Homework11

Jun Rao

10/26/2020

Segmentation model selection and evaluation

The goals of this homework are to explore how a subtrain/validation split can be used to select the segmentation model size parameter. compare segmentation models in terms of label error rates and ROC curves

library(ggplot2)  
library(data.table)  
data(neuroblastoma, package="neuroblastoma")

1.For data sequence profile.id=79, chromosome=2 in the neuroblastoma data, assign observations to 50% subtrain 50% validation sets. How many observations are in each set? Hint: use table(set)

pro.dt <- data.table(neuroblastoma$profiles)[profile.id=="79" & chromosome=="2"]  
pro.dt[, set := rep(rep(c("subtrain","validation"), each=2), l=.N)]  
pro.dt[, index := 1:.N]  
  
subtrain.dt <- pro.dt[set=="subtrain"]  
validation.dt <- pro.dt[set== "validation"]  
  
str(subtrain.dt)

## Classes 'data.table' and 'data.frame': 98 obs. of 6 variables:  
## $ profile.id: Factor w/ 575 levels "1","2","4","5",..: 76 76 76 76 76 76 76 76 76 76 ...  
## $ chromosome: Factor w/ 24 levels "1","2","3","4",..: 2 2 2 2 2 2 2 2 2 2 ...  
## $ position : int 18094 2063049 8107742 8179390 11607188 11885382 15647903 15998838 17554158 18160714 ...  
## $ logratio : num 0.2 0.506 0.474 0.43 0.455 ...  
## $ set : chr "subtrain" "subtrain" "subtrain" "subtrain" ...  
## $ index : int 1 2 5 6 9 10 13 14 17 18 ...  
## - attr(\*, ".internal.selfref")=<externalptr>

***We total have 196 observations.***

***In our set, each set have 98 observations, because I use each = 2 at here. It will equally split the original data.***

2.Use binsegRcpp::binseg\_normal to compute models from 1 to 10 segments on the subtrain data, then compute one data table with predicted changepoint positions

subtrain.binseg.models <- binsegRcpp::binseg\_normal(subtrain.dt$logratio, 10)  
subtrain.binseg.dt <- coef(subtrain.binseg.models)  
  
subtrain.getPos <- function(subtrain.index){  
 orig.index <- subtrain.dt$index[ subtrain.index ]  
 pro.dt$position[ orig.index ]  
}  
  
  
subtrain.binseg.dt[, start\_in\_all\_data := subtrain.getPos(subtrain.binseg.dt$start)]  
subtrain.binseg.dt[, end\_in\_all\_data := subtrain.getPos(subtrain.binseg.dt$end)]  
subtrain.binseg.dt[, type := rep("subtrain", l=.N)]  
head(subtrain.binseg.dt)

## segments start end mean start\_in\_all\_data end\_in\_all\_data type  
## 1: 1 1 98 0.14558115 18094 240618997 subtrain  
## 2: 2 1 8 1.30763500 18094 15998838 subtrain  
## 3: 2 9 98 0.04228748 17554158 240618997 subtrain  
## 4: 3 1 6 0.40969361 18094 11885382 subtrain  
## 5: 3 7 8 4.00145916 15647903 15998838 subtrain  
## 6: 3 9 98 0.04228748 17554158 240618997 subtrain

and another data table with segment start/end values (in terms of the full data indices/positions, not just the subtrain set).

validation.binseg.models <- binsegRcpp::binseg\_normal(validation.dt$logratio, 10)  
validation.binseg.dt <- coef(validation.binseg.models)  
  
validation.getPos <- function(validation.index){  
 orig.index <- validation.dt$index[ validation.index ]  
 pro.dt$position[orig.index ]  
}  
  
  
validation.binseg.dt[, start\_in\_all\_data := validation.getPos(validation.binseg.dt$start)]  
validation.binseg.dt[, end\_in\_all\_data := validation.getPos(validation.binseg.dt$end)]  
validation.binseg.dt[, type := rep("validation", l=.N)]  
head(validation.binseg.dt)

