a4

Mushi Wang

04/08/2020

q1 (a) If we have multicollinearity, the estimation of coefficients will be inaccurate and the variance of the estimators of coefficients will be very large.

(b)

```r
HigherEducation = read.csv("HigherEducation.csv")
HigherEducation_Modelling = HigherEducation[1:600,]
HigherEducation_Test = HigherEducation[601:777,]
```

```r
library(car)
```

```
## Loading required package: carData
```

```r
vif(lm(HigherEducation_Modelling[,1] ~ ., data = HigherEducation_Modelling[,-1]))
```

```
##      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
```

There are several multicollinearity issues. Accept, Enroll, Top10perc, Top25perc, F.Undergrad are problematic. Since all of them have a very high VIF, which indicates $R^2_{X_j|X_{-j}}$ is close to one and there is a linear relationship between the variables and the other 16 variables.

(c)

```r
library(glmnet)
```

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

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

```r
x = as.matrix(HigherEducation_Modelling[,-1])
y = HigherEducation_Modelling$Apps

fit.Lasso=glmnet(x, y, lambda=35, family = "gaussian")
coef(fit.Lasso)
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept) -9.644157e+02
## Accept       1.432557e+00
## Enroll       .
## Top10perc    3.256394e+01
## Top25perc    .
## F.Undergrad  .
## P.Undergrad  5.339685e-03
## Outstate    -6.597971e-02
## Room.Board   8.111502e-02
## Books        .
## Personal     .
## PhD         -2.291504e+00
## Terminal    -2.076292e+00
## S.F.Ratio    4.202810e+00
## perc.alumni -2.254493e+00
## Expend       6.242130e-02
## Grad.Rate    3.823482e+00
```

Accept, Top10perc, P.Undergrad, Outstate, Room.Board, PhD, Terminal, S.F.Ratio, perc.alumni, Expend and Grad.Rate are selected. It is a little different to the model in part (b). Two problematic variables, Accept and Top10perc, are in the lasso model. But the lasso model elimnates the variables with very high VIF.

(d)

```
rss1 = sum((HigherEducation_Test$Apps - predict(lm(y ~ ., data = HigherEducation_Modelling[,-1]), Higher
rss2 = sum((HigherEducation_Test$Apps - predict(fit.Lasso, newx = as.matrix(HigherEducation_Test[,-1]))

rss1
```

```
## [1] 98553375
```

```
rss2
```

```
## [1] 92792554
```

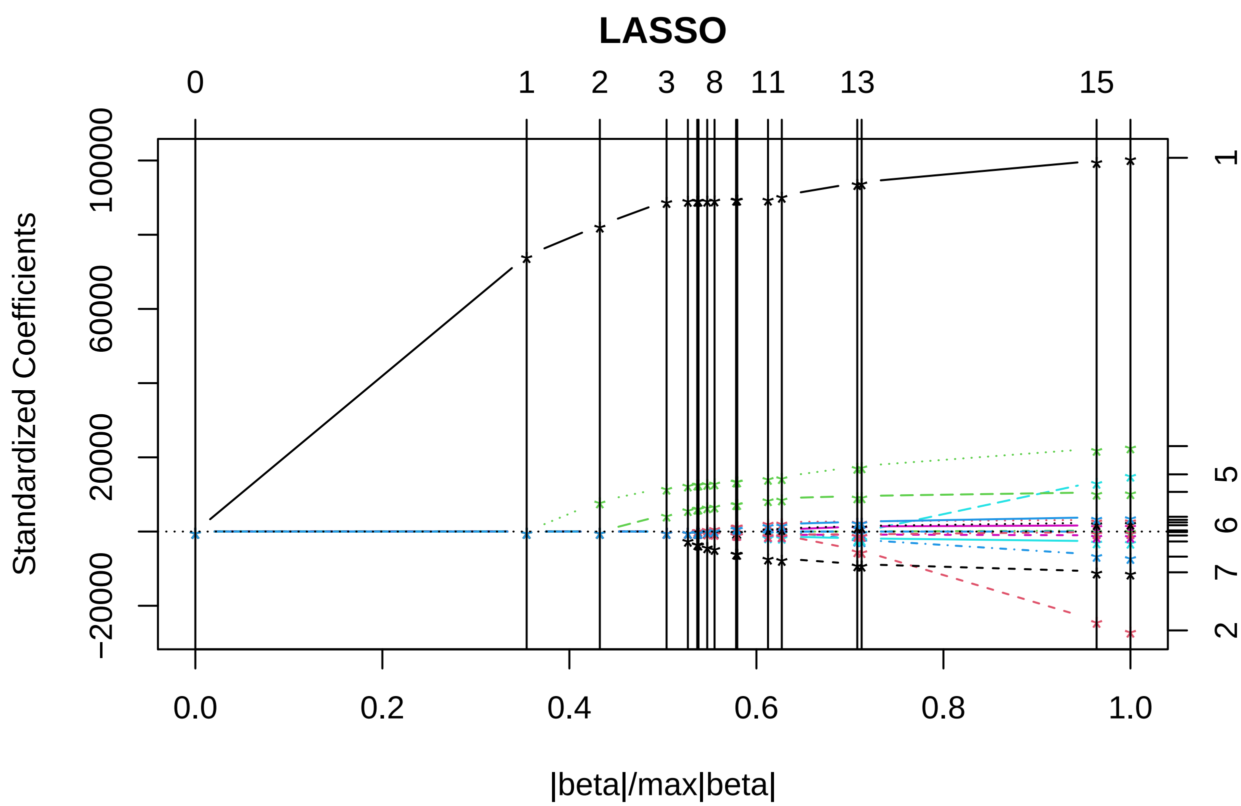
THe residuals sum of squares of model in part (b) is 98553375.
THe residuals sum of squares of model in part (c) is 92792554.
The model in part(c) is much better.

