

# Package ‘singscore’

June 25, 2019

**Type** Package

**Title** Rank-based single-sample gene set scoring method

**Version** 1.5.0

**Description** A simple single-sample gene signature scoring method that uses rank-based statistics to analyze the sample's gene expression profile. It scores the expression activities of gene sets at a single-sample level.

**biocViews** Software, GeneExpression, GeneSetEnrichment

**Depends** R (>= 3.5),GSEABase

**Imports** methods, stats, graphics, ggplot2, grDevices, ggrepel, plotly, tidy, plyr, magrittr, reshape, edgeR, RColorBrewer, Biobase, BiocParallel, SummarizedExperiment, matrixStats, reshape2, S4Vectors

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**Roxygen** list(markdown = TRUE)

**Collate** 'singscore.R' 'rankAndScoring.R' 'permuTest.R'  
'generatNullGeneric.R' 'multiScoreGeneric.R' 'plot.R'  
'plotRankDensityGeneric.R' 'rankGenesGeneric.R'  
'simpleScoreGeneric.R'

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**URL** <https://davislaboratory.github.io/singscore>

**BugReports** <https://github.com/DavisLaboratory/singscore/issues>

**git\_url** <https://git.bioconductor.org/packages/singscore>

**git\_branch** master

**git\_last\_commit** 90479f2

**git\_last\_commit\_date** 2019-05-02

**Date/Publication** 2019-06-24

**Author** Ruqian Lyu [aut, ctb],

Momeneh Foroutan [aut, ctb] (<<https://orcid.org/0000-0002-1440-0457>>),

Dharmesh D. Bhuvan [aut, cre] (<<https://orcid.org/0000-0002-6398-9157>>)

**Maintainer** Dharmesh D. Bhuvan <[bhuvan.d@wehi.edu.au](mailto:bhuvan.d@wehi.edu.au)>

## R topics documented:

generateNull . . . . .	2
getPvals . . . . .	4
multiScore . . . . .	5
plotDispersion . . . . .	6
plotNull . . . . .	7
plotRankDensity . . . . .	8
plotScoreLandscape . . . . .	9
projectScoreLandscape . . . . .	10
rankGenes . . . . .	11
scoredf_ccle_epi . . . . .	12
scoredf_ccle_mes . . . . .	13
scoredf_tcga_epi . . . . .	13
scoredf_tcga_mes . . . . .	14
simpleScore . . . . .	15
singscore . . . . .	16
tgfb_expr_10_se . . . . .	16
tgfb_gs_dn . . . . .	17
tgfb_gs_up . . . . .	18
toy_expr_se . . . . .	18
toy_gs_dn . . . . .	19
toy_gs_up . . . . .	19

<b>Index</b>	<b>20</b>
--------------	-----------

---

generateNull	<i>Permutation test for the derived scores of each sample</i>
--------------	---

---

### Description

This function generates a number of random gene sets that have the same number of genes as the scored gene set. It scores each random gene set and returns a matrix of scores for all samples. The empirical scores are used to calculate the empirical p-values and plot the null distribution. The implementation uses `BiocParallel::bplapply()` for easy access to parallel backends. Note that one should pass the same values to the `upSet`, `downSet`, `centerScore` and `bidirectional` arguments as what they provide for the `simpleScore()` function to generate a proper null distribution.

### Usage

```
generateNull(upSet, downSet = NULL, rankData, centerScore = TRUE,
  knownDirection = TRUE, B = 1000, ncores = 1,
  seed = sample.int(1e+06, 1), useBPPARAM = NULL)

## S4 method for signature 'vector,missing'
generateNull(upSet, downSet = NULL, rankData,
  centerScore = TRUE, knownDirection = TRUE, B = 1000, ncores = 1,
  seed = sample.int(1e+06, 1), useBPPARAM = NULL)

## S4 method for signature 'GeneSet,missing'
generateNull(upSet, downSet = NULL, rankData,
  centerScore = TRUE, knownDirection = TRUE, B = 1000, ncores = 1,
```

```

seed = sample.int(1e+06, 1), useBPPARAM = NULL)

## S4 method for signature 'vector,vector'
generateNull(upSet, downSet = NULL, rankData,
  centerScore = TRUE, knownDirection = TRUE, B = 1000, ncores = 1,
  seed = sample.int(1e+06, 1), useBPPARAM = NULL)

## S4 method for signature 'GeneSet,GeneSet'
generateNull(upSet, downSet = NULL, rankData,
  centerScore = TRUE, knownDirection = TRUE, B = 1000, ncores = 1,
  seed = sample.int(1e+06, 1), useBPPARAM = NULL)

