Package 'ADAPTS'

October 29, 2019

Type Package

Title Automated Deconvolution Augmentation of Profiles for Tissue Specific Cells

Version 0.9.26

Author Samuel A Danziger

Maintainer Samuel A Danziger <sam.danziger@gmail.com>

Description Augments existing or de-novo cell-type signature matrices to deconvolve bulk gene expression data.
Useful for building signature matrices from single cell RNAseq data, determine cell type deconvolution
spillover, and hierarchical deconvolution to use spillover to increase deconvolution accuracy. Please cite:
Danziger SA et al. (2019) ADAPTS: Automated Deconvolution Augmentation of Profiles for Tissue Specific cells <doi:10.1101/633958>. This package expands on the techniques outlined in Newman AM,
Liu CL, Green MR, Gentles AJ, Feng W, Xu Y,
Hoang CD, Diehn M, Alizadeh, AA (2015) <doi:10.1038/nmeth.3337>'s Nature Methods paper: 'Robust enumeration of cell subsets from tissue expression profiles' to allow a user to easily add their own cell types (e.g. a tumor specific cell type) to Newman's LM22 or other signature matrix.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Depends R (>= 3.3.0)

biocViews

Imports missForest, e1071, WGCNA, ComICS, pheatmap, doParallel, quantmod, preprocessCore, pcaMethods, foreach, DeconRNASeq, nnls

Suggests R.rsp

VignetteBuilder R.rsp

NeedsCompilation no

Repository CRAN

Date/Publication 2019-10-29 05:10:02 UTC

R topics documented:

`AugmentSigMatrix` *Make an augmented signature matrix*

Description

Build an augmented signature matrix from an initial signature matrix, source data, and a list of differentially expressed genes (gList). The user might want to modify gList to make certain that particular genes are included in the matrix. The algorithm will be to add one additional gene from each new cell type Record the condition number, and plot those. Will only consider adding rows shared by fullData and newData

newMatData <- AugmentSigMatrix(origMatrix, fullData, newData, gList)

Usage

```
AugmentSigMatrix(origMatrix, fullData, newData, gList, nGenes = 1:100,
  plotToPDF = TRUE, imputeMissing = TRUE, condTol = 1.01,
  postNorm = FALSE, minSumToRem = NA, addTitle = NULL,
  autoDetectMin = FALSE, calcSpillOver = FALSE, pdfDir = tempdir())
```

Arguments

origMatrix	The original signature matrix
fullData	The full data for the signature matrix
newData	The new data to add signatures from
gList	The ordered list of genes from running rankByT() on newData. NOTE: best genes at the bottom!!
nGenes	The number of additional genes to consider (DEFAULT: 1:100)
plotToPDF	Plot the output condition numbers to a pdf file. (DEFAULT: TRUE)
imputeMissing	Set to TRUE to impute missing values. NOTE: adds stoachasiticity (DEFAULT: TRUE)
condTol	Setting higher tolerances will result in smaller numbers extra genes. 1.00 minimizes compliment number (DEFAULT: 1.00)
postNorm	Set to TRUE to normalize new signatures to match old signatures. (DEFAULT: FALSE)
minSumToRem	Set to non-NA to remove any row with the sum(abs(row)) < minSumToRem (DEFAULT: NA)
addTitle	An optional string to add to the plot and savefile (DEFAULT: NULL)
autoDetectMin	Set to true to automatically detect the first local minima. GOOD PRELIMINARY RESULTS (DEAFULT: FALSE)
calcSpillOver	Use the training data to calculate a spillover matrix (DEFAULT: FALSE)
pdfDir	A fold to write the pdf file to if plotToPDF=TRUE (DEFAULT: tempdir())

Value

an augmented cell type signature matrix

Examples

```
#This toy example treats the LM22 deconvolution matrix as if it were all of the data
#  For a real example, look at the vignette or comments in exprData, fullLM22, small LM22
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:200, 1:8]
#Make a fake signature matrix out of 100 genes and the first 8 cell types
smallLM22 <- fullLM22[1:100, 1:8]

#Make fake data representing two replicates of purified Mast.cells
exprData <- ADAPTS::LM22[1:200, c("Mast.cells.resting","Mast.cells.activated")]
colnames(exprData) <- c("Mast.cells", "Mast.cells")
```

```
#Fake source data with replicates for all purified cell types.
#  Note in this fake data set, many cell types have exactly one replicate
fakeAllData <- cbind(fullLM22, as.data.frame(exprData))
gList <- rankByT(geneExpr = fakeAllData, qCut=0.3, oneCore=TRUE)

newSig <- AugmentSigMatrix(origMatrix=smallLM22, fullData=fullLM22, newData=exprData,
    gList=gList, plotToPDF=FALSE)
```

