Package 'TDCor'

May 7, 2026

Type Package

Title Gene Network Inference from Time-Series Transcriptomic Data

Version 0.1-2

Date 2015-10-05

Author Julien Lavenus

Maintainer Mikael Lucas <mikael.lucas@ird.fr>

Imports parallel

Depends R (>= 3.1.2), deSolve

Description The Time-Delay Correlation algorithm (TDCor) reconstructs the topology of a gene regulatory network (GRN) from time-series transcriptomic data. The algorithm is described in details in Lavenus et al., Plant Cell, 2015. It was initially developed to infer the topology of the GRN controlling lateral root formation in Arabidopsis thaliana. The time-series transcriptomic dataset which was used in this study is included in the package to illustrate how to use it.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-26 15:58:36

Contents

`TDCor-package` *TDCor algorithm for gene regulatory network inference*

Description

TDCor (Time-Delay Correlation) is an algorithm designed to infer the topology of a gene regulatory network (GRN) from time-series transcriptomic data. The algorithm is described in details in Lavenus *et al.*, Plant Cell, 2015. It was initially developped to infer the topology of the GRN controlling lateral root formation in *Arabidopsis thaliana*. The time-series transcriptomic dataset analysed in this study is included in the package.

Details

Package:	TDCor
Type:	Package
Version:	1.2
Date:	2015-10-05
License:	GNU General Public License Version 2

The reconstruction of a gene network using the TDCor package involves six steps.

1. Load the averaged non-log2 time series transcriptomic data into the R workspace.
2. Define the vector `times` containing the times (in hours) at which the samples were collected.
3. Define the vector containing the gene codes of the genes you want to reconstruct the network with (e.g. see `l_genes`), as well as the associated gene names (e.g. see `l_names`) and the associated prior (e.g. see `l_prior`).
4. Build or update the TPI database using the `CalculateTPI` or `UpdateTPI` functions.
5. Build or update the DPI database using the `CalculateDPI` or `UpdateDPI` functions.
6. Reconstruct the network using the `TDCOR` main function.

See examples below.

Besides the functions of the TDCor algorithm, the package also contains the lateral root transcriptomic dataset (`LR_dataset`), the `times` vector to use with this dataset (`times`), the vector of AGI gene codes used to reconstruct the network shown in the original paper (`l_genes`), the vector of the gene names (`l_names`) and the prior (`l_prior`). The associated TPI and DPI databases (`TPI10` and `DPI15`) which were used to build the network shown in the original paper are not included. Hence

to reconstruct the lateral root network, these first need to be generated. A database of about 1800 *Arabidopsis* transcription factors is also included (TF).

Three side functions, estimate.delay, shortest.path and draw.profile are also available to the user. These can be used to visualize the transcriptomic data, optimize some of the TDCOR parameters, and analyze the networks.

Author(s)

Author: Julien Lavenus <jl.tdcor@gmail.com>
Maintainer: Mikael Lucas <mikael.lucas@ird.fr>

References

Lavenus *et al.* (2015), **Inference of the Arabidopsis lateral root gene regulatory network suggests a bifurcation mechanism that defines primordia flanking and central zones.** The Plant Cell, *in press*.

See Also

See also CalculateDPI, CalculateTPI, UpdateDPI, UpdateTPI, TDCOR, estimate.delay.

Examples

```
## Not run:
# Load the LR transcriptomic dataset
data(LR_dataset)

# Load the vectors of genes codes, gene names and prior
data(l_genes)
data(l_names)
data(l_prior)

# Load the vector of time points for the LR_dataset
data(times)

# Generate the TPI database (this may take several hours)

TPI10=CalculateTPI(dataset=LR_dataset,l_genes=l_genes,l_prior=l_prior,
times=times,time_step=1,N=10000,ks_int=c(0.5,3),kd_int=c(0.5,3),
delta_int=c(0.5,3),noise=0.1,delay=3)

# Generate the DPI database (this may take several hours)

DPI15=CalculateDPI(dataset=LR_dataset,l_genes=l_genes,l_prior=l_prior,
times=times,time_step=1,N=10000,ks_int=c(0.5,3),kd_int=c(0.5,3),
delta_int=c(0.5,3), noise=0.15, delay=3)

# Check/update if necessary the databases

TPI10=UpdateTPI(TPI10,LR_dataset,l_genes,l_prior)
DPI15=UpdateDPI(DPI15,LR_dataset,l_genes,l_prior)
```

```
# Choose your parameters

ptime_step=1
ptol=0.13
pdelayspan=12
pthr_cor=c(0.65,0.8)
pdelaymax=c(2.5,3.5)
pdelaymin=0
pdelay=3
pthrpTPI=c(0.55,0.8)
pthrpDPI=c(0.65,0.8)
pthr_overlap=c(0.4,0.6)
pthr_ind1=0.65
pthr_ind2=3.5
pn0=1000
pn1=10
pregmax=5
pthr_isr=c(4,6)
pTPI=TPI10
pDPI=DPI15
pMinTarNumber=5
pMinProp=0.6
poutfile_name="TDCor_output.txt"


# Reconstruct the network

tdcor_out= TDCOR(dataset=LR_dataset, l_genes=l_genes,l_names=l_names,n0=pn0,n1=pn1,
l_prior=l_prior, thr_ind1=pthr_ind1,thr_ind2=pthr_ind2,regmax=pregmax,thr_cor=pthr_cor,
delayspan=pdelayspan,delaymax=pdelaymax,delaymin=pdelaymin,delay=pdelay,thrpTPI=pthrpTPI,
thrpDPI=pthrpDPI,TPI=pTPI,DPI=pDPI,thr_isr=pthr_isr,time_step=ptime_step,thr_overlap=pthr_overlap,
tol=ptol,MinProp=pMinProp,MinTarNumber=pMinTarNumber,outfile_name=poutfile_name)


## End(Not run)
```

`CalculateDPI` *Generate the DPI database to be used by the TDCOR main function*

Description

`CalculateDPI` builds a DPI database for the TDCOR main function to prune diamond motifs

