Documentation for inferTPBN

July 23, 2017

`reproduce` *Function to reproduce all data and plots.*

Description

Running this script reproduces all the data and plots which have also been used in the PDF report. The only thing which might differ should be the times taken to do the inference, depending on your setup.

Usage

```
reproduce()
```

Examples

```
## Not run: reproduce()
```

`FullRun` *Complete run of network generation and inference.*

Description

Automatically generate a network, generate timeseries data from it, and do inference. Saves the inferred and true networks to a subdirectory.

Usage

```
FullRun(n = 20, k = 5, p = 0.01, num.timepoints = 10,
  num.experiments = 50, topology = "homogeneous", gamma = 2.5,
  n.cores = detectCores() - 1, seed = 111, partial = FALSE,
  verbal = FALSE)
```

Arguments

n	Size of the network.
k	The number of inputs per regulatory function for each gene, if homogeneous topology is used
p	The probability of a perturbation.
num.timepoints	The number of time points per timeseries generated.
num.experiments	The number of timeseries to generate.
topology	The topology to be used. Can be "homogeneous" or "scale_free".
gamma	The exponent for the power law if topology = "scale_free".
n.cores	The number of cores to use in the inference.
seed	The random seed to use.
partial	If TRUE, a network using partial optimization should be inferred. Defaults to FALSE.
verbal	If TRUE, show progress as to which genes are currently being worked on.

See Also

setupPBN inferPBN

---

setupPBN *Generate a random TPBN and a timeseries simulated from it.*

Description

Creates a network with the desired characteristics and simulates a list of timeseries from it which can be used to test the network inference algorithm against.

Usage

```
setupPBN(n = 20, k = 5, p = 0.01, inputProbabilities = c(0.5, 0.4, 0.1),
  topology = "homogeneous", gamma = 2.5, num.timepoints = 10,
  num.experiments = 20, seed = 111, verbal = FALSE)
```

Arguments

n	Size of the network.
k	The number of inputs per regulatory function for each gene, if homogeneous topology is used.
p	The probability of a perturbation.
inputProbabilities	Probabilities for creating one, two, three, ... regulatory functions for a gene.
topology	The topology to be used. Can be "homogeneous" or "scale_free".
gamma	The exponent for the power law if topology = "scale_free".
num.timepoints	The number of time points per timeseries generated.
num.experiments	The number of timeseries to generate.
seed	The random seed to use.

Value

A list containing the generated network and a list of timeseries simulated from it.

See Also

transformNetwork simulateNetwork

Examples

```
setupPBN(n=5, k=3, topology="scale_free", gamma=2.5)
```

---

inferPBN *Infer a Threshold PBN from a list of timeseries.*

Description

This is a wrapper for the whole inference procedure and makes use of innovateGeneUntilSaturated. For every gene it infers regulatory functions until the upper bound is reached or there is no more improvement from adding more functions.

Usage

```
inferPBN(ts.multi, p = 0.01, L.max = 2, n.cores = detectCores() - 1,
  seed = 111, partial = FALSE, verbal = FALSE)
```

Arguments

ts.multi	A list of timeseries.
p	The noise probability to use for the network.
L.max	The maximum number of regulatory networks to try to infer.
n.cores	The number of cores to be used.
seed	The random seed to use.
partial	Whether or not to *also* infer a network where only partial optimization over the last threshold parameter is performed. Defaults to FALSE.
verbal	Show progress report? Defaults to FALSE.

Value

A list containing the inferred network and the time taken to perform the inference.

See Also

FullRun

Examples

```
net.true <- createNetwork(inputProbabilities=c(0.5, 0.5), n=5, k=2)
ts.multi <- simulateNetwork(net.true, 10, 20)
inferred.list <- inferPBN(ts.multi, n.cores=1)
net.inferred <- inferred.list$net.inferred
time <- inferred.list$time.complete
```

Loss — *Compute loss of the network on the timeseries.*

Description

This computes the complete loss of the network on a timeseries. If desired, the prior penalty can be included.

Usage

```
Loss(ts.multi, net, prior = TRUE)
```

Arguments

ts.multi	List of timeseries.
net	A PBN.
prior	If TRUE, loss for the prior is included. Defaults to TRUE.

Value

A float.

See Also

LossGeneTotal LossGeneTimeseries LossPriorGene

Examples

```
net <- createNetwork(inputProbabilities=1, 5, 2, "homogeneous")
ts.multi <- simulateNetwork(net, 10, 50) # Creates 50 timeseries à 10 points
loss <- Loss(ts.multi, net)
```

createNetwork — *Create random threshold PBN.*

Description

This first uses its parameters to let BoolNet create a network with the desired nodes, number of edges, topology, etc. It then creates adds all the relevant parameters to make it usable as a TPBN.

Usage

```
createNetwork(inputProbabilities = 1, ...)
```

Arguments

inputProbabilities	List of probabilities determining how many regulatory sets there are per gene.
...	Other parameters, used by BoolNet.