`buildSpilloverMat` *Build a spillover matrix*

Description

Build a spillover matrix, i.e. what do purified samples deconvolve as?

spillExpr <- buildSpilloverMat(refExpr, geneExpr, method='DCQ')

Usage

```
buildSpilloverMat(refExpr, geneExpr, method = "DCQ")
```

Arguments

`refExpr`	The deconvolution matrix, e.g. LM22 or MGSM27
`geneExpr`	The full gene expression for purified cell types. Multiple columns (examples) for each column in the reference expr.
`method`	One of 'DCQ', 'SVMDECON', 'DeconRNASeq', 'proportionsInAdmixture', 'nnls' (DEFAULT: DCQ)

Value

A spillover matrix showing how purified cell types deconvolve

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

spillover <- buildSpilloverMat(refExpr=smallLM22, geneExpr=fullLM22, method='DCQ')
```

clustWspillOver *Cluster with spillover*

Description

Build clusters based on n-pass spillover matrix

Usage

```
clustWspillOver(sigMatrix, geneExpr, nPasses = 100,
  deconMatrices = NULL, method = "DCQ")
```

Arguments

sigMatrix	The deconvolution matrix, e.g. LM22 or MGSM27
geneExpr	The source gene expression matrix used to calculate sigMatrix.
nPasses	The maximum number of iterations for spillToConvergence (DEFAULT: 100)
deconMatrices	Optional pre-computed results from spillToConvergence (DEFAULT: NULL)
method	One of 'DCQ', 'SVMDECON', 'DeconRNASeq', 'proportionsInAdmixture', 'nnls' (DEFAULT: DCQ)

Value

Cell types grouped by cluster

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

clusters <- clustWspillOver(sigMatrix=smallLM22, geneExpr=fullLM22, nPasses=10)
```

collapseCellTypes *Collapse cell yypes*

Description

Collapse the cell types (in rows) to super-classes Including MGSM36 cell types

Usage

```
collapseCellTypes(cellCounts, method = "Pheno4")
```

Arguments

cellCounts	A matrix with cell counts
method	The method for combining cell types ('Default: 'Pheno2') Pheno1: Original cell-type based combinations Pheno2: Original cell-type based combinations, omitting Macrophages Pheno3: Alt Phenotype definitions based on WMB deconvolution correlations Pheno4: Consensus cell types Pheno5: Consensus cell types, combined myeloma & plasma Spillover1: Empirical combinations based on compToLM22source Spillover2: More agressive combination based on empirical combinations based on compToLM22source Spillover3: Combinations determined by spillToConvergence on 36 cell types

Value

a cell estimate matrix with the names changed

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

cellEst <- estCellPercent.DCQ(refExpr=smallLM22, geneExpr=fullLM22)
collapseCounts <- collapseCellTypes(cellCounts=cellEst)
```

---

estCellCounts.nPass *Deconvolve with an n-pass spillover matrix*

Description

curExpr <- estCellCounts.nPass(sigMatrix, deconMatrices)

Usage

```
estCellCounts.nPass(geneExpr, deconMatrices, method = "DCQ")
```

Arguments

geneExpr	The gene expression matrix
deconMatrices	The results from spillToConvergence()
method	One of 'DCQ', 'SVMDECON', 'DeconRNASeq', 'proportionsInAdmixture', 'nnls' (DEFAULT: DCQ)

Value

An estimate of cell counts

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

deconMatrices <- spillToConvergence(sigMatrix=smallLM22, geneExpr=fullLM22, nPasses=10)
cellCounts <- estCellCounts.nPass(geneExpr=fullLM22, deconMatrices=deconMatrices, method='DCQ')
```

