Using quantile to make 3 clusters and 2 clusters

Wanying Ma

12/18/2018

load data.

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
##
## Attaching package: 'data.table'
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

add group information, and focus only on football players.

```
player.info <- read.csv("/Users/mawanying/Dropbox/Luo_Jarek_YC/MRI_Injury_Description_03OCT18_All_Concus
basic.info <- player.info %>% select(SUBJECTSTUDYNUM, INJURYSPORTVARSITY, CONTACTSPORTQGID, NONCONTACTS
football.info <- basic.info %>% filter(INJURYSPORTVARSITY=='Football') #football.info is 88x4
#first in second
d1 <- football.info$SUBJECTSTUDYNUM %in% football.info$CONTACTSPORTQGID
#first in third
d2 <- football.info$SUBJECTSTUDYNUM %in% football.info$NONCONTACTSPORTQGID
#second in first
d3 <- football.info$CONTACTSPORTQGID %in% football.info$SUBJECTSTUDYNUM
#second in third
d4 <- football.info$CONTACTSPORTQGID %in% football.info$NONCONTACTSPORTQGID
#third in first
d5 <- football.info$NONCONTACTSPORTQGID %in% football.info$SUBJECTSTUDYNUM
#third in second
d6 <- football.info$NONCONTACTSPORTQGID %in% football.info$CONTACTSPORTQGID
dupli <- cbind(d1, d2, d3, d4, d5, d6) #88x6
football.noDup <- football.info[rowSums(dupli)<1, ] # football.noDup is 53x4

##############################################################################
############## construct group index for football players ###################
##############      group1: SUBJECTSTUDYNUM                 ############
##############      group2: CONTACTSPORTQGID                ############
##############      group3: NONCONTACTSPORTQGID             ############
##############################################################################
```

```r
group1 <- unique(as.character(football.noDup$SUBJECTSTUDYNUM))     #SUBJECTSTUDYNUM 43x1
group2 <- unique(as.character(football.noDup$CONTACTSPORTQGID))    #CONTACTSPORTQGID 43x1
group3 <- unique(as.character(football.noDup$NONCONTACTSPORTQGID)) #NONCONTACTSPORTQGID 43x1

# after merge, alldata_fb will have 28 players in group 1, 30players in group 2, 28 players in group 3
###############################################################################
### add group index restriced to football players to alldata
addgroupindex <- function(alldata, groups){
  alldata.player <- as.character(alldata$player) #21052x1
  alldata.fb.player.ind <- alldata.player %in% groups #indicator vector: 9280x1
  alldata.fb.player <- alldata.player[alldata.fb.player.ind]
  groupindex <- c()
  for(i in alldata.fb.player){
    if(i %in% group1){
      groupindex <- c(groupindex, 1)
    }
    if(i %in% group2){
      groupindex <- c(groupindex, 2)
    }
    if(i %in% group3){
      groupindex <- c(groupindex, 3)
    }
  }
  fbdata <- alldata[alldata.fb.player.ind, ]
  data.fb <- fbdata %>% mutate(group=groupindex)
  colnames(data.fb)[grep("ij", colnames(data.fb))] <- paste0("den", 1:200)

  return(data.fb)
}

### also change last [4:203] column names
alldata.fb_q <- addgroupindex(alldata_q, c(group1, group2, group3)) #9280x204
```

perform clustering procedure using each tract and meausurement.

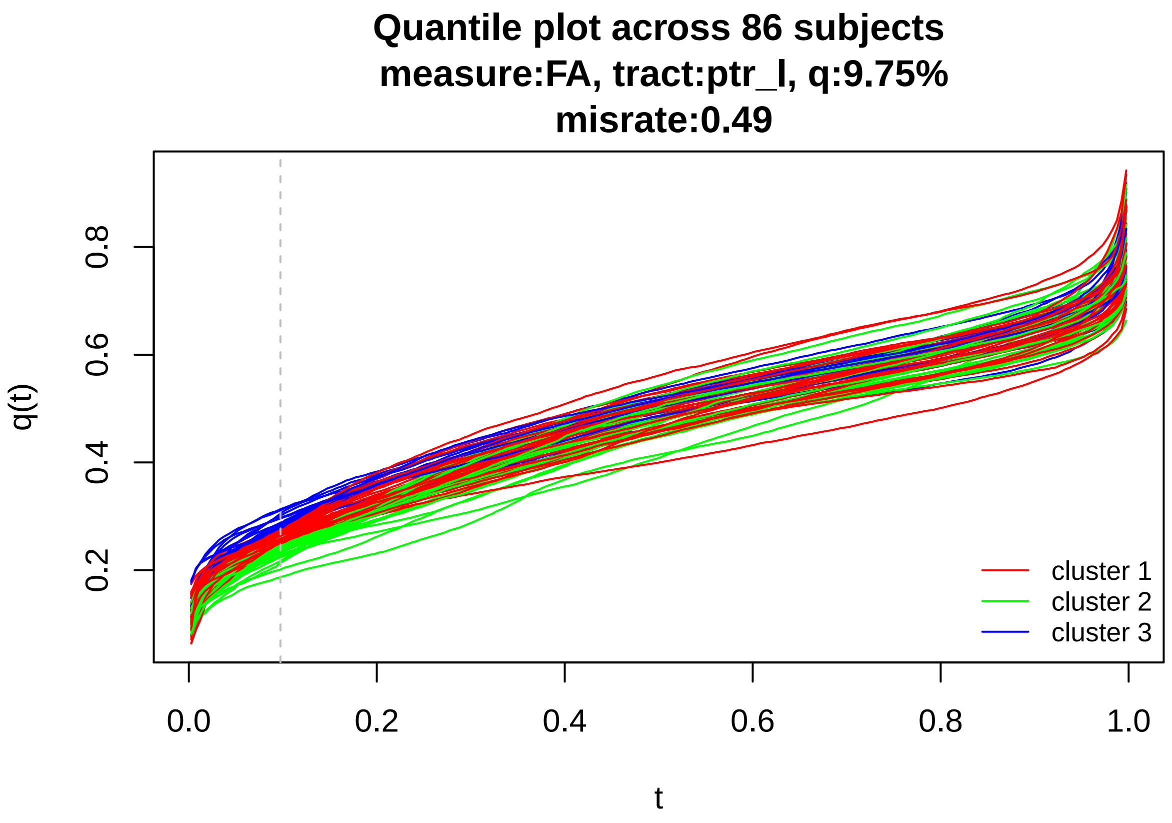
```r
para <- expand.grid(tractnames, c("FA","MD","Da","Dr"))

# permute function
permute <- function(vec, p1, p2, p3){
  for(i in 1:length(vec)){
    if(vec[i]==1){
      vec[i] <- p1
    }else{
      if(vec[i]==2){vec[i] <- p2}else{
        vec[i] <- p3
      }
    }
  }
  return(vec)
}
```