q2 (a)

```r
library(lars)
```

```
## Loaded lars 1.2
```

```r
edu_matrix = as.matrix(HigherEducation_Modelling)
lasso = lars(edu_matrix[,-1], HigherEducation_Modelling$Apps, type = "lasso")
plot(lasso)
```




```r
lasso$actions[1:6]
```

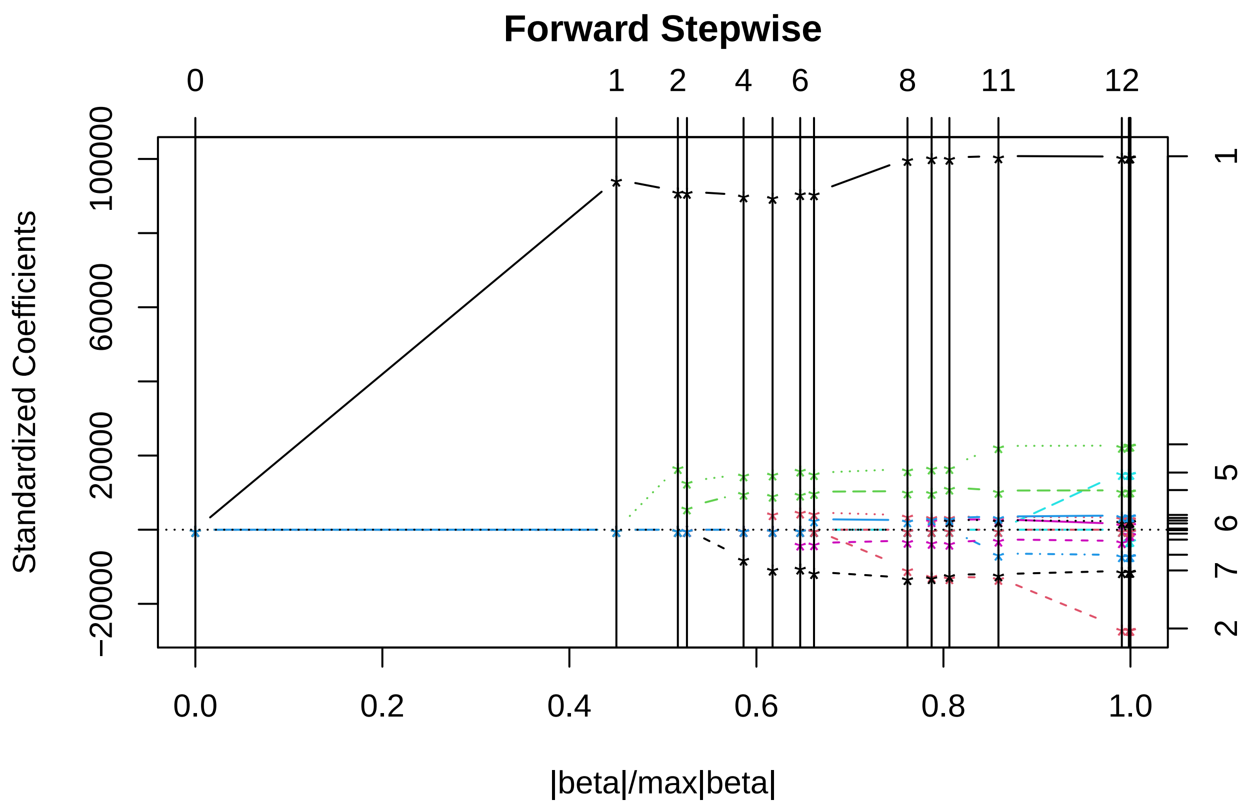
```
## [[1]]
## Accept
##      1
##
## [[2]]
## Top10perc
##         3
##
## [[3]]
## Expend
##     15
##
```

```
## [[4]]
## Outstate
##        7
##
## [[5]]
## Room.Board
##          8
##
## [[6]]
## Grad.Rate
##        16
```

