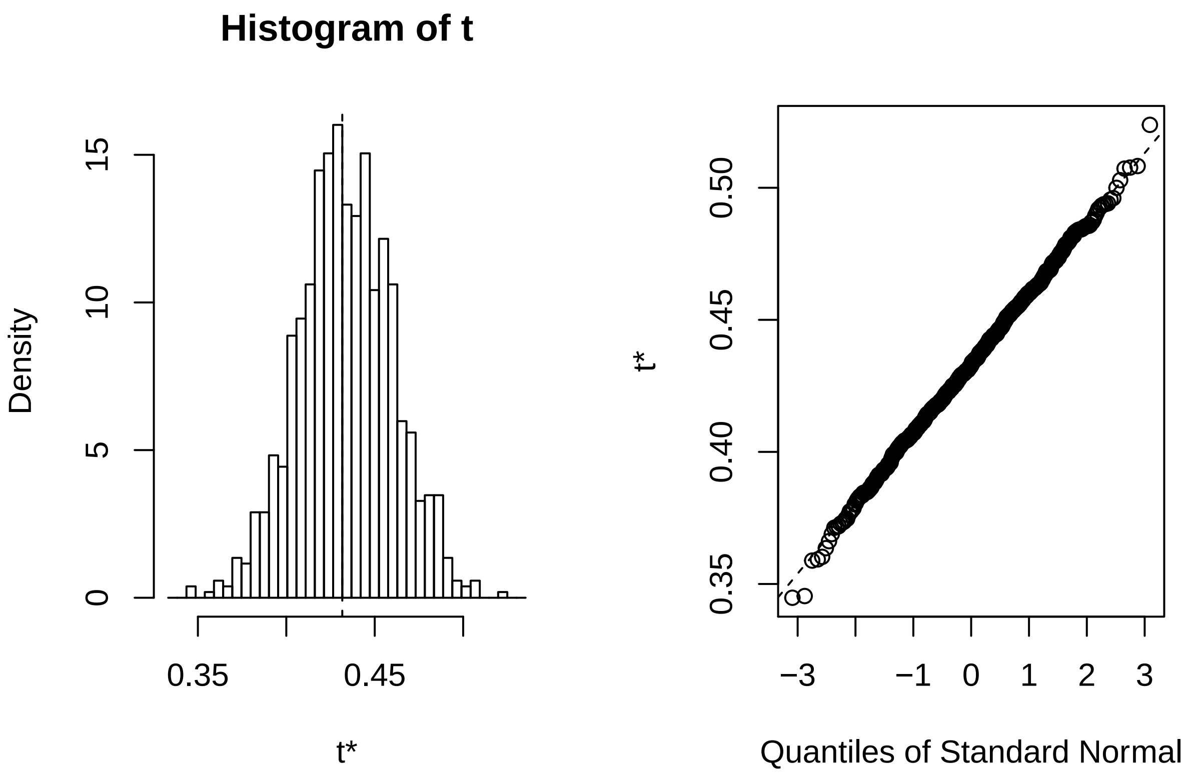
Let's count 95% confidence interval of determinacy coefficient using bootstrap

```
boot.data <-data[,4:7]
rsq <- function(data, indices) {
         d <- data[indices,]
         target <- d$Salary
         d$Salary <- NULL
         fit <- lm(target~., data=d)
         return(summary(fit)$r.square)
}

library(boot)
set.seed(1234)
results <- boot(data=boot.data, statistic=rsq,
                R=1000)
print(results)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = boot.data, statistic = rsq, R = 1000)
##
##
## Bootstrap Statistics :
##      original    bias    std. error
## t1* 0.4316618 0.0018613  0.02653647
```

```
plot(results)
```



We see that the Determinacy coefficient from the summary of the linear regression from the previous task and computed Determinacy coefficient are the same

Confidence interval

```r
boot.ci(results, conf=0.95, type=c("bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, conf = 0.95, type = c("bca"))
##
## Intervals :
## Level       BCa
## 95%   ( 0.3783,  0.4841 )
## Calculations and Intervals on Original Scale
```

It says us about that Gm, Age and G explain from 38% to 48% of the variance of the Salary with 95% probability

Problem 5.3

Devide our dataset in two groups: forwards and defencemans. Take Salary P60, PenD, CF%, PDO, PSh%, ZSO%Rel, TOI/Gm as input features. To find an appropriate Linear discriminant rule put 1 for all forwards and -1 for all defencemans and apply the least-squares criterion of linear regression. After additional rounding we get the following contingency table.

```r
target <- c()
index <- which(data$pos == "LD"|data$pos == "DR"|data$pos == "D"|data$pos == "RD")
target[index] <- 2
target[is.na(target)] <- 1

library(MASS)
std.data <- data[,12:18]
hockey.lda <- lda(target ~ ., data=std.data)
hockey.lda
```

```
## Call:
## lda(target ~ ., data = std.data)
##
## Prior probabilities of groups:
##         1         2
## 0.6483254 0.3516746
##
## Group means:
##         P60      PenD      CF.      PDO     PSh.    ZSO.Rel   TOI.Gm
## 1 1.4884317  1.191882 49.89670 99.26934 8.906734  2.2403321 13.60673
## 2 0.7866667 -4.595238 48.48218 98.85680 4.082619 -0.4764286 18.30323
##
## Coefficients of linear discriminants:
##                 LD1
## P60     -1.36554168
## PenD    -0.04597232
## CF.     -0.01335651
## PDO      0.05965449
## PSh.    -0.02324725
## ZSO.Rel  0.01245540
## TOI.Gm   0.28433435
```

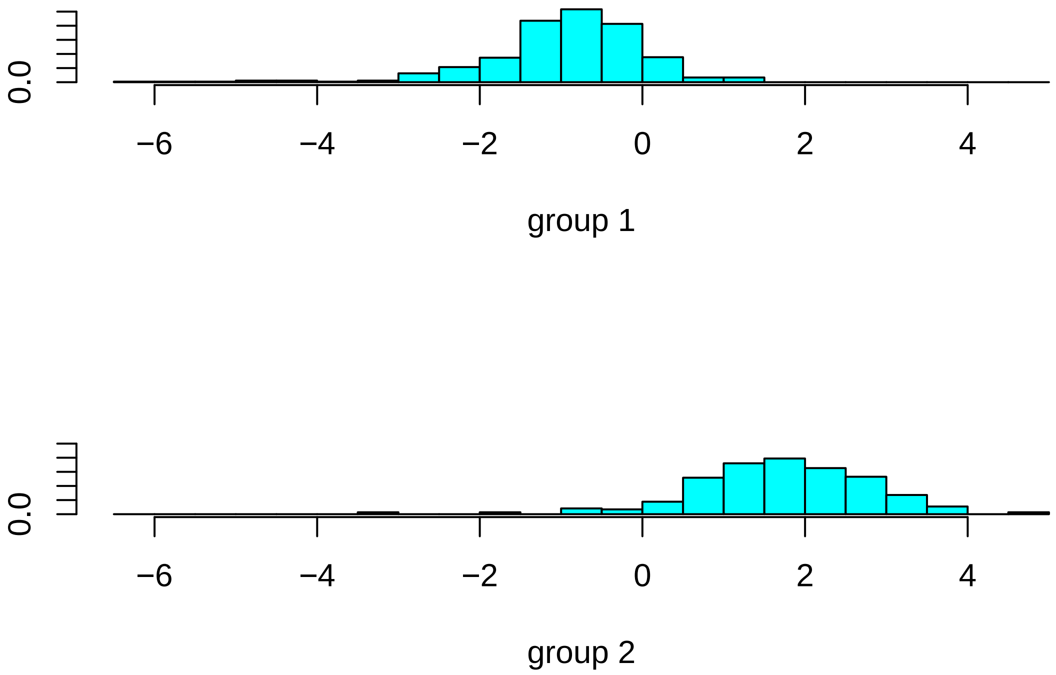
```r
hockey.lda.values <- predict(hockey.lda)
```

