

Package ‘mgsa’

August 29, 2011

Imports gplots, graphics, stats, utils

Maintainer Sebastian Bauer <Sebastian.Bauer@charite.de>

License Artistic-2.0

Title Model-based gene set analysis

LazyLoad yes

Author Sebastian Bauer <Sebastian.Bauer@charite.de>, Julien Gagneur
<julien.gagneur@embl.de>

Description Model-based Gene Set Analysis (MGSA) is a Bayesian modeling approach for gene set enrichment. The package mgsa implements MGSA and tools to use MGSA together with the Gene Ontology.

Version 1.1.1

biocViews Pathways, GO

Date 2011-04-28

Depends R (>= 2.9.0), methods, gplots

Suggests DBI, RSQLite, GO.db

Collate 'dottedTable.R' 'MgsaSets-class.R' 'MgsaGoSets-class.R'
'MgsaResults-class.R' 'mgsa-methods.R' 'mgsa-package.R' 'zzz.R'

R topics documented:

mgsa-package	2
alphaMcmcPost-methods	3
alphaPost-methods	3
betaMcmcPost-methods	4
betaPost-methods	4
createMgsaGoSets	5
example-go	5

example-o	5
itemAnnotations-methods	6
itemIndices-methods	6
length-methods	7
mgsa-methods	7
MgsaGoSets-class	9
MgsaMcmcResults-class	9
MgsaResults-class	10
MgsaSets-class	11
plot,MgsaResults-method	13
pMcmcPost-methods	13
populationSize-methods	14
pPost-methods	14
readGAF	15
restarts-methods	16
setAnnotations-methods	16
setsMcmcPost-methods	17
setsResults-methods	17
show,MgsaResults-method	18
show,MgsaSets-method	18
steps-methods	19
studySetSizeInPopulation-methods	19
subMgsaSets-methods	20
Index	21

mgsa-package	<i>Model-based gene set analysis</i>
--------------	--------------------------------------

Description

Model-based Gene Set Analysis (MGSA) is a Bayesian modeling approach for gene set enrichment. The package mgsa implements MGSA and tools to use MGSA together with the Gene Ontology.

Author(s)

Sebastian Bauer <Sebastian.Bauer@charite.de>, Julien Gagneur <julien.gagneur@embl.de>

References

S. Bauer, J. Gagneur and P. N. Robinson. GOing Bayesian: model-based gene set analysis of genome-scale data. Nucleic acids research, 2010.

`alphaMcmcPost-methods`*posterior estimates of the parameter alpha for each MCMC run*

Description

Posterior estimates of the parameter alpha for each MCMC run.

Usage

```
## S4 method for signature 'MgsaMcmcResults'
alphaMcmcPost(x)
```

Arguments

x a [MgsaMcmcResults](#).

Value

alphaMcmcPost-methods: matrix: Posterior estimates of the parameter alpha for each MCMC run.

`alphaPost-methods` *Posterior for alpha*

Description

Realization values, posterior estimate and standard error for the parameter alpha.

Usage

```
## S4 method for signature 'MgsaResults'
alphaPost(x)
```

Arguments

x a [MgsaResults](#).

Value

alphaPost-methods: data.frame: realization values, posterior estimate and standard error for the parameter alpha.

`betaMcmcPost-methods`*posterior estimates of the parameter beta for each MCMC run*

Description

Posterior estimates of the parameter beta for each MCMC run.

Usage

```
## S4 method for signature 'MgsaMcmcResults'
betaMcmcPost(x)
```

Arguments

x a [MgsaMcmcResults](#).

Value

`betaMcmcPost-methods: matrix`: Posterior estimates of the parameter beta for each MCMC run.

`betaPost-methods` *Posterior for beta*

Description

Realization values, posterior estimate and standard error for the parameter beta.

Usage

```
## S4 method for signature 'MgsaResults'
betaPost(x)
```

Arguments

x a [MgsaResults](#).