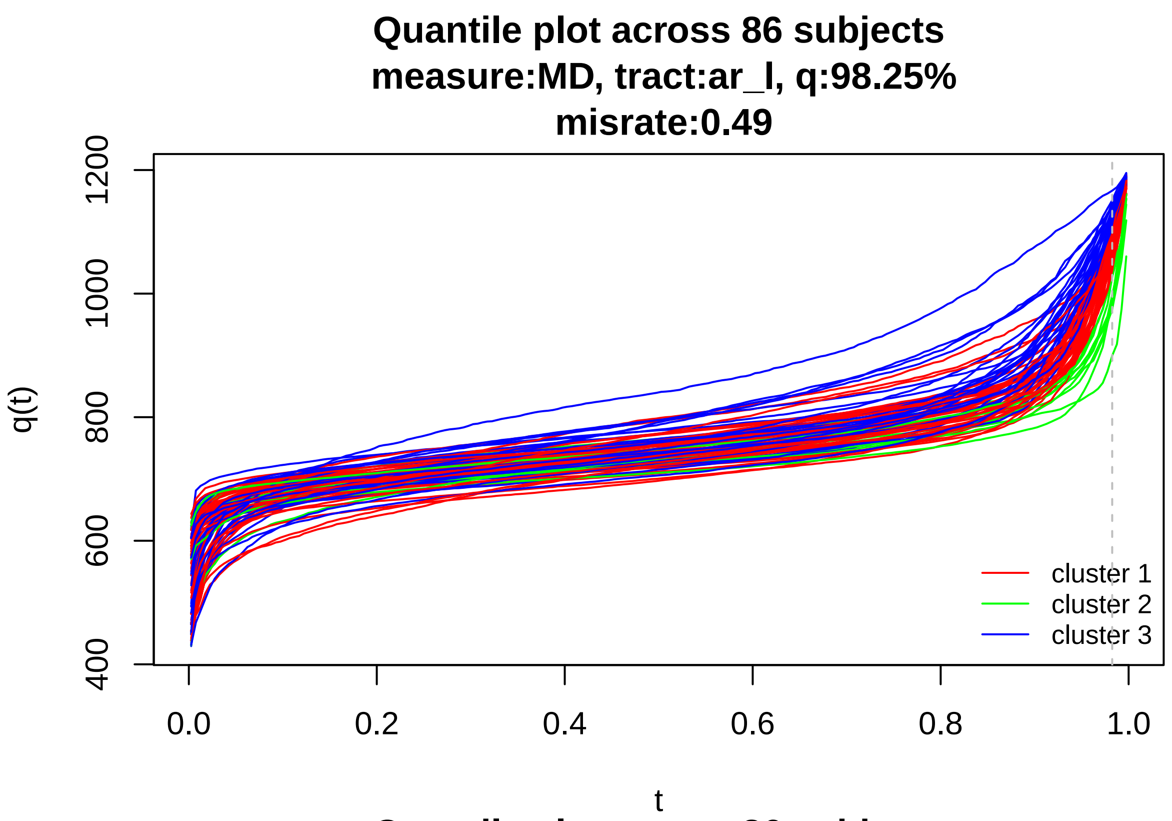
```r
# cluster on 3 groups
threecluster <- function(alldata, iter){
  # alldata is the dataset merged with football information: alldata.fb_q
  # iter: 1~108
  m <- para[iter, 2]
  t <- para[iter, 1]
  alldata_mt <- alldata  %>% filter(measure==m, tract==t) %>% select(group, den1:den200)
  group_mt <- alldata_mt$group
  alldata_mt <- select(alldata_mt, -group)

  calcu_misrate <- sapply(1:200, function(i){
    x <- alldata_mt[,i]
    den.cluster <- kmeans(x, centers = 3)
    perm1 <- permute(den.cluster$cluster, 1,2,3)
    perm2 <- permute(den.cluster$cluster, 1,3,2)
    perm3 <- permute(den.cluster$cluster, 2,1,3)
    perm4 <- permute(den.cluster$cluster, 2,3,1)
    perm5 <- permute(den.cluster$cluster, 3,1,2)
    perm6 <- permute(den.cluster$cluster, 3,2,1)
    perm <- cbind(perm1, perm2, perm3, perm4, perm5, perm6)
    misrate <- apply(perm, 2, function(x){
      sum(x!=group_mt)/length(group_mt)
    })
    min_perm <- order(misrate)[1]
    min_group <- perm[,min_perm]
    return(misrate[min_perm])
  })
  return(calcu_misrate)
}
# cluster on two groups
twocluster <- function(alldata, iter){
  # alldata is the dataset merged with football information: alldata.fb_q
  # iter: 1~108
  m <- para[iter, 2]
  t <- para[iter, 1]
  alldata_mt <- alldata  %>% filter(measure==m, tract==t, group!=3) %>% select(group, den1:den200)
  group_mt <- alldata_mt$group
  alldata_mt <- select(alldata_mt, -group)

  calcu_misrate <- sapply(1:200, function(i){
    x <- alldata_mt[,i]
    den.cluster <- kmeans(x, centers = 3)
    perm1 <- den.cluster$cluster
    perm2 <- permute(den.cluster$cluster, 2,1, NA)
    perm <- cbind(perm1, perm2)
    misrate <- apply(perm, 2, function(x){
      sum(x!=group_mt)/length(group_mt)
    })
    min_perm <- order(misrate)[1]
    min_group <- perm[,min_perm]
    return(misrate[min_perm])
  })
  return(calcu_misrate)
}

result_3cl <- sapply(1:108, function(x){threecluster(alldata.fb_q, x)})
result_2cl <- sapply(1:108, function(x){twocluster(alldata.fb_q, x)})
```

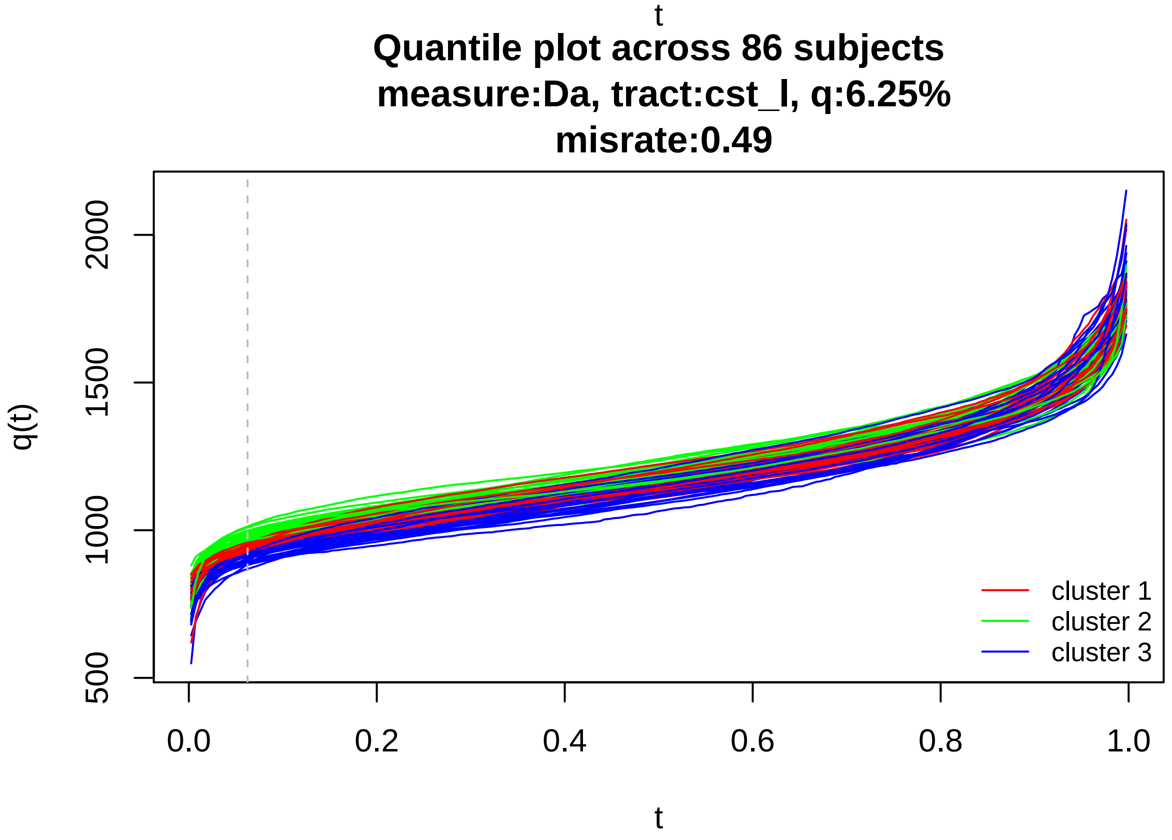
Find the tract and measure which give the smallest cluster error, and visualization.

