Final Project

Anh Trinh - 40069870

12/16/2020

Problem 1 - Olympic games

1.

```
#read the data and store in a data frame speed.data
speed.data <- read.table("speed.txt", header = TRUE)

#print the first 5 rows
speed.data[1:5, ]
```

```
##   Year Distance.100 Time Altitude
## 1 1900            2 22.2       25
## 2 1904            2 21.6      455
## 3 1908            2 22.4        8
## 4 1912            2 21.7       46
## 5 1920            2 22.0        3
```

2.

```
#calculate the average speed (in m/s)
Speed <- speed.data$Distance.100 * 100 / speed.data$Time

#add this data as a new column
modified.speed.data <- cbind(speed.data, Speed)
modified.speed.data
```

```
##    Year Distance.100  Time Altitude    Speed
## 1  1900            2 22.20       25 9.009009
## 2  1904            2 21.60      455 9.259259
## 3  1908            2 22.40        8 8.928571
## 4  1912            2 21.70       46 9.216590
## 5  1920            2 22.00        3 9.090909
## 6  1924            2 21.60       25 9.259259
## 7  1928            2 21.80        8 9.174312
## 8  1932            2 21.20      340 9.433962
## 9  1936            2 20.70      115 9.661836
## 10 1948            2 21.10        8 9.478673
## 11 1952            2 20.70       25 9.661836
## 12 1956            2 20.60        3 9.708738
## 13 1960            2 20.50       66 9.756098
```

```
## 14 1964             2  20.30       45  9.852217
## 15 1968             2  19.83     7349 10.085729
## 16 1972             2  20.00     1699 10.000000
## 17 1976             2  20.23      104  9.886307
## 18 1980             2  20.19      497  9.905894
## 19 1984             2  19.80      340 10.101010
## 20 1988             2  19.75      111 10.126582
## 21 1992             2  20.01        3  9.995002
## 22 1996             2  19.32     1026 10.351967
## 23 2000             2  20.09        3  9.955202
## 24 1900             4  49.40       25  8.097166
## 25 1904             4  49.20      455  8.130081
## 26 1908             4  50.00        8  8.000000
## 27 1912             4  48.20       46  8.298755
## 28 1920             4  49.60        3  8.064516
## 29 1924             4  47.60       25  8.403361
## 30 1928             4  47.80        8  8.368201
## 31 1932             4  46.20      340  8.658009
## 32 1936             4  46.50      115  8.602151
## 33 1948             4  46.20        8  8.658009
## 34 1952             4  45.90       25  8.714597
## 35 1956             4  46.70        3  8.565310
## 36 1960             4  44.90       66  8.908686
## 37 1964             4  45.10       45  8.869180
## 38 1968             4  43.80     7349  9.132420
## 39 1972             4  44.66     1699  8.956561
## 40 1976             4  44.26      104  9.037506
## 41 1980             4  44.60      497  8.968610
## 42 1984             4  44.27      340  9.035464
## 43 1988             4  43.87      111  9.117848
## 44 1992             4  43.50        3  9.195402
## 45 1996             4  43.49     1026  9.197517
## 46 2000             4  43.84        3  9.124088
## 47 1900             8 121.40       25  6.589786
## 48 1904             8 116.00      455  6.896552
## 49 1908             8 112.80        8  7.092199
## 50 1912             8 111.90       46  7.149240
## 51 1920             8 113.40        3  7.054674
## 52 1924             8 112.40       25  7.117438
## 53 1928             8 111.80        8  7.155635
## 54 1932             8 109.80      340  7.285974
## 55 1936             8 112.90      115  7.085917
## 56 1948             8 109.20        8  7.326007
## 57 1952             8 109.20       25  7.326007
## 58 1956             8 107.70        3  7.428041
## 59 1960             8 106.30       66  7.525870
## 60 1964             8 105.10       45  7.611798
## 61 1968             8 104.30     7349  7.670182
## 62 1972             8 105.90     1699  7.554297
## 63 1976             8 103.50      104  7.729469
## 64 1980             8 105.40      497  7.590133
## 65 1984             8 103.00      340  7.766990
## 66 1988             8 103.45      111  7.733204
## 67 1992             8 103.66        3  7.717538
```

```
## 68 1996             8 102.58     1026  7.798791
## 69 2000             8 105.08        3  7.613247
## 70 1900            15 246.00       25  6.097561
## 71 1904            15 245.40      455  6.112469
## 72 1908            15 243.40        8  6.162695
## 73 1912            15 236.80       46  6.334459
## 74 1920            15 241.80        3  6.203474
## 75 1924            15 233.60       25  6.421233
## 76 1928            15 233.20        8  6.432247
## 77 1932            15 231.20      340  6.487889
## 78 1936            15 227.80      115  6.584723
## 79 1948            15 225.20        8  6.660746
## 80 1952            15 225.20       25  6.660746
## 81 1956            15 221.20        3  6.781193
## 82 1960            15 215.60       66  6.957328
## 83 1964            15 218.10       45  6.877579
## 84 1968            15 214.90     7349  6.979991
## 85 1972            15 216.30     1699  6.934813
## 86 1976            15 219.20      104  6.843066
## 87 1980            15 218.40      497  6.868132
## 88 1984            15 212.50      340  7.058824
## 89 1988            15 215.96      111  6.945731
## 90 1992            15 220.12        3  6.814465
## 91 1996            15 215.78     1026  6.951525
## 92 2000            15 212.07        3  7.073136
```

```r
#print the first 5 rows of the modified data
modified.speed.data[1:5, ]
```

```
##   Year Distance.100 Time Altitude    Speed
## 1 1900            2 22.2       25 9.009009
## 2 1904            2 21.6      455 9.259259
## 3 1908            2 22.4        8 8.928571
## 4 1912            2 21.7       46 9.216590
## 5 1920            2 22.0        3 9.090909
```

