a4

1(a)

suppose we want to estimate how much the following colors are in the color crimson
1) how much red, green, blue
2) how much red, scarlet, cherry (some color looks like red)
we know it's easier and more precisise to estimate with set 1). this is because the variables red green blue are very different from each other, the answer would be like 90% red 5% green 5% blue
however it's hard to estimate the second set because the variables are very much alike, the result of mixing 33% red 33% scarlet 33% cherry could be very similar to mixing 100% cherry 0% others. thus we don't have confidence on our esitmated result.
when the variables are alike (scale multiple of one another) or when the variables are corelated to each other(linearly dependent or close to it), as the case of 2), we call it multicoliearity and it result in uncertainty of our estimate.

1(b)

```r
data <- read.csv("HigherEducation.csv", header=T)
attach(data)
X=cbind(Accept, Enroll,Top10perc, Top25perc, F.Undergrad, P.Undergrad, Outstate, Room.Board, Books, Pers
detach(data)
y=data$Apps
Xtrain=X[1:600,]
ytrain=y[1:600]
Xtest=X[601:777,]
ytest=y[601:777]
data.train=data[1:600,]
data.test=data[601:777,]
library(regclass)
```

```
## Loading required package: bestglm
```

```
## Loading required package: leaps
```

```
## Loading required package: VGAM
```

```
## Loading required package: stats4
```

```
## Loading required package: splines
```

```
## Loading required package: rpart
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
## Important regclass change from 1.3:
## All functions that had a . in the name now have an _
## all.correlations -> all_correlations, cor.demo -> cor_demo, etc.
```

```r
fit.ls=lm(Apps~.,data=data.train)
VIF(fit.ls)
```

```
##      Accept      Enroll   Top10perc   Top25perc F.Undergrad P.Undergrad
##    7.261634   24.063788    6.564348    5.470781   19.900635    1.742497
##    Outstate  Room.Board       Books    Personal         PhD    Terminal
##    3.572827    1.971827    1.130445    1.328470    4.066085    3.877562
##   S.F.Ratio perc.alumni      Expend   Grad.Rate
##    1.839082    1.923219    2.773314    1.860827
```

We observe there is multicoliearity issue, typically, 2(Enroll) and 5(F.Undergrad) are most problematic because the can be well esitmated by the other variables (variables excpet itself and App), esitmating them with other variables have small MSE so they have large VIF.

1(c)

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.0-2
```

```r
fit.Lasso=glmnet(Xtrain , ytrain , alpha=1, lambda=35,
                 standardize=TRUE, intercept = TRUE ,  family = "gaussian")
fit.Lasso$beta
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## Accept       1.432557305
## Enroll       .
## Top10perc   32.563937537
## Top25perc    .
## F.Undergrad  .
## P.Undergrad  0.005339685
## Outstate    -0.065979706
## Room.Board   0.081115020
## Books        .
## Personal     .
## PhD         -2.291504154
## Terminal    -2.076292458
## S.F.Ratio    4.202810150
## perc.alumni -2.254493155
## Expend       0.062421302
## Grad.Rate    3.823481536
```

```r
lassos=abs(as.array(fit.Lasso$beta))
order=order(lassos,decreasing=TRUE)
sort(lassos,decreasing = TRUE)
```

```
##  [1] 32.563937537  4.202810150  3.823481536  2.291504154  2.254493155
##  [6]  2.076292458  1.432557305  0.081115020  0.065979706  0.062421302
## [11]  0.005339685  0.000000000  0.000000000  0.000000000  0.000000000
## [16]  0.000000000
```

```r
order
```

```
##  [1]  3 13 16 11 14 12  1  8  7 15  6  2  4  5  9 10
```

we choose variable 3, 13, 16, 11, 14, 12, 1. the ones that has beta above 0.1. we observe the problematic variables in (b) are not selected, this is expected because we shouldn't incorporate variables with high multicolinearity issue.\ however some variable with small multicolinearity issue (VIF $>$ 5) are choosen, this could be reasonable if the variables they are correlated with isn't in the model, however this information is not shown by VIF.

1(d)

```
predict.lasso = predict(fit.Lasso,newx=Xtest)
ndata=subset(data.test, select = -c(Apps))
predict.ls = predict(fit.ls, ndata)
summary(predict.ls-ytest)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2715.238  -343.949    -1.399    64.947   415.525  2620.532
```

```
summary(predict.lasso-ytest)
```

```
##        s0
##  Min.   :-2549.59
##  1st Qu.: -272.36
##  Median :   51.38
##  Mean   :   95.79
##  3rd Qu.:  478.88
##  Max.   : 2860.70
```

```
t(predict.lasso-ytest)%*%(predict.lasso-ytest)/length(ytest)
```

```
##          s0
## s0 524251.7
```

```
t(predict.ls-ytest)%*%(predict.ls-ytest)/length(ytest)
```

```
##          [,1]
## [1,] 556798.7
```

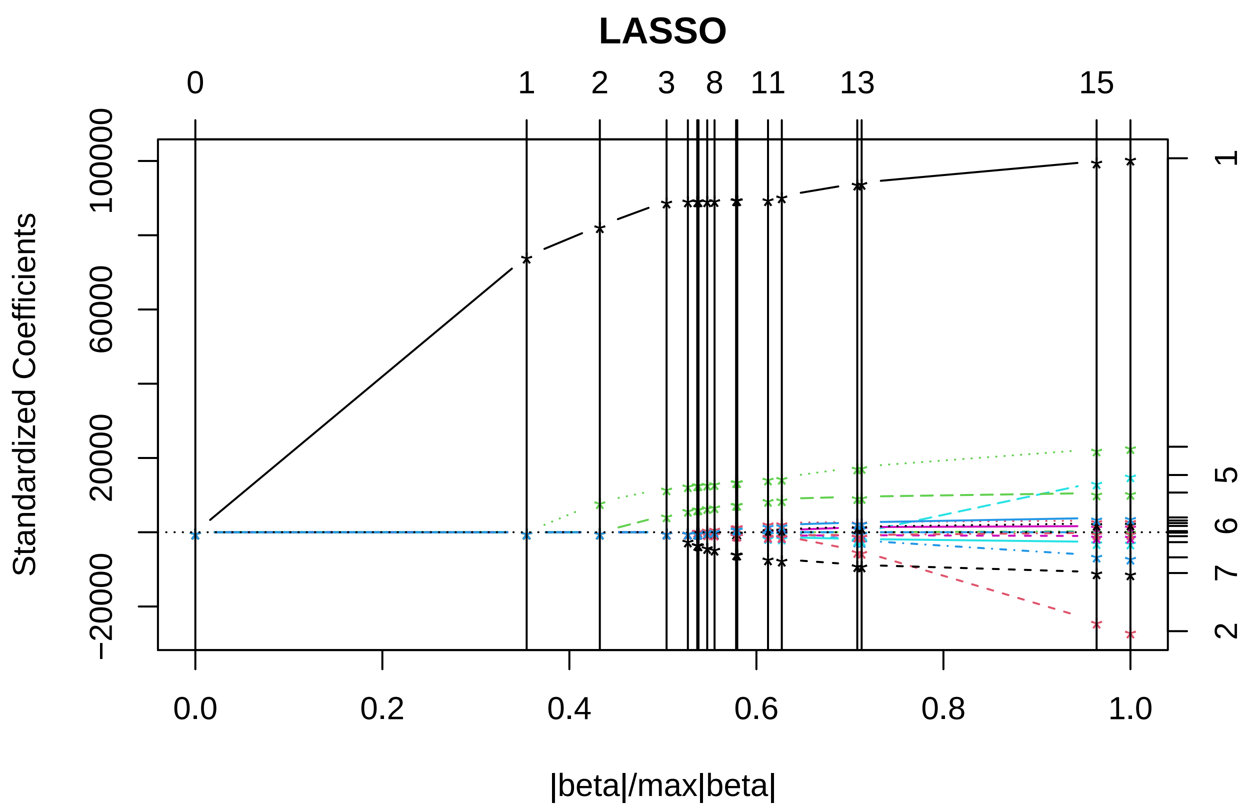
we observe lasso performs slightly better than lease squares. this is because lasso avoid overfitting through variable selection

2(a)

