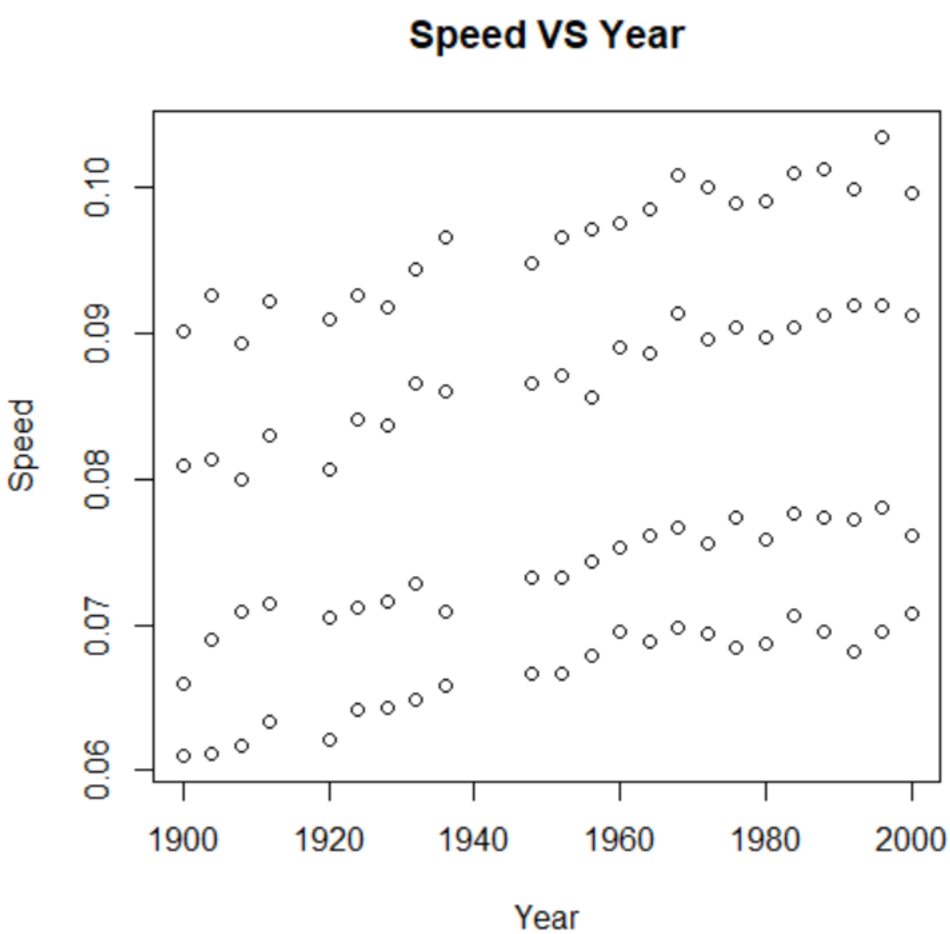
# Final project

## problem 1 Olympic game

```
> #1)
> getwd()
[1] "C:/Users/10436/OneDrive/Documents"
> #change the working directory to where the file is located
> setwd("C:/Users/10436/OneDrive/桌面")
> #read the table and store in a data frame called speed.data
> speed.data<-read.table("speed.txt",header=TRUE)
> #print first five row of the data
> 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
> #create a new column
> Speed<-100*speed.data$Distance.100/speed.data$Time
> speed.data<-cbind(speed.data,Speed)
> 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
>
>#give it an order according to year
> speed.data<-speed.data[order(speed.data$Year),]
> speed.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
> Year<-speed.data$Year
> Speed<-speed.data$Speed
> speed.year<-data.frame(Year,Speed)
#plot it
> plot(Year,Speed,main=" Speed VS Year")
```

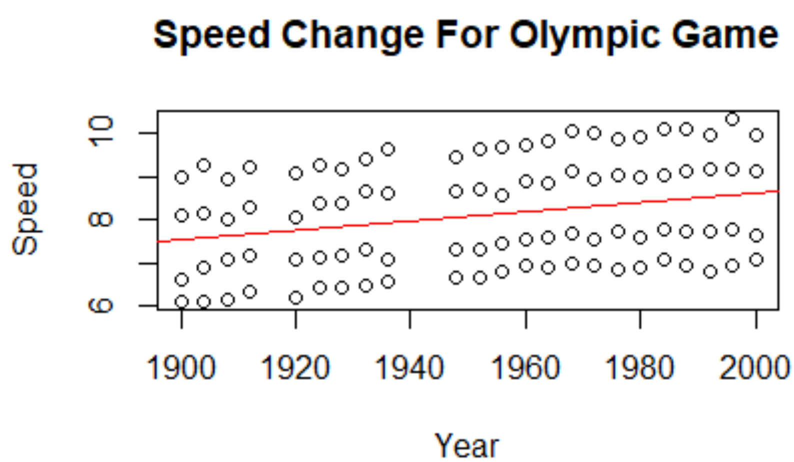



>

```
>
> #i observed that as year increases, the speed increases as well.
>
>
>
>
```

