Assignment Computer Lab 1

Greeshma Jeev Koothuparambil(greko370), Sangeeth Sankunny Menon(sansa237)

2023-11-08

Question 1: Maximization of a likelihood function in one variable

We have independent data x1, . . . , xn from a Cauchy-distribution with unknown location parameter θ and known scale parameter 1. The log likelihood function is

$$-n\log(\pi) - \sum_{i=1}^{n} \log(1 + (x_i - \theta)^2),$$

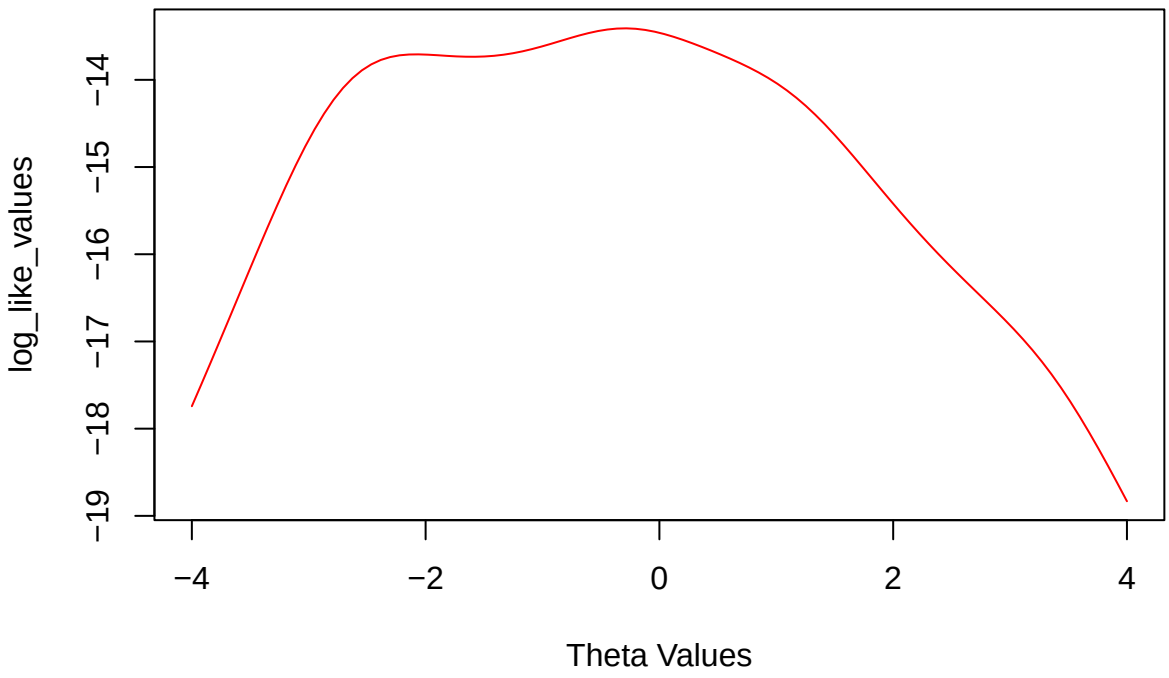
and it's derivative with respect to θ is

