Package 'RSiena'

July 2, 2014

Type Package

Title Siena - Simulation Investigation for Empirical Network Analysis

Version 1.1-232

Date 2013-06-18

Author Ruth Ripley, Krists Boitmanis, Tom A.B. Snijders

Depends R (>= 2.15.0)

Imports Matrix

Suggests tcltk, network, codetools, lattice, MASS, parallel, xtable,tools

SystemRequirements GNU make, tcl/tk 8.5, Tktable

Maintainer Tom A.B. Snijders <tom.snijders@nuffield.ox.ac.uk>

Description Fits models to longitudinal network data

License GPL-2

LazyLoad yes

LazyData yes

NeedsCompilation yes

BuildResaveData no

URL http://www.stats.ox.ac.uk/~snijders/siena

Repository CRAN

Date/Publication 2013-06-19 17:47:06

R topics documented:

RSiena-package *Simulation Investigation for Empirical Network Analysis*

Description

Fits statistical models to longitudinal sets of networks, and to longitudinal sets of networks and behavioral variables. Not only one-mode networks but also two-mode networks and multivariate networks are allowed. The models are stochastic actor-oriented models.

Package "RSienaTest" is the development version, and is distributed through R-Forge, see http://r-forge.r-project.org/R/?group_id=461. Package "RSiena" is the official release.

Details

Use siena07 to fit a model.

Data objects can be created from matrices and vectors using sienaDependent, coCovar etc., and finally sienaDataCreate. Another possibility (but with less flexibility) is via the Gui displayed by siena01Gui, or via sienaDataCreateFromSession.

Effects are selected using an *effects* object, which can be created using getEffects.

Control of the estimation algorithm requires a sienaAlgorithm object that defines the settings (parameters) of the algorithm,
and which can be created by sienaAlgorithmCreate (alias sienaModelCreate).

More detailed help is available in the manual which you can display using
RShowDoc("RSiena_Manual", package="RSiena")

Package:	RSiena
Type:	Package
Version:	1.1-232
Date:	2013-06-18
Depends:	R (>= 2.15.0)
Imports:	Matrix
Suggests:	tcltk, network, codetools, lattice, MASS, parallel, xtable, tools
SystemRequirements:	GNU make, tcl/tk 8.5, Tktable
License:	GPL-2
LazyData:	yes
NeedsCompilation:	yes
BuildResaveData:	no

Author(s)

Ruth Ripley, Krists Boitmanis, Tom Snijders. Contributions by Josh Lospinoso, Charlotte Greenan, Christian Steglich, Johan Koskinen, Mark Ortmann, and Nynke Niezink.

Maintainer: Tom A.B. Snijders <tom.snijders@nuffield.ox.ac.uk>

References

- Schweinberger, Michael, and Snijders, Tom A.B. (2007). Markov models for digraph panel data: Monte Carlo-based derivative estimation. *Computational Statistics and Data Analysis* 51, 4465-4483.
- Snijders, Tom A.B. (2001). The statistical evaluation of social network dynamics. *Sociological Methodology*, 31, 361-395.
- Snijders, Tom A.B., Steglich, Christian E.G., and Schweinberger, Michael (2007). Modeling the co-evolution of networks and behavior. Pp. 41-71 in *Longitudinal models in the behavioral and related sciences*, edited by Kees van Montfort, Han Oud and Albert Satorra; Lawrence Erlbaum.
- Steglich, Christian E. G., Snijders, Tom A. B., and Pearson, Michael A. (2010). Dynamic networks and behavior: Separating selection from influence. Sociological Methodology, 40, 329-393.
- Further see the extensive manual accessible by the command
  `RShowDoc("RSiena_Manual", package="RSiena")`
  and the website http://www.stats.ox.ac.uk/~snijders/siena/ .

See Also

siena07

Examples

```
mynet1 <- sienaDependent(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip)
myeff
myalgorithm <- sienaAlgorithmCreate(nsub=3, n3=200)
ans <- siena07(myalgorithm, data=mydata, effects=myeff, batch=TRUE)
summary(ans)
```

---

allEffects *Internal data frame used to construct effect objects.*

Description

This data frame is used internally to construct effect objects.

Usage

```
data(allEffects)
```

Format

A data frame with 169 observations on the following 23 variables.

`effectGroup` a character vector

`effectName` a character vector

`functionName` a character vector

`shortName` a character vector

`endowment` a logical vector

`interaction1` a character vector

`interaction2` a character vector

`type` a character vector

`basicRate` a logical vector

`include` a logical vector

`randomEffects` a logical vector

`fix` a logical vector

`test` a logical vector

`timeDummy` a character vector, default ","

`initialValue` a numeric vector

`parm` a numeric vector

`functionType` a character vector

`period` a character vector

`rateType` a character vector

`untrimmedValue` a numeric vector

`effect1` a logical vector

`effect2` a logical vector

`effect3` a logical vector

`interactionType` a character vector

Details

Not for general user use.