Value

`betaPost-methods: data.frame`: realization values, posterior estimate and standard error for the parameter beta.

createMgsaGoSets	<i>This functions takes a 1:1 mapping of go...</i>
------------------	--

Description

This functions takes a 1:1 mapping of go.ids to items and returns a full MgsaGOSets instance. The structure of GO is gathered from GO.db. It is sufficient to specify just the directly asserted mapping (or annotation), i.e., the most specific ones. The true path rule is taken account, that is, if an item is annotated to a term then it will be also annotated to more general terms (some people prefer to say that just the transitive closure is calculated).

Usage

```
createMgsaGoSets(go.ids, items)
```

Arguments

go.ids	a character vector of GO ids (GO:00001234)
items	a vector of identifiers that are annotated to the term in the corresponding position of the go.ids vector.

example-go	<i>Example GO sets for mgsa</i>
------------	---------------------------------

Description

This data is an example GO set for mgsa.

example-o	<i>Example objects for mgsa</i>
-----------	---------------------------------

Description

This data is an example objects for mgsa.

itemAnnotations-methods

Item annotations of a MgsaSets

Description

Item annotations of a [MgsaSets](#).

Usage

```
## S4 method for signature 'MgsaSets,missing'
itemAnnotations(sets, items)
## S4 method for signature 'MgsaSets,character'
itemAnnotations(sets, items)
```

Arguments

sets	an instance of class MgsaSets .
items	character an optional vector specifying the items of interest.

Value

itemAnnotations-methods: a `data.frame`: the item annotations.

itemIndices-methods

Item indices of a MgsaSets

Description

Returns the indices corresponding to the items

Usage

```
## S4 method for signature 'MgsaSets,character'
itemIndices(sets, items)
## S4 method for signature 'MgsaSets,numeric'
itemIndices(sets, items)
```

Arguments

sets	an instance of class MgsaSets .
items	character or numeric the items of interest.

Value

itemIndices-methods: a integer: the item indices.

length-methods *Length of a MgsaSets.*

Description

Length (number of sets) of [MgsaSets](#).

Usage

```
## S4 method for signature 'MgsaSets'
length(x)
```

Arguments

x an instance of class [MgsaSets](#).

Value

integer vector.

mgsa-methods *Performs an MGSA analysis*

Description

Estimate marginal posterior of the MGSA problem with an MCMC sampling algorithm.

Usage

```
mgsa(
  o, sets, population=NULL,
  alpha=seq(0.01,0.3, length.out=10), beta=seq(0.1,0.8, length.out=10),
  p=seq( min(0.1, 1/length(sets)), min(0.3, 20/length(sets)), length.out=10),
  steps=1e6, restarts=5, threads=0
)
```

Arguments

o	The observations. It can be a numeric, integer, character or logical. See details.
sets	The sets. It can be an MgsaSets or a list. In this case, each list entry is a vector of type numeric, integer, character. See details.
population	The total population. Optional. A numeric, integer or character vector. Default to NULL. See details.

<code>alpha</code>	Grid of values for the parameter alpha. Values represent probabilities of false-positive events and hence must be in [0,1]. <code>numeric</code> .
<code>beta</code>	Grid of values for the parameter beta. Values represent probabilities of false-negative events and hence must be in [0,1]. <code>numeric</code> .
<code>p</code>	Grid of values for the parameter p. Values represent probabilities of term activity and therefore must be in [0,1]. <code>numeric</code> .
<code>steps</code>	The number of steps of each run of the MCMC sampler. <code>integer</code> of length 1. A recommended value is 1e6 or greater.
<code>restarts</code>	The number of different runs of the MCMC sampler. <code>integer</code> of length 1. Must be greater or equal to 1. A recommended value is 5 or greater.
<code>threads</code>	The number of threads that should be used for concurrent restarts. A value of 0 means to use all available cores. Default to 0.