```

### Arguments

upSet	GeneSet object or a vector of gene Ids, up-regulated gene set
downSet	GeneSet object or a vector of gene Ids, down-regulated gene
rankData	matrix, outcome of function <a href="#">rankGenes()</a>
centerScore	A Boolean, specifying whether scores should be centered around 0, default as TRUE
knownDirection	A boolean flag, it determines whether the scoring method should derive the scores in a directional manner when the gene signature only contains one set of gene set (passing the gene set via upSet). It is default as TRUE but one can set the argument to be FALSE to derive the score for a single gene set in a unidirectional way. This parameter becomes irrelevant when both upSet and downSet are provided.
B	integer, the number of permutation repeats or the number of random gene sets to be generated, default as 1000
ncores,	integer, the number of CPU cores the function can use
seed	integer, set the seed for randomisation
useBPPARAM,	the backend the function uses, if NULL is provided, the function uses the default parallel backend which is the first on the list returned by <code>BiocParallel::registered()</code> i.e <code>BiocParallel::registered()[[1]]</code> for your machine. It can be changed explicitly by passing a BPPARAM

### Value

A matrix of empirical scores for all samples

### Author(s)

Ruqian Lyu

### See Also

[Post about BiocParallel](#) `browseVignettes("BiocParallel")`

### Examples

```

ranked <- rankGenes(toy_expr_se)
scoredf <- simpleScore(ranked, upSet = toy_gs_up, downSet = toy_gs_dn)

```

```
# find out what backends can be registered on your machine
BiocParallel::registered()
# the first one is the default backend
# ncores = ncores <- parallel::detectCores() - 2
permuteResult = generateNull(upSet = toy_gs_up, downSet = toy_gs_dn, ranked,
centerScore = TRUE, B =10, seed = 1, ncores = 1 )
```

---

getPvals

*Estimate the empirical p-values*


---

### Description

With null distributions estimated using the [generateNull\(\)](#) function, p-values are estimated using a one-tailed test. A minimum p-value of  $1/B$  can be achieved with  $B$  permutations.

### Usage

```
getPvals(permuteResult, scoredf)
```

### Arguments

permuteResult	A matrix, null distributions for each sample generated using the <a href="#">generateNull()</a> function
scoredf	A dataframe, the scored results of samples under test generated using the <a href="#">simpleScore()</a> function

### Value

Estimated p-values for enrichment of the signature in each sample. A p-value of  $1/B$  indicates that the estimated p-value is less than or equal to  $1/B$ .

### Examples

```
ranked <- rankGenes(toy_expr_se)
scoredf <- simpleScore(ranked, upSet = toy_gs_up, downSet = toy_gs_dn)
# find out what backends can be registered on your machine
BiocParallel::registered()
# the first one is the default backend, and it can be changed explicitly.
# See vignette for more details
permuteResult = generateNull(upSet = toy_gs_up, downSet = toy_gs_dn, ranked,
B =10, seed = 1, useBPPARAM = NULL)

# call the permutation function to generate the empirical scores
# for B times.
pvals <- getPvals(permuteResult,scoredf)
```

multiScore

*single-sample gene-set scoring method for multiple signatures***Description**

This function computes 'singscores' using a ranked gene expression matrix obtained from the [rankGenes\(\)](#) function and a GeneSetCollection object or a list of GeneSet objects. It returns a list of two matrices containing the scores and dispersions. This function should be used when scoring needs to be performed for multiple signatures. It is faster than applying [simpleScore\(\)](#) across the different signatures independently.

**Usage**

```
multiScore(rankData, upSetColc, downSetColc, subSamples = NULL,
           centerScore = TRUE, dispersionFun = mad, knownDirection = TRUE)

## S4 method for signature 'matrix, GeneSetCollection, missing'
multiScore(rankData,
           upSetColc, downSetColc, subSamples = NULL, centerScore = TRUE,
           dispersionFun = mad, knownDirection = TRUE)

## S4 method for signature 'matrix, GeneSetCollection, GeneSetCollection'
multiScore(rankData,
           upSetColc, downSetColc, subSamples = NULL, centerScore = TRUE,
           dispersionFun = mad, knownDirection = TRUE)

## S4 method for signature 'matrix, list, missing'
multiScore(rankData, upSetColc,
           downSetColc, subSamples = NULL, centerScore = TRUE,
           dispersionFun = mad, knownDirection = TRUE)

## S4 method for signature 'matrix, list, list'
multiScore(rankData, upSetColc, downSetColc,
           subSamples = NULL, centerScore = TRUE, dispersionFun = mad,
           knownDirection = TRUE)
```

**Arguments**

rankData	A matrix object, ranked gene expression matrix data generated using the <a href="#">rankGenes()</a> function
upSetColc	A GeneSetCollection object or a list of GeneSet objects of up-regulated (or mixed, see <a href="#">simpleScore</a> ) gene sets.
downSetColc	A GeneSetCollection object or a list of GeneSet objects of down-regulated gene sets. NULL otherwise.
subSamples	A vector of sample labels/indices that will be used to subset the rankData matrix. All samples will be scored if not provided
centerScore	A Boolean, specifying whether scores should be centered around 0, default as TRUE. Note: scores never centered if knownDirection = FALSE

`dispersionFun` A function, dispersion function with default being `mad`

`knownDirection` A boolean, determining whether the gene set should be considered to be directional or not. A gene set is directional if the type of genes in it are known i.e. up- or down-regulated. This should be set to `TRUE` if the gene set is composed of both up- AND down-regulated genes. Defaults to `TRUE`. This parameter becomes irrelevant when both `upSet(Colc)` and `downSet(Colc)` are provided.