3.

```r
#sort the data by increasing value of year
sorted.data <- modified.speed.data[order(modified.speed.data$Year), ]
sorted.data
```

```
##    Year Distance.100   Time Altitude     Speed
## 1  1900            2  22.20       25  9.009009
## 24 1900            4  49.40       25  8.097166
## 47 1900            8 121.40       25  6.589786
## 70 1900           15 246.00       25  6.097561
## 2  1904            2  21.60      455  9.259259
## 25 1904            4  49.20      455  8.130081
## 48 1904            8 116.00      455  6.896552
## 71 1904           15 245.40      455  6.112469
## 3  1908            2  22.40        8  8.928571
## 26 1908            4  50.00        8  8.000000
```

```
## 49 1908            8 112.80       8  7.092199
## 72 1908           15 243.40       8  6.162695
## 4  1912            2  21.70      46  9.216590
## 27 1912            4  48.20      46  8.298755
## 50 1912            8 111.90      46  7.149240
## 73 1912           15 236.80      46  6.334459
## 5  1920            2  22.00       3  9.090909
## 28 1920            4  49.60       3  8.064516
## 51 1920            8 113.40       3  7.054674
## 74 1920           15 241.80       3  6.203474
## 6  1924            2  21.60      25  9.259259
## 29 1924            4  47.60      25  8.403361
## 52 1924            8 112.40      25  7.117438
## 75 1924           15 233.60      25  6.421233
## 7  1928            2  21.80       8  9.174312
## 30 1928            4  47.80       8  8.368201
## 53 1928            8 111.80       8  7.155635
## 76 1928           15 233.20       8  6.432247
## 8  1932            2  21.20     340  9.433962
## 31 1932            4  46.20     340  8.658009
## 54 1932            8 109.80     340  7.285974
## 77 1932           15 231.20     340  6.487889
## 9  1936            2  20.70     115  9.661836
## 32 1936            4  46.50     115  8.602151
## 55 1936            8 112.90     115  7.085917
## 78 1936           15 227.80     115  6.584723
## 10 1948            2  21.10       8  9.478673
## 33 1948            4  46.20       8  8.658009
## 56 1948            8 109.20       8  7.326007
## 79 1948           15 225.20       8  6.660746
## 11 1952            2  20.70      25  9.661836
## 34 1952            4  45.90      25  8.714597
## 57 1952            8 109.20      25  7.326007
## 80 1952           15 225.20      25  6.660746
## 12 1956            2  20.60       3  9.708738
## 35 1956            4  46.70       3  8.565310
## 58 1956            8 107.70       3  7.428041
## 81 1956           15 221.20       3  6.781193
## 13 1960            2  20.50      66  9.756098
## 36 1960            4  44.90      66  8.908686
## 59 1960            8 106.30      66  7.525870
## 82 1960           15 215.60      66  6.957328
## 14 1964            2  20.30      45  9.852217
## 37 1964            4  45.10      45  8.869180
## 60 1964            8 105.10      45  7.611798
## 83 1964           15 218.10      45  6.877579
## 15 1968            2  19.83    7349 10.085729
## 38 1968            4  43.80    7349  9.132420
## 61 1968            8 104.30    7349  7.670182
## 84 1968           15 214.90    7349  6.979991
## 16 1972            2  20.00    1699 10.000000
## 39 1972            4  44.66    1699  8.956561
## 62 1972            8 105.90    1699  7.554297
## 85 1972           15 216.30    1699  6.934813
```

```
## 17 1976            2  20.23      104  9.886307
## 40 1976            4  44.26      104  9.037506
## 63 1976            8 103.50      104  7.729469
## 86 1976           15 219.20      104  6.843066
## 18 1980            2  20.19      497  9.905894
## 41 1980            4  44.60      497  8.968610
## 64 1980            8 105.40      497  7.590133
## 87 1980           15 218.40      497  6.868132
## 19 1984            2  19.80      340 10.101010
## 42 1984            4  44.27      340  9.035464
## 65 1984            8 103.00      340  7.766990
## 88 1984           15 212.50      340  7.058824
## 20 1988            2  19.75      111 10.126582
## 43 1988            4  43.87      111  9.117848
## 66 1988            8 103.45      111  7.733204
## 89 1988           15 215.96      111  6.945731
## 21 1992            2  20.01        3  9.995002
## 44 1992            4  43.50        3  9.195402
## 67 1992            8 103.66        3  7.717538
## 90 1992           15 220.12        3  6.814465
## 22 1996            2  19.32     1026 10.351967
## 45 1996            4  43.49     1026  9.197517
## 68 1996            8 102.58     1026  7.798791
## 91 1996           15 215.78     1026  6.951525
## 23 2000            2  20.09        3  9.955202
## 46 2000            4  43.84        3  9.124088
## 69 2000            8 105.08        3  7.613247
## 92 2000           15 212.07        3  7.073136
```

```
#print the first 10 rows of the sorted data
sorted.data[1:10, ]
```

```
##    Year Distance.100  Time Altitude    Speed
## 1  1900            2  22.2       25 9.009009
## 24 1900            4  49.4       25 8.097166
## 47 1900            8 121.4       25 6.589786
## 70 1900           15 246.0       25 6.097561
## 2  1904            2  21.6      455 9.259259
## 25 1904            4  49.2      455 8.130081
## 48 1904            8 116.0      455 6.896552
## 71 1904           15 245.4      455 6.112469
## 3  1908            2  22.4        8 8.928571
## 26 1908            4  50.0        8 8.000000
```

4.

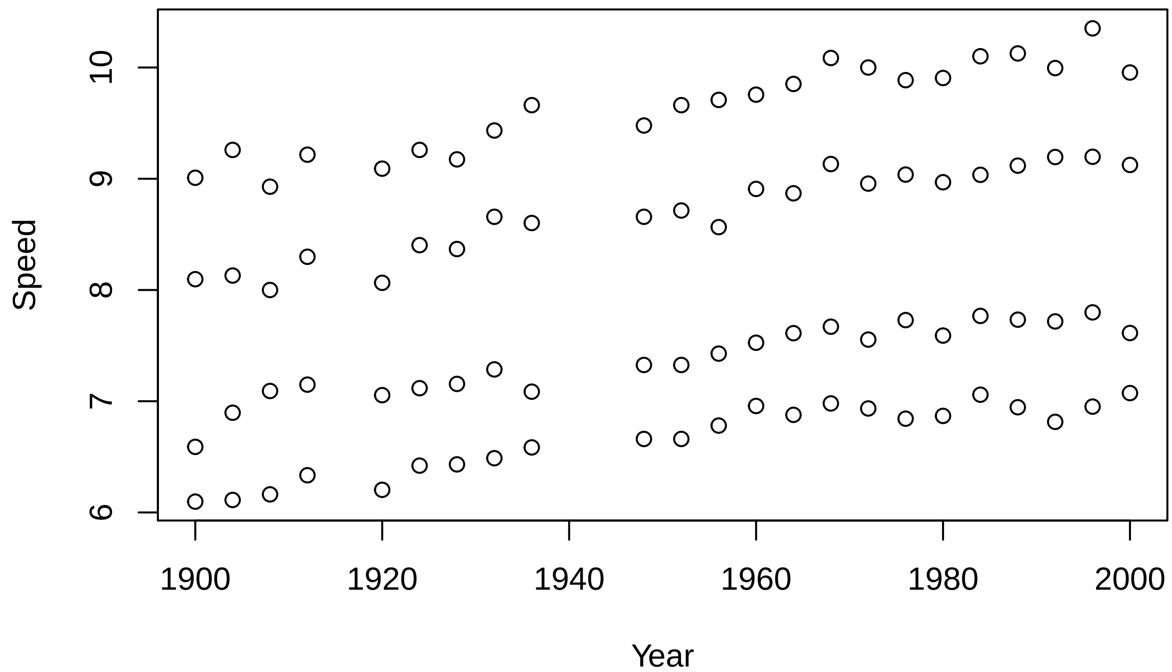
```
#create a new data frame containing 2 columns
speed.year <- data.frame("Year"=modified.speed.data$Year,
          "Speed"=modified.speed.data$Speed)
speed.year
```

```
##    Year     Speed
```

```
## 1  1900  9.009009
## 2  1904  9.259259
## 3  1908  8.928571
## 4  1912  9.216590
## 5  1920  9.090909
## 6  1924  9.259259
## 7  1928  9.174312
## 8  1932  9.433962
## 9  1936  9.661836
## 10 1948  9.478673
## 11 1952  9.661836
## 12 1956  9.708738
## 13 1960  9.756098
## 14 1964  9.852217
## 15 1968 10.085729
## 16 1972 10.000000
## 17 1976  9.886307
## 18 1980  9.905894
## 19 1984 10.101010
## 20 1988 10.126582
## 21 1992  9.995002
## 22 1996 10.351967
## 23 2000  9.955202
## 24 1900  8.097166
## 25 1904  8.130081
## 26 1908  8.000000
## 27 1912  8.298755
## 28 1920  8.064516
## 29 1924  8.403361
## 30 1928  8.368201
## 31 1932  8.658009
## 32 1936  8.602151
## 33 1948  8.658009
## 34 1952  8.714597
## 35 1956  8.565310
## 36 1960  8.908686
## 37 1964  8.869180
## 38 1968  9.132420
## 39 1972  8.956561
## 40 1976  9.037506
## 41 1980  8.968610
## 42 1984  9.035464
## 43 1988  9.117848
## 44 1992  9.195402
## 45 1996  9.197517
## 46 2000  9.124088
## 47 1900  6.589786
## 48 1904  6.896552
## 49 1908  7.092199
## 50 1912  7.149240
## 51 1920  7.054674
## 52 1924  7.117438
## 53 1928  7.155635
## 54 1932  7.285974
```

```
## 55 1936  7.085917
## 56 1948  7.326007
## 57 1952  7.326007
## 58 1956  7.428041
## 59 1960  7.525870
## 60 1964  7.611798
## 61 1968  7.670182
## 62 1972  7.554297
## 63 1976  7.729469
## 64 1980  7.590133
## 65 1984  7.766990
## 66 1988  7.733204
## 67 1992  7.717538
## 68 1996  7.798791
## 69 2000  7.613247
## 70 1900  6.097561
## 71 1904  6.112469
## 72 1908  6.162695
## 73 1912  6.334459
## 74 1920  6.203474
## 75 1924  6.421233
## 76 1928  6.432247
## 77 1932  6.487889
## 78 1936  6.584723
## 79 1948  6.660746
## 80 1952  6.660746
## 81 1956  6.781193
## 82 1960  6.957328
## 83 1964  6.877579
## 84 1968  6.979991
## 85 1972  6.934813
## 86 1976  6.843066
## 87 1980  6.868132
## 88 1984  7.058824
## 89 1988  6.945731
## 90 1992  6.814465
## 91 1996  6.951525
## 92 2000  7.073136
```

```r
#plot the speed as a function of the year
plot(speed.year$Year, speed.year$Speed, xlab = "Year",
     ylab = "Speed")
```