`estCellPercent` *Wrapper for deconvolution methods*

Description

A wrapper function to call any of the estCellPercent functions

Usage

```
estCellPercent(refExpr, geneExpr, method = "DCQ", ...)
```

Arguments

`refExpr`	a data frame representing immune cell expression profiles. Each row represents an expression of a gene, and each column represents a different immune cell type. colnames contains the name of each immune cell type and the rownames includes the genes' symbol. The names of each immune cell type and the symbol of each gene should be unique. Any gene with missing expression values must be excluded.
`geneExpr`	a data frame representing RNA-seq or microarray gene-expression profiles of a given complex tissue. Each row represents an expression of a gene, and each column represents a different experimental sample. colnames contain the name of each sample and rownames includes the genes' symbol. The name of each individual sample and the symbol of each gene should be unique. Any gene with missing expression values should be excluded.
`method`	One of 'DCQ', 'SVMDECON', 'DeconRNASeq', 'proportionsInAdmixture', 'nnls' (DEFAULT: DCQ)
`...`	Parameters for estCellPercent.X (e.g. number_of_repeats for .DCQ)

Value

A matrix with cell type estimates for each samples

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

cellEst <- estCellPercent(refExpr=smallLM22, geneExpr=fullLM22)
```

`estCellPercent.DCQ` *DCQ Deconvolution*

Description

Use DCQ to estimate the cell count percentage Requires installation of package 'ComICS' To Do: Also report the standard deviation as a confidence metric

Usage

```
estCellPercent.DCQ(refExpr, geneExpr, marker_set = NULL,
  number_of_repeats = 10, alpha = 0.05, lambda = 0.2)
```

Arguments

`refExpr`	a data frame representing immune cell expression profiles. Each row represents an expression of a gene, and each column represents a different immune cell type. colnames contains the name of each immune cell type and the rownames includes the genes' symbol. The names of each immune cell type and the symbol of each gene should be unique. Any gene with missing expression values must be excluded.
`geneExpr`	a data frame representing RNA-seq or microarray gene-expression profiles of a given complex tissue. Each row represents an expression of a gene, and each column represents a different experimental sample. colnames contain the name of each sample and rownames includes the genes' symbol. The name of each individual sample and the symbol of each gene should be unique. Any gene with missing expression values should be excluded.
`marker_set`	data frames of one column, that includes a preselected list of genes that likely discriminate well between the immune-cell types given in the reference data. (DEFAULT: NULL, i.e. one for each gene in the refExpr)
`number_of_repeats`	using one repeat will generate only one output model. Using many repeats, DCQ calculates a collection of models, and outputs the average and standard deviation for each predicted relative cell quantity. (DEFAULT: 1)
`alpha`	The elasticnet mixing parameter, with 0 <= alpha <= 1. alpha=1 is the lasso penalty, and alpha=0 the ridge penalty. (DEFAULT: 0.05)
`lambda`	A minimum value for the elastic net lambda parameter (DEFAULT: 0.2)

Value

A matrix with cell type estimates for each samples

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

cellEst <- estCellPercent.DCQ(refExpr=smallLM22, geneExpr=fullLM22)
```

`estCellPercent.DeconRNASeq`
DeconRNASeq deconvolution

Description

Use DeconRNASeq to estimate the cell count percentage Performs with similar effectiveness as DCQ, but identifies different proportions of cell-types Requires installation of package 'DeconRNASeq': source("https://bioconductor.org/biocLite.R") biocLite("DeconRNASeq")

<joseph.szustakowski@novartis.com> TGJDS (2013). DeconRNASeq: Deconvolution of Heterogeneous Tissue Samples for mRNA-Seq data. R package version 1.18.0.

cellEst <- estCellPercent.DeconRNASeq(refExpr, geneExpr, marker_set=NULL)

Usage

```
estCellPercent.DeconRNASeq(refExpr, geneExpr, marker_set = NULL)
```

Arguments

`refExpr`	a data frame representing immune cell expression profiles. Each row represents an expression of a gene, and each column represents a different immune cell type. colnames contains the name of each immune cell type and the rownames includes the genes' symbol. The names of each immune cell type and the symbol of each gene should be unique. Any gene with missing expression values must be excluded.
`geneExpr`	a data frame representing RNA-seq or microarray gene-expression profiles of a given complex tissue. Each row represents an expression of a gene, and each column represents a different experimental sample. colnames contain the name of each sample and rownames includes the genes' symbol. The name of each individual sample and the symbol of each gene should be unique. Any gene with missing expression values should be excluded.
`marker_set`	data frames of one column, that includes a preselected list of genes that likely discriminate well between the immune-cell types given in the reference data. (DEFAULT: NULL, i.e. one for each gene in the refExpr)