### Value

A list of two matrices containing the scores and dispersions

### See Also

[rank "GeneSet"](#)

### Examples

```
ranked <- rankGenes(toy_expr_se)
setName(toy_gs_up) = "toy_gs_up"
setName(toy_gs_dn) = "toy_gs_dn"
gslist <- list(toy_gs_up, toy_gs_dn)

gscolc <- GSEABase::GeneSetCollection(gslist)
scoredf <- multiScore(ranked, upSetColc = gscolc)
```

---

`plotDispersion` *Plot the score v.s. dispersion for all samples*

---

### Description

This function takes the output from the `simpleScore()` function and generates scatter plots of score vs. dispersion for the total score, the up score and the down score of samples. If you wish to use the plotting function but with some customized inputs (instead of outputs from `simpleScore` function), you need to make sure the formats are the same. To be specific, you need to have column names "TotalScore" "TotalDispersion" "UpScore" "UpDispersion" "DownScore" "DownDispersion" and rows names as samples.

### Usage

```
plotDispersion(scoredf, annot = NULL, annot_name = "", alpha = 1,
  size = 1, textSize = 1.2, isInteractive = FALSE)
```

### Arguments

`scoredf` data.frame, generated using the [simpleScore\(\)](#) function

`annot` any numeric, character or factor annotation provided by the user that needs to be plot. Alternatively, this can be a character specifying the column of `scoredf` holding the annotation. Annotations must be ordered in the same way as the scores

`annot_name` character, legend title for the annotation

`alpha` numeric, set the transparency of points

size	numeric, set the size of each point
textSize	numeric, relative text sizes for title, labels, and axis values
isInteractive	Boolean, determine whether the plot is interactive

**Value**

A ggplot object

**Examples**

```
ranked <- rankGenes(toy_expr_se)
scoredf <- simpleScore(ranked, upSet = toy_gs_up, downSet = toy_gs_dn)
plotDispersion(scoredf)
plotDispersion(scoredf, isInteractive = TRUE)
```

---

plotNull	<i>Plot the empirically estimated null distribution and associated p-values</i>
----------	---

---

**Description**

This function takes the results from function [generateNull\(\)](#) and plots the density curves of permuted scores for the provided samples via `sampleNames` parameter. It can plot null distribution(s) for a single sample or multiple samples.

**Usage**

```
plotNull(permuteResult, scoredf, pvals, sampleNames = NULL,
         cutoff = 0.01, textSize = 2, labelSize = 5)
```

**Arguments**

permuteResult	A matrix, null distributions for each sample generated using the <a href="#">generateNull()</a> function
scoredf	A dataframe, singscores generated using the <a href="#">simpleScore()</a> function
pvals	A vector, estimated p-values using the <a href="#">getPvals()</a> function permuteResult, scoredf and pvals are the results for the same samples.
sampleNames	A character vector, sample IDs for which null distributions will be plotted
cutoff	numeric, the cutoff value for determining significance
textSize	numeric, size of axes labels, axes values and title
labelSize	numeric, size of label texts

**Value**

a ggplot object

**Author(s)**

Ruqian Lyu

**Examples**

```

ranked <- rankGenes(toy_expr_se)
scoredf <- simpleScore(ranked, upSet = toy_gs_up, downSet = toy_gs_dn)
# find out what backends can be registered on your machine
BiocParallel::registered()
# the first one is the default backend, and it can be changed explicitly.
permuteResult = generateNull(upSet = toy_gs_up, downSet = toy_gs_dn, ranked,
B =10, seed = 1,useBPPARAM = NULL)
# call the permutation function to generate the empirical scores
#for B times.
pvals <- getPvals(permuteResult,scoredf)
# plot for all samples
plotNull(permuteResult,scoredf,pvals,sampleNames = names(pvals))
#plot for the first sample
plotNull(permuteResult,scoredf,pvals,sampleNames = names(pvals)[1])

```

---

plotRankDensity

*Plot the densities of ranks for one sample*


---

**Description**

This function takes a single-column data frame, which is a single-column subset of the ranked matrix data generated using `rankGenes()` function, and the gene sets of interest as inputs. It plots the density of ranks for genes in the gene set and overlays a barcode plot of these ranks. Ranks are normalized by dividing them by the maximum rank. Densities are estimated using KDE.