I observe that the speed for 200m, 400m, 800m, 1500m, is decreasing respectively for each year (means that the speed for 200m is greatest, and one of 1500m is smallest)

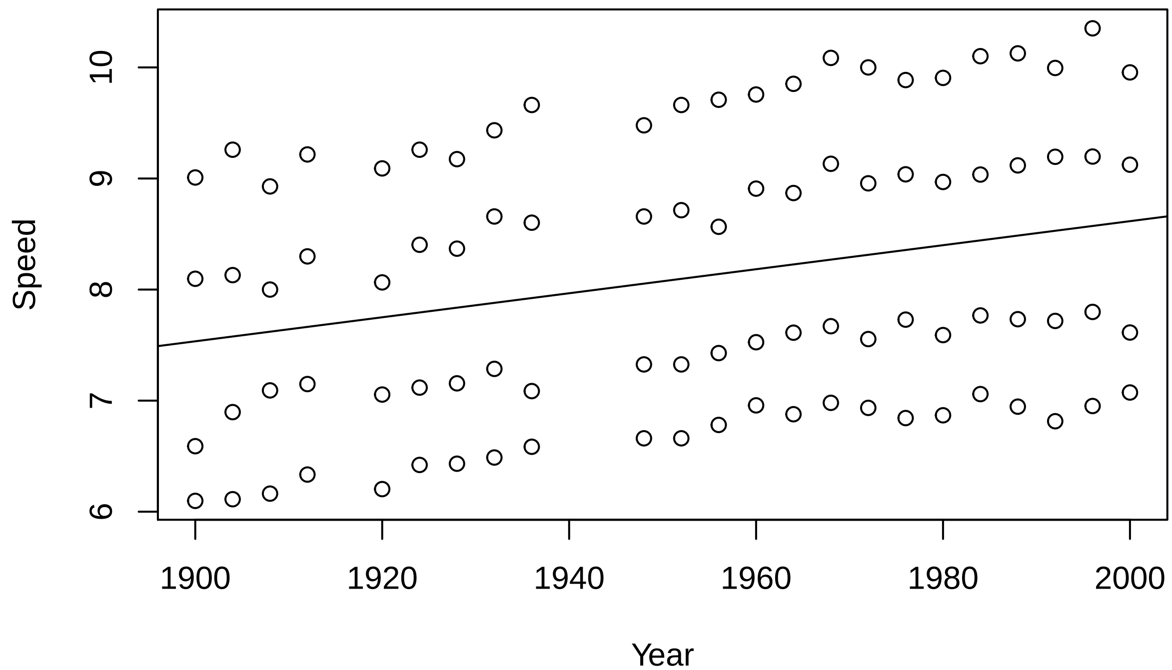
5.

```
lm(speed.year$Speed ~ speed.year$Year)
```

```
##
## Call:
## lm(formula = speed.year$Speed ~ speed.year$Year)
##
## Coefficients:
##     (Intercept)  speed.year$Year
##       -13.01660          0.01082
```

The best fit line is $y = -13.01660 + 0.01082x$

```
plot(speed.year$Year, speed.year$Speed, abline(lm(speed.year$Speed ~ speed.year$Year)), xlab = "Year",
     ylab = "Speed")
```



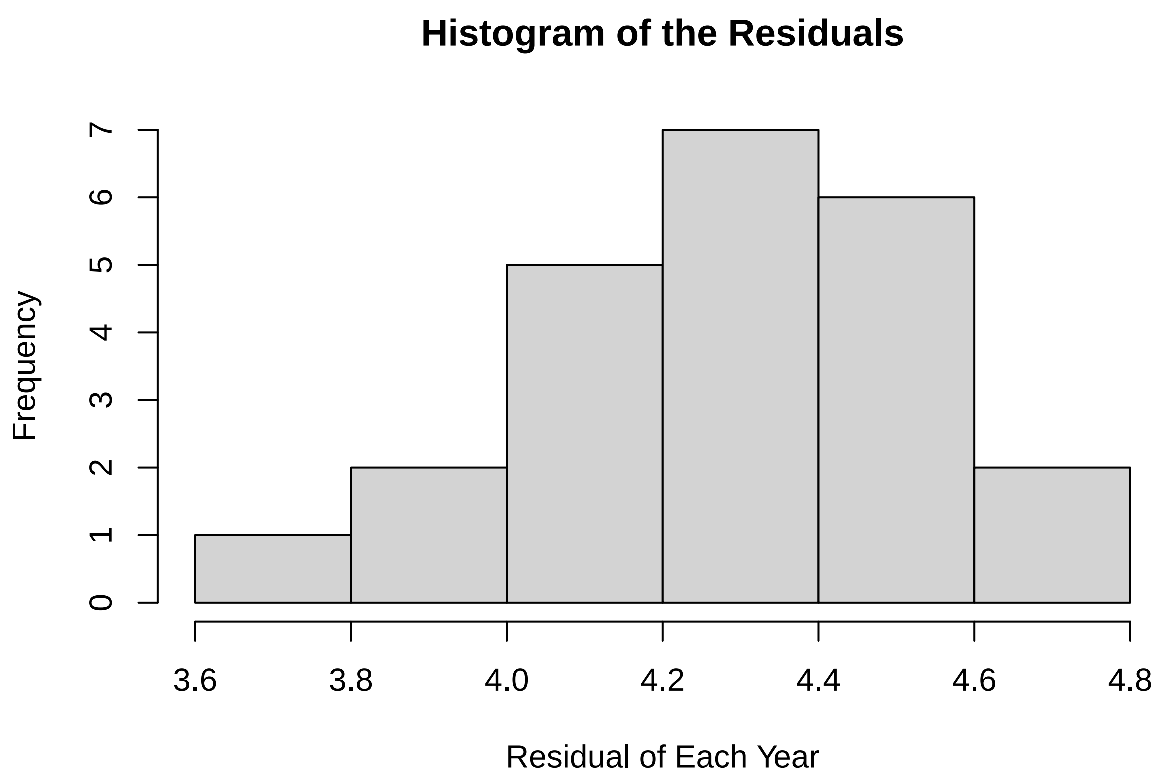
6. Assume the best fitting line computed in part 5 has the form $y = mx + q$, so $m = 0.01082$ and $q = -13.01660$