We select accept, Top10perc, Expend, Outstate, Room.Board, Grad.Rate.

(b)

```r
fw_step = lars(edu_matrix[,-1], HigherEducation_Modelling$Apps, type = "step")
plot(fw_step)
```




```r
fw_step$actions
```

```
## [[1]]
## Accept
##      1
##
## [[2]]
## Top10perc
##         3
##
## [[3]]
## Expend
##     15
##
## [[4]]
## Outstate
##        7
##
## [[5]]
## Room.Board
```

```
##          8
##
## [[6]]
## Terminal
##       12
##
## [[7]]
## Grad.Rate
##        16
##
## [[8]]
## Enroll
##      2
##
## [[9]]
## P.Undergrad
##           6
##
## [[10]]
## S.F.Ratio
##        13
##
## [[11]]
## Top25perc
##         4
##
## [[12]]
## F.Undergrad
##           5
##
## [[13]]
## PhD
##  11
##
## [[14]]
## Books
##     9
##
## [[15]]
## Personal
##       10
##
## [[16]]
## perc.alumni
##          14
```

```
lasso$actions
```

```
## [[1]]
## Accept
##      1
##
## [[2]]
## Top10perc
```

```
##         3
##
## [[3]]
## Expend
##     15
##
## [[4]]
## Outstate
##        7
##
## [[5]]
## Room.Board
##          8
##
## [[6]]
## Grad.Rate
##        16
##
## [[7]]
## perc.alumni
##          14
##
## [[8]]
## Terminal
##       12
##
## [[9]]
## PhD
##  11
##
## [[10]]
## P.Undergrad
##           6
##
## [[11]]
## S.F.Ratio
##        13
##
## [[12]]
## Enroll
##      2
##
## [[13]]
## Top25perc
##         4
##
## [[14]]
## Books
##     9
##
## [[15]]
## F.Undergrad
##           5
##
```

```
## [[16]]
## Personal
##       10
```

Both two models select the same first five variables. But starting from the sixth variable, the selections are quite different.

(c)

```r
set.seed(444)
min_lam = cv.glmnet(x, y, alpha = 1, family = "gaussian")$lambda.min
min_lam
```

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

```r
fit.Lasso2 = glmnet(x, y, lambda=min_lam, family = "gaussian")
sum((HigherEducation_Test$Apps - predict(fit.Lasso2, newx = as.matrix(HigherEducation_Test[,-1])))^2)
```

```
## [1] 95993059
```

(d)

```r
set.seed(444)
alphas = c(0, 0.25, 0.5, 0.75, 1)
lambda_min = rep(0, length(alphas))

for(i in 1:length(alphas)) {
  lambda_min[i] = cv.glmnet(x, y, alpha = alphas[i], family = "gaussian")$lambda.min
}
min(lambda_min)
```

```
## [1] 2.50087
```

```r
alphas[which.min(lambda_min)]
```

```
## [1] 0.75
```

```r
fit.Lasso3 = glmnet(x, y, lambda = min(lambda_min),
                    alpha = alphas[which.min(lambda_min)], family = "gaussian")
sum((HigherEducation_Test$Apps -
       predict(fit.Lasso3, newx = as.matrix(HigherEducation_Test[,-1])))^2)
```

```
## [1] 96037975
```

q3 (a) Let $X_{aug} = \sqrt{\lambda} I_{p\times p}$ and $Y_{aug} = \vec{0}_{p\times p}$.
So

$$X^* = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ X_{aug} \end{bmatrix}$$

where $x_i = [x_{i1}, \cdots, x_{ip}]$
and

$$Y^* = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_{aug} \end{bmatrix}$$

