Package ‘decone’

October 13, 2022

Type Package

Date 2022-09-18

Title Cell Type Deconvolution Evaluator

Version 1.0.0

Description The evaluation platform for cell type deconvolution. This platform provides multiple simulation and real mixture expression datasets for evaluate the performance of deconvlution tools. This toolkit also plots preety figure for the comparison of different deconvolution methods.

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Depends R (>= 4.1.0)

Imports Metrics,
stats,
utils,
Matrix,
tibble,
data.table,
tools,
magrittr,
ggplot2,
pheatmap,
reshape2,
grDevices,
forcats,
stringr,
R.utils,
ggpubr,
progress

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

addNoise 2
addNoiseExpr 3
boxplot_NcrossCompare 4
boxplot_NGrad 4
boxplot_simple 5
cheatmap_NcrossCompare 6
cheatmap_RcrossCompare 7
CPM 8
exprSim 8
heatmap_NcrossCompare 9
heatmap_NGradCT 10
heatmap_RcrossCompare 11
pseudoData 12
pseudoExpr 12
rareExprSim 13
rarescExprSim 14
regMetrics 15
scatter_R 16
scatter_simple 16
scExprSim 17
TPM 18
unExprSim 19
unscExprSim 20
v_norm 21

Index **22**

`addNoise` *Add noise to gene expression data*

Description

Add noise based on negtive bionomial distribution. Please see "A benchmark for RNA-seq deconvolution analysis under dynamic testing environments" in Genome Biology.

Usage

```
addNoise(x = NULL, pt = 0.1, type = "NB")
```

Arguments

`x`	a gene expression numeric vector.
`pt`	parameter to control noised level. Default: 0.1
`type`	"NB", "N" or "LN". "NB" means Negative binomial model, "N" means normal model, "LN" means Log-normal model. Default: "NB"

Value

the sample length vector.

Examples

```
res <- addNoise(x = seq(100))
```

---

addNoiseExpr *Add noise to the simulated expression matrix*

Description

Add different level noise to the simulated expression matrix based on the negative binomial distribution. The related model can be found in this paper (Jin, H., Liu, Z. A benchmark for RNA-seq deconvolution analysis under dynamic testing environments. Genome Biol 22, 102 (2021). https://doi.org/10.1186/s13059-021-02290-6).

Usage

```
addNoiseExpr(
  exprFile,
  outputPath = NULL,
  prefix = NULL,
  Pt = seq(0.1, 1, 0.1),
  type = "NB"
)
```

Arguments

`exprFile`	The input expression file. Must be in csv format. Each row is a gene, each column is a sample. rownames and colnames are required. Please check the output of the function `exprSim`.
`outputPath`	Output path, create if not exists. Default: a new folder based on the exprFile. For example, if the exprFile is "/data/test.txt", the outputPath will be a new folder "/data/test".
`prefix`	The prefix of the output file. Note: the suffix will be added automatically based on the noise level.
`Pt`	Parameter to control noise level. Default: seq(0.1, 1, 0.1)
`type`	"NB", "N" or "LN". Noise type. "NB" means Negative binomial model, "N" means normal model, "LN" means Log-normal model. Default: "NB"

Value

All the information will be written in the output path, and each file is the generated data in a specific noise level.

Examples

```
res <- pseudoExpr()
write.csv(x = res$mix, file = "mix.csv", row.names = T, quote = F)
addNoiseExpr(exprFile = "mix.csv", Pt = seq(0.1, 1, 0.1), type = "NB")
```

`boxplot_NcrossCompare` *Comparing boxplot for noise testing.*

Description

Boxplot for illustrating the deconvlition performance with the noised in silico data. this function is used for comparing multiple methods.

Usage

```
boxplot_NcrossCompare(actual, predicted, label = NULL, method, title = "")
```

Arguments

`actual`	The groundtruth proportion of cell types in matrix. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`predicted`	a list of vectors. Each vector contains all the files with the predicted proportions in different noise level. Must be in csv file with row and column names. row: cell types, column: samples. For example, list(method1 = c("m1_noise0.csv", "m1_noise1.csv", "m1_noise2.csv"), method2 = c("m2_noise0.csv", "m2_noise1.csv", "m2_noise2.csv")). Note: names for each method will be used for assign colors.
`label`	a vector contains the label corresponding to the predicted proportion file. Default: c("condition1", "condition2", "condition3", ...)
`method`	"rmse", "mape", "mae", "pearson" or "spearman". Note: for mape, cell types with read proportion 0 will be ignored.
`title`	Boxplot title in character.

