SC-HW4

ID : 111024517 Name : 鄭家豪

Problem 1

$$Y_i \overset{\text{indep.}}{\sim} N(\mu_i, \sigma_i^2), i = 1, 2, ..., n;$$

$$\mu_i = \beta_0 + \beta_1 x_i,\ \ \sigma_i^2 = e^{\alpha_0+\alpha_1 x_i}, \alpha = (\alpha_0, \alpha_1)', \beta = (\beta_0, \beta_1)'$$

The -log likelihood function :

$$\text{The likelihood: } L(\alpha, \beta) = \prod_{i=1}^{n} \frac{1}{2\pi} \exp\left(-\frac{1}{2}(\alpha_0 + \alpha_1 x_i)\right) \exp\left(-\frac{(y_i - (\beta_0 + \beta_1 x_i))^2}{2e^{\alpha_0+\alpha_1 x_i}}\right)$$

$$\log L(\alpha, \beta) = l(\alpha, \beta) \propto \frac{-1}{2} \sum_{i=1}^{n} [\alpha_0 + \alpha_1 x_i + \frac{(y_i - (\beta_0 + \beta_1 x_i))^2}{e^{\alpha_0+\alpha_1 x_i}}]$$

$$-l(\alpha, \beta) \propto \frac{1}{2} \sum_{i=1}^{n} [\alpha_0 + \alpha_1 x_i + \frac{(y_i - (\beta_0 + \beta_1 x_i))^2}{e^{\alpha_0+\alpha_1 x_i}}] = g(\alpha, \beta)$$

Then ,use block coordinate descent Algorithm:

Algorithm:

1. Initialize $\beta^{(0)}$,choose $\beta^{(0)} = (X^T X)^{-1} X^T Y$.

2. For t=0,1,2,...,

$\alpha^{(t)} = arg\,\min_{\alpha} g(\alpha, \beta^{(t)})$;$\beta^{(t+1)} = (X^T \Sigma^{-1}(\alpha^{(t)})X)^{-1}(X^T \Sigma^{-1}(\alpha^{(t)})Y)$,where $\Sigma(\alpha) = \text{diag}[\exp(X\alpha)]$.

3. Repeat 2 until $||\alpha^{(t)}||_1 < 0.000001$ and $||\beta^{(t)}||_1 < 0.000001$.

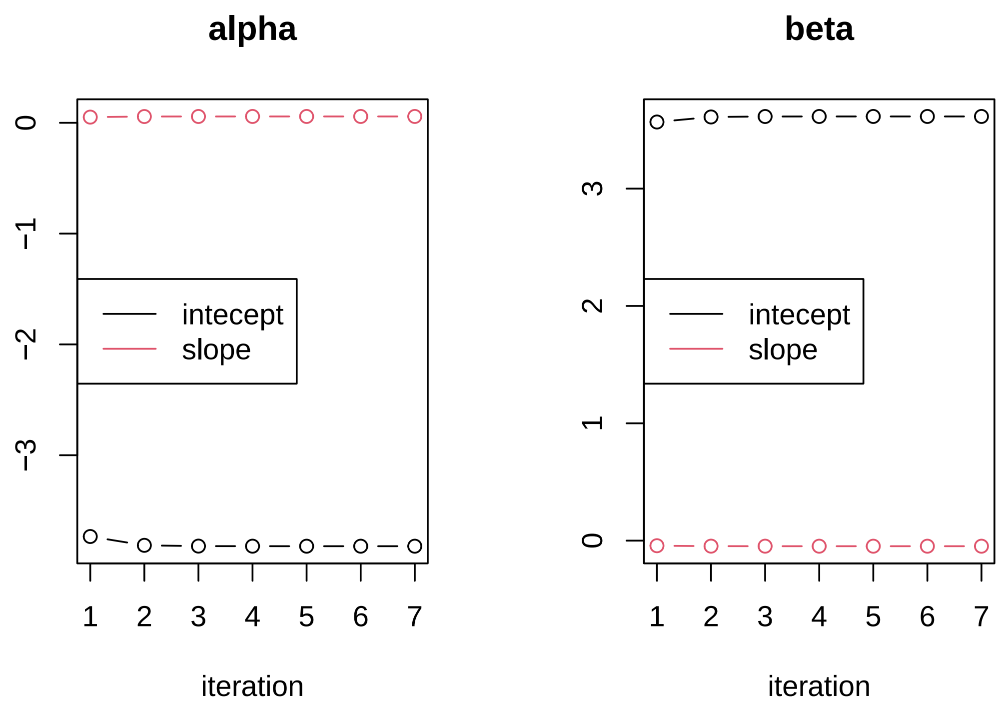
```
data1 <- read.csv("DataA.csv",header = T)
nglogL <- function(alpha,beta){
  sum(X%*%alpha+(y-X%*%beta)^2/exp(X%*%alpha))
}
X <- cbind(1,data1$lstat)
y <- data1$medv.log
err0 =0.000001
count = 1
beta.cd <- beta.all<- c(solve(t(X)%*%X)%*%t(X)%*%y) #initial beta
alpha.cd <-alpha.all<- nlminb(start = c(0,1),
                    obj = nglogL,beta=beta.cd)$par #initial alpha
g <- nglogL(alpha.cd,beta.cd) #initial -loglikelihood
repeat{
  err1=err2 = err0
```

```r
  sigma <- solve(diag(exp(c(X %*% alpha.cd))))
  b = c(solve(t(X)%*%sigma%*%X)%*%t(X)%*%sigma%*%y)
  a = nlminb(start=c(0,1),obj=nglogL,beta=b)
  err1 = max(err1,abs(a$par-alpha.cd))
  err2 = max(err2,abs(b-beta.cd))
  alpha.cd = a$par
  beta.cd = b
  g = c(g,nglogL(alpha.cd,beta.cd))
  beta.all <- cbind(beta.all,beta.cd)
  alpha.all <- cbind(alpha.all,alpha.cd)
  count = count+1
  if (err1 == err0 & err2 == err0) break
}
```

```
The iteration number: 7
```