3 clusters

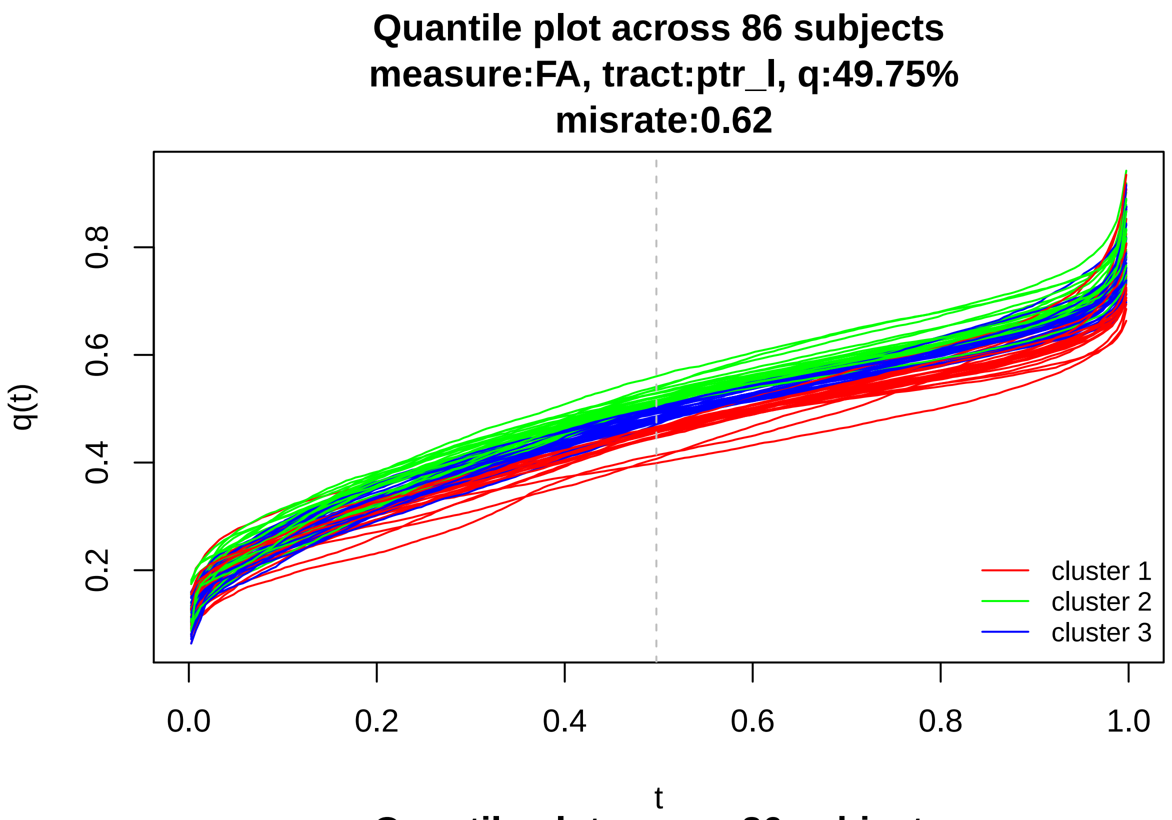
```r
#find which measure and which tract give the smallest cluster error
#3 clusters
mt_select3 <- which(apply(result_3cl, 2, min)==min(result_3cl))
q_select3 <- lapply(mt_select3, function(x){
 data_mt <- result_3cl[,x]
 q.temp  <- which(data_mt==min(result_3cl))
 cbind(rep(x, length(q.temp)), q.temp)
 })
select3 <- do.call(rbind, q_select3)
colnames(select3) <- c("mt", "q")

#2 clusters
mt_select2 <- which(apply(result_2cl, 2, min)==min(result_2cl))
q_select2 <- lapply(mt_select2, function(x){
  data_mt <- result_2cl[,x]
  q.temp  <- which(data_mt==min(result_2cl))
  cbind(rep(x, length(q.temp)), q.temp)
})
select2 <- do.call(rbind, q_select2)
colnames(select2) <- c("mt", "q")


draw_plot3 <- function(data, mt, q){
  #mt is the selected optimal para, q is the corresponding quantile
  t <- para[mt,1]
  m <- para[mt,2]
  alldata_mt <- data  %>% filter(measure==m, tract==t) %>% select(group, den1:den200)
  group_mt <- alldata_mt$group
  alldata_mt <- select(alldata_mt, -group)

  alldata_mtq <- alldata_mt[,q]
  den.cluster <- kmeans(alldata_mtq, centers = 3)
  perm1 <- permute(den.cluster$cluster, 1,2,3)
  perm2 <- permute(den.cluster$cluster, 1,3,2)
  perm3 <- permute(den.cluster$cluster, 2,1,3)
  perm4 <- permute(den.cluster$cluster, 2,3,1)
  perm5 <- permute(den.cluster$cluster, 3,1,2)
  perm6 <- permute(den.cluster$cluster, 3,2,1)
  perm <- cbind(perm1, perm2, perm3, perm4, perm5, perm6)
  misrate <- apply(perm, 2, function(x){
    sum(x!=group_mt)/length(group_mt)
  })
  min_perm <- order(misrate)[1]
  min_group <- perm[,min_perm]
  matplot(den.quantiles/100, t(alldata_mt), type="l",
          main=paste0("Quantile plot across 86 subjects\n measure:", m, ", tract:", t,
                      ", q:", round(den.quantiles[q], digits=2), "%\n misrate:", round(misrate[min_perm]
          xlab="t", ylab="q(t)",
          col=ifelse(min_group==1, "red", ifelse(min_group==2, "green", "blue")),
          lty=1)
  legend("bottomright", c("cluster 1", "cluster 2", "cluster 3"), bty='n', cex=0.8,
         lty=c(1,1,1), col=c("red","green","blue"))
  abline(v=den.quantiles[q]/100, col="grey", lwd=1, lty=2)
}
for (i in 1:dim(select3)[1]){
  draw_plot3(alldata.fb_q, select3[i,1], select3[i,2])
}
```

