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

```
knitr::opts_chunk$set(echo = TRUE)
```

How to install MIXTURE

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

Testing MIXTURE in a SelfTest

```
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,          #N x ncol(signatureMatrix) gene expresion matrix
                                                      ##rownames(M) should be the GeneSymbols
              signatureMatrix = LM22,                 #the gene signature matrix (W) such that M = W*be
                                                      #(i.e the LM22 from Newman et al)
              # iter = 10,                            #iterations for the statistical test (null distri
              functionMixture = nu.svm.robust.RFE,    #cibersort, nu.svm.robust.rfe, ls.rfe.abbas,
              useCores = 10L,                         #cores for parallel processing/ if using windows
              verbose = TRUE,                         #TRUE or FALSE messages
              nullDist = "PopulationBased"           #"none" or "PopulationBased" if the statistical te
                                                      #be performed
                                                      )    #EXCEL file name to stare the results
```

```
##
## 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)
```

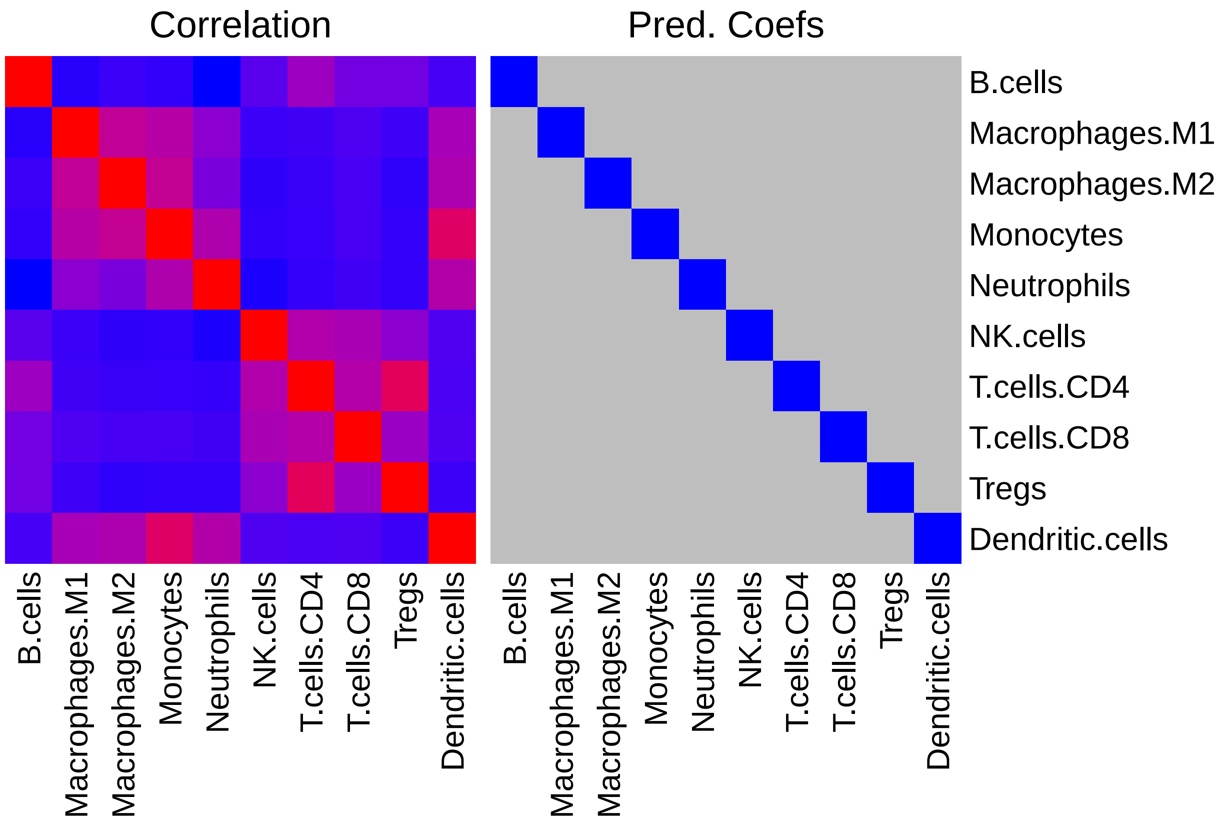
```
## [1] "Subjects"       "PermutedMetrix" "method"         "usedGenes"
## [5] "p.values"
```

How to test a new signature matrix with MIXTURE algorithm?

