STAT3701HW3

Mingming Xu

7/22/2019

Problem 1

i.

```
sim.beta.hat=function(x.mat,beta,sigma,reps){

n=nrow(x.mat)
p=ncol(x.mat)
beta.hat=matrix(NA,reps,p)
for(r in 1:reps){
##generate the error
epsilon.vetc=sigma*(rexp(n,1)-1)
##compute the Y
y.vec=x.mat %*% beta+epsilon.vetc
## compute beta hat
beta.hat[r,]=qr.coef(qr=qr(x.mat),y=y.vec)

}
  return(beta.hat)

}
```

ii.

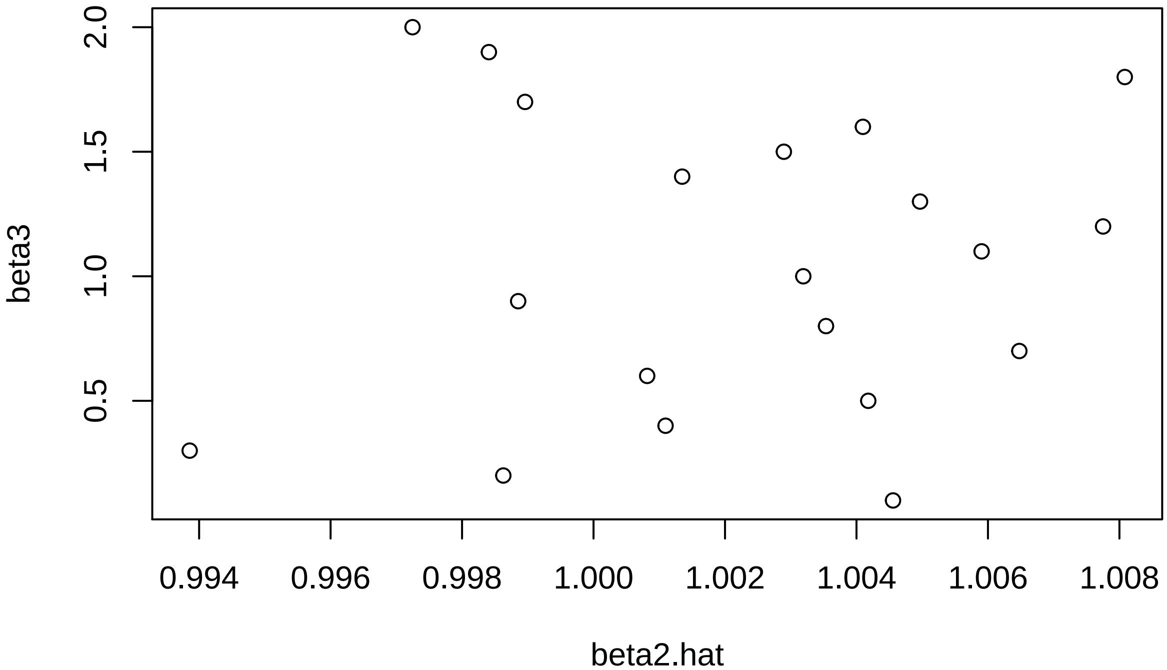
```
## estimate of beta2
x.mat.1=read.csv("X.mat.1.csv")
x.mat1=as.matrix(x.mat.1)
beta.1=c(-1,1,0,1)
sigma=1.5
reps=5e3
beta.hat=apply(sim.beta.hat(x.mat1,beta.1,sigma,reps),2,mean)
##beta 2
beta.hat[2]  ## the result is so closed to the true value of beta2
```

```
## [1] 1.001365
```

iii.

```
beta3=seq(0.1,2,by=0.1)
beta2.hat=numeric(length = length(beta3))
for(i in 1:length(beta3)){
  beta.2=c(-1,1,beta3[i],0)
  beta.hat=apply(sim.beta.hat(x.mat1,beta.2,sigma,reps),2,mean)
  beta2.hat[i]=beta.hat[2]
}

plot(beta2.hat,beta3)
```




```
## By this plot, we could see thaht the most of points are around the true value of beta2.
```

Problem 2

i

```
x.df.2=read.csv("X.df.2.csv")
x.df2=as.matrix(x.df.2)
y=x.df2[,1]
x.mat=x.df2[,2:4]
qr.solve(crossprod(x.mat)) %*% crossprod(x.mat, y)
```

```
##                          [,1]
## Intercept           5.1073949
## Father.Height       0.7661411
## Grandfather.height 0.1632550
```

The linear regressio model is $Son\hat{H}eight = 5.11 + 0.77 * Father.Height + 0.16 * Grandfather.height$.

