Package 'PCIT'

February 19, 2015

Version 1.5-3

Date 2015-02-11

Title Partial Correlation Coefficient with Information Theory

Author Nathan S. Watson-Haigh

Maintainer Nathan S. Watson-Haigh <nathan.haigh@acpfg.com.au>

Description Apply Partial Correlation coefficient with Information Theory (PCIT) to a correlation matrix.
The PCIT algorithm identifies meaningful correlations to define edges in a weighted network. The algorithm can be applied to any correlation-based network including but not limited to gene co-expression networks.
To reduce compute time by making use of multiple compute cores, simply run PCIT on a computer with has multiple cores and also has the Rmpi package installed. PCIT will then auto-detect the multicore environment and run in parallel mode without the need to rewrite your scripts. This makes scripts, using PCIT, portable across single core (or no Rmpi package installed) computers which will run in serial mode and multicore (with Rmpi package installed) computers which will run in parallel mode.

Type Package

License GPL-3

Suggests Rmpi

URL http://dx.doi.org/10.1093/bioinformatics/btn482
http://bioinformatics.oxfordjournals.org/cgi/content/abstract/26/3/411

LazyLoad yes

Depends R (>= 2.10)

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-02-16 17:39:23

R topics documented:

`PCIT-package` *Partial Correlation coefficient with Information Theory (PCIT)*

Description

This package provides the necessary functions for performing the Partial Correlation coefficient with Information Theory (PCIT) algorithm developed by Reverter and Chan (2008). The PCIT algorithm identifies meaningful correlations to define edges in a weighted network.

The algorithm can be applied to any correlation-based network including but not limited to gene co-expression networks.

Details

Package:	PCIT
Type:	Package
Version:	1.04-3
Date:	2011-05-11
License:	GPL-3
LazyLoad:	yes

Author(s)

Nathan S. Watson-Haigh Maintainer: Nathan S. Watson-Haigh <nathan.watson-haigh@awri.com.au>

References

A. Reverter and E.K.F. Chan. (2008) Combining partial correlation and an information theory approach to the reversed engineering of gene co-expression networks. Bioinformatics. **24**(21), 2491-2497.

`clusteringCoefficient` *Calculate the clustering coefficient*

Description

Calculate the clustering coefficient for an adjacency matrix. By default, the local clustering coefficient is calculated.

Usage

```
clusteringCoefficient(adj, FUN='localClusteringCoefficient', ...)
```

Arguments

`adj`	- An adjacency matrix. Calculating the clustering coefficient only makes sense if some connections are zero i.e. no connection.
`FUN`	- The function for calculating the clustering coefficient.
`...`	- Arguments to pass to FUN

Value

The clustering coefficient(s) for the adjacency matrix.

Author(s)

Nathan S. Watson-Haigh

See Also

`localClusteringCoefficient` `clusteringCoefficientPercent`

Examples

```
data(PCIT)
m <- m[1:200,1:200]        # just use a small subset of the data
result <- pcit(m)
m[idx(result)] <- 0

clusteringCoefficient(m)
```

clusteringCoefficientPercent *Calculate the clustering coefficient as a percentage*

Description

Given an adjacency matrix, calculate the clustering coefficient as a percentage of non-zero adjacencies.

Usage

```
clusteringCoefficientPercent(adj)
```

Arguments

adj	- An adjacency matrix. Calculating the clustering coefficient percentage only makes sense if some connections are zero i.e. no connection.

Value

A numerical between 0 and 100.

Author(s)

Nathan S. Watson-Haigh

See Also

clusteringCoefficient

Examples

```
data(PCIT)
m <- m[1:200,1:200]       # just use a small subset of the data
result <- pcit(m)
m[idx(result)] <- 0

clusteringCoefficientPercent(m)
```

data *Demo Data*

Description

Example/mock correlation data set for use as input for running the demo.

Usage

```
data(PCIT)
```

Author(s)

Nathan S. Watson-Haigh

Examples

```
data(PCIT)
m <- m[1:200,1:200]       # just use a small subset of the data
result <- pcit(m)
idx <- idx(result)
```

defineTasks *Define a list of tasks for slave CPUs*

Description

This function defines a list of PCIT tasks to be undertaken by slave CPUs in a parallel environment. The set of calculations performed for each trio of genes are entirely independent. Workload is distributed evenly among the slaves in order to maximise speed by assigning sets of gene trios to a task. The task is then carried out by a slave CPU. This assignment of gene trios into sets is based on which gene is in the first position in a given trio of genes. For example, if we have 5 genes (A, B, C, D and E) we have a set containing 6 gene trios with A in the first position (ABC, ABD, ABE, ACD, ACE and ADE), a set containing 3 gene trios with B in the first position (BCD, BCE and BDE) and a set containing 1 gene trio with C in the first position (CDE) giving a total of 10 gene trios (given by C(n,3), where n is the total number of genes) assigned to 1 of 3 sets. The computational workload for each trio is the same, thus the workload for a set of trios is directly proportional to the number of trios it contains, C(n-m,k-1), Where n is the total number of genes and the gene constrained in the first position of a gene trio is in the mth row/column in the correlation matrix and takes values in the interval [1,n-2].

