

MIXTURE

18/08/2020

A noise constrained Recursive Feature Extraction algorithm for robust deconvolution of cell-types mixture from molecular signatures

Since the significant impact of immunotherapy in cancer, the estimation of the immune cell-type proportions present in a tumor becomes crucial. Currently, the deconvolution of the cell mixture content of a tumor is carried out by different analytic tools, yet the accuracy of inferred cell type proportions has room for improvement. We improve tumor immune environment characterization developing MIXTURE, an analytical method based on a noise constrained recursive variable selection for a support vector regression

How to install MIXTURE

```
install.packages("devtools")
library(devtools)
install_github("elmerfer/MIXTURE")
```

Running MIXTURE

In this example we will use the LM22 signature matrix as a 22 subject expression matrix cohort

```
library(MIXTURE)
```

```
## Loading required package: BiocParallel
## Loading required package: e1071
## Loading required package: ComplexHeatmap
## Loading required package: grid
## =====
## ComplexHeatmap version 2.1.0
## Bioconductor page: http://bioconductor.org/packages/ComplexHeatmap/
## Github page: https://github.com/jokergoo/ComplexHeatmap
## Documentation: http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite:
## Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
## genomic data. Bioinformatics 2016.
## =====
## Loading required package: gridExtra
## Loading required package: ggExtra
## Loading required package: stringr
## Loading required package: plyr
```

```

## Loading required package: abind
## Loading required package: openxlsx
## Loading required package: ggplot2
## Loading required package: parallel

##Load signature matrix
data(LM22)
## Run the self test on LM22 signature
mix.test <- MIXTURE(expressionMatrix = LM22,

                    signatureMatrix = LM22,

                    functionMixture = nu.svm.robust.RFE,
                    useCores = 10L,
                    verbose = TRUE,
                    nullDist = "PopulationBased"
                    )

##
## Running...
## Original Samples run
##
## Population based null distribution
##
## Building random population
##
## Building null distribution
##
## finish

# Showing the predicted proportions
head(GetMixture(mix.test)[,1:3])

##
## B cells naive B cells memory Plasma cells
## B cells naive 1 0 0
## B cells memory 0 1 0
## Plasma cells 0 0 1
## T cells CD8 0 0 0
## T cells CD4 naive 0 0 0
## T cells CD4 memory resting 0 0 0

# Showing the predicted absolute coefficients
head(GetMixture(mix.test, type = "absolute")[,1:3])

##
## B cells naive B cells memory Plasma cells
## B cells naive 0.9596281 0.000000 0.0000000
## B cells memory 0.0000000 1.070502 0.0000000
## Plasma cells 0.0000000 0.000000 0.6977581
## T cells CD8 0.0000000 0.000000 0.0000000
## T cells CD4 naive 0.0000000 0.000000 0.0000000
## T cells CD4 memory resting 0.0000000 0.000000 0.0000000

```

```
# Showing the slots names of the MIXTURE object
names(mix.test)
```

```
## [1] "Subjects"          "PermutedMetrix" "method"          "usedGenes"
## [5] "p.values"
```