A nice way of displaying the results of a linear discriminant analysis (LDA) is to make a stacked histogram of the values of the discriminant function for the samples from different groups (different targets in our example).

```r
summary(hockey.lda.values)
```

```
##           Length Class  Mode
## class      836   factor numeric
## posterior 1672   -none- numeric
## x          836   -none- numeric
```

```
ldahist(data = hockey.lda.values$x[,1], g=target)
```



To cross-validate the accuracy of the Linear discriminant rule use the following code:

```
library("crossval")
predfun.lda = function(train.x, train.y, test.x, test.y, negative)
{
  require("MASS")
  lda.fit = lda(train.x, grouping=train.y)
  ynew = predict(lda.fit, test.x)$class
  out = confusionMatrix(test.y, ynew, negative=negative)
  return( out )
}
```

2-fold validation scheme for the Linear discriminant rule gives us the following:

```
cv.out = crossval(predfun.lda, std.data, target, K=2, B=1, negative="1",verbose=FALSE)
diagnosticErrors(cv.out$stat)
```

```
##       acc      sens      spec       ppv       npv       lor
## 0.9354067 0.8741497 0.9686347 0.9379562 0.9341637 5.3683431
```

10-fold validation scheme for the Linear discriminant rule gives us the following:

```
cv.out = crossval(predfun.lda, std.data, target, K=10, B=1, negative="1",verbose=FALSE)
diagnosticErrors(cv.out$stat)
```

```
##       acc      sens      spec       ppv       npv       lor
## 0.9413876 0.8775510 0.9760148 0.9520295 0.9362832 5.6754797
```

Discriminant analysis by Fishers rule.

```
library(penalizedLDA)
set.seed(1)
n <- nrow(std.data)
p <- ncol(std.data)
x <- std.data
y <- target
out <- PenalizedLDA(x,y,lambda=.14,K=1)
out$discrim
```

```
##             [,1]
## [1,] -0.47432721
## [2,] -0.46073507
## [3,] -0.03063876
## [4,]  0.00000000
## [5,] -0.34089848
## [6,] -0.01871167
## [7,]  0.66726027
```

```
pred.out <- predict(out,xte=std.data)
table(pred.out$ypred,target)
```

```
##    target
##       1   2
##   1 527  61
##   2  15 233
```

```
accuracy <- (table(pred.out$ypred,target)[1]
             +table(pred.out$ypred,target)[4])/sum(table(pred.out$ypred,target))
accuracy
```

```
## [1] 0.9090909
```

To cross-validate the accuracy of the Fisher discriminant rule use the following code:

```
library("crossval")
predfun.PenalizedLDA = function(train.x, train.y, test.x, test.y, negative)
{
require("MASS")
PenalizedLDA.fit = PenalizedLDA(train.x,train.y,lambda=.14, K=1)
ynew = predict(PenalizedLDA.fit, test.x)$ypred
out = confusionMatrix(test.y, ynew, negative=negative)
return( out )
}
```

2-fold validation scheme for the Fisher discriminant rule gives us the following:

```
cv.out = crossval(predfun.PenalizedLDA, std.data, target, K=2, B=1, negative="1",verbose=FALSE)
diagnosticErrors(cv.out$stat)[1]
```

```
##       acc
## 0.9078947
```

10-fold validation scheme for the Fisher discriminant rule gives us the following:

```
cv.out = crossval(predfun.PenalizedLDA, std.data, target, K=10, B=1, negative="1",verbose=FALSE)
diagnosticErrors(cv.out$stat)[1]
```

```
##       acc
## 0.9055024
```

The Linear discriminant rule and the Fisher discriminant rule use different numerical codes for "yes" and "no" classes. That's why they have slightly different accuracies. Accuracy of LDA is slightly better compared to FLDA. And there is much more advanced library for LDA.

Problem 6.1

Consider such a factor as player's value. There is a huge number of different statistics in hockey. However, many ice hockey experts consider three most important features that characterize how useful (resp. how valuable) player is. These are P60, PenD and CF%. The first item shows how many points player get every game. The more P60 is, the more valuable player is. The second item is a differential between number of penalties drawn by the player and number of penalties taken by the player. The more PenD is, the more player's team has Powerplays, the more useful player is. The last one reflects the ration between shots made by player?s team and shots made by opponents while player is on the ice. It is clear that the more CF% value is, the more valuable player is.

```
data = read.csv("data.csv", sep = ",")
data <- data[order(data$ZSO,decreasing=FALSE),]
data1 <- data[,12:14] #derive three features
summary(data1)
```

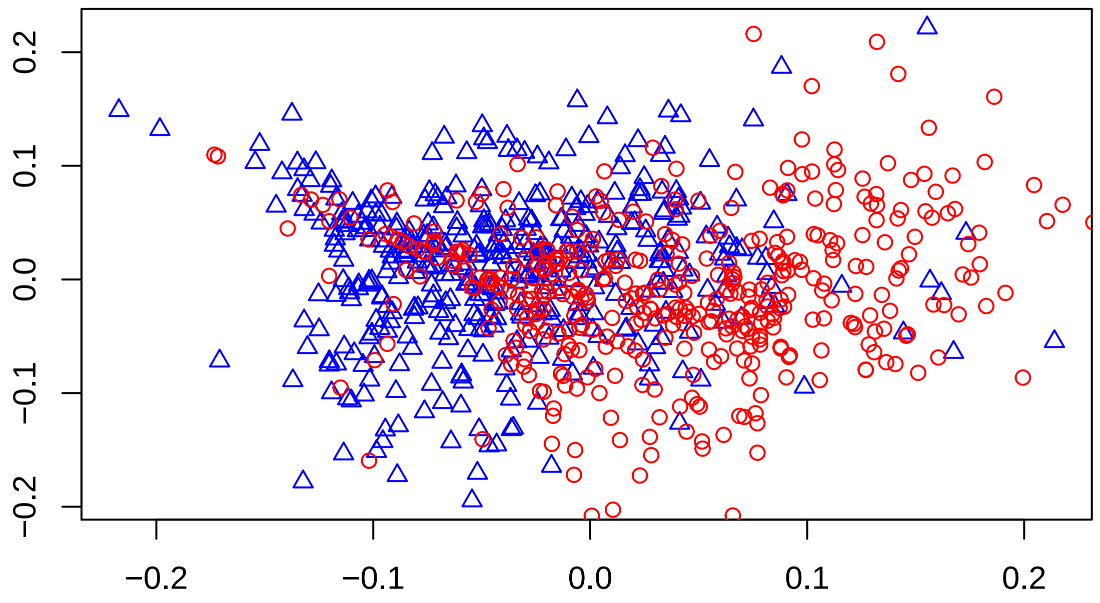
```
##       P60             PenD               CF.
##  Min.   :0.000   Min.   :-25.0000   Min.   :20.00
##  1st Qu.:0.630   1st Qu.: -4.0000   1st Qu.:44.44
##  Median :1.140   Median :  0.0000   Median :49.72
##  Mean   :1.242   Mean   : -0.8433   Mean   :49.40
##  3rd Qu.:1.780   3rd Qu.:  2.0000   3rd Qu.:54.81
##  Max.   :5.530   Max.   : 27.0000   Max.   :72.22
```