When father's height and grandfather's height are 0,the expected son's height is on average 5.11 inches.

When father's height increases one inch and grandfather's height is fixed , on average, the son's heights is expected to increase 0.77 inches.

When grandfather's height increases one inch and father's height is fixed , n average , ths son's height is expected to increase 0.16 inches.

ii.

```
beta.hat=qr.coef(qr=qr(x=x.mat),y=y)
beta2.hat=beta.hat[2]
n=nrow(x.mat)
p=ncol(x.mat)
alpha=0.05
tper=qt(1-alpha/2,n-p)
xtxin=qr.solve(crossprod(x.mat))
s=sqrt(sum((y-x.mat %*% beta.hat)^2)/(n-p))
```

```r
moe=tper*s*sqrt(xtxin[2,2])
left=beta2.hat-moe
right=beta2.hat+moe
ci=c(left,right)
ci
```

```
## Father.Height Father.Height
##     0.4901989     1.0420832
```

Yes. Because the $\hat{\beta_{FH}}$ is in this interval.

iii.

The random test statistic is $T = \frac{\hat{\beta_{GH}}}{S\sqrt{[X'X]^{-1}_{33}}}$

```r
sqrt.form=sqrt(xtxin[3,3])
t.test=beta.hat[3]/(s*sqrt.form)
pval=2*pt(-abs(t.test),n-p)
pval
```

```
## Grandfather.height
##          0.2301573
```

Because the p value is greater than 0.05, we fail to reject the null hypothesis.

iv.

```r
xnew=c(1,67,66)
point.est=xnew %*% beta.hat
## 96% CI
xtxin=qr.solve(crossprod(x.mat))
sqform=sqrt(t(xnew) %*% xtxin %*% xnew)
alpha=0.04
t.per=pt(1-alpha/2,n-p)
m=t.per*s*sqform
ci=c(point.est-m ,point.est+m)
ci
```

```
## [1] 66.94346 67.48390
```

```r
## 96% PI
sqform.pi=sqrt(t(xnew) %*% xtxin %*% xnew +1)
m.pi=t.per*s*sqform.pi
pI=c(point.est-m.pi ,point.est+m.pi)
pI
```

```
## [1] 66.60356 67.82379
```