Value

The computed metrics and plot data.

Examples

```
res <- pseudoData(type = 3, outputPath = "test_file")
res <- boxplot_NcrossCompare(actual = res$actual, predicted = res$predicted,
label = res$noise_level, method = "pearson", title = "pearson")
```

`boxplot_NGrad` *Boxplot of deconvolution results for multiple test.*

Description

Boxplot for illustrating the deconvolution performance with the noised in silico data.

Usage

```
boxplot_NGrad(actual, predicted, label = NULL, method, title = "Boxplot")
```

Arguments

`actual`	The groundtruth proportion of cell types in matrix. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`predicted`	a vector contains all the files with the predicted proportions in different noise level. Must be in csv file with row and column names. row: cell types, column: samples. For example, c("noise0.csv", "noise1.csv", "noise2.csv").
`label`	a vector contains the label corresponding to the predicted proportion file. Default: base file name in parameter 'predicted'. For example, c("noise0", "noise1", "noise2").
`method`	"rmse", "mape", "mae", "pearson" or "spearman". Note: for mape, cell types with read proportion 0 will be ignored.
`title`	Boxplot title in character.

Value

The computed metrics and plot data.

Examples

```
res <- pseudoData(type = 2)
res <-boxplot_NGrad(actual = res$actual, predicted = res$predicted,
label = res$noise_level, method = "rmse")
```

`boxplot_simple` *Boxplot of deconvolution results for a specific method.*

Description

Boxplot for deconvolution results with multiple samples.

Usage

```
boxplot_simple(actual, predicted, method = NULL)
```

Arguments

`actual`	The groundtruth proportion of cell types in matrix. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`predicted`	The predicted proportion of cell types in matrix. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`method`	"rmse", "mape", "mae", "pearson" or "spearman". Note: for mape, cell types with read proportion 0 will be ignored.

Value

The computed metrics and plot data.

Examples

```
res <- pseudoData(type = 1)
res <- boxplot_simple(actual = res$actual, predicted = res$predicted, method = "mape")
```

`cheatmap_NcrossCompare` *Plot circle heatmap for deconvolution results.*

Description

Usually, only one metric cannot reveal the deconvolution performance well. Therefore, the circle heatmap which contains two different dimension information can be used for a better illustration.

Usage

```
cheatmap_NcrossCompare(actual, predicted, label = NULL, method1, method2)
```

Arguments

`actual`	The groundtruth proportion of cell types in matrix. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`predicted`	a list of vectors. Each vector contains all the files with the predicted proportions in different noise level. Must be in csv file with row and column names. row: cell types, column: samples. For example, list(method1 = c("m1_noise0.csv", "m1_noise1.csv", "m1_noise2.csv"), method2 = c("m2_noise0.csv", "m2_noise1.csv", "m2_noise2.csv")). Note: names for each method will be used for assign colors.
`label`	a vector contains the label corresponding to the predicted proportion file. Default: c("condition1", "condition2", "condition3", ...)
`method1`	"rmse", "mape", "mae", "pearson" or "spearman". Note: for mape, cell types with read proportion 0 will be ignored.
`method2`	"rmse", "mape", "mae", "pearson" or "spearman". Note: for mape, cell types with read proportion 0 will be ignored.

Value

The computed metrics and plot data.

Examples

```
res <- pseudoData(type = 3, outputPath = "test_file")
res <- cheatmap_NcrossCompare(actual = res$actual, predicted = res$predicted,
label = res$noise_level,  method1 = "pearson", method2 = "rmse")
```

cheatmap_RcrossCompare

Plot circle heatmap for deconvolution results rare cell types.

Description

Generate circle heatmap for different rare cell type proportions. The dataset should be generated from the function rareExprSim.

