Package 'ternarynet'

September 19, 2024

Version 1.49.0

Date 2024-02-19

Title Ternary Network Estimation

Description Gene-regulatory network (GRN) modeling seeks to infer dependencies between genes and thereby provide insight into the regulatory relationships that exist within a cell. This package provides a computational Bayesian approach to GRN estimation from perturbation experiments using a ternary network model, in which gene expression is discretized into one of 3 states: up, unchanged, or down). The ternarynet package includes a parallel implementation of the replica exchange Monte Carlo algorithm for fitting network models, using MPI.

Author Matthew N. McCall <mccallm@gmail.com>, Anthony Almudevar <Anthony_Alumudevar@urmc.rochester.edu>, David Burton <David_Burton@urmc.rochester.edu>, Harry Stern <harry.stern@rochester.edu>

Depends R (>= 4.0)

Imports utils, igraph, methods, graphics, stats, BiocParallel

Suggests testthat

Enhances Rmpi, snow

License GPL (>= 2)

biocViews Software, CellBiology, GraphAndNetwork, Network, Bayesian

NeedsCompilation yes

Maintainer McCall N. Matthew <mccallm@gmail.com>

git_url https://git.bioconductor.org/packages/ternarynet

git_branch devel

git_last_commit 4ec3c81

git_last_commit_date 2024-04-30

Repository Bioconductor 3.20

Date/Publication 2024-09-18

Contents

`attractorSummary` *Summarize Attractors*

Description

This function summarizes the posterior probability of possible attractors.

Usage

```
attractorSummary(tpost, post.prob.limit = 0.01, wildtype = TRUE)
```

Arguments

`tpost`	a ternaryPost object
`post.prob.limit`	the minimum posterior probability for an attractor to be listed
`wildtype`	if TRUE, the wildtype attractors are summarized; if FALSE, the perturbed attractors are summarized.

Value

The function returns a matrix of attractors and posterior probabilities for each perturbation.

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
ssObj <- matrix(c(1,1,1,0,1,1,0,0,1),nrow=3)
pObj <- matrix(c(1,0,0,0,1,0,0,0,1),nrow=3)
rownames(ssObj) <- rownames(pObj) <- colnames(ssObj) <- colnames(pObj) <- c("Gene1","Gene2","Gene3")
tnfitObj <- tnetfit(ssObj, pObj)
tnpostObj <- tnetpost(tnfitObj, mdelta=10, msample=10)
attractorSummary(tnpostObj)
```

---

graphPosterior *Network Topology*

Description

This function summarizes the topology of the ternary network using marginal edge probabilities.

Usage

```
graphPosterior(tpost)
```

Arguments

tpost	a ternaryPost object

Value

The function returns a matrix of marginal posterior probabilities of each possible network edge – rows are children and columns are parents. The first column represents no parents.

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
ssObj <- matrix(c(1,1,1,0,1,1,0,0,1),nrow=3)
pObj <- matrix(c(1,0,0,0,1,0,0,0,1),nrow=3)
rownames(ssObj) <- rownames(pObj) <- colnames(ssObj) <- colnames(pObj) <- c("Gene1","Gene2","Gene3")
tnfitObj <- tnetfit(ssObj, pObj)
tnpostObj <- tnetpost(tnfitObj, mdelta=10, msample=10)
graphPosterior(tnpostObj)
```

`parallelFit` *Fit ternary network models using parallel tempering*

Description

Fit ternary network models using parallel tempering

Usage

```
parallelFit(experiment_set,
    max_parents,
    n_cycles,
    n_write,
    T_lo,
    T_hi,
    target_score,
    n_proc,
    logfile,
    n_thread,
    init_parents,
    init_outcomes,
    exchange_interval,
    adjust_move_size_interval,
    max_states,
    callback)
```

Arguments

`experiment_set`	data frame containing five columns: i_exp (experiment index), i_node (node index), outcome (-1/0/1), value (cost for that outcome), is_perturbation (0 or 1)
`max_parents`	maximum number of parents allowed for each node
`n_cycles`	maximum number of Monte Carlo cycles
`n_write`	number of times to write output during the run
`T_lo`	T for lowest-temperature replica
`T_hi`	T for highest-temperature replica
`target_score`	target_score - run will terminate if this is reached
`n_proc`	number of replicas
`logfile`	filename for log file
`n_thread`	number of openMP threads to run per process; default=1
`init_parents`	initial parents; randomized if null
`init_outcomes`	initial outcomes; set to '.' if null
`exchange_interval`	steps between exchanges; default=1000
`adjust_move_size_interval`	steps between move size adjustment; default=7001
`max_states`	max states to propagate when testing for repetition; default=10
`callback`	callback function, should take one integer argument (the replica number), used to call set.seed with different seed for each replica

Value

The return value is a list with an element for each replica. Each element is itself a list of the best unnormalized score, normalized score (unnormalized score divided by product of number of nodes and number of experiments), list of parents for each node, and array describing the transition rule, giving the outcome of a node for each possible configuration of parent nodes.