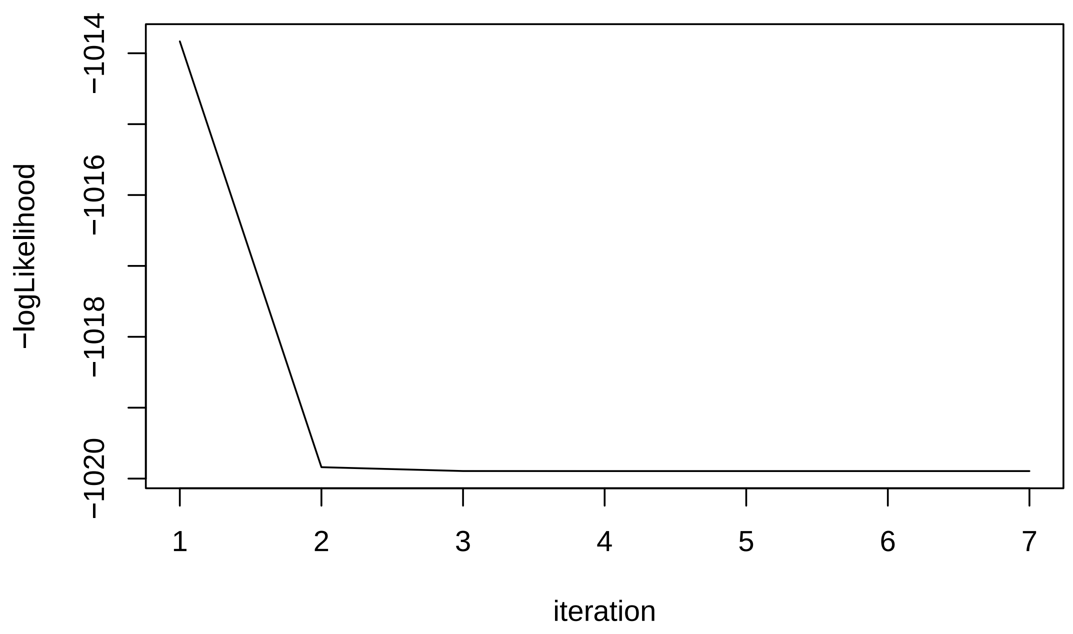
For α and β:

```r
par(mfrow=c(1,2))
ts.plot(t(alpha.all),col=1:2,type="b",
        xlab="iteration",main="alpha")
legend("left", legend = c("intecept","slope"), col = 1:2, lty = 1)
ts.plot(t(beta.all),col=1:2,type="b",
        xlab="iteration",main="beta")
legend("left", legend = c("intecept","slope"), col = 1:2, lty = 1)
```



For negative logLikelihood:

```
ts.plot(g,xlab="iteration",ylab="-logLikelihood")
```



From the above,the parameter(α, β) will be stable , and -logLikelihood value is minimized.

Finally,print the MLE of (α, β):

```r
result <- rbind(t(beta.cd),t(alpha.cd))
row.names(result) <- c("$\\beta$","$\\alpha$")
knitr::kable(result,row.names = T,
             col.names = c("intecept(mle)","slope(mle)"))
```

	intecept(mle)	slope(mle)
β	3.614909	-0.0475170
α	-3.821260	0.0572457

Problem 2

- ADMM algorithm:

$$L_\rho(\beta, \delta, \lambda) = \frac{1}{2}||y - \beta||^2 + \tau|\delta| + \lambda^{'}(DB - \delta) + \rho/2||D\beta - \delta||^2$$

1. Given $\tau, \rho, \text{eps.conv}, \text{iter.max}$.
2. t =1,2,...,

For β:

$$\beta^{(t)} = \text{arg min}_\beta\{\frac{1}{2}||y - \beta||^2 + \lambda^{'(t-1)}(D\beta - \delta^{(t-1)}) + \rho/2||D\beta - \delta^{(t-1)}||^2\}$$
$$\beta^{(t)} \leftarrow (I + \rho^{(t-1)}D^{'}D)^{-1}[y + \rho D^{'}(\delta - \lambda^{(t-1)}/\rho)]$$