Usage

```
CalculateDPI(dataset,l_genes, l_prior, times, time_step, N, ks_int, kd_int,
delta_int, noise, delay)
```

Arguments

dataset	Numerical matrix storing the transcriptomic data. The rows of this matrix must be named by gene codes (like AGI gene codes for Arabidopsis data).
l_genes	A character vector containing the gene codes of the genes included in the analysis (i.e. to be used to build the network)
l_prior	A numerical vector containing the prior information on the genes included in the network recontruction. By defining the l_prior vector, the user defines which genes should be regarded as positive regulators, which others as negative regulators and which can only be targets. The prior code is defined as follow: -1 for negative regulator; 0 for non-regulator (target only); 1 for positive regulator; 2 for both positive and negative regulator. The i-th element of the vector is the prior to associate to the i-th gene in l_genes.
times	A numerical vector containing the successive times at which the samples were collected to generate the time-series transcriptomic dataset.
time_step	A positive number corresponding to the time step (in hours) i.e. the temporal resolution at which the gene profiles are analysed.
N	An integer corresponding to the number of iterations that are carried out in order to estimate the DPI distributions. N should be >5000.
ks_int	A numerical vector containing two positive elements in increasing order. The first (second) element is the lower (upper) boundary of the interval into which the equation parameters corresponding to the regulation strength of the targets by their regulators are randomly sampled.
kd_int	A numerical vector containing two positive elements in increasing order. The first (second) element is the lower (upper) boundary of the interval into which the equation parameters corresponding to the transcripts degradation rates are randomly sampled.
delta_int	A numerical vector containing two positive elements in increasing order and expressed in hours. The first (second) element is the lower (upper) boundary of the sampling interval for the equation parameters corresponding to the time needed for the transcripts of the regulator to mature, to get exported out of the nucleus, to get translated and for the regulator protein to get imported into the nucleus and to bind its target promoter.
noise	A positive number between 0 and 1 corresponding to the noisiness of the system. (0 = no noise, 1 = very strong noise). noise should not be too high (for instance below 0.2).
delay	A positive number corresponding to the time shift (in hours) that is expected between the profile of a regulator and its direct target. This parameter is used to generate a reference target profile from the profile of the regulator and calculate the DPI index.

Details

CalculateDPI models two 4-genes networks showing slightly different topologies. Each network topology is modelled using a specific system of delay differential equations. For all genes listed in l_genes whose corresponding prior in l_prior is not null (i.e. the genes that are regarded as

transcriptional regulators), the two systems of differential equations are solved N times with N different sets of random parameters. The Diamond Pruning Index (DPI) is calculated for all of these 2N networks. From these in silico data the conditional probability distribution of the DPI index given the regulator and the topology can be estimated. The probability distribution of the topology given DPI and the regulator is next calculated using Bayes' theorem and returned by the function. These shall be used when reconstructing the network to prune the "diamond" motifs.

CalculateDPI returns a list object which works as a database. It not only stores the conditional probability distributions but also all the necessary information for TDCOR to access the data, and the input parameters. The latter are read by the UpdateDPI function to update the database.

Value

CalculateDPI returns a list object.

prob_DPI_ind	A numerical vector whose elements are named by the vector l_genes; The element named gene i contains 0 if no probability distribution has been calculated for this gene (because its prior is 0) or a positive integer if this has been done. This positive integer then correponds to the number of the element in the list prob_DPI that stores the spline functions of the calculated conditional probability distributions associated with this particular regulator.
prob_DPI	A list storing lists of 3 spline functions of probability distributions. Each of the spline functions corresponds to the probability distribution of one topology given a regulator and a DPI value. The information about which regulator was used to generate the distributions stored in the i-th element of prob_DPI is stored in the prob_DPI_ind vector.
prob_DPI_domain	A list storing vectors of two elements. The first (second) element of element i is the lowest (greatest) DPI value obtained during the simulation with the regulator i.
input	A list that stores the input parameters used to generate the database.

Note

The computation of the TPI and DPI databases is time-consuming as it requires many systems of differential equations to be solved. It may take several hours to build a database for a hundred genes.

Author(s)

Julien Lavenus <jl.tdcor@gmail.com>

See Also

See also UpdateDPI, TDCor-package.

Examples

```
## Not run:
# Load the LR transcriptomic dataset
data(LR_dataset)

# Load the vector of gene codes, gene names and prior
data(l_genes)
data(l_names)
data(l_prior)

# Load the vector of time points for the LR_dataset
data(times)

# Generate a small DPI database (3 genes)
DPI_example=CalculateDPI(dataset=LR_dataset,l_genes=l_genes[4:6],l_prior=l_prior[4:6],
times=times,time_step=1,N=5000,ks_int=c(0.5,3),kd_int=c(0.5,3),delta_int=c(0.5,3),
noise=0.15,delay=3)


## End(Not run)
```

`CalculateTPI` *Generate the TPI database to be used by the TDCOR main function*

Description

`CalculateTPI` builds a TPI database for the TDCOR main function to prune triangle motifs

Usage

```
CalculateTPI(dataset,l_genes, l_prior, times, time_step, N, ks_int, kd_int,
delta_int, noise, delay)
```

Arguments

`dataset`	Numerical matrix storing the transcriptomic data. The rows of this matrix must be named by gene codes (AGI gene codes for Arabidospis data).
`l_genes`	A character vector containing the gene codes of the genes included in the analysis (i.e. to be used to build the network)
`l_prior`	A numerical vector containing the prior information on the genes included in the network recontruction. By defining the `l_prior` vector, the user defines which genes should be regarded as positive regulators, which others as negative regulators and which can only be targets. The prior code is defined as follow: -1 for negative regulator; 0 for non-regulator (target only); 1 for positive regulator; 2 for both positive and negative regulator. The i-th element of the vector is the prior to associate to the i-th gene in `l_genes`.
`times`	A numerical vector containing the successive times at which the samples were collected to generate the time-series transcriptomic dataset.

`time_step` A positive number corresponding to the time step (in hours) i.e. the temporal resolution at which the gene profiles are analysed.

`N` An integer corresponding to the number of iterations that are carried out in order to estimate the TPI distributions. `N` should be >5000.