Author(s)

Harry A. Stern and Matthew N. McCall

Examples

```
i_exp <- as.integer(c(0,0,0, 0,0,0, 0,0,0, 0,0,0,
                      1,1,1, 1,1,1, 1,1,1, 1,1,1,
                      2,2,2, 2,2,2, 2,2,2, 2,2,2,
                      3,3,3, 3,3,3, 3,3,3, 3,3,3,
                      4,4,4, 4,4,4, 4,4,4, 4,4,4,
                      5,5,5, 5,5,5, 5,5,5, 5,5,5,
                      6,6,6, 6,6,6, 6,6,6, 6,6,6,
                      7,7,7, 7,7,7, 7,7,7, 7,7,7))

i_node <- as.integer(c(0,0,0, 1,1,1, 2,2,2, 3,3,3,
                      0,0,0, 1,1,1, 2,2,2, 3,3,3,
                      0,0,0, 1,1,1, 2,2,2, 3,3,3,
                      0,0,0, 1,1,1, 2,2,2, 3,3,3,
                      0,0,0, 1,1,1, 2,2,2, 3,3,3,
                      0,0,0, 1,1,1, 2,2,2, 3,3,3,
                      0,0,0, 1,1,1, 2,2,2, 3,3,3,
                      0,0,0, 1,1,1, 2,2,2, 3,3,3))

outcome <- as.integer(c(-1,0,1, -1,0,1, -1,0,1, -1,0,1,
                      -1,0,1, -1,0,1, -1,0,1, -1,0,1,
                      -1,0,1, -1,0,1, -1,0,1, -1,0,1,
                      -1,0,1, -1,0,1, -1,0,1, -1,0,1,
                      -1,0,1, -1,0,1, -1,0,1, -1,0,1,
                      -1,0,1, -1,0,1, -1,0,1, -1,0,1,
                      -1,0,1, -1,0,1, -1,0,1, -1,0,1,
                      -1,0,1, -1,0,1, -1,0,1, -1,0,1))

value <- c(0,1,2, 0,1,2, 0,1,2, 0,1,2,
            2,1,0, 0,1,2, 0,1,2, 0,1,2,
            2,1,0, 2,1,0, 0,1,2, 0,1,2,
            2,1,0, 2,1,0, 2,1,0, 0,1,2,
            2,1,0, 2,1,0, 2,1,0, 2,1,0,
            0,1,2, 2,1,0, 2,1,0, 2,1,0,
            0,1,2, 0,1,2, 2,1,0, 2,1,0,
            0,1,2, 0,1,2, 0,1,2, 2,1,0)

is_perturbation <-
c(TRUE,TRUE,TRUE,  FALSE,FALSE,FALSE, FALSE,FALSE,FALSE, FALSE,FALSE,FALSE,
    FALSE,FALSE,FALSE,  TRUE,TRUE,TRUE, FALSE,FALSE,FALSE, FALSE,FALSE,FALSE,
    FALSE,FALSE,FALSE,  FALSE,FALSE,FALSE, TRUE,TRUE,TRUE, FALSE,FALSE,FALSE,
    FALSE,FALSE,FALSE,  FALSE,FALSE,FALSE, FALSE,FALSE,FALSE, TRUE,TRUE,TRUE,
    TRUE,TRUE,TRUE,  FALSE,FALSE,FALSE, FALSE,FALSE,FALSE, FALSE,FALSE,FALSE,
    FALSE,FALSE,FALSE,  TRUE,TRUE,TRUE, FALSE,FALSE,FALSE, FALSE,FALSE,FALSE,
    FALSE,FALSE,FALSE,  FALSE,FALSE,FALSE, TRUE,TRUE,TRUE, FALSE,FALSE,FALSE,
```

```
    FALSE,FALSE,FALSE,  FALSE,FALSE,FALSE, FALSE,FALSE,FALSE, TRUE,TRUE,TRUE)

indata <- data.frame(i_exp,i_node,outcome,value,is_perturbation)

results <- parallelFit(indata,
                       max_parents=1,
                       n_cycles=100000,
                       n_write=10,
                       T_lo=0.001,
                       T_hi=2.0,
                       target_score=0,
                       n_proc=1,
                       logfile='try.log')

lowest_temp_results <- results[[1]]

print('Unnormalized score:')
print(lowest_temp_results$unnormalized_score)

print('Normalized score:')
print(lowest_temp_results$normalized_score)

print('Parents:')
print(lowest_temp_results$parents)

print('Outcomes:')
print(lowest_temp_results$outcomes)
```

`plotFit` *Network Fit Plot*

Description

This function plots the graph corresponding to the minimum scoring network.