Problem 6.2

For this task we need to determine the different group of players. Consider ZSO feature. It is higher if a player starts his game change in the oponent's zone more often than in the home zone. P60, PenD, CF% are all related to the ZSO-value, because the more ZSO is the more chances a player has to get the points, gain the Powerplay and make his CF higher. So for this reason we have sorted our data by ZSO increase in the Task1 and we will take into account players with low/high ZSO-value when visualizing our data.

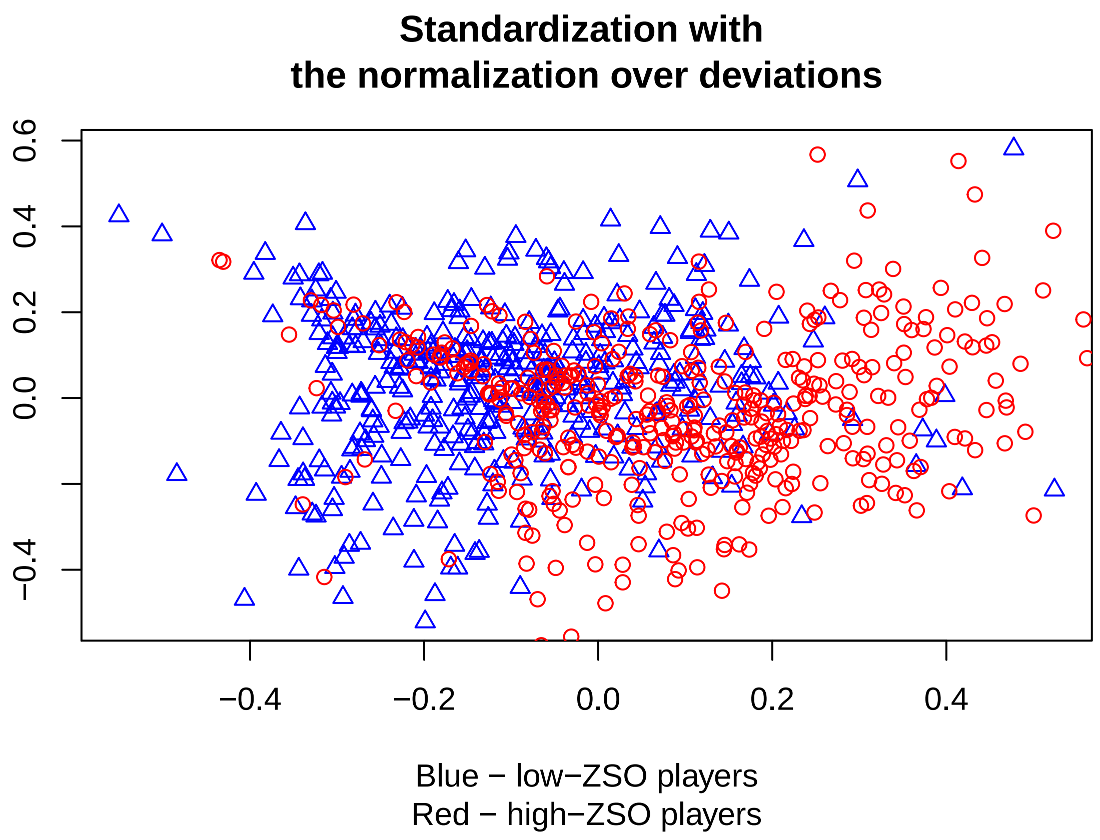
```r
#standardization with the normalization over deviations (Z-scoring)
standardizeByDeviation <- function(x) {
    y <- (x - mean(x))/(sd(x))
    y
}
#standardization with the normalization over ranges
standardizeByRange <- function(x) {
    y <- (x - mean(x))/(max(x)-min(x))
    y
}
data2 <- data1
stdByDeviationData2 <- as.data.frame(lapply(data2, standardizeByDeviation))
stdByRangeData2 <- as.data.frame(lapply(data2, standardizeByRange))
#Compute two first singular triplets:
resultOfSVD1 <- svd(stdByRangeData2, nu=2, nv=2, LINPACK = FALSE)
Z1_1 <- resultOfSVD1$u[,1]*sqrt(resultOfSVD1$d[1])
Z2_1 <- resultOfSVD1$u[,2]*sqrt(resultOfSVD1$d[2])
#Determine the proportion of the variance taken into account:
p1 <- (resultOfSVD1$d[1]^2+resultOfSVD1$d[2]^2)/sum(sum(stdByRangeData2*stdByRangeData2))
plot(Z1_1[1:418],Z2_1[1:418],col="blue",pch=2,xlab=NA,ylab=NA,main="Standardization with
the normalization over ranges",sub="Blue - low-ZSO players
Red - high-ZSO players") #plot low-ZSO
points(Z1_1[419:836],Z2_1[419:836],col="red",pch=1) #plot high-ZSO
```

Standardization with
the normalization over ranges



Blue − low−ZSO players
Red − high−ZSO players

```
resultOfSVD2 <- svd(stdByDeviationData2, nu=2, nv=2, LINPACK = FALSE)
Z1_2 <- resultOfSVD2$u[,1]*sqrt(resultOfSVD2$d[1])
Z2_2 <- resultOfSVD2$u[,2]*sqrt(resultOfSVD2$d[2])
#Determine the proportion of the variance taken into account:
p2 <- (resultOfSVD2$d[1]^2+resultOfSVD2$d[2]^2)/sum(sum(stdByDeviationData2*stdByDeviationData2))
plot(Z1_2[1:418],Z2_2[1:418],col="blue",pch=2,xlab=NA,ylab=NA,main="Standardization with
the normalization over deviations",sub="Blue - low-ZSO players
Red - high-ZSO players") #plot low-ZSO
points(Z1_2[419:836],Z2_2[419:836],col="red",pch=1) #plot high-ZSO
```



```
summary(stdByRangeData2) #display summary of stdByRangeData2
```

```
##       P60                  PenD                 CF.          
##  Min.   :-0.22453   Min.   :-0.46455   Min.   :-0.562988  
##  1st Qu.:-0.11060   1st Qu.:-0.06071   1st Qu.:-0.095064  
##  Median :-0.01838   Median : 0.01622   Median : 0.006142  
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.000000  
##  3rd Qu.: 0.09735   3rd Qu.: 0.05468   3rd Qu.: 0.103615  
##  Max.   : 0.77547   Max.   : 0.53545   Max.   : 0.437012
```

```
p1 #display the proportion of the variance taken into account by stdByRangeData2
```

```
## [1] 0.8099296
```

```
summary(stdByDeviationData2) #display summary of stdByDeviationData2
```

```
##       P60                 PenD                 CF.          
##  Min.   :-1.5145   Min.   :-3.4999   Min.   :-3.94447  
##  1st Qu.:-0.7460   1st Qu.:-0.4574   1st Qu.:-0.66605  
##  Median :-0.1240   Median : 0.1222   Median : 0.04303  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000  
##  3rd Qu.: 0.6567   3rd Qu.: 0.4119   3rd Qu.: 0.72596  
##  Max.   : 5.2306   Max.   : 4.0340   Max.   : 3.06184
```

```
p2 #display the proportion of the variance taken into account by stdByDeviationData2
```

```
## [1] 0.8193398
```

