Package 'dslice'

November 17, 2018

Type Package

Title Dynamic Slicing

Version 1.2.0

Date 2018-11-16

Author Chao Ye and Bo Jiang

Maintainer Chao Ye <yechao1009@gmail.com>

Description Dynamic slicing is a method designed for dependency detection between a categorical variable and a continuous variable. It could be applied for non-parametric hypothesis testing and gene set enrichment analysis.

License GPL (>= 2)

Depends R (>= 2.10), stats, utils, Rcpp (>= 0.11.1), ggplot2 (>= 0.9.3.1), scales

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-11-17 05:40:07 UTC

R topics documented:

`bfslice_c` *Dependency and conditional dependency detection between a level k ($k > 1$) categorical variable and a continuous variable via Bayes factor.*

Description

Conditional dependency detection between a level k_x ($k_x > 1$) categorical variable `x` and a continuous variable `y` via Bayes factor given a level k_z categorical variable `z`. If $k_z = 1$, it is unconditional dependency detection method. It could be applied for non-parametric variable selecltion.

Usage

```
bfslice_c(z, x, zdim, xdim, lambda, alpha)
```

Arguments

`z`	Vector: observations of given (preselected) categorical variable, $0, 1, \ldots, k_z - 1$ for level k_z categorical variable, should be ranked according to values of continuous variable `y` with `x` in advanced, either ascending or descending.
`x`	Vector: observations of categorical variable, $0, 1, \ldots, k_x - 1$ for level k_x categorical variable, should be ranked according to values of continuous variable `y` with `z` in advanced, either ascending or descending.
`zdim`	Level of `z`, equals k_z.
`xdim`	Level of `x`, equals k_x.
`lambda`	`lambda` corresponds to the probability that makes slice in each possible position. `lambda` should be greater than 0.
`alpha`	`alpha` is hyper-parameter of the prior distribution of frequency in each slice. `alpha` should be greater than 0 and less equal than k_x.

Value

Value of Bayes factor (nonnegative). Bayes factor could be treated as a statistic and one can take some threshold then calculates the corresponded Type I error rate. One can also take the value of Bayes factor for judgement.

References

Jiang, B., Ye, C. and Liu, J.S. Bayesian nonparametric tests via sliced inverse modeling. *Bayesian Analysis*, 12(1): 89-112, 2017.

See Also

bfslice_u, bfslice_eqp_c.

Examples

```
n <- 100
mu <- 0.5

## Unconditional test
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n), rep(1, n))
z <- rep(0, 2*n)

## Conditional test
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n/5), rep(1, n), rep(0, 4*n/5))
z <- c(rep(0, n), rep(1, n))
z <- z[order(y)]

x <- x[order(y)]
zdim <- max(z) + 1
xdim <- max(x) + 1
lambda <- 1.0
alpha <- 1.0
bfval <- bfslice_c(z, x, zdim, xdim, lambda, alpha)
```

---

bfslice_eqp_c *Dependency and conditional dependency detection between a level k ($k > 1$) categorical variable and a continuous variable via Bayes factor.*

Description

Conditional dependency detection between a level k_x ($k_x > 1$) categorical variable x and a continuous variable y via Bayes factor given a level k_z categorical variable z with $O(n^{1/2})$-resolution. The basic idea is almost the same as bfslice_c. The only different is that bfslice_eqp_c groups samples into approximate $O(n^{1/2})$ groups which contain approximate $O(n^{1/2})$ samples and treat the groups as a sample to calculate Bayes facor. If $k_z = 1$, it is unconditional dependency detection method. It could be applied for non-parametric variable selecltion.

Usage

```
bfslice_eqp_c(z, x, zdim, xdim, lambda, alpha)
```

Arguments

z	Vector: observations of given (preselected) categorical variable, $0, 1, \ldots, k_z - 1$ for level k_z categorical variable, should be ranked according to values of continuous variable y with x in advanced, either ascending or descending.
x	Vector: observations of categorical variable, $0, 1, \ldots, k_x - 1$ for level k_x categorical variable, should be ranked according to values of continuous variable y with z in advanced, either ascending or descending.
zdim	Level of z, equals k_z.
xdim	Level of x, equals k_x.
lambda	lambda corresponds to the probability that makes slice in each possible position. lambda should be greater than 0.
alpha	alpha is hyper-parameter of the prior distribution of frequency in each slice. alpha should be greater than 0 and less equal than k_x.