Let $n' = n + dim(X_{aug})$
Hence,

$$\begin{aligned} Rss(\lambda) &= \sum_{i=1}^{n'} (Y_i^* - X^*\beta)^2 \\ &= \sum_{i=1}^{n'} (Y_i - \beta_1 x_{i1} - \cdots - \beta_n x_{ip})^2 \\ &= \sum_{i=1}^{n} (Y_i - X\beta)^2 + (0 - \sqrt{\lambda}\beta_1)^2 + \cdots + (0 - \sqrt{\lambda}\beta_p)^2 \\ &= \sum_{i=1}^{n} (Y_i - X\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \end{aligned}$$

Since, $Rss(\lambda) = Rss_{Ridge}(\lambda)$,
$\hat{\beta}_{Ridge} = ({X^*}^T X^*)^{-1} {X^*}^T Y^* = (X^T X + \lambda I)^{-1} X^T Y$

(b)

```r
standardize = function(x) {
  return((x - mean(x)) / sd(x))
}

lm_ridge = data.frame(Apps = HigherEducation$Apps,
                      Top10perc = standardize(HigherEducation$Top10perc),
                      PhD = standardize(HigherEducation$PhD),
                      perc.alumni = standardize(HigherEducation$perc.alumni))
lambda = 100
x_aug = sqrt(lambda) * diag(3)
aug = data.frame(Apps = rep(0, 3), Top10perc = x_aug[,1], PhD = x_aug[,2], perc.alumni = x_aug[,3])
lm_ridge_aug = rbind(lm_ridge, aug)
summary(lm(lm_ridge_aug$Apps ~ . -1, data = lm_ridge_aug[, -1]))
```

```
##
## Call:
## lm(formula = lm_ridge_aug$Apps ~ . - 1, data = lm_ridge_aug[,
##     -1])
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -10561   1176   2324   3889  46175
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## Top10perc     1054.6      185.7   5.679 1.92e-08 ***
## PhD           1056.1      174.2   6.062 2.09e-09 ***
## perc.alumni   -967.8      167.9  -5.763 1.19e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4545 on 777 degrees of freedom
## Multiple R-squared:  0.1382, Adjusted R-squared:  0.1349
## F-statistic: 41.55 on 3 and 777 DF,  p-value: < 2.2e-16
```

```r
solve(t(as.matrix(lm_ridge[, -1])) %*% as.matrix(lm_ridge[, -1]) + lambda * diag(3)) %*%
t(as.matrix(lm_ridge[, -1])) %*% lm_ridge[, 1]
```