By default, the number of tasks to be created is equal to the number for slave CPUs. Therefore, multiple gene trio sets are assigned to each task but in such a way as to balance the work load. Since gene trio sets created with small values of m have many more gene trio members than those created with larger values of m. A plot can be produced which shows the amount of work taken for a gene trio set constrained with the mth gene in the first position. The plot also shows how the cumlative work for all n-2 gene trio sets is divided into nSlave*tasksPerSlave equally sized tasks, so they can be completed in approximately the same amount of time.

Usage

```
defineTasks(n, nSlaves, tasksPerSlave = 1, plot = FALSE)
```

Arguments

n	- The total number of genes.
nSlaves	- The number of slave CPUs.
tasksPerSlave	- The number of tasks to create for each slave CPU.
plot	- A boolean as to whether to generate a plot showing the total "work" to be carried out by each task.

Author(s)

Nathan S. Watson-Haigh

Examples

```
defineTasks(n=100, nSlaves=5, plot=TRUE)
```

---

getEdgeList *Converts an adjacency matrix into edge list representation*

Description

Given an adjacency matrix, converts it into edge list representation. This edge list can be written to a file for easy import into other software such as cytoscape.

Only the upper triangle is returned as it is assumed the matrix is symmetric. The edge list is returned as a data frame with 3 columns: 'From', 'To' and 'Weight'.

Usage

```
getEdgeList(m, rm.zero=TRUE)
```

Arguments

m	- An adjacency matrix.
rm.zero	- A boolean to indicate whether zero weight edges should be excluded from the edge list.

Author(s)

Nathan S. Watson-Haigh

Examples

```
data(PCIT)
m <- m[1:200,1:200]        # just use a small subset of the data

el <- getEdgeList(m)

# modify the edge list to include some useful attributes for cytoscape
el$sign[el$Weight<0] <- '-'
el$sign[el$Weight>0] <- '+'
el$Weight <- abs(el$Weight)
# write the edge list stuff to a file suitable for import into cytoscape
write.table(el, file="el.txt", row.names=FALSE, col.names=TRUE, sep="\t",
quote=FALSE)
```

---

idx *Get indicies for significant edges*

Description

Get the indicies for the significant edges in a network.

Usage

```
idx(result)
```

Arguments

result	- A result object returned from pcit()

Value

Linear indices are returned for those correlations found to be significant.

Author(s)

Nathan S. Watson-Haigh

Examples

```
data(PCIT)
m <- m[1:200,1:200]        # just use a small subset of the data
result <- pcit(m)

idx <- idx(result)
```

idxInvert *Invert linear indices from a matrix*

Description

Given a matrix from which which a subset of linear indices were obtained, invert those indices.

Usage

```
idxInvert(m, idx)
```

Arguments

m	- A matrix from which idx is a subset of linear indices, OR the number of rows/columns from such a matirx
idx	- A vector containing a subset of linear indices from the matrix m

Author(s)

Nathan S. Watson-Haigh

Examples

```
m <- matrix(1, 5, 5)
diag(m) <- 0
m
idx <- which(m==0)
idx
idxInvert(m, idx)
idxInvert(5, idx)
```

---

localClusteringCoefficient
Calculate the local clustering coefficient

Description

Calculate the local clustering coefficient for each node in an adjacency matrix. The clustering coefficient is defined as the proportion of existing connections from the total possible (Watts and Strogatz, 1998).

Usage

```
localClusteringCoefficient(adj)
```

Arguments

adj	- An adjacency matrix. Calculating the clustering coefficient only makes sense if some connections are zero i.e. no connection.

Value

A vector of local clustering coefficients for each node/gene of the adjacency matrix.

Author(s)

Nathan S. Watson-Haigh

References

D.J. Watts and S.H. Strogatz. (1998) Collective dynamics of 'small-world' networks. Nature. 393(6684). 440-442.