7. The 100m race will be likely to run in less than 7 seconds, which means the speed has to be greater than $\frac{100}{7}$ m/s
   According to the best fit line found in part 6, the year that 100m race will be run at exactly 7 seconds is $\frac{\frac{100}{7}+13.01660}{0.01082} = 2523.32$
   Thus, 100 meters race will be likely to be run in less than 7 seconds in year 2524

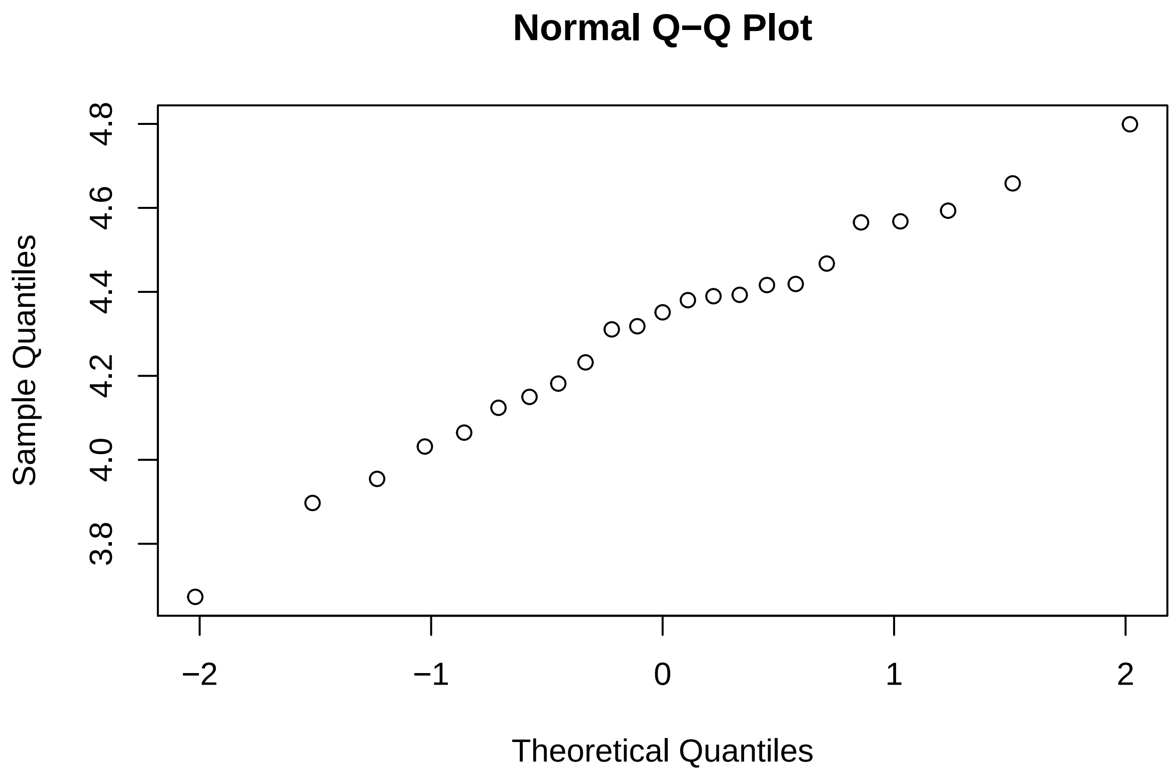
8 + 9

```
#compute the residuals i.e. the differences between the actual average speed
#and the speed predicted by the best fitting line
predicted.speed <- -13.01660 + 0.01082 * speed.year$Year

Residual <- abs(speed.year$Speed - predicted.speed)

#to find residual each year, first we have to sort by increasing value of year
#then calculate sum of all residuals of each year
speed.year <- cbind(speed.year, Residual)
sorted.speed.year <- speed.year[order(speed.year$Year),]
i=1
residual <- c()
while (i<=92){
  res <- sum(sorted.speed.year$Residual[i:(i+3)])
```

```
  residual <- c(residual,res)
  i=i+4
}

#histogram
hist(residual, xlab = "Residual of Each Year", main = "Histogram of the Residuals")
```




```
qqnorm(residual)
```



By looking at the histogram as well as the approximately linear qq plot, we can see that the residuals are very close to be normally distributed

10.

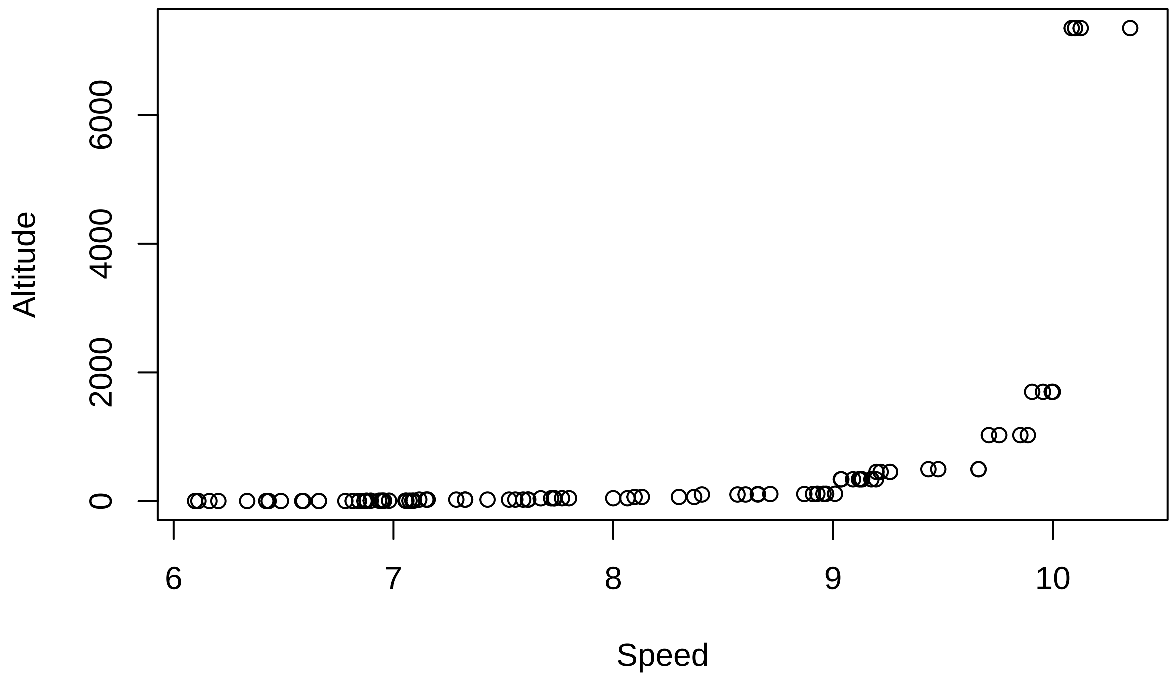
```r
# the original matrix A from which QR was constructed
A <- qr.X(lm(speed.year$Speed ~ speed.year$Year)$qr)
#using qr solve
qr.solve(A, speed.year$Speed)
```

```
##     (Intercept) speed.year$Year
##    -13.01660062      0.01081622
```