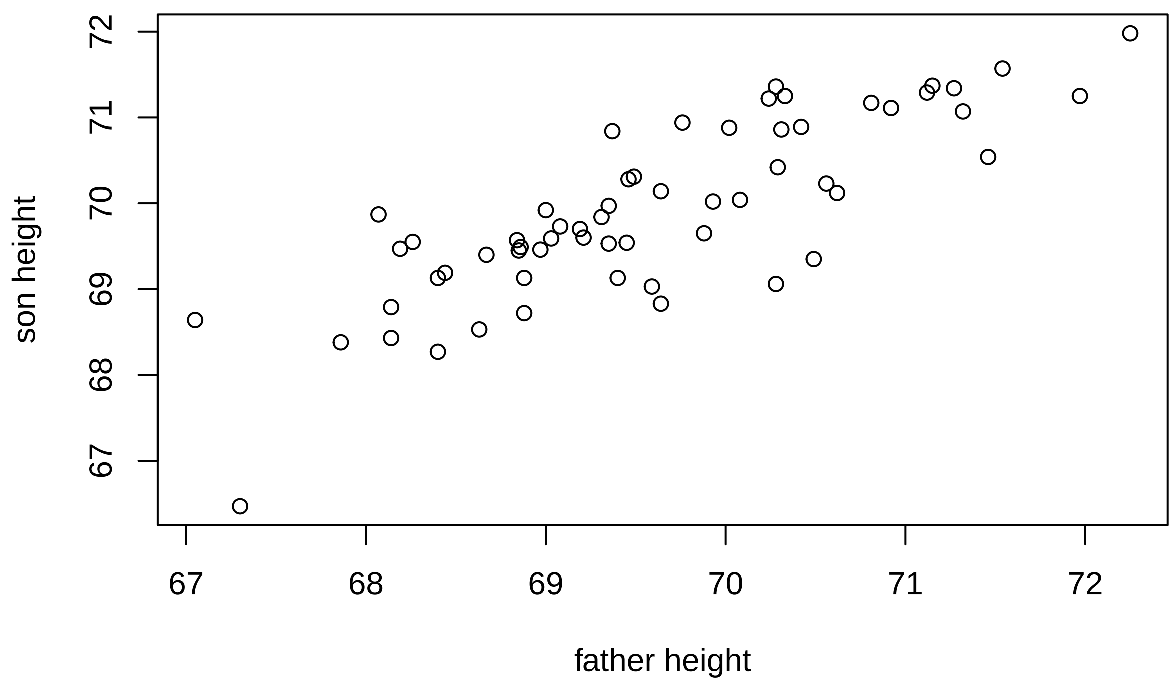
v.

```r
fh=seq(66,72,by=0.5)
len=length(fh)
## 96% CI
low=numeric(len)
upp=numeric(len)
for(r in 1:len){
  xnew=c(1,fh[r],66)
  beta.hat=qr.solve(crossprod(x.mat)) %*% crossprod(x.mat, y)
  point.est[r]=xnew %*% beta.hat
```

```r
  xtxin=qr.solve(crossprod(x.mat))
  sqform=sqrt(t(xnew) %*% xtxin %*% xnew)
   alpha=0.04
   t.per=pt(1-alpha/2,n-p)
   m=t.per*s*sqform
   low[r]=point.est[r]-m
   upp[r]=point.est[r]+m
}

## 96% PI
lower.pi=numeric(len)
upper.pi=numeric(len)
point.est.pi=numeric(len)
for(r in 1:len){
  xnew=c(1,fh[r],66)
  beta.hat=qr.solve(crossprod(x.mat)) %*% crossprod(x.mat, y)
  point.est.pi[r]=xnew %*% beta.hat
  xtxin=qr.solve(crossprod(x.mat))
  sqform.pi=sqrt(t(xnew) %*% xtxin %*% xnew+1)
   alpha=0.04
   t.per=pt(1-alpha/2,n-p)
   m.pi=t.per*s*sqform.pi
   lower.pi[r]=point.est.pi[r]-m.pi
   upper.pi[r]=point.est.pi[r]+m.pi
}

plot(x.df.2[,1],x.df.2[,3],xlab = "father height",ylab="son height")
```




```r
##lines(point.est,x.df.2[,3],col="yellow")
##lines(point.est.pi,x.df.2[,3],col="green")
## ci bound lines
##lines(low,x.df.2[,3],col="red")
```

```
##lines(upp,x.df.2[,3],col="red)
##pi bound lines
##lines(lower.pi,x.df.2[,3],col="blue")
##lines(upper.pi,x.df.2[,3],col="blue")
```

Problem 3

```
x.mat3=read.csv("X.mat.3.csv")
x.mat.3=as.matrix(x.mat3)
n=nrow(x.mat.3)
p=ncol(x.mat.3)
beta=c(1,2,3,4,5)
x.new=c(1,25,20,15,10)
reps=1e3
sigma=1.5
```

i.

```
## coverage probability of a 90% prediction interval
captured.list.pi=numeric(reps)
 for(r in 1:reps){
  ## generate the realizations of the n errors
  epsilon.vec=sigma*(rexp(n,1)-1)
  ## compute the y
  y.vec=x.mat.3 %*%beta+epsilon.vec
  ## compute the realization of betahat
  beta.hat=qr.coef(qr=qr(x.mat.3), y=y.vec)
  s=sqrt(sum((y.vec-x.mat.3 %*% beta.hat)^2 )/(n-p))
  alpha.pi=0.1
  tper.pi=pt(1-alpha.pi/2,n-p)
  xtxin=qr.solve(crossprod(x.mat.3))
  sqform.pi=sqrt(1+t(x.new) %*% xtxin %*% x.new)
  moe=tper.pi*s*sqform.pi
  est.mean=sum(beta.hat * x.new)
  left.pi=est.mean-moe
  right.pi=est.mean+moe
  ## generate the realization of Y_new
  ynew=sum(beta*x.new)+ sigma*(rexp(1,1)-1)
  ## check if ynew was captured
  captured.list.pi[r]=1*(left.pi<=ynew)&(ynew<=right.pi)
 }

mean(captured.list.pi)
```

```
## [1] 0.552
```

```
prop.test(x=sum(captured.list.pi),n=reps,conf.level = 0.99,correct = FALSE)$conf.int[1:2]
```

```
## [1] 0.5112829 0.5920316
```