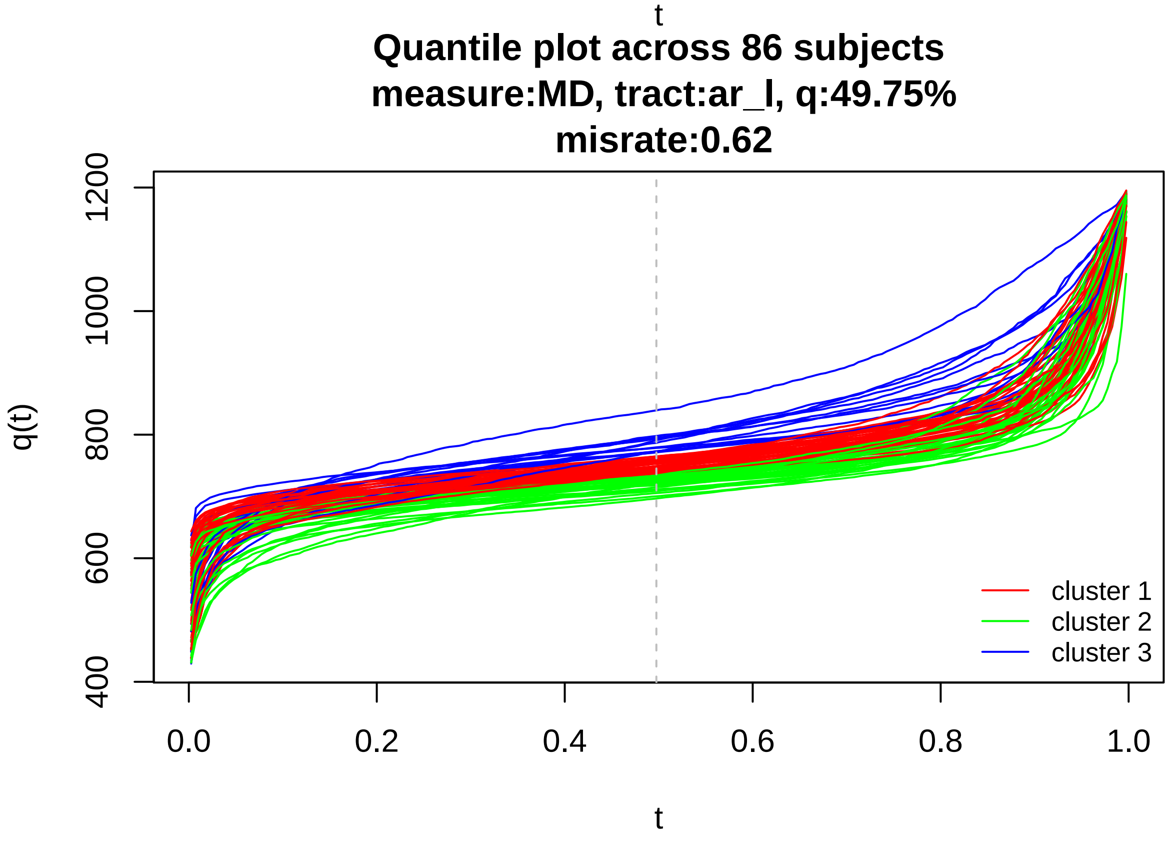


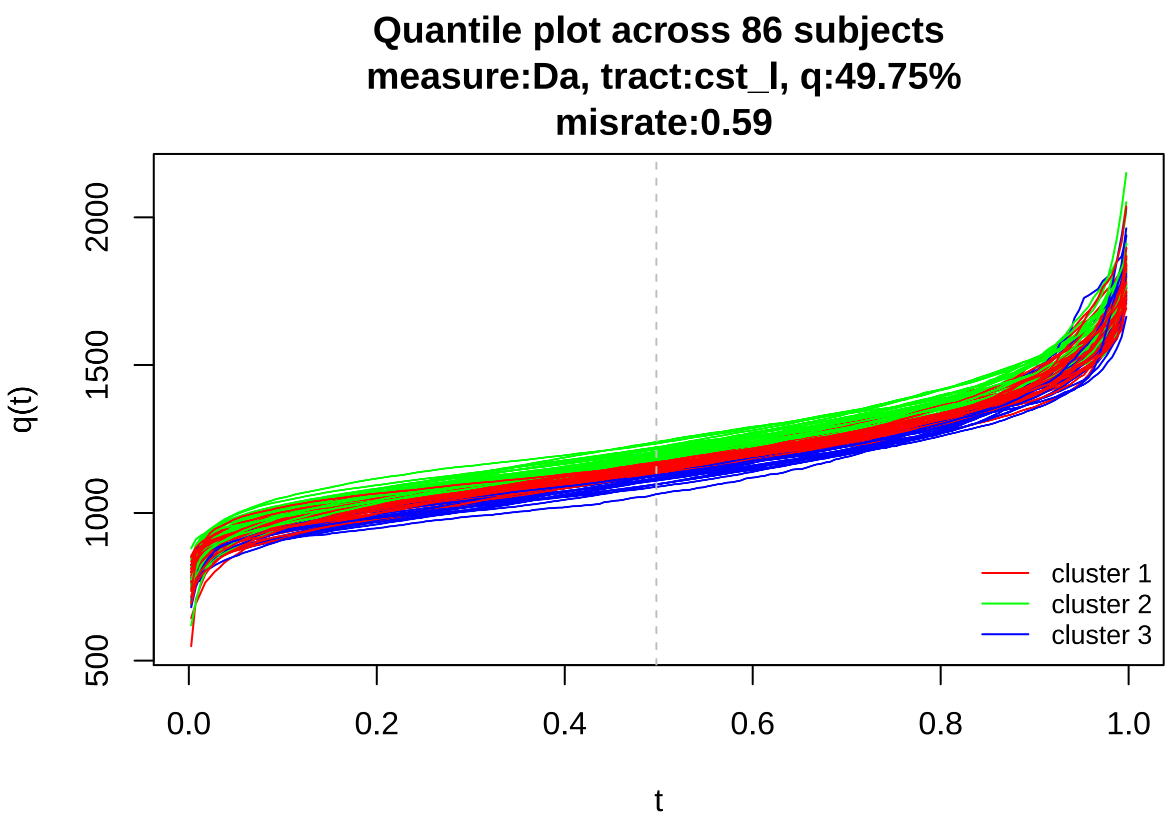




```
# compare with cluster result using 50% quantile with the same tract and measurement.
for (i in 1:dim(select3)[1]){
  draw_plot3(alldata.fb_q, select3[i,1], 100)
}
```


Quantile plot across 86 subjects
measure:FA, tract:ptr_l, q:49.75%
misrate:0.62
q(t)
0.2
0.4
0.6
0.8
0.0
0.2
0.4
0.6
0.8
1.0
t
cluster 1
cluster 2
cluster 3