$m = 0.01081622$ and $q = -13.01660062$

11.

```r
qqplot(modified.speed.data$Speed, modified.speed.data$Altitude, xlab = "Speed", ylab = "Altitude")
```




```r
cor.test(modified.speed.data$Speed, modified.speed.data$Altitude)$p.value
```

```
## [1] 0.407621
```

Since the p-value is > 5%, we can conclude that the altitude of the venue and the average speed are not significantly correlated

Problem 2 - Modified Newton's method

1.

```r
EvalPoly <- function(c, x){
  p <- 0
  for (i in 1:length(c)){
    p <- p + c[i] * x^(i-1)
  }
  return (p)
}
```

2.

```r
EvalPoly(c(1, -1.7, 0, 3.5), 13.4)
```

```
## [1] 8399.584
```

3.

```r
PolyDerEval <- function(c, x){
  p <- 0
  for (i in 2: length(c)){
    p <- p + (i-1)*c[i]*x^(i-2)
  }
  return(p)
}
```

4.

```r
PolyDerEval(c(1, -1.7, 0, 3.5), 13.4)
```

```
## [1] 1883.68
```

5.

```r
NewtonPoly <- function(c, x0, TOL){
  k <- 1
  approxi <- c(x0)
  while (abs(EvalPoly(c, x0)) > TOL & k <= 1000){
    x <- x0 - EvalPoly(c, x0)/PolyDerEval(c,x0) #newton's method
    k <- k + 1
    approxi <- c(approxi,x)
    x0 <- x

  }
  return(approxi)
}
```

6.

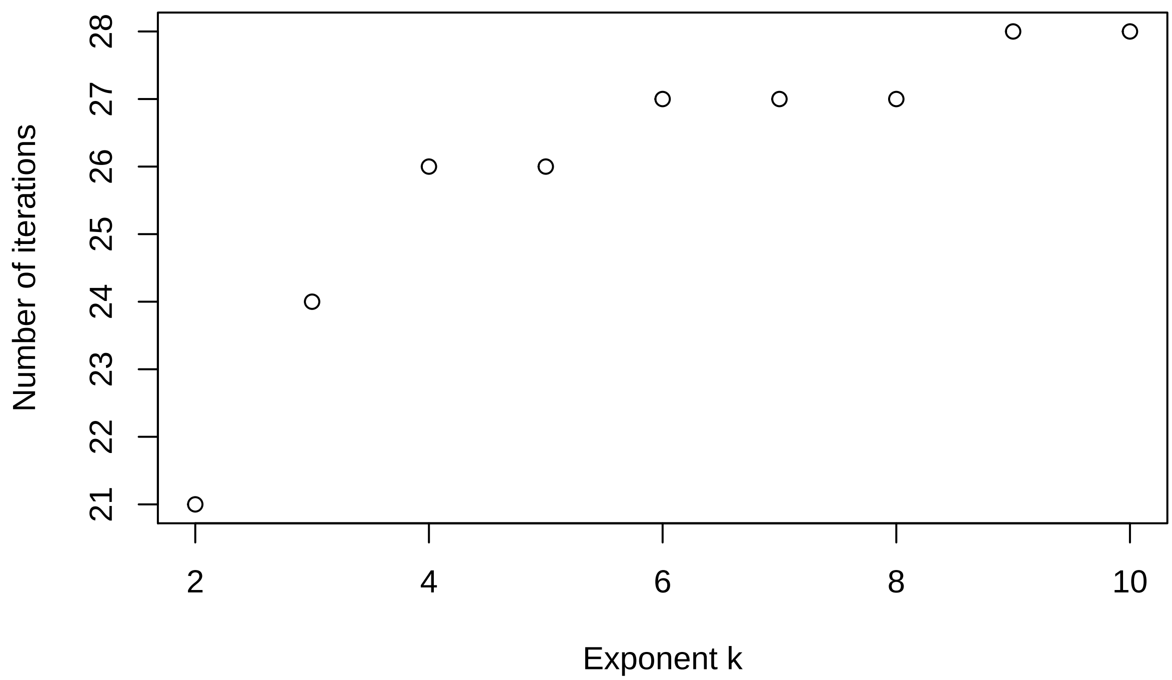
```r
#print the sequence of approximations by Newton's method
NewtonPoly(c(2.3, -7.1, 0, 1), -1, 10^-10)
```

```
## [1] -1.00000000  1.04878049 -0.00189262  0.32394415  0.32895377  0.32895739
```

7.

```r
iter <- c()
for (k in 2:10){
  #number of iterations needed by Newton's method
  iter <- c(iter, length(NewtonPoly(c(rep(0,k),1), 1, 1e-12)))
  #since x^2 = 0 + 0x + 1x^2
```

```r
  #       x^3 = 0 + 0x + 0x^2 + 1x^3
  #       x^4 = 0 + 0x + 0x^2 + 0x^3 + 1x^4 etc
}

plot(2:10, iter, xlab = "Exponent k", ylab = "Number of iterations")
```



8.

```r
PolyDer2Eval <- function(c, x){
  p <- 0
  for (i in 3: length(c)){
    p <- (i-2)*(i-1)*c[i]*x^(i-3)
  }
  return(p)
}

PolyDer2Eval(c(1, -1.7, 0, 3.5), 13.4)
```

```
## [1] 281.4
```

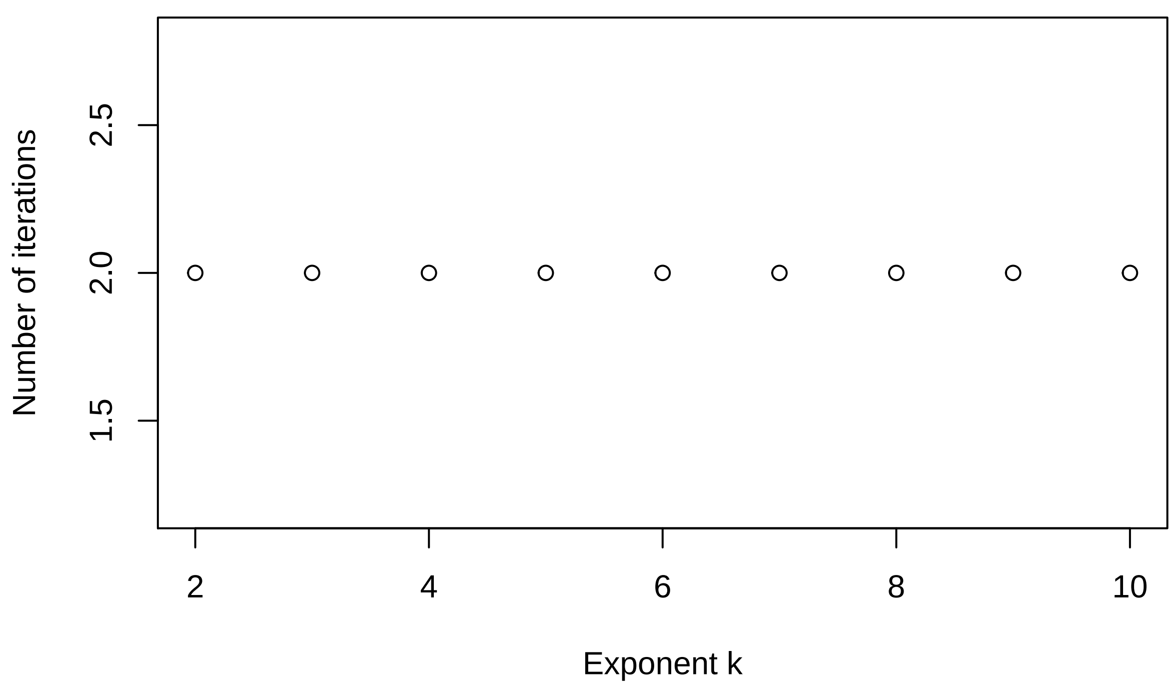
9.

```r
ModifiedNewtonPoly <- function(c, x0, TOL){
  k <- 1
  approxi <- c(x0)
```

```
  while (abs(EvalPoly(c, x0)) > TOL & k <= 1000){
    x <- x0 -
      (EvalPoly(c,x0) * PolyDerEval(c,x0)) /
      ((PolyDerEval(c,x0))^2 - EvalPoly(c,x0) * PolyDer2Eval(c,x0)) #modified newton's method
    k <- k + 1
    approxi <- c(approxi,x)
    x0 <- x
  }
  return(approxi)
}
```