Value

A network.

See Also

transformNetwork setupPBN

Examples

```
net <- createNetwork(inputProbabilities=1, 5, 2, topology="homogeneous")
```

---

transformNetwork *Add TPBN parameters to a BoolNet network.*

Description

Add perturbation probability, threshold parameters, and Boolean functions to the network, to turn it into a TPBN.

Usage

```
transformNetwork(net, p = 0.01)
```

Arguments

net	A network generated by createNetwork.
p	Noise strength.

Value

A threshold PBN.

See Also

createNetwork A aToFunction

Examples

```
# Create a BoolNet network and transform it.
net <- generateRandomNKNetwork(5, 2, "homogeneous")
net <- transformNetwork(net)
```

simulateNetwork *Simulate a threshold PBN.*

Description

From a given starting configurations, simulate the network Emax times for Tmax steps in every single timeseries.

Usage

```
simulateNetwork(net, Tmax, Emax, start = "random", start.values,
  fix.genes = -1, fix.values = -1)
```

Arguments

net	The network from which the timeseries is generated.
Tmax	The number of time points per timeseries.
Emax	The number of timeseries.
start	If "random", gene values are initialized randomly.
start.values	If start is not "random" these start values are used.
fix.genes	Vector of genes to be kept fixed at a certain level.
fix.values	Vector of values at which genes should be fixed.

Value

A list of timeseries.

Examples

```
net <- createNetwork(inputProbabilities=1, 5, 2, "homogeneous")
ts.multi <- simulateNetwork(net, 10, 50) # Creates 50 timeseries à 10 points
```

A *Add threshold parameters to network.*

Description

Threshold parameters determine whether an input will have a positive or negative relationship with the target genes.

Usage

```
A(net)
```

Arguments

net	The network to be transformed.

Value

The network with associated parameters.

aToFunction — *Turn threshold parameters into Boolean functions.*

Description

Take a network with a set of threshold parameters and compute the correct Boolean functions belonging to these parameters.

Usage

```
aToFunction(net)
```

Arguments

net	A network which has threshold parameters.

Value

A Network with logic functions associated.

Examples

```
net <- generateRandomNKNetwork(5, 2, "homogeneous")
net <- A(net)
net <- aToFunction(net)
```

predict — *Generate gene's next state given current state.*

Description

Generate gene's next state given current state.

Usage

```
predict(net, g, state)
```

Arguments

net	The network to be simulated.
g	The gene whose value is to be predicted.
state	The current state of the netwrok.

Value

The simulated value of the gene.

`innovateGene` *Greedily add new gene to regulatory function.*

Description

Find the best regulatory input given the current regulatory function.

Usage

```
innovateGene(ts.multi, net, g, partial = FALSE, verbal = TRUE)
```

Arguments

`ts.multi`	A list of timeseries.
`net`	A network.
`g`	A gene.
`partial`	Whether only the most recently added gene should have its parameter optimized. Defaults to FALSE.
`verbal`	Whether messages to indicate progress should be shown. Defaults to TRUE.

Details

Try to add regulators to the "most recent" regulatory set. Stop only when there is no improvement any longer. The choice of the gene to add is done greedily.

Value

A network.

See Also

`innovateGeneUntilSaturated` `innovateGeneFunction`

`innovateGeneUntilSaturated` *Add new genes to regulatory set until no more improvement is made.*

Description

Greedily add regulatory functions to explain a gene's time progression.

Usage

```
innovateGeneUntilSaturated(ts.multi, net, g, partial = FALSE, verbal = TRUE,
  thresh = 0.1, k.max = 3)
```

Arguments

ts.multi	A list of timeseries.
net	A network.
g	A gene.
partial	Whether optimization of threshold parameter a is done only for latest gene. Defaults to FALSE.
verbal	Whether progress messages should be displayed.
thresh	By how much the improvement has to be to count. This has to be greater than zero as otherwise new genes will always be added to the brim.
k.max	How many regulatory genes to allow maximally. To avoid excessive computation, this should be set to a small constant. Defaults to 3.

Details

Genes are added until no more improvement is made or the upper limit is reached. In biological networks, a common assumption for the an upper bound on the number of inputs is in the range 3-5.

Value

A network.

See Also

innovateGene innovateGeneFunction

---

innovateGeneFunction *Add regulatory function to a gene.*

Description

Add a new regulatory function complete with inputs.

Usage

```
innovateGeneFunction(ts.multi, net, g, partial = FALSE, verbal = TRUE,
  thresh = 0.1, k.max = 3, new.max = 1)
```

Arguments

ts.multi	A list of timeseries.
net	A network.
g	A gene.
partial	Whether optimization of threshold parameter a is done only for latest gene. Defaults to FALSE.
verbal	Whether progress messages should be displayed.
thresh	By how much the improvement has to be to count. This has to be greater than zero as otherwise new genes will always be added to the brim.
k.max	How many regulatory genes to allow maximally. To avoid excessive computation, this should be set to a small constant. Defaults to 3.
new.max	An integer indicating how many new regulatory sets can be generated at once.