Two plots are very similar, except only displaying ranges. We can see, maybe not quite clear, two groups of players with high/low ZSO-value. It is hard to say which normalization is better from the plots. However, normalization over deviations should make all features contribute similarly to the data scatter. This is confirmed by the fact that the computed proportion of the variance taken into account when normalizing over deviations (82%) is a little bit more than the computed proportion of the variance taken into account when normalizing over ranges (81%).

Problem 6.3

```
library(base)
#convert base features into the same scale, 0 to 100, to form matrix data1 for using in PCA
data1[1] <- data1[1]*100/max(data1[1])
data1[2] <- data1[2]-min(data1[2])
data1[2] <- data1[2]*100/max(data1[2])
data1[3] <- data1[3]-min(data1[3])
data1[3] <- data1[3]*100/max(data1[3])
summary(data1) #normalized features from 0 to 100
```

```
##       P60              PenD              CF.
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00
##  1st Qu.: 11.39   1st Qu.: 40.38   1st Qu.: 46.79
##  Median : 20.61   Median : 48.08   Median : 56.91
##  Mean   : 22.45   Mean   : 46.46   Mean   : 56.30
##  3rd Qu.: 32.19   3rd Qu.: 51.92   3rd Qu.: 66.66
##  Max.   :100.00   Max.   :100.00   Max.   :100.00
```

```
resultOfSVD <- svd(data1, nu=1, nv=1, LINPACK = FALSE) #apply PCA to the data to find
#the First singular triplet
z <- -resultOfSVD$u #first singular 836D scoring vector
Mu <- resultOfSVD$d[1] #maximum singular value
c <- -resultOfSVD$v #loadings
#we need to rescale z to convert it to 0 ? 100 scale
#we have equation for the hidden factor:
#Z = (c[1]*P60+c[2]*PenD+c[3]*CF)*alpha
#Find alpha that Z=100 at all features being 100
alpha <- 100/(c[1]*100+c[2]*100+c[3]*100)
#The FINAL equation is:
Z <- c[1]*alpha*data1$P60+c[2]*alpha*data1$PenD+c[3]*alpha*data1$CF #Here Z reflects
#how valuable player is from 0 to 100
ds <- sum(sum(data1*data1)) #data scatter
contrib <- 100*(Mu^2)/ds #Contribution of thefirst component to the data scatter

t(c) #display loadings
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.3078509 0.6038147 0.7352793
```

```
contrib #display contribution
```

```
## [1] 95.07798
```

```
cbind(data1, Z)[10:20,] #display 10 players summary table for clarity
```

```
##           P60     PenD      CF.        Z
## 239 15.189873 53.84615 37.16967 39.17520
## 324 18.444846 26.92308 37.55266 30.08389
## 734 16.998192 46.15385 25.81386 31.62321
## 344 22.965642 44.23077 43.33589 39.85632
## 799  0.000000 48.07692 50.44044 40.14546
## 702  0.000000 50.00000 54.99809 42.88528
## 348 14.466546 63.46154 42.79969 45.07879
## 776  0.000000 51.92308 29.52892 32.21961
## 367 13.200723 46.15385 22.44351 29.40868
## 214  7.775769 42.30769 29.26082 30.02812
## 726  9.041591 46.15385 14.20912 24.95500
```

```
#with computed hidden factor (value of the player)
```

From loadings' interpretation we can conclude that CF% plays more important role in player's utility estimation (by about 1.23 more than PenD, by about 2.39 more than P60). It may seem strange from the one hand. Usual people think that the number of points is the most significant value. However, our results confirm the fact that the coaching staff and scouts appreciate player's ability to gain the Powerplays for the team and defensive skills (CF% shows not only how many shots was made by player?s team, but also how few shots was made by opponents while player is on the ice). Also, we got a good contribution: 95.08%

Problem 7.1

Function helping to make right choice about number of K

```
wssplot <- function(data, nc=15, seed=1234){
               wss <- (nrow(data)-1)*sum(apply(data,2,var))
               for (i in 2:nc){
                    set.seed(seed)
                    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
                plot(1:nc, wss, type="b", xlab="Number of Clusters",
                     ylab="Within groups sum of squares")}
```

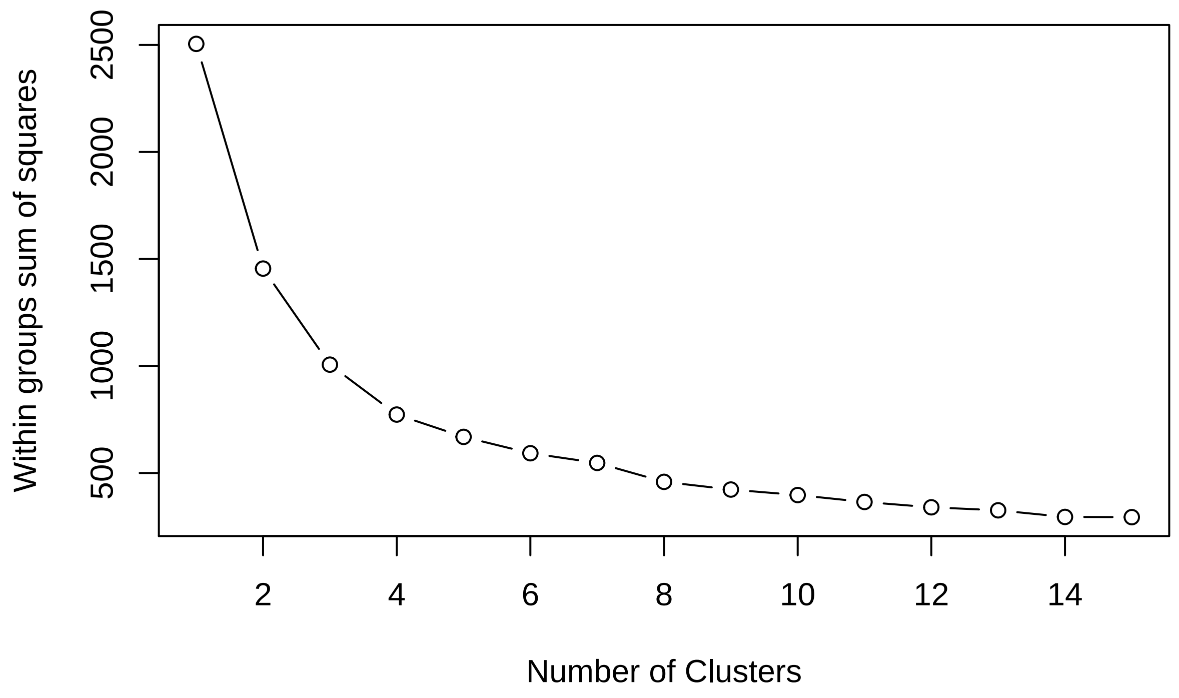
Data that we want to use for clusterization

```
kmeans.data <- data[4:6]
head(kmeans.data)
```

```
##    Gm Age Salary
## 1  74  28   9.50
## 2  77  29   9.25
## 3  83  29   9.00
## 4  93  29   8.75
## 5  81  29   8.75
## 6 108  24   8.00
```

