Package ‘netbenchmark’

April 23, 2016

Type Package

Title Benchmarking of several gene network inference methods

Version 1.2.0

Date 2015-08-08

Author Pau Bellot, Catharina Olsen, Patrick Meyer, with contributions from Alexandre Irrthum

Maintainer Pau Bellot <pau.bellot@upc.edu>

Contact Pau Bellot <pau.bellot@upc.edu>, Patrick Meyer <Patrick.Meyer@ulg.ac.be>

Description This package implements a benchmarking of several gene network inference algorithms from gene expression data.

License CC BY-NC-SA 4.0

URL https://imatge.upc.edu/netbenchmark/

Imports Rcpp (>= 0.11.0), minet, randomForest, c3net, PCIT, GeneNet, tools, pracma, Matrix, corpcor, fdrtool

LinkingTo Rcpp

Depends grndata (>= 0.99.3)

Suggests RUnit, BiocGenerics, knitr, graph

biocViews Microarray, GraphAndNetwork, Network, NetworkInference, GeneExpression

VignetteBuilder knitr

NeedsCompilation yes

R topics documented:

`netbenchmark-package` *Benchmarking of several inference networks methods*

Description

For a given list of network inference algorithms, `netbenchmark` performs a benchmark between them. It makes use of five different big gene datasources, it relies on a random subsampling of each one of the datasource and noise addition in order to generate the datasets. This package is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International.

Author(s)

Pau Bellot, Catharina Olsen, Patrick E Meyer, with contributions from Alexandre Irrthum

Maintainer: Pau Bellot <pau.bellot@upc.edu>

References

Pau Bellot, Catharina Olsen, Philippe Salembier, Albert Oliveras-Verges, and Patrick E Meyer. Net-Benchmark: A Bioconductor Package for Reproducible Benchmarks of Gene Regulatory Network Inference. Submitted, 2015.

Examples

```
## Not run:
    AUPR20.list<-netbenchmark(datasources.names=c("syntren300",
        "rogers1000"),datasets.num=7)
    AUPR20.300exp.list<-netbenchmark(methods=c("aracne.wrap","mrnet.wrap",
        "GeneNet.wrap"),datasources.names=c("syntren300","rogers1000"),
        experiments=300,global.noise=10,noiseType="lognormal")
        AUPR20.n30.list<-netbenchmark(methods=c("all.fast","mrnet.wrap",
            "Genie3.wrap"),eval="AUROC",local.noise=30)

## End(Not run)
```

`aracne.wrap` *aracne wrapper function*

Description

Default wrapper function for the aracne network inference algorithm

Usage

```
aracne.wrap(data)
```

Arguments

`data`	Numeric matrix with the microarray dataset to infer the network.Columns contain variables and rows contain samples.

Details

The motivation of the Algorithm for the Reconstruction of Accurate Cellular NEtworks (ARACNE) is that many similar measures between variables may be the result of indirect effects. In order to delete the indirect effect the algorithm relies on the “Data Processing Inequality”, this process removes the weakest link in every triplet of connected variables.

Value

`aracne.wrapper` returns a matrix which is the weighted adjacency matrix of the network inferred by aracne algorithm. The wrapper uses the "spearman" correlation (can be used with continuous data) to estimate the entropy - see `build.mim`

References

Margolin, Adam A., et al. "ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context." BMC Bioinformatics 7.Suppl 1 (2006): S7.

Meyer, Patrick E., Frederic Lafitte, and Gianluca Bontempi. "minet: AR/Bioconductor package for inferring large transcriptional networks using mutual information." BMC Bioinformatics 9.1 (2008): 461.

See Also

`netbenchmark`, `evaluate`, `aracne`

Examples

```
 # Data
data <- grndata::getData(datasource.name = "toy",FALSE)
# Inference
net <- aracne.wrap(data)
```

---

c3net.wrap *c3net wrapper function*

Description

Default wrapper function for the C3net network inference algorithm

Usage

```
c3net.wrap(data)
```

Arguments

data	Numeric matrix with the microarray dataset to infer the network. Columns contain variables and rows contain samples.

Details

The Conservative Causal Core NETwork (C3NET) consists of two main steps. The first step is the elimination of non-significant edges, and the second step selects for each gene the edge among the remaining ones with maximum mutual information value. C3NET does not aim at inferring the entire network underlying gene regulation but mainly tries to recover the core structure.

Value

`c3net.wrap` returns a matrix which is the weighted adjacency matrix of the network inferred by c3net algorithm. The Mutual Information threshold is set to 0 - see `c3net`.

References

Altay, G\"okmen, and Frank Emmert-Streib. "Inferring the conservative causal core of gene regulatory networks." BMC Systems Biology 4.1 (2010): 132.