See Also

clusteringCoefficient

Examples

```
data(PCIT)
m <- m[1:200,1:200]        # just use a small subset of the data
result <- pcit(m)
m[idx(result)] <- 0

localClusteringCoefficient(m)
```

---

maxMatrixSize *Calculate the maximum correlation matrix size which PCIT can handle*

Description

This function attempts to determine the maximum sized correlation matrix which can be handled by PCIT given an amount of computer memory for the serial implementation.

Usage

```
maxMatrixSize(ram, units=c("MB", "bytes", "KB", "GB", "TB"), nCopies=3)
```

Arguments

ram	- The amount of RAM memory available
units	- The units in which RAM was specified
nCopies	- The maximum number of copies of the correlation matrix which PCIT holds at any one time

Author(s)

Nathan S. Watson-Haigh

See Also

pcitMemoryRequirement

Examples

```
maxMatrixSize(1, "GB")
maxMatrixSize(512, "MB")
```

---

pcit *Apply the PCIT algorithm*

Description

Given a correlation matrix the PCIT algorithm (Reverter & Chan 2008) is applied to identify significant correlations. If a parallel environment running Rmpi is detected, a parallel implementation will be run unless force.serial=TRUE

Usage

```
pcit(m, force.serial=FALSE, force.parallel=FALSE, nslaves=NULL,
verbose=getOption("verbose"),
tol.type=c("mean", "min", "max", "median"),
pass.type=c("file", "memory", "db"))
```

Arguments

m	- A correlation matrix.
force.serial	- A boolean to indicate if the serial implementation of PCIT should be forced.
force.parallel	- A boolean to indicate if the parallel implementation of PCIT should be forced.
nslaves	- The number of slaves to spawn. By default, as many slaves as possible are spawned. UNTESTED OPTION.
verbose	- A boolean to indicate if verbose output should be used.
tol.type	- The type of tolerance measure to be used in PCIT. Current options are "mean", "min" and "max".
pass.type	- The type of approach used to pass the correlation matrix from the master CPU to the slave CPUs. Current options are "file", "memory" and "db".

Value

Linear indices are returned for those correlations found to be significant.

Author(s)

Nathan S. Watson-Haigh

References

Reverter, A. & Chan, E.K., 2008. Combining partial correlation and an information theory approach to the reversed-engineering of gene co-expression networks. Bioinformatics, btn482.

Examples

```
data(PCIT)
m <- m[1:200,1:200]       # just use a small subset of the data

result <- pcit(m)
```

---

pcitMemoryRequirement *Calculate the memory requirement for running PCIT*

Description

This function attempts to determine the amount of computer memory that would be required to run PCIT of a given correlation matrix.

Usage

```
pcitMemoryRequirement(m, units=c("MB", "bytes", "KB", "GB", "TB"),
nCopies=3)
```

Arguments

m	- A correlation matrix on which PCIT may be run, OR the number of rows/columns from such a matirx
units	- The units of RAM memory to be use for the returned value
nCopies	- The maximum number of copies of the correlation matrix which PCIT holds at any one time

Author(s)

Nathan S. Watson-Haigh

See Also

maxMatrixSize

Examples

```
m <- matrix(1, 20, 20)
diag(m) <- 0
m
pcitMemoryRequirement(m, "KB")
pcitMemoryRequirement(10000, "GB")
```

`plotCorCoeff` *Plot superimposed histograms of correlation coefficients*

Description

Given a complete correlation matrix and a list of linear indices, superimpose the distributions of correlation coefficients, defined by their indices, on top of the distribution of all correlations.

Only data from the upper triangle is used to plot the distributions since a correlation matrix should be symmetrical. This means that any specified indices which fall within the diagonal or the lower triangle are effectively ignored.

Usage

```
plotCorCoeff(m, idx, col, breaks="Scott", ...)
```

Arguments

`m`	- A correlation matrix representing the raw data.
`idx`	- A list of indices for a subset of correlations to be superimposed on the plot.
`col`	- A vector of colours of equal length to idx. These are used to colour each of the distributions specified by the indices in idx.
`breaks`	- Defaults to "Scott", see the `hist` documentation for info and other options.
`...`	- Additional parameters to be passed to rect().

Author(s)

Nathan S. Watson-Haigh

See Also

`hist`

Examples

```
data(PCIT)
m <- m[1:200,1:200]        # just use a small subset of the data
result <- pcit(m)

op <- par(mfrow=c(2,1))
plotCorCoeff(m, list("PCIT Significant" = idx(result)), col=c("black"))
plotCorCoeff(m, list("PCIT Significant" = idx(result),
"abs(adj) >= 0.7" = which(abs(m) >= 0.7)), col=c("black", "red"))
par(op)
```

Index