Scaling the data

```
kmeans.data.scaled <- scale(kmeans.data)
wssplot(kmeans.data.scaled)
```



Let's do some clusterization

3 clusters

```r
# K-Means Cluster Analysis
fit3 <- kmeans(kmeans.data.scaled, 3)
# get cluster means
aggregate(kmeans.data.scaled,by=list(fit3$cluster),FUN=mean)
```

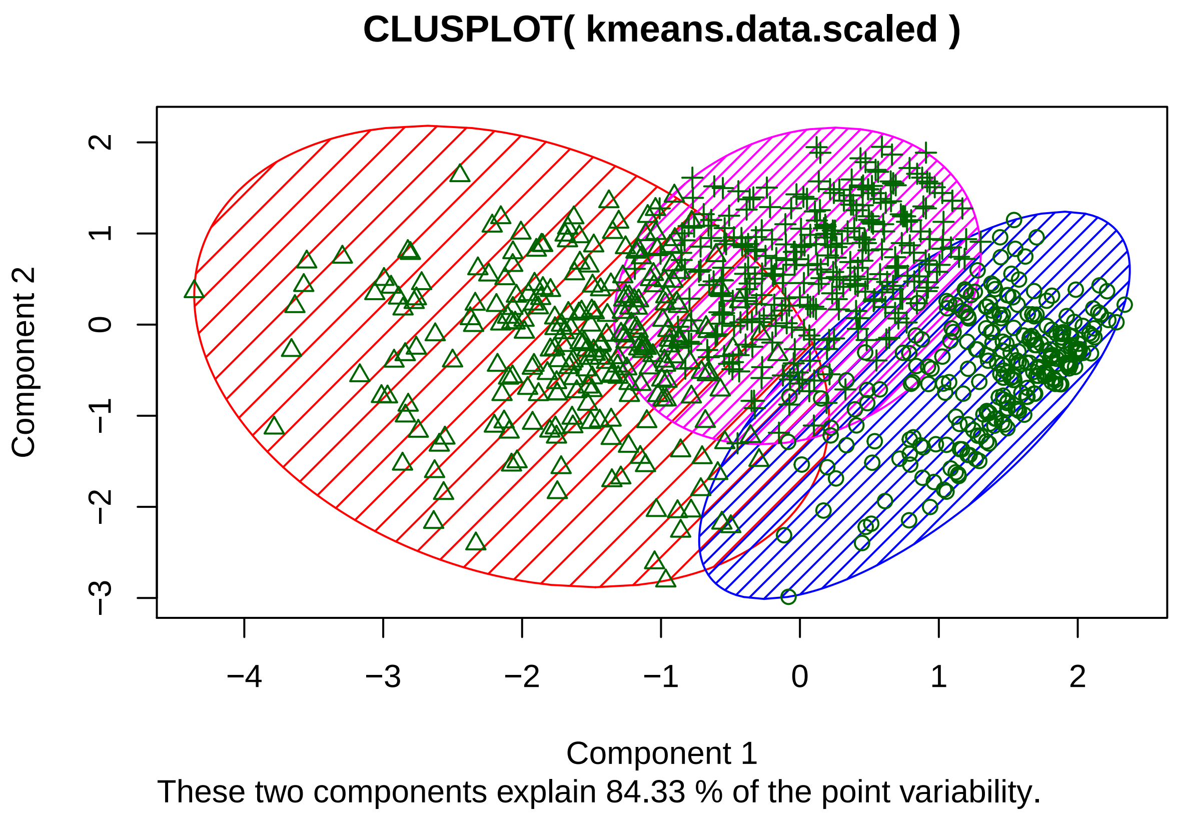
```
##   Group.1         Gm        Age     Salary
## 1       1 -1.2750839 -0.4807131 -0.6641629
## 2       2  0.6240421  0.9326274  1.1868775
## 3       3  0.5830986 -0.3564442 -0.4084625
```

```r
# append cluster assignment
fit3$tot.withinss #total within-cluster sum of squares
```

```
## [1] 1006.333
```

Cluster Plot against 1st 2 principal components

```r
# vary parameters for most readable graph
library(cluster)
clusplot(kmeans.data.scaled, fit3$cluster, color=TRUE, shade=TRUE)
```



4 clusters

```r
# K-Means Cluster Analysis
fit4 <- kmeans(kmeans.data.scaled, 4)
# get cluster means
aggregate(kmeans.data.scaled,by=list(fit4$cluster),FUN=mean)
```

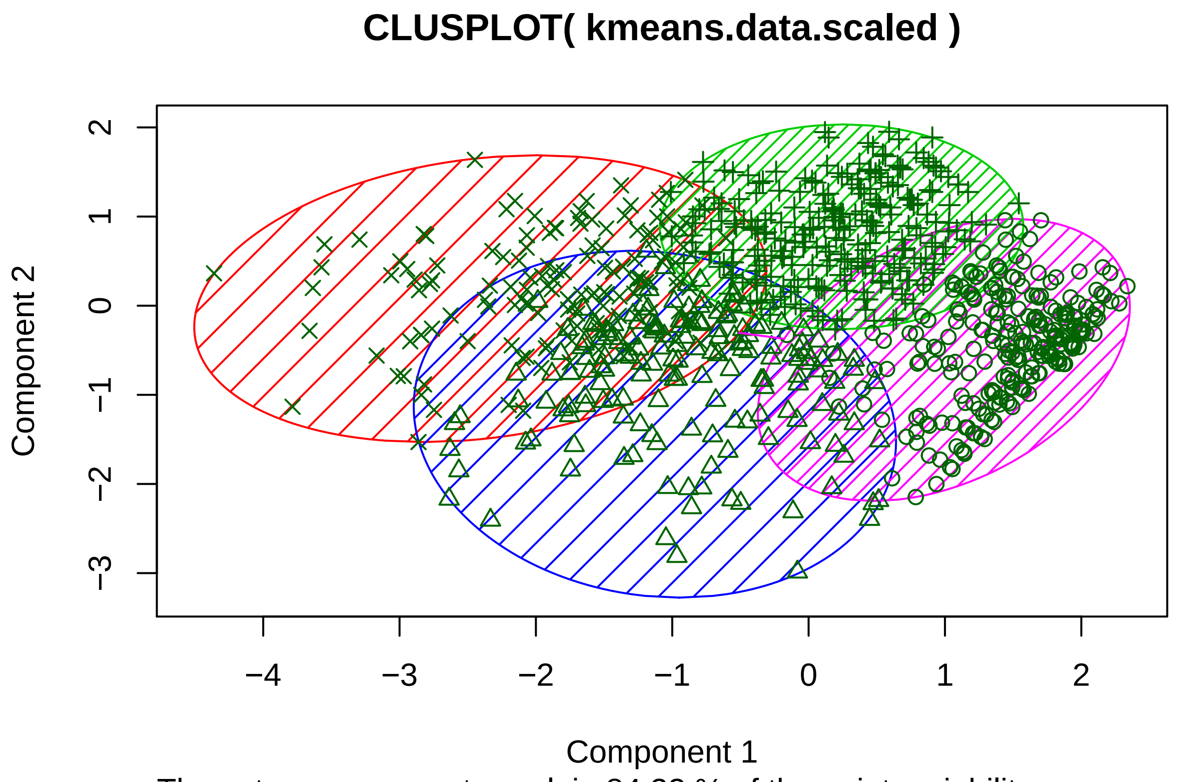
```
##   Group.1         Gm        Age      Salary
## 1       1 -1.2910727 -0.5847154 -0.67308221
## 2       2  0.2674915  1.3058219  0.08614001
## 3       3  0.6108817 -0.5813090 -0.38003721
## 4       4  0.7990348  0.5224068  1.73816663
```

```r
# append cluster assignment
fit4$tot.withinss #total within-cluster sum of squares
```

```
## [1] 772.9491
```