```r
library(lars)
```

```
## Loaded lars 1.2
```

```r
HE.lasso = lars(Xtrain , ytrain, type="lasso")

plot(HE.lasso)
```



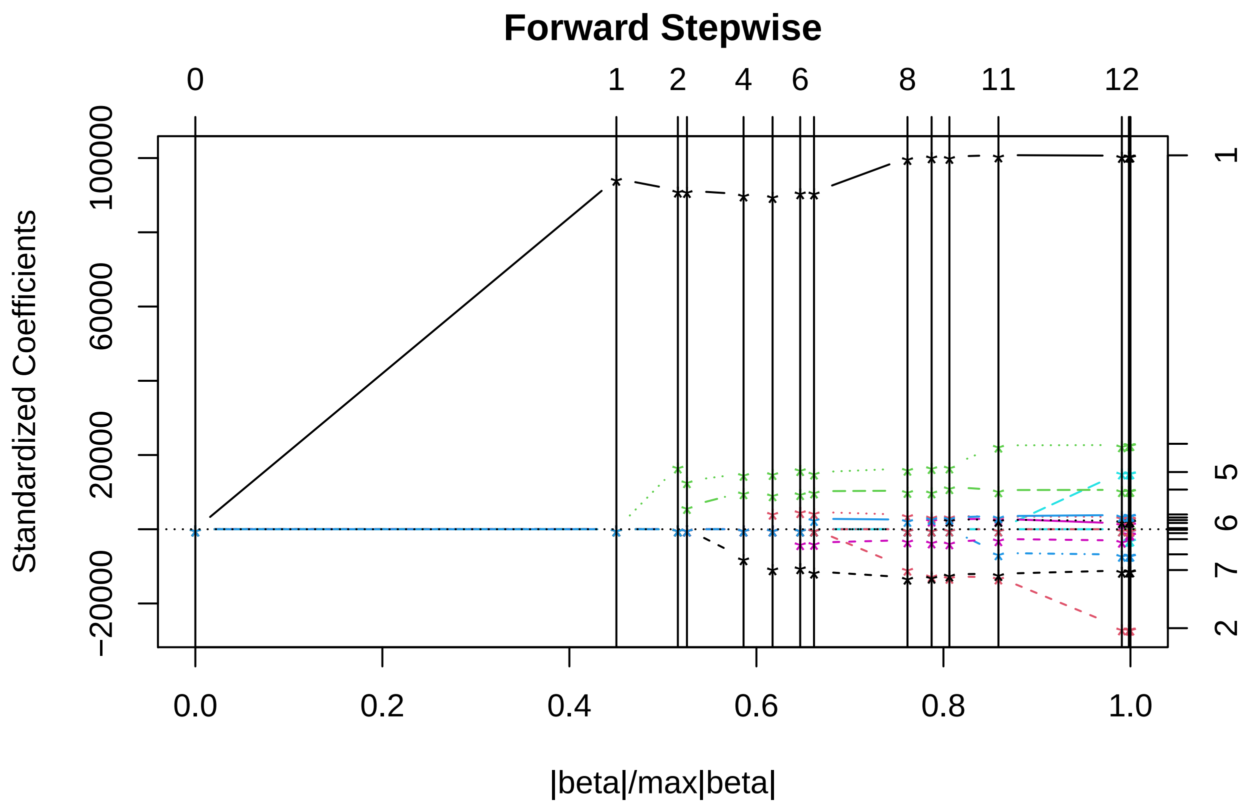

```r
HE.lasso
```

```
##
## Call:
## lars(x = Xtrain, y = ytrain, type = "lasso")
## R-squared: 0.926
## Sequence of LASSO moves:
##      Accept Top10perc Expend Outstate Room.Board Grad.Rate perc.alumni Terminal
## Var       1         3     15        7          8        16          14       12
## Step      1         2      3        4          5         6           7        8
##      PhD P.Undergrad S.F.Ratio Enroll Top25perc Books F.Undergrad Personal
## Var   11           6        13      2         4     9           5       10
## Step   9          10        11     12        13    14          15       16
```

we choose 1th, 3th, 15th, 7th, 8th, 16th entry

2(b)

```r
HE.step = lars(Xtrain , ytrain , type="step")

plot(HE.step)
```




```r
HE.step
```

```
##
## Call:
## lars(x = Xtrain, y = ytrain, type = "step")
## R-squared: 0.926
## Sequence of Forward Stepwise moves:
##      Accept Top10perc Expend Outstate Room.Board Terminal Grad.Rate Enroll
## Var       1         3     15        7          8       12        16      2
## Step      1         2      3        4          5        6         7      8
##      P.Undergrad S.F.Ratio Top25perc F.Undergrad PhD Books Personal perc.alumni
## Var            6        13         4           5  11     9       10          14
## Step           9        10        11          12  13    14       15          16
```

We obeserve the list of first 5 variable that enter is exactly the same for lasso and stepwise. they are very much similar

2(c)

```r
set.seed(444)
fit.cv = cv.glmnet(Xtrain,ytrain,nfolds = 10, type.measure = 'mse')
# choice of lambda
lambd=fit.cv$lambda.min
lambd
```

```
## [1] 2.058527
```

```r
fit.Lasso=glmnet(Xtrain , ytrain , alpha=1, lambda=lambd,
                 standardize=TRUE, intercept = TRUE ,  family = "gaussian")


predict.lasso = predict(fit.Lasso,newx=Xtest)
ndata=subset(data.test, select = -c(Apps))
predict.ls = predict(fit.ls, ndata)

t(predict.lasso-ytest)%*%(predict.lasso-ytest)/length(ytest)
```

```
##          s0
## s0 542333.7
```

2(d)