Usage

```
cheatmap_RcrossCompare(
  actual,
  predicted,
  p_rare = c(0.001, 0.003, 0.005, 0.008, 0.01, 0.03, 0.05),
  label = NULL,
  method1,
  method2
)
```

Arguments

actual	The groundtruth proportion of cell types in matrix or csv file. row: cell types, column: samples. Also can be a csv matrix with row and column names.
predicted	a vector contains all the files with the predicted proportions in different method. Must be in csv file with row and column names. row: cell types, column: samples. For example, c("method1.csv", "method2.csv", "method3.csv").
p_rare	A vector of proportions. should be the same with function rareExprSim. Default: c(0.001, 0.003, 0.005, 0.008, 0.01, 0.03, 0.05)
label	a vector contains the label corresponding to the predicted proportion file. Default: c("condition1", "condition2", "condition3", ...)
method1	"rmse", "mape", "mae". Note: for mape, cell types with read proportion 0 will be ignored.
method2	"rmse", "mape", "mae". Note: for mape, cell types with read proportion 0 will be ignored.

Value

The plot data.

Examples

```
rareExprSim()
# after the simulation, just pass the predicted file
# to the parameter "actual" and "predicted"
```

CPM *Convert a gene expression count matrix into CPM matrix.*

Description

Convert a gene expression count matrix into CPM matrix

Usage

```
CPM(data = NULL)
```

Arguments

data	A dataframe or matrix object. Note: Must contain a column named "Length", "Length" means gene length.

Value

A dataframe or matrix object.

Examples

```
data <- data.frame(s1 = seq(100), Length = seq(100))
res <- CPM(data)
```

exprSim *Generate in silico mixture expression matrix*

Description

Generate in silico mixture expression matrix based on the internal RNA-seq database. The internal RNA-seq database is collected from multiple studies. All the samples are passed the quality filter. This function is different from pseudoExpr. Counts in pseudoExpr is randomly generated from uniform distribution. This function use the real cell type specific expression data to generate mixture data. We provide two types of simulation called "coarse" and "fine". This idea is from DREAM Challenge Tumor Deconvolution problem. https://www.synapse.org/#!Synapse:syn15589870/wiki/.

Usage

```
exprSim(
  n_sample = 50,
  p = 2/3,
  type = "coarse",
  transform = "TPM",
  outputPath = NULL,
  mix_name = "coarse_gene_expr.csv",
  ref_name = "coarse_ref.csv",
  prop_name = "coarse_prop.csv",
  refVar_name = NULL,
  train_name = NULL
)
```

Arguments

`n_sample`	Sample number to be generated, default: 50.
`p`	Proportion of sample in train set, default: 0.6.
`type`	"coarse" or "fine". "coarse" means the simulation will be performed in a coarse level. Only 8 cell types will be used, including ("B.cells", "CD4.T.cells", "CD8.T.cells", "endothelial.cells", "macrophages", "monocytes","neutrophils", "NK.cells"). "fine" means the simulation will be performed in a fine level. 14 cell types will be used, including ("memory.B.cells", "naive.B.cells", "memory.CD4.T.cells", "naive.CD4.T.cells", "regulatory.T.cells", "memory.CD8.T.cells", "naive.CD8.T.cells", "NK.cells", "neutrophils", "monocytes", "myeloid.dendritic.cells", "macrophages", "fibroblasts", "endothelial.cells")
`transform`	"TPM", "CPM" or "NO". Transform the data into TPM, CPM or in counts.
`outputPath`	output file save path.
`mix_name`	mixture output file name.
`ref_name`	reference output file name in csv.
`prop_name`	simulated proportion file name in csv.
`refVar_name`	reference variance file name in csv.
`train_name`	file name for all data in train set in csv. This data can be used for differential gene analysis.

Value

All the information will be written in the output path.

Examples

```
exprSim()
```

`heatmap_NcrossCompare` *Plot heatmap for deconvolution results.*

Description

Generate heatmap comparison for different methods.

Usage

```
heatmap_NcrossCompare(actual, predicted, label = NULL, method)
```

Arguments

`actual`	The groundtruth proportion of cell types in matrix. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`predicted`	a list of vectors. Each vector contains all the files with the predicted proportions in different noise level. Must be in csv file with row and column names. row: cell types, column: samples. For example, list(method1 = c("m1_noise0.csv", "m1_noise1.csv", "m1_noise2.csv"), method2 = c("m2_noise0.csv", "m2_noise1.csv", "m2_noise2.csv")). Note: names for each method will be used for assign colors.
`label`	a vector contains the label corresponding to the predicted proportion file. Default: c("condition1", "condition2", "condition3", ...)
`method`	"rmse", "mape", "mae", "pearson" or "spearman". Note: for mape, cell types with read proportion 0 will be ignored.