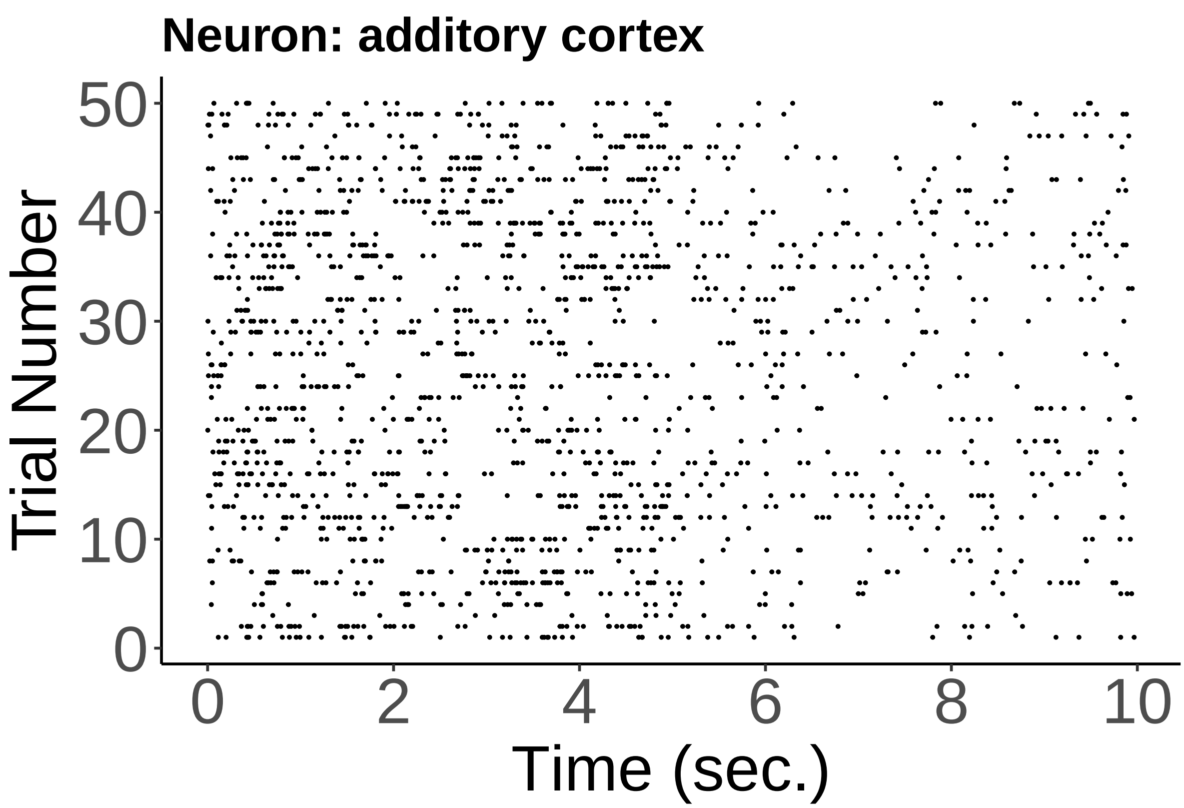
```
##                  [,1]
## Top10perc   1054.6099
## PhD         1056.1041
## perc.alumni -967.8126
```

They are the same. Since $Rss(\lambda)$ is the same by part (a).

q4 (a)

```
library(mmnst)
acd = read.csv("AuditoryCortexData.csv")

removeNA = function(acd) {
  v = vector("list", length = ncol(acd))
  for(i in 1:ncol(acd)) {
    temp = c()
    for(j in 1:nrow(acd)) {
      if(! is.na(acd[j, i])) {
        temp = append(temp, acd[j, i])
      }
    }
    v[[i]] = temp
  }
  return(v)
}
RasterPlot("additory cortex", removeNA(acd))
```



it is homogeneous.

(b)

```
acd_list = removeNA(acd)
cv.output1 = RDPCrossValidation(acd_list, t.end = 10, max.J = 6, pct.diff.plot = FALSE, print.J.value =
cv.output1$lambda.ISE
```

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

```
cv.output1$J.ISE
```

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

(c)

```
Unlist.Data = sort(unlist(acd_list))
t.min = floor(min(Unlist.Data))
t.max = ceiling(max(Unlist.Data))
cv.output2 = RDPCrossValidation(Unlist.Data, t.min , t.max , poss.lambda = seq(0, 5, by = 0.1), max.J =
                                pct.diff.plot = FALSE, print.J.value = FALSE)
cv.output2$lambda.ISE
```

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

```
cv.output2$J.ISE
```

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

(d)

```r
cvs = data.frame(lambda = rep(0, ncol(acd)), N = rep(0, ncol(acd)))
for(i in 1: ncol(acd)) {
  cv.out = RDPCrossValidation(acd_list[[i]], t.end = 10, max.J = 6,
                              pct.diff.plot = FALSE, print.J.value = FALSE)
  cvs[i, 1] = cv.out$lambda.ISE
  cvs[i, 2] = 2^cv.out$J.ISE
}

Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

Mode(cvs$N)
```

```
## [1] 2
```

```r
mean(cvs$lambda)
```