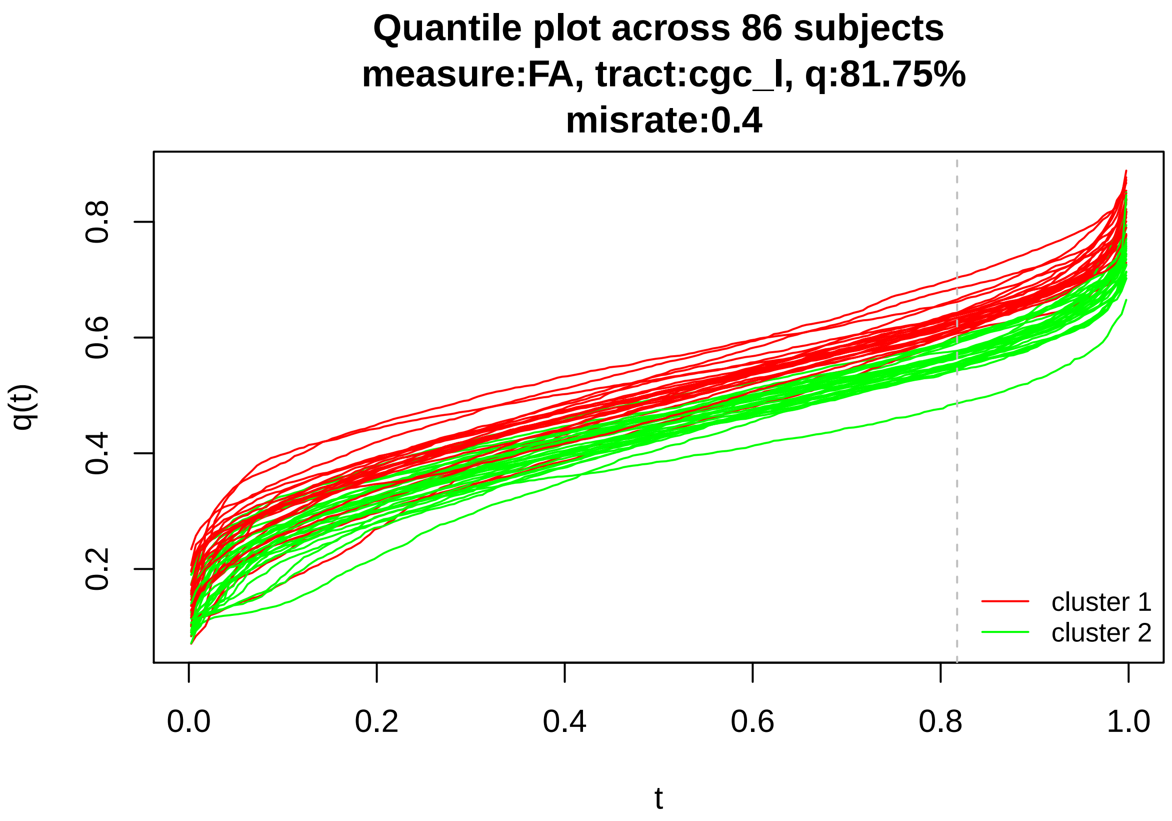

Quantile plot across 86 subjects
measure:MD, tract:ar_l, q:49.75%
misrate:0.62
q(t)
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0
t
cluster 1
cluster 2
cluster 3



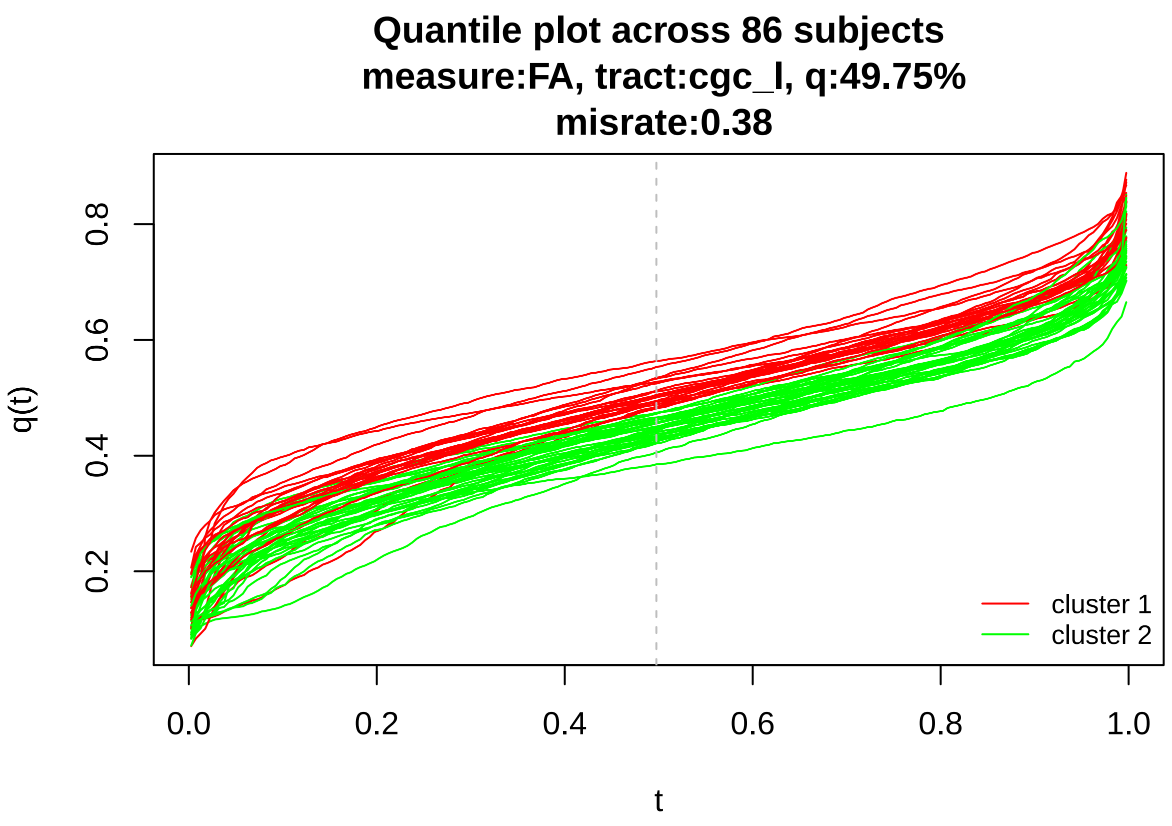
For 3 clusters performance, we can see that 50% quantile can separate the quantile functions for football players better while with higher cluster errors, compared to selected quantiles. Also, selected quantiles which have the best cluster results always are the tails of quantile, that is, either these selected quantiles are very close to 5% or they are close to 99%.

2 clusters

```
draw_plot2 <- function(data, mt, q){
  #mt is the selected optimal para, q is the corresponding quantile
  t <- para[mt,1]
  m <- para[mt,2]
  alldata_mt <- data  %>% filter(measure==m, tract==t, group!=3) %>% select(group, den1:den200)
  group_mt <- alldata_mt$group
  alldata_mt <- select(alldata_mt, -group)

  alldata_mtq <- alldata_mt[,q]
  den.cluster <- kmeans(alldata_mtq, centers = 2)
  perm1 <- permute(den.cluster$cluster, 1,2,NULL)
  perm2 <- permute(den.cluster$cluster, 2,1,NULL)
  perm <- cbind(perm1, perm2)
  misrate <- apply(perm, 2, function(x){
    sum(x!=group_mt)/length(group_mt)
  })
  min_perm <- order(misrate)[1]
  min_group <- perm[,min_perm]
  matplot(den.quantiles/100, t(alldata_mt), type="l",
          main=paste0("Quantile plot across 86 subjects\n measure:", m, ", tract:", t,
                      ", q:", round(den.quantiles[q], digits=2), "%\n misrate:", round(misrate[min_perm
          xlab="t", ylab="q(t)",
          col=ifelse(min_group==1, "red", "green"),
          lty=1)
  legend("bottomright", c("cluster 1", "cluster 2"), bty='n', cex=0.8,
         lty=c(1,1), col=c("red","green"))
  abline(v=den.quantiles[q]/100, col="grey", lwd=1, lty=2)
}
for (i in 1:dim(select2)[1]){
  draw_plot2(alldata.fb_q, select2[i,1], select2[i,2])
}
```



```
# compare with cluster result using 50% quantile with the same tract and measurement.
for (i in 1:dim(select2)[1]){
  draw_plot2(alldata.fb_q, select2[i,1], 100)
}
```



For 2 clusters performance, we can see that both 50% quantile and selected quantiles can separate the quantile functions for football players similarly.

random forest classifier for 3 groups.

use one dataset to see how random forest works.

```
mt <- select3[1,1]
q <- select3[1,2]
t <- para[mt,1]
m <- para[mt,2]
cat("use tract:", as.character(para$Var1)[mt], ", measure:", as.character(para$Var2)[mt])
```