Value

A matrix with cell type estimates for each samples

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

cellEst <- estCellPercent.DeconRNASeq(refExpr=smallLM22, geneExpr=fullLM22)
```

`estCellPercent.nnls` *Non-negative least squares deconvolution*

Description

Use non-negative least squares regression to deconvolve a sample This is going to be to simple to be useful This might be more interesting if I used non-positive least squares to detect 'other'

Usage

```
estCellPercent.nnls(refExpr, geneExpr)
```

Arguments

`refExpr`	a data frame representing immune cell expression profiles. Each row represents an expression of a gene, and each column represents a different immune cell type. colnames contains the name of each immune cell type and the rownames includes the genes' symbol. The names of each immune cell type and the symbol of each gene should be unique. Any gene with missing expression values must be excluded.
`geneExpr`	a data frame representing RNA-seq or microarray gene-expression profiles of a given complex tissue. Each row represents an expression of a gene, and each column represents a different experimental sample. colnames contain the name of each sample and rownames includes the genes' symbol. The name of each individual sample and the symbol of each gene should be unique. Any gene with missing expression values should be excluded.

Value

A matrix with cell type estimates for each samples

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

cellEst <- estCellPercent.nnls(refExpr=smallLM22, geneExpr=fullLM22)
```

`estCellPercent.proportionsInAdmixture`
WGCNA::proportionsInAdmixture deconvolution

Description

Use R function proportionsInAdmixture to estimate the cell count percentage Uses the 'WGCNA' package

cellEst <- estCellPercent.proportionsInAdmixture(refExpr)

Usage

```
estCellPercent.proportionsInAdmixture(refExpr, geneExpr,
  marker_set = NULL)
```

Arguments

`refExpr`	a data frame representing immune cell expression profiles. Each row represents an expression of a gene, and each column represents a different immune cell type. colnames contains the name of each immune cell type and the rownames includes the genes' symbol. The names of each immune cell type and the symbol of each gene should be unique. Any gene with missing expression values must be excluded.
`geneExpr`	a data frame representing RNA-seq or microarray gene-expression profiles of a given complex tissue. Each row represents an expression of a gene, and each column represents a different experimental sample. colnames contain the name of each sample and rownames includes the genes' symbol. The name of each individual sample and the symbol of each gene should be unique. Any gene with missing expression values should be excluded.
`marker_set`	data frames of one column, that includes a preselected list of genes that likely discriminate well between the immune-cell types given in the reference data. (DEFAULT: NULL, i.e. one for each gene in the refExpr)

Value

A matrix with cell type estimates for each samples

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

cellEst <- estCellPercent.proportionsInAdmixture(refExpr=smallLM22, geneExpr=fullLM22)
```

`estCellPercent.spillOver`

Estimate cell percentage from spillover

Description

Use a spillover matrix to deconvolve a samples

Usage

```
estCellPercent.spillOver(spillExpr, refExpr, geneExpr, method = "DCQ",
  ...)
```

Arguments

`spillExpr`	A spill over matrix, as calculated by buildSpilloverMat(). (e.g. LM22.spillover.csv.gz)
`refExpr`	a data frame representing immune cell expression profiles. Each row represents an expression of a gene, and each column represents a different immune cell type. colnames contains the name of each immune cell type and the rownames includes the genes' symbol. The names of each immune cell type and the symbol of each gene should be unique. Any gene with missing expression values must be excluded.
`geneExpr`	a data frame representing RNA-seq or microarray gene-expression profiles of a given complex tissue. Each row represents an expression of a gene, and each column represents a different experimental sample. colnames contain the name of each sample and rownames includes the genes' symbol. The name of each individual sample and the symbol of each gene should be unique. Any gene with missing expression values should be excluded.
`method`	One of 'DCQ', 'SVMDECON', 'DeconRNASeq', 'proportionsInAdmixture', 'nnls' (DEFAULT: DCQ)
`...`	Parameters for estCellPercent.X (e.g. number_of_repeats for .DCQ)

Value

a matrix of estimate cell type percentages in samples

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

spillover <- buildSpilloverMat(refExpr=smallLM22, geneExpr=fullLM22)
cellEst <- estCellPercent.spillOver(spillExpr=spillover, refExpr=smallLM22, geneExpr=fullLM22)
```