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

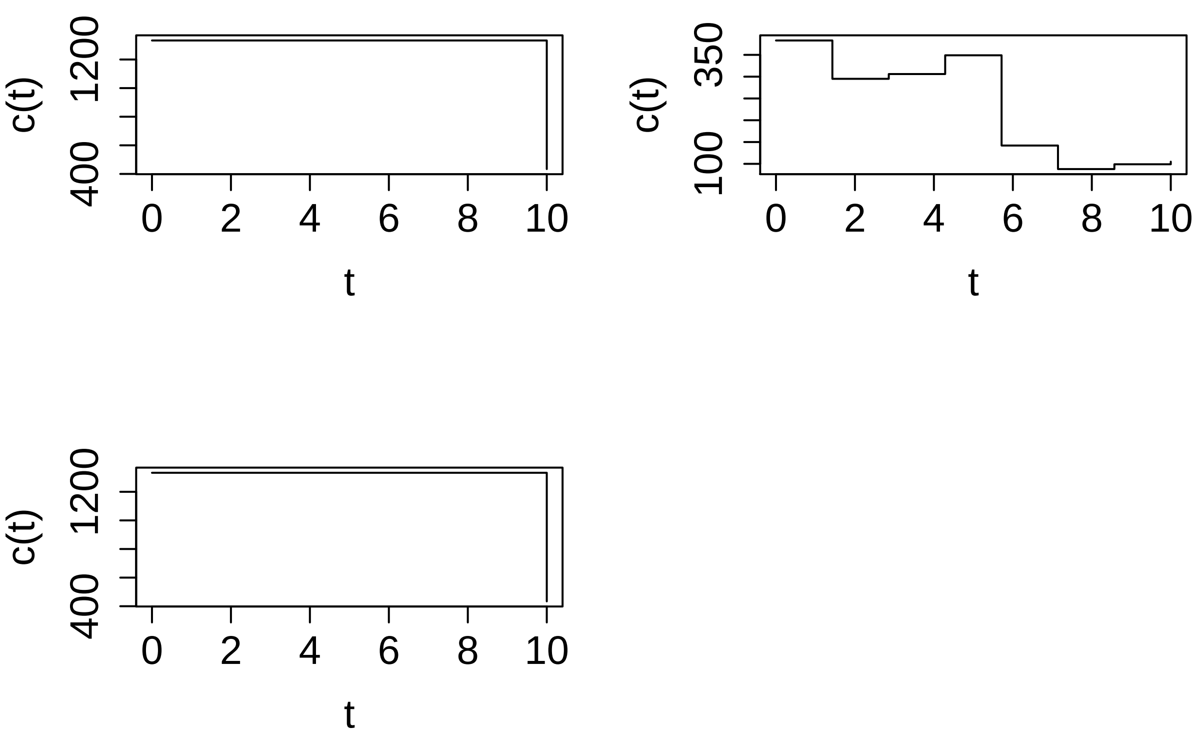
(e)

```
plotIF = function(cv.output) {
  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
  t = seq(0,10,length=length(ct))
  plot(ct~t,type="s", cex.axis=1.5 , cex.lab = 1.5 ,ylab="c(t)")
}

par(mfrow = c(2, 2))
plotIF(cv.output1)
plotIF(cv.output2)
plotIF(list(lambda.ISE = cvs$lambda, J.ISE = 1))
```

```
## Warning in bestFit[xind] <- bestFit[xind] * (pl1 > pl0) + bestFit2 * (pl1 <= :
## number of items to replace is not a multiple of replacement length
```



The models from (b) and (d) are very similar, the model from (c) is very different to others.

(f)