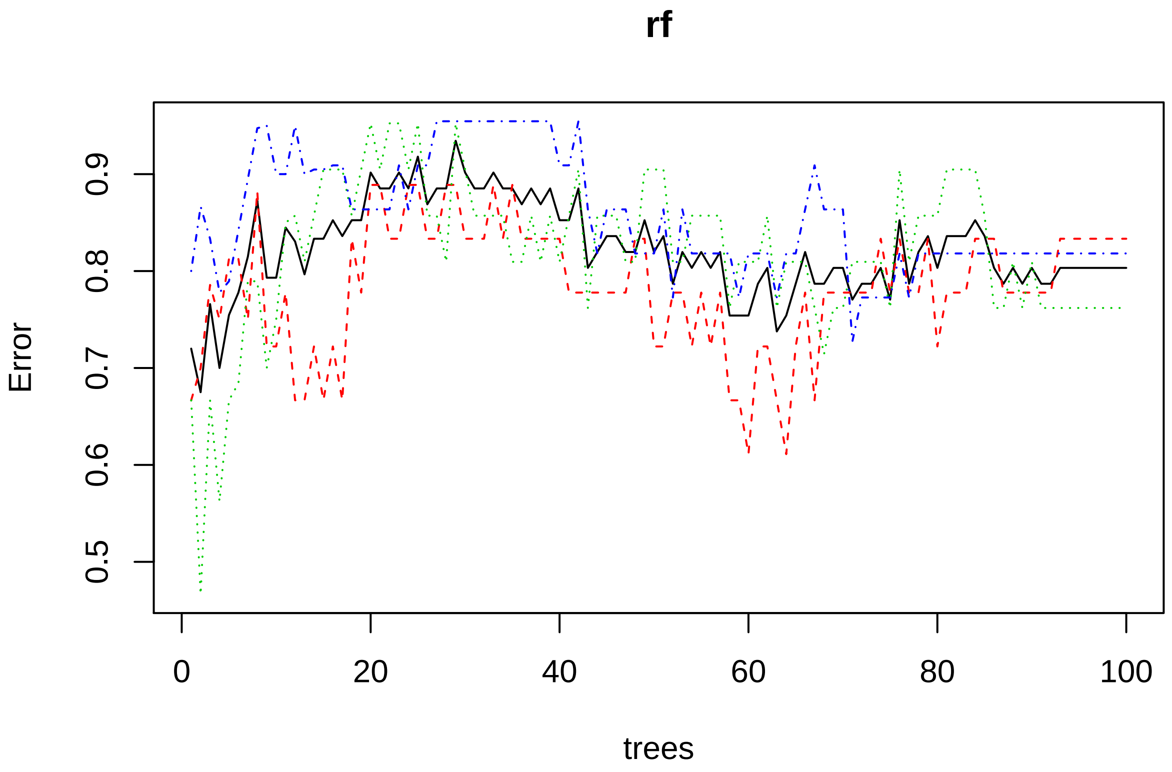
```
## use tract: ptr_l , measure: FA
```

```
alldata_mt <- alldata.fb_q  %>% filter(measure==m, tract==t) %>% select(group, den1:den100)
alldata_mt$group <- as.factor(alldata_mt$group)
group_mt <- alldata_mt$group
## create 70% data as training data, 30% data as validating dataset, 1 is for training, 2 is for valida
data_ind <-sample(2, nrow(alldata_mt), replace=T, prob=c(0.7, 0.3))

data.tr = alldata_mt[data_ind==1,]
data.val = alldata_mt[data_ind==2,]
#table(group_mt[data_ind==1])/sum(data_ind==1)
#table(group_mt[data_ind==2])/sum(data_ind==2)
#orginal proportion, expect to see the proportion similar
#table(group_mt)/length(group_mt)
```

```r
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine
```

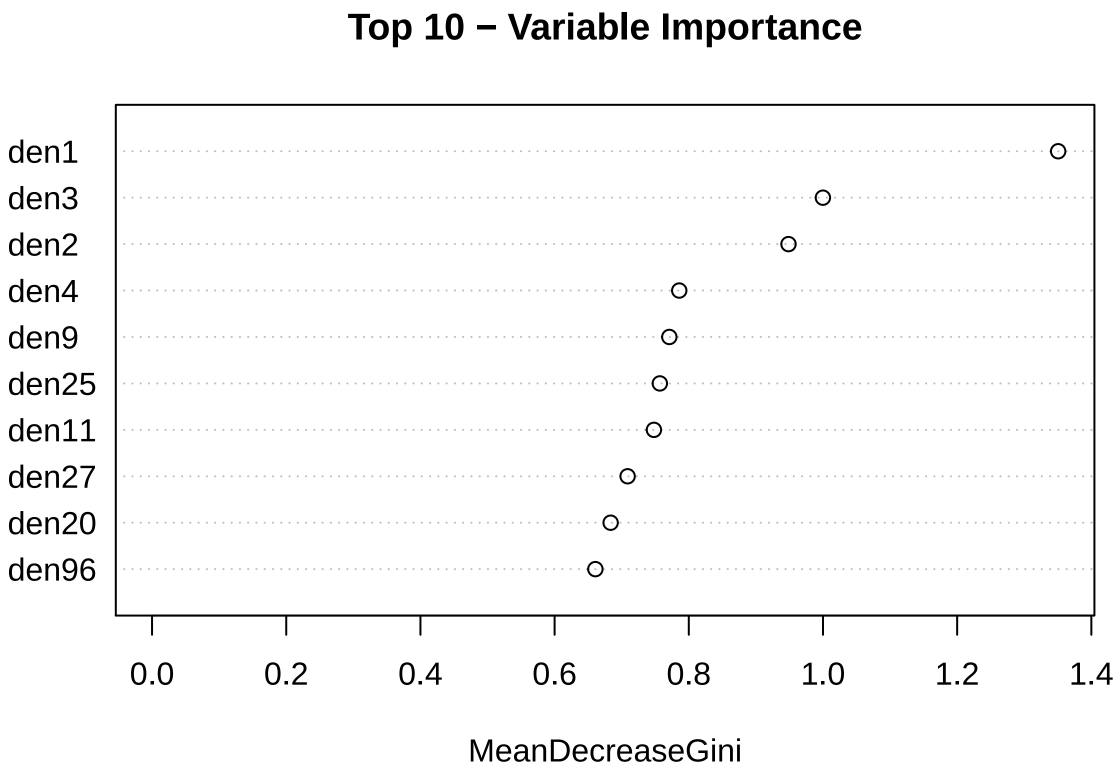
```r
#Fit Random Forest Model
rf = randomForest(group ~ .,
                  ntree = 100, data=data.tr)
plot(rf)
```