**Usage**

```

plotRankDensity(rankData, upSet, downSet = NULL, isInteractive = FALSE,
  textSize = 1.5)

```

```

## S4 method for signature 'ANY,vector,missing'
plotRankDensity(rankData, upSet,
  downSet = NULL, isInteractive = FALSE, textSize = 1.5)

```

```

## S4 method for signature 'ANY,GeneSet,missing'
plotRankDensity(rankData, upSet,
  downSet = NULL, isInteractive = FALSE, textSize = 1.5)

```

```

## S4 method for signature 'ANY,vector,vector'
plotRankDensity(rankData, upSet,
  downSet = NULL, isInteractive = FALSE, textSize = 1.5)

```

```

## S4 method for signature 'ANY,GeneSet,GeneSet'
plotRankDensity(rankData, upSet,
  downSet = NULL, isInteractive = FALSE, textSize = 1.5)

```

**Arguments**

`rankData` one column of the ranked gene expression matrix obtained from the `rankGenes()` function, use `drop = FALSE` when subsetting the ranked gene expression matrix, see examples.

upSet	GeneSet object or a vector of gene Ids, up-regulated gene set
downSet	GeneSet object or a vector of gene Ids, down-regulated gene set
isInteractive	Boolean, determine whether the returned plot is interactive
textSize	numeric, set the size of text on the plot

**Value**

A ggplot object (or a plotly object) with a rank density plot overlaid with a barcode plot

**Examples**

```
ranked <- rankGenes(toy_expr_se)
plotRankDensity(ranked[,2,drop = FALSE], upSet = toy_gs_up)
```

---

plotScoreLandscape      *Plot landscape of two gene signatures scores*

---

**Description**

This function takes two data frames which are outputs from the simpleScore() function and plots the relationship between the two gene set scores for samples in the gene expression matrix. Scoredf1 and Scoredf2 are two scoring results of the same set of samples against two different gene signatures. If you wish to use the plotting function but with some customized inputs (instead of outputs from the simpleScore function), you need to make sure the formats are the same. To be specific, you need to have column names "TotalScore" "TotalDispersion" "UpScore" "UpDispersion" "DownScore" "DownDispersion" and rows names as samples.

**Usage**

```
plotScoreLandscape(scoredf1, scoredf2, scorenames = c(),
  textSize = 1.2, isInteractive = FALSE, hexMin = 100)
```

**Arguments**

scoredf1	data.frame, result of the simpleScore() function which scores the gene expression matrix against a gene set of interest
scoredf2	data.frame, result of the simpleScore() function which scores the gene expression matrix against another gene set of interest
scorenames	character vector of length 2, names for the two scored gene set/signatures stored in scoredf1 and scoredf2
textSize	numeric, set the text size for the plot, default as 1.5
isInteractive	boolean, whether the plot is interactive default as FALSE
hexMin	integer, the threshold which decides whether hex bin plot or scatter plot is displayed, default as 100

**Value**

A ggplot object, a scatter plot, demonstrating the relationship between scores from two signatures on the same set of samples.

**Examples**

```
ranked <- rankGenes(toy_expr_se)
scoredf <- simpleScore(ranked, upSet = toy_gs_up, downSet = toy_gs_dn)
scoredf2 <- simpleScore(ranked, upSet = toy_gs_up)
plotScoreLandscape(scoredf, scoredf2)
```

---

projectScoreLandscape *Project data on the landscape plot obtained from plotScoreLandscape()*

---

**Description**

This function takes the output (ggplot object) of the function `plotScoreLandscape()` and a new dataset. It projects the new data points onto the landscape plot and returns a new ggplot object with projected data points.

**Usage**

```
projectScoreLandscape(plotObj = NULL, scoredf1, scoredf2, annot = NULL,
  annot_name = NULL, subSamples = NULL, sampleLabels = NULL,
  isInteractive = FALSE)
```

**Arguments**

<code>plotObj</code>	a ggplot object, resulted from <code>plotScoreLandscape()</code>
<code>scoredf1</code>	data.frame, result of the <code>simpleScore()</code> function which scores the gene expression matrix against a gene set of interest
<code>scoredf2</code>	data.frame, result of the <code>simpleScore()</code> function which scores the gene expression matrix against another gene set of interest. Scores in <code>scoredf1</code> and <code>scoredf2</code> consist of the new data points that will be projected on the <code>plotObj</code> landscape plot.
<code>annot</code>	any numeric, character or factor annotation provided by the user that needs to be plot. Alternatively, this can be a character specifying the column of <code>scoredf1</code> holding the annotation. Annotations must be ordered in the same way as the scores
<code>annot_name</code>	character, legend title for the annotation
<code>subSamples</code>	vector of character or indices for subsetting the <code>scoredfs</code> , default as <code>NULL</code> and all samples in <code>scoredfs</code> will be plotted. The subsetted samples are projected onto the landscape plot of <code>plotObj</code> .
<code>sampleLabels</code>	vector of character, sample names to display, ordered in the same way as samples are ordered in the 'scoredfs' data matrix, default as <code>NULL</code> which means the projected points are not labelled.
<code>isInteractive</code>	boolean, whether the plot is interactive default as <code>FALSE</code>

