Package 'iTOP'

July 22, 2025

Type Package

Title Inferring the Topology of Omics Data

Version 1.0.2

Author Nanne Aben

Maintainer Nanne Aben <nanne.aben@gmail.com>

Description Infers a topology of relationships between different datasets, such as multi-omics and phenotypic data recorded on the same samples. We based this methodology on the RV coefficient (Robert & Escoufier, 1976, <doi:10.2307/2347233>), a measure of matrix correlation, which we have extended for partial matrix correlations and binary data (Aben et al., 2018, <doi:10.1101/293993>).

Imports Matrix, corpcor

License GPL-2

Encoding UTF-8

LazyData true

Suggests knitr, rmarkdown, NMF, pcalg, Rgraphviz

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-06-13 09:16:06 UTC

Contents

`bootstrap.config.matrices`

Performing a single bootstrap

Description

Helper function for run.bootstraps(). It's unlikely you'll ever need to run this function directly.

Usage

```
bootstrap.config.matrices(config_matrices)
```

Arguments

`config_matrices`

The result from compute.config.matrices().

Value

An n x n matrix of RV coefficients for the bootstrapped data, where n is the number of datasets.

`compute.config.matrices`

Compute configuration matrices

Description

Given a list of n data matrices (corresponding to n datasets), this function computes the configuration matrix for each of these configuration matrices. By default inner product similarity is used, but other similarity (such as Jaccard similarity for binary data) can also be used (see the vignette 'A quick introduction to iTOP' for more information). In addition, the configuration matrices can be centered and prepared for use with the modified RV coefficient, both of which we will briefly explain here.

Usage

```
compute.config.matrices(data, similarity_fun = inner.product, center = TRUE,
  mod.rv = TRUE)
```

Arguments

data	List of datasets.
similarity_fun	Either a function pointer to the similarity function to be used for all datasets; or a list of function pointers, if different similarity functions need to be used for different datasets (default=inner.product).
center	Either a boolean indicating whether centering should be used for all datasets; or a list of booleans, if centering should be used for some datasets but not all of them (default=TRUE).
mod.rv	Either a boolean indicating whether the modified RV coefficient should be used for all datasets; or a list of booleans, if the modified RV should be used for some datasets but not all of them (default=TRUE).

Details

The RV coefficient often results in values very close to one when both datasets are not centered around zero, even for orthogonal data. For inner product similarity and Jaccard similarity, we recommend using centering. However, for some other similarity measures, centering may not be beneficial (for example, because the measure itself is already centered, such as in the case of Pearson correlation). For more information on centering of binary (and other non-continuous) data, for which we used kernel centering of the configuration matrix, we refer to our manuscript: Aben et al., 2018, doi.org/10.1101/293993.

The modified RV coefficient was proposed for high-dimensional data, as the regular RV coefficient would result in values close to one even for orthogonal data. We recommend always using the modified RV coefficient.

Value

A list of n configuration matrices, where n is the number of datasets.

Examples

```
set.seed(2)
n = 100
p = 100
x1 = matrix(rnorm(n*p), n, p)
x2 = x1 + matrix(rnorm(n*p), n, p)
x3 = x2 + matrix(rnorm(n*p), n, p)
data = list(x1=x1, x2=x2, x3=x3)
config_matrices = compute.config.matrices(data)
cors = rv.cor.matrix(config_matrices)
```

`compute.config.matrix` *Computes a configuration matrix*

Description

Given a data matrix, this function computes the configuration matrix for the corresponding dataset. You'll typically won't need to call this function directly, but should use compute.config.matrices() instead, as it will make determining partial RV coefficients, p-values and confidence intervals easier later on.

Usage

```
compute.config.matrix(x, similarity_fun = inner.product, center = TRUE,
  mod.rv = TRUE)
```

Arguments

`x`	Data matrix.
`similarity_fun`	A function pointer to the similarity function to be used (default=inner.product).
`center`	A boolean indicating whether centering should be used (default=TRUE).
`mod.rv`	A boolean indicating whether the modified RV coefficient should be used (default=TRUE).

Value

A configuration matrix.