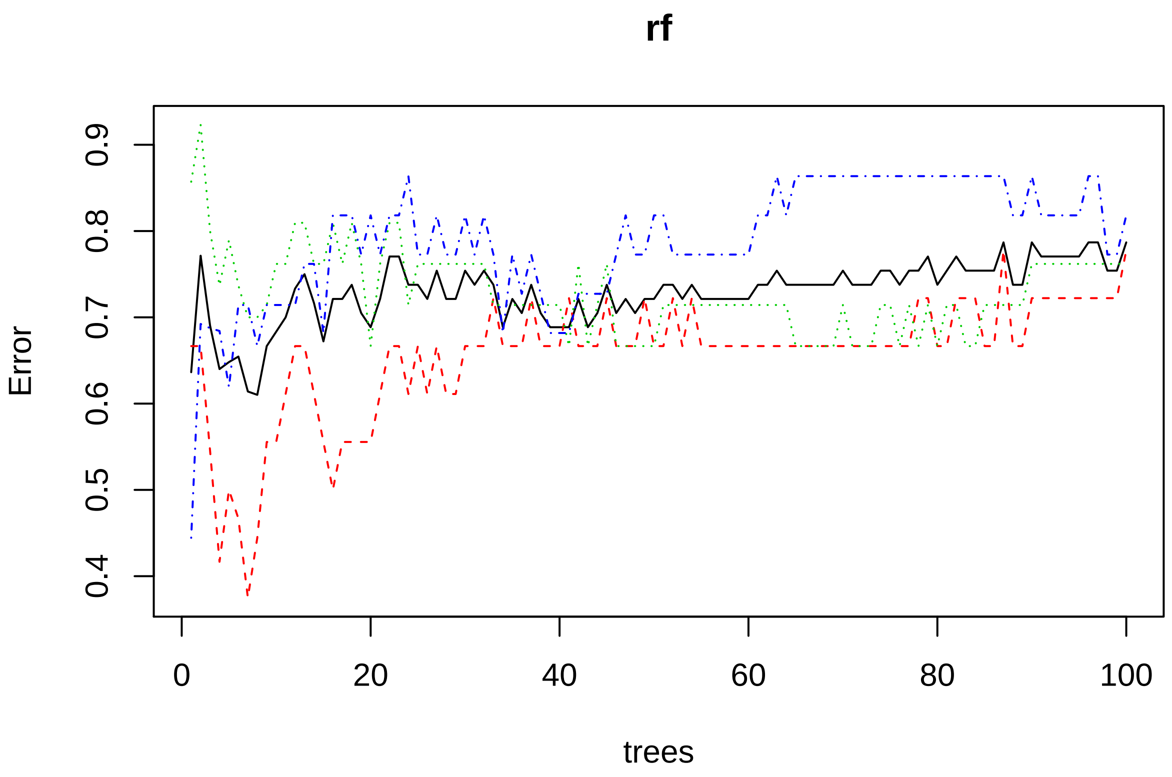
```r
print(rf)
```

```
##
## Call:
##  randomForest(formula = group ~ ., data = data.tr, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 80.33%
## Confusion matrix:
##   1  2 3 class.error
## 1 3  7 8   0.8333333
## 2 7  5 9   0.7619048
## 3 7 11 4   0.8181818
```

```
# Variable Importance
varImpPlot(rf,
           sort = T,
           n.var=10,
           main="Top 10 - Variable Importance")
```



```
#Variable Importance
var.imp = data.frame(importance(rf,
                                type=2))
# make row names as columns
var.imp$Variables = row.names(var.imp)
#print(var.imp[order(var.imp$MeanDecreaseGini,decreasing = T),])
var.imp <- var.imp[order(var.imp$MeanDecreaseGini,decreasing = T),]
data.tr.imp <- select(data.tr, group, var.imp$Variables[1:20])
rf = randomForest(group ~ .,
                  ntree = 100, data=data.tr.imp)
plot(rf)
```



```r
print(rf)
```

```
##
## Call:
##  randomForest(formula = group ~ ., data = data.tr.imp, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 78.69%
## Confusion matrix:
##   1  2  3 class.error
## 1 4  7  7   0.7777778
## 2 5  5 11   0.7619048
## 3 6 12  4   0.8181818
```

```r
# Predicting response variable
data.val$predicted.response = predict(rf, data.val)

# Create Confusion Matrix
confusion3 <- t(sapply(1:3, function(x){as.numeric(table(data.val$predicted.response[data.val$group==x]
confusion3 <- cbind(confusion3, (rowSums(confusion3)-diag(confusion3))/rowSums(confusion3))
rownames(confusion3) <- paste0("true", 1:3)
colnames(confusion3) <- c(paste0("claim", 1:3), "class.error" )
# confusion matrix
confusion3
```

```
##       claim1 claim2 claim3 class.error
## true1      2      7      1   0.8000000
## true2      1      7      1   0.2222222
## true3      1      1      4   0.3333333
```

```r
misrate3 <- (sum(confusion3)-sum(diag(confusion3)))/sum(confusion3)
# misclassification rate
misrate3
```

```
## [1] 0.5067454
```

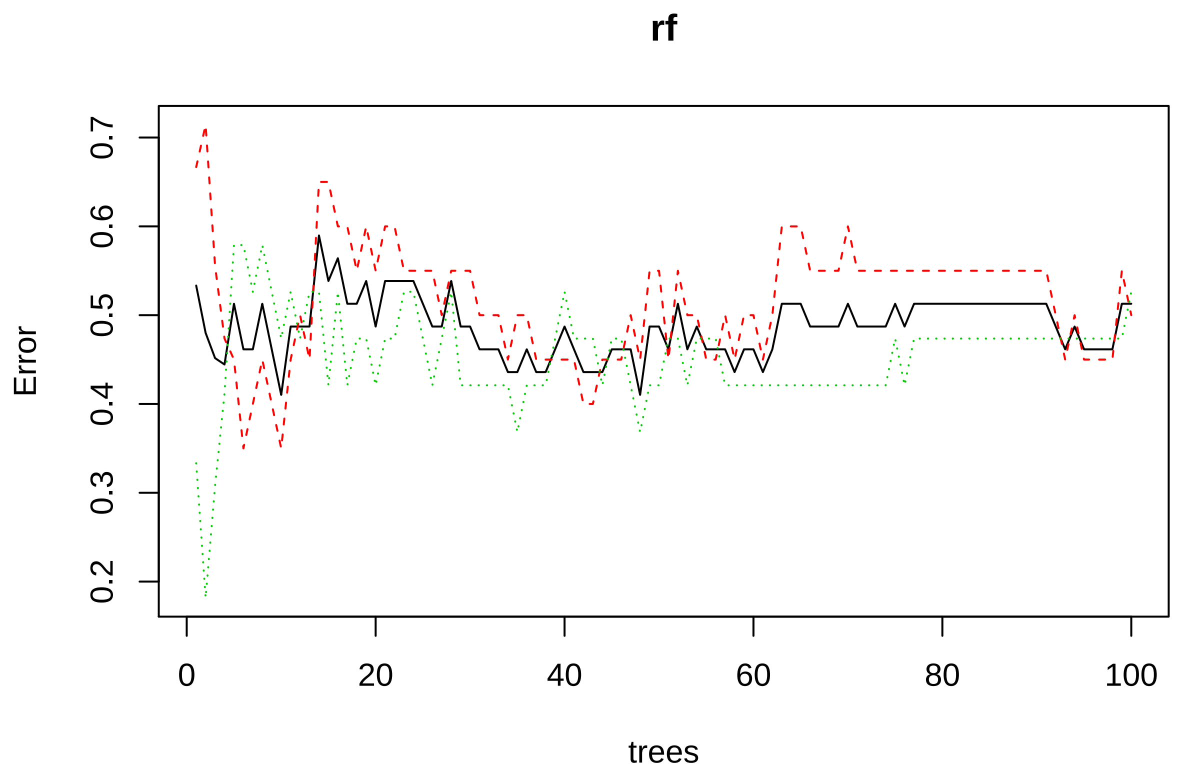
random forest classifier for 2 groups.

use one dataset to see how random forest works.

```r
mt <- select2[1,1]
q <- select2[1,2]
t <- para[mt,1]
m <- para[mt,2]
cat("use tract:", as.character(para$Var1)[mt], ", measure:", as.character(para$Var2)[mt])
```

```
## use tract: cgc_l , measure: FA
```