**Value**

New data points on the already plotted ggplot object from `plotScoreLandscape()`

**See Also**

```
plotScoreLandscape() @examples ranked <- rankGenes(toy_expr_se) scoredf1 <- simpleScore(ranked,
upSet = toy_gs_up, downSet = toy_gs_dn) scoredf2 <- simpleScore(ranked, upSet = toy_gs_up) psl
<- plotScoreLandscape(scoredf1, scoredf2) projectScoreLandscape(psl,scoredf1, scoredf2)
```

rankGenes

*Rank genes by the gene expression intensities***Description**

The rankGenes function is a generic function that can deal with multiple types of inputs. Given a matrix of gene expression that has samples in columns, genes in rows, and values being gene expression intensity, rankGenes ranks gene expression intensities in each sample.

It can also work with S4 objects that have gene expression matrix as a component (i.e ExpressionSet, DGEList, SummarizedExperiment). It calls the rank function in the base package which ranks the gene expression matrix by its absolute expression level. If the input is S4 object of DGEList, ExpressionSet, or SummarizedExperiment, it will extract the gene expression matrix from the object and rank the genes. The default 'tiesMethod' is set to 'min'.

**Usage**

```
rankGenes(expreMatrix, tiesMethod = "min")

## S4 method for signature 'matrix'
rankGenes(expreMatrix, tiesMethod = "min")

## S4 method for signature 'data.frame'
rankGenes(expreMatrix, tiesMethod = "min")

## S4 method for signature 'DGEList'
rankGenes(expreMatrix, tiesMethod = "min")

## S4 method for signature 'ExpressionSet'
rankGenes(expreMatrix, tiesMethod = "min")

## S4 method for signature 'SummarizedExperiment'
rankGenes(expreMatrix,
  tiesMethod = "min")
```

**Arguments**

expreMatrix	A gene expression matrix (matrix,data.frame) or S4 object (ExpressionSet,DGEList, SummarizedExperiment)
tiesMethod	A character indicating what method to use when dealing with ties

**Value**

The ranked gene expression matrix that has samples in columns and genes in rows

**See Also**

[rank ExpressionSet SummarizedExperiment DGEList](#)

**Examples**

```
rankGenes(toy_expr_se) # toy_expr_se is a gene expression dataset
tiesMethod = 'min'
# get counts from toy_expr_se
counts <- SummarizedExperiment::assay(toy_expr_se)
# or it can be a ExpressionSet object

e <- Biobase::ExpressionSet(assayData = as.matrix(counts))
rankGenes(e)
```

---

scoredf_ccle_epi	<i>Pre-computed scores of the CCLE dataset against an epithelial gene signature</i>
------------------	---

---

**Description**

This data.frame stores pre-computed scores of the CCLE dataset [Barretina et al](#) calculated using the [simpleScore\(\)](#) function against the epithelial gene signature from [Tan, Tuan Zea et al](#). The data.frame has scores for 55 samples. Please refer to the vignettes for instructions on how to obtain the full datasets.

**Usage**

```
scoredf_ccle_epi
```

**Format**

An object of class `data.frame` with 55 rows and 2 columns.

**References**

Barretina, Jordi, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A Margolin, Sungjoon Kim, Christopher J Wilson, et al. 2012. “The Cancer Cell Line Encyclopedia Enables Predictive Modelling of Anticancer Drug Sensitivity.” *Nature* 483 (7391): 603–7.

Tan, Tuan Zea, Qing Hao Miow, Yoshio Miki, Tetsuo Noda, Seiichi Mori, Ruby Yun-Ju Huang, and Jean Paul Thiery. 2014–10AD. “Epithelial-Mesenchymal Transition Spectrum Quantification and Its Efficacy in Deciphering Survival and Drug Responses of Cancer Patients.” *EMBO Molecular Medicine* 6 (10). Oxford, UK: BlackWell Publishing Ltd: 1279–93. doi:10.15252/emmm.201404208.

**See Also**

[scoredf\\_ccle\\_mes](#)

---

scoredf_ccle_mes	<i>Pre-computed scores of the CCLE dataset against a mesenchymal gene signature</i>
------------------	---

---

### Description

This data.frame stores pre-computed scores of the CCLE dataset [Barretina et al](#) calculated using the [simpleScore\(\)](#) function against the mesenchymal gene signature from [Tan, Tuan Zea et al](#). The data.frame has scores for 55 samples. Please refer to the vignettes for instructions on how to obtain the full datasets.

### Usage

```
scoredf_ccle_mes
```

### Format

An object of class `data.frame` with 55 rows and 2 columns.

### References

Barretina, Jordi, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A Margolin, Sungjoon Kim, Christopher J Wilson, et al. 2012. “The Cancer Cell Line Encyclopedia Enables Predictive Modelling of Anticancer Drug Sensitivity.” *Nature* 483 (7391): 603–7.