Details

`mgsa-methods`: The function can handle items (such as genes) encoded as character or integer. For convenience `numeric` items can also be provided but these values should essentially be integers. The type of items in the observations `o`, the `sets` and in the optional population should be consistent. In the case of character items, `o` and population should be of type character and `sets` can either be an `MgsaSets` or a list of character vectors. In the case of integer items, `o` should be of type integer, `numeric` (but essentially with integer values), or logical and entries in `sets` as well as the population should be integer. When `o` is logical, it is first coerced to integer with a call on `which`. Observations outside the population are not taken into account. If population is `NULL`, it is defined as the union of all sets.

The default grid value for `p` is such that between 1 and 20 sets are active in expectation. The lower limit is constrained to be lower than 0\1 and the upper limit lower than 0\3 independently of the total number of sets to make sure that complex solutions are penalized. Marginal posteriors of activity of each set are estimated using an MCMC sampler as described in Bauer et al., 2010. Because convergence of an MCM sampler is difficult to assess, it is recommended to run it several times (using `restarts`). If variations between runs are too large (see `MgsaResults`), the number of steps (`steps`) of each MCMC run should be increased.

Value

`mgsa-methods`: An `MgsaMcmcResults` object.

References

Bauer S., Gagneur J. and Robinson P. GOing Bayesian: model-based gene set analysis of genome-scale data. Nucleic Acids Research (2010) <http://nar.oxfordjournals.org/content/38/11/3523.full>

See Also

`MgsaResults`, `MgsaMcmcResults`

Examples

```
## observing items A and B, with sets {A,B,C} and {B,C,D}
mgsa(c("A", "B"), list(set1 = LETTERS[1:3], set2 = LETTERS[2:4]))

## same case with integer representation of the items and logical observation
mgsa(c(TRUE,TRUE,FALSE,FALSE), list(set1 = 1:3, set2 = 2:4))

## a small example with gene ontology sets and plot
data(example)
fit = mgsa(example_o, example_go)
## Not run:
plot(fit)
## End(Not run)
```

MgsaGoSets-class	<i>Gene Ontology annotations</i>
------------------	----------------------------------

Description

This class represents gene ontology annotations.

Details

For now, it is identical to the parental class [MgsaSets](#).

Extends

[MgsaSets](#)

See Also

[readGAF](#)

MgsaMcmcResults-class	<i>MgsaMcmcResults-class</i>
-----------------------	------------------------------

Description

Instances of this class are used to hold the additional information that was provided by running (possibly multiple times) an MCMC algorithm.

Slots

`steps`: (numeric) how many steps per MCMC run

`restarts`: (numeric) how many MCMC runs

`alphaMcmcPost`: (matrix) posterior estimates for each MCMC run of the parameter alpha

`betaMcmcPost`: (matrix) posterior estimates for each MCMC run of the parameter beta

`pMcmcPost`: (matrix) posterior estimates for each MCMC run of the parameter p

`setsMcmcPost`: (matrix) posterior estimates for each MCMC run of the sets marginal posterior probabilities

The columns of the matrices `alphaMcmcPost`, `betaMcmcPost`, `pMcmcPost` and `setsMcmcPost` stores the posterior estimates for each individual MCMC run. The row order matches the one of the slot `alphaPost`, `betaPost`, `pPots`, and `setsResults` respectively.

Accessor methods exist for each slot.

Extends

`MgsaResults`

Methods

`steps`

`restarts`

`alphaMcmcPost`

`betaMcmcPost`

`pMcmcPost`

`setsMcmcPost`

See Also

`mgsa`

`MgsaResults-class` *Results of an MGSA analysis*

Description

The results of an MGSA analysis.

Slots

populationSize: ([numeric](#)) The number of items in the population.

studySetSizeInPopulation: ([numeric](#)) The number of items both in the study set and in the population.

alphaPost: ([data.frame](#)) with columns value, estimate and std.error.

betaPost: ([data.frame](#)) with columns value, estimate and std.error.

pPost: ([data.frame](#)) with columns value, estimate and std.error.

setsResults: ([data.frame](#)) with columns inPopulation, inStudySet, estimate and std.error.