See Also

`netbenchmark`, `evaluate`, `c3net`

Examples

```
 # Data
data <- grndata::getData(datasource.name = "toy",FALSE)
# Inference
net <- c3net.wrap(data)
```

clr.wrap *CLR wrapper function*

Description

Default wrapper function for the CLR network inference algorithm

Usage

```
clr.wrap(data)
```

Arguments

data	Numeric matrix with the microarray dataset to infer the network. Columns contain variables and rows contain samples.

Details

The Context Likelihood or Relatedness network (CLR) method derives a score that is associated to the empirical distribution of the mutual information values, in practice the score between gene X_i and gene X_j is defined as follows $z_{ij} = \sqrt{z_i^2 + z_j^2}$, where:

$$z_i = max\left(0, \frac{I(X_i;X_j) - \mu_i}{\sigma_i}\right)$$

μ_i and σ_i are respectively the mean and standard deviation of the empirical distribution of the mutual information between both genes.

Value

`clr.wrap` returns a matrix which is the weighted adjacency matrix of the network inferred by CLR algorithm. The wrapper uses the "spearman" correlation (can be used with continuous data) to estimate the entropy - see `build.mim`.

References

Faith, Jeremiah J., et al. "Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles." PLoS biology 5.1 (2007): e8.

See Also

`netbenchmark`, `evaluate`, `clr`

Examples

```
# Data
data <- grndata::getData(datasource.name = "toy",FALSE)
# Inference
net <- clr.wrap(data)
```

comp.metr *Compute metrics*

Description

A group of functions to plot precision-recall and ROC curves and to compute f-scores from the matrix returned by the `evaluate` function.

Usage

```
fscore(table, beta=1)
auroc(table,k=-1)
aupr(table,k=-1)
pr.plot(table,device=-1,...)
roc.plot(table,device=-1,...)
```

Arguments

`table`	This is the matrix returned by the `evaluate` function where columns contain the confusion matrix `TP,FP,TN,FN` values. - see `evaluate`.
`beta`	Numeric used as the weight of the recall in the f-score formula - see details. The default value of this argument is -1, meaning precision as important as recall.
`k`	Numeric used as the index to compute the area under the curve until that point-see details. The default value of this argument is -1, meaning that the whole area under the curve is computed
`device`	The device to be used. This parameter allows the user to plot precision-recall and receiver operating characteristic curves for various inference algorithms on the same plotting window - see examples.
`...`	Arguments passed to `plot`.

Details

A confusion matrix contains FP,TP,FN,FP values.

- "true positive rate" $tpr = \frac{TP}{TN+TP}$
- "false positive rate" $fpr = \frac{FP}{FN+FP}$
- "precision" $p = \frac{TP}{FP+TP}$
- "recall" $r = \frac{TP}{TP+FN}$
- "f-beta-score" $F_\beta = (1+\beta)\frac{pr}{r+\beta p}$ Fbeta = (1+beta) * p*r/(r + beta*p)

Value

The function `roc.plot` (`pr.plot`) plots the ROC-curve (PR-curve) and returns the device associated with the plotting window.

The function `auroc` (`aupr`) computes the area under the ROC-curve (PR-curve) using the trapezoidal approximation until point k.

The function `fscore` returns fscores according to the confusion matrices contained in the 'table' argument - see details.

See Also

`evaluate`, `plot`

Examples

```
# Inference
Net <- cor(syntren300.data)
# Validation
tbl <-  evaluate(Net,syntren300.net)
# Plot PR-Curves
max(fscore(tbl))
dev <- pr.plot(tbl, col="green", type="l")
aupr(tbl)
idx <- which.max(fscore(tbl))
```

`datasource.subsample` *Subsampling datasource procedure*

Description

`datasource.subsample` picks randomly the specified amount of samples from the original datasource and also adds noise to the subsampled dataset if it is specified.

Usage

```
datasource.subsample(datasource,experiments=NA,datasets.num=5,
    local.noise=20,global.noise=0,noiseType="normal",
    samplevar=TRUE, seed = NULL)
```

Arguments

`datasource`	data.frame where columns contain variables and rows contain experiments.
`experiments`	Integer specifying the number of experiments that for performing the subsampling of datasources (default: NA).
`datasets.num`	Integer specifying the number of datasets to be generated for each of the selected original datasources (default: 5).

`local.noise`	Integer specifying the desired percentage of local noise to be added at each of the subsampled datasets (default: 20).
`global.noise`	Integer specifying the desired percentage of global noise to be added at each of the subsampled datasets (default: 0).
`noiseType`	Character specifying the type of the noise to be added: "normal" (default: "normal").
`samplevar`	Logical specifying if the datasets should have variability in the number of experiments between them (default: TRUE).
`seed`	A single value, interpreted as an integer to specify seeds, useful for creating simulations that can be reproduced (default: `NULL`) - see `set.seed`.

Details

If the argument `experiments` is NA, the value `experiments` will be calculated automatically in order to have `datasets.num` smaller datasets that does not have the same experiment twice inside each dataset. Each of the subsampled datasets `experiments` would have a number of experiments around `experiments` $\pm 20\%$ that would be chosen randomly among the original the original number of experiments without replacement.