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

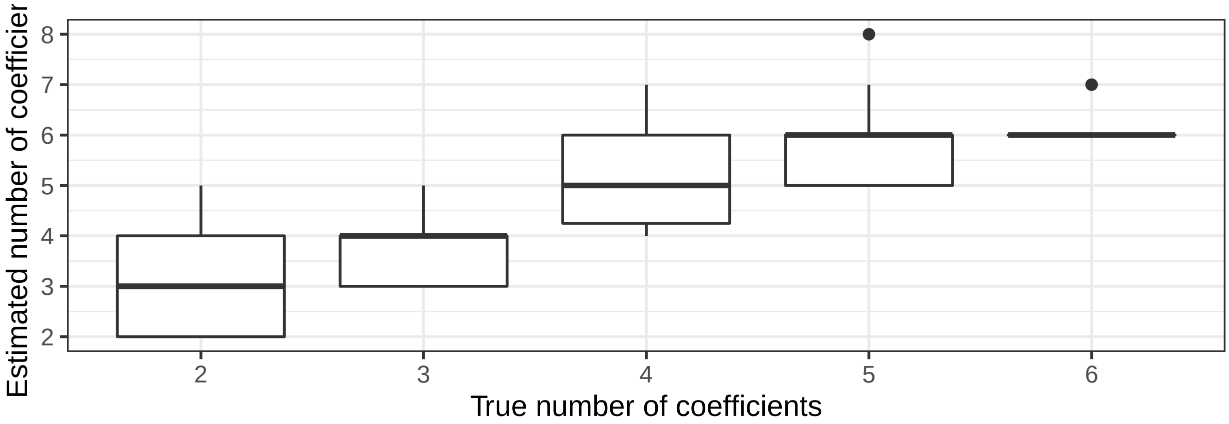
```
##             B.cells Macrophages.M1 Macrophages.M2   Monocytes Neutrophils
## ABCB4      22.71382      2.2317995      1.9715545  0.10599474  0.00000000
## ADAM28    138.12232     44.5491734      4.8615437  6.15419539  6.92361545
## ADAM6     311.52431      0.1855003      0.3871935  0.03094342  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
```

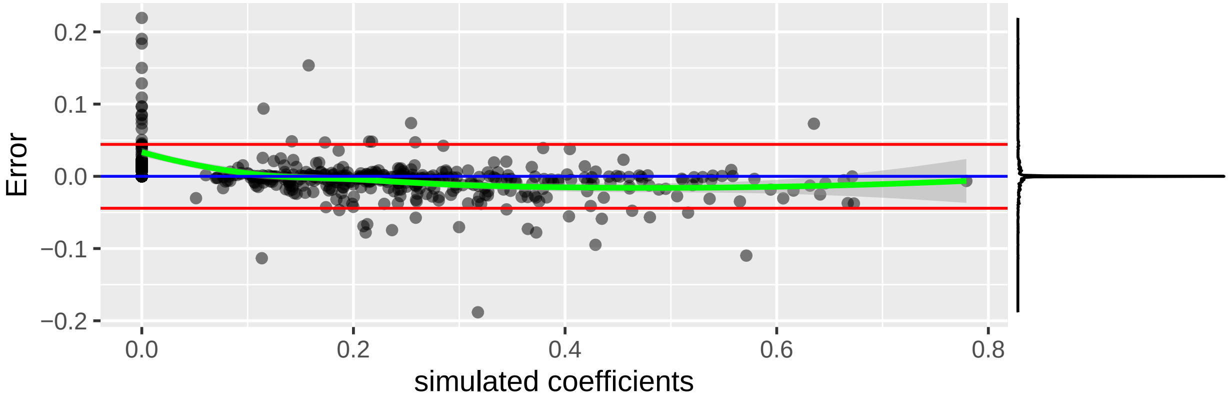
```
SelfTest(TIL10)
```



```
# Run the MIXTURE on simulated samples built from the given signature
res <- SimulationTest(signatureMatrix = TIL10, maxCoefs = 6, maxSamples = 100, noisy = TRUE, 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      2.410244 271.83051  75.3730596  17.23333
## ADAM28    25.723002  82.00708  16.3382705  71.79569
## ADAM6      9.457783 120.84765   0.3415102 106.15500
## AFF3      14.589338  29.98622 100.1391331  35.33161
## AKAP2      8.892805 583.78859 152.4630496  36.12702
## ARHGAP24   6.210987  76.36304  15.0545202 171.13699
```