For δ:

$$\delta^{(t)} = \text{arg min}_\delta\{\tau|\delta| + \lambda^{'(t-1)}(D\beta^{(t-1)} - \delta) + \rho/2||D\beta^{(t-1)} - \delta||^2\}$$
$$\delta^{(t)} \leftarrow \text{sign}(D\beta^{(t-1)} + \frac{1}{\rho}\lambda^{(t-1)})(|D\beta^{(t-1)} + \frac{1}{\rho}\lambda^{(t-1)}| - \frac{\tau}{\rho})$$

For λ:

$$\lambda^{(t)} \leftarrow \lambda^{(t-1)} + \rho(D\beta^{(t-1)} - \delta^{(t-1)})$$

3. Repeat 2 until $||\beta||_1 < \text{eps.conv}$.

4. If the iteration exceeds iter.max, break.

- $\text{Setting}: \tau = 1, \rho = 0.1, \text{eps.conv} = 0.001, \text{iter.max} = 5000.$

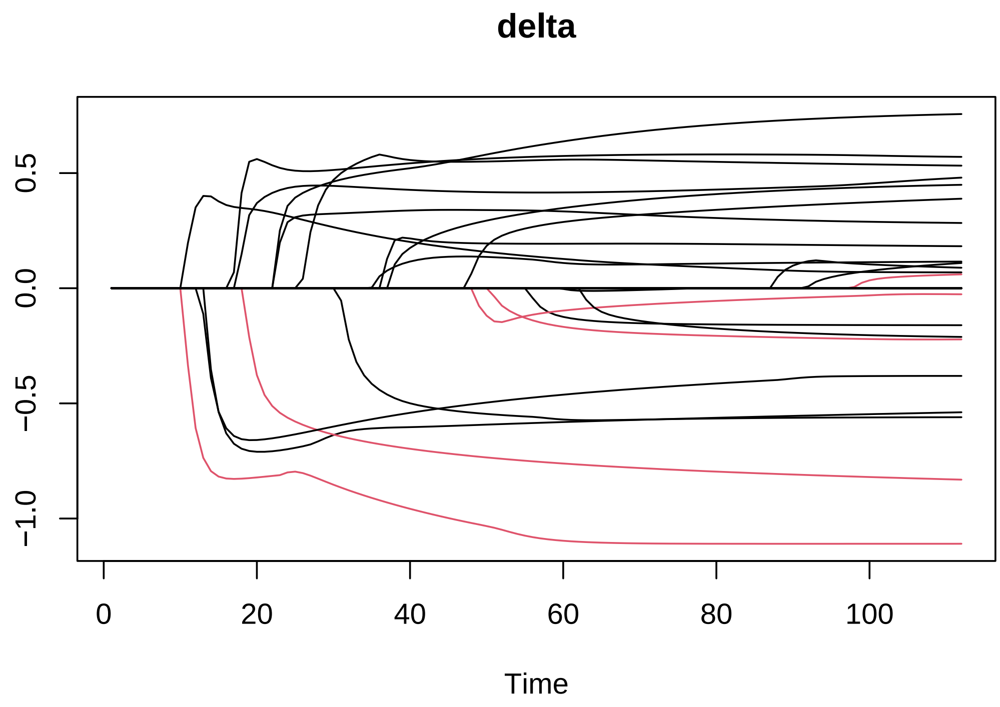
```r
data2 <- read.csv("DataB.csv",header = T)
soft_threshold <- function(a, threshold){
  sign(a)*max(0, abs(a)-threshold)
```

```r
}
soft_threshold <- Vectorize(soft_threshold)

admm <- function(yvec, tau, rho, eps.conv, iter.max){
  n <- length(yvec)
  beta.all <- beta <- rep(mean(yvec),n)
  delta <- delta.all <- matrix(0,n-2,1)
  lambda <- lambda.all <- matrix(0,n-2,1)
  count <- 0

  D <- cbind(diag(n-2),0,0)+cbind(0,diag(-2,n-2),0)+cbind(0,0,diag(n-2))
  IDtD_inv <- solve(diag(n)+rho*t(D)%*%D)
  loss <-  loss.all <- 0.5*sum((yvec-beta)^2)+ tau*sum(abs(diff(beta)))
  err.all <- c()

  repeat{
    beta <- IDtD_inv%*%(yvec+rho*t(D)%*%(delta-lambda/rho))
    delta <- soft_threshold(D%*%beta+lambda/rho, tau/rho)
    lambda <- lambda + rho*(D%*%beta-delta)
    beta.all <- cbind(beta.all, beta)
    delta.all <- cbind(delta.all, delta)
    lambda.all <- cbind(lambda.all, lambda)
    count <- count+1
    loss1 <- 0.5*sum((yvec-beta)^2)+
      tau*sum(abs(diff(beta)))
    loss.all <- c(loss.all, loss1)
    err <- max(abs(beta- beta.all[,count]))
    err.all <- c(err.all, err)

    if (err < eps.conv | count > iter.max) break
  }

  return(list(count=count, beta=beta.all,
              lambda=lambda.all, delta=delta.all,
              err=err.all, loss=loss.all))
}
x <- data2$x
tmp <- admm(x, tau=1, rho=0.1, eps.conv=0.001, iter.max=5000)

cat("The iteration number:",tmp$count)
```

```
The iteration number: 111
```