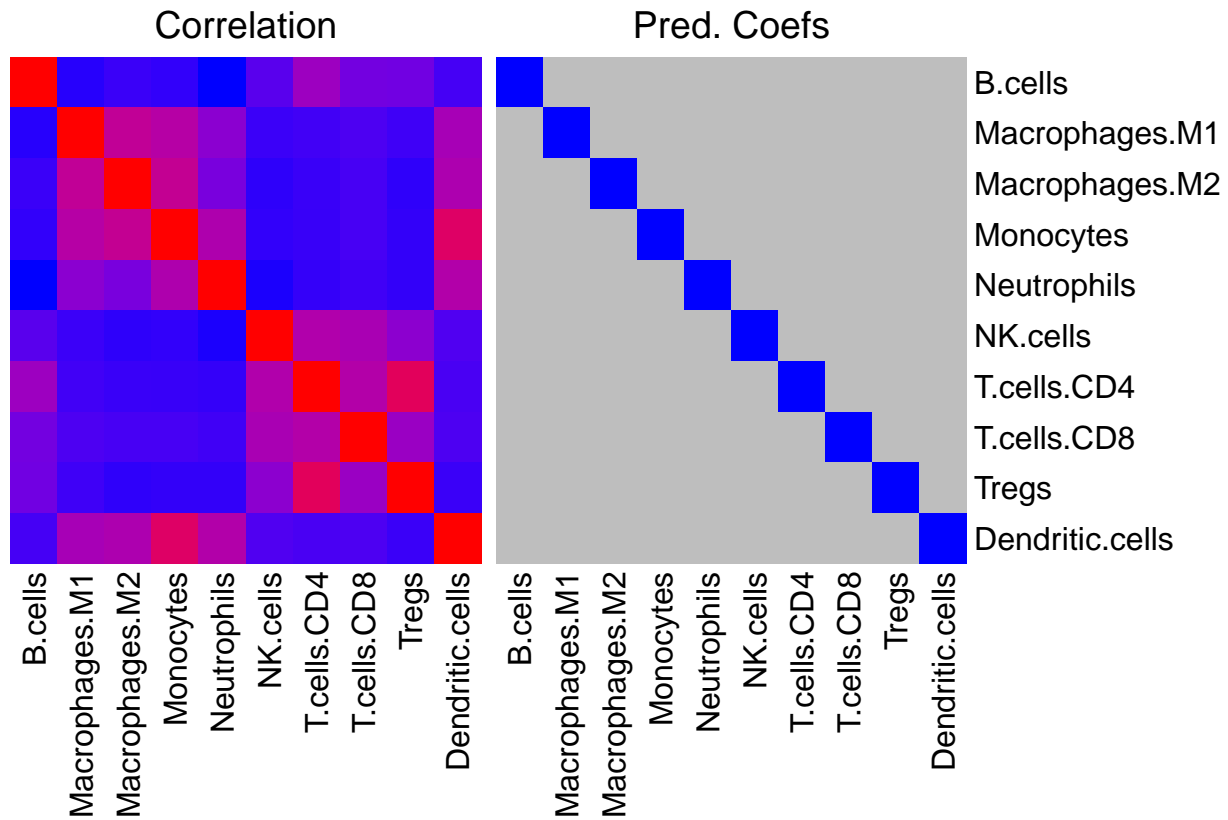
How to performe a molecular signature SelfTest analysis with MIX-TURE ?

A self test analysis is intended to evaluate if the signature profiles can be accurately predicted by using MIXTURE. This means that each signature profiles (Columns) will be assumed as a pure cell mixture profile, thus, the algorithm should only provide one coefficients as being 1 for each signature profile

```
library(MIXTURE)
# Load the TIL10 signature from Finotello et al.
data(TIL10)
# Signature Format
head(TIL10)
```

##	B.cells	Macrophages.M1	Macrophages.M2	Monocytes	Neutrophils
## ABCB4	22.71382	2.2317995	1.9715545	0.10599474	0.00000000
## ADAM28	138.12232	44.5491734	4.8615437	6.15419539	6.92361545
## ADAM6	311.52431	0.1855003	0.3871935	0.03094342	0.03136992
## AFF3	74.37987	1.4194759	1.7991814	2.90259698	1.79219822
## AKAP2	90.53809	17.5227275	3.3263218	0.22109067	0.00000000
## ARHGAP24	48.92311	10.4272520	2.5223667	17.63984171	3.12847521
##	NK.cells	T.cells.CD4	T.cells.CD8	Tregs	Dendritic.cells
## ABCB4	0.23659583	0.04195356	0.10700210	0.04641025	2.14790312
## ADAM28	16.03814946	0.08486099	0.07933068	0.66266673	18.91589470
## ADAM6	0.50123572	0.48061524	0.39069977	0.31171570	0.07778422
## AFF3	5.31305821	3.33874243	1.67199536	3.41073424	9.31677002
## AKAP2	6.84541045	5.89055945	1.27121635	18.36339914	3.30470201
## ARHGAP24	0.05250449	0.01065365	0.00000000	0.21506861	2.34976504