`ks_int` A numerical vector containing two positive elements in increasing order. The first (second) element is the lower (upper) boundary of the interval into which the equation parameters corresponding to the regulation strength of the targets by their regulators are randomly sampled.

`kd_int` A numerical vector containing two positive elements in increasing order. The first (second) element is the lower (upper) boundary of the interval into which the equation parameters corresponding to the transcripts degradation rates are randomly sampled.

`delta_int` A numerical vector containing two positive elements in increasing order and expressed in hours. The first (second) element is the lower (upper) boundary of the sampling interval for the equation parameters corresponding to the time needed for the transcripts of the regulator to mature, to get exported out of the nucleus, to get translated and for the regulator protein to get imported into the nucleus and to bind its target promoter.

`noise` A positive number between 0 and 1 corresponding to the noisiness of the system. (0 = no noise, 1 = very strong noise). `noise` should not be too high (for instance below 0.2).

`delay` A positive number corresponding to the time shift (in hours) that is expected between the profile of a regulator and its direct target. This parameter is used to generate a reference target profile from the profile of the regulator and calculate the TPI index.

Details

`CalculateTPI` models three 3-genes networks showing slightly different topologies. Each network topology is modelled using a specific system of delay differential equations. For all genes listed in `l_genes` whose corresponding prior in `l_prior` is not null (i.e. the genes that are regarded as transcriptional regulators), the three systems of differential equations are solved `N` times with `N` different sets of random parameters. The Triangle Pruning Index (TPI) is calculated for all of these `3N` networks. From these in silico data the conditional probability distribution of the TPI index given the regulator and the topology can be estimated. The probability distribution of the topology given TPI and the regulator is next calculated using Bayes' theorem and returned by the function. These shall be used when reconstructing the network to prune the "triangle" motifs.

`CalculateTPI` returns a list object which works as a database. It not only stores the calculated probability distributions but also information on how to access the data, and the input parameters. The latter are read by the `UpdateTPI` function to update the database.

Value

`CalculateTPI` returns a list object.

prob_TPI_ind	A numerical vector whose elements are named by the vector l_genes; The element named gene i contains 0 if no probability distribution has been calculated for this gene (because its prior is 0) or a positive integer if this has been done. This positive integer then correponds to the number of the element in the list prob_TPI that stores the spline functions of the calculated conditional probability distributions associated with this particular regulator.
prob_TPI	A list storing lists of 3 spline functions of probability distributions. Each of the spline functions corresponds to the probability distribution of one topology given a regulator and a TPI value. The information about which regulator was used to generate the distributions stored in the i-th element of prob_TPI is stored in the prob_TPI_ind vector.
prob_TPI_domain	A list storing vectors of two elements. he first (second) element of element i is the lowest (greatest) TPI value obtained during the simulation with the regulator i.
input	A list that stores the input parameters used to generate the database.

Note

The computation of the TPI and DPI databases is time-consuming as it requires many systems of differential equations to be solved. It may take several hours to build a database for a hundred genes.

Author(s)

Julien Lavenus <jl.tdcor@gmail.com>

See Also

See also UpdateTPI, TDCor-package.

Examples

```
## Not run:
# Load the lateral root transcriptomic dataset
data(LR_dataset)

# Load the vectors of gene codes, gene names and prior
data(l_genes)
data(l_names)
data(l_prior)

# Load the vector of time points for the the lateral root dataset
data(times)

# Generate a small TPI database (3 genes)

TPI_example=CalculateTPI(dataset=LR_dataset,l_genes=l_genes[4:6],
l_prior=l_prior[4:6],times=times,time_step=1,N=5000,ks_int=c(0.5,3),
kd_int=c(0.5,3),delta_int=c(0.5,3),noise=0.1,delay=3)

## End(Not run)
```

clean.at *Elimininate from a vector of gene codes the genes for which no data is available.*

Description

clean.at removes from a vector of gene codes l_genes all the elements for which no data is present in the matrix dataset.

Usage

```
clean.at(dataset,l_genes)
```

Arguments

dataset	A matrix containing the time-series transcriptomic data whose rows must be named by gene codes (like AGI gene codes).
l_genes	A character vector which contains gene codes (AGI gene codes in the case of the lateral root dataset).

Examples

```
## Load lateral root transcriptomic dataset and the l_genes vector
data(LR_dataset)
data(l_genes)

# Clean the l_gene vector
clean.at(LR_dataset,l_genes)
```

draw.profile *Plot the expression profile of a* gene *in* dataset

Description

draw.profile plots the expression profile of gene in dataset with respect to times.

Usage

```
draw.profile(dataset, gene, ...)
```

Arguments

dataset	The matrix storing the time-serie transcriptomic data.
gene	The AGI code of the gene of interest.
...	Additional arguments to be passed to the function: • col: String. Color of the curve. • type: String. Type of curve. "l", lines; "p", points; "b", both etc... For more information see the help file of the plot R function. • main: String. Title of the graph.

Author(s)

Julien Lavenus (<jl.tdcor@gmail.com>)

Examples

```
# draw the profile of GATA23 in the LR dataset

data(LR_dataset)
data(times)
draw.profile(LR_dataset,"AT5G26930",col="blue",main="GATA23")
```

---

estimate.delay *Estimate the time shift between two gene profiles and make a plot*

Description

estimate.delay computes the delay/time shift between two gene expression profiles contained in dataset. It returns a list with one or two estimated time shifts and their associated correlation. By default the function also returns a plot composed of four panels which show in more details how these estimate were obtained. This can help the user finding the appropriate parameter values to be used with the TDCOR main function. For more details see below.

Usage

```
estimate.delay(dataset, tar, reg, times, time_step, thr_cor, tol,
delaymax, delayspan, ...)
```

Arguments

dataset	Numerical matrix storing the non-log2 transcriptomic data (average of replicates). The rows of this matrix must be named by gene codes (e.g. the AGI gene code for Arabidopsis datasets). The columns must be organized in chronological order from the left to the right.
tar	The gene code of the gene to be regarded as the target.
reg	The gene code of the gene to be regarded as the regulator.