References

See http://www.stats.ox.ac.uk/~snijders/siena/

coCovar *Function to create a constant covariate object*

Description

This function creates a constant covariate object from a vector.

Usage

```
coCovar(val, nodeSet='Actors')
```

Arguments

val	Vector of covariate values
nodeSet	Name of node set: character string. If the entire data set contains more than one node set, then the node sets must be specified in all data objects.

Details

When part of a Siena data object, the covariate is associated with the node set nodeSet of the Siena data object.

Value

Returns the covariate as an object of class "coCovar", in which form it can be used as an argument to SienaDataCreate.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

sienaDataCreate, varCovar, coDyadCovar, varDyadCovar

Examples

```
myconstCovar <- coCovar(s50a[,1])
```

coDyadCovar *Function to create a constant dyadic covariate object.*

Description

This function creates a constant dyadic covariate object from a matrix.

Usage

```
coDyadCovar(val, nodeSets=c("Actors", "Actors"),
    sparse=is(val,"dgTMatrix"), type=c("oneMode", "bipartite"))
```

Arguments

val	Matrix of covariate values. May be sparse, of type "dgTMatrix".
nodeSets	The name of the node sets with which this covariate is associated. If the entire data set contains more than one node set, then the node sets must be specified in all data objects.
sparse	Boolean: whether a sparse matrix or not.
type	oneMode or bipartite: whether the matrix refers to a one-mode or a bipartite (two-mode) network.

Details

When part of a Siena data object, the covariate is assumed to be associated with the node sets named in nodeSets of the Siena data object. The name of the associated node sets will only be checked when the Siena data object is created.

Value

Returns the covariate as an object of class "coDyadCovar", in which form it can be used as an argument to sienaDataCreate.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

sienaDataCreate, varDyadCovar, coCovar, varCovar

Examples

```
mydyadvar <- coDyadCovar(s503)
```

`edit.sienaEffects` *Allow editing of a sienaEffects object if a gui is available.*

Description

Interactive editor for an effects object. A wrapper to edit.data.frame.

Usage

```
## S3 method for class 'sienaEffects'
edit(name, ...)
```

Arguments

`name`	An object of class `sienaEffects`
`...`	For extra arguments (none used at present)

Details

Will be invoked by `fix(name)` for an object of class `sienaEffects`.

Value

The updated object. There is no backup copy, and the edits cannot be undone.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

`getEffects`

Examples

```
mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mybeh <- sienaDependent(s50a, type='behavior')
mycovar <- coCovar(rnorm(50))
mydyadcovar <- coDyadCovar(matrix(as.numeric(rnorm(2500) > 2), nrow=50))
mydata <- sienaDataCreate(mynet1, mybeh, mycovar, mydyadcovar)
myeff <- getEffects(mydata)
## Not run:
fix(myeff)

## End(Not run)
```

---

effectsDocumentation *Function to create a table of documentation of effect names, short names etc.*

Description

Produces a table of the shortnames and other information for effects, either in html or latex.

Usage

```
effectsDocumentation(effects = NULL, type = "html", display = (type=="html"),
  filename = ifelse(is.null(effects), "effects", deparse(substitute(effects))))
```

Arguments

effects	A Siena effects object, or NULL.
type	Type of output required. Valid options are "html" or "latex".
display	Boolean: should the output be displayed after creation. Only applicable to html output.
filename	Character string denoting file name.

Details

If effects=NULL, the allEffects object is written to a table, either latex or html. This table presents all the available effects present in this version of RSiena, not delimited by a particular data set. The default file name is "effects.tex" or "effects.pdf", respectively.

The table lists all effects, with their name, shortName, whether an endowment (and creation) effect exists, the value of an effect parameter - if any -, and the interactionType (which can be empty or: "ego" or "dyadic" for dependent network variables; "OK" for dependent behavior variables). The latter is important for knowing how the effects can be used in interaction effects. (See includeInteraction).

If an existing effects object is specified for effects, then all available effects in this effects object are listed. This table lists the name (i.e., dependent variable), effect name, shortName, type (rate/evaluation/endowment/creation), the variables defined as interaction1 and interaction2 (see includeEffects) that specify this effect, the value of an effect parameter - if any -, and the interactionType. The default root file name is the name of the input effects object.

Value

Nothing returned. Output files are created in the current working directory.

Author(s)

Ruth Ripley, Tom A.B. Snijders

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

getEffects, includeEffects, summary.sienaEffects, includeInteraction.

Examples

```
## Not run: effectsDocumentation()
```

---

getEffects *Function to create a Siena effects object*

Description

Creates a basic list of effects for all dependent variables in the input siena object.