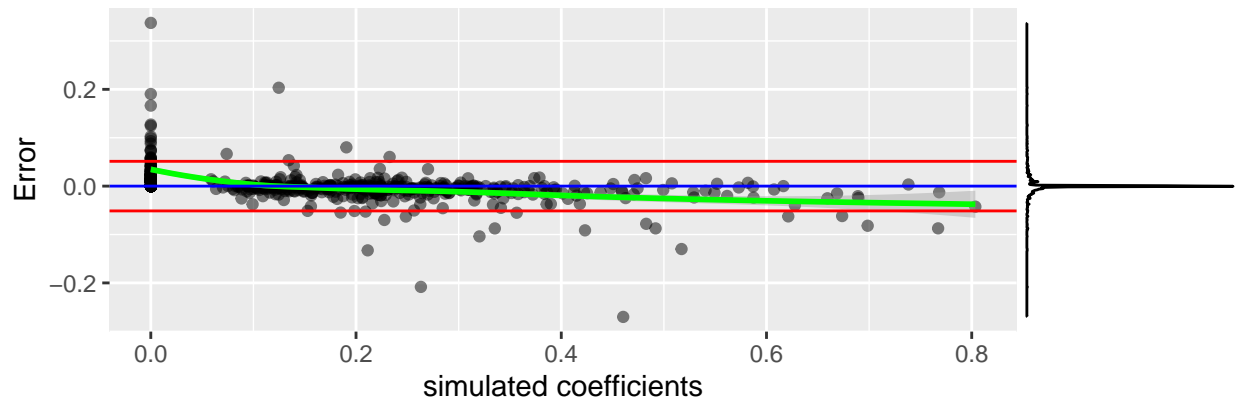
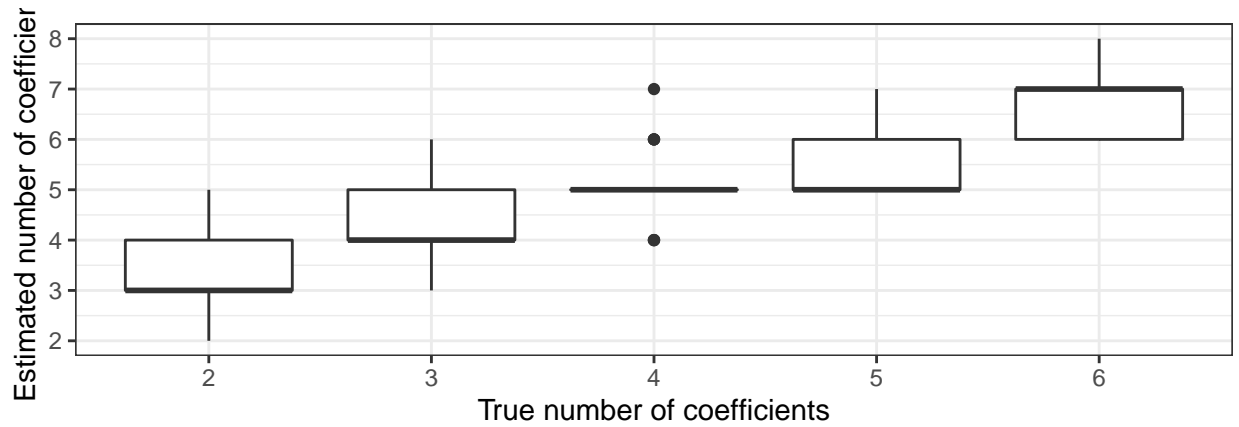
```
SelfTest(TIL10)
```