`times` A numerical vector containing the successive times (in hours) at which the samples were collected to generate the time-series transcriptomic dataset.

`time_step` A positive number corresponding to the time step (in hours) i.e. the temporal resolution at which the gene profiles are analysed.

`thr_cor` A number between 0 and 1 corresponding to the threshold on Pearson's correlation. The delay will be computed only if the absolute correlation between the profiles is higher than this threshold. Otherwise the genes are considered to have profiles that are too dissimilar, and computing the time shift would not make any sense.

`tol` The tolerance threshold for the score. The score is a positive number used to rank the time shift estimates. The best score possible for a time shift estimate is 0. If the score is above the tolerance threshold, the time shift estimate will be ignored.

`delaymax` The maximum time shift possible for a direct interaction (in hours).

`delayspan` The maximum time shift (in hours) which will be analysed. It should be high enough for the time shift estimation process to be successful but relatively small in comparison to the overall duration of the time series. (e.g. for the LR dataset which has data spanning over 54 hours, `delayspan` was set to 12 hours).

`...` Additional optional arguments.

- `make.graph`: A boolean. Set to FALSE to prevent the function from generating a graph.
- `tar.name`: A string. "Everyday name" of the target. This name will be used in the plots instead of the default value (gene code).
- `reg.name`: A string. "Everyday name" of the regulator. This name will be used in the plots instead of the default value (gene code).
- `main`: A string. Main title of the plot. By default the title of the plot is automatically generated from `tar.name` and `reg.name`.

Details

Negative time shifts occur when the gene which the user set as being the regulator could actually be the target. When two time shifts are returned, one is necessarily positive and the other is negative. When the only time shift estimate is zero, the function does not return any estimate.

The function automatically guess the sign of the potential interaction (stimulatory or inhibitory) and adapt the analysis based on it. The sign of the potential interaction is indicated in the main title of the graph by (+) or (-). When both types of interaction are possible, the function generates two graphs (one for each sign).

The function returns by default a graph composed of four panels. The top panel shows the spline functions of the two normalised expression profiles with respect to time. The second panel consists of the plots of the F1 and F2 functions with respect to the time shift (mu). The third one is for the F3 and F4 functions. All of these four functions aim at estimating the time shift between the two expression profiles by minimizing a distance-like measurement. But they each do it in a slightly different manner. F1 and F3 use Pearson's correlation as a measure of distance while F2 and F4

use the sum of squares. Moreover F1 and F2 measure the distances directly between the spline functions while F3 and F4 do it between the first derivatives of these functions. The vertical red and purple lines in the second and third panel indicate the position of the respective maximum or minimum of the functions. In the fourth and last panel, the final score function is plotted. This score is computed for each possible time shift analysed by combining the four above-mentionned functions. The green horizontal line indicate the position of the tolerance threshold (`tol`) above which time shift estimates are rejected. The vertical dark grey line(s) represent(s) the position of the final estimated time shift(s). All these lines necessarily fall into regions where the score function is below the threshold (painted in light green). Other vertical light grey line(s) may indicate other time shift estimate(s) that have a score above the tolerance threshold and were therefore rejected.

Value

The function returns a list. The first element (delay) is a numerical vector containing the time-shift estimate(s). The second element (correlation) is another numerical vector containing the associated correlation. The function also returns a graph as explained above.

Author(s)

Julien Lavenus <jl.tdcor@gmail.com>

Examples

```
# Load the data

data(LR_dataset)
data(l_genes)
data(l_names)
data(times)

# Estimate the time shift between LBD16 and PUCHI (one time shift estimate returned)

estimate.delay(dataset=LR_dataset, tar=l_genes[which(l_names=="PUCHI")],
reg=l_genes[which(l_names=="LBD16")], times=times, time_step=1, thr_cor=0.7,
tol=0.15, delaymax=3, delayspan=12, reg.name="LBD16",tar.name="PUCHI")

# Estimate the time shift between ARF8 and PLT1 (two time shift estimates returned)

estimate.delay(dataset=LR_dataset, tar=l_genes[which(l_names=="PLT1")],
reg=l_genes[which(l_names=="ARF8")], times=times, time_step=1, thr_cor=0.7,
tol=0.15, delaymax=3, delayspan=12, reg.name="ARF8",tar.name="PLT1")
```

`LR_dataset` *Lateral root transcriptomic dataset*

Description

LR_dataset is a matrix of dimension 15240 lines x 18 columns. It stores a time-series transcriptomic dataset following the changes occuring in a young Arabidopsis root during the formation of a lateral root. To generate this dataset, lateral root formation was locally induced by a gravistimulus at t=0 and the stimulated part of the roots was collected every 3 hours from 6 hours to 54 hours. The transcriptomes were analyzed using the ATH1 affymetrix chip. For time point 0, unstimulated young primary root was taken as a control. Importantly, the transcript accumulation levels stored in this dataset are non-log2 values.

Usage

```
data("LR_dataset")
```

Details

The experiment spanned 54 hours in order to cover all aspects of lateral root development. Using this method, lateral root initiation (the first pericycle divisions) occurs synchroneously in all stimulated roots around 12 hours after stimulation and the fully formed lateral root emerges from the parental root around 45 hours. The dataset contains data for all significantly differentially expressed genes.

Each column is the average of 4 independent replicates. The columns are organized in the following order: 0, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51 and 54 hours. Each line of the matrix is labbelled with an AGI gene code (Arabidopsis Genome Initiative gene code).

Source

Voss *et al.*, Lateral root organ initiation re-phases the circadian clock in Arabidopsis thaliana. Nature communication, *in revision*.

Examples

```
# Load the dataset
data(LR_dataset)

# Have a look at the first rows
head(LR_dataset)
```

`l_genes` *l_genes*

Description

Character vector containing the AGI gene codes of the genes used to reconstruct the network in Lavenus et al. 2015, Plant Cell.

Usage

```
data("l_genes")
```

Examples

```
# Load the vector
data(l_genes)

# Have a look at it
l_genes
```

`l_names` *l_names*

Description

Character vector containing the 'everyday names' of the genes used to reconstruct the network in Lavenus et al. 2015, Plant Cell.