Value

Value of Bayes factor (nonnegative). Bayes factor could be treated as a statistic and one can take some threshold then calculates the corresponded Type I error rate. One can also take the value of Bayes factor for judgement.

References

Jiang, B., Ye, C. and Liu, J.S. Bayesian nonparametric tests via sliced inverse modeling. *Bayesian Analysis*, 12(1): 89-112, 2017.

See Also

bfslice_c, bfslice_eqp_u.

Examples

```
n <- 1000
mu <- 0.2

## Unconditional test
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n), rep(1, n))
z <- rep(0, 2*n)

## Conditional test
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n/5), rep(1, n), rep(0, 4*n/5))
z <- c(rep(0, n), rep(1, n))
z <- z[order(y)]

x <- x[order(y)]
zdim <- max(z) + 1
xdim <- max(x) + 1
lambda <- 1.0
```

```
alpha <- 1.0
bfval <- bfslice_eqp_c(z, x, zdim, xdim, lambda, alpha)
```

`bfslice_eqp_u` *Dependency detection between a level k ($k > 1$) categorical variable and a continuous variable via Bayes factor with given size of each group.*

Description

Dependency detection between a level k ($k > 1$) categorical variable `x` and a continuous variable `y` via Bayes factor with $O(n^{1/2})$-resolution. The basic idea is almost the same as `bfslice_u`. The only different is that `bfslice_eqp_u` groups samples into approximate $O(n^{1/2})$ groups which contain approximate $O(n^{1/2})$ samples and treat the groups as a sample to calculate Bayes facor.

Usage

```
bfslice_eqp_u(x, dim, lambda, alpha)
```

Arguments

`x`	Vector: observations of categorical variable, $0, 1, \ldots, k-1$ for level k categorical variable, should be ranked according to values of continuous variable `y`, either ascending or descending.
`dim`	Level of `x`, equals k.
`lambda`	`lambda` corresponds to the probability that makes slice in each possible position. `lambda` should be greater than 0.
`alpha`	`alpha` is hyper-parameter of the prior distribution of frequency in each slice. `alpha` should be greater than 0 and less equal than k.

Value

Value of Bayes factor (nonnegative). Bayes factor could be treated as a statistic and one can take some threshold then calculates the corresponded Type I error rate. One can also take the value of Bayes factor for judgement.

References

Jiang, B., Ye, C. and Liu, J.S. Bayesian nonparametric tests via sliced inverse modeling. *Bayesian Analysis*, 12(1): 89-112, 2017.

See Also

`bfslice_u`, `bfslice_eqp_c`.

Examples

```
n <- 1000
mu <- 0.2
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n), rep(1, n))
x <- x[order(y)]
dim <- max(x) + 1
lambda <- 1.0
alpha <- 1.0
bfval <- bfslice_eqp_u(x, dim, lambda, alpha)
```

`bfslice_u` *Dependency detection between a level k ($k > 1$) categorical variable and a continuous variable via Bayes factor.*

Description

Dependency detection between a level k ($k > 1$) categorical variable `x` and a continuous variable `y` via Bayes factor.

Usage

```
bfslice_u(x, dim, lambda, alpha)
```

Arguments

`x`	Vector: observations of categorical variable, $0, 1, \ldots, k-1$ for level k categorical variable, should be ranked according to values of continuous variable `y`, either ascending or descending.
`dim`	Level of `x`, equals k.
`lambda`	`lambda` corresponds to the probability that makes slice in each possible position. `lambda` should be greater than 0.
`alpha`	`alpha` is hyper-parameter of the prior distribution of frequency in each slice. `alpha` should be greater than 0 and less equal than k.

Value

Value of Bayes factor (nonnegative). Bayes factor could be treated as a statistic and one can take some threshold then calculates the corresponded Type I error rate. One can also take the value of Bayes factor for judgement.

References

Jiang, B., Ye, C. and Liu, J.S. Bayesian nonparametric tests via sliced inverse modeling. *Bayesian Analysis*, 12(1): 89-112, 2017.

See Also

bfslice_c, bfslice_eqp_u.