$$\sum_{i=1}^{n} \frac{2(x_i - \theta)}{1 + (x_i - \theta)^2}$$

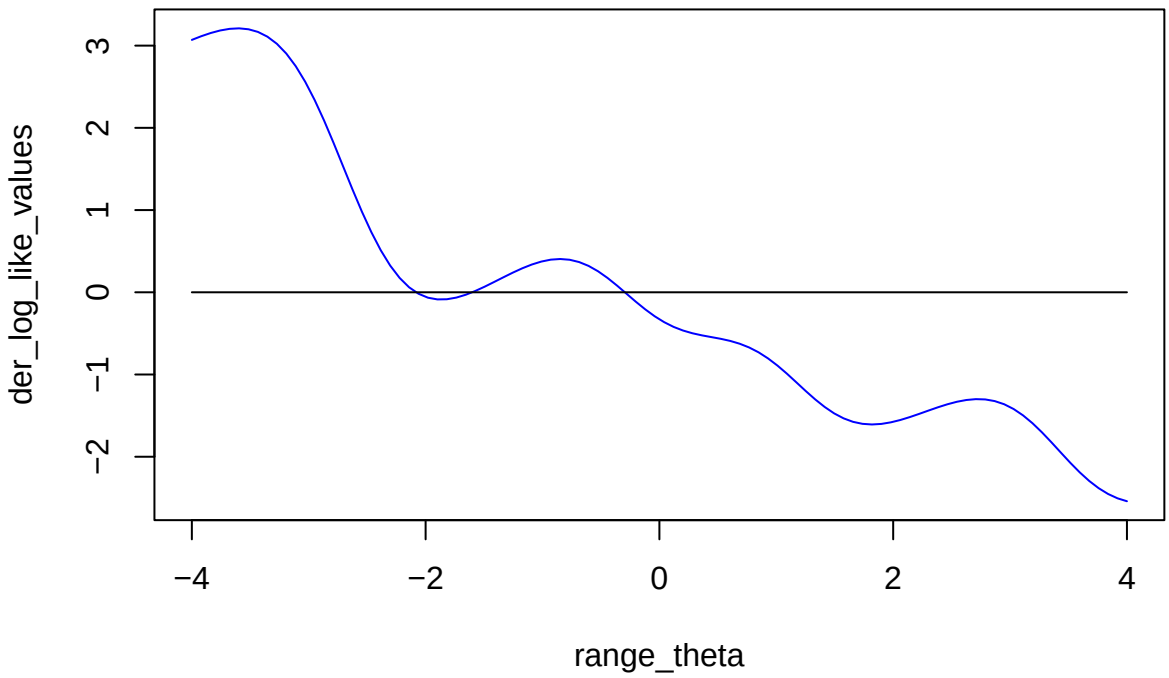
Data of size $n = 5$ is given: $x = (-2.8,\ 3.4,\ 1.2, -0.3, -2.6)$.

a. Plot the log likelihood function for the given data in the range from -4 to 4. Plot the derivative in the same range and check visually how often the derivative is equal to 0.

```
#1.2
library(ggplot2)
x <-c(-2.8, 3.4, 1.2,-0.3,-2.6)


log_like <-function(x,theta){
  value<-0
  for (i in 1:length(x)) {
    value= sum(log(1+(x-theta)^2))
  }
  return(-length(x)*log(pi)-value)
}

der_log_like<-function(x,theta){
  der_value<-0
  for (i in 1:length(x)){
    der_value<-sum((2*(x-theta))/(1+(x-theta)^2))
  }
  return(der_value)
}

range_theta<-seq(-4,4,length.out=100)
log_like_values<-sapply(range_theta, function(ta) log_like(x,ta))
der_log_like_values<-sapply(range_theta,function(ta) der_log_like(x,ta))
plot(range_theta,log_like_values,type="l",xlab="Theta Values",col="red")
```




```
plot(range_theta,der_log_like_values,type="l",col="blue")
lines(range_theta,rep(0,100),col="black")
```



b. Choose one of the following methods: bisection, secant, or Newton-Raphson. Write your own codeto optimize with the chosen method. If you have chosen Newton-Raphson, describe how you derivedthe second derivative.

```
#1.b
range_theta <- round(range_theta, 3)
der_log_like_values <- round(der_log_like_values, 3)
derdf <- data.frame(range_theta, der_log_like_values)

secantopt <- function(xt,xpt) {
  xppt <- xpt
  print("Optimizing theta value:")
  while (xt!=xpt&&xppt) {
    txt <- xt
    gxt <- der_log_like(x, xt)
    gxpt <- der_log_like(x, xpt)
    xt <- xt - (gxt*((xt-xpt)/(gxt-gxpt)))
    xt <- round(xt,4)
    xppt <- xpt
    xpt <- txt
    print(xt)
    if(xt < -4|xt>4){
      print(paste0("Theta value ", xt, " is out of bound. Exiting..."))
      break
    }

  }
```

```
  return(xt)
}
```