Tan, Tuan Zea, Qing Hao Miow, Yoshio Miki, Tetsuo Noda, Seiichi Mori, Ruby Yun-Ju Huang, and Jean Paul Thiery. 2014–10AD. “Epithelial-Mesenchymal Transition Spectrum Quantification and Its Efficacy in Deciphering Survival and Drug Responses of Cancer Patients.” *EMBO Molecular Medicine* 6 (10). Oxford, UK: BlackWell Publishing Ltd: 1279–93 doi:10.15252/emmm.201404208.

### See Also

[scoredf\\_ccle\\_epi](#)

---

scoredf_tcga_epi	<i>Pre-computed scores of the TCGA breast cancer gene expression matrix against an epithelial signature</i>
------------------	---

---

### Description

This data.frame stores pre-computed scores of the TCGA dataset calculated using the [simpleScore\(\)](#) function against the epithelial gene signature from [Tan, Tuan Zea et al](#). Please refer to the vignettes for instructions on how to obtain the full datasets.

### Usage

```
scoredf_tcga_epi
```

### Format

An object of class `data.frame` with 1119 rows and 2 columns.

## References

Tan, Tuan Zea, Qing Hao Miow, Yoshio Miki, Tetsuo Noda, Seiichi Mori, Ruby Yun-Ju Huang, and Jean Paul Thiery. 2014–10AD. “Epithelial-Mesenchymal Transition Spectrum Quantification and Its Efficacy in Deciphering Survival and Drug Responses of Cancer Patients.” *EMBO Molecular Medicine* 6 (10). Oxford, UK: BlackWell Publishing Ltd: 1279–93 doi:10.15252/emmm.201404208.

## See Also

[scoredf\\_tcga\\_mes](#)

---

scoredf_tcga_mes	<i>Pre-computed scores of the TCGA breast cancer gene expression matrix against a mesenchymal signature</i>
------------------	---

---

## Description

This data.frame stores pre-computed scores of the **TCGA** dataset calculated using the `simpleScore()` function against the mesenchymal gene signature from **Tan, Tuan Zea et al.** Please refer to the vignettes for instructions on how to obtain the full datasets.

## Usage

```
scoredf_tcga_mes
```

## Format

An object of class `data.frame` with 1119 rows and 2 columns.

## References

Tan, Tuan Zea, Qing Hao Miow, Yoshio Miki, Tetsuo Noda, Seiichi Mori, Ruby Yun-Ju Huang, and Jean Paul Thiery. 2014–10AD. “Epithelial-Mesenchymal Transition Spectrum Quantification and Its Efficacy in Deciphering Survival and Drug Responses of Cancer Patients.” *EMBO Molecular Medicine* 6 (10). Oxford, UK: BlackWell Publishing Ltd: 1279–93 doi:10.15252/emmm.201404208.

## See Also

[scoredf\\_tcga\\_epi](#)

simpleScore

*single-sample gene-set scoring method***Description**

This function computes 'singscores' using a ranked gene expression matrix obtained from the [rankGenes\(\)](#) function and a gene set or a pair of up-regulated and down-regulated gene sets. It returns a data.frame of scores and dispersions for each sample. The gene sets can be in vector format or as GeneSet objects (from GSEABase packages). If samples need to be scored against a single gene set, the upSet argument should be used to pass the gene set while the downSet argument is set to NULL. This setting is ideal for gene sets representing gene ontologies where the nature of the genes is unknown (up- or down-regulated).

**Usage**

```
simpleScore(rankData, upSet, downSet = NULL, subSamples = NULL,
           centerScore = TRUE, dispersionFun = mad, knownDirection = TRUE)
```

```
## S4 method for signature 'matrix,vector,missing'
simpleScore(rankData, upSet,
           downSet = NULL, subSamples = NULL, centerScore = TRUE,
           dispersionFun = mad, knownDirection = TRUE)
```

```
## S4 method for signature 'matrix,GeneSet,missing'
simpleScore(rankData, upSet,
           downSet = NULL, subSamples = NULL, centerScore = TRUE,
           dispersionFun = mad, knownDirection = TRUE)
```

```
## S4 method for signature 'matrix,vector,vector'
simpleScore(rankData, upSet,
           downSet = NULL, subSamples = NULL, centerScore = TRUE,
           dispersionFun = mad, knownDirection = TRUE)
```

```
## S4 method for signature 'matrix,GeneSet,GeneSet'
simpleScore(rankData, upSet,
           downSet = NULL, subSamples = NULL, centerScore = TRUE,
           dispersionFun = mad, knownDirection = TRUE)
```

**Arguments**

rankData	A matrix object, ranked gene expression matrix data generated using the <a href="#">rankGenes()</a> function
upSet	A GeneSet object or character vector of gene IDs of up-regulated gene set or a gene set where the nature of genes is not known
downSet	A GeneSet object or character vector of gene IDs of down-regulated gene set or NULL where only a single gene set is provided
subSamples	A vector of sample labels/indices that will be used to subset the rankData matrix. All samples will be scored if not provided
centerScore	A Boolean, specifying whether scores should be centered around 0, default as TRUE. Note: scores never centered if knownDirection = FALSE

`dispersionFun` A function, dispersion function with default being `mad`

`knownDirection` A boolean, determining whether the gene set should be considered to be directional or not. A gene set is directional if the type of genes in it are known i.e. up- or down-regulated. This should be set to `TRUE` if the gene set is composed of both up- AND down-regulated genes. Defaults to `TRUE`. This parameter becomes irrelevant when both `upSet(Colc)` and `downSet(Colc)` are provided.

