lute gmc-example

Sean Maden

2023-03-08

This vignette demonstrates how to use the deconvolution generic to obtain cell type proportion estimates using the nnls::nnls() implementation of non-negative least squares (NNLS).

First, load lute and get the example datasets for a reference-based deconvolution method.

library(lute)

```
## Loading required package: SummarizedExperiment
## Loading required package: MatrixGenerics
## Warning: package 'MatrixGenerics' was built under R version 4.2.1
## Loading required package: matrixStats
## Warning: package 'matrixStats' was built under R version 4.2.2
##
## Attaching package: 'MatrixGenerics'
## The following objects are masked from 'package:matrixStats':
##
##
       colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
       colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##
       colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##
       colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##
##
       colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##
       colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##
       colWeightedMeans, colWeightedMedians, colWeightedSds,
##
       colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
       rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##
##
       rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##
       rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##
       rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##
       rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
       rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##
##
       rowWeightedSds, rowWeightedVars
## Loading required package: GenomicRanges
```

```
## Warning: package 'GenomicRanges' was built under R version 4.2.2
## Loading required package: stats4
## Loading required package: BiocGenerics
## Warning: package 'BiocGenerics' was built under R version 4.2.1
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:stats':
##
##
       IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
       anyDuplicated, aperm, 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.max, which.min
## Loading required package: S4Vectors
## Warning: package 'S4Vectors' was built under R version 4.2.2
##
## Attaching package: 'S4Vectors'
## The following objects are masked from 'package:base':
##
##
       expand.grid, I, unname
## Loading required package: IRanges
## Warning: package 'IRanges' was built under R version 4.2.1
##
## Attaching package: 'IRanges'
## The following object is masked from 'package:grDevices':
##
##
       windows
## Loading required package: GenomeInfoDb
## Warning: package 'GenomeInfoDb' was built under R version 4.2.2
```

```
## Loading required package: Biobase
## Warning: package 'Biobase' was built under R version 4.2.1
## Welcome to Bioconductor
##
##
       Vignettes contain introductory material; view with
       'browseVignettes()'. To cite Bioconductor, see
##
       'citation("Biobase")', and for packages 'citation("pkgname")'.
##
## Attaching package: 'Biobase'
## The following object is masked from 'package:MatrixGenerics':
##
##
       rowMedians
## The following objects are masked from 'package:matrixStats':
##
##
       anyMissing, rowMedians
## Loading required package: SingleCellExperiment
lexample <- lute:::.get decon example data()</pre>
s <- lexample[["s"]]
y <- lexample[["y"]]
z \leftarrow lexample[["z"]]
```

Use the constructor function nnlsParam to instantiate the parameters to use nnls::nnls().

```
param <- nnlsParam(s = s, y = y, z = z)</pre>
```

We can inspect the new object param to see details about our deconvolution data inputs.

param

```
## class: nnlsParam
## key deconvolution run info:
##
## marker info:
## signature markers (Gz): 10
## unique marker labels (Gy | Gz): 10
## overlapping marker labels (Gy & Gz): 10
##
## samples info:
## number of bulk samples (J): 1
## sample labels: sample1
## cell size factor properties:
##
## types info:
## number of types (K): 2
## unique type labels: type1;type2
```

Executing the deconvolution generic on param will run nnls::nnls. By default, only the predicted cell type proportions are returned.

```
deconvolution(param)
```

```
## Loading required package: nnls
## Transforming Z signature matrix using provided cell size factors S...
## type1 type2
## 0.48908543 0.05896868
```

To get more extensive results, we can set the attribute return.info==T in param. Running the deconvolution generic now returns a list containing the items proportions (cell type proportion estimates), results (original results returned by the deconvolution function, nnls::nnls in this case), and metadata (listed metadata containing the input data summaries).

```
param@return.info <- T
deconvolution(param)</pre>
```

```
## Transforming Z signature matrix using provided cell size factors S\dots
```

```
## $predictions
##
        type1
                   type2
## 0.48908543 0.05896868
##
## $result.info
## Nonnegative least squares model
## x estimates: 0.4890854 0.05896868
## residual sum-of-squares: 262
## reason terminated: The solution has been computed sucessfully.
##
## $metadata
## $metadata$g
## [1] 10
## $metadata$j
## NULL
##
## $metadata$k
## [1] 2
##
## $metadata$s
## [1]
       1 10
## $metadata$unique.types
## [1] "type1" "type2"
##
## $metadata$markers.y
   [1] "marker1" "marker2"
##
                              "marker3"
                                         "marker4" "marker5" "marker6"
   [7] "marker7" "marker8"
                              "marker9"
                                         "marker10"
##
```

\$metadata\$marker.z
[1] "marker1" "marker2" "marker3" "marker4" "marker5" "marker6"

[7] "marker7" "marker8" "marker9" "marker10"