CellMix FAQ and HowTos

Renaud Gaujoux

June 4, 2013

Abstract

This vignette contains hints and pointers on how to perform common tasks with the *CellMix* package. In particular, it will incorporate answers to user queries that would come up over time.

Contents

1 Marker lists **1**
1.1 How to list all available marker gene lists? 1
1.2 How to load a registered marker gene list? 1
1.3 How to get a summary view of a marker gene list? 2
1.4 How to subset marker gene lists? 2
1.5 How to convert MarkerList objects into plain lists? 4
1.6 How to create MarkerList objects 4

2 Datasets **5**
2.1 How to list all available datasets? 6
2.2 How to load a dataset? 6
2.3 How to retrieve data from an ExpressionMix object? 6

3 Deconvolution methods **7**
3.1 How to list all available methods? 7

1 Marker lists

The *CellMix* package ships with a set of marker lists gathered from a variety of public databases. This section shows how to perform some common task with these gene lists.

1.1 How to list all available marker gene lists?

```
# list access keys
cellMarkers()
```

```
## [1] "IRIS"      "Abbas"     "TDDB_HS"   "TDDB_RN"   "Palmer"    "HaemAtlas"
## [7] "TIGER"     "VeryGene"  "Grigoryev"
```

```
# show full property table
cellMarkersInfo()
```

1.2 How to load a registered marker gene list?

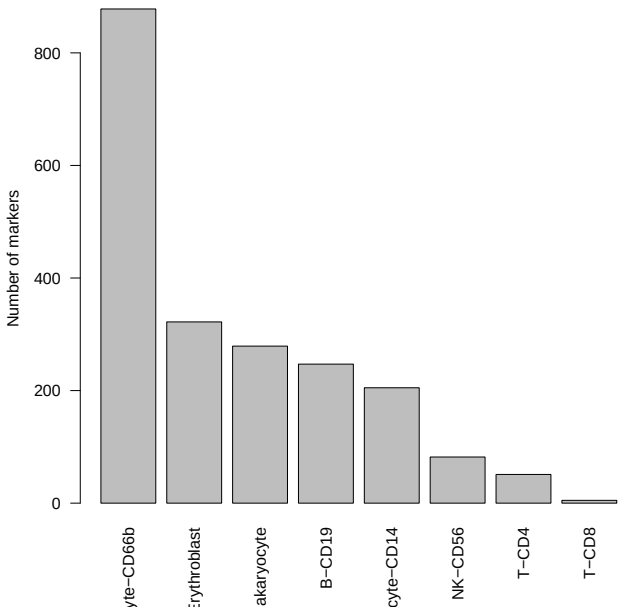
```
# load HaemAtlas markers
m <- cellMarkers("HaemAtlas")
# or
m <- MarkerList("HaemAtlas")
```

1.3 How to get a summary view of a marker gene list?

```
# load
m <- MarkerList("HaemAtlas")
# show summary
summary(m)

## <object of class MarkerList>
## Types: 8 ['B-CD19', 'Erythroblast', ..., 'T-CD8']
## Mode: character
## Markers: 2069
## IDtype: .Illumina ['ILMN_1793637', 'ILMN_1663575', ..., 'ILMN_1815673']
## Source: illuminaHumanv2.db
## Breakdown:
##            B-CD19      Erythroblast Granulocyte-CD66b     Megakaryocyte
##               247               322               878               279
##     Monocyte-CD14          NK-CD56            T-CD4            T-CD8
##               205                82                51                5

# plot number of markers for each cell type
barplot(m)
```