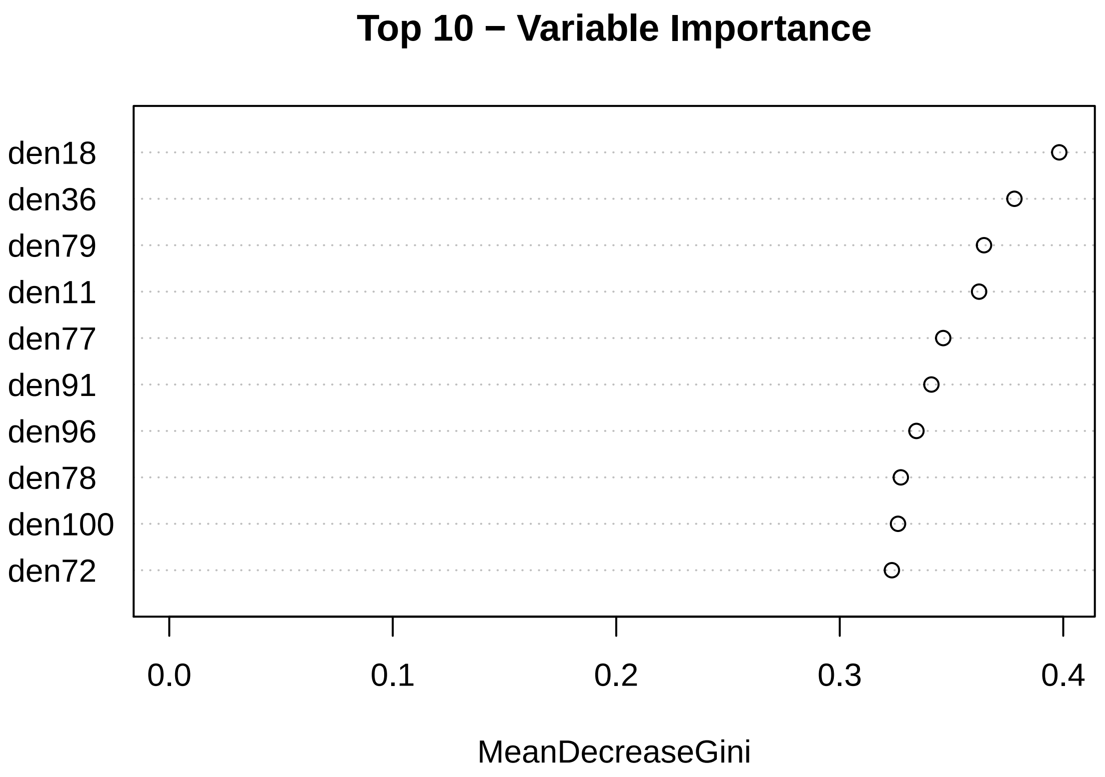
```r
alldata_mt <- alldata.fb_q  %>% filter(measure==m, tract==t, group!=3) %>% select(group, den1:den100)
alldata_mt$group <- as.factor(alldata_mt$group)
group_mt <- alldata_mt$group
## create 70% data as training data, 30% data as validating dataset, 1 is for training, 2 is for valida
data_ind <-sample(2, nrow(alldata_mt), replace=T, prob=c(0.7, 0.3))

data.tr = alldata_mt[data_ind==1,]
data.val = alldata_mt[data_ind==2,]
#table(group_mt[data_ind==1])/sum(data_ind==1)
#table(group_mt[data_ind==2])/sum(data_ind==2)
#orginal proportion, expect to see the proportion similar
#table(group_mt)/length(group_mt)

library(randomForest)
#Fit Random Forest Model
rf = randomForest(group ~ .,
                  ntree = 100, data=data.tr)
plot(rf)
```



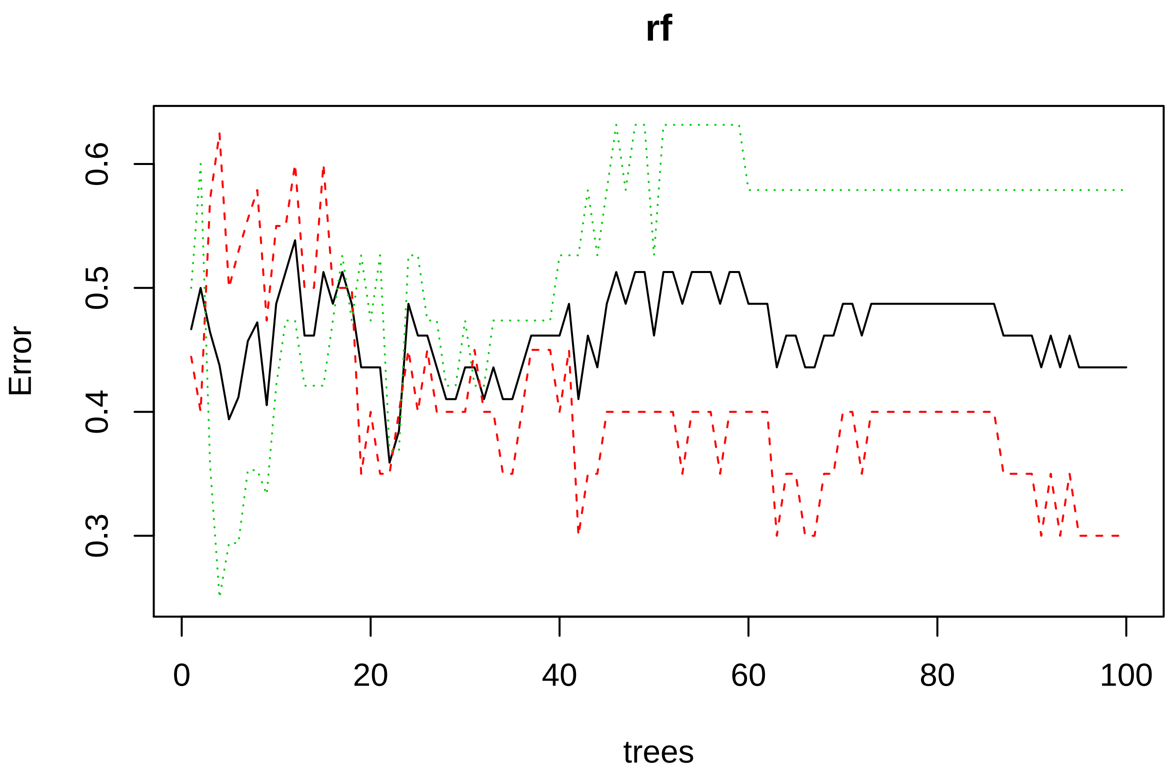
```
print(rf)
```

```
##
## Call:
##  randomForest(formula = group ~ ., data = data.tr, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 51.28%
## Confusion matrix:
##    1  2 class.error
## 1 10 10   0.5000000
## 2 10  9   0.5263158
```

```
# Variable Importance
varImpPlot(rf,
           sort = T,
           n.var=10,
           main="Top 10 - Variable Importance")
```



```
#Variable Importance
var.imp = data.frame(importance(rf,
                                type=2))
# make row names as columns
var.imp$Variables = row.names(var.imp)
#print(var.imp[order(var.imp$MeanDecreaseGini,decreasing = T),])
var.imp <- var.imp[order(var.imp$MeanDecreaseGini,decreasing = T),]
data.tr.imp <- select(data.tr, group, var.imp$Variables[1:20])
rf = randomForest(group ~ .,
                  ntree = 100, data=data.tr.imp)
plot(rf)
```



```r
print(rf)
```

```
##
## Call:
##  randomForest(formula = group ~ ., data = data.tr.imp, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 43.59%
## Confusion matrix:
##    1 2 class.error
## 1 14 6   0.3000000
## 2 11 8   0.5789474
```

```r
# Predicting response variable
data.val$predicted.response = predict(rf, data.val)

# Create Confusion Matrix
confusion2 <- t(sapply(1:2, function(x){as.numeric(table(data.val$predicted.response[data.val$group==x]
confusion2 <- cbind(confusion2, (rowSums(confusion2)-diag(confusion2))/rowSums(confusion2))
rownames(confusion2) <- paste0("true", 1:2)
colnames(confusion2) <- c(paste0("claim", 1:2), "class.error" )
# confusion matrix
confusion2
```

```
##       claim1 claim2 class.error
## true1      2      6   0.7500000
## true2      7      4   0.6363636
```

```r
misrate2 <- (sum(confusion2)-sum(diag(confusion2)))/sum(confusion2)
# misclassification rate
misrate2
```

```
## [1] 0.7056856
```