How to generate simulated mixture profiles from a molecular signature and test MIXTURE algorithm ?

```
# Run the MIXTURE on simulated samples built from the given signature
res <- SimulationTest(signatureMatrix = TIL10,
  maxCoefs = 6, #maximum number of cell types to include in the mixture (from 2 to 10)
  maxSamples = 100,
  noisy = TRUE, #Add a noisy profile to the simulated pure mix
  useCores=3L)

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



```
# Getting simulated data
# Simulated Samples
dim(res$SimulatedData$M)
```

```
## [1] 170 100
```

```
head(res$SimulatedData$M[, 1:4])
```

```
##           Sim1      Sim2      Sim3      Sim4
## ABCB4      79.68234  4.5027387 0.7352097  5.010265
## ADAM28     29.14865 14.5505854 6.5968228 28.592595
## ADAM6      44.83458  0.2877625 0.1237566 53.675810
## AFF3       16.10404  2.3535283 5.8373048 15.400226
## AKAP2      14.65200 50.4921573 8.3606223 21.938721
## ARHGAP24   17.15537  7.0740253 5.2074914 14.834459
```

```
#Simulated betas (coefficients)
head(res$SimulatedData$B)
```

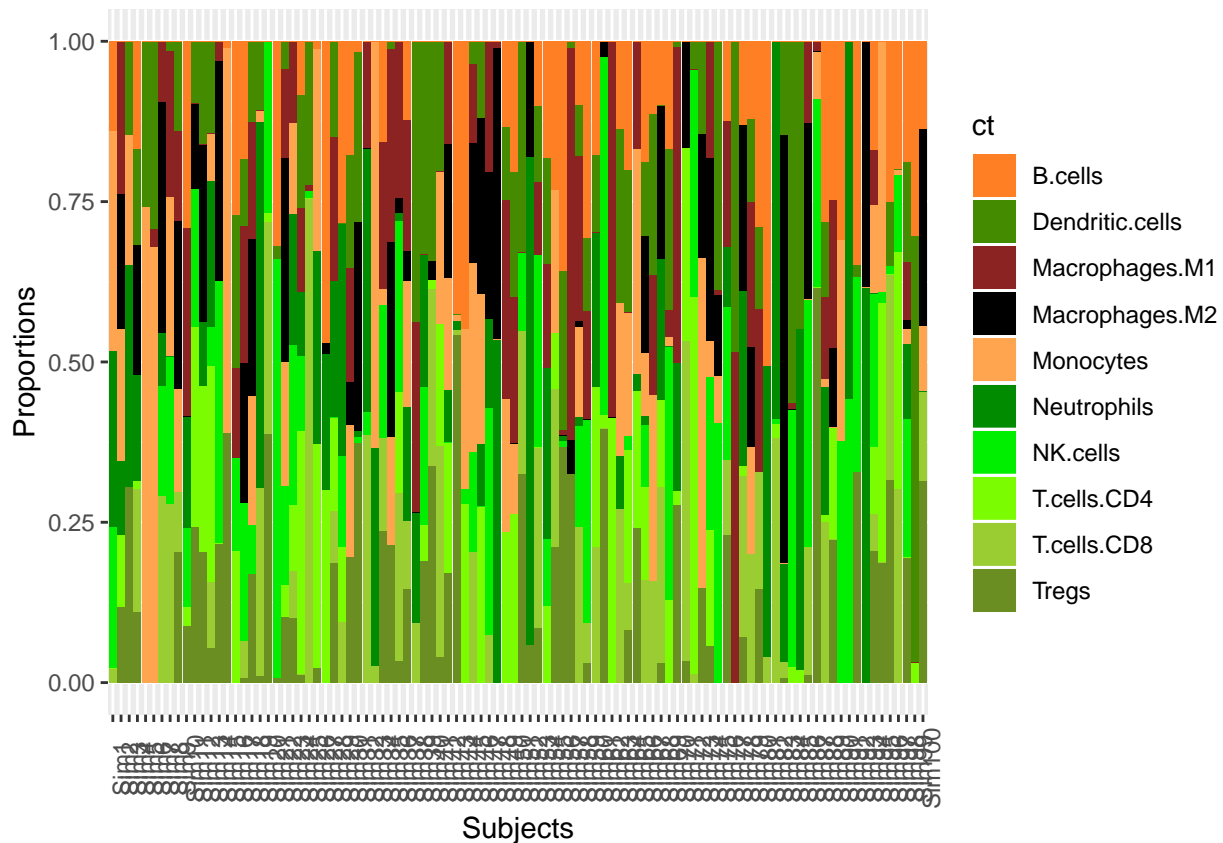
```
##           B.cells Macrophages.M1 Macrophages.M2 Monocytes Neutrophils NK.cells
## [1,] 0.1433304      0.0000000      0.0000000 0.3458298  0.2831312 0.2277085
## [2,] 0.0000000      0.2401428      0.2136496 0.2054409  0.1136213 0.0000000
## [3,] 0.0000000      0.0000000      0.0000000 0.2028676  0.3459445 0.0000000
## [4,] 0.1705711      0.0000000      0.2098957 0.0000000  0.1712381 0.0000000
## [5,] 0.0000000      0.0000000      0.0000000 0.7383492  0.0000000 0.0000000
## [6,] 0.0000000      0.0000000      0.0000000 0.7672522  0.0000000 0.0000000
##           T.cells.CD4 T.cells.CD8      Tregs Dendritic.cells
## [1,] 0.0000000      0.0000000 0.0000000      0.0000000
## [2,] 0.1150141      0.0000000 0.1121313      0.0000000
```