Usage

```
plotFit(ternaryFit, type="interactive", ...)
```

Arguments

`ternaryFit`	a ternaryFit object
`type`	the type of plot to produce. "interactive" produces a plot that can be altered in the plotting window using the tkplot function from the igraph package. "static" produces a standard plot in any R graphics device.
`...`	additional parameters passed to the plotting function

Value

A plot of the network corresponding to the minimum score (stored in the graphObjMin slot) is plotted.

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
ssObj <- matrix(c(1,1,1,0,1,1,0,0,1),nrow=3)
pObj <- matrix(c(1,0,0,0,1,0,0,0,1),nrow=3)
tnfitObj <- tnetfit(ssObj, pObj)
plotFit(tnfitObj, type="static")
```

---

plotPost *Network Posterior Plot*

Description

This function plots the graph consisting of all edges with a marginal posterior probability greater than the selected threshold.

Usage

```
plotPost(ternaryPost, threshold=0.5, type="interactive", ...)
```

Arguments

ternaryPost	a ternaryPost object
type	the type of plot to produce. "interactive" produces a plot that can be altered in the plotting window using the tkplot function from the igraph package. "static" produces a standard plot in any R graphics device.
threshold	the marginal posterior probability required for an edge to be included in the plot.
...	additional parameters passed to the plotting function

Value

A plot of the network consisting of all edges with a marginal posterior probability greater than the selected threshold.

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
ssObj <- matrix(c(1,1,1,0,1,1,0,0,1),nrow=3)
pObj <- matrix(c(1,0,0,0,1,0,0,0,1),nrow=3)
tnfitObj <- tnetfit(ssObj, pObj)
tnpostObj <- tnetpost(tnfitObj, mdelta=10, msample=10)
plotPost(tnpostObj, type="static")
```

`plotTraces` *Network Fit Traces*

Description

This function plots the trace of four model parameters.

Usage

```
plotTraces(tfit)
```

Arguments

`tfit`	a ternaryFit object

Value

The function creates a 2x2 grid of the four trace plots.

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
ssObj <- matrix(c(1,1,1,0,1,1,0,0,1),nrow=3)
pObj <- matrix(c(1,0,0,0,1,0,0,0,1),nrow=3)
tnfitObj <- tnetfit(ssObj, pObj)
plotTraces(tnfitObj)
```

predictAttractor *Predict the attractor(s) resulting from a given perturbation*

Description

This function computes the posterior probabilities of attractors reached for a given perturbation using the networks from a ternaryPost object.

Usage

```
predictAttractor(tpost, perturbations, wildtype = TRUE, verbose = FALSE)
```

Arguments

tpost	a ternaryPost object
perturbations	a list with two elements: perturbed.genes and forced.states
wildtype	if TRUE, the wildtype attractors are summarized; if FALSE, the perturbed attractors are summarized.
verbose	if TRUE, periodic reports on progress are printed.

Value

The function returns a list with two elements: \ post.prob: the posterior probability of each attractor \ attractor.summary: a single vector of steady states based on the resulting attractor

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
ssObj <- matrix(c(1,1,1,0,1,1,0,0,1),nrow=3)
pObj <- matrix(c(1,0,0,0,1,0,0,0,1),nrow=3)
rownames(ssObj) <- rownames(pObj) <- colnames(ssObj) <- colnames(pObj) <- c("Gene1","Gene2","Gene3")
tnfitObj <- tnetfit(ssObj, pObj)
tnpostObj <- tnetpost(tnfitObj, mdelta=10, msample=10)
predictAttractor(tnpostObj, list(perturbed.genes=c(1,2),forced.states=c(1,1)))
```

`simulateSteadyState` *Simulate Steady State Data*

Description

This function generates simulated steady state data from a given network.