Usage

```
getEffects(x, nintn = 10, behNintn=4, getDocumentation=FALSE)
```

Arguments

x	an object of class 'siena' or 'sienaGroup'
nintn	Number of user-defined network interaction that can later be created.
behNintn	Number of user-defined behavior interactions that can later be created.
getDocumentation	Flag to allow documentation of internal functions, not for use by users.

Details

Creates a data frame of effects for use in siena model fits. Note that the class of the return object may be lost if the data.frame is edited using fix. See fix and edit.data.frame.

Value

An object of class `sienaEffects` or `sienaGroupEffects`: this is a data frame, each part of which relates to one dependent variable in the input object, with columns

`name`	name of the dependent variable
`effectName`	name of the effect
`functionName`	name of the function
`shortName`	short name for the effect
`interaction1`	second variable to define the effect, if any
`interaction2`	third variable to define the effect, if any
`type`	"eval", "endow", "creation", or "rate"
`basicRate`	boolean: whether a basic rate parameter
`include`	boolean: include in the model to be fitted or not
`randomEffects`	boolean: random or fixed effect. Currently not used.
`fix`	boolean: fix parameter value or not
`test`	boolean: test parameter value or not
`timeDummy`	comma separated list of periods, or "all", or ',' for none – which time dummy interacted parameters should be included?
`initialValue`	starting value for estimation, also used for `fix` and `test`.
`parm`	parameter values
`functionType`	"objective" or "rate"
`period`	period for basic rate parameters
`rateType`	"Structural", "covariate", "diffusion"
`untrimmedValue`	Used to store initial values which could be trimmed
`effect1`	Used to indicate effect number in user-specified interactions
`effect2`	Used to indicate effect number in user-specified interactions
`effect3`	Used to indicate effect number in user-specified interactions
`interactionType`	Defines "dyadic" or "ego" or "OK" effects
`effectFn`	here NULL, but could be replaced by a function later
`statisticFn`	here NULL, but could be replaced by a function later
`netType`	Type of dependent variable: "oneMode", "behavior", or "bipartite"
`groupName`	name of relevant group data object
`group`	sequential number of relevant group data object in total
`effectNumber`	a unique identifier of the row

The combination of `name`, `shortName`, `interaction1`, `interaction2`, and `type` uniquely identifies any effect other than basic rate effects.The combination `name`, `shortName`, `period` and `group` uniquely identifies a basic rate effect.

A list of all effects in a given effects object (e.g., `myeff`), including their names of dependent variables, effect names, short names, and values of interaction1 and interaction2 (if any), is obtained by executing `effectsDocumentation(myeff)`.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

sienaDataCreate, effectsDocumentation

Examples

```
mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mybeh <- sienaDependent(s50a, type='behavior')
mycovar <- coCovar(rnorm(50))
mydyadcovar <- coDyadCovar(matrix(as.numeric(rnorm(2500) > 2), nrow=50))
mydata <- sienaDataCreate(mynet1, mybeh, mycovar, mydyadcovar)
myeff <- getEffects(mydata)
```

---

hn3401 — *Network data: excerpt from 'Dutch Social Behavior Data Set' of Chris Baerveldt.*

Description

Matrices N3401, N3403, N3404, N3406, and HN3401, HN3403, HN3404, HN3406 are two waves of networks for four schools (numbered 1, 3, 4, 6): there is a tie from pupil i to pupil j if i says that he/she receives and/or gives emotional support from/to pupil j. The data are part of a larger data set (see source below) and were collected under the direction of Chris Baerveldt.

Format

Adjacency matrices for the network at two time points. The matrices with name N... are the first wave, those with name HN... are the second wave.

Source

http://www.stats.ox.ac.uk/~snijders/siena/CB_data.zip

References

Houtzager, B. & Baerveldt, C. (1999). Just like Normal. A Social Network Study of the Relation between Petty Crime and the Intimacy of Adolescent Friendships. *Social Behavior and Personality* 27(2), 177-192.

Snijders, Tom A.B, and Baerveldt, Chris (2003). A Multilevel Network Study of the Effects of Delinquent Behavior on Friendship Evolution. *Journal of Mathematical Sociology* 27, 123-151.