```r
# 10 fold cross-validation
set.seed(444)
alpha=seq(0,1,0.25)
CV.To.Plot=data.frame(alpha=NA,lambda=NA,MSE=NA)
for (i in 1:length(alpha)){
  cv10fold=cv.glmnet(Xtrain, ytrain, type.measure = "mse", nfolds=10,alpha=alpha[i])
  lambda=cv10fold$lambda.min
  mse=min(cv10fold$cvm)
  CV.To.Plot[i,]=c(alpha[i],lambda,mse)
}
lambd=CV.To.Plot[which.min(CV.To.Plot$MSE),]$lambda
alph=CV.To.Plot[which.min(CV.To.Plot$MSE),]$alpha
fit.Lasso=glmnet(Xtrain , ytrain , alpha=alph, lambda=lambd,
                 standardize=TRUE, intercept = TRUE ,  family = "gaussian")


predict.lasso = predict(fit.Lasso,newx=Xtest)
ndata=subset(data.test, select = -c(Apps))
predict.ls = predict(fit.ls, ndata)

t(predict.lasso-ytest)%*%(predict.lasso-ytest)/length(ytest)
```

```
##          s0
## s0 542587.4
```

3(a)

we have p variables and n datas\ we minimuze

$$\sum_{i=1}^{n}(y_i - X_i^T\beta)^2 + \lambda\sum_{j=1}^{p}(\beta_j^2)$$

which is

$$\sum_{i=1}^{n}(y_i - X_i^T\beta)^2 + \sum_{j=1}^{p}(0 - \sqrt{\lambda}\beta_j)^2$$

which can be expressed as matrix X^*

$$\begin{matrix} Z_{11} & Z_{12} & ... & Z_{1p} \\ Z_{21} & Z_{22} & ... & Z_{np} \\ \sqrt{\lambda} & 0 & ... & 0 \\ 0 & \sqrt{\lambda} & ... & 0 \\ 0 & 0 & ... & \sqrt{\lambda} \end{matrix}$$

and Y^* as $[Y_1, ..., Y_N, 0, 0...0]^T$

3(b)

```r
# scale X_test


Xscale=X[ , c(3,11,14)]
for (i in 1:ncol(Xscale)){
  Xscale[,i]=(Xscale[,i]-mean(Xscale[,i]))/sd(Xscale[,i])
}
inv=diag(ncol(Xscale))*10

Xscaled=rbind(Xscale,inv)
Y=c(y, rep(0,3))

length(Y)
```

```
## [1] 780
```

```r
nrow(X)
```

```
## [1] 777
```

```r
model=lm(Y~Xscaled)
summary(model)
```

```
##
## Call:
## lm(formula = Y ~ Xscaled)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -13381  -1799   -641    848  43222
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2976.2      122.9  24.214  < 2e-16 ***
## XscaledTop10perc      1040.5      140.2   7.419 3.10e-13 ***
## XscaledPhD            1033.9      131.6   7.858 1.30e-14 ***
## Xscaledperc.alumni    -991.2      126.8  -7.815 1.79e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3432 on 776 degrees of freedom
## Multiple R-squared:  0.2153, Adjusted R-squared:  0.2123
## F-statistic: 70.97 on 3 and 776 DF,  p-value: < 2.2e-16
```

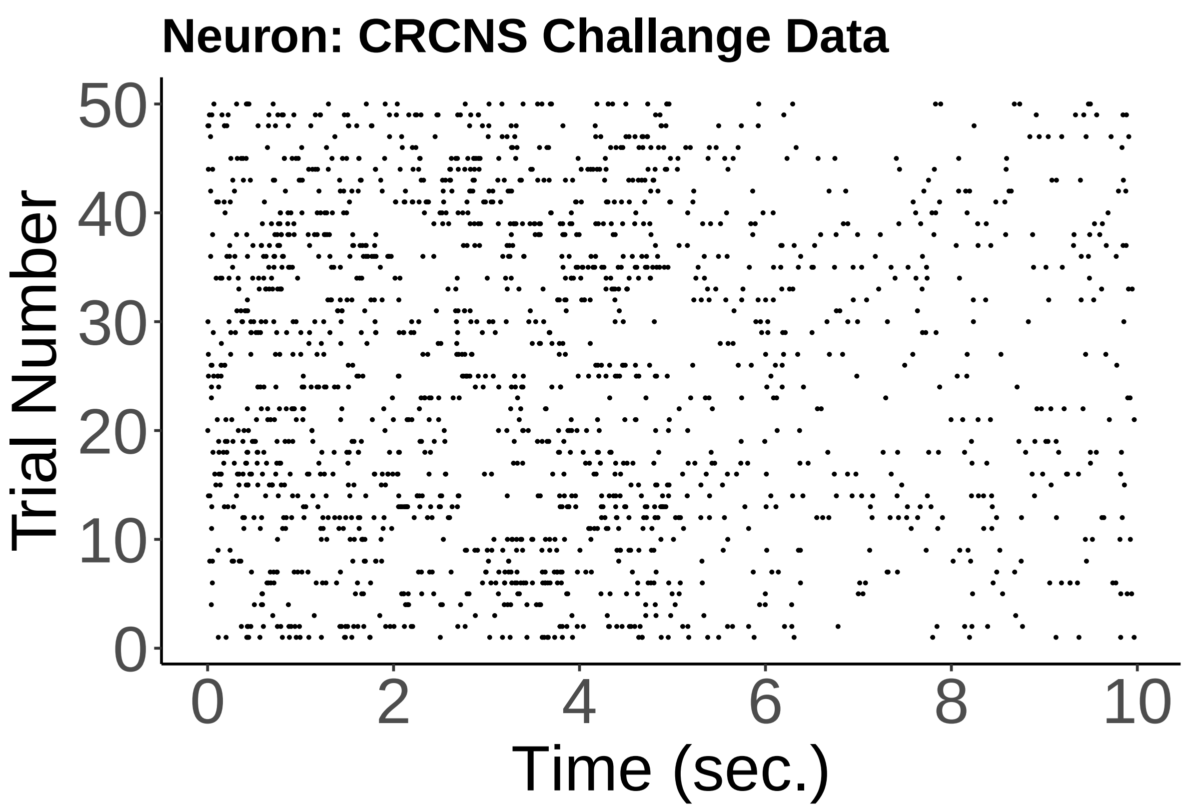
```r
library(MASS)
inv=diag(ncol(Xscale))*100
ginv(t(Xscale)%*%Xscale+inv)%*%t(Xscale)%*%y
```

```
##           [,1]
## [1,] 1054.6099
## [2,] 1056.1041
## [3,] -967.8126
```

we observe the result on $\hat{\beta}$ is almost exactly the same. the difference is due to intercept is ommit.

4(a)

```
library(mmnst)
data=read.csv('AuditoryCortexData.csv')
Data=list(c())
j=1
for(i in 1:50){
  Data[[i]] =  data[,i]
  good = complete.cases(Data[[i]])
  Data[[i]] = Data[[i]][good]