Examples

```
set.seed(2)
n = 100
p = 100
x1 = matrix(rnorm(n*p), n, p)
x2 = x1 + matrix(rnorm(n*p), n, p)
S1 = compute.config.matrix(x1)
S2 = compute.config.matrix(x1)
rv.coef(S1, S2)
```

inner.product *Inner product similarity.*

Description

Computes the inner product between x and y.

Usage

```
inner.product(x, y)
```

Arguments

x	A vector of numbers.
y	A vector of numbers.

Value

The inner product similarity between x and y.

Examples

```
set.seed(2)
n = 100
x = rnorm(n)
y = rnorm(n)
inner.product(x, y)
```

intersect.samples *Intersect samples between datasets.*

Description

In order to make all datasets comparable, we have to make sure they describe the same set of samples. This function takes a list of datasets (i.e. data matrices), takes the intersect of all rownames, and returns a list of datasets with only those samples.

Usage

```
intersect.samples(data)
```

Arguments

data	A list of data matrices. The data matrices need to have rownames.

Value

A list with of data matrices, all with the same set of samples.

Examples

```
set.seed(2)
n = 100
p = 100
x1 = matrix(rnorm(n*p), n, p)
x2 = matrix(rnorm(n*p), n, p)
rownames(x1) = rownames(x2) = paste0("X",1:n)
data = list(x1=x1[1:90,], x2=x2[10:100,])
data = intersect.samples(data)
```

`jaccard` *Jaccard similarity.*

Description

Computes the Jaccard similarity between x and y. When both x and y only contain zeroes, the Jaccard similarity it not defined. This function returns zero for that specific case.

Usage

```
jaccard(x, y)
```

Arguments

x	A vector of zeroes and ones.
y	A vector of zeroes and ones.

Value

The Jaccard similarity between x and y.

Examples

```
set.seed(2)
n = 100
x = rbinom(n, 1, 0.5)
y = rbinom(n, 1, 0.5)
jaccard(x, y)
```

permute.config.matrices

Performing a permutation

Description

Helper function for run.permutations(). It's unlikely you'll ever need to run this function directly.

Usage

```
permute.config.matrices(config_matrices)
```

Arguments

config_matrices	The result from compute.config.matrices().

Value

An n x n matrix of RV coefficients for the permutated data, where n is the number of datasets.

process.custom.config.matrix

Process a custom configuration matrix.

Description

This function can be used to process a custom-made configuration matrix (i.e. similarity matrix) for use with the RV coefficient. The function can perform two tasks: centering and preparation for the modified RV coefficient, both of which we will briefly explain here.

Usage

```
process.custom.config.matrix(S, center = TRUE, mod.rv = TRUE)
```

Arguments

S	A configuration matrix.
center	Should the configuration matrix be centered using kernel centering?
mod.rv	Should the configuration matrix be prepared for the modified RV coefficient?

Details

The RV coefficient often results in values very close to one when both datasets are not centered around zero, even for orthogonal data. For inner product similarity and Jaccard similarity, we recommend using centering. However, for some other similarity measures, centering may not be beneficial (for example, because the measure itself is already centered, such as in the case of Pearson correlation). For more information on centering of binary (and other non-continuous) data, for which we used kernel centering of the configuration matrix, we refer to our manuscript: Aben et al., 2018, doi.org/10.1101/293993.

The modified RV coefficient was proposed for high-dimensional data, as the regular RV coefficient would result in values close to one even for orthogonal data. We recommend always using the modified RV coefficient.

Value

The processed configuration matrix.

Examples

```
set.seed(2)
n = 100
p = 100
x = matrix(rnorm(n*p)+10, n, p)
S = x%*%t(x)
S_dash = process.custom.config.matrix(S, center=TRUE, mod.rv=TRUE)
```

---

run.bootstraps *Bootstrapping procedure*

Description

Performs a bootstrapping procedure. The result from this function can be used with rv.conf.interval() to determine confidence intervals. By decoupling this into two functions, you don't have to redo the bootstrapping for every confidence interval, hence increasing the runtime speed.

Usage

```
run.bootstraps(config_matrices, nboots = 1000)
```

Arguments

config_matrices	The result from compute.config.matrices().
nboots	The number of bootstraps to perform (default=1000).

Value

An n x n x nboots array of RV coefficients for the bootstrapped data, where n is the number of datasets.