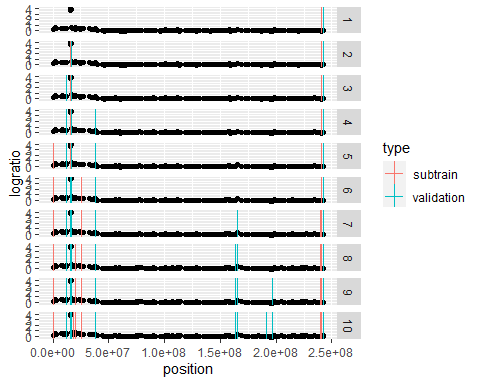
## segments start end mean start\_in\_all\_data end\_in\_all\_data type  
## 1: 1 1 98 0.14231781 3098882 242801018 validation  
## 2: 2 1 7 1.36618893 3098882 16084178 validation  
## 3: 2 8 98 0.04817387 16498022 242801018 validation  
## 4: 3 1 5 0.39228628 3098882 12173642 validation  
## 5: 3 6 7 3.80094555 15291502 16084178 validation  
## 6: 3 8 98 0.04817387 16498022 242801018 validation

Plot these models on top of the data (use points with different color/fill for different sets, black=subtrain, red=validation), and use facet\_grid(segments ~ .) to draw each model size in a different panel. After how many segments does the model appear to overfit?

total.dt <- rbind(subtrain.binseg.dt,validation.binseg.dt)  
  
for(col.name in c("start\_in\_all\_data", "end\_in\_all\_data")){  
 col.value <- total.dt[[col.name]]  
 set(total.dt, j=paste0(col.name, ".pos"),  
 value=pro.dt$position[col.value])  
}  
  
total.dt[, end.before := c(NA, end\_in\_all\_data[-.N]), by=.(type, segments) ]  
change.dt <- data.table(pro.dt, total.dt[1 < start\_in\_all\_data])

## Warning in as.data.table.list(x, keep.rownames = keep.rownames, check.names  
## = check.names, : Item 2 has 110 rows but longest item has 196; recycled with  
## remainder.

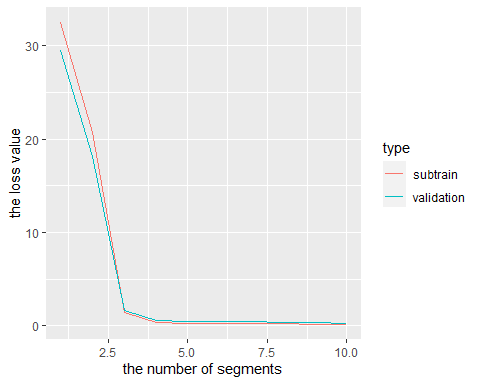
change.dt[, changepoint := (start\_in\_all\_data+end.before)/2]  
  
  
gg <- ggplot(aes(x=position, y=logratio), data=pro.dt)+geom\_point()  
  
(gg+ facet\_grid(segments ~ .)+  
 geom\_segment(aes(  
 x=start, y=mean,  
 xend=end, yend=mean,  
 color=type),  
 data=total.dt) +   
 geom\_vline(aes(  
 xintercept=end\_in\_all\_data,  
 color=type),  
 data=change.dt)+  
 scale\_size\_manual(values=c(subtrain=2, validation=1))  
)



***In our result, it looks like after 3 segments, the model appears to overfit***

3.Compute the square loss for each model size and set, and store these numbers in a data table with columns segments, set,loss. Plot the loss as a function of the number of segments, with different sets in different colors (black=subtrain, red=validation). Based on the loss values/plot, what is the number of segments you should select?

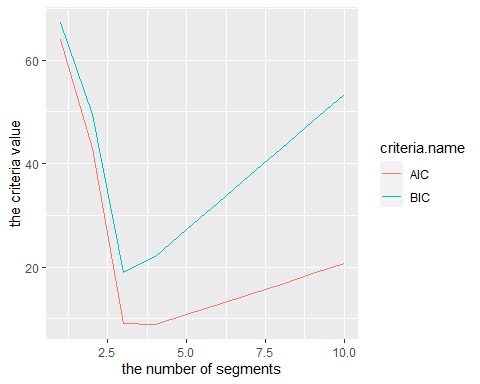
subtrain.loss.dt<- data.table()  
validation.loss.dt <- data.table()  
subtrain.loss.dt[, loss := subtrain.binseg.models$loss]  
subtrain.loss.dt[, segments := subtrain.binseg.models$segments]  
subtrain.loss.dt[, type := "subtrain"]  
validation.loss.dt[, loss := validation.binseg.models$loss]  
validation.loss.dt[, segments := validation.binseg.models$segments]  
validation.loss.dt[, type := "validation"]  
loss.dt <- rbind(subtrain.loss.dt,validation.loss.dt)  
  
  
ggplot()+  
 geom\_line(aes(  
 segments, loss, color=type),  
 data=loss.dt)+  
 xlab("the number of segments") + ylab("the loss value")