`estCellPercent.svmdecon` *SVMDECON deconvolution*

Description

Use SVMDECON to estimate the cell count percentage Performs considerably worse in deconvolution than DCQ

cellEst <- estCellPercent.svmdecon(refExpr, geneExpr)

Usage

```
estCellPercent.svmdecon(refExpr, geneExpr, marker_set = NULL,
  useOldVersion = F, progressBar = T)
```

Arguments

`refExpr`	a data frame representing immune cell expression profiles. Each row represents an expression of a gene, and each column represents a different immune cell type. colnames contains the name of each immune cell type and the rownames includes the genes' symbol. The names of each immune cell type and the symbol of each gene should be unique. Any gene with missing expression values must be excluded.
`geneExpr`	a data frame representing RNA-seq or microarray gene-expression profiles of a given complex tissue. Each row represents an expression of a gene, and each column represents a different experimental sample. colnames contain the name of each sample and rownames includes the genes' symbol. The name of each individual sample and the symbol of each gene should be unique. Any gene with missing expression values should be excluded.
`marker_set`	data frames of one column, that includes a preselected list of genes that likely discriminate well between the immune-cell types given in the reference data. (DEFAULT: NULL, i.e. one for each gene in the refExpr)
`useOldVersion`	Set the TRUE to 2^ the data (DEFAULT: FALSE)
`progressBar`	Set to TRUE to show a progress bar (DEFAULT: TRUE)

Value

A matrix with cell type estimates for each samples #This toy example library(ADAPTS) fullLM22 <- ADAPTS::LM22[1:30, 1:4] smallLM22 <- fullLM22[1:25,]

cellEst <- estCellPercent.svmdecon(refExpr=smallLM22, geneExpr=fullLM22)

`getLM22cells` *LM22 look up table*

Description

Load a map of cell type names

Usage

```
getLM22cells()
```

Value

a map of cell types names

Examples

```
cellMap <- getLM22cells()
```

`hierarchicalClassify` *Hierarchical Deconvolution*

Description

Deconvolve cell types based on clusters detected by an n-pass spillover matrix

Usage

```
hierarchicalClassify(sigMatrix, geneExpr, toPred, hierarchData = NULL,
  pdfDir = tempdir(), oneCore = FALSE, nPasses = 100,
  remZinf = TRUE, method = "DCQ")
```

Arguments

sigMatrix	The deconvolution matrix, e.g. LM22 or MGSM27
geneExpr	The source gene expression matrix used to calculate sigMatrix
toPred	The gene expression to ultimately deconvolve
hierarchData	The results of hierarchicalSplit OR hierarchicalSplit.sc (DEFAULT: NULL, ie hierarchicalSplit)
pdfDir	A fold to write the pdf file to (DEFAULT: tempdir())
oneCore	Set to TRUE to disable parallelization (DEFAULT: FALSE)
nPasses	The maximum number of iterations for spillToConvergence (DEFAULT: 100)
remZinf	Set to TRUE to remove any ratio with zero or infinity when generating gList (DEFAULT: FALSE)
method	One of 'DCQ', 'SVMDECON', 'DeconRNASeq', 'proportionsInAdmixture', 'nnls' (DEFAULT: DCQ)

Value

a matrix of cell counts

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

cellCounts <- hierarchicalClassify(sigMatrix=smallLM22, geneExpr=fullLM22, toPred=fullLM22,
    oneCore=TRUE, nPasses=10, method='DCQ')
```

---

hierarchicalSplit *Build hierarchical cell clusters.*

Description

Attempt to deconvolve cell types by building a hierarchy of cell types using spillToConvergence to determine cell types that are not signficantly different. First deconvolve those clusters of cell types. Deconvolution matrices are then built to separate the cell types that formerly could not be separated.

Usage