c. Choose suitable starting values based on your plots to identify all local maxima of the likelihoodfunction. Note that you need pairs of interval boundaries for the bisection or pairs of starting values for secant. Are there any (pairs of) starting values which do not lead to a local maximum? Decide which is the global maximum based on programming results.

```
#1.c
xmt <- 0.768
x0 <- 0.444
firstlocal <- secantopt(x0, xmt)
```

```
## [1] "Optimizing theta value:"
## [1] -0.9511
## [1] -0.3673
## [1] -0.1997
## [1] -0.2956
## [1] -0.2952
## [1] -0.2952
```

```
firstmax <- log_like(x, firstlocal)

xmt <- -2.4
x0 <- -2.2
secondlocal <- secantopt(x0, xmt)
```

```
## [1] "Optimizing theta value:"
## [1] -2.1323
## [1] -2.0939
## [1] -2.0836
## [1] -2.0822
## [1] -2.0821
## [1] -2.0821
```

```
secondmax <- log_like(x, secondlocal)

xmt <- -1.6
x0 <- -1.8
thirdlocal <- secantopt(x0, xmt)
```

```
## [1] "Optimizing theta value:"
## [1] -1.6025
## [1] -1.6015
## [1] -1.6018
## [1] -1.6018
```

```
thirdmax <- log_like(x, thirdlocal)


print(paste0("The first maximum is ", firstmax, " with theta value ",firstlocal))
```

```
## [1] "The first maximum is -13.4094200567535 with theta value -0.2952"
```

```
print(paste0("The second maximum is ", secondmax, " with theta value ",secondlocal))
```

```
## [1] "The second maximum is -13.7077039269938 with theta value -2.0821"
```

```
print(paste0("The first minimum is ", thirdmax, " with theta value ",thirdlocal))
```

```
## [1] "The first minimum is -13.735720175941 with theta value -1.6018"
```

```
print(paste0("Therefore the global maximum is :", firstmax))
```

```
## [1] "Therefore the global maximum is :-13.4094200567535"
```

```
xmt <-2.6
x0 <- 2.8
inflocal <- secantopt(x0, xmt)
```

```
## [1] "Optimizing theta value:"
## [1] 27.3862
## [1] "Theta value 27.3862 is out of bound. Exiting..."
```

d. Assume now that you are in a situation where you cannot choose starting values based on a plot since the program should run automatised. How could you modify your program to identify the global maximum even in the presence of other local optima?

```
#1.d
secantauto <- function(mintheta, maxtheta) {
  tempmin <- mintheta
  tempmax <- mintheta+0.1
  localpeak <- c()
  while(mintheta<maxtheta){
    xt <- round(runif(1,tempmin, tempmax),3)
    xpt <- round(runif(1,xt, tempmax),3)
    halflocal <- secantopt(xt, xpt)
    localpeak <- append(localpeak, halflocal)
    xpt <- round(runif(1,tempmin, xt),3)
    secondhalflocal <- secantopt(xt, xpt)
    localpeak <- append(localpeak, secondhalflocal)
    mintheta <- tempmax
    tempmin <- tempmax
    tempmax <- tempmax+2
  }
  return(localpeak)
}


auto <- secantauto(-4,4)
```

```
## [1] "Optimizing theta value:"
## [1] -9.305
## [1] "Theta value -9.305 is out of bound. Exiting..."
## [1] "Optimizing theta value:"
## [1] -9.2587
## [1] "Theta value -9.2587 is out of bound. Exiting..."
## [1] "Optimizing theta value:"
## [1] -11.9214
## [1] "Theta value -11.9214 is out of bound. Exiting..."
## [1] "Optimizing theta value:"
## [1] -10.6305
## [1] "Theta value -10.6305 is out of bound. Exiting..."
## [1] "Optimizing theta value:"
## [1] -0.2946
## [1] -0.2958
## [1] -0.2952
## [1] -0.2952
## [1] "Optimizing theta value:"
## [1] -1.602
## [1] -1.6018
## [1] -1.6018
## [1] "Optimizing theta value:"
## [1] -0.1458
## [1] -0.4651
## [1] -0.2982
## [1] -0.2951
## [1] -0.2952
## [1] -0.2952
## [1] "Optimizing theta value:"
## [1] -1.4819
## [1] -1.2406
## [1] -1.5909
## [1] -1.6002
## [1] -1.6018
## [1] -1.6018
## [1] "Optimizing theta value:"
## [1] 2.0523
## [1] 128.5893
## [1] "Theta value 128.5893 is out of bound. Exiting..."
## [1] "Optimizing theta value:"
## [1] 1.4119
## [1] -14.6906
## [1] "Theta value -14.6906 is out of bound. Exiting..."
```