Cluster Plot against 1st 2 principal components

```r
# vary parameters for most readable graph
clusplot(kmeans.data.scaled, fit4$cluster, color=TRUE, shade=TRUE)
```



These two components explain 84.33 % of the point variability.

7 clusters

```r
# K-Means Cluster Analysis
fit7 <- kmeans(kmeans.data.scaled, 7)
# get cluster means
aggregate(kmeans.data.scaled,by=list(fit7$cluster),FUN=mean)
```

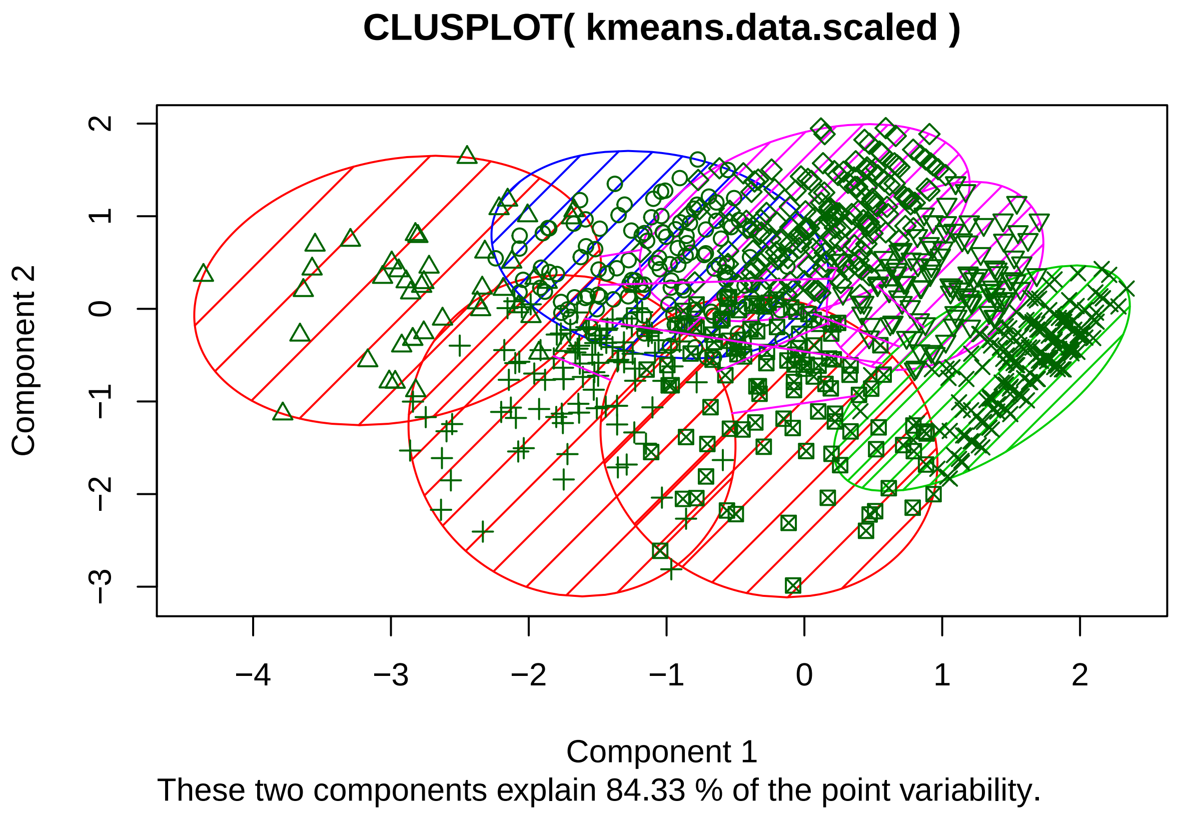
```
##   Group.1         Gm         Age     Salary
## 1       1  0.7198899  0.08273893  1.0259907
## 2       2  0.9529056  0.75135856  2.7388054
## 3       3  0.5314117  1.54295287  0.7870524
## 4       4 -1.5100957 -0.63245174 -0.6874792
## 5       5  0.8137360 -0.63694294 -0.4563627
## 6       6 -0.3322552 -0.77335084 -0.6681983
## 7       7 -0.1372308  0.99114365 -0.4660662
```

```
# append cluster assignment
fit7$tot.withinss #total within-cluster sum of squares
```

```
## [1] 545.4409
```

Cluster Plot against 1st 2 principal components

```
# vary parameters for most readable graph
clusplot(kmeans.data.scaled, fit7$cluster, color=TRUE, shade=TRUE)
```



comparing 2 cluster solutions 3-means and 4-means

```
library(fpc)
d <- dist(kmeans.data.scaled)
cluster.stats(d, fit3$cluster, fit4$cluster, compareonly = TRUE)
```

```
## $corrected.rand
## [1] 0.7083329
##
## $vi
## [1] 0.6987167
```

comparing 2 cluster solutions 4-means and 7-means

```r
library(fpc)
d <- dist(kmeans.data.scaled)
cluster.stats(d, fit4$cluster, fit7$cluster, compareonly = TRUE)
```

```
## $corrected.rand
## [1] 0.5015822
##
## $vi
## [1] 1.306123
```