10.

```
iter <- c()
for (k in 2:10){
  #number of iterations needed by Modified Newton's method
  iter <- c(iter, length(ModifiedNewtonPoly(c(rep(0,k),1), 1, 1e-12)))
  #since x^2 = 0 + 0x + 1x^2
  #       x^3 = 0 + 0x + 0x^2 + 1x^3
  #       x^4 = 0 + 0x + 0x^2 + 0x^3 + 1x^4 etc
}

plot(2:10, iter, xlab = "Exponent k", ylab = "Number of iterations")
```



For each k, the number of iterations are the same to each other.

Problem 3 - Ada's walk

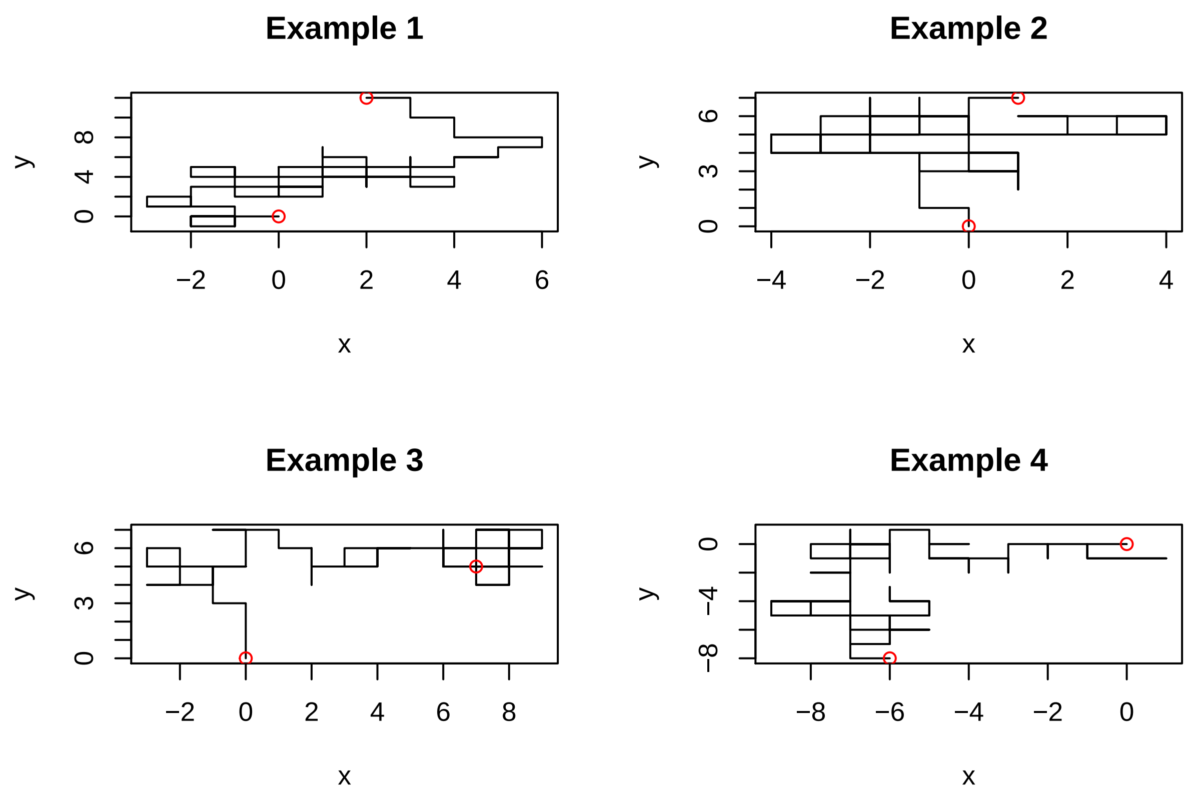
```r
AdaWalk <- function(){
  #at A_0
  x <- 0
  y <- 0
  xpos <- c() #x coordinates of each move
  ypos <- c() #y coordinates of each move
  xpos[1] <- 0
  ypos[1] <- 0
  for (i in 1:100){ # final position is A_100
    r <- runif(1) #random uniform numbers
    if (r <= 0.25){
      x <- x + 1 #move right
    }
    if (r > 0.25 & r <= 0.5){
      x <- x - 1 #move left
    }
    if (r > 0.5 & r <= 0.75){
      y <- y + 1 #move up
    }
    if(r > 0.75){
      y <- y - 1 #move down
    }
    xpos[i+1] <- x #update into the vector xpos
    ypos[i+1] <- y #update into the vector ypos
    if (xpos[i+1] == xpos[i] & ypos[i+1] == ypos[i]) #A_t != A_{t+1}
      break
    if(xpos[i+1] == 0 & ypos[i+1] == 0){ #if Ada is back in position (0,0)
      break
    }
  }
  return(rbind(xpos,ypos))
}
AdaWalk()
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## xpos    0    0   -1   -2   -1    0   -1   -1    0
## ypos    0    1    1    1    1    1    1    0    0
```

2.

```r
#example 1
A1 <- AdaWalk()
x1 <- A1[1,]
y1 <- A1[2,]
x1.min <- min(x1)
x1.max <- max(x1)
y1.min <- min(y1)
y1.max <- max(y1)

par(mfrow = c(2,2))
```

```r
plot(x1, y1, type="l", xlab="x", ylab = "y", main = "Example 1",
     xlim = range(x1.min:x1.max), ylim = range(y1.min:y1.max))
points(cbind(0,0),pch=1,col="red")
points(cbind(A1[1,ncol(A1)],A1[2,ncol(A1)]),pch=1,col="red")

#example 2
A2 <- AdaWalk()
x2 <- A2[1,]
y2 <- A2[2,]
x2.min <- min(x2)
x2.max <- max(x2)
y2.min <- min(y2)
y2.max <- max(y2)

plot(x2, y2, type="l", xlab="x", ylab = "y", main = "Example 2",
     xlim = range(x2.min:x2.max), ylim = range(y2.min:y2.max))
points(cbind(0,0),pch=1,col="red")
points(cbind(A2[1,ncol(A2)],A2[2,ncol(A2)]),pch=1,col="red")

#example 3
A3 <- AdaWalk()
x3 <- A3[1,]
y3 <- A3[2,]
x3.min <- min(x3)
x3.max <- max(x3)
y3.min <- min(y3)
y3.max <- max(y3)

plot(x3, y3, type="l", xlab="x", ylab = "y", main = "Example 3",
     xlim = range(x3.min:x3.max), ylim = range(y3.min:y3.max))
points(cbind(0,0),pch=1,col="red")
points(cbind(A3[1,ncol(A3)],A3[2,ncol(A3)]),pch=1,col="red")

#example 4
A4 <- AdaWalk()
x4 <- A4[1,]
y4 <- A4[2,]
x4.min <- min(x4)
x4.max <- max(x4)
y4.min <- min(y4)
y4.max <- max(y4)

plot(x4, y4, type="l", xlab="x", ylab = "y", main = "Example 4",
     xlim = range(x4.min:x4.max), ylim = range(y4.min:y4.max))
points(cbind(0,0),pch=1,col="red")
points(cbind(A4[1,ncol(A4)],A4[2,ncol(A4)]),pch=1,col="red")
```