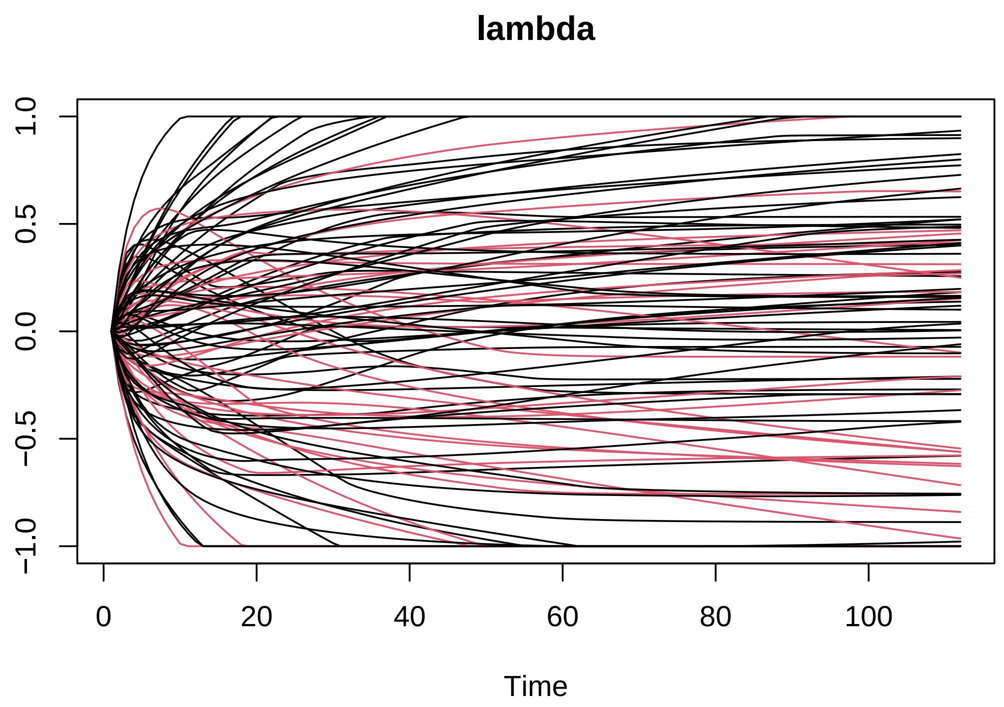
- See δ:

```
idx <- 31:60
col.idx <- rep(1,100)
col.idx[idx] <- 2
ts.plot(t(tmp$delta), col=col.idx); title("delta")
```



- See λ:

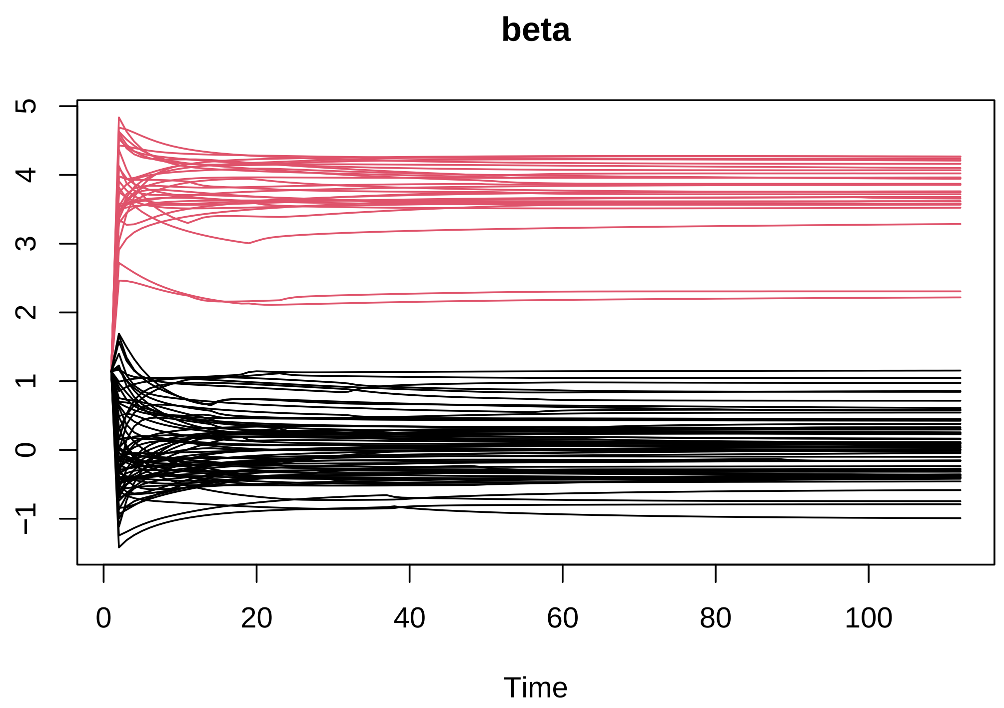
```
ts.plot(t(tmp$lambda), col=col.idx); title("lambda")
```