Examples

```
n <- 100
mu <- 0.5
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n), rep(1, n))
x <- x[order(y)]
dim <- max(x) + 1
lambda <- 1.0
alpha <- 1.0
bfval <- bfslice_u(x, dim, lambda, alpha)
```

---

ds_1 *Non-parametric one-sample hypothesis testing via dynamic slicing*

Description

Non-parametric one-sample hypothesis testing via dynamic slicing. By mapping sample values to the quantile of null distribution, ds_1 test whether they follow uniform distribution on [0, 1] via a regularized likelihood-ratio. Its calculated is based on a dynamic programming procedure.

Usage

```
ds_1(y, lambda, alpha)
```

Arguments

y	Vector: quantiles of observations according to null distribution.
lambda	lambda penalizes the number of slices to avoid too many slices. lambda should be greater than 0.
alpha	alpha penalizes both the width and the number of slices to avoid too many slices and degenerate slice (interval). alpha should be greater than 1.

Value

Value of dynamic slicing statistic for one-sample test. It is nonnegative. The null hypothesis that observations are from the null distribution is rejected if this statistic is greater than zero, otherwise accept the null hypothesis.

See Also

ds_eqp_1.

Examples

```
n <- 100
mu <- 0.5
x <- rnorm(n, mu, 1)
y <- pnorm(sort(x), 0, 1)
lambda <- 1.0
alpha <- 1.0
dsres <- ds_1(y, lambda, alpha)
```

---

ds_eqp_1 *Non-parametric one-sample hypothesis testing via dynamic slicing*

Description

Non-parametric one-sample hypothesis testing via dynamic slicing with $O(n)$-resolution. The basic idea of ds_eqp_1 is almost the same as ds_1. Difference between these two functions is that ds_eqp_1 considers an equal partition on [0, 1] but ds_1 does not. Candidate slicing boundaries in ds_eqp_1 only depend on the total number of samples and are unrelated to sample quantiles. In ds_1 they are immediately to the left or right of sample quantile.

Usage

```
ds_eqp_1(y, lambda)
```

Arguments

y	Vector: quantiles of observations according to null distribution.
lambda	lambda penalizes the number of slices to avoid too many slices. Since the interval [0, 1] is divided into n equal size element-slice and slicing strategy only consider boundaries of them, this version of dynamic slicing does not require penlaty lambda as ds_1. lambda should be greater than 0.

Value

Value of dynamic slicing statistic for one-sample test. It is nonnegative. The null hypothesis that observations are from the null distribution is rejected if this statistic is greater than zero, otherwise accept the null hypothesis.

See Also

ds_1.

Examples

```
n <- 100
mu <- 0.5
x <- rnorm(n, mu, 1)
y <- pnorm(sort(x), 0, 1)
lambda <- 1.0
dsres <- ds_eqp_1(y, lambda)
```

`ds_eqp_k` *Dependency detection between level k ($k > 1$) categorical variable and continuous variable*

Description

Dependency detection between level k ($k > 1$) categorical variable and continuous variable via dynamic slicing with $O(n^{1/2})$-resolution. The basic idea is almost the same as `ds_k`. The only different is that `ds_eqp_k` groups samples into approximate $O(n^{1/2})$ groups which contain approximate $O(n^{1/2})$ samples and performs dynamic slicing on their boundaries. This much faster version could reduce computation time substantially without too much power loss. Based on the strategy of `ds_eqp_k`, we recommend to apply it in large sample size problem and use `ds_k` for ordinary problem. For more details please refer to Jiang, Ye & Liu (2015). Results contains value of dynamic slicing statistic and slicing strategy. It could be applied for non-parametric *K*-sample hypothesis testing.

Usage

```
ds_eqp_k(x, xdim, lambda, slice = FALSE)
```

Arguments

`x`	Vector: observations of categorical variable, $0, 1, \ldots, k-1$ for level k categorical variable, should be ranked according to values of continuous variable in advanced, either ascending or descending.
`xdim`	Level of `x`, equals k.
`lambda`	Penalty for introducing an additional slice, which is used to avoid making too many slices. It corresponds to the type I error under the scenario that the two variables are independent. `lambda` should be greater than 0.
`slice`	Indicator for reporting slicing strategy or not.

Value

`dsval`	Value of dynamic slicing statistic. It is nonnegative. If it equals zero, the categorical variable and continuous variable will be treated as independent of each other, otherwise they will be treated as dependent.

`slices` Slicing strategy that maximize dynamic slicing statistic based on currently ranked vector `x`. It will be reported if `slice` is true. Each row stands for a slice. Each column except the last one stands for the number of observations take each value in each slice. The last column is the number of observations in each slice *i.e.*, the sum of the first column to the *k*th column.