1.4 How to subset marker gene lists?

```r
# subset the cell types
summary(m[1:3])
```

```
## <object of class MarkerList>
## Types: 3 ['B-CD19', 'Erythroblast', 'Granulocyte-CD66b']
## Mode: character
## Markers: 1447
## IDtype: .Illumina ['ILMN_1793637', 'ILMN_1663575', ..., 'ILMN_1857413']
## Source: illuminaHumanv2.db
## Breakdown:
##            B-CD19      Erythroblast Granulocyte-CD66b
##               247               322               878
```

```r
# Take only first n markers of each cell type
summary(m[, 1:3])
```

```
## <object of class MarkerList>
## Types: 8 ['B-CD19', 'Erythroblast', ..., 'T-CD8']
## Mode: character
## Markers: 24
## IDtype: .Illumina ['ILMN_1793637', 'ILMN_1663575', ..., 'ILMN_1726589']
## Source: illuminaHumanv2.db
## Breakdown:
##            B-CD19      Erythroblast Granulocyte-CD66b     Megakaryocyte
##                 3                 3                 3                 3
##     Monocyte-CD14           NK-CD56            T-CD4            T-CD8
##                 3                 3                 3                 3
```

```r
# subset markers that are present in some dataset => this converts/maps
# IDs if necessary
x <- ExpressionMix("GSE11058")
subset(m, x, verbose = TRUE)
```

```
## # Converting 2069 markers from Annotation (illuminaHumanv2.db) to Annotation (hgu133plus2.db)
## # Processing 2069 markers from Annotation (illuminaHumanv2.db) to Annotation (hgu133plus2.db)

## Matching character marker list against 54675 strings ['1007_s_at', '1053_at', ...,
'AFFX-TrpnX-M_at'] ...
## OK [1643/1643 matche(s)]

## <object of class: MarkerList>
## Types: B-CD19, Erythroblast, ..., T-CD8 (total: 8)
## Mode: character
## setName: HaemAtlas
## geneIds: 206513_at, 207655_s_at, ..., 221126_at (total: 1643)
## geneIdType: Annotation (hgu133plus2.db)
## collectionType: Null
## geneValues: NA
## details: use 'details(object)'
```

1.5 How to convert MarkerList objects into plain lists?

`MarkerList` objects can be converted to plain list objects using the methods `geneIds`, or `geneValues` if one wants to keep numeric scores associated with each marker:

```
# load marker list that contains scores
ml <- cellMarkers("TIGER")

# plain list dropping values
l <- geneIds(ml)
str(head(l))

## List of 6
##  $ bladder    : chr [1:199] "Hs.405866" "Hs.281295" "Hs.534503" "Hs.549507" ...
##  $ blood      : chr [1:413] "Hs.552036" "Hs.557039" "Hs.23118" "Hs.448401" ...
##  $ bone       : chr [1:114] "Hs.518726" "Hs.2936" "Hs.1584" "Hs.98785" ...
##  $ bone_marrow: chr [1:269] "Hs.474119" "Hs.458263" "Hs.289232" "Hs.200929" ...
##  $ brain      : chr [1:342] "Hs.7124" "Hs.12440" "Hs.13284" "Hs.20945" ...
##  $ cervix     : chr [1:216] "Hs.343864" "Hs.74082" "Hs.256632" "Hs.528920" ...


# plain list keeping values
l <- geneValues(ml)
str(head(l))

## List of 6
##  $ bladder    : Named num [1:199] 99.9 78.5 52.3 47.1 45.8 ...
##   ..- attr(*, "names")= chr [1:199] "Hs.405866" "Hs.281295" "Hs.534503" "Hs.549507" ...
##  $ blood      : Named num [1:413] 68.3 68.3 66.4 64.7 63 ...
##   ..- attr(*, "names")= chr [1:413] "Hs.552036" "Hs.557039" "Hs.23118" "Hs.448401" ...
##  $ bone       : Named num [1:114] 50.7 40.6 40 28.9 27.4 ...
##   ..- attr(*, "names")= chr [1:114] "Hs.518726" "Hs.2936" "Hs.1584" "Hs.98785" ...
##  $ bone_marrow: Named num [1:269] 88.6 71.3 58.3 54.4 50.2 ...
##   ..- attr(*, "names")= chr [1:269] "Hs.474119" "Hs.458263" "Hs.289232" "Hs.200929" ...
##  $ brain      : Named num [1:342] 7.27 7.27 7.27 7.27 7.27 ...
##   ..- attr(*, "names")= chr [1:342] "Hs.7124" "Hs.12440" "Hs.13284" "Hs.20945" ...
##  $ cervix     : Named num [1:216] 66.1 44.5 41.3 34.4 33.1 ...
##   ..- attr(*, "names")= chr [1:216] "Hs.343864" "Hs.74082" "Hs.256632" "Hs.528920" ...
```