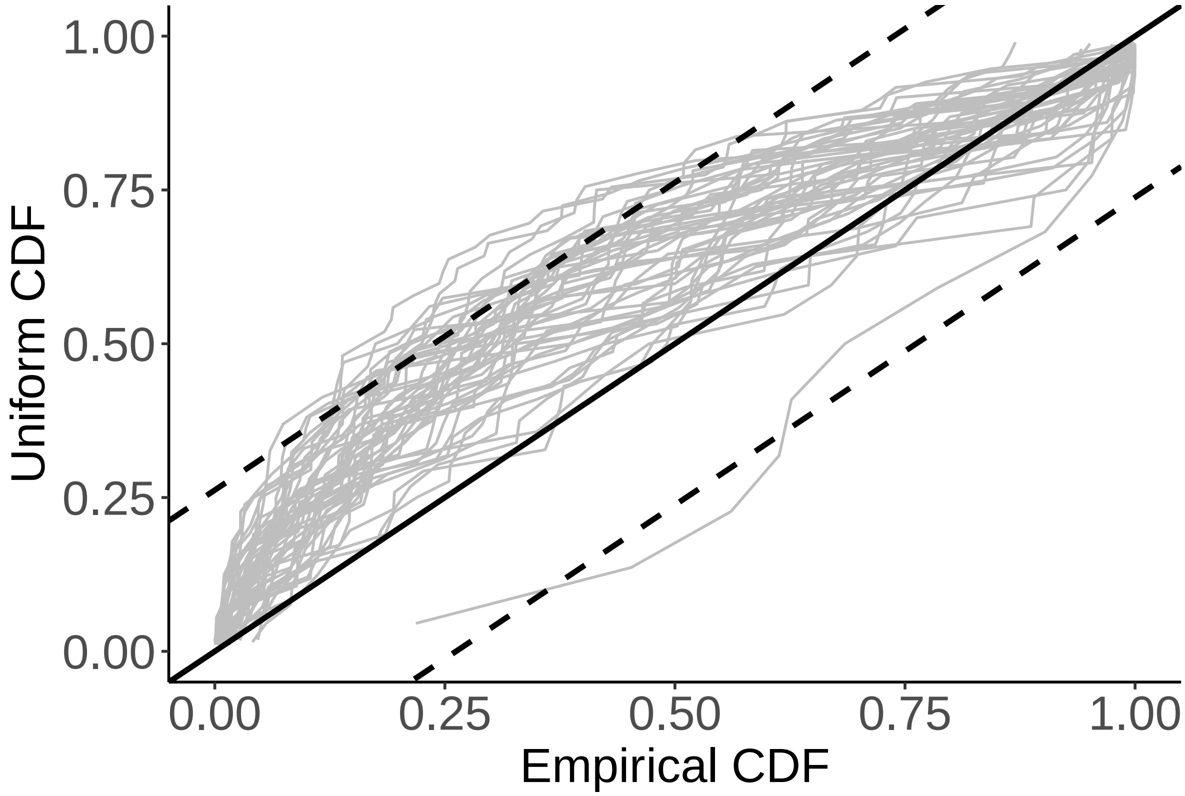
```r
ct<-FindCt(acd_list, t.min , t.max, cv.output1$lambda.ISE,cv.output1$J.ISE)

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^cv.output1$J.ISE+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)
}

GOFPlot(
  acd_list,
  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
```

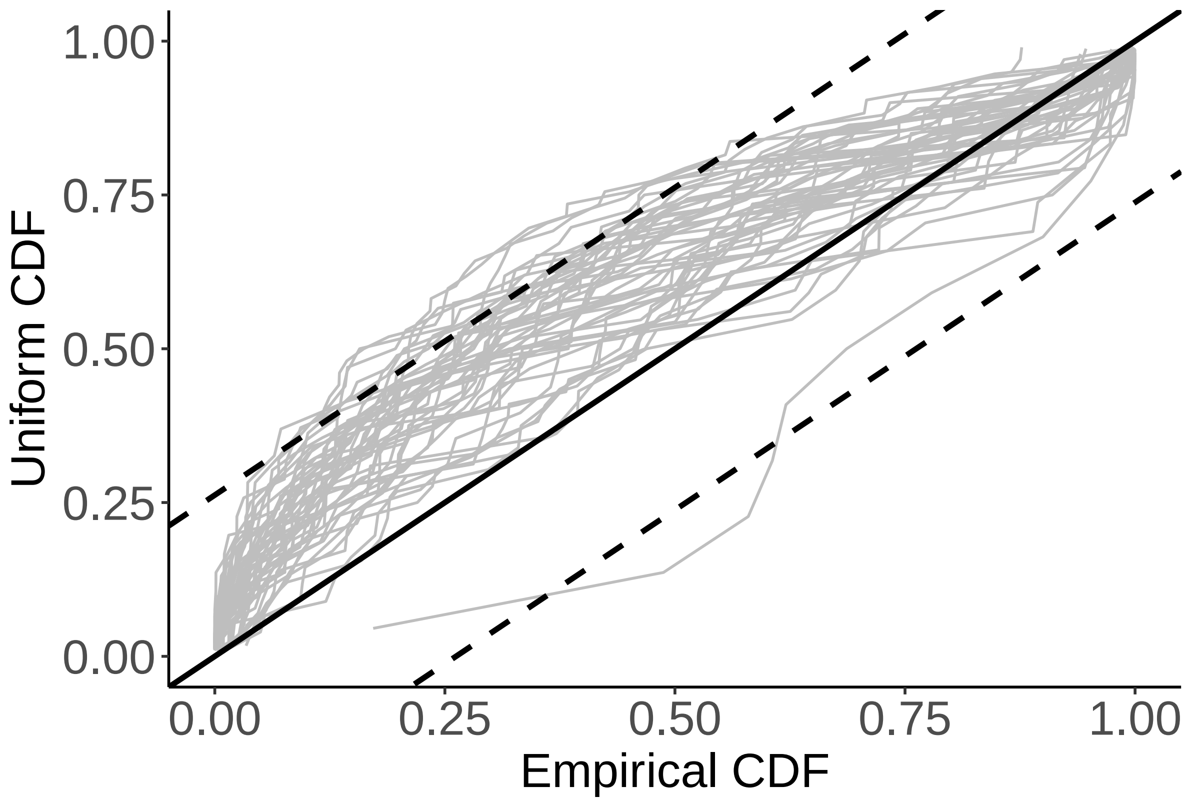



```
ct<-FindCt(acd_list, t.min , t.max, cv.output2$lambda.ISE,cv.output2$J.ISE)

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^cv.output2$J.ISE+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)
}

GOFPlot(
  acd_list,
  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
```