References

Jiang, B., Ye, C. and Liu, J.S. Non-parametric *K*-sample tests via dynamic slicing. *Journal of the American Statistical Association*, 110(510): 642-653, 2015.

See Also

`ds_k`.

Examples

```
n <- 100
mu <- 0.5
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep("1", n), rep("2", n))
x <- relabel(x)
x <- x[order(y)]
xdim <- max(x) + 1
lambda <- 1.0
dsres <- ds_eqp_k(x, xdim, lambda, slice = TRUE)
```

---

ds_gsa *Gene set analysis via dynamic slicing*

Description

Gene set analysis via dynamic slicing.

Usage

```
ds_gsa(expdat, geneset, label, generank, ..., lambda = 1, bycol = FALSE,
       minsize = 15, maxsize = 500, randseed = 11235, rounds = 1000)
```

Arguments

`expdat` Either a character string of gene expression file name (.gct file), or an expression matrix with rownames, each row is a gene and each column is a sample.

`geneset` Either a character string of gene set file name (.gmt file), or a list contains a vector of gene set names, a vector of gene set description and a list of gene symbols in each gene set.

`label` Either a character string of phenotypes file (.cls file), or a list contains a vector of types of pheotype and a vector of encoded pheotypes of samples. It should match gene expression matrix.

generank	Either an integer vector of rank of each gene according to some statistic, or a character string naming a function which takes gene expression matrix as input and returns a vector of gene rank (not tie).
...	Parameters of the function specified (as a character string) by generank.
lambda	Penalty for introducing an additional slice in dynamic slicing procedure, which is used to avoid making too many slices. It corresponds to the type I error under the scenario that the two variables are independent. lambda should be greater than 0.
bycol	Type of permutation, by row (default) or by column. Permutation by row means shuffling the gene rank. Permutation by column means shuffling pheotypes then obtain gene rank.
minsize	Minimum number of genes in genesets to be considered.
maxsize	Maximum number of genes in genesets to be considered.
randseed	Optional initial seed for random number generator (integer).
rounds	Number of permutations for estimating significant level of results.

Details

ds_gsa performs gene set analysis via dynamic slicing. It returns the DS statistics and slicing strategy of each gene set. ds_gsa does not attempt to integrate the ranking method into it. It requires ranking method or directly the gene rank as a parameter. Leaving ranking method as an optional input parameter is convenience for users who would like to use any ranking methods they want.

Value

A list with informations of gene sets whose size satisfy the minimum and maximum size thresholds. Its contains the following components:

set_name	A vector of gene set names.
set_size	A vector of gene set sizes.
DS_value	A vector of dynamic slicing statistic of each gene set.
pvalue	A vector of *p*-value of each gene set.
FDR	A vector of FDR of each gene set.
slices	A list of slicing strategy of each gene set. Each component is a matrix of slices.

References

Jiang, B., Ye, C. and Liu, J.S. Non-parametric *K*-sample tests via dynamic slicing. *Journal of the American Statistical Association*, 110(510): 642-653, 2015.

Subramanian, A., Tamayo, P., Mootha, V. K., *et al*. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 2005, 102(43): 15545-15550.

Benjamini, Y. and Hochberg, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1995, 57(1): 289-300.

See Also

ds_k.

Examples

```
##  Loading data from files
## Not run:
gctpath <- "P53.gct"
clspath <- "P53.cls"
gmtpath <- "C2.gmt"
expdat <- load_gct(gctpath)
label <- load_cls(clspath)
geneset <- load_gmt(gmtpath)
fc <- function(x, label)
{
  d0 <- apply(x[,which(label == 0)], 1, mean)
  d1 <- apply(x[,which(label == 1)], 1, mean)
  d <- d1 / d0
  return(order(d))
}
ds_gsa_obj <- ds_gsa(expdat, geneset, label, "fc", lambda = 1.2, bycol = TRUE,
                     minsize = 15, maxsize = 500, randseed = 11235, rounds = 100)

## End(Not run)
```

---

ds_k *Dependency detection between level k ($k > 1$) categorical variable and continuous variable*

Description

Dependency detection between level k ($k > 1$) categorical variable and continuous variable. The basic idea is that the different values of categorical variable correspond to different distribution of continuous variable if there exist dependency between this two varibles, otherwise the distributions of continuous variable do not show difference conditioning on the values of categorical variable. Statistic for this dynamic slicing method is a regularized likelihood-ratio calculated via a dynamic programming procedure. For more details please refer to Jiang, Ye & Liu (2015). Results contains value of dynamic slicing statistic and slicing strategy. It could be applied for non-parametric *K*-sample hypothesis testing.