Usage

```
simulateSteadyState(perturbationObj, tableObj, graphObj, degreeObj, wildtype=FALSE)
```

Arguments

`perturbationObj`	a matrix of perturbation experiments. Rows are genes and columns are experiments.
`tableObj`	a matrix containing the transition function tables
`graphObj`	a matrix containing the parents of each node
`degreeObj`	a vector containing the in-degree of each node
`wildtype`	if TRUE, the preturbations are assumed to be transient; if FALSE, the perturbations are assumed to be persistent.

Value

The function creates a steadyStateObj.

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
pObj <- matrix(c(1,0,0,0,1,0,0,0,1),nrow=3)
degreeObj <- c(0,1,1)
graphObj <- matrix(nrow=1,ncol=3)
graphObj[1,1] <- 0
graphObj[1,2] <- 1
graphObj[1,3] <- 2
tableObj <- matrix(nrow=3,ncol=3)
tableObj[,1] <- rep(0,3)
tableObj[,2] <- c(-1,0,1)
tableObj[,3] <- c(-1,0,1)
ssObj <- simulateSteadyState(pObj, tableObj, graphObj, degreeObj)
```

ternaryFit-class *Ternary Network Fit*

Description

This is a class representation of the output of the ternary network fitting algorithm implemented in the function tnetfit.

Creating Objects

While one can create their own objects using the function ternaryFit(), this is highly discouraged. Typically this class is created by running the tnetfit function.

Slots

perturbationObj: a matrix of perturbation experiments. Rows are genes and columns are experiments.

steadyStateObj: a matrix of steady gene expression observations from a perturbation experiment. Rows are genes and columns are experiments.

geneNames: a vector of gene names corresponding to the rows of the perturbationObj and steadyStateObj.

experimentNames: a vector of experiment names corresponding to the columns of the perturbationObj and steadyStateObj.

degreeObjMin: a vector containing the in-degree of each node in the fit achieving the minimum score

graphObjMin: a matrix containing the parents of each node in the fit achieving the minimum score

tableObjMin: a matrix containing the table in the fit achieving the minimum score

newScore: the most recent score

minScore: the minimum score

finalTemperature: the final value of the temperature parameter

traces: a dataframe contain the traces for 4 parameters

stageCount: the number of stages

xSeed: the random seed.

inputParams: the ternaryFitParameters object used.

Methods

All named elements can be accessed and set in the standard way (e.g. xSeed(object) and xSeed(object)<-).

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

tnetpost, ternaryFitParameters-class, ternaryPost-class. Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
ssObj <- matrix(c(1,1,1,0,1,1,0,0,1),nrow=3)
pObj <- matrix(c(1,0,0,0,1,0,0,0,1),nrow=3)
tnfitObj <- tnetfit(ssObj, pObj)
class(tnfitObj)
```

---

ternaryFitParameters-class

Ternary Network Fitting Parameters

Description

This is a class representation of the input parameters for the ternary network fitting algorithm implemented in the function `tnetfit`.

Creating Objects

```
ternaryFitParameters()
```

This creates a `ternaryFitParameters` object with the default fitting parameters.

Slots

`perturbationType`: this parameter currently can only be set to 1

`scoreType`: the method to score networks. Can be set to either 1 or 2, corresponding the the score types in Almudevar et al. (2011).

`backupStage`: current fit is output periodically according to this parameter

`maxStage`: the maximum number of stages permitted. Ideally, the actual number of stages required until convergence should be much less than this value.

`maxTransition`: This parameter provides an adaptive truncation of the stage sample size. The stage terminates before the specified fixed sample size if the number of transitions resulting in a strict increase of the score reaches this value. If the sampler is in steady state, then this count should be approximately half the number of transitions in which the score changes value.

`epsilon`: Convergence tolerance.

`beta0`: Algorithm terminates when this number of consecutive convergence events have occurred.

`chi0`: The target initial acceptance rate. This should be close to 1, although setting it too close will increase computation time.

`delta`: The increment change in steady state distribution between stages (as variational distance). Larger values tend to decrease computation time, but too large a value will result in spurious convergence.

`ne`: The fixed sample size (number of MCMC transitions) per stage.

`m0`: The sample size (number of transitions) used to determine the initial temperature.

`maxDegree`: Maximum number of parents per node permitted in model topology.

`pAddParent`: This is the probability of adding a parent to a randomly selected node in the proposal function.

`pExchangeParent`: This parameter gives the probability of a parent exchange in the proposal function. The AddParent operation takes precedence, so this probability should be interpreted as being conditional on the rejection of the AddParent operation.

`neighborDegree`: Number of applications of the proposal function.