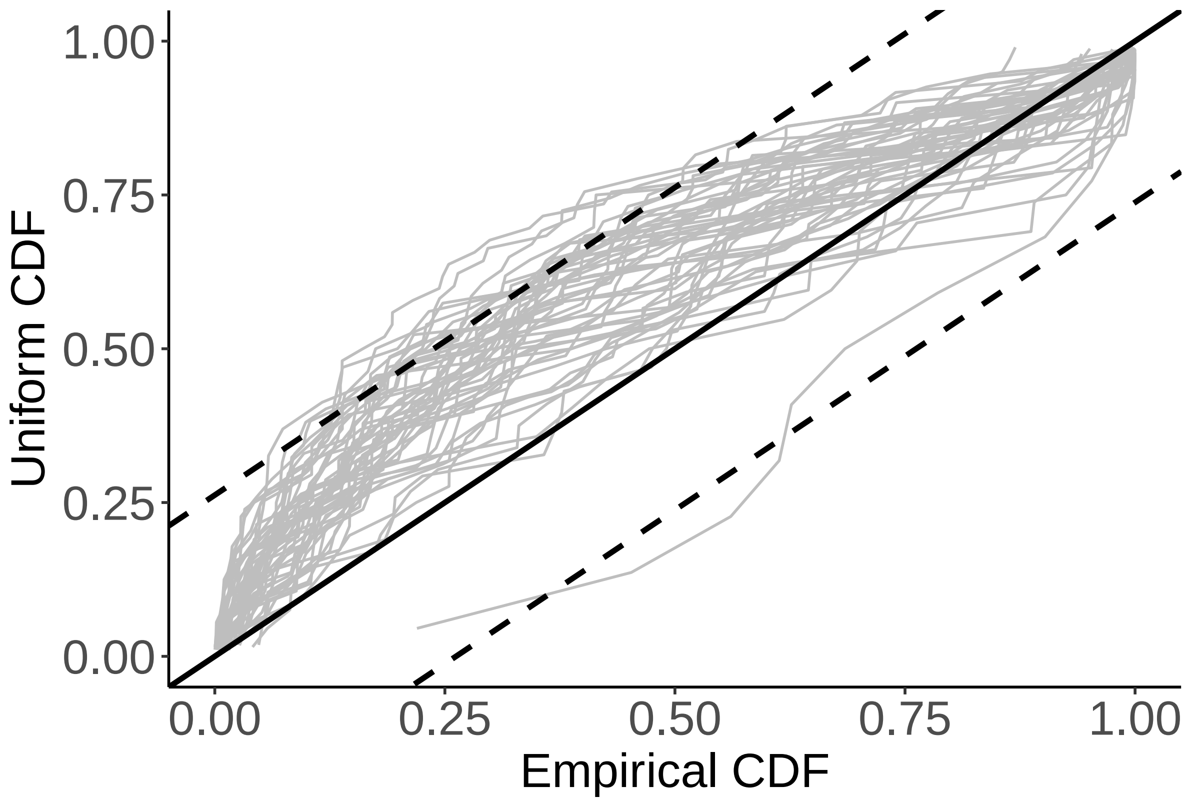



```
ct<-FindCt(acd_list, t.min , t.max, 0, 1)

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)
}

GOFPlot(
  acd_list,
  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
```



Since all models lies in the band, but most of them are above the 45 degree line, the fits are reasonably good.