Usage

```
ds_k(x, xdim, lambda, slice = FALSE)
```

Arguments

x	Vector: observations of categorical variable, $0, 1, \ldots, k-1$ for level k categorical variable, should be ranked according to values of continuous variable in advanced, either ascending or descending.

xdim	Level of x, equals k.
lambda	Penalty for introducing an additional slice, which is used to avoid making too many slices. It corresponds to the type I error under the scenario that the two variables are independent. lambda should be greater than 0.
slice	Indicator for reporting slicing strategy or not.

Value

dsval	Value of dynamic slicing statistic. It is nonnegative. If it equals zero, the categorical variable and continuous variable will be treated as independent of each other, otherwise they will be treated as dependent.
slices	Slicing strategy that maximize dynamic slicing statistic based on currently ranked vector x. It will be reported if slice is true. Each row stands for a slice. Each column except the last one stands for the number of observations take each value in each slice. The last column is the number of observations in each slice *i.e.*, the sum of the first column to the *k*th column.

References

Jiang, B., Ye, C. and Liu, J.S. Non-parametric *K*-sample tests via dynamic slicing. *Journal of the American Statistical Association*, 110(510): 642-653, 2015.

See Also

ds_eqp_k.

Examples

```
n <- 100
mu <- 0.5
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep("1", n), rep("2", n))
x <- relabel(x)
x <- x[order(y)]
xdim <- max(x) + 1
lambda <- 1.0
dsres <- ds_k(x, xdim, lambda, slice = TRUE)
```

---

ds_test *Hypothesis testing via dynamic slicing*

Description

Perform a one- or *K*-sample ($K > 1$) hypothesis testing via dynamic slicing.

Usage

```
ds_test(y, x, ..., type = c("ds", "eqp"), lambda = 1, alpha = 1, rounds = 0)
```

Arguments

y	A numeric vector of data values.
x	Either an integer vector of data values, from 0 to $K-1$, or a character string naming a cumulative distribution function or an actual cumulative distribution function such as pnorm. Only continuous CDFs are valid.
...	Parameters of the distribution specified (as a character string) by x.
type	Methods applied for dynamic slicing. "ds" (default) stands for original dynamic slicing scheme. "eqp" stands for dynamic slicing scheme with $n^{1/2}$-resolution (for *K*-sample test, $K > 1$) or n-resolution (for one-sample test).
lambda	Penalty for introducing an additional slice, which is used to avoid making too many slices. It corresponds to the type I error under the scenario that the two variables are independent. lambda should be greater than 0.
alpha	Penalty required for "ds" type in one-sample test. It penalizes both the width and the number of slices to avoid too many slices and degenerate slice (interval). alpha should be greater than 1.
rounds	Number of permutations for estimating empirical *p*-value.

Details

If x is an integer vector, ds_test performs *K*-sample test ($K > 1$).

Under this scenario, suppose that there are observations y drawn from some *continuous* populations. Let x be a vector that stores values of indicator of samples from different populations, *i.e.*, x has values $0, 1, \ldots, K-1$. The null hypothesis is that these populations have the same distribution.

If x is a character string naming a continuous (cumulative) distribution function, ds_test performs one-sample test with the null hypothesis that the distribution function which generated y is distribution x with parameters specified by .... The parameters specified in ... must be pre-specified and not estimated from the data.

Only empirical *p*-values are available by specifying the value of parameter rounds, the number of permutation. lambda and alpha (for one-sample test with type "ds") contributes to *p*-value.

The procedure of choosing parameter lambda was described in Jiang, Ye & Liu (2015). Refer to http://www.people.fas.harvard.edu/~junliu/DS/lambda-table.html for the empirical relationship of lambda, sample size and type I error.

Value

A list with class "htest" containing the following components:

statistic	The value of the dynamic slicing statistic.
p.value	The *p*-value of the test.
alternative	A character string describing the alternative hypothesis.
method	A character string indicating what type of test was performed.
data.name	A character string giving the name(s) of the data.
slices	Slicing strategy that maximize dynamic slicing statistic in *K*-sample test. Each row stands for a slice. Each column except the last one stands for the number of observations take each value in each slice. The last column is the number of observations in each slice *i.e.*, the sum of the first column to the *k*th column.