Question 2: Computer arithmetics (variance)

A known formula for estimating the variance based on a vector of n observations is

$$Var(\overline{x}) = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right)$$

a. Write your own R function, myvar, to estimate the variance in this way.

```
#2.a
myvar <- function(x){
  n <- length(x)
  xsqsum <- sum(x^2)
  xsum <- sum(x)
  vari <- (1/(n-1))*(xsqsum - ((1/n)*((xsum)^2)))
  return(vari)
}

vectorval <- c(4, 6, 7, 3, 16, 14, 8, 10, 17, 2)
variance <- myvar(vectorval)
```

b. Generate a vector x = (x1, . . . , x10000) with 10000 random numbers with mean 10^8 and variance 1.

```
#2.b
x <- rnorm(10000, 10^8, 1)
```

c. For each subset Xi = {x1, . . . , xi}, i = 1, . . . , 10000 compute the difference

$$Y_i = myvar(X_i) - var(X_i)$$

, where

$$var(X_i)$$

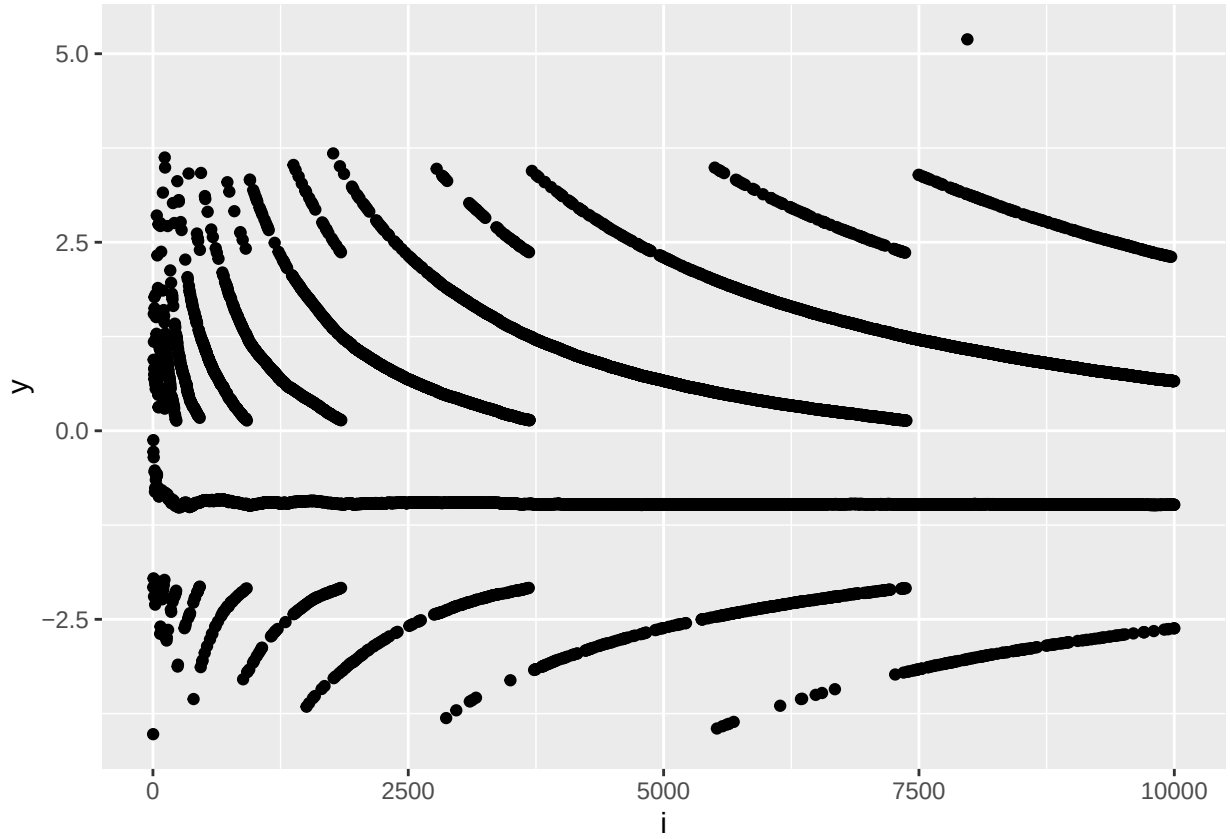
is the standard variance estimation function in R. Plot the dependence