Usage

```
data("l_names")
```

Examples

```
# Load the vector
data(l_names)

# Have a look at it
l_names
```

`l_prior` *l_prior*

Description

Vector containing the prior associated with the genes included in the network reconstruction in Lavenus et al. 2015, Plant Cell.

By defining the `l_prior` vector, the user defines which genes should be regarded as positive regulators, which others as negative regulators and which can only be targets. The prior code is defined as follow: -1 for negative regulator; 0 for non-regulator (target only); 1 for positive regulator; 2 for both positive and negative regulator. The i-th element of the vector is the prior to associate to the i-th gene in `l_genes`.

Usage

```
data("l_prior")
```

Examples

```
# Load the vector
data(l_prior)

# Have a look at it
l_prior
```

`shortest.path` *Calculate the shortest path linking every pairs of nodes in the network*

Description

`shortest.path` computes the shortest influence path (in number of edges) linking every possible regulator/target pairs in the network.

Usage

```
shortest.path(bootstrap, BS_thr)
```

Arguments

`bootstrap`	A square numerical matrix representing a network. The element [i,j] of this matrix is the signed bootstrap value for the edge 'gene j to gene i'. The sign of this element indicates the sign of the predicted interation (i.e. whether it is inhibitory or stimulatory) and the absolute value of the element is the bootstrap.
`BS_thr`	Minimum bootstrap threshold for an edge to be taken into consideration in the analysis. The edges with bootstrap values below this threshold are ignored.

Details

The paths are signed in order to keep track of the type of influence that the genes have on each other. If a path leads to the inhibition of a gene by another, `shortest.path` will return a negative number for this "pair" (Note that the (i,j) pair is not regarded as being the same than the (j,i) pair). Because the network is directed, edges can only be followed in one direction: from the regulator to the target. Hence if the network contains an edge from gene i to gene j, the length of the shortest path from i to j is 1 edge and therefore the function returns either 1 or -1 (depending on the sign of the interaction) for the length of i to j path. In absence of feedback loops between i and j, the network does not contain any path from gene j to gene i. In this case `shortest.path` shall return 0 for the length of the j to i path. Otherwise it will return the minimum number of edges to follow to go from j to i.

Value

shortest.path returns a list containing two matrices.

SP	A square numerical matrix. The element [i,j] stores the signed shortest path from gene j to gene i. The sign indicates of the type of regulatory influence (stimulatory or inhitory) that gene j has on gene i through the shortest path.
BS	A square numerical matrix. The element [i,j] stores the geometric mean of the bootstrap values of the edges located on the shortest path from gene j to gene i.

Author(s)

Julien Lavenus <jl.tdcor@gmail.com>

Examples

```
## Example with a 3-genes network where gene A upregulates B which upregulates A; and C represses B.
## the three edges have different bootstrap values (100, 60 and 55)

network=data.frame(matrix(c(0,100,0,0,60,0,0,-55,0),3,3))
names(network)=c("gene A","gene B","gene C")
rownames(network)=c("gene A","gene B","gene C")

shortest.path(as.matrix(network),1)
```

TDCOR *The TDCOR main function*

Description

This is the main function to run the TDCOR algorithm and reconstruct the gene network topology.

Usage

```
TDCOR(dataset,l_genes, TPI, DPI, ...)
```

Arguments

dataset	Numerical matrix storing the non-log2 transcriptomic data (average of replicates). The rows of this matrix must be named by gene codes (e.g. the AGI gene code for Arabidopsis datasets). The columns must be organized in chronological order from the left to the right.
l_genes	A character vector containing the (AGI) gene codes of the genes one wishes to build the network with (gene codes -e.g. "AT5G26930"- by opposed to gene names -e.g."GATA23"- which are provided by the optional l_names argument).

TPI — A TPI database generated by CalculateTPI which contains some necessary statistical information for triangle motifs pruning. In particular it must have an entry for all the regulators included in the network analysis. The TPI database may also contain data for genes that are not included in l_genes.

DPI — A DPI database generated by CalculateDPI which contains some necessary statistical information for diamond motifs pruning. In particular it must have an entry for all the regulators included in the network analysis. The DPI database may also contain data for genes that are not included in l_genes.

... — Additional arguments to be passed to the TDCOR function (Some are necessary if dataset is not the LR dataset):

- l_names: A character vector containing the 'everyday names' of the genes included in the analysis (e.g. "LBD16") . These are the names by which genes will be refered as to in the final network table. Gene names must be unique; repeats of the same name are not allowed. If no l_names parameter is given, it is by default equal to l_genes. The i-th element of the vector l_names contains the 'everyday name' of the i-th gene in l_genes.
- l_prior: A numerical vector containing the prior information on the genes included in the analysis. By defining the l_prior, the user defines which genes are positive regulators, which are negative regulators and which can only be targets. The prior code is defined as follow: -1 for negative regulator; 0 for non-regulator (target only); 1 for positive regulator; 2 for genes that can act as both positive and negative regulators. If no l_prior parameter is provided, it is by default equal to a vector of 2s, meaning that all genes are regarded as being both potential activators and repressors. The i-th element of the vector l_prior contains the prior to associate to the i-th gene in l_genes.
- times: A numerical vector containing the successive times (in hours) at which the samples were collected to generate the time-series transcriptomic dataset.If no times parameter is given, it is by default equal to the times parameter used for the lateral root transcriptomic dataset.
- n0: An integer corresponding to the number of iterations to be performed in the external bootstrap loop. At the beginning of every iteration of this external loop, new random parameter values are sampled in the user-defined bootstrapping interval. If no n0 parameter is given, it is by default equal to 1000.
- n1 : An integer corresponding to the number of iterations to be performed in the internal bootstrap loop. In this loop parameter values are kept the same but the order of node analysis is randomized at each iteration. If no n1 parameter is provided, it is by default equal to 10.
- time_step: A positive number corresponding to the time step (in hours) i.e. the temporal resolution at which the gene profiles are analysed. If no time_step parameter is provided, it is by default equal to 1 hour.
- delayspan: A positive number. It is the maximum time shift (in hours) which is analysed. It should be high enough for the time shift estimation process to be successful but argueably relatively small in comparison to the overall duration of the time series. (e.g. for the LR dataset which has data spanning over 54 hours, delayspan was set to 12 hours).