Examples

```
set.seed(2)
n = 100
p = 100
x1 = matrix(rnorm(n*p), n, p)
x2 = x1 + matrix(rnorm(n*p), n, p)
x3 = x2 + matrix(rnorm(n*p), n, p)
data = list(x1=x1, x2=x2, x3=x3)
config_matrices = compute.config.matrices(data)
cors_boot = run.bootstraps(config_matrices, nboots=1000)
rv.conf.interval(cors_boot, "x1", "x3", "x2")
```

---

run.permutations *Permutations for significance testing*

Description

Performs a permutations for significance testing. The result from this function can be used with rv.pval() to determine a p-value. By decoupling this into two functions, you don't have to redo the permutations for every p-value, hence increasing the runtime speed.

Usage

```
run.permutations(config_matrices, nperm = 1000)
```

Arguments

config_matrices	The result from compute.config.matrices().
nperm	The number of permutations to perform (default=1000).

Value

An n x n x nperms array of RV coefficients for the permutated data, where n is the number of datasets.

Examples

```
set.seed(2)
n = 100
p = 100
x1 = matrix(rnorm(n*p), n, p)
x2 = x1 + matrix(rnorm(n*p), n, p)
x3 = x2 + matrix(rnorm(n*p), n, p)
data = list(x1=x1, x2=x2, x3=x3)
config_matrices = compute.config.matrices(data)
cors = rv.cor.matrix(config_matrices)
cors_perm = run.permutations(config_matrices, nperm=1000)
rv.pval(cors, cors_perm, "x1", "x3", "x2")
```

rv.coef *Computes the RV coefficient*

Description

Computes the RV coefficient between dataset 1 and dataset 2. You'll typically won't need to call this function directly, but should use rv.cor.matrix() instead, as it will make determining partial RV coefficients, p-values and confidence intervals easier later on.

Usage

```
rv.coef(S1, S2)
```

Arguments

S1	Configuration matrix corresponding to dataset 1
S2	Configuration matrix corresponding to dataset 2

Value

The RV coefficient between dataset 1 and dataset 2

Examples

```
set.seed(2)
n = 100
p = 100
x1 = matrix(rnorm(n*p), n, p)
x2 = x1 + matrix(rnorm(n*p), n, p)
S1 = compute.config.matrix(x1)
S2 = compute.config.matrix(x1)
rv.coef(S1, S2)
```

rv.conf.interval *Determining a confidence interval for the (partial) RV coefficient*

Description

This function uses a bootstrapping procedure to determine a confidence interval for the RV coefficient RV(a, b) or the partial RV coefficient RV(a, b | set).

Usage

```
rv.conf.interval(cors_boot, a, b, set = NULL, conf = 0.95)
```

Arguments

cors_boot	The result from run.bootstraps().
a	Either an index or a string to identify dataset a.
b	Either an index or a string to identify dataset b.
set	Optional parameter to define the datasets that need to be partialized for. If set consists of one dataset, then provide an index or a string to identify set. If set consists of multiple datasets, then provide a vector of indices or a vector of strings.
conf	The size of the confidence interval (default=0.95).

Value

The confidence interval.

Examples

```
set.seed(2)
n = 100
p = 100
x1 = matrix(rnorm(n*p), n, p)
x2 = x1 + matrix(rnorm(n*p), n, p)
x3 = x2 + matrix(rnorm(n*p), n, p)
data = list(x1=x1, x2=x2, x3=x3)
config_matrices = compute.config.matrices(data)
cors_boot = run.bootstraps(config_matrices, nboots=1000)
rv.conf.interval(cors_boot, "x1", "x3", "x2")
```

---

rv.cor.matrix *A correlation matrix of RV coefficients*

Description

Given a list of n configuration matrices (corresponding to n datasets), this function computes an n x n matrix of pairwise RV coefficients.

Usage

```
rv.cor.matrix(config_matrices)
```

Arguments

config_matrices	The result from compute.config.matrices().

Value

An n x n matrix of pairwise RV coefficients, where n is the number of datasets.