See http://www.stats.ox.ac.uk/~snijders/siena/BaerveldtData.html

Examples

```
mynet <- sienaDependent(array(c(N3401, HN3401), dim=c(45, 45, 2)))
mydata <- sienaDataCreate(mynet)
```

`includeEffects` *Function to include effects in a Siena model*

Description

This function provides an interface to set the include column on selected rows of a Siena effects object

Usage

```
includeEffects(myeff, ..., include = TRUE, name = myeff$name[1],
 type = "eval", interaction1 = "", interaction2 = "", character=FALSE)
```

Arguments

`myeff`	a Siena effects object as created by `getEffects`
`...`	short names to identify the effects which should be included or excluded.
`include`	Boolean. default TRUE, but can be switched to FALSE to turn off an effect.
`name`	Name of network for which effects are being included. Defaults to the first in the effects object.
`type`	Type of effects to be included: "eval", "endow", "creation", or "rate".
`interaction1`	Name of siena object where needed to completely identify the effects e.g. covariate name or behavior variable name.
`interaction2`	Name of siena object where needed to completely identify the effects e.g. covariate name or behavior variable name.
`character`	Boolean: are the effect names character strings or not

Details

The effects indicated by the arguments `...`, `type`, and (if necessary) `interaction1` and `interaction2` are included or excluded from the model specified by the effects object. The names `interaction1` and `interaction2` do not refer to interactions between effects, but to dependence of effects on other variables in the data set. The arguments should identify the effects completely.

A list of all effects available in a given effects object (e.g., `myeff`), including their names of dependent variables, effect names, short names, and values of interaction1 and interaction2 (if any), is obtained by executing `effectsDocumentation(myeff)`.

The function includeEffects operates by providing an interface to set the "include" column on selected rows of the effects object, to the value requested (TRUE or FALSE).

Value

An updated version of the input effects object, with the include columns for one or more rows updated. Details of the rows altered will be printed.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

getEffects, setEffect, includeInteraction, effectsDocumentation

Examples

```
mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mybeh  <- sienaDependent(s50a, type="behavior")
mydata <- sienaDataCreate(mynet1, mybeh)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, balance)
myeff <- includeEffects(myeff, avAlt, name="mybeh", interaction1="mynet1")
myeff
```

---

includeInteraction *Function to create user-specified interactions for a Siena model.*

Description

This function allows the user to include or exclude an interaction effect in a Siena effects object.

Usage

```
includeInteraction(myeff, ..., include = TRUE, name = myeff$name[1],
    type = "eval", interaction1 = rep("", 3), interaction2 = rep("", 3),
    character = FALSE, verbose = TRUE)
```

Arguments

myeff	a Siena effects object as created by getEffects
...	2 or 3 short names to identify the effects which should be interacted.
include	Boolean. default TRUE, but can be switched to FALSE to turn off an interaction.
name	Name of dependent variable (network or behavior) for which interactions are being defined. Defaults to the first in the effects object.
type	Type of effects to be interacted.

interaction1	Vector of Siena objects where needed to completely identify the effect e.g. covariate name or behavior variable name. Trailing blanks may be omitted.
interaction2	Vector of siena objects where needed to completely identify the effect e.g. covariate name or behavior variable name. Trailing blanks may be omitted.
character	Boolean: are the effect names character strings or not
verbose	Boolean: should the print of altered effects be produced.

Details

The details provided should uniquely identify up to three effects. If so, an interaction effect will be created and included or not in the model.
Whether interactions between two or three given effects can be created depends on their `interactionType` (which can be, for dependent network variables, empty, ego, or dyadic; and for dependent behavioral variables, empty or OK). Consult the section on Interaction Effects in the manual for this. The `interactionType` is shown in the list of effects obtained from the function `effectsDocumentation`.
From the point of view of model building it is usually advisable, when including an interaction effect in a model, also to include the corresponding main effects. This is however not enforced by `includeInteraction()`.

The input names `interaction1` and `interaction2` do not themselves refer to a created interactions, but to dependence of the base effects on other variables in the data set. They are used to completely identify the effects.

A list of all effects in a given effects object (e.g., `myeff`), including their names of dependent variables, effect names, short names, and values of interaction1 and interaction2 (if any), is obtained by executing `effectsDocumentation(myeff)`.

Value

An updated version of the input effects object, with the "include" column and the "effect1" and "effect2" and possibly "effect3" columns of one row updated.
If `verbose=TRUE`, details of the fields altered will be printed.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

`getEffects`, `includeEffects`, `effectsDocumentation`

Examples

```
mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mybeh  <- sienaDependent(s50a, type="behavior")
mydata <- sienaDataCreate(mynet1, mybeh)
myeff <- getEffects(mydata)
```

```
myeff <- includeEffects(myeff, transTrip)
myeff <- includeEffects(myeff, egoX, interaction1="mybeh")
myeff <- includeInteraction(myeff, transTrip, egoX,
         interaction1=c("", "mybeh"))
myeff
```

`includeTimeDummy` *Function to include time dummy effects in a Siena model*

Description

This function specifies time heterogeneity for selected effects in a Siena model, by interacting them with time dummies, without explicitly using time-dependent covariates.