***Based on the picture, I will choose segments = 3.***

4.Compute model selection criteria = loss + penalty\*segments, for penalty=2 (AIC) and penalty=log(N.data) (BIC). Save these numbers in a single data table with columns segments, crit.name(AIC/BIC), crit.value(numeric value defined in equation above). Then plot them using different colors for different penalties (e.g., AIC=blue, BIC=orange). Do the two criteria select the same number of segments?

binseg.models <- binsegRcpp::binseg\_normal(pro.dt$logratio, 10)  
AIC.criteria.dt<- data.table()  
BIC.criteria.dt <- data.table()  
  
AIC.criteria.dt[, criteria.value := binseg.models$loss + 2 \* binseg.models$segments]  
AIC.criteria.dt[, segments := binseg.models$segments]  
AIC.criteria.dt[, criteria.name := "AIC"]  
  
BIC.criteria.dt[, criteria.value := binseg.models$loss + log(length(pro.dt$logratio)) \* binseg.models$segments]  
BIC.criteria.dt[, segments := binseg.models$segments]  
BIC.criteria.dt[, criteria.name := "BIC"]  
  
criteria.dt <- rbind(AIC.criteria.dt,BIC.criteria.dt)  
  
  
ggplot()+  
 geom\_line(aes(  
 segments, criteria.value, color=criteria.name),  
 data=criteria.dt)+  
 xlab("the number of segments") + ylab("the criteria value")



***Yes. The two criteria select the same number of segments which is 3.***

The goal is to run binsegRcpp::binseg\_normal on ALL labeled neuroblastoma data sequences, and compare two model selection criteria (AIC/BIC) in terms of label error rates and ROC curves. You should have a for loop over rows of the “annotations” data table, since there is only one label/row per data sequence. For each sequence (unique value of profile.id, chromosome) you should run binary segmentation up to 20 segments. Use penaltyLearning::modelSelection to compute a data table representing the model selection function (which maps penalty values to model sizes).

AIC.loss.dt.list <- list()  
BIC.loss.dt.list <- list()  
change.dt.list <- list()  
  
all.profiles <- data.table(neuroblastoma$profiles)  
all.labels <- data.table(neuroblastoma$annotations)  
label.i.todo <- 1:nrow(all.labels)  
  
# label.i.done <- as.integer(unique(sub(" .\*", "", names(loss.dt.list))))  
# label.i.new <- label.i.todo[! label.i.todo %in% label.i.done]  
  
max.segments <- 10  
  
for (label.i in label.i.todo) {  
 # cat(sprintf("label.i=%d\n", label.i))  
 one.label <- all.labels[label.i]  
 select.dt <- one.label[, .(profile.id, chromosome)]  
 pro.dt <- all.profiles[select.dt, on=names(select.dt)]  
   
   
 this.max <- if(nrow(pro.dt) < max.segments){  
 nrow(pro.dt)  
 }else{  
 max.segments  
 }  
   
 binseg.models <- binsegRcpp::binseg\_normal(pro.dt$logratio, this.max)  
   
 segs.dt.list <- list()  
   
 for (n.segs in 1:this.max) {  
 end <- binseg.models$end[1:n.segs]  
 start <- c(1, end[-length(end)]+1)  
 segs.dt.list[[paste(n.segs)]] <- data.table(start, end)[, .(  
 segments=n.segs,  
 mean=mean(pro.dt$logratio[start:end]),  
 algorithm="DP"   
 ), by=.(start, end)]  
 }  
   
 segs.dt <- do.call(rbind, segs.dt.list)  
   
 for(col.name in c("start", "end")){  
 col.value <- segs.dt[[col.name]]  
 set(segs.dt, j=paste0(col.name, ".pos"),  
 value=pro.dt$position[col.value])  
 }  
   
 segs.dt[, end.before := c(NA, end.pos[-.N]), by=.(segments) ]  
 change.dt <- data.table(select.dt, segs.dt[1 < start])  
 change.dt[, changepoint := (start.pos+end.before)/2]  
 AIC.this.loss.dt <- data.table(  
 segments=1:this.max,  
 loss=binseg.models$loss)  
   