Examples

```
set.seed(2)
n = 100
p = 100
x1 = matrix(rnorm(n*p), n, p)
x2 = x1 + matrix(rnorm(n*p), n, p)
x3 = x2 + matrix(rnorm(n*p), n, p)
data = list(x1=x1, x2=x2, x3=x3)
config_matrices = compute.config.matrices(data)
cors = rv.cor.matrix(config_matrices)
```

---

rv.link.significance *Wrapper function to determine significance in the PC algorithm*

Description

This function is a wrapper function around rv.pval(), such that it can easily be used with pc() from the pcalg package. If you have trouble installing the pcalg package, have a look at our vignette 'A quick start to iTOP'.

Usage

```
rv.link.significance(a, b, set, suffStat)
```

Arguments

a	Either an index or a string to identify dataset a.
b	Either an index or a string to identify dataset b.
set	Datasets that need to be partialized for. Set to NULL if there are none (i.e. if you're computing a regular, non-partial RV). If set consists of one dataset, then provide an index or a string to identify set. If set consists of multiple datasets, then provide a vector of indices or a vector of strings.
suffStat	A named list with two items: cors, which is the result from rv.cor.matrix(); and cors_perm, which is the result from run.permutations().

Value

The p-value.

Examples

```
set.seed(2)
n = 100
p = 100
x1 = matrix(rnorm(n*p), n, p)
x2 = x1 + matrix(rnorm(n*p), n, p)
x3 = x2 + matrix(rnorm(n*p), n, p)
```

```
data = list(x1=x1, x2=x2, x3=x3)
config_matrices = compute.config.matrices(data)
cors = rv.cor.matrix(config_matrices)
cors_perm = run.permutations(config_matrices, nperm=1000)

## Not run:
library(pcalg)
suffStat = list(cors=cors, cors_perm=cors_perm)
pc.fit = pc(suffStat=suffStat, indepTest=rv.link.significance, labels=names(data),
            alpha=0.05, conservative=TRUE, solve.confl=TRUE)
plot(pc.fit, main="")
## End(Not run)
```

`rv.pcor` *Determining a (partial) RV coefficient*

Description

Determines the RV coefficient RV(a, b) or the partial RV coefficient RV(a, b | set).

Usage

```
rv.pcor(cors, a, b, set = NULL)
```

Arguments

`cors`	The result from rv.cor.matrix().
`a`	Either an index or a string to identify dataset a.
`b`	Either an index or a string to identify dataset b.
`set`	Optional parameter to define the datasets that need to be partialized for. If set consists of one dataset, then provide an index or a string to identify set. If set consists of multiple datasets, then provide a vector of indices or a vector of strings.

Value

The (partial) RV coefficient.

Examples

```
set.seed(2)
n = 100
p = 100
x1 = matrix(rnorm(n*p), n, p)
x2 = x1 + matrix(rnorm(n*p), n, p)
x3 = x2 + matrix(rnorm(n*p), n, p)
data = list(x1=x1, x2=x2, x3=x3)
config_matrices = compute.config.matrices(data)
cors = rv.cor.matrix(config_matrices)
rv.pcor(cors, "x1", "x3", "x2")
```

rv.pval *Determining a p-value the (partial) RV coefficient*

Description

This function uses a permutation test to determine a p-value for the RV coefficient RV(a, b) or the partial RV coefficient RV(a, b | set).

Usage

```
rv.pval(cors, cors_perm, a, b, set = NULL)
```

Arguments

cors	The result from rv.cor.matrix().
cors_perm	The result from run.permutations().
a	Either an index or a string to identify dataset a.
b	Either an index or a string to identify dataset b.
set	Optional parameter to define the datasets that need to be partialized for. If set consists of one dataset, then provide an index or a string to identify set. If set consists of multiple datasets, then provide a vector of indices or a vector of strings.

Value

The p-value.

Examples

```
set.seed(2)
n = 100
p = 100
x1 = matrix(rnorm(n*p), n, p)
x2 = x1 + matrix(rnorm(n*p), n, p)
x3 = x2 + matrix(rnorm(n*p), n, p)
data = list(x1=x1, x2=x2, x3=x3)
config_matrices = compute.config.matrices(data)
cors = rv.cor.matrix(config_matrices)
cors_perm = run.permutations(config_matrices, nperm=1000)
rv.pval(cors, cors_perm, "x1", "x3", "x2")
```

Index