  # Data[[i]] =  Data[[j]][ Data[[j]]>0 &  Data[[j]]<10]
  i=i+1
}

RasterPlot("CRCNS Challange Data" , Data)
```

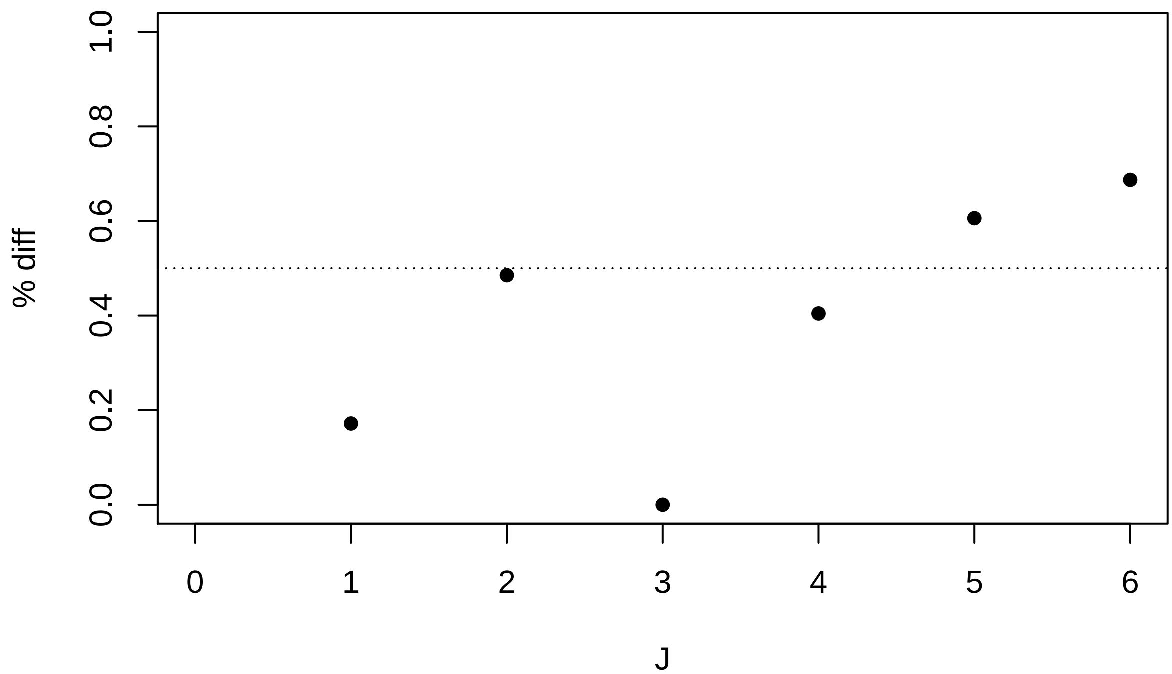


we dont observe strong wave light pattern. however, it's clear that from time 0-5, there is more spikes then time 6-10

(b)

```r
t.min = 0
t.max = 10
Unlist.Data=unlist(Data)
cv.output <- RDPCrossValidation(Data, 0 , t.max, max.J=6)
```

```
## J= 1
## J= 2
## J= 3
## J= 4
## J= 5
## J= 6
```




```r
cv.output$lambda.ISE #Optimum lambda
```

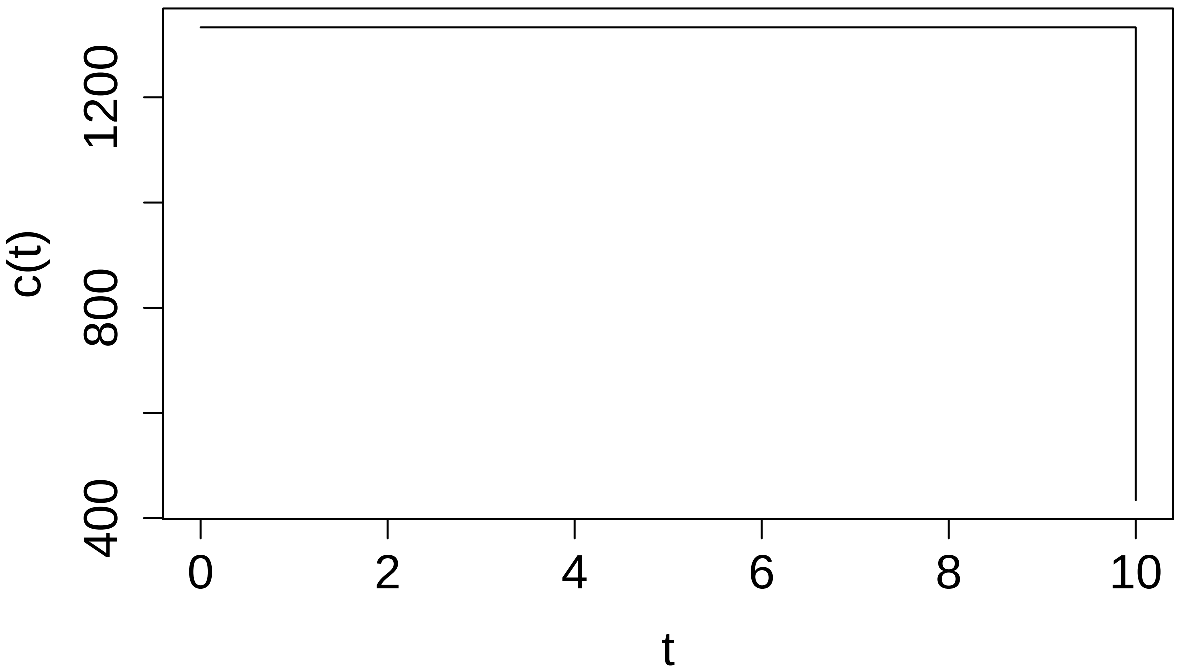
```
## [1] 0
```

```r
cv.output$J.ISE #Optimum J
```

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

```r
Terminal.Points = seq(t.min,t.max,length=2^cv.output$J.ISE+1)

Sig = sum(Unlist.Data<=Terminal.Points[2])
for(i in 3:length(Terminal.Points)){
  Sig[(i-1)] = sum(Unlist.Data<=Terminal.Points[i] &
                     Unlist.Data>Terminal.Points[(i-1)] )
}
```