1.6 How to create MarkerList objects

`MarkerList` objects can be manually created from a variety of format/object types, using the factory generic `MarkerList()`:

```
# basic data
m <- setNames(letters[1:10], rep(c("CT1", "CT2"), 5))
m

## CT1 CT2 CT1 CT2 CT1 CT2 CT1 CT2 CT1 CT2
## "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"


# from character vector with names corresponding to cell types
ml <- MarkerList(m)
geneIds(ml)
```

```
## $CT1
## [1] "a" "c" "e" "g" "i"
##
## $CT2
## [1] "b" "d" "f" "h" "j"


# from a list
m_list <- split(m, names(m))
ml <- MarkerList(m_list)
geneIds(ml)

## $CT1
## [1] "a" "c" "e" "g" "i"
##
## $CT2
## [1] "b" "d" "f" "h" "j"


# from a delimited text file: marker names, cell type
mf <- cbind(m, names(m))
mf

##     m
## CT1 "a" "CT1"
## CT2 "b" "CT2"
## CT1 "c" "CT1"
## CT2 "d" "CT2"
## CT1 "e" "CT1"
## CT2 "f" "CT2"
## CT1 "g" "CT1"
## CT2 "h" "CT2"
## CT1 "i" "CT1"
## CT2 "j" "CT2"

write.table(mf, file = "markers.txt", row.names = FALSE)
ml <- MarkerList(file = "markers.txt", header = TRUE)
geneIds(ml)

## $CT1
## [1] "a" "c" "e" "g" "i"
##
## $CT2
## [1] "b" "d" "f" "h" "j"

file.remove("markers.txt")

## [1] TRUE
```

2 Datasets

The *CellMix* package ships a with a set of pre-processing pipelines for some public datasets on GEO, that can be used as benchmark data for gene expression deconvolution methods.

2.1 How to list all available datasets?

```
# list access keys
gedData()
```

```
##  [1] "GSE29832"   "GSE24223"   "GSE19830"   "GSE20300"   "GSE5350"
##  [6] "GSE11057"   "GSE11058"   "GSE22886_A" "GSE22886_B" "GSE33076"
## [11] "GSE3649"    "E-TABM-633" "GSE24759"
```

```
# show full property table
gedDataInfo()
```

2.2 How to load a dataset?

Datasets are loaded using the function `ExpressionMix`. This requires an internet connection. The first call will create a data directory in the user home directory, where all files related to datasets will be stored (GSE matrix files, GPL files, cache files, etc..)

```
# load GSE29832 from Gong et al. (2011)
mix <- ExpressionMix("GSE29832")
mix
```

```
## ExpressionMix (storageMode: lockedEnvironment)
## assayData: 54675 features, 15 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: GSM739214 GSM739215 ... GSM739228 (15 total)
##   varLabels: title geo_accession ... Biorep (36 total)
##   varMetadata: labelDescription
## featureData
##   featureNames: 1007_s_at 1053_at ... AFFX-TrpnX-M_at (54675
##     total)
##   fvarLabels: ID GB_ACC ... Gene Ontology Molecular Function (16
##     total)
##   fvarMetadata: Column Description labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: hgu133plus2.db
## Composition: 'Blood', 'Breast' (2 total)
```

2.3 How to retrieve data from an ExpressionMix object?

`ExpressionMix` objects are containers for multiple types of data, which can be retrieved with deticated methods. The idea is to hold both gene expression and cell composition data in a single object, facilitating common dataset operations (e.g. subsetting features or samples).

```
# dimensions of an ExpressionMix object
dim(mix)
```

```
##   Features    Samples Components
##      54675         15          2
```

Expression data: it is stored as an `ExpressionSet` object and is accessible via `eset` or `exprs`, if only the expression matrix is needed:

```
class(eset(mix))

## [1] "ExpressionSet"
## attr(,"package")
## [1] "Biobase"

class(exprs(mix))

## [1] "matrix"

dim(exprs(mix))

## [1] 54675    15
```

Cell proportions: if available, they are stored in the mixuture coefficient matrix of the embedded `NMF` model and are accessible with the method `coef`:

```
dim(coef(mix))

## [1]  2 15
```

Cell-specific signatures if available, they are stored in the basis matrix of the embedded `NMF` model and are accessible with the method `basis`:

```
dim(basis(mix))

## [1] 54675     2
```

3 Deconvolution methods

3.1 How to list all available methods?

```
# list access keys
gedAlgorithm()

##  [1] "lsfit"       "cs-lsfit"    "qprog"       "cs-qprog"    "DSA"
##  [6] "csSAM"       "DSection"    "ssKL"        "ssFrobenius" "meanProfile"
## [11] "deconf"
```

```
# show full property table
gedAlgorithmInfo()
```