Value

The computed metrics and plot data.

Examples

```
res <- pseudoData(type = 3, outputPath = "test_file")
res <- heatmap_NcrossCompare(actual = res$actual, predicted = res$predicted,
label = res$noise_level, method = "mape")
```

`heatmap_NGradCT` *Plot heatmap for different cell types.*

Description

Generate heatmap comparison for different cell types.

Usage

```
heatmap_NGradCT(actual, predicted, label = NULL, method)
```

Arguments

`actual`	The groundtruth proportion of cell types in matrix. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`predicted`	a vector contains all the files with the predicted proportions in different noise level. Must be in csv file with row and column names. row: cell types, column: samples. For example, c("noise0.csv", "noise1.csv", "noise2.csv").
`label`	a vector contains the label corresponding to the predicted proportion file. Default: c("condition1", "condition2", "condition3", ...)
`method`	"rmse", "mape", "mae", "pearson" or "spearman". Note: for mape, cell types with read proportion 0 will be ignored.

Value

The computed metrics and plot data.

Examples

```
res <- pseudoData(type = 2)
res <- heatmap_NGradCT(actual = res$actual, predicted = res$predicted,
label = res$noise_level, method = "rmse")
```

`heatmap_RcrossCompare` *Plot heatmap for deconvolution results.*

Description

Generate heatmap comparison for different methods.

Usage

```
heatmap_RcrossCompare(
  actual,
  predicted,
  p_rare = c(0.001, 0.003, 0.005, 0.008, 0.01, 0.03, 0.05),
  label = NULL,
  method
)
```

Arguments

`actual`	The groundtruth proportion of cell types in matrix. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`predicted`	a vector contains all the files with the predicted proportions in different method. Must be in csv file with row and column names. row: cell types, column: samples. For example, c("method1.csv", "method2.csv", "method3.csv").
`p_rare`	A vector of proportions. should be the same with function `rareExprSim`. Default: c(0.001, 0.003, 0.005, 0.008, 0.01, 0.03, 0.05)
`label`	a vector contains the label corresponding to the predicted proportion file. Default: c("condition1", "condition2", "condition3", ...)
`method`	"rmse", "mape", "mae". Note: for mape, cell types with read proportion 0 will be ignored.

Value

The computed metrics and plot data.

Examples

```
rareExprSim()
# after the simulation, just pass the predicted file
# to the parameter "actual" and "predicted"
```

`pseudoData` *Generate pseudo data for plotting*

Description

Generate pseudo data (prediction values) for plotting

Usage

```
pseudoData(type = 1, outputPath = "test_file")
```

Arguments

`type`	1, 2 or 3 for different usage
`outputPath`	path to save tmp file

Value

The pseudo data

Examples

```
res <- pseudoData(type = 1)
```

`pseudoExpr` *Generate pseudo mixture expression matrix*

Description

Generate pseudo mixture expression matrix based on uniform distribution, without any noise.

Usage

```
pseudoExpr(n_sample = 50, n_gene = 1000, n_ct = 10)
```

Arguments

`n_sample`	Sample number to be generated, default: 50
`n_gene`	Gene number to be generated, default: 1000
`n_ct`	Cell type number to be generated, default: 10

Value

a list, prop means the pseudo proportion, ref means the pseudo external reference, mix is the output of ref

Examples

```
res <- pseudoExpr()
```

`rareExprSim` *Generate in silico expression dataset with rare component.*

Description

Generate in silico mixture expression matrix based on the internal RNA-seq database. This function is different from the function `exprSim`. `exprSim` generates all the proportion randomly, while this function takes one cell type as rare component and all Other cell type proportion will be set randomly from the uniform distribution. The internal RNA-seq database is collected from multiple studies. All the samples are passed the quality filter. We provide two types of simulation called "coarse" and "fine". This idea is from DREAM Challenge Tumor Deconvolution problem. https://www.synapse.org/#!Synapse:syn15589870/wiki/.