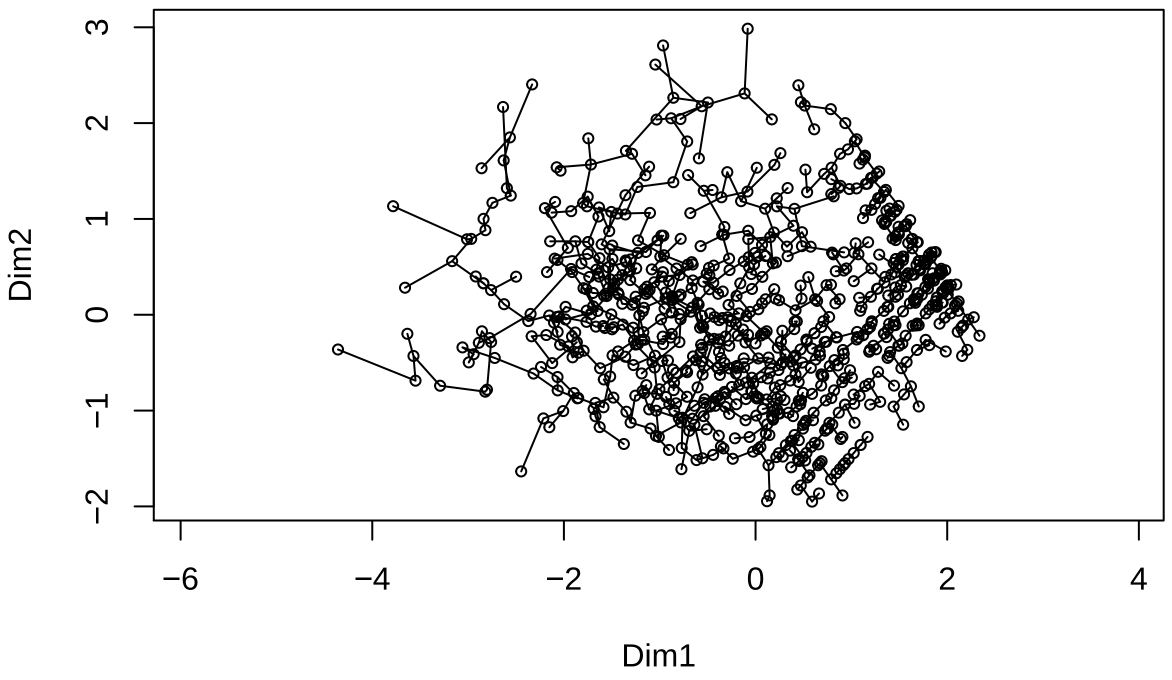
The correct choice of k is often ambiguous, with interpretations depending on the shape and scale of the distribution of points in a data set and the desired clustering resolution of the user. In addition, increasing k without penalty will always reduce the amount of error in the resulting clustering, to the extreme case of zero error if each data point is considered its own cluster (i.e., when k equals the number of data points, n). Intuitively then, the optimal choice of k will strike a balance between maximum compression of the data using a single cluster, and maximum accuracy by assigning each data point to its own cluster.

So, in our case we see that the total within-cluster sum of squares is redusing as k is getting bigger.

Problem 7.2

Build a minimum spanning tree connecting all points and delete 6 maximum edges step by step.

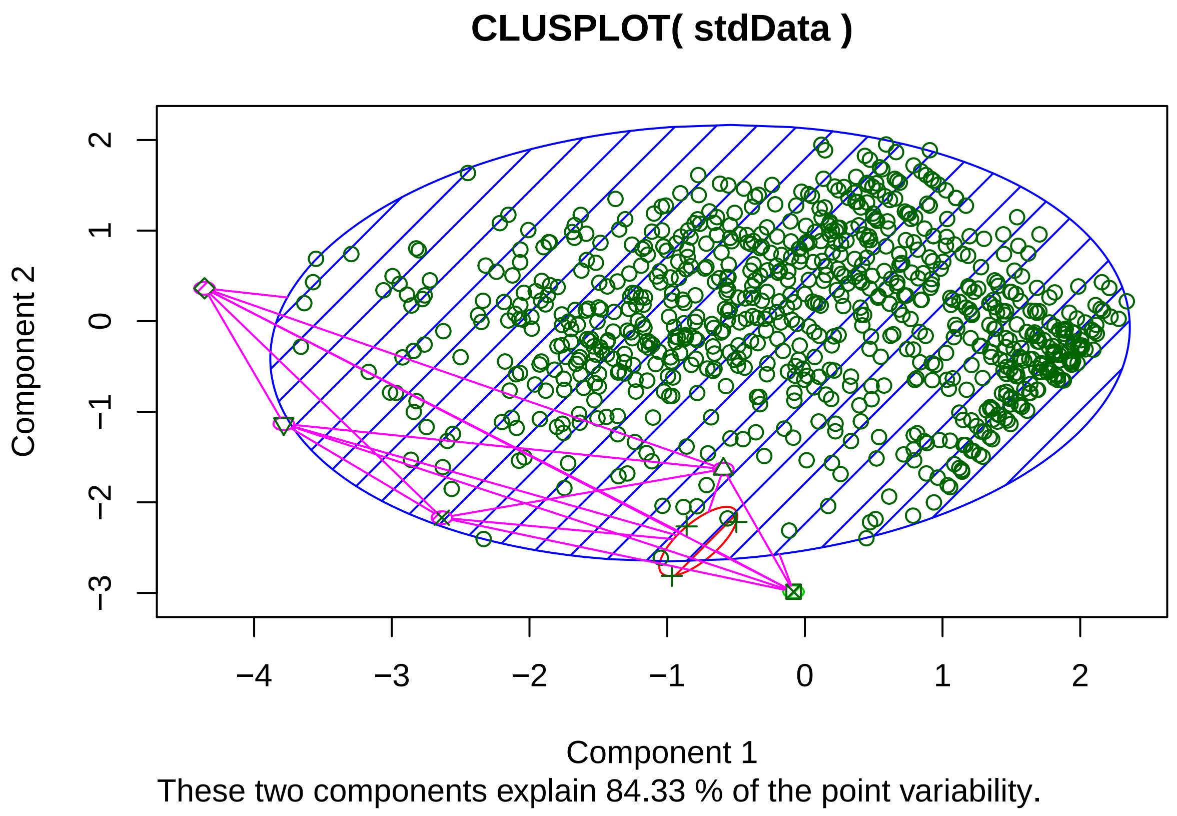
```
stdData <- kmeans.data.scaled
dis <- dist(stdData) #dissimilarity data
tr <- spantree(dis, toolong = 0) #build an MST
plot(tr, cmdscale(dis)) #plot an MST
```