```r
ct = PoissonRDP(Sig , cv.output$lambda.ISE)  # the original cv.output$lambda.ISE/log(length(Sig)) is in
t = seq(0,10,length=length(ct))
plot(ct~t,type="s", cex.axis=1.5 , cex.lab = 1.5 ,ylab="c(t)")
```

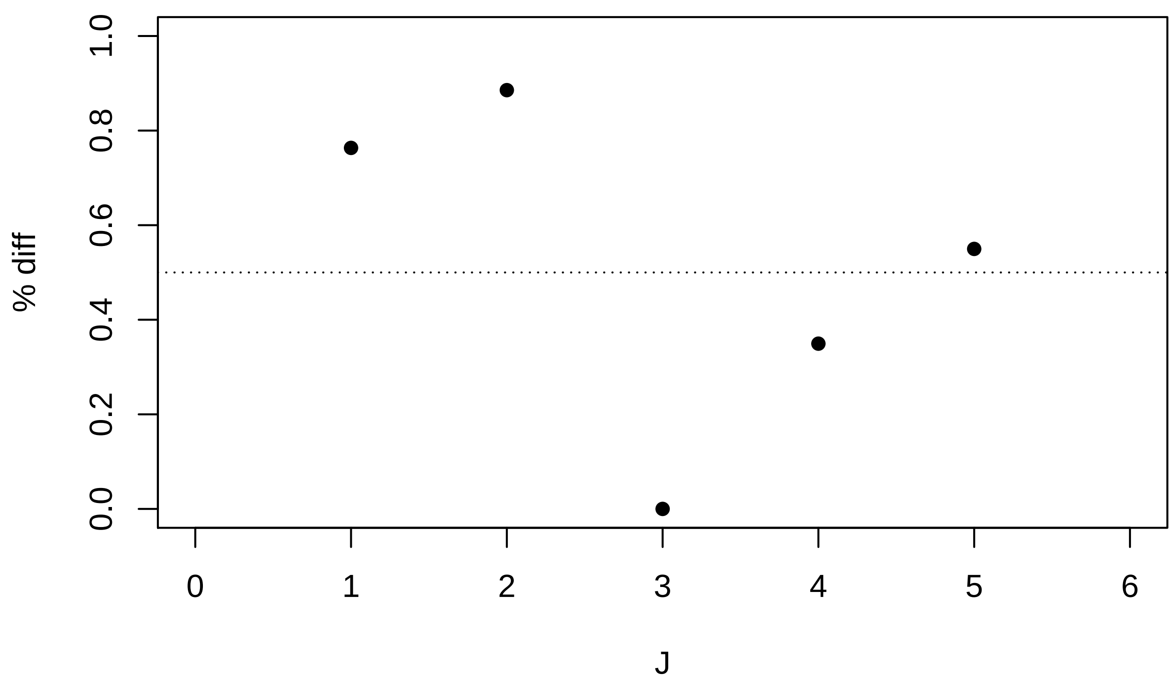



```r
ct1=ct
```

(c)

```
s=Sys.time()
pooldata=list()
uData=unlist(Data)
for (i in 1:length(unlist(Data))){
  pooldata[[i]]=uData[i]
}


cv.output <- RDPCrossValidation(pooldata, 0 , t.max, max.J=6,  poss.lambda = seq(0, 10, by = 2))
```

```
## J= 1
## J= 2
## J= 3
## J= 4
## J= 5
## J= 6
```




```
cv.output$J.ISE #Optimum J
```

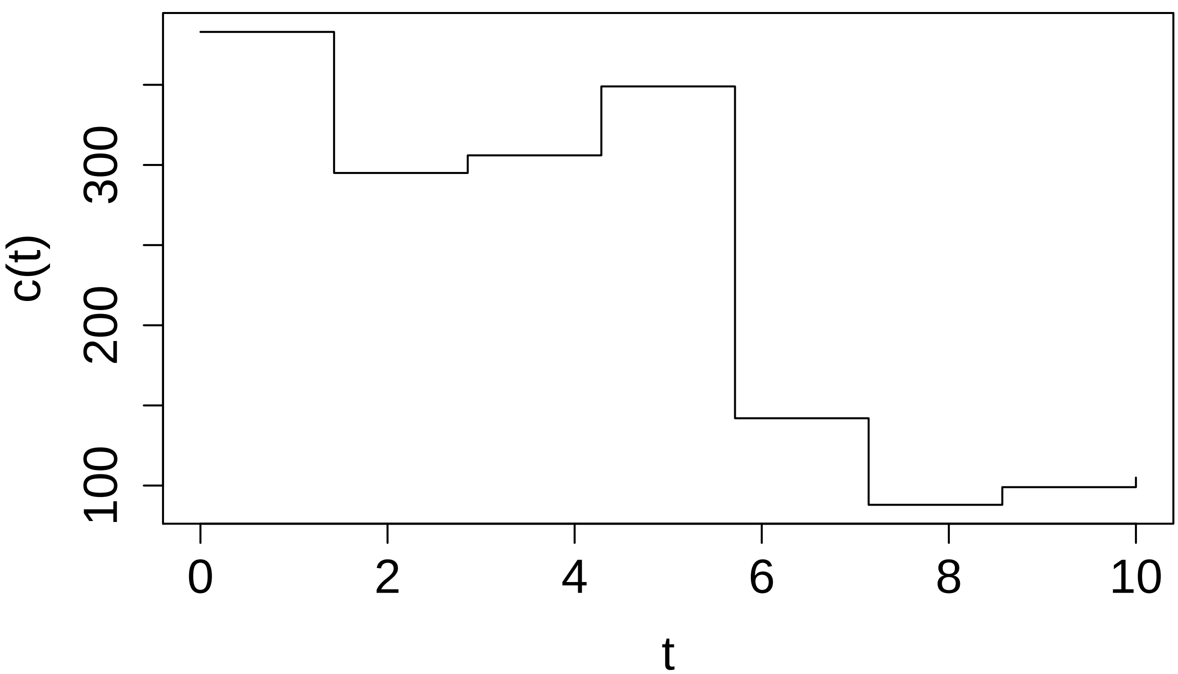
```
## [1] 3
```

```
cv.output$lambda.ISE#optimum lambda
```

```
## [1] 0
```

```
Terminal.Points = seq(t.min,t.max,length=2^cv.output$J.ISE+1)
```

```
Sig = sum(Unlist.Data<=Terminal.Points[2])
for(i in 3:length(Terminal.Points)){
  Sig[(i-1)] = sum(Unlist.Data<=Terminal.Points[i] &
                   Unlist.Data>Terminal.Points[(i-1)] )
}
ct = PoissonRDP(Sig , cv.output$lambda.ISE)  # the original cv.output$lambda.ISE/log(length(Sig)) is in
ct2=ct
t = seq(0,10,length=length(ct))
plot(ct~t,type="s", cex.axis=1.5 , cex.lab = 1.5 ,ylab="c(t)")
```

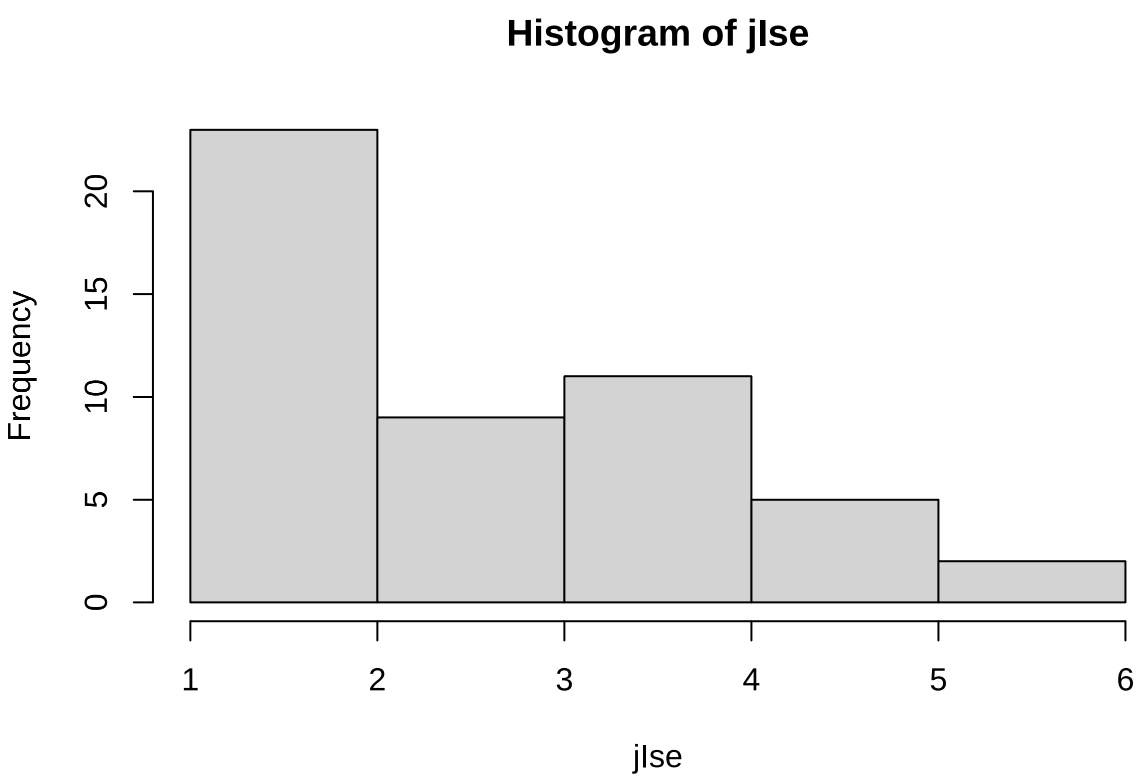