$$Y_i$$

on i.Draw conclusions from this plot. How well does your function work? Can you explain the behaviour?

```
#2.c
y <-c()
for (i in 2:10000) {
  y[i] <- myvar(x[1:i]) - var(x[1:i])
}
i<- c(1:10000)

df <- data.frame(i, y)
ggplot(df)+ geom_point(mapping = aes(x=i, y=y))
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```




```
summary(df)
```

```
##        i                y
##  Min.   :    1   Min.   :-4.0226
##  1st Qu.: 2501   1st Qu.:-0.9756
##  Median : 5000   Median : 0.2915
##  Mean   : 5000   Mean   : 0.0211
##  3rd Qu.: 7500   3rd Qu.: 0.9244
##  Max.   :10000   Max.   : 5.1905
##                  NA's   :1
```

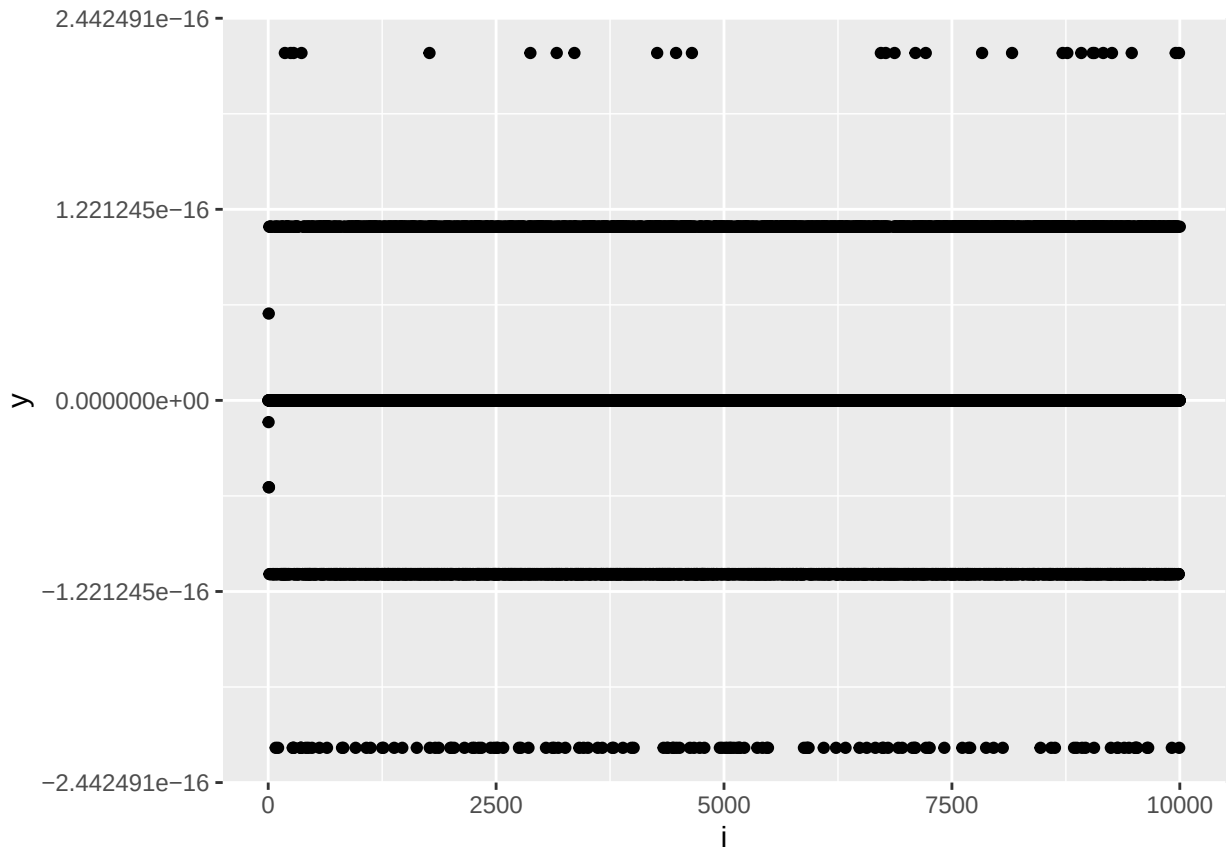
The differences in the customised variance and r variance varies from -4.6 to 5. Almost none of the values set at zero. The graph shows concurrent exponential parallel curves happening for different ranges of values. It is evident that the customised variance is not efficient enough to be on par with R variance.

d. How can you better implement a variance estimator? Find and implement a formula that will givethe same results as var()?

```
#2.d

newmyvar <- function(x){
  n <- length(x)
  xmean <- mean(x)
  vari <- (1/(n-1))*(sum((x-xmean)^2))
  return(vari)
```

```
}

y <-c()
for (i in 2:10000) {
  y[i] <- newmyvar(x[1:i]) - var(x[1:i])
}
i<- c(1:10000)

dfnew <- data.frame(i, y)
ggplot(dfnew)+ geom_point(mapping = aes(x=i, y=y))
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```




```
summary(df)
```

```
##        i                y
##  Min.   :    1   Min.   :-4.0226
##  1st Qu.: 2501   1st Qu.:-0.9756
##  Median : 5000   Median : 0.2915
##  Mean   : 5000   Mean   : 0.0211
##  3rd Qu.: 7500   3rd Qu.: 0.9244
##  Max.   :10000   Max.   : 5.1905
##                  NA's   :1
```

Since we are producing data as a sample of normal distribution function the variance formula can be improved by the following calculation method:

$$Var(\overline{x}) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2$$

where μ is the sample mean.