If the argument `experiments` is a number, the number of `datasets.num` is calculated automatically. If the number of specified `experiments` is greater or equal than the original number of experiments, then only a replicate will be generated and the subsampled dataset would have the same dimensions as the original one but the experiments will be unsorted randomly.

Two different types of noises could be added, that are specified with the argument `noiseType`:

- "local": the variance of the noise is different for each variable and it is the percentage specified of the variance of each variable ($\pm 20\%$).
- "Globlal": the variance of the noise is the same for the whole datasource, it is the percentage specified of the mean variance of all the variables ($\pm 20\%$).

Value

`datasource.subsample` returns a list with `datasets.num` elements, each one of objects contains a data.frame of the subsampled dataset with the amount of Gaussian noise specified that would contain the same number of variables.

See Also

`netbenchmark`

Examples

```
# Subsample
data.list.1 <- datasource.subsample(syntren300.data)
data.list.2 <- datasource.subsample(syntren300.data,
    local.noise=10)
# Inference
inf.net.1 <- cor(data.list.1[[1]])
inf.net.2 <- cor(data.list.2[[4]])
```

`evaluate` *Inference Evaluation*

Description

`evaluate` compares the inferred network to the true underlying network for several threshold values and appends the resulting confusion matrices to the returned object.

Usage

```
evaluate(inf.net,true.net,sym=TRUE,extend=0)
```

Arguments

`inf.net`	An adjacency matrix representing the inferred network.
`true.net`	An adjacency matrix representing the true underlying network.
`sym`	Logical, make a symmetric evaluation (default = TRUE).
`extend`	Integer, specifying the desired number of links to extend in the network (default=0)

Details

The first edgelist network `inet` is compared to the true underlying network, `tnet`, in order to compute the metrics of the performance. If extend is specified, `extend` links that network `inet` has set to 0 are added to the inferred network randomly at the end of the edgelist.

Value

`evaluate` returns a matrix with four columns representing `TP,FP,TN,FN`. These values are computed for each of the predicted links that should be sorted. Thus, each row of the returned object contains the confusion matrix as a function of the cutoff in the edgelist.

See Also

`netbenchmark`

Examples

```
# Inference
inf.net <- cor(syntren300.data)
#Evaluate
table <- evaluate(inf.net, syntren300.net)
table.nosym <- evaluate(inf.net, syntren300.net,sym=FALSE)
```

`experiments.bench` *Noise sensitivity benchmark*

Description

For a given vector of character of the names of wrapper functions that compute a network inference methods, `experiments.bench` performs a number of experiments sensitivity test. It makes use of five different big gene datasets subsampling them to generate different datasets.num of the network with different number of experiments.

Usage

```
experiments.bench(methods = "all.fast", datasources.names = "all",
    experiments = c(20, 50, 150), eval = "AUPR",
    no.topedges = 20, datasets.num = 3, local.noise = 20,
    global.noise = 0, noiseType = "normal", sym = TRUE,
    seed = NULL, verbose= TRUE)
```

Arguments

`methods`	A vector of characters containing the names of network inference algorithms wrappers to be compared (default: "all.fast").
`datasources.names`	A vector of characters containing the names of network datasets to be included in the benchmark (default: "all").
`experiments`	A vector to set the number of experiments to test the methods (default=c(20,50,150)).
`eval`	The name of the evaluation metric among the following ones: "no.truepos", "AUROC" or "AUPR" (default : "AUPR") - see `evaluate`.
`no.topedges`	Float specifying the percentage number of links to be considered in the evaluation (default: 20).
`datasets.num`	Number of repetitions in the noise evaluation, for each method and each dataset and each noise intensity (default: 3).
`local.noise`	Integer specifying the desired percentage of local noise to be added at each of the subsampled datasets (default: 20) - see `datasource.subsample`.
`global.noise`	Integer specifying the desired percentage of global noise to be added at each of the subsampled datasets (default: 20) - see `datasource.subsample`.
`noiseType`	Character specifying the type of the noise to be added: "normal" or "lognormal" (default: "normal") - see `datasource.subsample`.
`sym`	Logical specifying if the evaluation is symmetric (default: TRUE) - see `evaluate`.
`seed`	A single value, interpreted as an integer to specify seeds, useful for creating simulations that can be reproduced (default: `NULL`) - see `set.seed`.
`verbose`	Logical specifying if the code should provide a log about what the function is doing (default: TRUE).

Details

The argument methods accepts "all.fast" and "all" (case insensitive) as a parameters:

- "all.fast" performs network inference with "aracne", "c3net", "clr", "GeneNet", "mutual ranking", "mrnet", "pcit"
- "all" performs network inference with "aracne", "c3net", "clr", "GeneNet", "Genie3", "mutual ranking", "mrnet", "mrnetb", "pcit"

It evaluates the first no.topedges % of the possible links inferred by each algorithm at each dataset.