### Value

A data.frame consists of singscores and dispersions for all samples

### See Also

[rank "GeneSet"](#)

### Examples

```
ranked <- rankGenes(toy_expr_se)
scoredf <- simpleScore(ranked, upSet = toy_gs_up, downSet = toy_gs_dn)
# toy_gs_up is a GeneSet object, alternatively a vector of gene ids may also
# be supplied.
```

---

<code>singscore</code>	<i>singscore: A package for deriving gene-set scores at a single sample level</i>
------------------------	---

---

### Description

The package provides functions for calculating gene-set enrichment scores at a single-sample level using gene expression data. It includes functions to perform hypothesis testing and provides visualisations to enable diagnosis of scores and gene sets along with visualisations to enable exploration of results.

---

<code>tgfb_expr_10_se</code>	<i>An example gene expression dataset</i>
------------------------------	---

---

### Description

A microarray gene expression dataset that was originally obtained from the integrated TGFb-EMT data published by (Foroutan et al, 2017). (ComBat corrected values). `tgfb_expr_10` is a subset of the integrated TGFb-EMT data consisting of 10 samples (4 TGFb treated and 6 controls) each with expression values for 11900 genes.

### Usage

```
tgfb_expr_10_se
```

### Format

A SummarizedExperiment object

**Source**

[Foroutan et al,2017](#)

**References**

Foroutan, Momeneh, Joseph Cursons, Soroor Hadiyah-Zadeh, Erik W Thompson, and Melissa J Davis. 2017. "A Transcriptional Program for Detecting Tgfbeta-Induced Emt in Cancer." Molecular Cancer Research. American Association for Cancer Research. doi:10.1158/1541-7786.MCR-16-0313.

---

tgfb\_gs\_dn

*Gene set of down-regulated genes for the TGFb-induced EMT gene signature*

---

**Description**

A GeneSet object that contains the down-regulated genes of the TGFb-induced EMT gene signature that was derived by (Foroutan et al,2017), using two meta-analysis techniques. The gene signature contains an up-regulated gene set (up-set) and a down-regulated gene set (down-set). Please refer to the vignettes for the steps to acquire the exact data object.

**Usage**

tgfb\_gs\_dn

**Format**

A GeneSet object

**Source**

[Foroutan et al,2017](#)

**References**

Foroutan, Momeneh, Joseph Cursons, Soroor Hadiyah-Zadeh, Erik W Thompson, and Melissa J Davis. 2017. "A Transcriptional Program for Detecting Tgfbeta-Induced Emt in Cancer." Molecular Cancer Research. American Association for Cancer Research. doi:10.1158/1541-7786.MCR-16-0313.

**See Also**

"GeneSet", [tgfb\\_gs\\_up](#)

---

tgfb_gs_up	<i>Gene set of up-regulated genes for the TGFb-induced EMT gene signature</i>
------------	---

---

### Description

A GeneSet object that contains the up-regulated genes of the TGFb-induced EMT gene signature that was derived by (Foroutan et al.,2017), using two meta-analysis techniques. The gene signature contains an up-regulated gene set (up-set) and a down-regulated gene set (down-set). Please refer to the vignettes for the steps to acquire the exact data object.

### Usage

```
tgfb_gs_up
```

### Format

A GeneSet object

### Source

[Foroutan et al,2017](#)

### References

Foroutan, Momeneh, Joseph Cursons, Soroor Hediye-Zadeh, Erik W Thompson, and Melissa J Davis. 2017. "A Transcriptional Program for Detecting Tgfbeta-Induced Emt in Cancer." Molecular Cancer Research. American Association for Cancer Research. doi:10.1158/1541-7786.MCR-16-0313.

### See Also

["GeneSet",tgfb\\_gs\\_dn](#)

---

toy_expr_se	<i>A toy gene expression dataset of two samples</i>
-------------	---

---

### Description

A toy dataset consisting of 2 samples with the expression values of 20 genes. The data was created by sampling 2 samples and 20 genes from the dataset by Foroutan et al, 2017.

