

# features

*Nazanin Zounemat Kermani*

*January 25, 2018*

## finding discriminative features

The features are going to be selected based on different models. To find the significant features permutation test is applied.

```
data <- t(read.table("C:/Users/nz1413/Desktop/dataTAC/sputum_508genes.txt",
dataScaled = scale(data)
```

```
library(mclust)
```

```
## Package 'mclust' version 5.4
## Type 'citation("mclust")' for citing this R package in publications.
```

```
mbc<-Mclust(dataScaled)
summary(mbc)
```

```
## -----
## Gaussian finite mixture model fitted by EM algorithm
## -----
##
## Mclust VII (spherical, varying volume) model with 4 components:
##
##   log.likelihood    n    df        BIC        ICL
##      -63604.42  104  2039   -136678.7   -136678.8
##
## Clustering table:
##   1  2  3  4
## 24 23 28 29
```

```
#clusters
library(CMA)
```

```
## Loading required package: Biobase
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##   clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##   clusterExport, clusterMap, parApply, parCapply, parLapply,
##   parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:stats':
##
##   IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
```

```

##
##   anyDuplicated, append, as.data.frame, cbind, colMeans,
##   colnames, colSums, do.call, duplicated, eval, evalq, Filter,
##   Find, get, grep, grepl, intersect, is.unsorted, lapply,
##   lengths, Map, mapply, match, mget, order, paste, pmax,
##   pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce,
##   rowMeans, rownames, rowSums, sapply, setdiff, sort, table,
##   tapply, union, unique, unsplit, which, which.max, which.min

## Welcome to Bioconductor
##
##   Vignettes contain introductory material; view with
##   'browseVignettes()'. To cite Bioconductor, see
##   'citation("Biobase")', and for packages 'citation("pkgname)".

set.seed(321)
options(warn=-1)
varsel_ftest <- GeneSelection(X = dataScaled, y=mbc$classification,
                             method = "f.test")

## GeneSelection: iteration 1
varsel_kruskaltest <- GeneSelection(X = dataScaled, y=mbc$classification,
                                   method = "kruskal.test")

## GeneSelection: iteration 1
varsel_rf <- GeneSelection(X = dataScaled, y=mbc$classification,
                          method = "rf")

## GeneSelection: iteration 1
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:Biobase':
##
##   combine
##
## The following object is masked from 'package:BiocGenerics':
##
##   combine
varsel_boosting <- GeneSelection(X = dataScaled, y=mbc$classification,
                                method = "boosting", mstop = 1000)

## GeneSelection: iteration 1
options(warn=0)

library(CMA)
options(warn=-1)
set.seed(113)
numberOfResampling = 1000
nvars = dim(dataScaled)[2]
nSamples = dim(dataScaled)[1]
varImportanceftest = matrix(0,numberOfResampling,nvars)

```

```
varImportancekruskaltest = matrix(0,numberOfResampling,nvars)
varImportancerf = matrix(0,numberOfResampling,nvars)
varImportanceboosting = matrix(0,numberOfResampling,nvars)
for(i in 1:numberOfResampling)
{
  labels_permutated = sample(mbc$classification)
  varsel_ftest <- GeneSelection(X = dataScaled, y=labels_permutated,
                              method = "f.test")
  varsel_kruskaltest <- GeneSelection(X = dataScaled, y=labels_permutated,
                                    method = "kruskal.test")
  varsel_rf <- GeneSelection(X = dataScaled, labels_permutated,
                            mtry= ceiling(10*sqrt(nvars)), nodesize = 8,
                            method = "rf", ntree = 1000)
  varsel_boosting <- GeneSelection(X = dataScaled, y=labels_permutated,
                                  method = "boosting", mstop = 1000)
  varImportanceftest[i,varsel_ftest@rankings[[1]][1:nvars]] = varsel_ftest@importance[[1]][1:nvars]
  varImportancekruskaltest[i,varsel_kruskaltest@rankings[[1]][1:nvars]] = varsel_kruskaltest@importance[[1]][1:nvars]
  varImportancerf[i,varsel_rf@rankings[[1]][1:nvars]] = varsel_rf@importance[[1]][1:nvars]
  varImportanceboosting[i,varsel_boosting@rankings[[1]][1:nvars]] = varsel_boosting@importance[[1]][1:nvars]
}
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