Two different types of noises are added independently:

- "Local": the standard deviation of the noise is different for each variable. local.noise specifies the percentage for each variable ($\pm 20\%$).
- "Global": the standard deviation of the noise is the same for the whole dataset. global.noise specifies the percentage of the mean standard deviation of all the variables ($\pm 20\%$).

The distribution of noise is set with noiseType, it is possible to choose between "normal" (rnorm) and "lognormal" (rlnorm). The argument noiseType can be a single character, this specifies the same distribution for both "Local" and "Global" noise, it also can be a vector of characters with two elements, the former specifies the distribution of "Local" noise and the later the distribution of "Global" noise.

Value

experiments.bench returns a list with three elements:

1. A data.frame which is the result table containing the number of true positives as an evaluation measure. It evaluates each algorithm specified at methods at each one of the specified datasources.names with different noise intensities.
2. A data.frame which is the corresponding pvalue table of the corresponding statistical test for each one of the datasets.num between the best algorithm and the others.
3. The seed of the random number generators that allows the replication of the results.

Author(s)

Pau Bellot and Patrick Meyer

See Also

netbenchmark, noise.bench

Examples

```
results <- experiments.bench(datasources.names="toy",
datasets.num=2,methods="all.fast",experiments=c(20,40))
```

GeneNet.wrap *GeneNet wrapper function*

Description

Default wrapper function for the GeneNet network inference algorithm

Usage

```
GeneNet.wrap(data)
```

Arguments

`data`	Numeric matrix with the microarray dataset to infer the network. Columns contain variables and rows contain samples.

Details

GeneNEt uses an heuristic for learning statistically a causal network. It relies on a conversion of a network inferred through correlation into a partial correlation graph. Then, a partial ordering of the nodes is assigned by means of a multiple testing of the log-ratio of standardized partial variances. This allows identifying a directed acyclic causal network as a sub-graph of the partial correlation network.

Value

`GeneNet.wrap` The function returns a matrix which is the weighted adjacency matrix of the network inferred by GeneNet algorithm. The shrinkage method used to estimate the partial correlation matrix is "static". - see `ggm.estimate.pcor`. The probability threshold is set to 0.8. - see `ggm.estimate.pcor`.

References

Opgen-Rhein, Rainer, and Korbinian Strimmer. "Inferring gene dependency networks from genomic longitudinal data: a functional data approach." RevStat 4.1 (2006): 53-65.

Opgen-Rhein, Rainer, and Korbinian Strimmer. "Using regularized dynamic correlation to infer gene dependency networks from time-series microarray data." Proceedings of the 4th International Workshop on Computational Systems Biology (WCSB 2006), Tampere. Vol. 4. 2006.

Sch\"afer, Juliane, and Korbinian Strimmer. "A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics." Statistical applications in genetics and molecular biology 4.1 (2005): 32.

See Also

`netbenchmark`, `evaluate`, `GeneNet-package`

Examples

```
# Data
data <- grndata::getData(datasource.name = "toy",FALSE)
# Inference
net <- GeneNet.wrap(data)
```

Genie3.wrap *Genie3 wrapper function*

Description

Default wrapper function for the Genie3 network inference algorithm

Usage

```
Genie3.wrap(data)
```

Arguments

data	Numeric matrix with the microarray dataset to infer the network. Columns contain variables and rows contain samples.

Details

GEne Network Inference with Ensemble of trees (Genie3) algorithm uses the Random Forests feature selection technique to solve a regression problem for each of the genes in the network. In each of the regression problems, the expression pattern of the target gene should be predicted from the expression patterns of all transcription factors. The importance of each transcription factor in the prediction of the target gene is taken as an indication of an apparent regulatory link. Then these candidate regulatory links are aggregated over all genes to generate a ranking for the whole network.

Value

`Genie3.wrap` returns a matrix which is the weighted adjacency matrix of the network inferred by Genie3 algorithm. 500 trees are used in ensemble for each target gene.

References

Irrthum, Alexandre, Louis Wehenkel, and Pierre Geurts. "Inferring regulatory networks from expression data using tree-based methods." PloS one 5.9 (2010): e12776.

Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

See Also

`netbenchmark`, `evaluate`

Examples

```
# Data
data <- grndata::getData(datasource.name = "toy",FALSE)
# Inference
net <- Genie3.wrap(data)
```

`mrnet.wrap` *mrnet wrapper function*

Description

Default function for the MRNET network inference algorithm

Usage

```
mrnet.wrap(data)
```

Arguments

`data`	Numeric matrix with the microarray dataset to infer the network. Columns contain variables and rows contain samples.

Details

The MRNET approach consists in repeating a MRMR feature selection procedure for each variable of the dataset. The MRMR method starts by selecting the variable X_i having the highest mutual information with the target Y. In the following steps, given a set $\mathcal{S}$ of selected variables, the criterion updates $\mathcal{S}$ by choosing the variable X_k that maximizes $I(X_k;Y) - \frac{1}{|\mathcal{S}|}\sum_{X_i \in \mathcal{S}} I(X_k;X_i)$
The weight of each pair X_i, X_j will be the maximum score between the one computed when X_i is the target and the one computed when X_j is the target.