- `tol`: A strictly positive number corresponding to the tolerance threshold on the final score of time shift estimates. The score is a positive number that measures the "level of disagreement" between the four time shift estimators. A time shift estimate is regarded as meaningful if it scores lower than the `tol` threshold. If all four estimators agree on a certain value of time shift, the estimate obtains the best possible score, which is 0. Increasing `tol` make the time shift estimation process LESS stringent. For more information see `estimate.delay`.
- `delaymin`: A numerical vector containing one or two positive elements corresponding to the boundaries of the boostrapping interval for the minimum time shift above which putative interactions are regarded as possible. If no `delaymin` parameter is provided, it is by default equal to 0 hour. Gene pairs with time shift lower than or equal to `delaymin` are regarded as co-regulation and are therefore not included in the network.
- `delaymax`: A numerical vector containing one or two positive elements corresponding to the boundaries of the boostrapping interval for the maximum time shift above which putative interactions are regarded as indirect. If no `delaymax` parameter is provided, it is by default equal to 3 hours. Putative indirect interactions are included in the network only when the putative target is not predicted any direct regulator.
- `thr_cor`: A numerical vector containing one or two positive elements between 0 and 1 corresponding to the boundaries of the boostrapping interval for the threshold of Pearson's correlation. A gene pair is included in the preliminary network only if the correlation between the profiles (with the time shift correction) is higher than or equal to the `thr_cor` threshold. If no `thr_cor` parameter is provided, it is by default equal to [0.7;0.9]. Note that increasing `thr_cor` makes the correlation filter MORE stringent.
- `delay`: A positive number corresponding to the most likely time shift (in hours) one could expect between the profile of a regulator and the profile of its direct targets. This parameter enables one to generate the reference profiles of the ideal regulator when calculating the index of directness (ID). If no `delay` parameter is provided, it is by default equal to 3 hours. Note that similar parameters serving the same purpose are also used to calculate the triangle pruning index and the diamond pruning index. But TDCOR reads the value to use for calculating those indices directly from the TPI and DPI databases (for consistency reasons).
- `thr_ind1`: A numerical vector containing one or two positive elements corresponding to the boundaries of the boostrapping interval for the index of directness (ID) lower threshold. Gene pairs showing an ID below this threshold will be regarded as co-regulation and therefore eliminated from the network. If no `thr_ind1` parameter is provided, it is by default equal to 0.5. Reminder: For direct interaction one expects ID values around 1. For indirect interactions one expect values greater than 1. For co-regulated genes, ID should be smaller than 1. Note that increasing `thr_ind1` makes the ID-based "anti-coregulation filter" MORE stringent.
- `thr_ind2`: A numerical vector containing one or two positive elements corresponding to the boundaries of the boostrapping interval for the index of directness (ID) upper threshold. Putative interactions showing an ID above

this threshold are regarded as indirect. If no thr_ind2 parameter is provided, it is by default equal to 4.5. Reminder: For direct interaction one expects ID values around 1. For indirect interactions one expect values greater than 1. For co-regulated genes, ID should be smaller than 1. Note that increasing thr_ind2 makes the ID-based filter against indirect interactions LESS stringent.

- thr_overlap : A numerical vector containing one or two positive elements smaller than 1. These correspond to the boundaries of the boostrapping interval for the index of overlap. If no thr_overlap parameter is provided, it is by default equal to [0.5,0.6]. Note that increasing thr_overlap makes the overlap filter MORE stringent. This filter aims at removing unlikely negative interactions where the putative regulator switches on too late to downregulate the putative target. Keep in mind that the filter is sensitive to the noise level in the data. It should only be used if the data has a very low level of noise. To inactivate the filter set the thr_overlap parameter to 0.
- thrpTPI: A numerical vector containing one or two positive numbers smaller or equal to 1 in increasing order. These correspond to the boundaries of the boostrapping interval for the probability threshold used in the triangle filter. If no thrpTPI parameter is provided, it is by default equal to [0.5,0.75]. Note that increasing thrpTPI makes the triangle filter LESS stringent.
- thrpDPI: A numerical vector containing one or two positive numbers smaller or equal to 1 in increasing order. These correspond to the boundaries of the boostrapping interval for the probability threshold used in the diamond filter. If no thrpTPI parameter is provided, it is by default equal to [0.8,0.9]. Note that increasing thrpDPI makes the diamond filter LESS stringent.
- thr_isr: A numerical vector containing one or two positive elements corresponding to the boundaries of the boostrapping interval for the threshold of the index of directness above which the gene is predicted to negatively self-regulate. Genes will be predicted to positively self-regulate if the index of directness is smaller than 1/thr_isr. If no thr_isr parameter is provided, it is by default equal to [3,6]. Note that increasing thr_isr makes the search for self-regulating genes MORE stringent.
- search.EP: A boolean to control whether Master-Regulator-Signal-Transducer (MRST) or signal Entry Point (EP) should be looked for or not. (If yes, set on TRUE which is the default value)
- thr_bool_EP: A number between 0 and 1 used as threshold to convert normalized expression profiles (values between 0 and 1) into boolean expression profiles (values equal to 0 or 1). If no thr_bool_EP parameter is provided, it is by default equal to 0.8. The conversion of the continuous profiles into boolean profiles is part of the process of MRST analysis.
- MinTarNumber: An integer. Minimum number of targets a regulator should have in order to be regarded as a potential MRST. If no MinTarNumber parameter is provided, it is by default equal to 5. Note that increasing MinTarNumber makes the search for MRST genes MORE stringent.
- MinProp: A number between 0 and 1. Minimum proportion of targets which are not at steady state at t=0 that a regulator should have in order to be regarded as a potential MRST. If no MinProp parameter is provided, it

is by default equal to 0.75. Note that increasing MinProp makes the search for MRST genes MORE stringent.