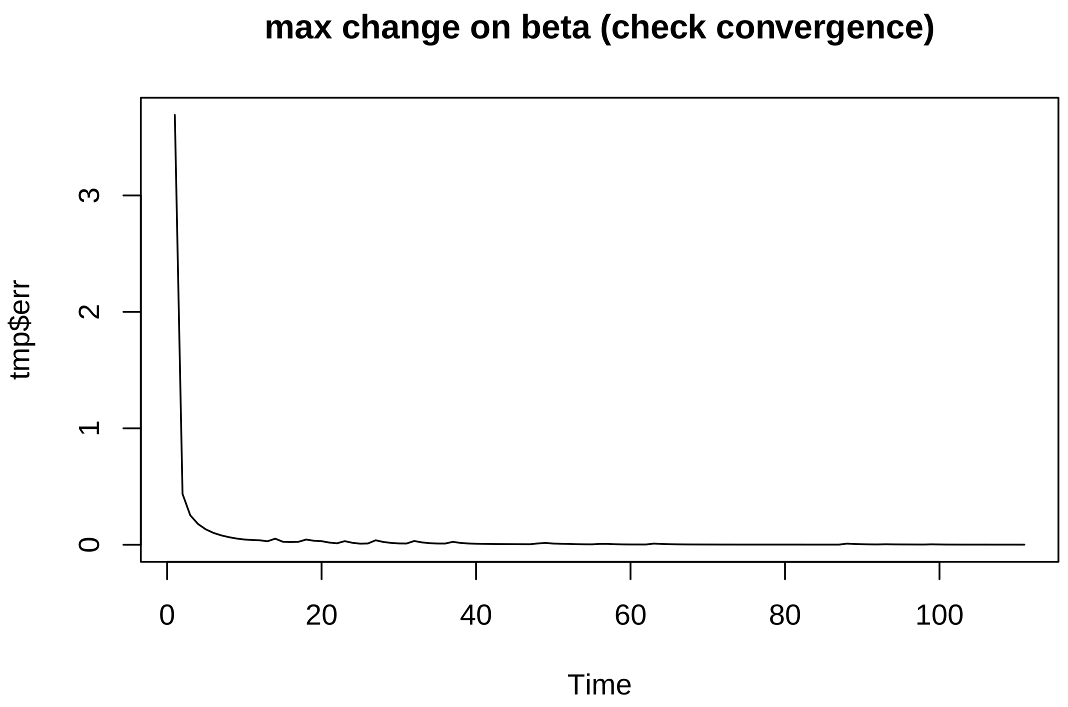
From the above graphs, we can know that these tunning parameters (δ, λ) will be stable.

- See β:

```
ts.plot(t(tmp$beta), col=col.idx); title("beta")
```



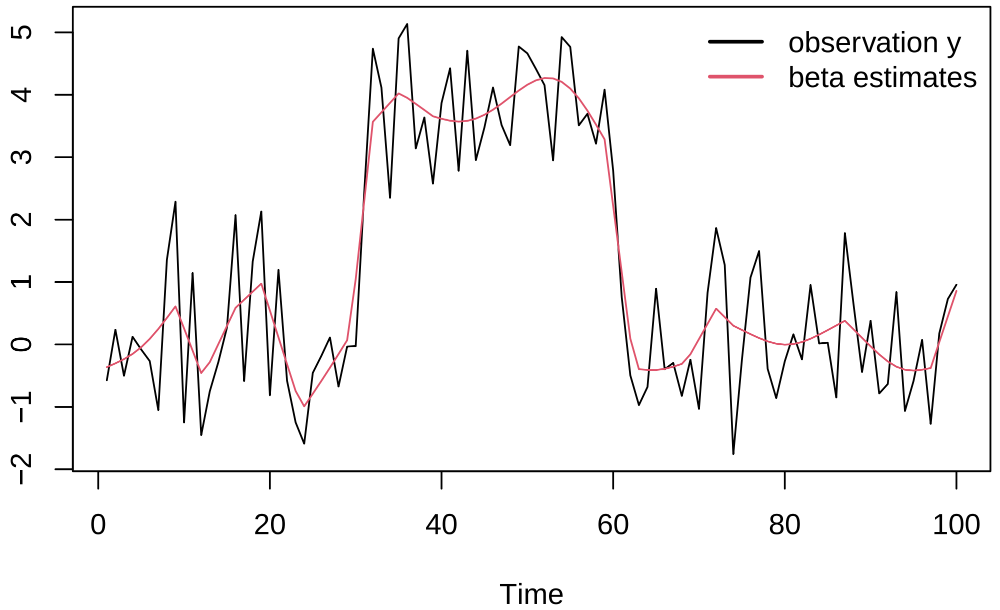
```
ts.plot(tmp$err); title("max change on beta (check convergence)")
```



From the above,the change in β will be stable.