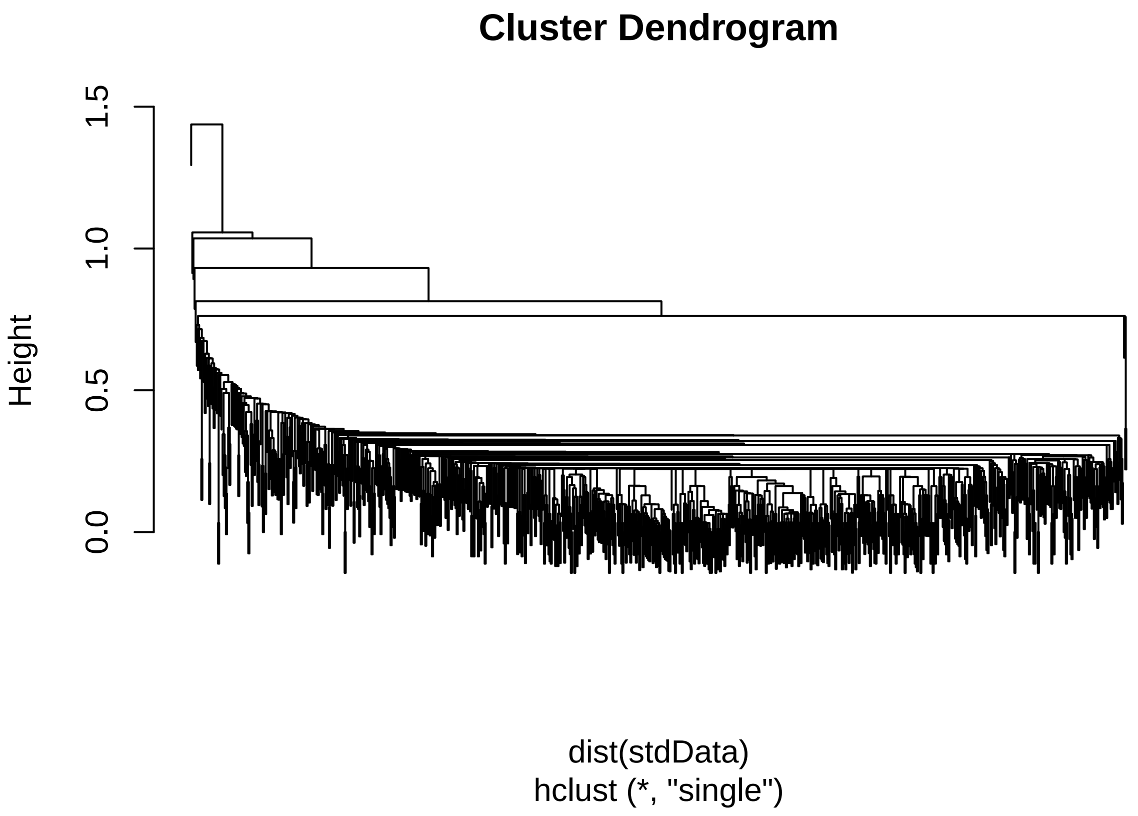
```
for(i in 1:6) #loop for finding 6 maximum edges in MST
{
  m <- which.max(tr$dist)
  tr$dist[m] <- NA
}
m <- which.max(tr$dist)
tres <- tr$dist[m] + 0.00001 #set a treshold
j <- distconnected(dis, toolong = tres) #find connected components in MST after deleting 6 edges
```

```
## Connectivity of distance matrix with threshold dissimilarity 0.758652
## Data are disconnected: 7 groups
## Groups sizes
##   1   2   3   4   5   6   7
## 828   1   3   1   1   1   1
```

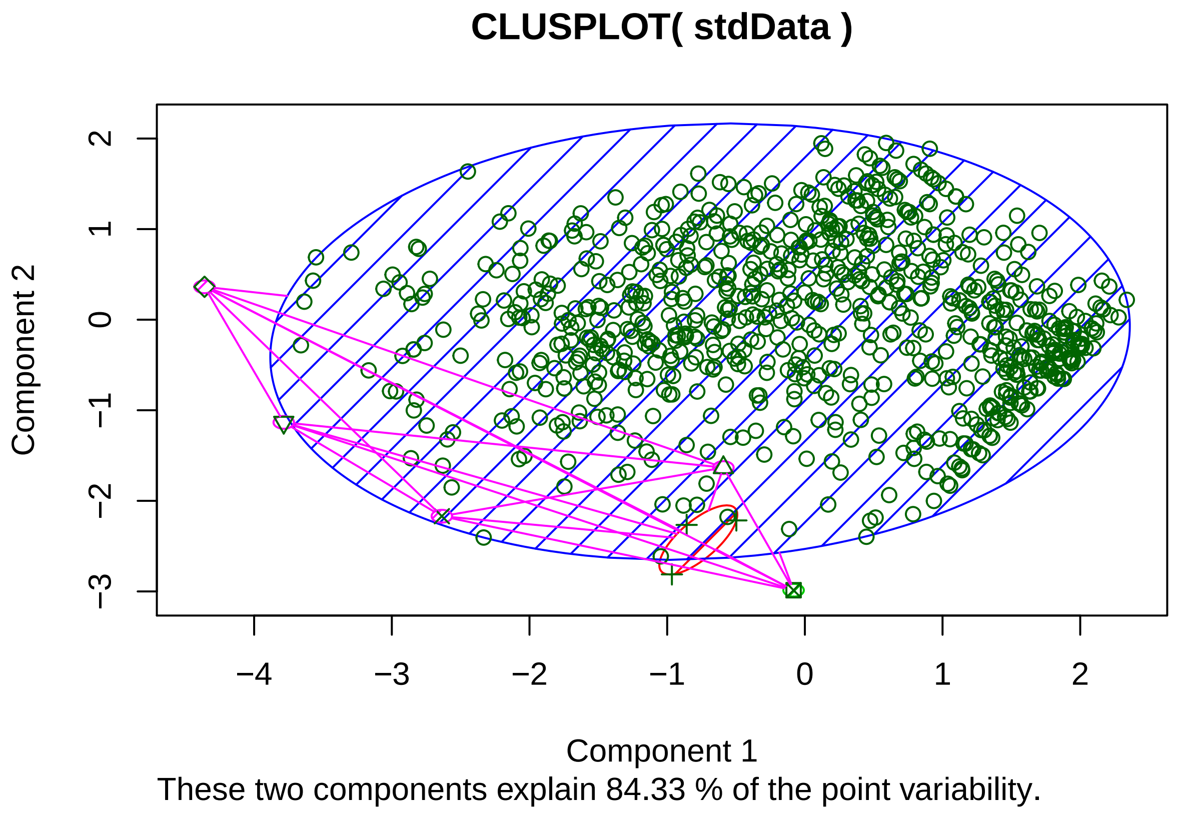
```
clusplot(stdData, j, color=TRUE, shade=TRUE) #plot clusters
```



```r
hc <- hclust(dist(stdData), method = "single")
plot(hc, labels = FALSE)
```



```r
memb <- cutree(hc, k = 7)
clusplot(stdData, memb, color=TRUE, shade=TRUE)
```



Consider a set of 2D points presented. Those in problem 2 clustered by using the single link approach, whereas those in problem 1, by using the square error criterion of K-Means.

Overall, this example demonstrates the major difference between conventional clustering and single link clustering: the latter finds elongated structures whereas the former cuts out convex parts. Sometimes, especially in the analysis of results of physical processes or experiments over real-world particles, the elongated structures do capture the essence of the data and are of great interest. In other cases, especially when entities/features have no intuitive geometric meaning - think of hockey players, for example, convex clusters make much more sense as groupings around their centroids.

By using MST (with single linkage) we get 1 big clusters and 6 small cluster consisting of 1/3 player/s. This is probably can be players that differs very much from other players and doesn't get to any big cluster. However, K-means placed these player to some more general cluster, because in this case clusters are more convex. The shape of such clusters lets the objects (that are standing far away from others in terms of features' distances) to get into them.