3+4.

```
#Estimate the probability that Ada comes back to the orgin (0, 0) in at most
#100 steps. Use a Monte Carlo simulation with at least 100 repeated experiments
#(or more, if your computer can)
n.mc <- 199
success <- 0

x <- 0
y <- 0
xpos <- c() #x coordinates of each move
ypos <- c() #y coordinates of each move
xpos[1] <- 0
ypos[1] <- 0

for (i.mc in 1:n.mc){
  flag <- F
  for (i in 1:100){ # final position is A_100
    r <- runif(1) #random uniform numbers
    if (r <= 0.25){
      x <- x + 1 #move right
    }
    if (r > 0.25 & r <= 0.5){
      x <- x - 1 #move left
    }
    if (r > 0.5 & r <= 0.75){
      y <- y + 1 #move up
```

```
    }
    if(r > 0.75){
      y <- y - 1 #move down
    }
    xpos[i+1] <- x #update into the vector xpos
    ypos[i+1] <- y #update into the vector ypos
    if (i <= 100 & xpos[i+1] == 0 & ypos[i+1] == 0 & !flag){
      flag <- T
      success <- success + 1
    }
    if (xpos[i+1] == xpos[i] & ypos[i+1] == ypos[i]){ #A_t != A_{t+1}
     break
    }
    if(xpos[i+1] == 0 & ypos[i+1] == 0){ #if Ada is back in position (0,0)
    break
    }
    # if (i <= 100 & xpos[i+1] == 0 & ypos[i+1] == 0 & !flag){
    #   flag <- T
    #   success <- success + 1
    # }
  }
}
cat("The estimated probability of X that Ada comes back to the orgin (0, 0)
in at most 100 steps is", success/n.mc, "and the number of steps are", success)
```

```
## The estimated probability of X that Ada comes back to the orgin (0, 0)
## in at most 100 steps is 0.03015075 and the number of steps are 6
```

Problem 4 - Gradient descent

1.

```
#euclidean norm = sqrt((x1)^2+(x2)^2+...+(xn)^2)
euclidean.norm <- function(x){
  total = 0
  for (i in 1:length(x)){
    total = total +(x[i])^2
  }
  sqrt(total)
}

n = 1
xk <- matrix()
GradientDescent <- function(A, b, h, x0, TOL, N.max){
  gradient.f <- function(x){A %*% x + b} #formula of gradient of f(x)
  x <-  x0 - h * gradient.f(x0)
  xk <- cbind(x)
  while (euclidean.norm(x-x0) > TOL || n <= N.max){
    x0 <- x
    x <-  x0 - h*gradient.f(x0)
    xk <- cbind(xk,x)
    n <- n+1
```

```
  }
  return (xk)
}
```

2.

```
A <- matrix(c(2,1,1,2), nrow=2)
b <- matrix(c(5,6))
x0 <- matrix(c(0,0))
GradientDescent(A, b, 0.1, x0, 1e-7, 100)
```