ii.

```
## coverage probability of a 90% prediction interval
captured.list.pi=numeric(reps)
 for(r in 1:reps){
  ## generate the realizations of the n errors
  epsilon.vec=sigma*sqrt(12)*(runif(n)-0.5)
```

```r
  ## compute the y
  y.vec=x.mat.3 %*%beta+epsilon.vec
  ## compute the realization of betahat
  beta.hat=qr.coef(qr=qr(x.mat.3), y=y.vec)
  s=sqrt(sum((y.vec-x.mat.3 %*% beta.hat)^2 )/(n-p))
  alpha.pi=0.1
  tper.pi=pt(1-alpha.pi/2,n-p)
  xtxin=qr.solve(crossprod(x.mat.3))
  sqform.pi=sqrt(1+t(x.new) %*% xtxin %*% x.new)
  moe=tper.pi*s*sqform.pi
  est.mean=sum(beta.hat * x.new)
  left.pi=est.mean-moe
  right.pi=est.mean+moe
  ## generate the realization of Y_new
  ynew=sum(beta*x.new)+ sigma*sqrt(12)*(runif(1)-0.5)
  ## check if ynew was captured
  captured.list.pi[r]=1*(left.pi<=ynew)&(ynew<=right.pi)
 }

mean(captured.list.pi)
```

```
## [1] 0.578
```

```r
prop.test(x=sum(captured.list.pi),n=reps,conf.level = 0.99,correct = FALSE)$conf.int[1:2]
```

```
## [1] 0.5373866 0.6175852
```

iii.

```r
## coverage probability of a 90% prediction interval
captured.list.pi=numeric(reps)
x=apply(x.mat.3,1,sum)
sd=numeric(n)
for( i in 1:n){
  x_1=x[i]
  dom=(sqrt(p* x_1))
  sd[i]=sigma/dom
}
  for(r in 1:reps){
     ## generate the realizations of the n errors
   epsilon.vec=numeric(n)
   for( i in 1:n){
      epsilon.vec[i]=rnorm(1,0,sd[i])
   }
  ## compute the y
  y.vec=x.mat.3 %*%beta+epsilon.vec
  ## compute the realization of betahat
  beta.hat=qr.coef(qr=qr(x.mat.3), y=y.vec)
  s=sqrt(sum((y.vec-x.mat.3 %*% beta.hat)^2 )/(n-p))
  alpha.pi=0.1
  tper.pi=pt(1-alpha.pi/2,n-p)
  xtxin=qr.solve(crossprod(x.mat.3))
  sqform.pi=sqrt(1+t(x.new) %*% xtxin %*% x.new)
  moe=tper.pi*s*sqform.pi
  est.mean=sum(beta.hat * x.new)
```

```r
  left.pi=est.mean-moe
  right.pi=est.mean+moe
  ## generate the realization of Y_new
  ynew=sum(beta*x.new)+rnorm(1,mean=0,sd=sigma/sqrt(p*sum(x.new)))
  ## check if ynew was captured
  captured.list.pi[r]=1*(left.pi<=ynew)&(ynew<=right.pi)
 }

mean(captured.list.pi)
```

```
## [1] 0.61
```

```r
prop.test(x=sum(captured.list.pi),n=reps,conf.level = 0.99,correct = FALSE)$conf.int[1:2]
```

```
## [1] 0.5696699 0.6488801
```

Problem 4

$h(a) = (E[a\bar{X} - \mu])^2 + Var(a\bar{X})$

Because $(E[a\bar{X} - \mu])^2 = (E[a\bar{X}] - E[\mu])^2 = (aE[\bar{X}] - \mu)^2 = (a\mu - \mu)^2 = \mu^2(a-1)^2$

$Var(a\bar{X}) = a^2 Var(\bar{X}) = a^2\frac{\sigma^2}{n}$

$h(a) = \mu^2(a-1)^2 + \frac{\sigma^2}{n}a^2$

i.

$\nabla h(a) = 2\mu^2(a-1) + \frac{2\sigma^2}{n}a = 2(\mu^2 + \frac{\sigma^2}{n})a - 2\mu^2$

$\nabla^2 h(a) = 2(\mu^2 + \frac{\sigma^2}{n})$

ii.

If the $\nabla h(\bar{a}) = 2(\mu^2 + \frac{\sigma^2}{n})\bar{a} - 2\mu^2 = 0$, then $\bar{a} = \frac{\mu^2}{\mu^2 + \frac{\sigma^2}{n}}$

Because $\nabla^2 h(a) = 2(\mu^2 + \frac{\sigma^2}{n})$ is always greater than 0 for every a.

$\bar{a} = \hat{a}$ is the global minimizer of f .

iii.