Value

`mrnet.wrap` returns a matrix which is the weighted adjacency matrix of the network inferred by MRNET algorithm. The wrapper uses the "spearman" correlation (can be used with continuous data) to estimate the entropy - see `build.mim`.

References

Patrick E. Meyer, Kevin Kontos, Frederic Lafitte and Gianluca Bontempi. Information-theoretic inference of large transcriptional regulatory networks. EURASIP Journal on Bioinformatics and Systems Biology, 2007.

Patrick E. Meyer, Frederic Lafitte and Gianluca Bontempi. minet: A R/Bioconductor Package for Inferring Large Transcriptional Networks Using Mutual Information. BMC Bioinformatics, Vol 9, 2008.

H. Peng, F.long and C.Ding. Feature selection based on mutual information: Criteria of max-dependency, max relevance and min redundancy. IEEE transaction on Pattern Analysis and Machine Intelligence, 2005.

See Also

netbenchmark, evaluate, mrnet

Examples

```
 # Data
data <- grndata::getData(datasource.name = "toy",FALSE)
# Inference
net <- mrnet.wrap(data)
```

---

mrnetb.wrap *mrnetb wrapper function*

Description

Default wrapper function for the MRNETB network inference algorithm

Usage

```
mrnetb.wrap(data)
```

Arguments

data	Numeric matrix with the microarray dataset to infer the network. Columns contain variables and rows contain samples.

Details

mrnetb takes the mutual information matrix as input in order to infer the network using the maximum relevance/minimum redundancy criterion combined with a backward elimination and a sequential replacement - see references. This method is a variant of mrnet.

Value

mrnetb.wrap returns a matrix which is the weighted adjacency matrix of the network inferred by mrnetb algorithm. The wrapper uses the "spearman" correlation (can be used with continuous data) to estimate the entropy - see build.mim.

References

Patrick E. Meyer, Daniel Marbach, Sushmita Roy and Manolis Kellis. Information-Theoretic Inference of Gene Networks Using Backward Elimination. The 2010 International Conference on Bioinformatics and Computational Biology.

Patrick E. Meyer, Kevin Kontos, Frederic Lafitte and Gianluca Bontempi. Information-theoretic inference of large transcriptional regulatory networks. EURASIP Journal on Bioinformatics and Systems Biology, 2007.

See Also

netbenchmark, evaluate, mrnet

Examples

```
# Data
data <- grndata::getData(datasource.name = "toy",FALSE)
# Inference
net <- mrnetb.wrap(data)
```

mutrank.wrap *Mutual Rank wrapper function*

Description

A wrapper function for mutual rank.

Usage

```
mutrank.wrap(data)
```

Arguments

data	Numeric matrix with the microarray dataset to infer the network. Columns contain variables and rows contain samples.

Value

mutrank.wrap returns a matrix which is the weighted adjacency matrix of the network inferred by Mutual Rank algorithm.

References

Obayashi, Takeshi, and Kengo Kinoshita. "Rank of correlation coefficient as a comparable measure for biological significance of gene coexpression." DNA research 16.5 (2009): 249-260.

Examples

```
 # Data
data <- grndata::getData(datasource.name = "toy",FALSE)
# Inference
net <- mutrank.wrap(data)
```

`netbenchmark` *Benchmarking of several network inference algorithms from data*

Description

For a given vector of character of the names of wrapper functions that compute a network inference methods, `netbenchmark` performs a benchmark between them. It makes use of four different big gene datasources, it relies on a random subsampling without repetition of each one of the datasets and noise addition in order to generate the source data.

Usage

```
netbenchmark(methods="all.fast",datasources.names="all",experiments=150,
    eval="AUPR",no.topedges=20,datasets.num=5,local.noise=20,
    global.noise=0,noiseType="normal",sym=TRUE,plot=FALSE,seed=NULL,
    verbose=TRUE)
```

Arguments

`methods`	A vector of characters containing the names of network inference algorithms wrappers to be compared (default: "all.fast").
`datasources.names`	A vector of characters containing the names of network datasources to be included in the benchmark (default: "all").
`experiments`	Integer specifying the number of experiments to generate the subsampled datasets (default: 150) - see `datasource.subsample`.
`eval`	The name of the evaluation metric among the following ones: "no.truepos", "AUROC" or "AUPR" (default : "AUPR") - see `evaluate`.
`no.topedges`	Float specifying the percentage number of links to be considered in the evaluation (default: 20).
`datasets.num`	Integer specifying the number of datasets.num to be generated for each of the selected original datasources (default: 5).
`local.noise`	Integer specifying the desired percentage of local noise to be added at each of the subsampled datasets (default: 20) - see `datasource.subsample`.
`global.noise`	Integer specifying the desired percentage of global noise to be added at each of the subsampled datasets (default: 20) - see `datasource.subsample`.
`noiseType`	Character specifying the type of the noise to be added: "normal" or "lognormal" (default: "normal") - see `datasource.subsample`.
`sym`	Logical specifying if the evaluation is symmetric (default: TRUE) - see `evaluate`.
`plot`	(default: FALSE)
`seed`	A single value, interpreted as an integer to specify seeds, useful for creating simulations that can be reproduced (default: `NULL`) - see `set.seed`.
`verbose`	Logical specifying if the code should provide a log about what the function is doing (default: TRUE).