The columns of the slots alphaPost, betaPost, and pPost contains a realization value, its posterior estimate and standard error for the parameters alpha, beta and p respectively.

The columns of the slot setsResults contains the number of items of the set in the population, the number of items of the set in the study set, the estimate of its marginal posterior probability and its standard error. The [rownames](#) are the names of the sets if available.

Because an MgsaResults is the outcome of an MGSA analysis (see [mgsa](#)), accessors but no replacement methods exist for each slot.

Methods

[populationSize](#)

[studySetSizeInPopulation](#)

[alphaPost](#)

[betaPost](#)

[pPost](#)

[setsResults](#)

[show](#) signature(= "MgsaResults"): Show an [MgsaResults](#).

[plot](#) signature(= "MgsaResults"): Plot method for MgsaResults objects

See Also

[mgsa](#)

MgsaSets-class

Sets of items and their annotations

Description

This class describes sets, items and their annotations.

Details

Internally, the method `mgsa` indexes all elements of the sets before fitting the model. In case `mgsa` must be run on several observations with the same gene sets, computations can be speeded up by performing this indexing once for all. This can be achieved by building a `MgsaSets`. In order to ensure consistency of the indexing, no `replace` method for any slot is provided. Accessors are available.

The data frames `setAnnotations` and `itemAnnotations` allow to store annotations. No constraint is imposed on the number and names of their columns.

Slots

`sets`: (`list`) A list whose elements are vector of item indices.

`itemName2ItemIndex`: (`integer`) The mapping of item names to index.

`numberOfItems`: (`integer`) How many items?

`setAnnotations`: (`data.frame`) Annotations of the sets. The `rownames` are set names.

`itemAnnotations`: (`data.frame`) Annotations of the items. The `rownames` are item names.

Methods

`initialize` signature(= "MgsaSets"): Initializes the mapping when some parameters are not specified

`itemAnnotations`

`itemAnnotations`

`setAnnotations`

`setAnnotations`

`length` signature(= "MgsaSets"): Length (number of sets) of `MgsaSets`.

`itemIndices`

`itemIndices`

`show` signature(= "MgsaSets"): Show an `MgsaSets`.

`subMgsaSets`

`mgsa`

See Also

`MgsaGoSets`, `readGAF`, `mgsa`

Examples

```
new("MgsaSets", sets=list(set1=c("a", "b"), set2=c("b", "c")))
```

```
plot,MgsaResults-method
      plot,MgsaResults-method
```

Description

Plot method for MgsaResults objects

Usage

```
## S4 method for signature 'MgsaResults'
plot(x, y, ...)
```

Arguments

x	a MgsaResults
y	unused
...	unused

```
pMcmcPost-methods  posterior estimates of the parameter p for each MCMC run
```

Description

Posterior estimates of the parameter p for each MCMC run.

Usage

```
## S4 method for signature 'MgsaMcmcResults'
pMcmcPost(x)
```

Arguments

x	a MgsaMcmcResults .
---	-------------------------------------

Value

pMcmcPost-methods: matrix: Posterior estimates of the parameter p for each MCMC run.

`populationSize-methods`*Size of the population of a MgsaResults*

Description

The size of the population on which the analysis was run.

Usage

```
## S4 method for signature 'MgsaResults'
populationSize(x)
```

Arguments

`x` a [MgsaResults](#).

Value

`populationSize-methods`: integer: the size of the population.

`pPost-methods`*Posterior for beta*

Description

Realization values, posterior estimate and standard error for the parameter p .

Usage

```
## S4 method for signature 'MgsaResults'
pPost(x)
```

Arguments

`x` a [MgsaResults](#).

Value

`pPost-methods`: `data.frame`: realization values, posterior estimate and standard error for the parameter p .