**> #5**

```
>#plot a best fit line
> plot(speed.year,main="Speed Change For Olympic Game")
> abline(lm(speed.year$Speed~speed.year$Year),col="red")
>
```



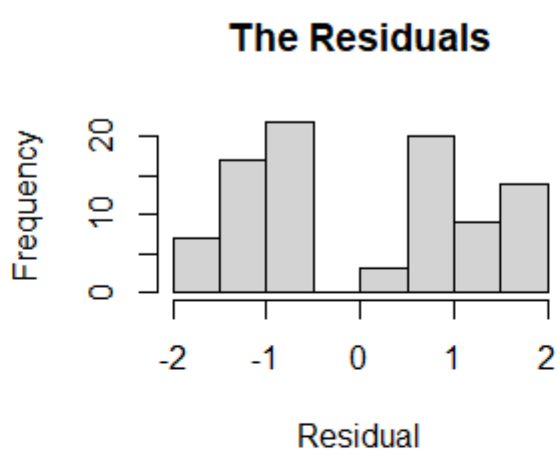

>

**> #6**

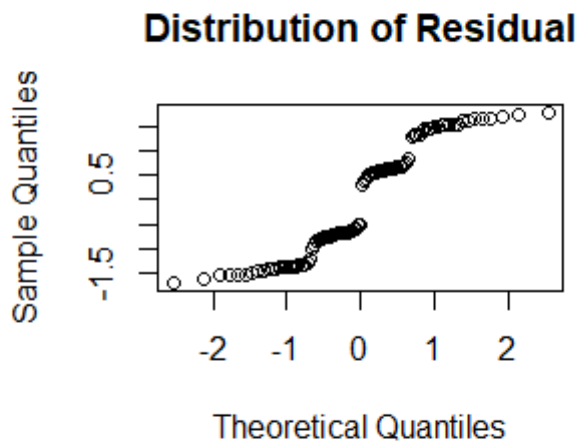
```
> lm(speed.year$Speed~speed.year$Year)

Call:
lm(formula = speed.year$Speed ~ speed.year$Year)

Coefficients:
    (Intercept)  speed.year$Year
      -13.01660         0.01082
#the value of p and m
> p<- -13.01660
> m<-0.01082
>
```

**> #7**

```
#function for predicted average
> prediction<-function(y){
+
+   return(((100/y)-p)/m)
+ }
> prediction(7)
[1] 2523.319
>
> #difference between the actual average and predicted average
> nrow(speed.data)
[1] 92
> residual<-c()
>
> for(i in 1:92){
+   #the actual speed-the speed by best fitting line
+   residual[i]<-speed.data$Speed[i]-speed.data$Year[i]*0.01082+13.01660
+ }
> #plot it
> hist(residual,xlab="Residual",main="The Residuals")
```




```
#create a qqnorm
> qqnorm(residual,main="Distribution of Residual")
```




```
> #distribution does not follow normal distribution
>
>
>
> qr.solve(matrix(c(speed.year$Year,rep(1,92)),nrow=92),speed.year$Speed)
[1]   0.01081622 -13.01660062
> #We got the same result
```