```
e=Sys.time()
e-s
```

```
## Time difference of 1.288487 mins
```

(d)

```
jIse=c()
lambdaIse=c()
for (i in 1:50){
  pdatai=list()
  ii=Data[[i]]
  for (j in 1:length(ii)){
    pdatai[[j]]=ii[j]
  }
  t.min=0
  t.max=10
  rdpcv=RDPCrossValidation(pdatai, t.min , t.max, max.J=6,  poss.lambda = seq(0, 5, by = 0.5),print.J.va
  jIse=c(jIse,rdpcv$J.ISE)
  lambdaIse=c(lambdaIse, rdpcv$lambda.ISE)
}

getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

hist(jIse)
```




```
table(jIse)
```

```
## jIse
```

```
## 1  2  3  4  5  6
## 14  9  9 11  5  2
```

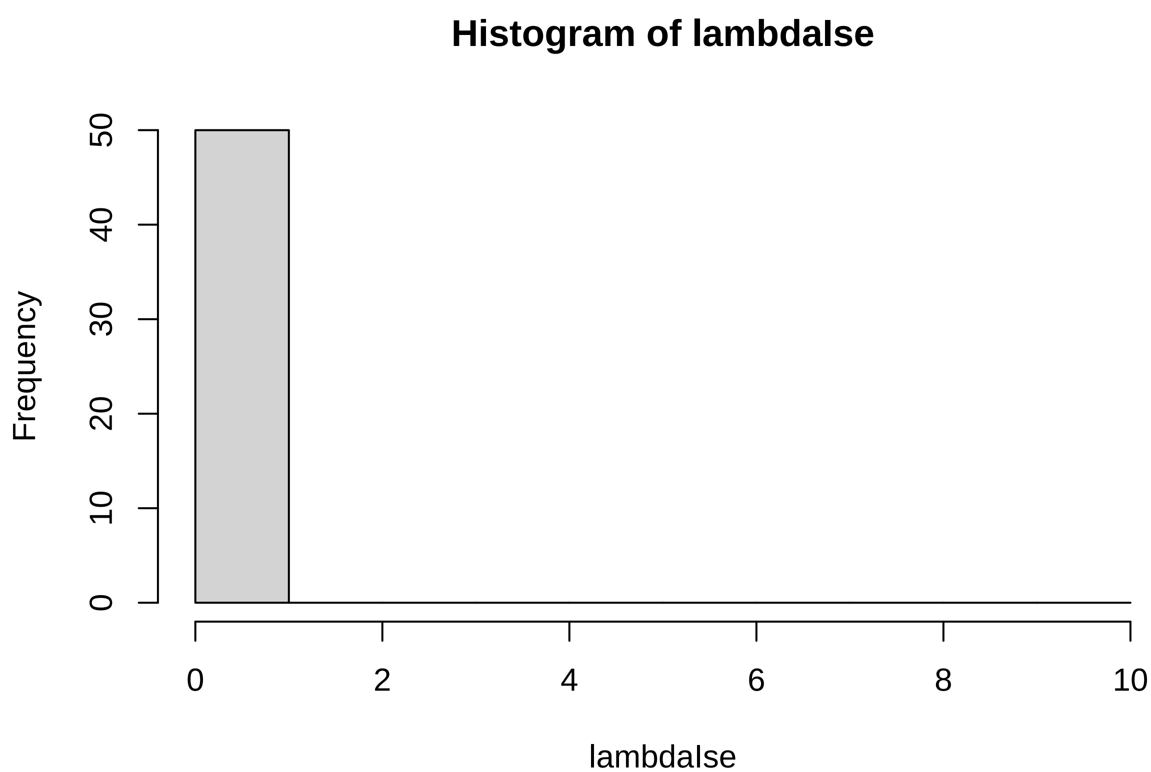
```r
# best lambda
blambda=mean(lambdaIse)
blambda
```

```
## [1] 0
```

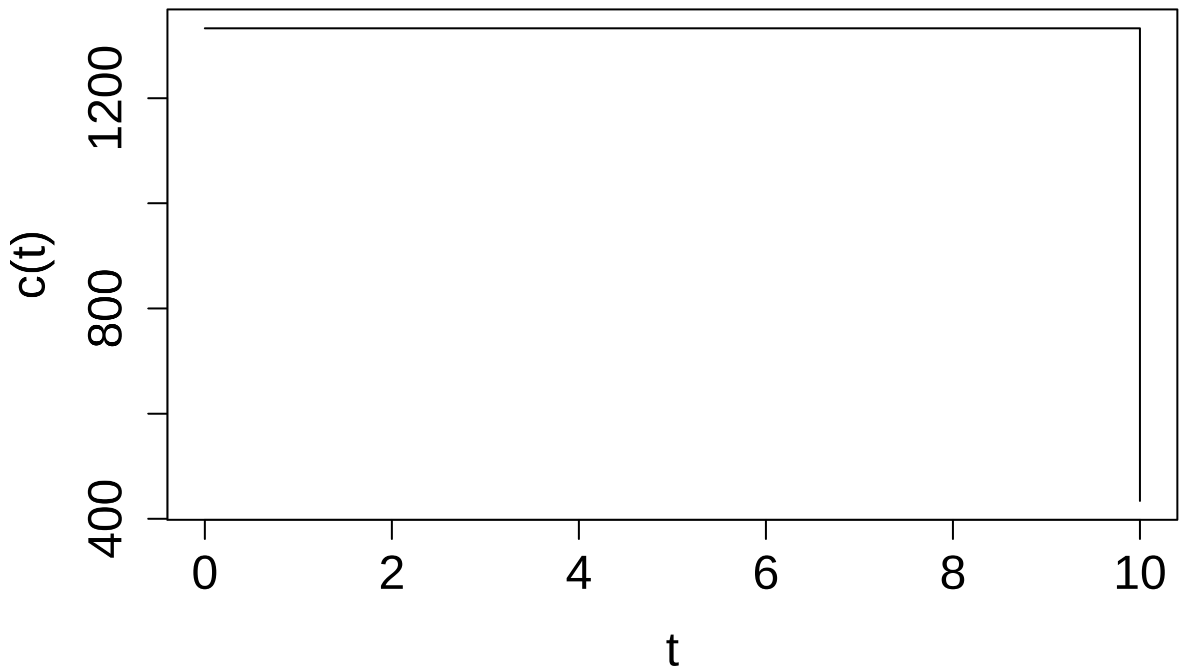
```r
# best j
bise=getmode(jIse)
bise
```

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