Usage

```
rareExprSim(
  p_rare = c(0.001, 0.003, 0.005, 0.008, 0.01, 0.03, 0.05),
  p = 0.6,
  type = "coarse",
  transform = "TPM",
  outputPath = NULL,
  mix_name = "coarse_gene_expr.csv",
  ref_name = "coarse_ref.csv",
  prop_name = "coarse_prop.csv",
  refVar_name = NULL,
  train_name = NULL
)
```

Arguments

`p_rare`	A vector of proportions. Default: c(0.001, 0.003, 0.005, 0.008, 0.01, 0.03, 0.05) Note: Every cell type will be treated as rare component. For example, if 8 cell types need to be tested, this function will generate 7 * 8 = 56 samples. 7 mean 7 rare proportions and 8 means 8 cell types.
`p`	Proportion of sample in train set, default: 0.6.
`type`	"coarse" or "fine". "coarse" means the simulation will be performed in a coarse level. Only 8 cell types will be used, including ("B.cells", "CD4.T.cells", "CD8.T.cells", "endothelial.cells", "macrophages", "monocytes","neutrophils", "NK.cells"). "fine" means the simulation will be performed in a fine level. 14 cell types will be used, including ("memory.B.cells", "naive.B.cells", "memory.CD4.T.cells", "naive.CD4.T.cells", "regulatory.T.cells", "memory.CD8.T.cells", "naive.CD8.T.cells", "NK.cells", "neutrophils", "monocytes", "myeloid.dendritic.cells", "macrophages", "fibroblasts", "endothelial.cells")
`transform`	"TPM", "CPM" or "NO". Transform the data into TPM, CPM or in counts.
`outputPath`	output file save path.
`mix_name`	mixture output file name.
`ref_name`	reference output file name in csv.
`prop_name`	simulated proportion file name in csv.
`refVar_name`	reference variance file name in csv.

`train_name`	file name for all data in train set in csv. This data can be used for differential gene analysis.

Value

All the information will be written in the output path.

Examples

```
rareExprSim()
```

---

rarescExprSim — *Generate in silico mixture expression matrix based on scRNA-seq with rare component.*

Description

Generate in silico mixture expression matrix based on the internal scRNA-seq database. The internal RNA-seq database is collected from PMID:29474909. All the samples are passed the quality filter. This function use the real cell type specific expression data to generate mixture data. The cell type include c("FetalStomach", "FetalLung", "FetalLiver", "FetalKidney", "FetalIntestine", "FetalBrain", "Female.fetal.Gonad")

Usage

```
rarescExprSim(
  p_rare = c(0.001, 0.003, 0.005, 0.008, 0.01, 0.03, 0.05),
  p = 2/3,
  transform = "TPM",
  outputPath = NULL,
  mix_name = "scMouse_gene_expr.csv",
  ref_name = "scMouse_ref.csv",
  prop_name = "scMouse_prop.csv",
  train_name = "scMouse_ref_rawCount.csv"
)
```

Arguments

`p_rare`	A vector of proportions. Default: c(0.001, 0.003, 0.005, 0.008, 0.01, 0.03, 0.05) Note: Every cell type will be treated as rare component. For example, if 8 cell types need to be tested, this function will generate 7 * 8 = 56 samples. 7 mean 7 rare proportions and 8 means 8 cell types.
`p`	Proportion of sample in train set, default: 2 / 3.
`transform`	"TPM", "CPM" or "NO". Transform the data into TPM, CPM or in counts.
`outputPath`	output file save path.
`mix_name`	mixture output file name.
`ref_name`	reference output file name in csv.
`prop_name`	simulated proportion file name in csv.
`train_name`	file name for all data in train set in csv. This data can be used for differential gene analysis.

Value

All the information will be written in the output path.

Examples

```
rarescExprSim()
```

`regMetrics` *Compute regression Metrics for matrix results.*

Description

Compute several regression Metrics for cell type deconcolution.

Usage

```
regMetrics(actual, predicted, method = NULL)
```

Arguments

`actual`	The groundcloth proportion of cell types in matrix. row: cell types, column: samples.
`predicted`	The predicted proportion of cell types in matrix. row: cell types, column: samples.
`method`	"rmse", "mape", "mae", "pearson", "spearman", "pearson2", "spearman2". Note: for mape, cell types with read proportion 0 will be ignored. For pearson2 and spearman2, all data in matrix will be taken into consideration, which mean only one number will be reported.

Value

a vector with the value for each sample.

Examples

```
res <- regMetrics(actual = matrix(data = seq(9), nrow = 3),
                  predicted = matrix(data = seq(9), nrow = 3),
                  method = "rmse")
```

scatter_R *Scatter plot for rare component.*

Description

Generate scatter plot for rare component.