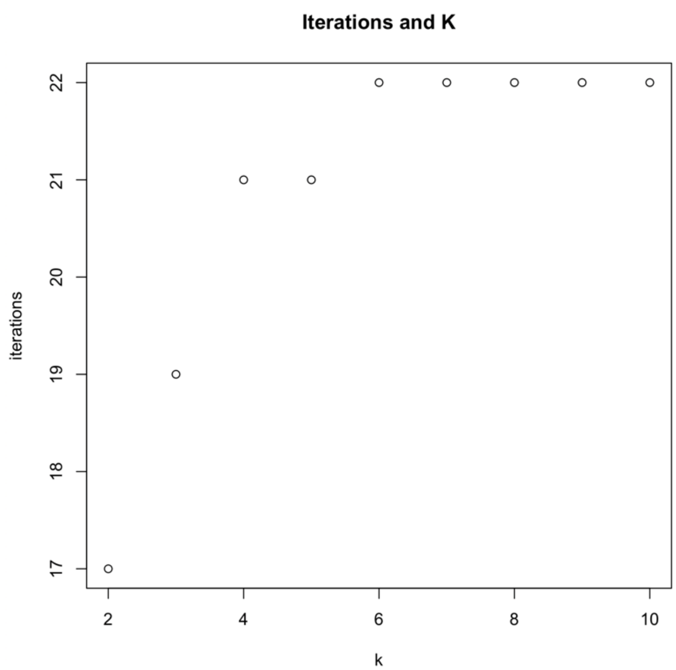
## question 2 Newton

**> #1**

```
> EvalPoly<-function(c,x){
+   #the number of repetition
+   k<-length(c)-1
+   sum<-0
+   #add each segments to sum to get a total value
+   for(i in 0:k){
```