```
hierarchicalSplit(sigMatrix, geneExpr, oneCore = FALSE, nPasses = 100,
  deconMatrices = NULL, remZinf = TRUE, method = "DCQ")
```

Arguments

sigMatrix	The deconvolution matrix, e.g. LM22 or MGSM27
geneExpr	The source gene expression matrix used to calculate sigMatrix
oneCore	Set to TRUE to disable parallelization (DEFAULT: FALSE)
nPasses	The maximum number of iterations for spillToConvergence (DEFAULT: 100)
deconMatrices	Optional pre-computed results from spillToConvergence (DEFAULT: NULL)
remZinf	Set to TRUE to remove any ratio with zero or infinity when generating gList (DEFAULT: FALSE)
method	One of 'DCQ', 'SVMDECON', 'DeconRNASeq', 'proportionsInAdmixture', 'nnls' (DEFAULT: DCQ)

Value

A list of clusters and a list of signature matrices for breaking those clusters

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

clusters <- hierarchicalSplit(sigMatrix=smallLM22, geneExpr=fullLM22, oneCore=TRUE, nPasses=10)
```

Licenses *Licenses required by Celgene legal*

Description

This software is covered by the MIT license. Celgene legal thought it was wise to break the license up into the two license files included in this list.

Usage

```
data("Licenses")
```

Format

A data frame with 0 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Source

https://www.r-project.org/Licenses/MIT

Examples

```
data(Licenses)
str(Licenses)
```

LM22 *Leukocyte 22 data matrix*

Description

Newman et al.'s 2015 22 leukocyte signature matrix.

Usage

```
data("LM22")
```

Format

A data frame with 547 observations on the following 22 variables.

`B.cells.naive` a numeric vector

`B.cells.memory` a numeric vector

`Plasma.cells` a numeric vector

`T.cells.CD8` a numeric vector

`T.cells.CD4.naive` a numeric vector

`T.cells.CD4.memory.resting` a numeric vector

`T.cells.CD4.memory.activated` a numeric vector

`T.cells.follicular.helper` a numeric vector

`T.cells.regulatory..Tregs.` a numeric vector

`T.cells.gamma.delta` a numeric vector

`NK.cells.resting` a numeric vector

`NK.cells.activated` a numeric vector

`Monocytes` a numeric vector

`Macrophages.M0` a numeric vector

`Macrophages.M1` a numeric vector

`Macrophages.M2` a numeric vector

`Dendritic.cells.resting` a numeric vector

`Dendritic.cells.activated` a numeric vector

`Mast.cells.resting` a numeric vector

`Mast.cells.activated` a numeric vector

`Eosinophils` a numeric vector

`Neutrophils` a numeric vector

Source

Newman, A. M. et al. Robust enumeration of cell subsets from tissue expression profiles. Nat. Methods 12, 453–457 (2015). https://media.nature.com/original/nature-assets/nmeth/journal/v12/n5/extref/nmeth.3337-S2.xls

Examples

```
data(LM22)
heatmap(as.matrix(LM22))
```

`loadMGSM27` *Load MGSM27*

Description

Load the MGSM27 signature matrix

Usage

```
loadMGSM27()
```

Value

The MGSM27 signature matrix from Identifying a High-risk Cellular Signature in the Multiple Myeloma Bone Marrow Microenvironment

Examples

```
MGSM27 <- loadMGSM27()
```

`loadModMap` *LM22 to xCell LUT*

Description

Load the LM22 xCell map

Usage

```
loadModMap()
```

Value

A map between xCell cell type names and LM22 cell type names

Examples

```
xcellMap <- loadModMap()
```

MGSM27 *Myeloma Genome Signature Matrix 27*

Description

Newman et al's 2015 plus 5 myeloma specific cell types. Osteoclasts, Adipocytes, Osteoblasts, Multiple Myeloma Plasma Cells, and Plasma Memory Cells

Usage

```
data("MGSM27")
```

Format

A data frame with 601 observations on the following 27 variables.