- Comparison between original observation and estimates:

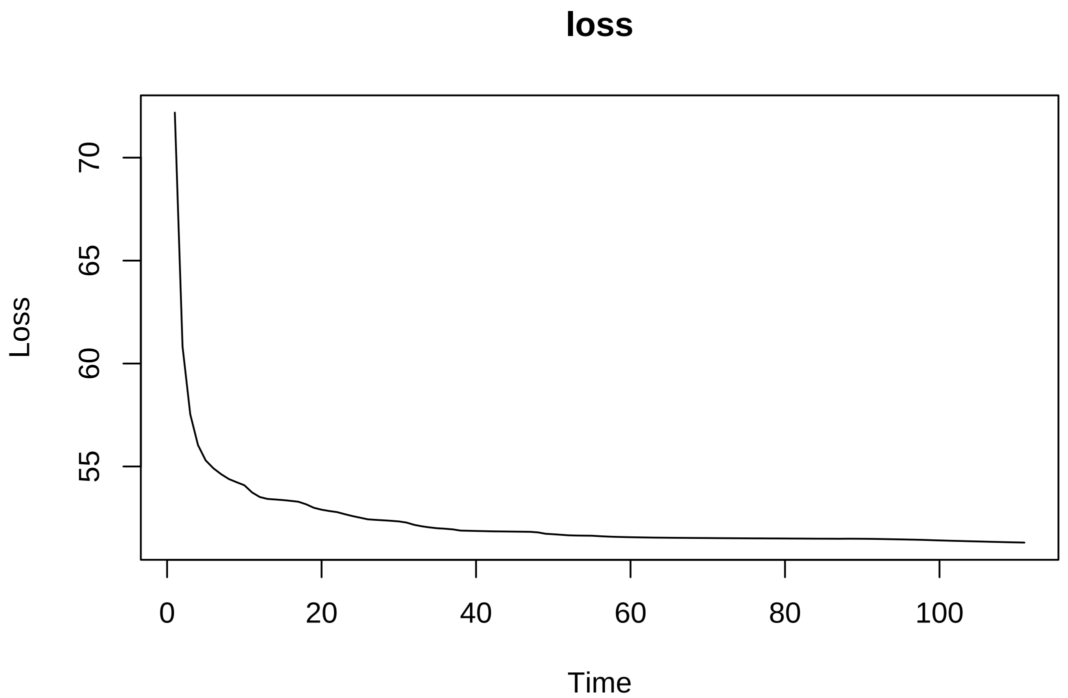
```
ts.plot(cbind(x,tmp$beta[,tmp$count+1]), col=1:2)
legend("topright", legend=c("observation y", "beta estimates"), col=1:2, lwd=2, lty=1, bty="n")
```



From the above,beta estimates are good.

- The change of loss:

```
ts.plot(tmp$loss[-1], ylab = "Loss"); title("loss")
```



The loss seems to tend to a stable constant value as the iteration increases.

Finally, the estimated $\{\beta_i\}$(observe the following from the left column to right column along its row):

```
answer <- matrix(tmp$beta[,tmp$count+1],10,10)
knitr::kable(round(answer,3))
```

-0.363	-0.100	0.114	2.306	3.584	4.231	1.155	0.331	0.006	-0.159
-0.303	-0.458	-0.316	3.567	3.571	4.268	0.094	0.575	0.039	-0.272
-0.235	-0.283	-0.745	3.719	3.582	4.263	-0.397	0.438	0.092	-0.356
-0.152	0.007	-0.992	3.871	3.619	4.207	-0.408	0.301	0.157	-0.405
-0.045	0.297	-0.790	4.022	3.680	4.100	-0.409	0.232	0.230	-0.418
0.089	0.587	-0.584	3.952	3.761	3.944	-0.392	0.166	0.304	-0.405
0.248	0.716	-0.372	3.855	3.858	3.747	-0.356	0.103	0.380	-0.378
0.425	0.846	-0.154	3.757	3.964	3.523	-0.311	0.050	0.243	0.038
0.610	0.975	0.068	3.656	4.068	3.286	-0.157	0.012	0.106	0.449
0.256	0.544	1.045	3.616	4.161	2.219	0.087	-0.004	-0.030	0.857

Problem 3

Consider penalize term λ to minimize:

$$\begin{aligned} F(x,\lambda) &= f(x) + \lambda||x||_2 - \frac{\lambda^2}{2}||x||_2^2 \\ &= -\frac{\lambda^2}{2}||x||_2^2 + \lambda||x||_2 - \sum_{j=1}^{10}\sin(\pi x_j)[\sin(j\pi x_j^2)]^{20}, x = (x_1, ..., x_{10})', x_j \in [0,1]. \end{aligned}$$

Algorithm(coordinate descent):