```r
hist(lambdaIse, breaks=seq(min(lambdaIse),10))
```

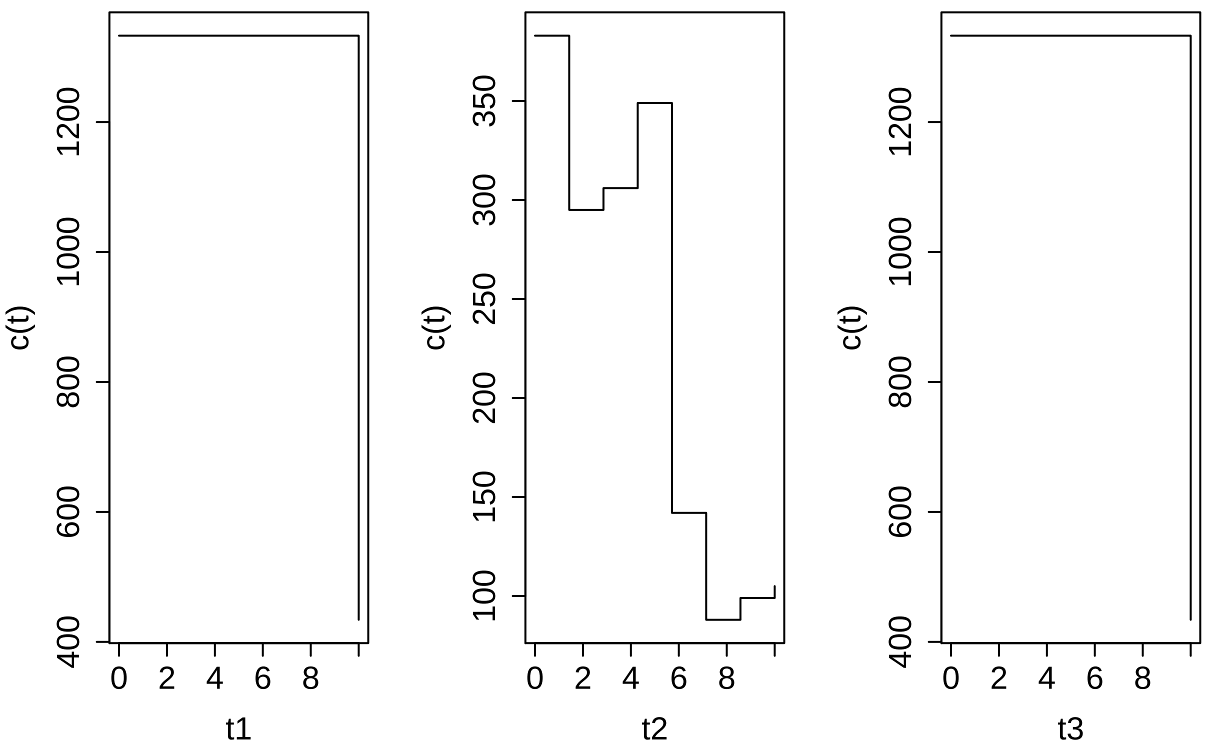



```r
Terminal.Points = seq(t.min,t.max,length=2^bise+1)

Sig = sum(Unlist.Data<=Terminal.Points[2])
for(i in 3:length(Terminal.Points)){
  Sig[(i-1)] = sum(Unlist.Data<=Terminal.Points[i] &
                   Unlist.Data>Terminal.Points[(i-1)] )
}
ct = PoissonRDP(Sig , blambda)  # the original cv.output$lambda.ISE/log(length(Sig)) is incorrect
ct3=ct
t = seq(0,10,length=length(ct))
plot(ct~t,type="s", cex.axis=1.5 , cex.lab = 1.5 ,ylab="c(t)")
```


c(t)
1200
800
400
0
2
4
6
8
10
t

(e)

```
#draw some graphs
t1 = seq(0,10,length=length(ct1))
t2 = seq(0,10,length=length(ct2))
t3 = seq(0,10,length=length(ct3))

par(mfrow=c(1,3))
plot(ct1~t1,type="s", cex.axis=1.5 , cex.lab = 1.5 ,ylab="c(t)")
plot(ct2~t2,type="s", cex.axis=1.5 , cex.lab = 1.5 ,ylab="c(t)")
plot(ct3~t3,type="s", cex.axis=1.5 , cex.lab = 1.5 ,ylab="c(t)")
```

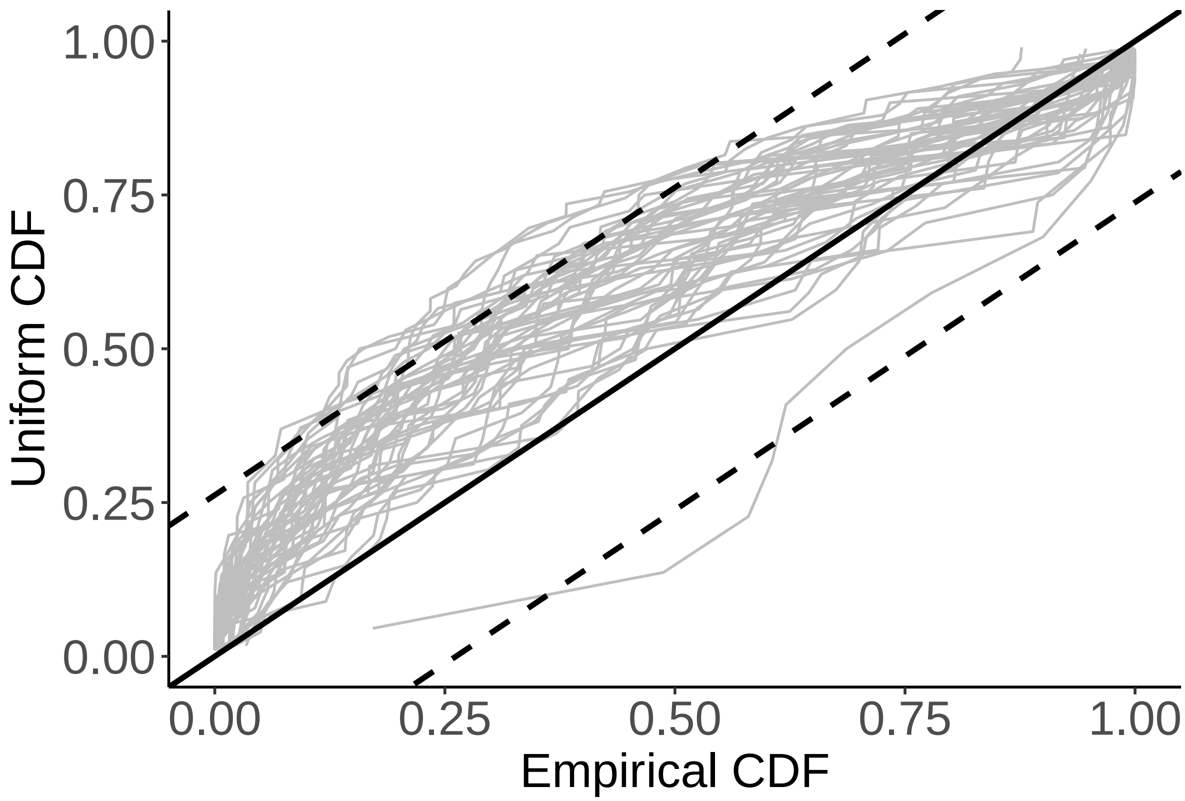


(f)