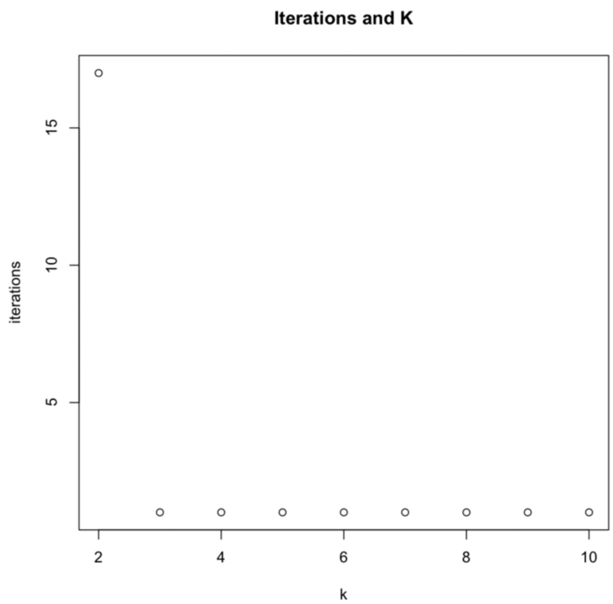
```
+    sum<-sum+c[i+1]*x^i}
+  return(sum)
+ }
> #2
> #input the vector c from integer to the coefficient of the highest power of x
> EvalPoly(c(1,-1.7,0,3.5),13.4)
[1] 8399.584
>
>
> #3
> polyDerEval<-function(c,x){
+  #the number of repetition
+  k<-length(c)-1
+  sum<-0
+  #add each segments to sum to get a total value
+  for(i in 1:k){
+    sum<-sum+i*c[i+1]*x^(i-1)}
+  return(sum)
+ }
>
> #4
> polyDerEval(c(1,-1.7,0,3.5),13.4)
[1] 1883.68
>
> #5
> NewtonPoly<-function(c,x0,TOL){
+  #newtons's method of root finding
+  x <- x0
+  vect_x<-c(x)
+  f<-EvalPoly(c,x)
+  while (abs(f) > TOL) {
+    f.prime <-polyDerEval(c,x)
+    x <- x - f / f.prime
+    f <- EvalPoly(c,x)
+    #collect value of x
+    vect_x<-c(vect_x,x)
+  }
+  return (vect_x)
+ }
> #6
> #check the value of the polynominal
> NewtonPoly(c(2.3,-7.1,0,1),-1,6*10^-10)
[1] -1.00000000  1.04878049 -0.00189262  0.32394415  0.32895377
[6]  0.32895739
```

```
>
>
>
> #7
> #create a iteration vector
> iteration<-c()
> #K=2
> k2 <- NewtonPoly(c(0,0,1), 1, 1e-10)
> #k2
> iteration[1]<-length(k2)
> #K=3
> k3 <- NewtonPoly(c(0,0,0,1), 1, 1e-10)
> iteration[2]<-length(k3)
> #K=4
> k4<- NewtonPoly(c(0,0,0,0,1), 1, 1e-10)
> iteration[3]<-length(k4)
> #K=5
> k5<- NewtonPoly(c(0,0,0,0,0,1), 1, 1e-10)
> iteration[4]<-length(k5)
> #K=6
> k6<- NewtonPoly(c(0,0,0,0,0,0,1), 1, 1e-10)
> iteration[5]<-length(k6)
> #K=7
> k7<- NewtonPoly(c(0,0,0,0,0,0,0,1), 1, 1e-10)
> iteration[6]<-length(k7)
> #K=8
> k8<- NewtonPoly(c(0,0,0,0,0,0,0,0,1), 1, 1e-10)
> iteration[7]<-length(k8)
> #K=9
> k9 <- NewtonPoly(c(0,0,0,0,0,0,0,0,0,1), 1, 1e-10)
> iteration[8]<-length(k9)
> #K=10
> k10<- NewtonPoly(c(0,0,0,0,0,0,0,0,0,0,1), 1, 1e-10)
> iteration[9]<-length(k10)
> #minus the first time of the loop,since it is not consider as iteration
> iteration<- iteration-rep(1,9)
> #plot it
> plot(seq(2,10,1), iteration,main="Iterations and K",xlab="k",ylab="iterations")
```




```
> #8
> #second derivative
> PolyDer2Eval <- function(c, x){
+
+   Pxx <- c[length(c)]*(length(c)-1)*(length(c)-2)*x^(length(c)-3)
+   for (i in (length(c)-1):3)
+     Pxx <- Pxx + c[i]*(i-1)*(i-2)*x^(i-3)
+   return(Pxx)
+ }
> #check the value of polynominal
> PolyDer2Eval(c(1,-1.7,0,3.5),13.4)
[1] 281.4
>
> #9
> #modified version
> ModifiedNewtonPoly<- function(c, x0,TOL){
+   x <- x0
+   vect_x<-c(x)
+   k<-abs(EvalPoly(c,x)) > TOL
```

```
+   while (abs(EvalPoly(c,x)) > TOL&& length( vect_x) <= 1000){
+
+     x <- x - (Evalpoly(c, x) * polyDerEval(c, x)) /(polyDerEval(c, x)^2 - Evalpoly(c, x) *
PolyDer2Eval(c, x))
+     vect_x<-c(vect_x,x)
+   }
+   return ( vect_x)
+ }
> #check the values
> ModifiedNewtonPoly(c(2.3,-7.1,0,1), -1, 1e-10)
[1] -1.0000000 -0.4875763  0.1416393  0.3300980  0.3289576  0.3289574
>
> #10
>#simulate k=2 to k=10
> k2<- ModifiedNewtonPoly(c(0,0,1), 1, 1e-10)
> iteration[1]<-length(k2)
> k3<- ModifiedNewtonPoly(c(0,0,0,1), 1, 1e-10)
> iteration[2]<-length(k3)
> k4<- ModifiedNewtonPoly(c(0,0,0,0,1), 1, 1e-10)
> iteration[3]<-length(k4)
> k5<- ModifiedNewtonPoly(c(0,0,0,0,0,1), 1, 1e-10)
> iteration[4]<-length(k5)
> k6<- ModifiedNewtonPoly(c(0,0,0,0,0,0,1), 1, 1e-10)
> iteration[5]<-length(k6)
> #K=7
> k7<- ModifiedNewtonPoly(c(0,0,0,0,0,0,0,1), 1, 1e-10)
> iteration[6]<-length(k7)
> #K=8
> k8<- ModifiedNewtonPoly(c(0,0,0,0,0,0,0,0,1), 1, 1e-10)
> iteration[7]<-length(k8)
> #K=9
> k9 <- ModifiedNewtonPoly(c(0,0,0,0,0,0,0,0,0,1), 1, 1e-10)
>
> iteration[8]<-length(k9)
> #K=10
> k10<- ModifiedNewtonPoly(c(0,0,0,0,0,0,0,0,0,0,1), 1, 1e-10)
> iteration[9]<-length(k10)
> #minus the first value which does not count as iteration
> iteration<- iteration-rep(1,9)
>
> plot(seq(2,10,1), iteration,main="Iterations and K",xlab="k",ylab="iterations")
```