References

Jiang, B., Ye, C. and Liu, J.S. Non-parametric *K*-sample tests via dynamic slicing. *Journal of the American Statistical Association*, 110(510): 642-653, 2015.

Examples

```
##  One-sample test
n <- 100
mu <- 0.5
y <- rnorm(n, mu, 1)
lambda <- 1.0
alpha <- 1.0
dsres <- ds_test(y, "pnorm", 0, 1, lambda = 1, alpha = 1, rounds = 100)
dsres <- ds_test(y, "pnorm", 0, 1, type = "ds", lambda = 1, alpha = 1)
dsres <- ds_test(y, "pnorm", 0, 1, type = "eqp", lambda = 1, rounds = 100)
dsres <- ds_test(y, "pnorm", 0, 1, type = "eqp", lambda = 1)

##  K-sample test
n <- 100
mu <- 0.5
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))

##  generate x in this way:
x <- c(rep(0, n), rep(1, n))
x <- as.integer(x)

##  or in this way:
x <- c(rep("G1", n), rep("G2", n))
x <- relabel(x)

lambda <- 1.0
dsres <- ds_test(y, x, lambda = 1, rounds = 100)
dsres <- ds_test(y, x, type = "eqp", lambda = 1, rounds = 100)
```

`export_res` *Export gene set analysis result*

Description

Export gene set analysis result.

Usage

```
export_res(ds_gsa_obj, file = "", ..., cutoff = 1, decreasing = FALSE,
           type = c("name", "size", "DS", "p-value", "FDR", "slice"))
```

Arguments

ds_gsa_obj	Object returned by ds_gsa function.
file	Either a character string naming a file or a connection open for writing. "" indicates output to the console.
...	Parameters of write.table.
cutoff	threshold for selecting gene set analysis result according to parameter type.
decreasing	Ranking results decreasingly (default) or ascendingly.
type	Opitons for ranking results by gene set name ("name"), gene set size ("size"), value of dynamic slicing statistic ("DS"), *p*-value ("p-value"), false discovery rate ("FDR") or number of slices ("slice").

Details

The usage of export_res is similar to write.table.

See Also

The 'R Data Import/Export' manual.

Examples

```
## Not run:
ds_gsa_obj <- ds_gsa(expdat, geneset, label, fc, lambda = 1.2, bycol = TRUE,
                  minsize = 15, maxsize = 500, randseed = 11235, rounds = 100)
export_res(ds_gsa_obj, "ds_gsa_res.txt", sep = "\t", type = "DS", cutoff = 0,
           row.names = F, col.names = T, quote = F, append = F)

## End(Not run)
```

---

gsa_exp *Gene expression matrix in gene set analysis*

Description

P53 NCI-60 data set provided by Subramanian *et al*., (2005). A gene expression matrix whose rows correspond to genes and column correspond to samples.

Usage

```
data(gsa_exp)
```

Format

A matrix.

Source

http://www.broadinstitute.org/gsea

References

Subramanian, A., Tamayo, P., Mootha, V. K., *et al*. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 2005, 102(43): 15545-15550.

Examples

```
data(gsa_exp)
```

---

gsa_label *Sample labels in gene set analysis*

Description

P53 NCI-60 data set provided by Subramanian *et al*., (2005). A list with phenotypes vector and a vector of sample label values.

Usage

```
data(gsa_label)
```

Format

A list on the following 2 variables.

`pheotype` a character vector contains two genotypes.

`value` a numeric vector contains sample label 0 and 1, where 0 and 1 stands for "MUT" and "WT", respectively.

Source

http://www.broadinstitute.org/gsea

References

Subramanian, A., Tamayo, P., Mootha, V. K., *et al*. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 2005, 102(43): 15545-15550.

Examples

```
data(gsa_label)
```

gsa_set *Gene set list in gene set analysis*

Description

Gene set provided by Subramanian *et al*., (2005). A list with gene set vector, gene set description vector and a list whose elements are vector of genes in each gene set.

Usage

```
data(gsa_set)
```

Format

A list with three elements: gene set name, gene set description and genes in gene set.

`set_name` a character vector of gene set name.

`set_description` a character vector of gene set description.

`set_description` a list whose elements are vector. Each of them contains genes in each gene set.