Details

This starts by taking some of the probability off the previous inferred function and tries to find optimal genes and probabilities for the new function.

Value

A network.

See Also

`innovateGene` `innovateGeneUntilSaturated`

`bestPartialA` *Optimize a subset of threshold parameters.*

Description

While holding everything but the subset constant, optimize over this subset. This essentially just tries every possible combination for the values on this subset. However, zeroes are not allowed, as they have been found never to come up while making the inference much, much, slower.

Usage

```
bestPartialA(ts.multi, net, g, i, subset)
```

Arguments

`ts.multi`	A list of timeseries.
`net`	A network.
`g`	A gene.
`i`	The index of the specific regulatory function for the optimization.
`subset`	The subset of parameters over which to optimize.

Value

A vector containing the best threshold parameters found.

See Also

`bestA`

bestA — *Jointly optimize all threshold parameters.*

Description

Wrapping bestPartialA, this doesn't hold any values constant.

Usage

```
bestA(ts.multi, net, g, i)
```

Arguments

ts.multi	A list of timeseries.
net	A network.
g	A gene.
i	The index of the regulatory function.

Value

A vector containing the best threshold parameters found.

See Also

bestPartialA

bestC — *Find optimal probabilities for a set of regulatory functions.*

Description

Using standard optimization solnp, this tries to find the probabilities for a single gene which minimizes the loss function.

Usage

```
bestC(ts.multi, net, g)
```

Arguments

ts.multi	List of timeseries.
net	A network.
g	A gene.

Value

A vector containing the probabilities optimizing the loss function.

See Also

optimizeC

optimizeC — *Optimize probabilities for regulatory functions to be used.*

Description

Find best probabilities. Prune those with small values.

Usage

```
optimizeC(ts.multi, net, g, prune = TRUE, thresh = 0.1)
```

Arguments

ts.multi	A list of timeseries.
net	A network.
g	A gene.
prune	Whether or not to prune regulatory functions with low probabilities assigned to them.
thresh	The threshold below which to prune functions if prune is TRUE.

Details

Using bestC, this finds the best probabilities. Afterwards, it prunes all functions whose probabilities is below a user-defined threshold.

Value

A network whose probabilities have been optimized.

See Also

bestC

LossGeneTotal — *Compute the total loss for a gene.*

Description

This includes both the predictions on the timeseries as well as the penalty for its complexity.

Usage

```
LossGeneTotal(ts.multi, net, g)
```

Arguments

ts.multi	A list of timeseries.
net	A network.
g	A gene.

Value

A float.

Examples

```
net <- createNetwork(inputProbabilities=1, 5, 2, "homogeneous")
ts.multi <- simulateNetwork(net, 10, 50) # Creates 50 timeseries à 10 points
loss <- LossGeneTotal(ts.multi, net, 1)
```

LossGeneTimeseries *Computes the loss for a gene over a list of timeseries.*

Description

Makes use of LossGene, which computes the loss for only a single timeseries.

Usage

```
LossGeneTimeseries(ts.multi, net, g)
```

Arguments

ts.multi	A list of timeseries.
net	A network.
g	A gene.

Value

A float.

Examples

```
net <- createNetwork(inputProbabilities=1, 5, 2, "homogeneous")
ts.multi <- simulateNetwork(net, 10, 50) # Creates 50 timeseries à 10 points
loss <- LossGeneTimeseries(ts.multi, net, 1)
```

LossPriorGene *Compute the prior penalty for the network.*

Description

Can take either a BoolNet network or a TPBN and computes the penalty on the network for its complexity. The penalty is given by having to encode the number of inputs, the set of actual inputs, and the Boolean function on these inputs.

Usage

```
LossPriorGene(net, g)
```

Arguments

net	A network, either threshold PBN or BoolNet net.
g	The gene to be penalized.

Value

A float.

Examples

```
net <- createNetwork(inputProbabilities=1, 5, 2, "homogeneous")
loss <- LossPriorGene(net, 1)
```

Index

A, *5*, 6
aToFunction, *5*, 7

bestA, *10*, 11
bestC, 11, *12*
bestPartialA, 10, *11*

createNetwork, 4, *5*

FullRun, 1, *3*

inferPBN, *2*, 3
innovateGene, 8, *9, 10*
innovateGeneFunction, *8, 9*, 9
innovateGeneUntilSaturated, *3*, *8*, 8, *10*

Loss, 4
LossGene, *13*
LossGeneTimeseries, *4*, 13
LossGeneTotal, *4*, 12
LossPriorGene, *4*, 13

optimizeC, *11*, 12

predict, 7

reproduce, 1

setupPBN, *2*, 2, *5*
simulateNetwork, *3*, 6

transformNetwork, *3*, *5*, 5