Details

The argument `methods` accepts "all.fast" and "all" (case insensitive) as a parameters:

- "all.fast" performs network inference with "aracne", "c3net", "clr", "GeneNet", "mutual ranking", "mrnet", "pcit" (and registered methods with `RegisterWrapper`.)
- "all" performs network inference with "aracne", "c3net", "clr", "GeneNet", "Genie3", "mutual ranking", "mrnet", "mrnetb", "pcit" (and registered methods with `RegisterWrapper`.)

The argument `datasources.names` accepts "all" or a selection of the following datasources `Availabledata`:

- "rogers1000"
- "syntren300"
- "syntren1000"
- "gnw1565"
- "gnw2000"

All the measures only evaluates the first `no.topedges` % of the possible links inferred by each algorithm at each dataset. The statistical used is the Wilcoxon Rank Sum Test (`wilcox.test`). This test compares the number of true positives of any method with number of trials specified with the best method at each replicate.

Value

`netbenchmark` returns a list with six elements.

1. A data.frame which is the result table of the selected measure.
2. A data.frame which is the corresponding pvalue table of the corresponding statistical test for each one of the `datasets.num` between the best algorithm and the others.
3. A data.frame that sumarizes the first data.frame presenting the mean and standard deviation of the measures of each algorithm per datasource.
4. A data.frame which contains the CPU Time Used (in seconds) by the algorithm to infer the network.
5. A list containing the mean precision recall curves of the different algorithms for each datasource.
6. The seed of the random number generators that allows the replication of the results.

Each of these data.frame will have the same number of columns as methods provided by the user and an additional one for a random method, and the number of rows will depend on the number of `datasets.num` and `datasources.name` specified by the user.

Author(s)

Pau Bellot, Catharina Olsen and Patrick E Meyer Maintainer: Pau Bellot <pau.bellot@upc.edu>

See Also

`datasource.subsample`, `evaluate`, `comp.metr`

Examples

```
top20.aupr <- netbenchmark(methods="all",datasources.names = "Toy",
                           local.noise=20,global.noise=10,
                           noiseType=c("normal","lognormal"),
                           datasets.num = 2,experiments = 40)
## Not run:
    # Other possible studies
    top20.fast.list <- netbenchmark()
    top20.list <- netbenchmark(methods="all",eval="no.truepos")
    top50.auroc.list <- netbenchmark(datasets.num=8,eval="AUROC",
        no.topedges=50,global.noise=10)
    top9.list <- netbenchmark(datasets.num=8,no.topedges=9,local.noise=15,
        noiseType="lognormal")
    #To export the tables to LaTeX
    # library(xtable)
    # xtable(top20.fast.list[[1]])

## End(Not run)
```

`netbenchmark.data` *Benchmarking of several network inference algorithms for your own data*

Description

Benchmarking of several network inference algorithms for your own data

Usage

```
netbenchmark.data(methods = "all.fast", data = NULL, true.net = NULL,
  eval = "AUPR", no.topedges = 20, sym = TRUE, plot = FALSE,
  verbose = TRUE)
```

Arguments

`methods`	A vector of characters containing the names of network i inference algorithms wrappers to be compared (default: "all.fast").
`data`	data.frame containing the data. Each row should contain a microarray experiment and each column a gene (default: NULL).
`true.net`	matrix containg underlying network in the form of adjacency matrix (default: NULL).
`eval`	The name of the evaluation metric among the following ones: "no.truepos", "AUROC" or "AUPR" (default : "AUPR").
`no.topedges`	Float specifying the percentage number of links to be considered in the evaluation (default: 20)
`sym`	Logical specifying if the evaluation is symmetric (default: TRUE) - see `evaluate`

`plot`	(default: FALSE)
`verbose`	Logical specifying if the code should provide a log about what the function is doing (default: TRUE).

Details

The argument `methods` accepts "all.fast" and "all" (case insensitive) as a parameters:

- "all.fast" performs network inference with "aracne", "c3net", "clr", "GeneNet", "mutual ranking", "mrnet", "pcit" (and registered methods with `RegisterWrapper`.)
- "all" performs network inference with "aracne", "c3net", "clr", "GeneNet", "Genie3", "mutual ranking", "mrnet", "mrnetb", "pcit" (and registered methods with `RegisterWrapper`.)