Usage

```
scatter_R(
  actual,
  predicted,
  p_rare = c(0.001, 0.003, 0.005, 0.008, 0.01, 0.03, 0.05),
  celltype = TRUE
)
```

Arguments

`actual`	The groundtruth proportion of cell types in matrix or csv file. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`predicted`	The predicted proportion of cell types in matrix or csv file. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`p_rare`	A vector of proportions. should be the same with function `rareExprSim`. Default: c(0.001, 0.003, 0.005, 0.008, 0.01, 0.03, 0.05)
`celltype`	TRUE or FALSE. Assign different type and color for different cell types. Default: TRUE.

Value

The plot data.

Examples

```
rareExprSim()
# after the simulation, just pass the predicted file
# to the parameter "actual" and "predicted"
```

scatter_simple *Scatter plot for proportion prediction.*

Description

Generate scatter plot comparison for different cell types.

Usage

```
scatter_simple(actual, predicted, method, celltype = TRUE)
```

Arguments

`actual`	The groundtruth proportion of cell types in matrix or csv file. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`predicted`	The predicted proportion of cell types in matrix or csv file. row: cell types, column: samples. Also can be a csv matrix with row and column names.
`method`	"pearson" (default), "kendall", or "spearman".
`celltype`	TRUE or FALSE. Assign different type and color for different cell types.

Value

The plot data.

Examples

```
res <- pseudoData(type = 1)
res <- scatter_simple(actual = res$actual, predicted = res$predicted, method = "pearson")
```

`scExprSim` *Generate in silico mixture expression matrix based on scRNA-seq*

Description

Generate in silico mixture expression matrix based on the internal scRNA-seq database. The internal RNA-seq database is collected from PMID:29474909. All the samples are passed the quality filter. This function use the real cell type specific expression data to generate mixture data. The cell type include c("FetalStomach", "FetalLung", "FetalLiver", "FetalKidney", "FetalIntestine", "FetalBrain", "Female.fetal.Gonad")

Usage

```
scExprSim(
  n_sample = 50,
  p = 2/3,
  transform = "TPM",
  outputPath = NULL,
  mix_name = "scMouse_gene_expr.csv",
  ref_name = "scMouse_ref.csv",
  prop_name = "scMouse_prop.csv",
  train_name = "scMouse_ref_rawCount.csv"
)
```

Arguments

`n_sample`	Sample number to be generated, default: 50.
`p`	Proportion of sample in train set, default: 2 / 3.
`transform`	"TPM", "CPM" or "NO". Transform the data into TPM, CPM or in counts.
`outputPath`	output file save path.
`mix_name`	mixture output file name.

`ref_name`	reference output file name in csv.
`prop_name`	simulated proportion file name in csv.
`train_name`	file name for all data in train set in csv. This data can be used for differential gene analysis.

Value

All the information will be written in the output path.

Examples

```
res <- scExprSim()
```

TPM *Convert a gene expression count matrix into TPM matrix.*

Description

Convert a gene expression count matrix into TPM matrix

Usage

```
TPM(data = NULL)
```

Arguments

`data`	A dataframe or matrix object. Note: Must contain a column named "Length", "Length" means gene length.

Value

A dataframe or matrix object.

Examples

```
data <- data.frame(s1 = seq(100), Length = seq(100))
res <- TPM(data)
```

unExprSim — *Generate in silico expression dataset with unknown component.*

Description

Generate in silico mixture expression matrix based on the internal RNA-seq database. This function is different from the function `exprSim`. `exprSim` generates all the proportion randomly, while this function takes one cell type as unknown component. The internal RNA-seq database is collected from multiple studies. All the samples are passed the quality filter. We provide two types of simulation called "coarse" and "fine". This idea is from DREAM Challenge Tumor Deconvolution problem. `https://www.synapse.org/#!Synapse:syn15589870/wiki/`.