Usage

```
includeTimeDummy(myeff, ..., timeDummy="all", name=myeff$name[1], type="eval",
              interaction1="", interaction2="", include=TRUE, character=FALSE)
```

Arguments

`myeff`	A Siena effects object as created by `getEffects`.
`...`	Short names to identify the effects for which interactions with time dummies should be included or excluded. This function cannot be used for regular interaction effects.
`timeDummy`	Character string. Either "all" or the periods for which to create dummies (from 1 to (number of waves - 1)), space delimited.
`include`	Boolean. default TRUE, but can be switched to FALSE to turn off an effect.
`name`	Name of dependent network or behavioral variable for which effects are being included. Defaults to the first in the effects object.
`type`	Type of dummy effects to be interacted.
`interaction1`	Name of variable where needed to completely identify the effects e.g. covariate name or behavior variable name.
`interaction2`	Name of variable where needed to completely identify the effects e.g. covariate name or behavior variable name.
`character`	Boolean: are the effect names character strings or not

Details

The arguments (`...`, `name`, `interaction1`, `interaction2`) should identify the effects completely. See `includeEffects` and `effectsDocumentation` for more information about this.

This function operates by setting the `timeDummy` column on selected rows of a Siena effects object, thereby specifying interactions of the specified effect or effects with dummy variables for the specified periods. The `timeDummy` column of `myeff` will be set to include the values requested if `include=TRUE`, and to exclude them for `include=FALSE`.

For an effects object in which the `timeDummy` column of some of the included effects includes some or all period numbers, interactions of those effects with ego effects of time dummies for the indicated periods will also be estimated by `siena07`. For the outdegree effect this is just the ego effect of the time dummies. If ... does not include the outdegree effect, then still this ego effect will be created, but its parameter will be fixed to 0.

An alternative to the use of `includeTimeDummy` is to define time-dependent actor covariates (dummy variables or other functions of wave number that are the same for all actors), include these in the data set through `sienaDataCreate`, and include interactions of other effects with ego effects of these time-dependent actor covariates by `includeInteraction`. Using `includeTimeDummy` is easier; on the other hand, using self-defined interactions with time-dependent variables gives more control (e.g., it will allow to specify linear time dependence and test time heterogeneity for interaction effects).

Value

An updated version of `myeff`, with the `timeDummy` column for one or more rows updated. Details of the rows altered will be printed.

Author(s)

Josh Lospinoso

References

See http://www.stats.ox.ac.uk/~snijders/siena/ for general information on RSiena.

See Also

`sienaTimeTest`, `getEffects`, `includeEffects`, `effectsDocumentation`.

Examples

```
## Not run:
## Estimate a restricted model
myalgorithm <- sienaAlgorithmCreate(nsub=4, n3=1000)
mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, balance)
myeff
(ans <- siena07(myalgorithm, data=mydata, effects=myeff))

## Conduct the score type test to assess whether heterogeneity is present.
tt <- sienaTimeTest(ans)
summary(tt)

## Suppose that we wish to include a time dummy.
## Since there are three waves, the number of periods is two.
## This means that only one time dummy can be included for
## the interactions. The default is for period 2;
## an equivalent model, but with different parameters
```

```
## (that can be transformed into each other) is obtained
## when the dummies are defined for period 1.
myeff <- includeTimeDummy(myeff, density, recip, timeDummy="2")
myeff       # Note the \code{timeDummy} column.
(ans2 <- siena07(myalgorithm, data=mydata, effects=myeff))

## Re-assess the time heterogeneity
tt2 <- sienaTimeTest(ans2)
summary(tt2)

## And so on..

## End(Not run)

## A demonstration of RateX heterogeneity.
## Note that rate interactions are not implemented in general,
## but they are for Rate x coCovar.
## Not run:
myalgorithm <- sienaAlgorithmCreate(nsub=4, n3=1000)
mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
myccov <- coCovar(s50a[,1])
mydata <- sienaDataCreate(mynet1, myccov)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, balance)
myeff <- includeTimeDummy(myeff, RateX, type="rate",
            interaction1="myccov")
myeff
(ans <- siena07(myalgorithm, data=mydata, effects=myeff))

## End(Not run)
```

`installGui` *Obsolete function to start up the installer for the standalone Gui.*

Description

Once started the installer for the standalone version of RSiena. Only for Windows. Not possible since R version 2.12.0. Use `siena01Gui` within R instead.

Usage

```
installGui()
```

Value

None.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

Examples

```
## Not run: installGui()
```

---

iwlsm *Function to fit an iterated weighted least squares model.*

Description

Fits an iterated weighted least squares model.