- MaxEPNumber: An integer. Maximum number of MRST that can be predicted at each iteration. If no MaxEPNumber parameter is provided, it is by default equal to 1.
- regmax: An integer. Maximum number of regulators that a target may have. If no regmax parameter is provided, it is by default equal to 6.
- outfile_name: A string. Name of the file to print the network table in. By default it is "TDCor_output.txt".

Details

The default values are certainly not the best values to work with. The TDCOR parameters have to be optimized by the user based on its own knowledge of the network, the quality of the data etc... Because TDCOR works by pruning interactions, it is probably easier (as a first go) to optimize the parameter values following the order of the filters.

Before starting inactivate all the filters using the less stringent parameter values possible or for the MRST filter by setting search.EP to FALSE. You should as well set the bootstrap parameters to a relatively low value (e.g. n0=100 and n1=1). Hence the runs will be quick and you will be able to rapidly assess whether the changes you made in the parameter values were a good thing.

Start by optimizing the parameters involved in time shifts estimation. That is to say, essentially delayspan, time_step, tol and delaymax. The latter (together with delaymin) is a biological parameters and the range of possible values is argueably limited. Though they ought to be adapted to the organism (e.g. in prokaryotes, the delays are extremely short since polysomes couple transcription and translation). Note that the estimate.delay function can be very helpful to optimize these various parameters thanks to the visual output. Use it with pairs of genes that have been shown to interact directly or indirectly in your system and for which the relationship in the dataset in clearly linear. For network reconstruction with TDCor, good time shift estimation is absolutely crucial. Once this is done, proceed with optimizing the threshold for correlation thr_cor and the thresholds on the index of directness (thr_ind1, thr_ind2). Then optimize the parameters of the triangle and diamond pruning filters (thrpTPI and thrpDPI). You may have to try a couple of different TPI and DPI databases (i.e. databases built with different input parameters). In particular increasing the noise level when generating these database enables one to decrease the stringency of the triangle and diamond filters, when increasing the thrpTPI and thrpDPI value is not sufficient. Subsequently fine-tune the parameters of the MRST filter (thr_bool_EP, MinTarNumber, MinProp, MaxEPNumber) if you want it on. Remember to set search.EP back to TRUE first. Next optimize thr_isr (self-regulation). Finally, restrict the number of maximum regulators if necessary (regmax).

Value

The TDCOR main function returns a list containing 7 elements

input	A list containing the input parameters (as a reminder).
intermediate	A list containing three intermediate matrices. mat_cor is the matrix that stores the correlations, mat_isr stores the indices of self-regulations and mat_overlap contains the indices of overlap.

network	A matrix containing the network. The element [i,j] of this matrix contains the bootstrap value for the edge "gene j to gene i". The sign indicates the sign of the predicted interaction.
ID	A matrix containing the computed indices of directness (ID). The element [i,j] contains the ID for the edge "gene j to gene i".
delay	A matrix containing the computed time shifts. The element [i,j] of this matrix contains the estimated time shift between the profile of gene j and the profile of gene i.
EP	A vector containing the bootstrap values for the MRST predictions.
predictions	The edge predictions in the form of a table. The columns are organized in following order: Regulator name, Type of interaction (+ or-), Target name, Bootstrap, Index of Directness, Estimated time shift between the target and regulator profiles.

The table of predictions (without header) and the input parameters are printed at the end of the run in two separate text files located in the current R working directory (If you are not sure which directory this is, use the command `getwd()`).

Note

For a parameter to be involved in the bootstrapping process, one must feed the function a vector containing two values as input. These two values are respectively the lower and upper boundaries of the bootstrapping interval. If one chooses not to use a parameter for bootstrapping, one can either feed the function an input vector containing twice the same value, or only one value.

Author(s)

Julien Lavenus <jl.tdcor@gmail.com>

References

Lavenus *et al.*, 2015, The Plant Cell

See Also

See also `CalculateDPI`, `CalculateTPI`, `UpdateDPI`, `UpdateTPI`, `TDCor-package`.

Examples

```
## Not run:
# Load the lateral root transcriptomic dataset
data(LR_dataset)

# Load the vectors of gene codes, gene names and prior
data(l_genes)
data(l_names)
data(l_prior)

# Load the vector of time points for the LR_dataset
```

```
data(times)

# Generate the DPI databases

DPI15=CalculateDPI(dataset=LR_dataset,l_genes=l_genes,l_prior=l_prior,
times=times,time_step=1,N=10000,ks_int=c(0.5,3),kd_int=c(0.5,3),delta_int=c(0.5,3),
noise=0.15,delay=3)

# Generate the TPI databases

TPI10=CalculateTPI(dataset=LR_dataset,l_genes=l_genes,
l_prior=l_prior,times=times,time_step=1,N=10000,ks_int=c(0.5,3),
kd_int=c(0.5,3),delta_int=c(0.5,3),noise=0.1,delay=3)

# Check/update if necessary the databases (Not necessary here though.
# This is just to illustrate how it would work.)

TPI10=UpdateTPI(TPI10,LR_dataset,l_genes,l_prior)
DPI15=UpdateDPI(DPI15,LR_dataset,l_genes,l_prior)


### Choose your TDCOR parameters ###

# Parameters for time shift estimatation
# and filter on time shift value
ptime_step=1
ptol=0.13
pdelayspan=12
pdelaymax=c(2.5,3.5)
pdelaymin=0

# Parameter of the correlation filter
pthr_cor=c(0.65,0.8)

# Parameters of the ID filter
pdelay=3
pthr_ind1=0.65
pthr_ind2=3.5

# Parameter of the overlap filter
pthr_overlap=c(0.4,0.6)

# Parameters of the triangle and diamond filters
pthrpTPI=c(0.55,0.8)
pthrpDPI=c(0.65,0.8)
pTPI=TPI10
pDPI=DPI15

# Parameter for identification of self-regulations
pthr_isr=c(4,6)

# Parameters for MRST identification
pMinTarNumber=5
```

```
pMinProp=0.6

# Max number of regulators
pregmax=5

# Bootstrap parameters
pn0=1000
pn1=10

# Name of the file to print network in
poutfile_name="TDCor_output.txt"

### Reconstruct the network ###

tdcor_out= TDCOR(dataset=LR_dataset,l_genes=l_genes,l_names=l_names,n0=pn0,n1=pn1,
l_prior=l_prior,thr_ind1=pthr_ind1,thr_ind2=pthr_ind2,regmax=pregmax,thr_cor=pthr_cor,
delayspan=pdelayspan,delaymax=pdelaymax,delaymin=pdelaymin,delay=pdelay,thrpTPI=pthrpTPI,
thrpDPI=pthrpDPI,TPI=pTPI,DPI=pDPI,thr_isr=pthr_isr,time_step=ptime_step,
thr_overlap=pthr_overlap,tol=ptol,MinProp=pMinProp,MinTarNumber=pMinTarNumber,
outfile_name=poutfile_name)


## End(Not run)
```