```
set.seed(113)
options(warn=-1)
varsel_ftest <- GeneSelection(X = dataScaled, y=mbc$classification,
                             method = "f.test")
```

```
## GeneSelection: iteration 1
varsel_kruskaltest <- GeneSelection(X = dataScaled, y=mbc$classification,
                                   method = "kruskal.test")
```

```
## GeneSelection: iteration 1
varsel_rf <- GeneSelection(X = dataScaled, y=mbc$classification,
                           mtry= ceiling(10*sqrt(nvars)), nodesize = 8 , method = "rf")
```

[illegible]

77

```

temp = which(varsel_kruskaltest@rankings[[1]]==i)
pValuekruskaltest[1,i] = length(which(varImportancekruskaltest[,i]>
                                     varsel_kruskaltest@importance[[1]][temp]))

temp = which(varsel_rf@rankings[[1]]==i)
pValuerf[1,i] = length(which(varImportancerf[,i]>
                              varsel_rf@importance[[1]][temp]))

temp = which(varsel_boosting@rankings[[1]]==i)
pValueboosting[1,i] = length(which(varImportanceboosting[,i]>
                                    varsel_boosting@importance[[1]][temp]))
}

fctestPvalue = t(pValuefctest)/1000
kruskalPvalue = t(pValuekruskaltest)/1000
boostingPvalue = t(pValueboosting)/1000
rfPvalue = t(pValuerf)/1000

# adjustment
indeces = order(fctestPvalue)
fctestPvalue[indeces] = p.adjust(fctestPvalue[indeces],
                                method = "bonferroni")

indeces = order(kruskalPvalue)
kruskalPvalue[indeces] = p.adjust(kruskalPvalue[indeces],
                                method = "bonferroni")

indeces = order(boostingPvalue)
boostingPvalue[indeces] = p.adjust(boostingPvalue[indeces],
                                method = "bonferroni")

indeces = order(rfPvalue)
rfPvalue[indeces] = p.adjust(rfPvalue[indeces],
                             method = "bonferroni")

result = cbind(varsel_fctest@importance[[1]][varsel_fctest@rankings[[1]][1:nvars]],
               fctestPvalue,
               varsel_kruskaltest@importance[[1]][varsel_kruskaltest@rankings[[1]][1:nvars]],
               kruskalPvalue,
               varsel_rf@importance[[1]][varsel_rf@rankings[[1]][1:nvars]],
               rfPvalue,
               varsel_boosting@importance[[1]][varsel_boosting@rankings[[1]][1:nvars]],
               boostingPvalue)

nvars = dim(dataScaled)[2]
library(gtools)
foldChange = 1:nvars
for (i in 1:nvars)
{
  foldChange[i] = foldchange(mean(dataScaled[which(mbc$classification==1 | mbc$classification==2),i]), 1)
}

FinalSelection = matrix(0,nvars,4 )
colnames(FinalSelection) = c("f.test", "kruskal.test", "component boosting","random Forst")
rownames(FinalSelection) = colnames(dataScaled)
for(i in seq(1,8,2))
{

```

```

    quantTemp = quantile(result[,i], 0.95)
    FinalSelection[, (floor(i/2)+1)] = as.numeric(c(result[,i] >= quantTemp & result[, (i+1)] <= 0.01))
}

FinalSelection = cbind(FinalSelection, foldChange)
rownames(FinalSelection) = colnames(dataScaled)

result = cbind(result, foldChange)
rownames(result) = colnames(dataScaled)

write.table(result, file = 'C:/Users/nz1413/Desktop/AnalysisTag/Final/data/featuresFor4Classes.txt' )

write.table(FinalSelection, file = 'C:/Users/nz1413/Desktop/AnalysisTag/Final/data/DesicionfeaturesFor4' )

```