Usage

```
iwlsm(x, ...)

## S3 method for class 'formula'
iwlsm(formula, data, weights, ses, ..., subset, na.action,
    method = c("M", "MM", "model.frame"),
    wt.method = c("inv.var", "case"),
    model = TRUE, x.ret = TRUE, y.ret = FALSE, contrasts = NULL)

## Default S3 method:
iwlsm(x, y, weights, ses, ..., w = rep(1/nrow(x), nrow(x)),
    init = "ls", psi = psi.iwlsm,
    scale.est = c("MAD", "Huber", "proposal 2"), k2 = 1.345,
    method = c("M", "MM"), wt.method = c("inv.var", "case"),
    maxit = 20, acc = 1e-4, test.vec = "resid", lqs.control = NULL)

psi.iwlsm(u, k, deriv = 0, w, sj2, hh)
```

Arguments

formula	a formula of the form `y ~ x1 + x2 + ...`.
data	data frame from which variables specified in `formula` are preferentially to be taken.
weights	a vector of prior weights for each case.
subset	An index vector specifying the cases to be used in fitting.
ses	Estimated variance of the responses. Will be paseed to `psi` as sj2
na.action	A function to specify the action to be taken if NAs are found. The 'factory-fresh' default action in R is `na.omit`, and can be changed by `options(na.action=)`.
x	a matrix or data frame containing the explanatory variables.
y	the response: a vector of length the number of rows of `x`.

method	Must be "M". (argument not used here).
wt.method	are the weights case weights (giving the relative importance of case, so a weight of 2 means there are two of these) or the inverse of the variances, so a weight of two means this error is half as variable? This will not work at present.
model	should the model frame be returned in the object?
x.ret	should the model matrix be returned in the object?
y.ret	should the response be returned in the object?
contrasts	optional contrast specifications: se lm.
w	(optional) initial down-weighting for each case. Will not work at present.
init	(optional) initial values for the coefficients OR a method to find initial values OR the result of a fit with a coef component. Known methods are "ls" (the default) for an initial least-squares fit using weights w*weights, and "lts" for an unweighted least-trimmed squares fit with 200 samples. Probably not functioning.
psi	the psi function is specified by this argument. It must give (possibly by name) a function g(x, ..., deriv, w) that for deriv=0 returns psi(x)/x and for deriv=1 returns some value. Extra arguments may be passed in via ....
scale.est	method of scale estimation: re-scaled MAD of the residuals (default) or Huber's proposal 2 (which can be selected by either "Huber" or "proposal 2").
k2	tuning constant used for Huber proposal 2 scale estimation.
maxit	the limit on the number of IWLS iterations.
acc	the accuracy for the stopping criterion.
test.vec	the stopping criterion is based on changes in this vector.
...	additional arguments to be passed to iwlsm.default or to the psi function.
lqs.control	An optional list of control values for lqs.
u	numeric vector of evaluation points.
k	tuning constant. Not used.
deriv	0 or 1: compute values of the psi function or of its first derivative. (Latter not used).
sj2	Estimated variance of the responses
hh	Diagonal values of the hat matrix

Details

This function is very slightly adapted from rlm in packages MASS. It alternates between weighted least squares and estimation of variance on the basis of a common variance. The function psi.iwlsm calculates the weights for the next iteration. Used by siena08 to combine estimates from different sienaFits.

Value

An object of class "iwlsm" inheriting from "lm". Note that the df.residual component is deliberately set to NA to avoid inappropriate estimation of the residual scale from the residual mean square by "lm" methods.

The additional components not in an lm object are

s	the robust scale estimate used
w	the weights used in the IWLS process
psi	the psi function with parameters substituted
conv	the convergence criteria at each iteration
converged	did the IWLS converge?
wresid	a working residual, weighted for "inv.var" weights only.

Note

The function has been changed as little as possible, but has only been used with default arguments. The other options have been retained just in case they may prove useful.

Author(s)

Ruth Ripley

References

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.
See also http://www.stats.ox.ac.uk/~snijders/siena/

See Also

siena08, sienaMeta, sienaFit

Examples

```
## Not run:
##not enough data here for a sensible example, but shows the idea.
myalgorithm <- sienaAlgorithmCreate(nsub=2, n3=100)
mynet1 <- sienaDependent(array(c(s501, s502), dim=c(50, 50, 2)))
mynet2 <- sienaDependent(array(c(s502, s503), dim=c(50, 50, 2)))
mydata1 <- sienaDataCreate(mynet1)
mydata2 <- sienaDataCreate(mynet2)
myeff1 <- getEffects(mydata1)
myeff2 <- getEffects(mydata2)
myeff1 <- setEffect(myeff1, transTrip, fix=TRUE, test=TRUE)
myeff2 <- setEffect(myeff2, transTrip, fix=TRUE, test=TRUE)
myeff1 <- setEffect(myeff1, cycle3, fix=TRUE, test=TRUE)
myeff2 <- setEffect(myeff2, cycle3, fix=TRUE, test=TRUE)
ans1 <- siena07(myalgorithm, data=mydata1, effects=myeff1, batch=TRUE)
ans2 <- siena07(myalgorithm, data=mydata2, effects=myeff2, batch=TRUE)
meta <- siena08(ans1, ans2)
```

```
metadf <- split(meta$thetadf, meta$thetadf$effects)[[1]]
metalm <- iwlsm(theta ~ tconv, metadf, ses=se^2)

## End(Not run)
```

---

maxlikefn *A ML version of FRAN*

Description

A function to be called as 'FRAN'. All in R. very slow. work in progress.