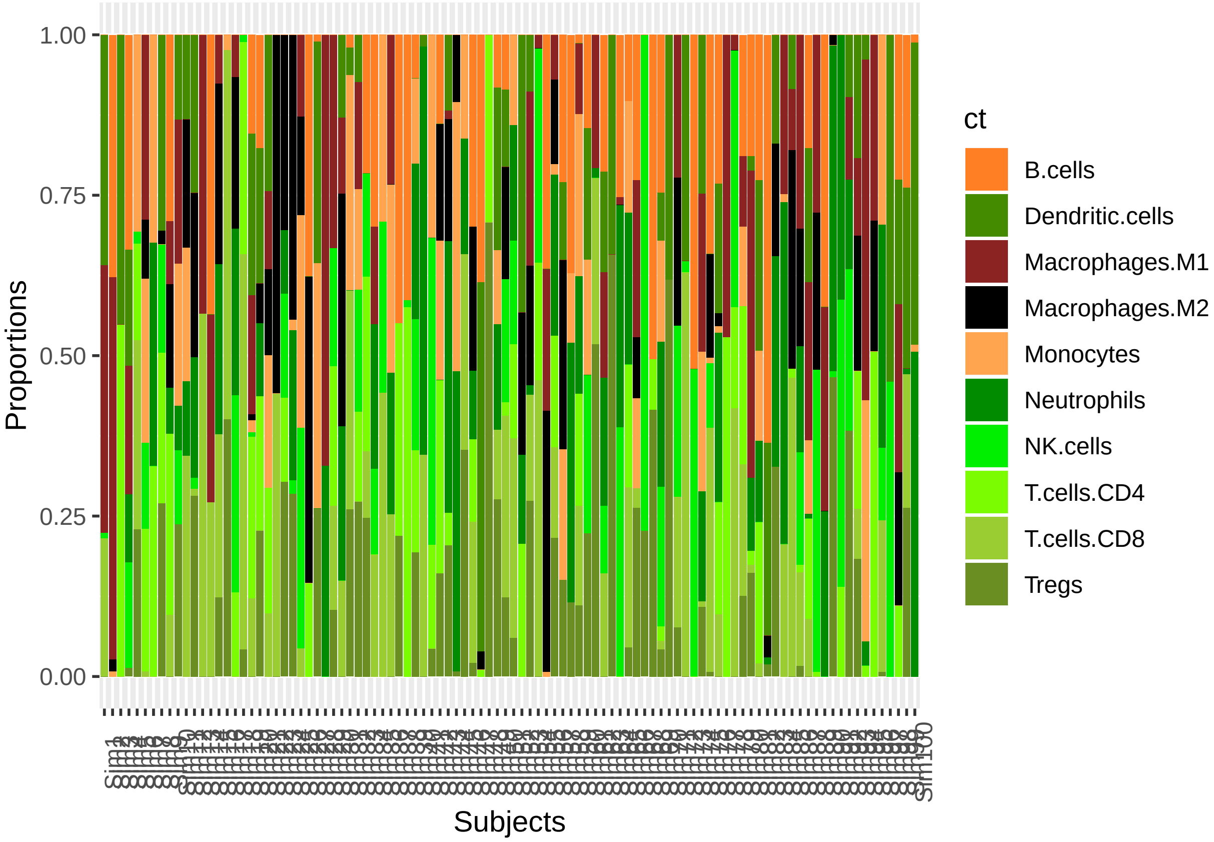
```
#Simulated betas (coefficients)
head(res$SimulatedData$B)
```

```
##        B.cells Macrophages.M1 Macrophages.M2 Monocytes Neutrophils  NK.cells
## [1,] 0.0000000      0.4185181     0.00000000 0.0000000   0.0000000 0.0000000
## [2,] 0.3841530      0.6158470     0.00000000 0.0000000   0.0000000 0.0000000
## [3,] 0.0000000      0.0000000     0.00000000 0.0000000   0.0000000 0.0000000
## [4,] 0.3398936      0.2037408     0.00000000 0.0000000   0.1058201 0.1667459
## [5,] 0.0000000      0.0000000     0.00000000 0.3274336   0.0000000 0.0000000
## [6,] 0.0000000      0.2897588     0.09014297 0.2571864   0.0000000 0.1362664
##      T.cells.CD4 T.cells.CD8     Tregs Dendritic.cells
## [1,]   0.0000000   0.2124377 0.0000000       0.3690442
## [2,]   0.0000000   0.0000000 0.0000000       0.0000000
## [3,]    0.5483092   0.0000000 0.0000000        0.4516908
## [4,]    0.0000000   0.0000000 0.0000000        0.1837997
## [5,]    0.0000000   0.3727567 0.2998097        0.0000000
## [6,]    0.2266453   0.0000000 0.0000000        0.0000000
```

```
#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

```
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_pro
fname <- basename(url)
download.file(url  =  url, destfile = fname, method = "auto")
unzip(fname)

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

#update annotation
```

```r
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$symbol, column="ENTREZID", keytype="SYMBOL", multiVals
b.entrezids[sapply(b.entrezids, is.null)]<- NA
b.annot$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<- avereps(x = b.elist, ID = b.elist$genes$entrezid)

saveRDS(b.elist, file = "data/celllines.rds")
```