# Question 3 Ada Walk

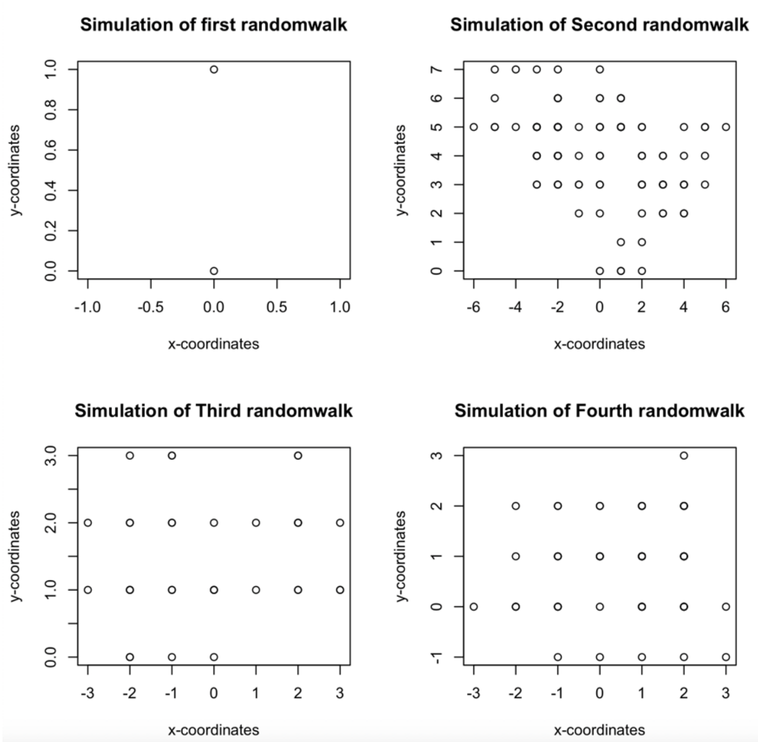
**> #1**

```
> AdaWalk<-function(){
+  #create components of (x,y) for the location of the walk, and see them as vectors
+  x<-c()
```

```
+  y<-c()
+  #set the orginal to (0,0). which is the first element of each vector
+  x[1]<-0
+  y[1]<-0
+
+  for(i in 2:101){
+   #simulate a randam number
+   a_walk<-runif(1)
+
+   #judge the position of a_walk
+   #on the right, one step
+   if(a_walk<0.25){
+     x[i]<-x[i-1]+1
+     y[i]<-y[i-1]
+   }
+
+   #on the left, one step
+   if(a_walk>=0.25 & a_walk<0.5){
+     x[i]<-x[i-1]-1
+     y[i]<-y[i-1]
+   }
+   #up, one step
+   if(a_walk>=0.5 & a_walk<0.75){
+     x[i]<-x[i-1]
+     y[i]<-y[i-1]+1
+     }
+   #down, one step
+   if(a_walk>=0.75){
+     x[i]<-x[i-1]
+     y[i]<-y[i-1]-1
+   }
+
+  #the simulation stops when it goes to original (0,0)
+   c<-x[i]==0 & y[i]==0
+   if(c) break
+   #combine 2 vectors to a matrix
+   randomwalk<-rbind(x,y)
+
+
+   }
+  #return the matrix
+  return (randomwalk)
+  }
```

**> #2**

```
> #simulate 4 sets of randomwalk
> randomwalk1<-AdaWalk()
> randomwalk2<-AdaWalk()
> randomwalk3<-AdaWalk()
> randomwalk4<-AdaWalk()
>
> #plot them
> par(mfrow=c(2,2))
> plot(randomwalk1[1,],randomwalk1[2,],xlab="x-coordinates",ylab="y-coordinates",main="Simulation of first randomwalk")
> plot(randomwalk2[1,],randomwalk2[2,],xlab="x-coordinates",ylab="y-coordinates",main="Simulation of Second randomwalk")
> plot(randomwalk3[1,],randomwalk3[2,],xlab="x-coordinates",ylab="y-coordinates",main="Simulation of Third randomwalk")
> plot(randomwalk4[1,],randomwalk4[2,],xlab="x-coordinates",ylab="y-coordinates",main="Simulation of Fourth randomwalk")
```



```
> # 3
> #count the number of eligible number of simulation to (0,0)
> num<-0
> for(i in 1:1000){
+  #simulate the randomwalk
+  randomwalk<-AdaWalk()
+  #check if it is more than 100
+  if(length(randomwalk[1,])>=100)
+    #count the total number of eligible simulation
+    num<-num+1
+
+ }
> #probability
>  num/1000
[1] 0.428
>
>  #4
>  #count the sum of the randomwalk that takes maximum 100 to back to (0,0)
>  sum<-0
>  for(i in 1:1000){
+   rw<-AdaWalk()
+   #check if reach the max 100
+   if(length(rw[1,])<=100)
+     sum=sum+length(rw[1,])
+
+  }
>  #calculate the average
>   sum/1000
[1] 7.092
>
```