 BIC.this.loss.dt <- data.table(  
 segments=1:this.max,  
 loss=binseg.models$loss)  
   
 penalty <- 0.12  
 AIC.this.loss.dt[, crit.value := loss + penalty\*segments]  
 BIC.this.loss.dt[, crit.value := loss + log10(length(pro.dt$logratio))\*segments]  
 AIC.this.loss.dt[, criteria.name := "AIC"]  
 BIC.this.loss.dt[, criteria.name := "BIC"]  
 AIC.loss.dt.list[[paste(label.i)]] <- data.table(  
 select.dt, AIC.this.loss.dt)  
 change.dt.list[[paste(label.i)]] <- change.dt[, data.table(  
 select.dt, changepoint, segments)]  
   
   
 BIC.loss.dt.list[[paste(label.i)]] <- data.table(  
 select.dt, BIC.this.loss.dt)  
 change.dt.list[[paste(label.i)]] <- change.dt[, data.table(  
 select.dt, changepoint, segments)]  
   
}  
  
change.dt <- do.call(rbind, change.dt.list)  
AIC.loss.dt <- do.call(rbind, AIC.loss.dt.list)  
BIC.loss.dt <- do.call(rbind, BIC.loss.dt.list)  
  
  
  
## Compute model selection function, which maps penalty (lambda)  
## values to model complexity (segments) values.  
AIC.all.model.selection <- AIC.loss.dt[, penaltyLearning::modelSelection(  
 .SD, "loss", "segments"),  
 by=.(profile.id, chromosome)]  
pred.penalty.dt <- AIC.loss.dt[, data.table(  
 pred.log.lambda=log(10)  
), by=.(profile.id, chromosome)]  
  
  
  
## Compute label error, fp/fn for each selected model.  
AIC.error.list <- penaltyLearning::labelError(  
 models=AIC.all.model.selection,  
 labels=all.labels[label.i.todo],  
 changes=change.dt,  
 problem.vars=c("profile.id", "chromosome"),  
 change.var="changepoint",  
 model.vars="segments")  
AIC.error.list$label.errors[, .(  
 profile.id, chromosome, min.lambda, max.lambda, segments, fp, fn)]

## profile.id chromosome min.lambda max.lambda segments fp fn  
## 1: 1 1 0.00000000 0.10981858 10 1 0  
## 2: 1 1 0.10981858 0.14642178 7 1 0  
## 3: 1 1 0.14642178 0.27946950 5 1 0  
## 4: 1 1 0.27946950 1.23652855 4 0 0  
## 5: 1 1 1.23652855 1.83184494 3 0 0  
## ---   
## 20062: 603 17 0.06763515 0.07307963 7 0 0  
## 20063: 603 17 0.07307963 0.11831418 6 0 0  
## 20064: 603 17 0.11831418 0.17108867 5 0 0  
## 20065: 603 17 0.17108867 2.10773718 2 0 1  
## 20066: 603 17 2.10773718 Inf 1 0 1

## Compute ROC curve, FPR/TPR for every prediction threshold (default  
## prediction threshold is zero).  
AIC.roc.list <- penaltyLearning::ROChange(  
 AIC.error.list$model.errors,  
 pred.penalty.dt,  
 problem.vars=c("profile.id", "chromosome"))  
  
AIC.roc.list$roc[, .(min.thresh, max.thresh, FPR, TPR, errors)]

## min.thresh max.thresh FPR TPR errors  
## 1: -Inf -7.526805 0.9785589 1.000000000 2784  
## 2: -7.526805 -7.475998 0.9782074 1.000000000 2783  
## 3: -7.475998 -7.366749 0.9778559 1.000000000 2782  
## 4: -7.366749 -7.362607 0.9775044 1.000000000 2781  
## 5: -7.362607 -7.337930 0.9771529 1.000000000 2780  
## ---   
## 3321: 1.943780 2.099433 0.0000000 0.006980803 569  
## 3322: 2.099433 2.121908 0.0000000 0.005235602 570  
## 3323: 2.121908 2.338209 0.0000000 0.003490401 571  
## 3324: 2.338209 2.491455 0.0000000 0.001745201 572  
## 3325: 2.491455 Inf 0.0000000 0.000000000 573

## Compute model selection function, which maps penalty (lambda)  
## values to model complexity (segments) values.  
BIC.all.model.selection <- BIC.loss.dt[, penaltyLearning::modelSelection(  
 .SD, "loss", "segments"),  
 by=.(profile.id, chromosome)]  
pred.penalty.dt <- BIC.loss.dt[, data.table(  
 pred.log.lambda=log(10)  
), by=.(profile.id, chromosome)]  
  