`B.cells.naive` a numeric vector

`B.cells.memory` a numeric vector

`Plasma.cells` a numeric vector

`T.cells.CD8` a numeric vector

`T.cells.CD4.naive` a numeric vector

`T.cells.CD4.memory.resting` a numeric vector

`T.cells.CD4.memory.activated` a numeric vector

`T.cells.follicular.helper` a numeric vector

`T.cells.regulatory..Tregs.` a numeric vector

`T.cells.gamma.delta` a numeric vector

`NK.cells.resting` a numeric vector

`NK.cells.activated` a numeric vector

`Monocytes` a numeric vector

`Macrophages.M0` a numeric vector

`Macrophages.M1` a numeric vector

`Macrophages.M2` a numeric vector

`Dendritic.cells.resting` a numeric vector

`Dendritic.cells.activated` a numeric vector

`Mast.cells.resting` a numeric vector

`Mast.cells.activated` a numeric vector

`Eosinophils` a numeric vector

`Neutrophils` a numeric vector

`MM.plasma.cell` a numeric vector

`osteoblast` a numeric vector

`osteoclast` a numeric vector

`PlasmaMemory` a numeric vector

`adipocyte` a numeric vector

Details

MGSM27 as constructed for Identifying a High-risk Cellular Signature in the Multiple Myeloma Bone Marrow Microenvironment.

Source

https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-3732/ https://www.ebi.ac.uk/arrayexpress/experiments/E-MEXP-3711/ https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-4152/

Examples

```
data(MGSM27)
heatmap(as.matrix(MGSM27))
```

---

missForest.par *Use parallel missForest to impute missing values.*

Description

This wrapper is helpful because missForest crashes if you have more cores than variables. This will default to no parellelization for Windows

newMatrix <- missForest.par(dataMat)

Usage

```
missForest.par(dataMat, parallelize = "variables")
```

Arguments

dataMat	Columns are features, Rows examples. The data with NA values. 'xmis' in missForest
parallelize	split on 'forests' or 'variables' (DEFAULT: 'variables')

Value

a matrix including imputed values

Examples

```
library(ADAPTS)
LM22 <- ADAPTS::LM22
LM22[2,3] <- as.numeric(NA) #Make some missing data to impute
LM22.imp <- missForest.par(LM22)
```

rankByT *Rank genes for each cell type*

Description

Use a t-test to rank to features for each cell type

gList <- rankByT(geneExpr, qCut=0.3)

Usage

```
rankByT(geneExpr, qCut = 0.3, oneCore = FALSE, secondPval = TRUE,
  remZinf = FALSE, reqRatGT1 = FALSE)
```

Arguments

geneExpr	The gene expression data
qCut	(DEFAULT: 0.3)
oneCore	Set to TRUE to disable paralellization (DEFAULT: FALSE)
secondPval	Set to TRUE to use p-Values as a second sort criteria (DEFAULT: TRUE)
remZinf	Set to TRUE to remove any ratio with zero or infinity. Good for scRNAseq. (DEFAULT: FALSE)
reqRatGT1	Set to TRUE to remove any gene with a ratio with less than 1. Good for scRNAseq. (DEFAULT: FALSE)

Value

a list of cell types with data frames ranking genes

Examples

```
#This toy example treats the LM22 deconvolution matrix as if it were all of the data
#  For a real example, look at the vignette or comments in exprData, fullLM22, small LM22
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:200, 1:8]
#Make a fake signature matrix out of 100 genes and the first 8 cell types
smallLM22 <- fullLM22[1:100, 1:8]

#Make fake data representing two replicates of purified Mast.cells
exprData <- ADAPTS::LM22[1:200, c("Mast.cells.resting","Mast.cells.activated")]
colnames(exprData) <- c("Mast.cells", "Mast.cells")

#Fake source data with replicates for all purified cell types.
#  Note in this fake data set, many cell types have exactly one replicate
fakeAllData <- cbind(fullLM22, as.data.frame(exprData))
gList <- rankByT(geneExpr = fakeAllData, qCut=0.3, oneCore=TRUE, reqRatGT1=FALSE)
```

remakeLM22p *Make an Augmented Signature Matrix*

Description

With the ADAPTSdata packge, it will use the full LM22 data matrix and add a few additional genes to cover osteoblasts, osteoclasts, Plasma.memory, MM. In many ways this is just a convenient wrapper for AugmentSigMatrix that calculates and caches a gList.