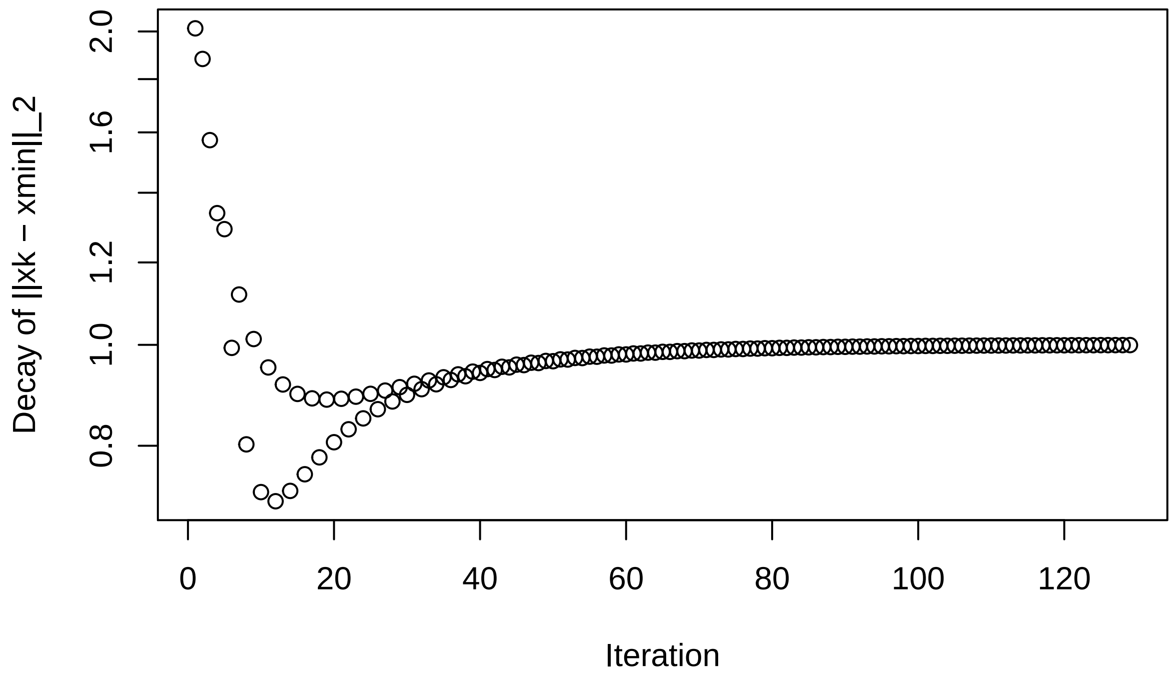
```
##       [,1]  [,2]   [,3]    [,4]     [,5]      [,6]      [,7]      [,8]      [,9]
## [1,] -0.5 -0.84 -1.069 -1.2212 -1.32045 -1.383364 -1.421499 -1.442879 -1.453062
## [2,] -0.6 -1.03 -1.340 -1.5651 -1.72996 -1.851923 -1.943202 -2.012412 -2.065641
##          [,10]     [,11]     [,12]     [,13]     [,14]     [,15]     [,16]
## [1,] -1.455885 -1.453988 -1.449172 -1.442664 -1.435283 -1.427575 -1.419892
## [2,] -2.107207 -2.140177 -2.166743 -2.188477 -2.206515 -2.221684 -2.234590
##          [,17]     [,18]     [,19]     [,20]     [,21]     [,22]     [,23]
## [1,] -1.412454 -1.405395 -1.398786 -1.392659 -1.387019 -1.381855 -1.377146
## [2,] -2.245683 -2.255301 -2.263701 -2.271082 -2.277600 -2.283378 -2.288517
##          [,24]     [,25]     [,26]     [,27]     [,28]     [,29]     [,30]
## [1,] -1.372865 -1.368982 -1.365467 -1.362288 -1.359416 -1.356825 -1.354488
## [2,] -2.293099 -2.297193 -2.300856 -2.304138 -2.307082 -2.309724 -2.312096
##          [,31]     [,32]     [,33]     [,34]     [,35]     [,36]     [,37]
## [1,] -1.352380 -1.350482 -1.348771 -1.347230 -1.345842 -1.344593 -1.343468
## [2,] -2.314228 -2.316145 -2.317868 -2.319417 -2.320811 -2.322064 -2.323192
##          [,38]     [,39]     [,40]     [,41]     [,42]     [,43]     [,44]
## [1,] -1.342455 -1.341543 -1.340723 -1.339984 -1.339319 -1.338721 -1.338182
## [2,] -2.324207 -2.325120 -2.325942 -2.326681 -2.327347 -2.327945 -2.328484
##          [,45]     [,46]     [,47]     [,48]     [,49]     [,50]     [,51]
## [1,] -1.337697 -1.337261 -1.336868 -1.336515 -1.336196 -1.335910 -1.335653
## [2,] -2.328969 -2.329406 -2.329798 -2.330152 -2.330470 -2.330756 -2.331014
##          [,52]     [,53]     [,54]     [,55]     [,56]     [,57]     [,58]
## [1,] -1.335421 -1.335212 -1.335024 -1.334855 -1.334703 -1.334566 -1.334443
## [2,] -2.331246 -2.331455 -2.331643 -2.331812 -2.331964 -2.332101 -2.332224
##          [,59]     [,60]     [,61]     [,62]     [,63]     [,64]     [,65]
## [1,] -1.334332 -1.334232 -1.334142 -1.334061 -1.333988 -1.333923 -1.333864
## [2,] -2.332335 -2.332435 -2.332525 -2.332606 -2.332678 -2.332744 -2.332803
##          [,66]     [,67]     [,68]     [,69]     [,70]     [,71]     [,72]
## [1,] -1.333811 -1.333763 -1.333720 -1.333681 -1.333647 -1.333615 -1.333587
## [2,] -2.332856 -2.332904 -2.332947 -2.332985 -2.333020 -2.333051 -2.333080
##          [,73]     [,74]     [,75]     [,76]     [,77]     [,78]     [,79]
## [1,] -1.333562 -1.333539 -1.333518 -1.333500 -1.333483 -1.333468 -1.333455
## [2,] -2.333105 -2.333128 -2.333148 -2.333167 -2.333183 -2.333198 -2.333212
##          [,80]     [,81]     [,82]     [,83]     [,84]     [,85]     [,86]
## [1,] -1.333443 -1.333432 -1.333422 -1.333413 -1.333405 -1.333398 -1.333391
## [2,] -2.333224 -2.333235 -2.333245 -2.333254 -2.333262 -2.333269 -2.333275
##          [,87]     [,88]     [,89]     [,90]     [,91]     [,92]     [,93]
## [1,] -1.333386 -1.333380 -1.333376 -1.333371 -1.333368 -1.333364 -1.333361
## [2,] -2.333281 -2.333286 -2.333291 -2.333295 -2.333299 -2.333302 -2.333306
##          [,94]     [,95]     [,96]     [,97]     [,98]     [,99]    [,100]
## [1,] -1.333358 -1.333356 -1.333354 -1.333352 -1.333350 -1.333348 -1.333347
## [2,] -2.333308 -2.333311 -2.333313 -2.333315 -2.333317 -2.333319 -2.333320
```

```
##          [,101]    [,102]    [,103]    [,104]    [,105]    [,106]    [,107]
## [1,] -1.333345 -1.333344 -1.333343 -1.333342 -1.333341 -1.333340 -1.333340
## [2,] -2.333321 -2.333323 -2.333324 -2.333325 -2.333325 -2.333326 -2.333327
##          [,108]    [,109]    [,110]    [,111]    [,112]    [,113]    [,114]
## [1,] -1.333339 -1.333338 -1.333338 -1.333338 -1.333337 -1.333337 -1.333336
## [2,] -2.333328 -2.333328 -2.333329 -2.333329 -2.333330 -2.333330 -2.333330
##          [,115]    [,116]    [,117]    [,118]    [,119]    [,120]    [,121]
## [1,] -1.333336 -1.333336 -1.333336 -1.333335 -1.333335 -1.333335 -1.333335
## [2,] -2.333331 -2.333331 -2.333331 -2.333331 -2.333332 -2.333332 -2.333332
##          [,122]    [,123]    [,124]    [,125]    [,126]    [,127]    [,128]
## [1,] -1.333335 -1.333335 -1.333334 -1.333334 -1.333334 -1.333334 -1.333334
## [2,] -2.333332 -2.333332 -2.333332 -2.333332 -2.333332 -2.333333 -2.333333
##          [,129]
## [1,] -1.333334
## [2,] -2.333333
```

```
#each column represents a vector of x_k
```

3.

```
xmin <- c(-4/3,-7/3)
xk <- GradientDescent(A, b, 0.1, x0, 1e-7, 100)
k = ncol(xk) #number of iterations
decay <- c()
for (i in 1:k){
  decay <- c(decay, euclidean.norm(xk[i] - xmin))
}

#convergence plot
plot(1:k, decay, log = "y", xlab = "Iteration", ylab = "Decay of ||xk - xmin||_2")
```



4.

```
m <- function(n){
  mat <- matrix(rep(0, n^2),nrow = n)
  for (i in 1:n){
    for (j in 1:n){
      if(i == j){mat[i,j] <- 2} #main diagonal
      if(i+1 == j || i-1 ==j){mat[i,j] <- -1} #first upper and lower diagonal
    }
  }

  return(mat)

}

vec <- function(n){
  c(rep(1,n))
}
m(10)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    2   -1    0    0    0    0    0    0    0     0
##  [2,]   -1    2   -1    0    0    0    0    0    0     0
##  [3,]    0   -1    2   -1    0    0    0    0    0     0
##  [4,]    0    0   -1    2   -1    0    0    0    0     0
```

```
##  [5,]    0    0    0   -1    2   -1    0    0    0     0
##  [6,]    0    0    0    0   -1    2   -1    0    0     0
##  [7,]    0    0    0    0    0   -1    2   -1    0     0
##  [8,]    0    0    0    0    0    0   -1    2   -1     0
##  [9,]    0    0    0    0    0    0    0   -1    2    -1
## [10,]    0    0    0    0    0    0    0    0   -1     2
```

```r
vec(10)
```

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

5.

```r
x0 <- matrix(c(rep(0,10)))
M <- m(10)
v <- vec(10)
xList <- GradientDescent(M, v, 0.5, x0, 1e-7, 100)
k <- ncol(xList) #number of iterations
x.min <- xList[,k] #last iteration
x.min
```

```
##  [1]  -5.000000  -8.999999 -11.999999 -13.999999 -14.999999 -14.999999
##  [7] -13.999999 -11.999999  -8.999999  -5.000000
```

```r
f <- function(x) {1/2*t(x)%*%M%*%x + t(v)%*%x + 1}
f(x.min)
```

```
##      [,1]
## [1,]  -54
```