`readGAF`*Read a Gene Ontology annotation file*

Description

Creates a `MgsaGoSets` using gene ontology annotations provided by a file in GAF 1.0 or 2.0 format.

Usage

```
readGAF(filename, evidence=NULL, aspect=c("P", "F", "C"))
```

Arguments

<code>filename</code>	The name of the Gene Ontology annotation file. It must be in the GAF 1.0 or 2.0 format. It may be gzip-compressed.
<code>evidence</code>	character or <code>NULL</code> . Only annotations with evidence code in <code>evidence</code> are returned. If <code>NULL</code> (default), annotations of all evidence codes are returned.
<code>aspect</code>	character with values in P, C or F. Only annotations of the listed GO namespaces P (biological process), F (molecular function) or C (cellular component) are returned. By default, annotations of the three namespaces are returned.

Details

The function extracts from the annotation file all direct gene annotations and infers from the Gene Ontology all the indirect annotations (due to term relationships). This is done using the package `Go.db` which provides the ontology as a database and `RSQLite` for querying the database.

Value

An `MgsaGoSets` object.

References

The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 2000. The GAF file format: <http://www.geneontology.org/GO.format.annotation.shtml> GO evidence codes: <http://www.geneontology.org/GO.evidence.shtml>

See Also

[MgsaGoSets](#), [mgsa](#)

Examples

```
## parsing provided example file (yeast)
gofile = system.file("example_files/gene_association_head.sgd", package="mgsa")
readGAF(gofile)
## only annotations inferred from experiment or a direct assay
readGAF(gofile, evidence=c("EXP", "IDA"))
```

restarts-methods	<i>How many MCMC runs</i>
------------------	---------------------------

Description

how many MCMC runs.

Usage

```
## S4 method for signature 'MgsaMcmcResults'
restarts(x)
```

Arguments

x a [MgsaMcmcResults](#).

Value

restarts-methods: integer: how many MCMC runs.

setAnnotations-methods	<i>Set annotations of a MgsaSets</i>
------------------------	--------------------------------------

Description

Set annotations of a [MgsaSets](#).

Usage

```
## S4 method for signature 'MgsaSets,missing'
setAnnotations(sets, names)
## S4 method for signature 'MgsaSets,character'
setAnnotations(sets, names)
```

Arguments

sets an instance of class [MgsaSets](#).
 names character an optional vector specifying the names of interest.

Value

setAnnotations-methods: a `data.frame`: the set annotations.

setsMcmcPost-methods

posterior estimates of the the set marginal probabilities for each MCMC run

Description

Posterior estimates of the set marginal probabilities for each MCMC run.

Usage

```
## S4 method for signature 'MgsaMcmcResults'
setsMcmcPost(x)
```

Arguments

x a [MgsaMcmcResults](#).

Value

setsMcmcPost-methods: matrix: Posterior estimates of the set marginal probabilities for each MCMC run.

setsResults-methods

Posterior for each set

Description

Number of items of the set in the population, the number of items of the set in the study set, the estimate of its marginal posterior probability and its standard error.

Usage

```
## S4 method for signature 'MgsaResults'
setsResults(x)
```

Arguments

x a [MgsaResults](#).

Value

setsResults-methods: data.frame: For each set, number of items of the set in the population, number of items of the set in the study set, estimate of its marginal posterior probability and standard error.

show,MgsaResults-method

Show an MgsaResults

Description

Show an [MgsaResults](#).

Usage

```
## S4 method for signature 'MgsaResults'
show(object)
```

Arguments

object an instance of class [MgsaResults](#).

Value

an invisible NULL

show,MgsaSets-method

Show an MgsaSets

Description

Show an [MgsaSets](#).

Usage

```
## S4 method for signature 'MgsaSets'
show(object)
```

Arguments

object an instance of class [MgsaSets](#).

Value

an invisible NULL

steps-methods	<i>How many steps per MCMC run</i>
---------------	------------------------------------

Description

how many steps per MCMC run.