Usage

```
maxlikefn(z, x, INIT = FALSE, TERM = FALSE, data,
effects = NULL, nstart = 1000, pinsdel = 0.6,
pperm = 0.3, prelins = 0.1, multfactor=2, promul = 0.1,
promul0 = 0.5, pdiaginsdel = 0.1, fromFiniteDiff = FALSE,
noSamples = 1, sampInterval = 50, int = 1)
```

Arguments

z	control object, passed in automatically in Siena07
x	model object, passed in automatically in Siena07
INIT	if TRUE, do initial processing. May be required to set up z
TERM	if TRUE, do end processing.
data	A siena object
effects	list of data frames as returned by getEffects
nstart	Number of MH steps at the start, after making the chain
pinsdel	Probability of insert/delete step
pperm	Probability of permutation step. (set to zero in startup phase.)
prelins	Insertion probability in InsDelPermute
multfactor	Factor controlling number of MH steps. Will be read from the model in preference, and that is easier to alter! But I don't want to alter that program yet..
promul	Probability of choosing a random single multiple in InsDelPermute in start up phase.
promul0	Probability of choosing a random single multiple in InsDelPermute not in startup phase
pdiaginsdel	Probability of insertion or deletion of a diagonal link.
fromFiniteDiff	Should always be FALSE
noSamples	Number of chains to be returned
sampInterval	If multiple chains are returned, the number of steps between each
int	Number of parallel MCMC chains to pursue.

Details

This can be used for the element FRAN of the model object. The arguments with no defaults must be passed in on the call to siena07. Also you must set the option `maxlike=TRUE` in the call to `sienaAlgorithmCreate()`

Value

Depends on the call. If `INIT` or `initC` or `TERM` are true, returns `z`, the control object. Otherwise, returns a list containing:

`fra`	Simulated scores
`dff`	2nd deriv, not phase 2
`OK`	could be set to FALSE if serious error has occurred

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

siena07

Examples

```
## Not run:
mynet1 <- sienaDependent(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(mynet1)
myeff<- getEffects(mydata)
myalgor<- sienaAlgorithmCreate(nsub=2, n3=100, maxlike=TRUE)
ans<- siena07(myalgor, data=mydata, effects=myeff,  batch=TRUE)

## End(Not run)
```

---

n3401 — *Network data: excerpt from 'Dutch Social Behavior Data Set' of Chris Baerveldt.*

Description

Matrices N3401, N3403, N3404, N3406, and HN3401, HN3403, HN3404, HN3406 are two waves of networks for four schools (numbered 1, 3, 4, 6): there is a tie from pupil i to pupil j if i says that he/she receives and/or gives emotional support from/to pupil j. The data are part of a larger data set (see source below) and were collected under the direction of Chris Baerveldt.

Format

Adjacency matrices for the network at two time points. The matrices with name N... are the first wave, those with name HN... are the second wave.

Source

http://www.stats.ox.ac.uk/~snijders/siena/CB_data.zip

References

Houtzager, B. & Baerveldt, C. (1999). Just like Normal. A Social Network Study of the Relation between Petty Crime and the Intimacy of Adolescent Friendships. *Social Behavior and Personality* 27(2), 177-192.

Snijders, Tom A.B, and Baerveldt, Chris (2003). A Multilevel Network Study of the Effects of Delinquent Behavior on Friendship Evolution. *Journal of Mathematical Sociology* 27, 123-151.

See http://www.stats.ox.ac.uk/~snijders/siena/BaerveldtData.html

Examples

```
mynet <- sienaDependent(array(c(N3401, HN3401), dim=c(45, 45, 2)))
mydata <- sienaDataCreate(mynet)
```

---

plot.sienaTimeTest *Functions to plot assessment of time heterogeneity of parameters*

Description

Plot method for sienaTimeTest objects.

Usage

```
## S3 method for class 'sienaTimeTest'
plot(x, pairwise=FALSE, effects,
 scale=.2, plevels=c(.1, .05, .025), ...)
```

Arguments

x	A sienaTimeTest object returned by sienaTimeTest.
pairwise	A Boolean value corresponding to whether the user would like a pairwise plot of the simulated statistics to assess correlation among the effects (pairwise=TRUE), or a plot of the estimates across waves in order to assess graphically the results of the score type test.
effects	A vector of integers corresponding to the indices given in the sienaTimeTest output for effects which are to be plotted.

scale	A positive number corresponding to the number of standard deviations on one step estimates to use for computing the maximum and minimum of the plotting range. We recommend experimenting with this number when the y-axes of the plots are not satisfactory. Smaller numbers shrink the axes.
plevels	A list of three decimals indicating the gradients at which to draw the confidence interval bars.
...	For extra arguments. The Lattice parameter layout can be used to control the layout of the graphs.