Appendix

Question 1

```
library(ggplot2)
x <-c(-2.8, 3.4, 1.2,-0.3,-2.6)


log_like <-function(x,theta){
  value<-0
  for (i in 1:length(x)) {
    value= sum(log(1+(x-theta)^2))
  }
  return(-length(x)*log(pi)-value)
}

der_log_like<-function(x,theta){
  der_value<-0
  for (i in 1:length(x)){
    der_value<-sum((2*(x-theta))/(1+(x-theta)^2))
  }
  return(der_value)
}


range_theta<-seq(-4,4,length.out=100)
log_like_values<-sapply(range_theta, function(ta) log_like(x,ta))
der_log_like_values<-sapply(range_theta,function(ta) der_log_like(x,ta))
plot(range_theta,log_like_values,type="l",xlab="Theta Values",col="red")

plot(range_theta,der_log_like_values,type="l",col="blue")
lines(range_theta,rep(0,100),col="black")


#1.b

range_theta <- round(range_theta, 3)
der_log_like_values <- round(der_log_like_values, 3)
derdf <- data.frame(range_theta, der_log_like_values)

secantopt <- function(xt,xpt) {
  xppt <- xpt
  print("Optimizing theta value:")
```

```
  while (xt!=xpt&&xppt) {
    txt <- xt
    gxt <- der_log_like(x, xt)
    gxpt <- der_log_like(x, xpt)
    xt <- xt - (gxt*((xt-xpt)/(gxt-gxpt)))
    xt <- round(xt,4)
    xppt <- xpt
    xpt <- txt
    print(xt)
    if(xt < -4|xt>4){
      print(paste0("Theta value ", xt, " is out of bound. Exiting..."))
      break
    }

  }
  return(xt)
}

xmt <- 0.768
x0 <- 0.444
firstlocal <- secantopt(x0, xmt)
firstmax <- log_like(x, firstlocal)

xmt <- -2.4
x0 <- -2.2
secondlocal <- secantopt(x0, xmt)
secondmax <- log_like(x, secondlocal)

xmt <- -1.6
x0 <- -1.8
thirdlocal <- secantopt(x0, xmt)
thirdmax <- log_like(x, thirdlocal)


print(paste0("The first maximum is ", firstmax, " with theta value ",firstlocal))
print(paste0("The second maximum is ", secondmax, " with theta value ",secondlocal))
print(paste0("The first minimum is ", thirdmax, " with theta value ",thirdlocal))
print(paste0("Therefore the global maximum is :", firstmax))

xmt <-2.6
x0 <- 2.8
inflocal <- secantopt(x0, xmt)


secantauto <- function(mintheta, maxtheta) {
  tempmin <- mintheta
  tempmax <- mintheta+0.1
  localpeak <- c()
  while(mintheta<maxtheta){
    xt <- round(runif(1,tempmin, tempmax),3)
    xpt <- round(runif(1,xt, tempmax),3)
    halflocal <- secantopt(xt, xpt)
    localpeak <- append(localpeak, halflocal)
```

```
    xpt <- round(runif(1,tempmin, xt),3)
    secondhalflocal <- secantopt(xt, xpt)
    localpeak <- append(localpeak, secondhalflocal)
    mintheta <- tempmax
    tempmin <- tempmax
    tempmax <- tempmax+2
  }
  return(localpeak)
}


auto <- secantauto(-4,4)
```

Question 2

```
#2.a
myvar <- function(x){
  n <- length(x)
  xsqsum <- sum(x^2)
  xsum <- sum(x)
  vari <- (1/(n-1))*(xsqsum - ((1/n)*((xsum)^2)))
  return(vari)
}

vectorval <- c(4, 6, 7, 3, 16, 14, 8, 10, 17, 2)
variance <- myvar(vectorval)


#2.b
x <- rnorm(10000, 10^8, 1)

#2.3
y <-c()
for (i in 2:10000) {
  y[i] <- myvar(x[1:i]) - var(x[1:i])
}
i<- c(1:10000)

df <- data.frame(i, y)
ggplot(df)+ geom_point(mapping = aes(x=i, y=y))
summary(df)
#2.4

newmyvar <- function(x){
  n <- length(x)
  xmean <- mean(x)
  vari <- (1/(n-1))*(sum((x-xmean)^2))
  return(vari)
}

y <-c()
for (i in 2:10000) {
  y[i] <- newmyvar(x[1:i]) - var(x[1:i])
```

```
}
i<- c(1:10000)

dfnew <- data.frame(i, y)
ggplot(dfnew)+ geom_point(mapping = aes(x=i, y=y))
summary(df)
```