Usage

```
## S4 method for signature 'MgsaMcmcResults'
steps(x)
```

Arguments

x a [MgsaMcmcResults](#).

Value

steps-methods: integer: how many steps per MCMC run.

studySetSizeInPopulation-methods	<i>Size of the study set of a MgsaResults</i>
----------------------------------	---

Description

The size of the study set on which the analysis was run.

Usage

```
## S4 method for signature 'MgsaResults'
studySetSizeInPopulation(x)
```

Arguments

x a [MgsaResults](#).

Value

studySetSizeInPopulation-methods: integer: the size of the study set.

`subMgsaSets-methods`*Subset of an MgsaSets*

Description

Returns a subset of an [MgsaSets](#) in which only the required items are kept. Empty sets are removed.

Usage

```
## S4 method for signature 'MgsaSets,numeric'  
subMgsaSets(sets, items)
```

Arguments

<code>sets</code>	an MgsaSets .
<code>items</code>	numeric. The items to restrict on.

Value

`subMgsaSets-methods`: an [MgsaSets](#).

Note

TODO: do we need that method? Does it work on items or item indexes?

Index

*Topic **data**

example-go, [5](#)

example-o, [5](#)

*Topic **package**

mgsa-package, [2](#)

alphaMcmcPost, [10](#)

alphaMcmcPost

(*alphaMcmcPost-methods*), [3](#)

alphaMcmcPost, MgsaMcmcResults-method

(*alphaMcmcPost-methods*), [3](#)

alphaMcmcPost-methods, [3](#)

alphaPost, [11](#)

alphaPost (*alphaPost-methods*), [3](#)

alphaPost, MgsaResults-method

(*alphaPost-methods*), [3](#)

alphaPost-methods, [3](#)

betaMcmcPost, [10](#)

betaMcmcPost

(*betaMcmcPost-methods*), [4](#)

betaMcmcPost, MgsaMcmcResults-method

(*betaMcmcPost-methods*), [4](#)

betaMcmcPost-methods, [4](#)

betaPost, [11](#)

betaPost (*betaPost-methods*), [4](#)

betaPost, MgsaResults-method

(*betaPost-methods*), [4](#)

betaPost-methods, [4](#)

createMgsaGoSets, [5](#)

data.frame, [11](#), [12](#)

example-go, [5](#)

example-o, [5](#)

example_go (*example-go*), [5](#)

example_o (*example-o*), [5](#)

initialize, [12](#)

integer, [12](#)

itemAnnotations, [12](#)

itemAnnotations

(*itemAnnotations-methods*),

[6](#)

itemAnnotations, MgsaSets, character-method

(*itemAnnotations-methods*),

[6](#)

itemAnnotations, MgsaSets, missing-method

(*itemAnnotations-methods*),

[6](#)

itemAnnotations-methods, [6](#)

itemIndices, [12](#)

itemIndices

(*itemIndices-methods*), [6](#)

itemIndices, MgsaSets, character-method

(*itemIndices-methods*), [6](#)

itemIndices, MgsaSets, numeric-method

(*itemIndices-methods*), [6](#)

itemIndices-methods, [6](#)

length, [12](#)

length (*length-methods*), [7](#)

length, MgsaSets-method

(*length-methods*), [7](#)

length-methods, [7](#)

list, [12](#)

matrix, [10](#)

mgsa, [10–12](#), [15](#)

mgsa (*mgsa-methods*), [7](#)

mgsa, character, list-method

(*mgsa-methods*), [7](#)

mgsa, character, MgsaSets-method

(*mgsa-methods*), [7](#)

mgsa, integer, list-method

(*mgsa-methods*), [7](#)

mgsa, logical, list-method

(*mgsa-methods*), [7](#)

mgsa, numeric, list-method

(*mgsa-methods*), [7](#)