Source

http://www.broadinstitute.org/gsea

References

Subramanian, A., Tamayo, P., Mootha, V. K., *et al*. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 2005, 102(43): 15545-15550.

Examples

```
data(gsa_set)
```

load_cls *Load phenotype file*

Description

Load phenotype file from .cls file

Usage

```
load_cls(file)
```

Arguments

file	Name of .cls file which contains phenotypes of samples, separared by spaces.

Value

A list with components:

pheotype	Pheotype of samples
value	A vector with encoded value of pheotypes. If there is *k* pheotypes, it takes values $0, 1, \ldots, k-1$.

Examples

```
# Pheotype files are available after registration at Broad institute GSEA website
# http://www.broadinstitute.org/gsea

## Not run:
filename <- "P53.cls"
label <- load_cls(filename)

## End(Not run)
```

---

load_gct *Load gene expression file*

Description

Load gene expression data from .gct file

Usage

```
load_gct(file)
```

Arguments

file	Name of .gct file which contains gene expression data. Should be a tab-separated text file. The first row is version and the second is the dimension of expression matrix. There is an expression matrix from the third row to the end. The third row is column name of expression matrix. The first column is gene symbol and the second is the description of gene. For the remaining rows and columns, each row is a gene and each column is a sample.

Value

A matrix with row names and column names.

Examples

```
# Gene expression files are available after registration at Broad institute GSEA website
# http://www.broadinstitute.org/gsea

## Not run:
filename <- "P53.gct"
expdat <- load_gct(filename)

## End(Not run)
```

`load_gmt` *Load gene set file*

Description

Load gene set from .gmt file

Usage

```
load_gmt(file)
```

Arguments

`file`	Name of .gmt file which contains gene sets. Should be a tab-separated text file. Each row is a gene set. The first column is gene set name and the second is its description. Remaining columns are gene symbols of genes in this set.

Value

A list with components:

`set_name`	Vector of gene set names.
`set_description`	Vector of gene set descriptions.
`gene_symbol`	List of gene symbols in each gene set.

Examples

```
# Gene set files are available after registration at Broad institute GSEA website
# http://www.broadinstitute.org/gsea

## Not run:
filename <- "C2.gmt"
geneset <- load_gmt(filename)

## End(Not run)
```

rank_by_s2n *Ranking genes by signal to noise ratio*

Description

Ranking genes by signal to noise ratio according to their expression data.

Usage

```
rank_by_s2n(expmat, label)
```

Arguments

expmat	A matrix of gene expression data. Each row is a gene and each column is a sample.
label	An integer vector of encoded pheotypes. Its value is 0 and 1. Its length should match the column number of expression matrix.

Value

A vector of rank of each gene according to signal to noise ratio.

Examples

```
expdat <- matrix(rnorm(500), nrow = 25, ncol = 20)
label <- rep(c(0, 1), 10)
ranklist <- rank_by_s2n(expdat, label)
```

relabel *Reassigning values of categorical variable*

Description

Reassigning values of categorical variable. It is used for generating legal value of categorical variable before applying dynamic slicing.

Usage

```
relabel(x)
```

Arguments

x	A vector of data values.

Value

An integer vector with values range from 0 to k (k > 0).

See Also

ds_test.

Examples

```
n <- 10
x <- c(rep("G1", n), rep("G2", n))
x <- relabel(x)

x <- c(rep(4, n), rep(5, n), rep(NA, n))
x <- relabel(x)
```

---

slice_show *Show the slicing result*

Description

Showing slicing result and plotting counts of observations in each slice.

Usage

```
slice_show(slices_obj, main="Counts in each slice", xlab="Slices", ylab="Percentage")
```

Arguments

slices_obj	A matrix stores slicing strategy. It is a component of object returned by function dslice_k or dslice_eqp_k.
main	An overall title for the plo
xlab	A title for the x axis
ylab	A title for the y axis

Value

A "ggplot" object which illustrates details of slicing.

See Also

ds_k, ds_eqp_k.

Examples

```
n <- 100
mu <- 0.5
y <- c(rnorm(n, -mu, 1), rnorm(n, mu, 1))
x <- c(rep(0, n), rep(1, n))
x <- x[order(y)]
xdim <- max(x) + 1
lambda <- 1.0
```

```
dsres <- ds_k(x, xdim, lambda, slice = TRUE)

ds_show <- slice_show(dsres$slices)
```

Index