1. initial $x^{(0)}$ = (0.5,0.5,0.5,0.6,0.5,0.5,0.5,0.6,0.5,0.5) and $\lambda^{(0)} = 0.5$.
2. For iteration t=0,1,2,...,update coordinate j $\in \{1,2,..,10\}$ via

$$\begin{aligned} x_j^{(t+1)} &= arg\ min_{x_j}\ F(x_j, x_{-j}^{(t)}, \lambda^{(t)}), \text{where } x_{-j}^{(t)} = (x_1^{(t+1)}, ..., x_{j-1}^{(t+1)}, x_{j+1}^{(t)}, ..., x_{10}^{(t)}). \\ \lambda^{(t+1)} &= arg\ max_\lambda\ F(x_j^{(t+1)}, \lambda) = \frac{||x||_2}{||x||_2^2} = \frac{1}{||x||_2}. \end{aligned}$$

3. Repeat 2 until $||x^{(t)}||_1 < 0.000001$.

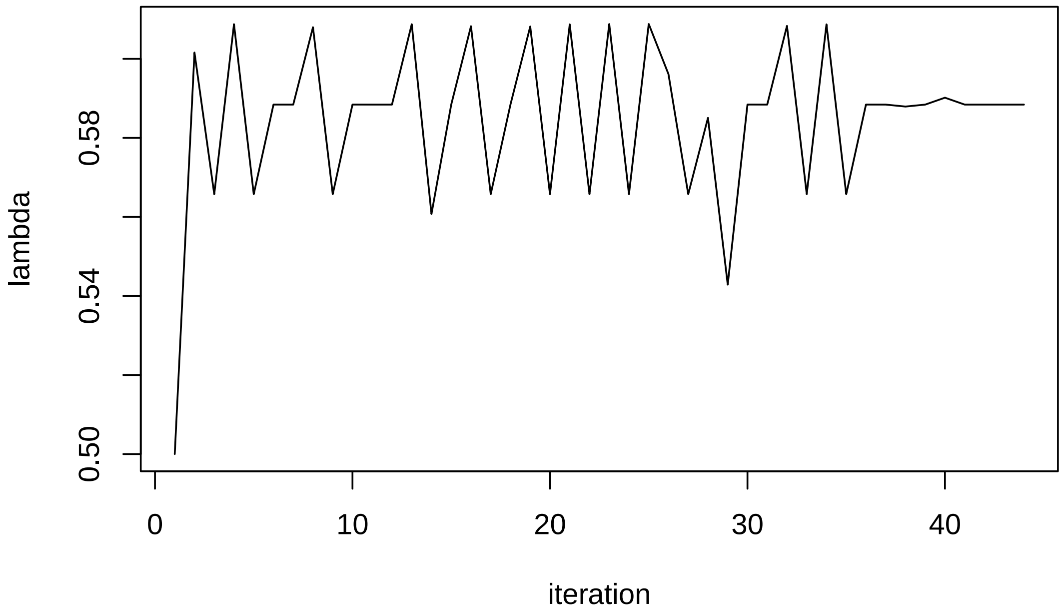
```
f3 = function(x,lambda){
  index <- 1:length(x)
  -sum(sin(pi*x)*(sin(index*pi*x^2))^20) +
    lambda*sqrt(sum(x^2))- lambda^2/2*sum(x^2)
} # function f
p=10
count = 1
err0 =err.all = 0.000001 #stop criteria
```

```r
l <- rep(0,10) #lower bound
u <- rep(1,10) #upper bound
lambda <- lambda.all <- 0.5
x.cd <-x.all<-x.initial <- c(0.5,0.5,0.5,0.6,0.5,0.5,0.5,0.6,0.5,0.5)
f3.all <- f3(x.cd,lambda)
repeat{
  err = err0
  for (i in 1:p){
    a = nlminb(start=x.initial,obj=f3,lambda=lambda.all[count],lower = l,upper = u)
    x.cd[i] = a$par[i]
  }
  lambda <- 1/sqrt(sum(x.cd^2))
  lambda.all <- cbind(lambda.all,lambda)
  x.all = cbind(x.all,x.cd)
  f3.all <- c(f3.all,f3(x.cd,0))
  count = count+1
  err = max(err,sum(abs(x.cd-x.all[,count-1])))
  err.all <- c(err.all,err)
  if (err==err0 | count ==100) break
}
```

```
The iteration: 44
```