TF *Table of 1834 Arabidopsis Transcription factors*

Description

TF is a dataframe with two columns. The first column contains the AGI gene code of 1834 genes encoding Arabidopsis transcription factors. The second column contains the associated gene names.

Usage

```
data("TF")
```

Source

Data published on the Agris database website (http://arabidopsis.med.ohio-state.edu/AtTFDB/).

References

Davuluri *et al.* (2003), **AGRIS: Arabidopsis Gene Regulatory Information Server, an information resource of Arabidopsis cis-regulatory elements and transcription factors**, BMC Bioinformatics, **4**:25

Examples

```
# Load the database
data(TF)

# Obtain the transcription factors for which data is available in the LR dataset
# i.e. present on ATH1 chip and differentially expressed.
data(LR_dataset)
clean.at(LR_dataset,TF[,1])
```

`times` *The* `times` *vector to use with the lateral root dataset*

Description

Contains the times (in hours) at which the samples were collected to generate the Lateral Root transcriptomic dataset (`data(LR_dataset)`)

Usage

```
data("times")
```

Examples

```
# Load the vector associated with the LR dataset
data(times)

# Have a look at it
times
```

`UpdateDPI` *Update or check the DPI database*

Description

`UpdateDPI` analyzes the DPI database and add new entries into it if it does not contain all the necessary data for reconstructing the network with the genes listed in the vector `l_genes`.

Usage

```
UpdateDPI(DPI,dataset,l_genes, l_prior)
```

Arguments

DPI	The DPI database to update or check before reconstructing the network.
dataset	Numerical matrix storing the transcriptomic data. The rows of this matrix must be named by gene codes (like AGI gene codes for Arabidopsis data).
l_genes	A character vector containing the gene codes of the genes we want to reconstruct the network with.
l_prior	A numerical vector containing the prior information on the genes included in the network recontruction. By defining the l_prior vector, the user defines which genes should be regarded as positive regulators, which others as negative regulators and which can only be targets. The prior code is defined as follow: -1 for negative regulator; 0 for non-regulator (target only); 1 for positive regulator; 2 for both positive and negative regulator. The i-th element of the vector is the prior to associate to the i-th gene in l_genes.

Value

UpdateDPI returns an updated DPI database containing data for at least all the genes in l_genes whose associated prior is not null.

Author(s)

Julien Lavenus <jl.tdcor@gmail.com>

See Also

See also CalculateDPI.

Examples

```
## Not run:
# Load the Lateral root transcriptomic dataset
data(LR_dataset)

# Load the vector of gene codes, gene names and prior
data(l_genes)
data(l_names)
data(l_prior)

# Load the vector of time points for the LR_dataset
data(times)

# Build a very small DPI database (3 genes)
DPI_example=CalculateDPI(dataset=LR_dataset,l_genes=l_genes[4:6],l_prior=l_prior[4:6],
times=times,time_step=1,N=5000,ks_int=c(0.5,3),kd_int=c(0.5,3),delta_int=c(0.5,3),
noise=0.15,delay=3)

# Add one gene in the database
DPI_example=UpdateDPI(DPI_example,dataset=LR_dataset,l_genes[4:7],l_prior[4:7])

## End(Not run)
```

UpdateTPI *Update or check the TPI database*

Description

UpdateTPI analyzes the TPI database and add new entries into it if it does not contain all the necessary data for reconstructing the network with the genes listed in the vector l_genes.

Usage

```
UpdateTPI(TPI, dataset, l_genes, l_prior)
```

Arguments

TPI	The TPI database to update or check before reconstructing the network.
dataset	Numerical matrix storing the transcriptomic data. The rows of this matrix must be named by gene codes (like AGI gene codes for Arabidopsis data).
l_genes	A character vector containing the gene codes of the genes we want to reconstruct the network with.
l_prior	A numerical vector containing the prior information on the genes included in the network recontruction. By defining the l_prior vector, the user defines which genes should be regarded as positive regulators, which others as negative regulators and which can only be targets. The prior code is defined as follow: -1 for negative regulator; 0 for non-regulator (target only); 1 for positive regulator; 2 for both positive and negative regulator. The i-th element of the vector is the prior to associate to the i-th gene in l_genes.

Value

UpdateTPI returns an updated TPI database containing data for at least all the genes in l_genes whose associated prior is not null.

Author(s)

Julien Lavenus <jl.tdcor@gmail.com>

See Also

See also CalculateTPI.

Examples

```
## Not run:
# Load the Lateral root transcriptomic dataset
data(LR_dataset)

# Load the vector of gene codes, gene names and prior
```

```
data(l_genes)
data(l_names)
data(l_prior)

# Load the vector of time points for the LR_dataset
data(times)

# Build a very small TPI database (3 genes)
TPI_example=CalculateTPI(dataset=LR_dataset,l_genes=l_genes[4:6],
l_prior=l_prior[4:6],times=times,time_step=1,N=5000,ks_int=c(0.5,3),
kd_int=c(0.5,3),delta_int=c(0.5,3),noise=0.1,delay=3)

# Add one gene in the database
TPI_example=UpdateTPI(TPI_example,dataset=LR_dataset,l_genes[4:7],l_prior[4:7])

## End(Not run)
```

Index