```
## [3,] 0.0000000 0.0000000 0.3057801 0.1454078
## [4,] 0.0000000 0.1953402 0.1108513 0.1421036
## [5,] 0.0000000 0.0000000 0.0000000 0.2616508
## [6,] 0.0000000 0.0000000 0.0000000 0.2327478
```

```
#Retrieving MIXTURE results
dim(GetMixture(res$MIXTURE))
```

```
## [1] 100 10
```

```
# Displaying the cell type proportions
ProportionPlot(res$MIXTURE)
```



How to Download CDC1000 pure cell lines data to test the MIXTURE algorithm ?

```
library(data.table)
library(openxlsx)
library(org.Hs.eg.db)
library(limma)

#download and unzip expression from
url <- "https://www.cancerrxgene.org/gdsc1000/GDSC1000_WebResources/
/Data/preprocessed/Cell_line_RMA_proc_basalExp.txt.zip"
fname <- basename(url)
# this will save the downloaded file in your working directory
```

```

download.file(url = url, destfile = fname, method = "auto")
unzip(fname)

#load expression matrix
a.data<- fread("Cell_line_RMA_proc_basalExp.txt")

#update annotation
b.annot<- a.data[,1:2]
colnames(b.annot)<- c("symbol", "name")
columns(org.Hs.eg.db)
b.entrezids <- mapIds(org.Hs.eg.db, keys=b.annot$symbol, column="ENTREZID",
                      keytype="SYMBOL", multiVals="first")
b.entrezids[sapply(b.entrezids, is.null)]<- NA
b.annot$entrezid<- unlist(b.entrezids)

#fix colnames
colnames(a.data)<- gsub("DATA.", "", colnames(a.data))

#make elist
b.elist<- new("EList", list(E=a.data[,-c(1,2)], genes= b.annot))
dim(b.elist)

#remove missing entrezid
b.elist<- b.elist[which(!is.na(b.elist$genes$entrezid)),]

#combine repeated entrezid expression
b.elist<- averseps(x = b.elist, ID = b.elist$genes$entrezid)
# this will save the cell lines gene expression matrix in your working directory
saveRDS(b.elist, file = "celllines.rds")

```

Once the file has been downloaded and stored in your hard disk, the you can proceed to analyze