  
  
## Compute label error, fp/fn for each selected model.  
BIC.error.list <- penaltyLearning::labelError(  
 models=BIC.all.model.selection,  
 labels=all.labels[label.i.todo],  
 changes=change.dt,  
 problem.vars=c("profile.id", "chromosome"),  
 change.var="changepoint",  
 model.vars="segments")  
BIC.error.list$model.errors[, .(  
 profile.id, chromosome, min.lambda, max.lambda, segments, fp, fn)]

## profile.id chromosome min.lambda max.lambda segments fp fn  
## 1: 1 1 0.00000000 0.10981858 10 1 0  
## 2: 1 1 0.10981858 0.14642178 7 1 0  
## 3: 1 1 0.14642178 0.27946950 5 1 0  
## 4: 1 1 0.27946950 1.23652855 4 0 0  
## 5: 1 1 1.23652855 1.83184494 3 0 0  
## ---   
## 20062: 603 17 0.06763515 0.07307963 7 0 0  
## 20063: 603 17 0.07307963 0.11831418 6 0 0  
## 20064: 603 17 0.11831418 0.17108867 5 0 0  
## 20065: 603 17 0.17108867 2.10773718 2 0 1  
## 20066: 603 17 2.10773718 Inf 1 0 1

## Compute ROC curve, FPR/TPR for every prediction threshold (default  
## prediction threshold is zero).  
BIC.roc.list <- penaltyLearning::ROChange(  
 BIC.error.list$model.errors,  
 pred.penalty.dt,  
 problem.vars=c("profile.id", "chromosome"))  
BIC.roc.list$roc[, .(min.thresh, max.thresh, FPR, TPR, errors)]

## min.thresh max.thresh FPR TPR errors  
## 1: -Inf -7.526805 0.9785589 1.000000000 2784  
## 2: -7.526805 -7.475998 0.9782074 1.000000000 2783  
## 3: -7.475998 -7.366749 0.9778559 1.000000000 2782  
## 4: -7.366749 -7.362607 0.9775044 1.000000000 2781  
## 5: -7.362607 -7.337930 0.9771529 1.000000000 2780  
## ---   
## 3321: 1.943780 2.099433 0.0000000 0.006980803 569  
## 3322: 2.099433 2.121908 0.0000000 0.005235602 570  
## 3323: 2.121908 2.338209 0.0000000 0.003490401 571  
## 3324: 2.338209 2.491455 0.0000000 0.001745201 572  
## 3325: 2.491455 Inf 0.0000000 0.000000000 573

Then use the model selection data table as input to penaltyLearning::labelError, which computes the number of incorrectly predicted labels as a function of the penalty/lambda. Also compute a data table of predicted penalty values for AIC/BIC with two rows and columns sequenceID,crit.name,pred.log.lambda. Hint: for reference/example see <http://members.cbio.mines-paristech.fr/~thocking/change-tutorial/Supervised.html>

head(BIC.error.list$label.errors)