```r
# draw gof graph for second fit
ct<-FindCt(Data, 0 , 10, 0, 3)

#Setting the fits for each trial to a funcion and calculating it at 500 points
t = seq(t.min,t.max,length=500)
theta = matrix(NA,nrow=500,ncol=dim(ct[[2]])[1])
Terminal.Points = seq(t.min,t.max,length=2^3+1)
for(i in 1:ncol(theta)){
  ct.function.i = stepfun(Terminal.Points , c(0,ct[[2]][i,],0))
  theta[,i] = ct.function.i(t) # the original FindCt[,i] = ct.function.i(t) is incorrect
}

# Goodness of fit test
GOFPlot(
  Data,
  theta,
  t.start = t.min,
  t.end = t.max,
  neuron.name = NULL,
  resolution = (t.max - t.min)/(length(theta) - 1),
  axis.label.size = 18,
  title.size = 24
)
```

```
## total count = 25000
## 5000 bins processed
## 10000 bins processed
## 15000 bins processed
## 20000 bins processed
```

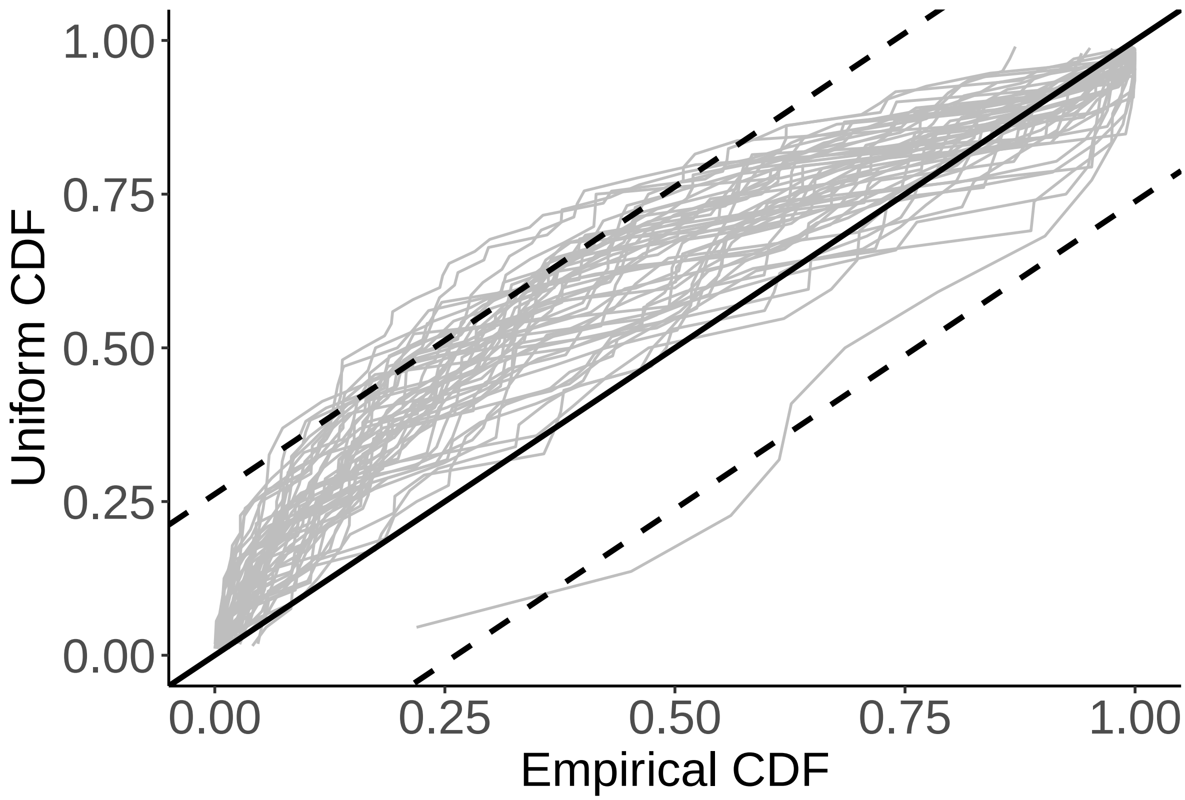
```
## fANCOVA 0.5-1 loaded
```




```r
# gof of first/third fit
ct<-FindCt(Data, 0 , 10, 0, 1)

#Setting the fits for each trial to a funcion and calculating it at 500 points
t = seq(t.min,t.max,length=500)
theta = matrix(NA,nrow=500,ncol=dim(ct[[2]])[1])
Terminal.Points = seq(t.min,t.max,length=2^1+1)
for(i in 1:ncol(theta)){
  ct.function.i = stepfun(Terminal.Points , c(0,ct[[2]][i,],0))
  theta[,i] = ct.function.i(t) # the original FindCt[,i] = ct.function.i(t) is incorrect
}

# Goodness of fit test
GOFPlot(
  Data,
  theta,
  t.start = t.min,
  t.end = t.max,
  neuron.name = NULL,
  resolution = (t.max - t.min)/(length(theta) - 1),
  axis.label.size = 18,
  title.size = 24
)
```

```
## total count = 25000
## 5000 bins processed
```

```
## 10000 bins processed
## 15000 bins processed
## 20000 bins processed
```



we don't observe a strong pattern in gof plot. However 2 bins histograms does not give much information and looks wierd, so j=3 seems a better choice. so the model in part(c) is more appealing. This does not mean part(a) and (d) are not good fit. As the data itself has no strong pattern except there's less spikes after 5 seconds.