### Usage

```
toy_expr_se
```

### Format

A SummarizedExperiment of 2 samples each with 20 genes

**D\_Ctrl\_R1** a control sample

**D\_TGFb\_R1** a TGFb-treated sample

**Source**

Foroutan et al.,2017

**References**

Foroutan, Momeneh, Joseph Cursons, Soroor Hadiyah-Zadeh, Erik W Thompson, and Melissa J Davis. 2017. "A Transcriptional Program for Detecting Tgfbeta-Induced Emt in Cancer." Molecular Cancer Research. American Association for Cancer Research. doi:10.1158/1541-7786.MCR-16-0313.

---

toy\_gs\_dn

*A gene set object of down-regulated genes for the toy dataset*

---

**Description**

A GeneSet object with 5 genes randomly selected from the toy dataset. These genes are independent of those in [toy\\_gs\\_up](#)

**Usage**

toy\_gs\_dn

**Format**

A GSEABase::GeneSet object with 5 genes

**See Also**

"GeneSet",[toy\\_expr\\_se](#),[toy\\_gs\\_up](#)

---

toy\_gs\_up

*A gene set object of up-regulated genes for the toy dataset*

---

**Description**

A GeneSet object with 5 genes randomly selected from the toy dataset. These genes are independent of those in [toy\\_gs\\_dn](#)

**Usage**

toy\_gs\_up

**Format**

A GeneSet object with 5 genes

**See Also**

"GeneSet",[toy\\_expr\\_se](#),[toy\\_gs\\_dn](#)

# Index

## \*Topic **datasets**

scoredf\_ccle\_epi, [12](#)  
scoredf\_ccle\_mes, [13](#)  
scoredf\_tcga\_epi, [13](#)  
scoredf\_tcga\_mes, [14](#)  
tgfb\_expr\_10\_se, [16](#)  
tgfb\_gs\_dn, [17](#)  
tgfb\_gs\_up, [18](#)  
toy\_expr\_se, [18](#)  
toy\_gs\_dn, [19](#)  
toy\_gs\_up, [19](#)

BiocParallel::bplapply(), [2](#)

DGEList, [12](#)

ExpressionSet, [12](#)

generateNull, [2](#)

generateNull(), [4, 7](#)

generateNull, GeneSet, GeneSet-method  
(generateNull), [2](#)

generateNull, GeneSet, missing-method  
(generateNull), [2](#)

generateNull, vector, missing-method  
(generateNull), [2](#)

generateNull, vector, vector-method  
(generateNull), [2](#)

GeneSet, [6, 16–19](#)

getPvals, [4](#)

getPvals(), [7](#)

multiScore, [5](#)

multiScore, matrix, GeneSetCollection, GeneSetCollection-method  
(multiScore), [5](#)

multiScore, matrix, GeneSetCollection, missing-method  
(multiScore), [5](#)

multiScore, matrix, list, list-method  
(multiScore), [5](#)

multiScore, matrix, list, missing-method  
(multiScore), [5](#)

plotDispersion, [6](#)

plotNull, [7](#)

plotRankDensity, [8](#)

plotRankDensity, ANY, GeneSet, GeneSet-method  
(plotRankDensity), [8](#)

plotRankDensity, ANY, GeneSet, missing-method  
(plotRankDensity), [8](#)

plotRankDensity, ANY, vector, missing-method  
(plotRankDensity), [8](#)

plotRankDensity, ANY, vector, vector-method  
(plotRankDensity), [8](#)

plotScoreLandscape, [9](#)

plotScoreLandscape(), [10, 11](#)

projectScoreLandscape, [10](#)

rank, [6, 12, 16](#)

rankGenes, [11](#)

rankGenes(), [3, 5, 8, 15](#)

rankGenes, data.frame-method  
(rankGenes), [11](#)

rankGenes, DGEList-method (rankGenes), [11](#)

rankGenes, ExpressionSet-method  
(rankGenes), [11](#)

rankGenes, matrix-method (rankGenes), [11](#)

rankGenes, SummarizedExperiment-method  
(rankGenes), [11](#)

scoredf\_ccle\_epi, [12, 13](#)

scoredf\_ccle\_mes, [12, 13](#)

scoredf\_tcga\_epi, [13, 14](#)

scoredf\_tcga\_mes, [14, 14](#)

simpleScore, [5, 15](#)

simpleScore(), [4–7, 12–14](#)

simpleScore, matrix, GeneSet, GeneSet-method  
(simpleScore), [15](#)

simpleScore, matrix, GeneSet, missing-method  
(simpleScore), [15](#)

simpleScore, matrix, vector, missing-method  
(simpleScore), [15](#)

simpleScore, matrix, vector, vector-method  
(simpleScore), [15](#)

singscore, [16](#)

singscore-package (singscore), [16](#)

SummarizedExperiment, [12](#)

tgfb\_expr\_10\_se, [16](#)

tgfb\_gs\_dn, [17, 18](#)

tgfb\_gs\_up, [17](#), [18](#)  
toy\_expr\_se, [18](#), [19](#)  
toy\_gs\_dn, [19](#), [19](#)  
toy\_gs\_up, [19](#), [19](#)