`pNeighborhood`: Vector of probabilities denoted, which generates the random number of proposal function iterations. The length is one less than neighborDegree. If neighborDegree equals 1 then no iteration is performed, and this vector is ignored.

`rho`: Weight parameter for the exponential smoothing of the variance estimate. For no smoothing set to 1.

`edgePenalty`: This parameter provides a complexity penalty. This number times the number of edges is added to the score. To apply no penalty set this parameter to 0.

Methods

All named elements can be accessed and set in the standard way (e.g. `scoreType(object)` and `scoreType(object)<-`).

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

`tnetfit`, `ternaryFit-class`, `ternaryPost-class`. Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
# create an instance
ternaryFitParameters()
```

`ternaryPost-class` *Ternary Network Posterior*

Description

This is a class representation of the output of the ternary network posterior sampling algorithm implemented in the function `tnetpost`.

Creating Objects

While one can create their own objects using the function `ternaryPost()`, this is highly discouraged. Typically this class is created by running the `tnetpost` function.

Slots

`perturbationObj`: a matrix of perturbation experiments. Rows are genes and columns are experiments.

`steadyStateObj`: a matrix of steady gene expression observations from a perturbation experiment. Rows are genes and columns are experiments.

`geneNames`: a vector of gene names corresponding to the rows of the perturbationObj and steadyStateObj.

`experimentNames`: a vector of experiment names corresponding to the columns of the perturbationObj and steadyStateObj.

`scores`: the score of each sample

`degreeObjs`: the in-degree vector for each sample

`graphObjs`: the graph matrix for each sample

`tableObjs`: the table matrix for each sample

`inputParams`: the ternaryFitParameters object used

Methods

All named elements can be accessed and set in the standard way (e.g. `scores(object)` and `scores(object)<-`).

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

`tnetfit`, `ternaryFitParameters-class`, `ternaryFit-class`. Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
ssObj <- matrix(c(1,1,1,0,1,1,0,0,1),nrow=3)
pObj <- matrix(c(1,0,0,0,1,0,0,0,1),nrow=3)
tnfitObj <- tnetfit(ssObj, pObj)
tnpostObj <- tnetpost(tnfitObj, mdelta=10, msample=10)
class(tnpostObj)
```

---

tnetfit *Ternary Network Fitting*

Description

This function fits a ternary network based on perturbation experiments.

Usage

```
tnetfit(steadyStateObj, perturbationObj, params=ternaryFitParameters(),
xSeed=NA)
```

Arguments

`steadyStateObj`	a matrix of steady gene expression observations from a perturbation experiment. Rows are genes and columns are experiments.
`perturbationObj`	a matrix of perturbation experiments. Rows are genes and columns are experiments.
`params`	a ternaryFitParameters object
`xSeed`	an integer random seed. If NA, a random seed is generated.

Value

The function returns a ternaryFit object.

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
ssObj <- matrix(c(1,1,1,0,1,1,0,0,1),nrow=3)
pObj <- matrix(c(1,0,0,0,1,0,0,0,1),nrow=3)
rownames(ssObj) <- rownames(pObj) <- colnames(ssObj) <- colnames(pObj) <- c("Gene1","Gene2","Gene3")
tnfitObj <- tnetfit(ssObj, pObj)
```

---

tnetpost *Ternary Network Posterior Sampling*

Description

This function samples from the posterior density of a ternary network based on perturbation experiments.

Usage

```
tnetpost(tfit, mdelta=as.integer(10000), msample=as.integer(2000), temperatureScale=1.0, xSeed=NA
```

Arguments

tfit	a ternaryFit object
mdelta	number of transitions between samples
msample	number of samples
temperatureScale	the final temperature is multipled by this value for sampling
xSeed	an integer random seed. If NA, a random seed is generated.

Value

The function returns a ternaryPost object.

Author(s)

Matthew N. McCall and Anthony Almudevar

See Also

Almudevar A, McCall MN, McMurray H, Land H (2011). Fitting Boolean Networks from Steady State Perturbation Data, Statistical Applications in Genetics and Molecular Biology, 10(1): Article 47.

Examples

```
ssObj <- matrix(c(1,1,1,0,1,1,0,0,1),nrow=3)
pObj <- matrix(c(1,0,0,0,1,0,0,0,1),nrow=3)
rownames(ssObj) <- rownames(pObj) <- colnames(ssObj) <- colnames(pObj) <- c("Gene1","Gene2","Gene3")
tnfitObj <- tnetfit(ssObj, pObj)
tnpostObj <- tnetpost(tnfitObj, mdelta=10, msample=10)
```

Index