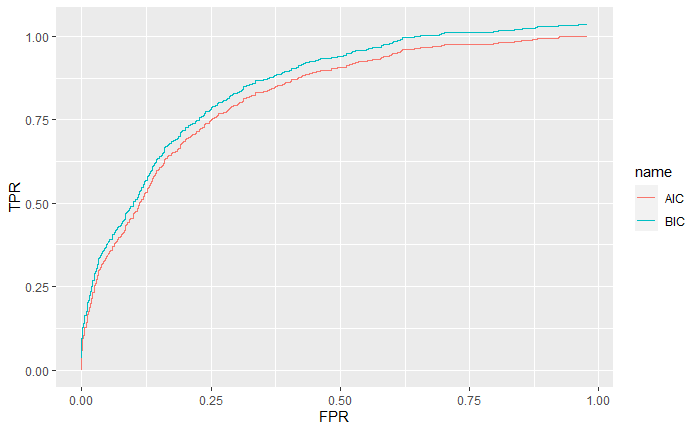
## profile.id chromosome segments min max pred.changes min.lambda  
## 1: 1 1 10 0 125000000 4 0.0000000  
## 2: 1 1 7 0 125000000 3 0.1098186  
## 3: 1 1 5 0 125000000 1 0.1464218  
## 4: 1 1 4 0 125000000 0 0.2794695  
## 5: 1 1 3 0 125000000 0 1.2365286  
## 6: 1 1 2 0 125000000 0 1.8318449  
## max.lambda min.log.lambda max.log.lambda cum.iterations loss crit.value  
## 1: 0.1098186 -Inf -2.2089256 12 3.434715 30.19250  
## 2: 0.1464218 -2.2089256 -1.9212639 7 3.764170 22.49462  
## 3: 0.2794695 -1.9212639 -1.2748621 4 4.057014 17.43591  
## 4: 1.2365286 -1.2748621 0.2123079 3 4.336483 15.03960  
## 5: 1.8318449 0.2123079 0.6053236 2 5.573012 13.60035  
## 6: 8.5101305 0.6053236 2.1412573 1 7.404857 12.75641  
## criteria.name annotation min.changes max.changes color possible.fn  
## 1: BIC normal 0 0 #f6f4bf 0  
## 2: BIC normal 0 0 #f6f4bf 0  
## 3: BIC normal 0 0 #f6f4bf 0  
## 4: BIC normal 0 0 #f6f4bf 0  
## 5: BIC normal 0 0 #f6f4bf 0  
## 6: BIC normal 0 0 #f6f4bf 0  
## possible.fp weight fp fn status  
## 1: 1 1 1 0 false positive  
## 2: 1 1 1 0 false positive  
## 3: 1 1 1 0 false positive  
## 4: 1 1 0 0 correct  
## 5: 1 1 0 0 correct  
## 6: 1 1 0 0 correct

head(AIC.error.list$label.errors)

## profile.id chromosome segments min max pred.changes min.lambda  
## 1: 1 1 10 0 125000000 4 0.0000000  
## 2: 1 1 7 0 125000000 3 0.1098186  
## 3: 1 1 5 0 125000000 1 0.1464218  
## 4: 1 1 4 0 125000000 0 0.2794695  
## 5: 1 1 3 0 125000000 0 1.2365286  
## 6: 1 1 2 0 125000000 0 1.8318449  
## max.lambda min.log.lambda max.log.lambda cum.iterations loss crit.value  
## 1: 0.1098186 -Inf -2.2089256 12 3.434715 4.634715  
## 2: 0.1464218 -2.2089256 -1.9212639 7 3.764170 4.604170  
## 3: 0.2794695 -1.9212639 -1.2748621 4 4.057014 4.657014  
## 4: 1.2365286 -1.2748621 0.2123079 3 4.336483 4.816483  
## 5: 1.8318449 0.2123079 0.6053236 2 5.573012 5.933012  
## 6: 8.5101305 0.6053236 2.1412573 1 7.404857 7.644857  
## criteria.name annotation min.changes max.changes color possible.fn  
## 1: AIC normal 0 0 #f6f4bf 0  
## 2: AIC normal 0 0 #f6f4bf 0  
## 3: AIC normal 0 0 #f6f4bf 0  
## 4: AIC normal 0 0 #f6f4bf 0  
## 5: AIC normal 0 0 #f6f4bf 0  
## 6: AIC normal 0 0 #f6f4bf 0  
## possible.fp weight fp fn status  
## 1: 1 1 1 0 false positive  
## 2: 1 1 1 0 false positive  
## 3: 1 1 1 0 false positive  
## 4: 1 1 0 0 correct  
## 5: 1 1 0 0 correct  
## 6: 1 1 0 0 correct

Also inspect the thresholds element of the ROChange result in order to see the number of label errors per model. Which model has a smaller number of predicted label errors?

AIC.list.dt <- data.table()  
AIC.list.dt[, FPR := AIC.roc.list$roc$FPR]  
AIC.list.dt[, TPR := AIC.roc.list$roc$TPR]  
AIC.list.dt[, name := 'AIC']  
BIC.list.dt <- data.table()  
BIC.list.dt[, FPR := BIC.roc.list$roc$FPR]  
BIC.list.dt[, TPR := BIC.roc.list$roc$TPR]  
BIC.list.dt[, name := 'BIC']  
  
roc.list.dt <- data.table()  
  
  
roc.list.dt <- rbind(AIC.list.dt,BIC.list.dt)  
  
  
ggplot()+  
 geom\_path(aes(  
 FPR, TPR, color=name),  
 data=roc.list.dt)



Based on the result, BIC model has a larger area under the curve

Based on the result, AIC model has a smaller number of predicted label errors