Because the $\nabla^2 h(a)$ is not relevant to a.

Problem 5

```r
g=function(x){
  return(sin(x)/x)
}
```

i.

$\nabla g(x) = \frac{xcosx - sinx}{x^2}, x \in (0, 6\pi]$

ii.

```r
deriv.g=function(x){
  return((x*cos(x)-sin(x))/(x^2))
}
bsearch=function(df, a, b, L=1e-7, quiet=FALSE, ...){
  k=0
  while((b-a)>L){
    k=k+1
```

```r
    ## compute the midpoint
    m=(a+b)/2
    ## compute the derivative at the midpoint
    df.at.m=df(m, ...)
    if(df.at.m<0){
      ## f is decreasing at
      ## the midpoint so the
      ## new interval is (m, b)
      a=m
    }
    else if(df.at.m>0){
      ## f is increasing at the
      ## midpoint so the
      ## new interval is (a, m)
      b=m
      }
    else{
      ## m is a stationary point
      a=m
      b=m
    }
    if(!quiet){
      cat("after iteration k =",k,"the interval is", a, b, "nn")


    }
  }## return the midpoint of the final interval
  return((a+b)/2)
}

a0=0.1
b0=6*pi
xbar=bsearch(df=deriv.g,a=a0,b=b0,L=1e-7,quiet = FALSE)
```

```
## after iteration k = 1 the interval is 9.474778 18.84956 nnafter iteration k = 2 the interval is 14.1
```

```r
xbar
```

```
## [1] 17.22076
```

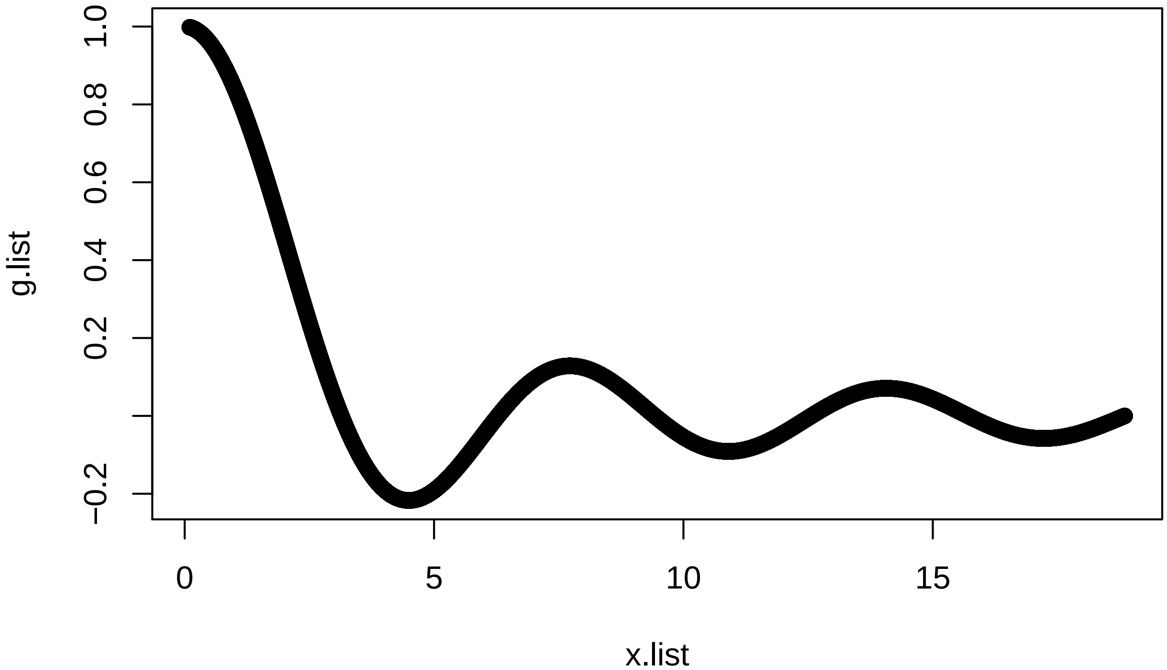
```r
g(xbar)
```

```
## [1] -0.0579718
```

iii.

```r
x.list=seq(0.1,6*pi,by=0.001)
g.list=numeric(length = length(x.list))
for(r in 1:length(x.list)){
  g.list[r]=g(x.list[r])
}

plot(x.list,g.list)
```



Based on this plot, the value of $\bar{x}$ I find in part ii is not a global minimizer.