All the measures only evaluates the first `no.topedges` % of the possible links inferred by each algorithm at each dataset.

Value

`netbenchmark.data` returns a list with three elements.

1. A data.frame which is the result table of the selected measure.
2. A data.frame which contains the CPU Time Used (in seconds) by the algorithm to infer the network.
3. A list containing the mean precision recall curves of the different algorithms for each datasource.

Each of these data.frame will have the same number of columns as methods provided by the user and an additional one for a random method.

Author(s)

Pau Bellot, Catharina Olsen and Patrick E Meyer Maintainer: Pau Bellot <pau.bellot@upc.edu>

See Also

`netbenchmark`, `evaluate`, `comp.metr`

Examples

```
Data <- grndata::getData(datasource.name="toy")
top20.aupr <- netbenchmark.data(methods="all",data = Data[[1]],
                               true.net= Data[[2]])
```

noise.bench *Noise sensitivity test*

Description

For a given vector of character of the names of wrapper functions that compute a network inference methods, `noise.bench` performs a noise sensitivity test. It makes use of different big gene datasets adding Gaussian noise with different intensity to evaluate the performance of the methods.

Usage

```
noise.bench(methods = "all.fast", datasources.names = "all",
    eval = "AUPR", no.topedges = 20,experiments=150,
    datasets.num = 3, local.noise = seq(0, 100, len = 3),
    global.noise = 0, noiseType = "normal", sym = TRUE,
    seed = NULL, verbose = TRUE)
```

Arguments

`methods`	A vector of characters containing the names of network inference algorithms wrappers to be compared (default: "all.fast").
`datasources.names`	A vector of characters containing the names of network datasets to be included in the benchmark (default: "all").
`eval`	The name of the evaluation metric among the following ones: "no.truepos", "AUROC" or "AUPR" (default : "AUPR") - see `evaluate`.
`experiments`	Integer specifying the number of experiments to generate the subsampled datasets (default: 150) - see `datasource.subsample`.
`datasets.num`	Number of repetitions in the noise evaluation, for each method and each dataset and each noise intensity (default: 5).
`no.topedges`	Float specifying the percentage number of links to be considered in the evaluation (default: 20).
`local.noise`	Vector specifying the desired percentage of local noise to be added at each of the subsampled datasets (default: seq(0, 100, len = 3)).
`global.noise`	Vector specifying the desired percentage of global noise to be added at each of the subsampled datasets (default: 0).
`noiseType`	Character specifying the type of the noise to be added: "normal" (default: "normal").
`sym`	Logical specifying if the evaluation is symmetric (default: TRUE) - see `evaluate`.
`seed`	A single value, interpreted as an integer to specify seeds, useful for creating simulations that can be reproduced (default: `NULL`) - see `set.seed`.
`verbose`	Logical specifying if the code should provide a log about what the function is doing (default: TRUE).

Details

The argument `methods` accepts "all.fast" and "all" (case insensitive) as a parameters:

- "all.fast" performs network inference with "aracne", "c3net", "clr", "GeneNet", "mutual ranking", "mrnetb", "pcit"
- "all" performs network inference with "aracne", "c3net", "clr", "GeneNet", "Genie3", "mutual ranking", "mrnet", "mrnetb", "pcit"

It evaluates the first `no.topedges` % of the possible links inferred by each algorithm at each dataset.

Value

`noise.bench` returns a list with three elements:

1. A data.frame which is the result table containing the number of true positives as an evaluation measure. It evaluates each algorithm specified at `methods` at each one of the specified `datasources.names` with the `local.noise` and `global.noise` specified. For each combination the algorithms are evaluated `datasets.num` times and their results are averaged.
2. A data.frame which is the corresponding pvalue table of the corresponding statistical test for each one of the `datasets.num` between the best algorithm and the others.
3. The seed of the random number generators that allows the replication of the results.

Author(s)

Pau Bellot and Patrick Meyer

See Also

`netbenchmark`, `experiments.bench`

Examples

```
    results <- noise.bench(datasources.names="toy",
        datasets.num=2,methods="all.fast",experiments=NULL)
```

`ntb_globals` *Available wrappers in the package of the fast methods*

Description

Environment containing a character vector containing the names of the wrappers in the package of the registered methods.

Usage

```
ntb_globals
```

Format

Character vector containing the names of the registered wrapper methods.

Examples

```
print(ntb_globals$Fast)
print(ntb_globals$All)
```

`pcit.wrap` *pcit wrapper function*

Description

Default wrapper function for the pcit network inference algorithm

Usage

```
pcit.wrap(data)
```

Arguments

`data`	Numeric matrix with the microarray dataset to infer the network. Columns contain variables and rows contain samples.

Details

The Partial Correlation coefficient with Information Theory (PCIT) algorithm, combines the concept of partial correlation coefficient with information theory to identify significant gene-to-gene associations.