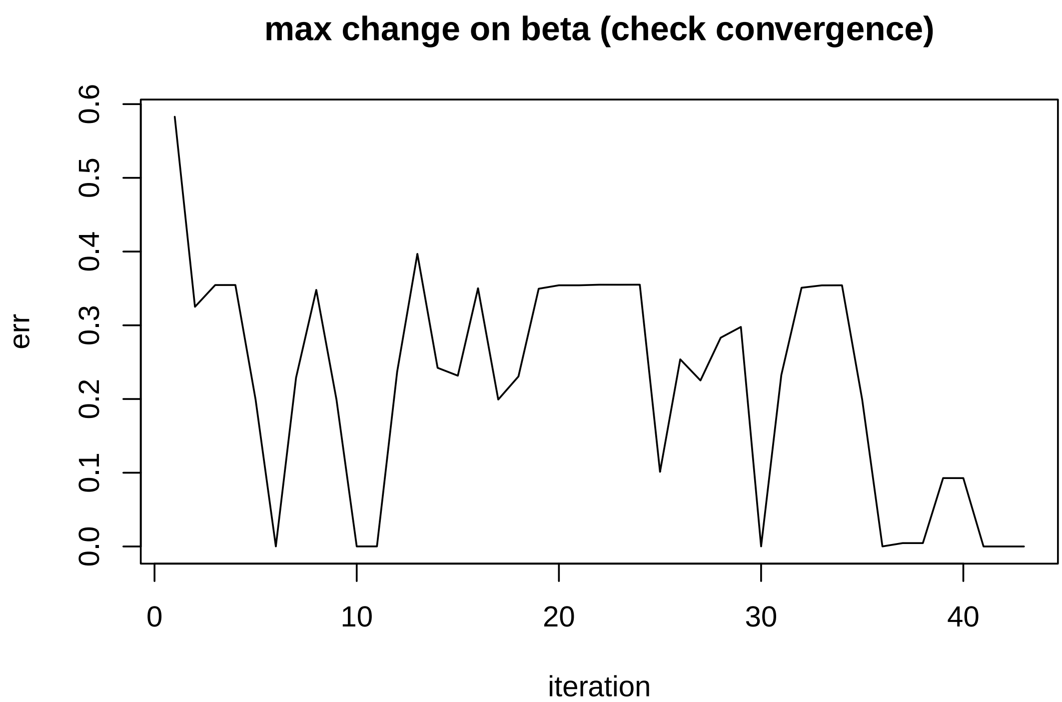
- See the change on penalize term:

```r
ts.plot(t(lambda.all),xlab="iteration",ylab="lambda")
```



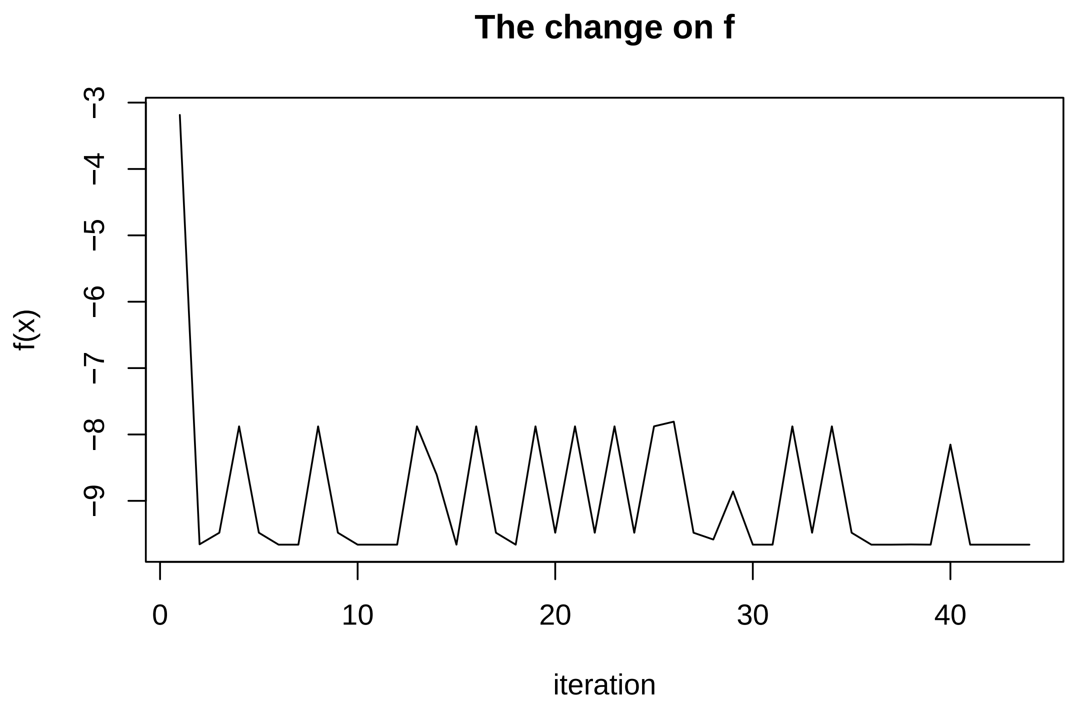
- See the change on x:

```
ts.plot(err.all[-1],xlab="iteration",ylab="err") ; title("max change on beta (check convergence)")
```



- See the change on f(x):

```r
ts.plot(f3.all,xlab="iteration",ylab="f(x)") ; title("The change on f")
```



Finally,print the desired result:

```
The minimum of f(x): -9.660152

And the corresponding x's (rounding to 4 digits):
 0.7012 0.5 0.409 0.6121 0.5476 0.5 0.463 0.559 0.527 0.5
```