```

# we will run over the first 10 cell lines
library(limma)
library(MIXTURE)
cells <- readRDS("celllines.rds")
# gene expression cell line data is in log2 scale
# so we anti log the data
M <- 2^cells$E

cn <- colnames(cells$E)
cn[1:10]

## [1] "906826" "687983" "910927" "1240138" "1240139" "906792" "910688"
## [8] "1240135" "1290812" "907045"

rownames(M) <- cells$genes$symbol
# we will only process the first ten cell lines
cells.mix <- MIXTURE(expressionMatrix = M[, 1:10], signatureMatrix = LM22, useCores = 3L)
head(GetMixture(cells.mix))

##           B cells naive B cells memory Plasma cells T cells CD8 T cells CD4 naive
## 906826           0           0 0.31962758           0 0.0000000
## 687983           0           0 0.41648729           0 0.0000000
## 910927          NA          NA           NA          NA           NA

```

## 1240138	0	0	0.00000000	0	0.2482379
## 1240139	0	0	0.09858075	0	0.2474324
## 906792	0	0	0.26761851	0	0.0000000
##	T cells CD4 memory resting T cells CD4 memory activated				
## 906826	0.4185829			0	
## 687983	0.0000000			0	
## 910927	NA			NA	
## 1240138	0.0000000			0	
## 1240139	0.0000000			0	
## 906792	0.0000000			0	
##	T cells follicular helper T cells regulatory (Tregs)				
## 906826	0.0000000			0	
## 687983	0.2297501			0	
## 910927	NA			NA	
## 1240138	0.0000000			0	
## 1240139	0.0000000			0	
## 906792	0.2209942			0	
##	T cells gamma delta NK cells resting NK cells activated Monocytes				
## 906826	0	0		0	0
## 687983	0	0		0	0
## 910927	NA	NA		NA	NA
## 1240138	0	0		0	0
## 1240139	0	0		0	0
## 906792	0	0		0	0
##	Macrophages M0 Macrophages M1 Macrophages M2 Dendritic cells resting				
## 906826	0.00000000	0	0		0
## 687983	0.00000000	0	0		0
## 910927	NA	NA	NA		NA
## 1240138	0.05308345	0	0		0
## 1240139	0.00000000	0	0		0
## 906792	0.07261115	0	0		0
##	Dendritic cells activated Mast cells resting Mast cells activated				
## 906826	0.05252302		0.2092665		0
## 687983	0.35376259		0.0000000		0
## 910927	NA		NA		NA
## 1240138	0.26332460		0.4353540		0
## 1240139	0.25975621		0.0000000		0
## 906792	0.43877617		0.0000000		0
##	Eosinophils Neutrophils				
## 906826	0.0000000	0			
## 687983	0.0000000	0			
## 910927	NA	NA			
## 1240138	0.0000000	0			
## 1240139	0.3942307	0			
## 906792	0.0000000	0			

How to process TCGA Data ?

```
#Download FPKM data with TCGAbioLinks
library(TCGAbiolinks)
library(SummarizedExperiment)

query_all_brca_fpkm <- GDCquery(project = "TCGA-BRCA",
```



```

        data.category = "Transcriptome Profiling",
        data.type = "Gene Expression Quantification",
        experimental.strategy = "RNA-Seq",
        workflow.type = "HTSeq - FPKM")

GDCdownload(query_all_brca_fpkm)
data_all_brca_fpkm <- GDCprepare(query_all_brca_fpkm, save = TRUE, save.filename = "BRCA.All.FPKM.rda")
##Chage the directory according to wahre you save your downloaded data