Usage

```
remakeLM22p(exprData, fullLM22, smallLM22 = NULL, plotToPDF = TRUE,
  condTol = 1.01, postNorm = TRUE, autoDetectMin = FALSE,
  pdfDir = tempdir(), oneCore = FALSE)
```

Arguments

exprData	The gene express data to use to augment LM22, e.g. ADAPTSdata::addMGSM27
fullLM22	LM22 data with all genes. Available in ADAPTSdata2::fullLM22
smallLM22	The small LM22 matrix, if it includes new cell types in exprData those will not be overwritten (DEFAULT: NULL, i.e. buildLM22plus(useLM22genes = TRUE)
plotToPDF	TRUE: pdf, FALSE: standard display (DEFAULT: TRUE)
condTol	The tolerance in the reconstruction algorithm. 1.0 = no tolerance, 1.05 = 5% tolerance (DEFAULT: 1.01)
postNorm	Set to TRUE to normalize new signatures to match old signatures. To Do: Redo Kappa curve? (DEFAULT: TRUE)
autoDetectMin	Set to true to automatically detect the first local minima. GOOD PRELIMINARY RESULTS (DEAFULT: FALSE)
pdfDir	A fold to write the pdf file to if plotToPDF=TRUE (DEFAULT: tempdir())
oneCore	Set to TRUE to disable parallelization (DEFAULT: FALSE)

Value

a cell type signature matrix

Examples

```
#This toy example treats the LM22 deconvolution matrix as if it were all of the data
#  For a real example, look at the vignette or comments in exprData, fullLM22, small LM22
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:200, 1:8]
#Make a fake signature matrix out of 100 genes and the first 8 cell types
smallLM22 <- fullLM22[1:100, 1:8]
```

```
#Make fake data representing two replicates of purified Mast.cells
exprData <- ADAPTS::LM22[1:200, c("Mast.cells.resting","Mast.cells.activated")]
colnames(exprData) <- c("Mast.cells", "Mast.cells")
newSig <- remakeLM22p(exprData=exprData, fullLM22=fullLM22, smallLM22=smallLM22,
    plotToPDF=FALSE, oneCore=TRUE)
```

`spillToConvergence` *Spillover to convergence*

Description

Build an n-pass spillover matrix, continuing until the results converge into clusters of cell types

deconMatrices <- spillToConvergence(sigMatrix, geneExpr, 100, FALSE, TRUE)

Usage

```
spillToConvergence(sigMatrix, geneExpr, nPasses = 100, plotIt = FALSE,
  imputNAs = FALSE, method = "DCQ")
```

Arguments

`sigMatrix`	The deconvolution matrix, e.g. LM22 or MGSM27
`geneExpr`	The source gene expression matrix used to calculate sigMatrix
`nPasses`	The maximum number of iterations (DEFAULT: 100)
`plotIt`	Set to TRUE to plot it (DEFAULT: FALSE)
`imputNAs`	Set to TRUE to imput genes with missing values & cache the imputed. FALSE will just remove them (DEFAULT: FALSE)
`method`	One of 'DCQ', 'SVMDECON', 'DeconRNASeq', 'proportionsInAdmixture', 'nnls' (DEFAULT: DCQ)

Value

A list of signature matrices

Examples

```
#This toy example
library(ADAPTS)
fullLM22 <- ADAPTS::LM22[1:30, 1:4]
smallLM22 <- fullLM22[1:25,]

deconMatrices <- spillToConvergence(sigMatrix=smallLM22, geneExpr=fullLM22, nPasses=10, plotIt=TRUE)
```

SVMDECON *Support vector machine deconvolution*

Description

Use SVMDECONV to estimate the cell count percentage David L Gibbs, dgibbs@systemsbiology.org June 9, 2017

v-SVR is applied with a linear kernel to solve for f, and the best result from three values of v = 0.25, 0.5, 0.75 is saved, where 'best' is defined as the lowest root mean squared error between m and the deconvolution result, f x B.

Our current implementation executes v-SVR using the 'svm' function in the R package, 'e1071'.

w2 <- SVMDECON(m, B)

Usage

```
SVMDECON(m, B)
```

Arguments

m	a matrix represenging the mixture (genes X 1 sample)
B	a matrix representing the references (genes X cells), m should be subset to match B

Value

A matrix with cell type estimates for each samples

---

weightNorm *SVMDECONV helper function*

Description

Use weightNorm to normalize the SVM weights. Used for SVMDECONV

w1 <- weightNorm(w)

Usage

```
weightNorm(w)
```

Arguments

w	The weight vector from fitting an SVM, something like something like t(fit1$coefs) %*% fit1$SV, where fit comes from <- svm(m~B, nu=0.25, kernel="linear"))

Value

a weight vector

Index