- mgsa-methods, [7](#)
- mgsa-package, [2](#)
- MgsaGoSets, [12](#), [15](#)
- MgsaGoSets (*MgsaGoSets-class*), [9](#)
- MgsaGoSets-class, [9](#)
- MgsaMcmcResults, [3](#), [4](#), [8](#), [13](#), [16](#), [17](#), [19](#)
- MgsaMcmcResults
 - (*MgsaMcmcResults-class*), [9](#)
- MgsaMcmcResults-class, [9](#)
- MgsaResults, [3](#), [4](#), [8](#), [10](#), [11](#), [13](#), [14](#), [17–19](#)
- MgsaResults (*MgsaResults-class*), [10](#)
- MgsaResults-class, [10](#)
- MgsaSets, [6–9](#), [12](#), [16](#), [18](#), [20](#)
- MgsaSets (*MgsaSets-class*), [11](#)
- MgsaSets-class, [11](#)
- numeric, [10](#), [11](#)
- plot, [11](#)
- plot (*plot, MgsaResults-method*), [13](#)
- plot, MgsaResults-method, [13](#)
- pMcmcPost, [10](#)
- pMcmcPost (*pMcmcPost-methods*), [13](#)
- pMcmcPost, MgsaMcmcResults-method
 - (*pMcmcPost-methods*), [13](#)
- pMcmcPost-methods, [13](#)
- populationSize, [11](#)
- populationSize
 - (*populationSize-methods*), [14](#)
- populationSize, MgsaResults-method
 - (*populationSize-methods*), [14](#)
- populationSize-methods, [14](#)
- pPost, [11](#)
- pPost (*pPost-methods*), [14](#)
- pPost, MgsaResults-method
 - (*pPost-methods*), [14](#)
- pPost-methods, [14](#)
- readGAF, [9](#), [12](#), [15](#)
- restarts, [10](#)
- restarts (*restarts-methods*), [16](#)
- restarts, MgsaMcmcResults-method
 - (*restarts-methods*), [16](#)
- restarts-methods, [16](#)
- rownames, [11](#), [12](#)
- setAnnotations, [12](#)
- setAnnotations
 - (*setAnnotations-methods*), [16](#)
- setAnnotations, MgsaSets, character-method
 - (*setAnnotations-methods*), [16](#)
- setAnnotations, MgsaSets, missing-method
 - (*setAnnotations-methods*), [16](#)
- setAnnotations-methods, [16](#)
- setsMcmcPost, [10](#)
- setsMcmcPost
 - (*setsMcmcPost-methods*), [17](#)
- setsMcmcPost, MgsaMcmcResults-method
 - (*setsMcmcPost-methods*), [17](#)
- setsMcmcPost-methods, [17](#)
- setsResults, [11](#)
- setsResults
 - (*setsResults-methods*), [17](#)
- setsResults, MgsaResults-method
 - (*setsResults-methods*), [17](#)
- setsResults-methods, [17](#)
- show, [11](#), [12](#)
- show (*show, MgsaResults-method*), [18](#)
- show (*show, MgsaSets-method*), [18](#)
- show, MgsaResults-method, [18](#)
- show, MgsaSets-method, [18](#)
- steps, [10](#)
- steps (*steps-methods*), [19](#)
- steps, MgsaMcmcResults-method
 - (*steps-methods*), [19](#)
- steps-methods, [19](#)
- studySetSizeInPopulation, [11](#)
- studySetSizeInPopulation
 - (*studySetSizeInPopulation-methods*), [19](#)
- studySetSizeInPopulation, MgsaResults-method
 - (*studySetSizeInPopulation-methods*), [19](#)
- studySetSizeInPopulation-methods, [19](#)
- subMgsaSets, [12](#)
- subMgsaSets
 - (*subMgsaSets-methods*), [20](#)
- subMgsaSets, MgsaSets, numeric-method
 - (*subMgsaSets-methods*), [20](#)
- subMgsaSets-methods, [20](#)
- which, [8](#)