## Loading required package: GenomicRanges
## Loading required package: stats4
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
## The following object is masked from 'package:limma':
##
##     plotMA
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
## The following object is masked from 'package:gridExtra':
##
##     combine
## 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, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which, which.max, which.min
## Loading required package: S4Vectors
##
## Attaching package: 'S4Vectors'
## The following object is masked from 'package:plyr':
##
##     rename
## The following object is masked from 'package:base':
##
##     expand.grid
## Loading required package: IRanges

```

```
##
## Attaching package: 'IRanges'
## The following object is masked from 'package:plyr':
##
##     desc
## Loading required package: GenomeInfoDb
## Loading required package: Biobase
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname)".
## Loading required package: DelayedArray
## Loading required package: matrixStats
##
## Attaching package: 'matrixStats'
## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians
## The following object is masked from 'package:plyr':
##
##     count
##
## Attaching package: 'DelayedArray'
## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
## The following objects are masked from 'package:base':
##
##     aperm, apply, rowsum
#set path where your BRCA.All.FPKM.rda is located
load(file.path(path, "BRCA.All.FPKM.rda"))

# Prepare your gene expression data matrix
gene_annot_all <- rowData(data)
exp.all <- assay(data)
target.data.all <- as.data.frame(colData(data))
tumor.type <- do.call(rbind, str_split(as.character(target.data.all$barcode), "-"))
# Tissue type distribution
table(tumor.type[,4])

##
## 01A 01B 01C 06A 11A 11B
## 1077 24 1 7 99 14

rownames(exp.all)[1:10]

## [1] "ENSG000000000003" "ENSG000000000005" "ENSG000000000419" "ENSG000000000457"
```

```
## [5] "ENSG00000000460" "ENSG00000000938" "ENSG00000000971" "ENSG00000001036"
## [9] "ENSG00000001084" "ENSG00000001167"
```

```
# only ensembl gene ids
exp.all <- exp.all[gene_annot_all$.ensembl_gene_id,]
#MIXTURE use rownames to identify genes and match with the molecular signature
rownames(exp.all) <- gene_annot_all$.external_gene_name
rownames(exp.all)[1:10]
```

```
## [1] "TSPAN6" "TNMD" "DPM1" "SCYL3" "C1orf112" "FGR"
## [7] "CFH" "FUCA2" "GCLC" "NFYA"
```

```
#an E list from limma package
a.data <- new("EList",list(E = exp.all, genes = gene_annot_all, targets = target.data.all))
a.data$targets$TissueType <- do.call(rbind,str_split(as.character(target.data.all$barcode),"-"))[,4]
# processing with MIXTURE
brca.mix <- MIXTURE(expressionMatrix = a.data$E, signatureMatrix = LM22, useCores = 6L)

round(apply(GetMixture(brca.mix),2,function(x) sum(x>0))/nrow(GetMixture(brca.mix)),2)
```

```
##          B cells naive          B cells memory
##                0.55                0.06
##          Plasma cells          T cells CD8
##                0.57                0.77
##          T cells CD4 naive  T cells CD4 memory resting
##                0.00                0.44
## T cells CD4 memory activated  T cells follicular helper
##                0.25                0.58
## T cells regulatory (Tregs)    T cells gamma delta
##                0.48                0.18
##          NK cells resting    NK cells activated
##                0.05                0.09
##          Monocytes          Macrophages M0
##                0.18                0.80
##          Macrophages M1    Macrophages M2
##                0.95                0.99
##          Dendritic cells resting  Dendritic cells activated
##                0.44                0.14
##          Mast cells resting    Mast cells activated
##                0.67                0.04
##          Eosinophils          Neutrophils
##                0.01                0.05
```

```
brca.prop <- GetMixture(brca.mix)
df <- data.frame(Proportions = c(brca.prop), CellTypes = rep(colnames(brca.prop), each = nrow(brca.prop)))

ggplot(df,aes(x=CellTypes, y=Proportions, fill = CellTypes)) + geom_boxplot(outlier.size = 0.7) +
  theme(axis.text.x = element_text(angle=45, vjust = 1,hjust = 1))
```