Usage

```
unExprSim(
  unknown = NULL,
  n_sample = 50,
  p = 0.6,
  type = "coarse",
  transform = "TPM",
  outputPath = NULL,
  mix_name = "coarse_gene_expr.csv",
  ref_name = "coarse_ref.csv",
  prop_name = "coarse_prop.csv",
  ref_total_name = NULL,
  prop_total_name = NULL,
  refVar_name = NULL,
  train_name = NULL
)
```

Arguments

`unknown`	Character to select one cell type as unknown component. Default: NULL, mean randomly select one cell type to drop.
`n_sample`	Sample number to be generated, default: 50.
`p`	Proportion of sample in train set, default: 0.6.
`type`	"coarse" or "fine". "coarse" means the simulation will be performed in a coarse level. Only 8 cell types will be used, including ("B.cells", "CD4.T.cells", "CD8.T.cells", "endothelial.cells", "macrophages", "monocytes","neutrophils", "NK.cells"). "fine" means the simulation will be performed in a fine level. 14 cell types will be used, including ("memory.B.cells", "naive.B.cells", "memory.CD4.T.cells", "naive.CD4.T.cells", "regulatory.T.cells", "memory.CD8.T.cells", "naive.CD8.T.cells", "NK.cells", "neutrophils", "monocytes", "myeloid.dendritic.cells", "macrophages", "fibroblasts", "endothelial.cells")
`transform`	"TPM", "CPM" or "NO". Transform the data into TPM, CPM or in counts.
`outputPath`	output file save path.
`mix_name`	mixture output file name.
`ref_name`	reference output file name in csv.
`prop_name`	simulated proportion file name in csv, unknown component will be removed.

`ref_total_name`	simulated reference file name in csv, unknown component is contained.
`prop_total_name`	simulated proportion file name in csv, unknown component is contained.
`refVar_name`	reference variance file name in csv.
`train_name`	file name for all data in train set in csv. This data can be used for differential gene analysis.

Value

All the information will be written in the output path.

Examples

```
res <- unExprSim()
```

`unscExprSim` *Generate in silico mixture expression matrix based on scRNA-seq*

Description

Generate in silico mixture expression matrix based on the internal scRNA-seq database. The internal RNA-seq database is collected from PMID:29474909. All the samples are passed the quality filter. This function use the real cell type specific expression data to generate mixture data. The cell type include c("FetalStomach", "FetalLung", "FetalLiver", "FetalKidney", "FetalIntestine", "FetalBrain", "Female.fetal.Gonad")

Usage

```
unscExprSim(
  unknown = NULL,
  n_sample = 50,
  p = 2/3,
  transform = "TPM",
  outputPath = NULL,
  mix_name = "scMouse_gene_expr.csv",
  ref_name = "scMouse_ref.csv",
  prop_name = "scMouse_prop.csv",
  train_name = "scMouse_ref_rawCount.csv",
  ref_total_name = NULL,
  prop_total_name = NULL
)
```

Arguments

`unknown`	Character to select one cell type as unknown component. Default: NULL, mean randomly select one cell type to drop.
`n_sample`	Sample number to be generated, default: 50.
`p`	Proportion of sample in train set, default: 2 / 3.
`transform`	"TPM", "CPM" or "NO". Transform the data into TPM, CPM or in counts.

`outputPath`	output file save path.
`mix_name`	mixture output file name.
`ref_name`	reference output file name in csv.
`prop_name`	simulated proportion file name in csv.
`train_name`	file name for all data in train set in csv. This data can be used for differential gene analysis.
`ref_total_name`	simulated reference file name in csv, unknown component is contained.
`prop_total_name`	simulated proportion file name in csv, unknown component is contained.

Value

All the information will be written in the output path.

Examples

```
res <- unscExprSim()
```

`v_norm` *Normalize a vector*

Description

Normalize a vector

Usage

```
v_norm(x = NULL)
```

Arguments

`x`	a numeric vector

Value

Normalized numeric vector

Examples

```
res <- v_norm(seq(100))
```

Index

addNoise, 2
addNoiseExpr, 3

boxplot_NcrossCompare, 4
boxplot_NGrad, 4
boxplot_simple, 5

cheatmap_NcrossCompare, 6
cheatmap_RcrossCompare, 7
CPM, 8

exprSim, *3*, 8, *13*, *19*

heatmap_NcrossCompare, 9
heatmap_NGradCT, 10
heatmap_RcrossCompare, 11

pseudoData, 12
pseudoExpr, *8*, 12

rareExprSim, *7*, *11*, 13, *16*
rarescExprSim, 14
regMetrics, 15

scatter_R, 16
scatter_simple, 16
scExprSim, 17

TPM, 18

unExprSim, 19
unscExprSim, 20

v_norm, 21