For every trio of genes in X_i, X_j and X_l, the three first-order partial correlation coefficients are computed. These coefficients indicate the strength of the linear relationship between X_i and X_j that is uncorrelated with X_l, being therefore a measure of conditional independence. Then, the average ratio of partial to direct correlation is computed in order to obtain the tolerance level to be used as the local threshold for eliminating non-significant associations.

Value

`pcit.wrap` returns a matrix which is the weighted adjacency matrix of the network inferred by pcit algorithm.

References

Reverter, Antonio, and Eva KF Chan. "Combining partial correlation and an information theory approach to the reversed engineering of gene co-expression networks." Bioinformatics 24.21 (2008): 2491-2497.

See Also

netbenchmark, evaluate, pcit

Examples

```
# Data
data <- grndata::getData(datasource.name = "toy",FALSE)
# Inference
net <- pcit.wrap(data)
```

---

rate *Inference Validation*

Description

`rate` compares the infered network to the true underlying network for all the sorted predictions provided and appends the resulting confusion matrices to the returned object.

Usage

```
rate(PredEdgeList, GSEdgeList, ngenes, sym)
```

Arguments

`PredEdgeList`	The inferred network in the form of a EdgeList.
`GSEdgeList`	The true underlying in the form of a EdgeList.
`ngenes`	Integrer denoting the number of total genes in the network.
`sym`	Logical specifying if the evaluation is symmetric (default: TRUE) - see evaluate.

Value

A matrix of numerics with the contingency table for each link in `PredEdgeList`.

Author(s)

Pau Bellot

See Also

netbenchmark, evaluate, comp.metr

Examples

```
# Data
net <- matrix(0,10,10)
net[sample(1:100,20)] <- 1
# Simulated Inference
inf <- net+matrix(rnorm(100,sd=0.5),10,10)
table <- evaluate(inf,net)
```

`RegisterWrapper` *Wrapper (un)registration routine*

Description

These function allows the registration and unregistration of a wrapper function to the all.fast or all methods of `netbenchmark`. After registring it `wrapper.name` function will belong to all.fast or all methods during the R session. Unregistring the `wrapper.name` function will remove it from all.fast or all methods during the R session.

Usage

```
RegisterWrapper(wrapper.name=NULL,all.fast=TRUE)
UnregisterWrapper(wrapper.name=NULL,all.fast=TRUE)
```

Arguments

`wrapper.name`	The character (vector) of wrapper names (default: NULL).
`all.fast`	Logical indicating if the `wrapper.name` should be added to all.fast or all methods (default: TRUE).

Value

Displays a message if the registration could be performed or not.

Author(s)

Pau Bellot, Catharina Olsen and Patrick E Meyer Maintainer: Pau Bellot <pau.bellot@upc.edu>

See Also

`netbenchmark`

Examples

```
# Define a wrapper function
Spearmancor <- function(data){
    cor(data,method="spearman")
}
## Not run:
    # Register it to all.fast methods
    RegisterWrapper("Spearmancor")
    # Register it to all methods
    RegisterWrapper("Spearmancor", all.fast=FALSE)
    # Unregister it from all.fast methods
    UnregisterWrapper("Spearmancor")
    # Unregister it from all methods
    UnregisterWrapper("Spearmancor", all.fast=FALSE)
```

```
## End(Not run)
```

---

zsc *Z-score c++ function*

Description

Z-score c++ function

Usage

```
zsc(x)
```

Arguments

x	Numeric matrix with the microarray dataset to infer the network. Columns contain variables and rows contain samples.

Value

A matrix of numerics with the inferred adjacency matrix.

References

Prill, Robert J., et al. "Towards a rigorous assessment of systems biology models: the DREAM3 challenges." PloS one 5.2 (2010): e9202.

See Also

netbenchmark

Examples

```
# Data
data <- runif(100)
dim(data) <- c(10,10)
# Inference
net <- zsc(data)
```

`zscore.wrap` *Zscore wrapper function*

Description

Z-score wrapper function.

Usage

```
zscore.wrap(data)
```

Arguments

`data`	Numeric matrix with the microarray dataset to infer the network. Columns contain variables and rows contain samples.

Details

Zscore is a method that assumes interventional data, more concretely knockout experiments that leads to a change in other genes. The assumption is that the knocked-out gene i in the experiment k affects more strongly to the genes that it regulates than the others, the effect of the gene i over the gene j is captured with the Zscore z_{ij}:

$$z_{ij} = |\frac{x_{jk} - \mu_j}{\sigma_j}|$$

mu_j and σ_j are respectively the mean and standard deviation of the empirical distribution of the gene j.

Value

`zscore.wrap` returns a matrix which is the weighted adjacency matrix of the network inferred by Zscore algorithm.

References

Prill, Robert J., et al. "Towards a rigorous assessment of systems biology models: the DREAM3 challenges." PloS one 5.2 (2010): e9202.

Examples

```
# Data
data <- grndata::getData(datasource.name = "toy",FALSE)
# Inference
net <- zscore.wrap(data)
```

Index