Details

The pairwise=TRUE plot may be used to assess whether effects are highly correlated. This information may be important when considering forward-model selection, since highly correlated effects may have highly correlated one-step estimates, particularly since the individual score type tests are not orthogonalized against the scores and deviations of yet-unestimated dummies. For example, reciprocity and outdegree may have highly correlated statistics as indicated by a strong, positive correlation coefficient. When considering whether to include dummy terms, it may be a good idea to include, e.g., outdegree, estimate the parameter, and see whether reciprocity dummies remain significant after method of moments estimation of the updated model–as opposed to including both outdegree and reciprocity.

The pairwise=FALSE plot displays the most of the information garnered from sienaTimeTest in a graphical fashion. For each effect, the method of moments parameter estimate for the base period (i.e. wave 1) is given as a blue, horizontal reference line. One step estimates are given for all of the parameters by dots at each wave. The dots are colored black if the parameter has been included in the model already (i.e. has been estimated via method of moments), or red if they have not been included. Confidence intervals are given based on pivots given at pvalues. Evidence of time heterogeneity is suggested by points with confidence intervals not overlapping with the base period.

Value

None

Author(s)

Josh Lospinoso

References

See http://www.stats.ox.ac.uk/~snijders/siena/ for general information on RSiena.

See Also

siena07, sienaTimeTest, xyplot

Examples

```
## Not run:
myalgorithm <- sienaAlgorithmCreate(nsub=4, n3=500)
mynet1 <- sienaDependent(array(c(s501, s502, s503, s501, s503, s502), dim=c(50, 50, 6)))
```

```
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, balance)
myeff <- includeTimeDummy(myeff, recip, timeDummy="2,3,5")
myeff <- includeTimeDummy(myeff, balance, timeDummy="4")
myeff <- includeTimeDummy(myeff, density, timeDummy="all")
ansp <- siena07(myalgorithm, data=mydata, effects=myeff, batch=TRUE)
ttp <- sienaTimeTest(ansp)

## Pairwise plots show
plot(ttp, pairwise=TRUE)

## Time test plots show
plot(ttp, effects=1:3) ## default layout
plot(ttp, effects=1:3, layout=c(3,1))

## End(Not run)
```

`print.sienaEffects` *Print methods for Siena effects objects*

Description

Print the major columns of the effects object. Or all, with any non atomic columns listed separately.

Usage

```
## S3 method for class 'sienaEffects'
print(x, fileName = NULL, includeOnly=TRUE,
expandDummies = FALSE, ...)
## S3 method for class 'sienaEffects'
summary(object, fileName = NULL,
includeOnly=TRUE, expandDummies = FALSE, ...)
## S3 method for class 'summary.sienaEffects'
print(x, fileName = NULL, ...)
```

Arguments

`object`	An object of class `sienaEffects`
`x`	An object of class `sienaEffects` or `summary.sienaEffects` as appropriate.
`fileName`	Character string denoting file name if file output desired.
`includeOnly`	Boolean. If TRUE, only effects with the include flag TRUE will be printed.
`expandDummies`	Interpret the `timeDummy` column and show any effects which would be added by `sienaTimeFix`
`...`	For extra arguments (none used at present)

Value

The function `print.sienaEffects` prints details of the main columns of the selected rows of the effects object.

The function `summary.sienaEffects` checks the rows for valid printing via `print.data.frame` and excludes any that will fail. The OK columns are printed first, followed by any others.

Output from either can be directed to a file by using the argument `filename`.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

`sienaTimeTest`, `effectsDocumentation`

Examples

```
mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mybeh <- sienaDependent(s50a, type='behavior')
mycovar <- coCovar(rnorm(50))
mydyadcovar <- coDyadCovar(matrix(as.numeric(rnorm(2500) > 2), nrow=50))
mydata <- sienaDataCreate(mynet1, mybeh, mycovar, mydyadcovar)
myeff <- getEffects(mydata)
myeff
summary(myeff)
```

---

print.sienaMeta *Methods for processing sienaMeta objects*

Description

`print`, `summary`, and `plot` methods for `sienaMeta` objects.

Usage

```
## S3 method for class 'sienaMeta'
print(x, file=FALSE, ...)

## S3 method for class 'sienaMeta'
summary(object, file=FALSE, extra=TRUE, ...)

## S3 method for class 'summary.sienaMeta'
print(x, file=FALSE, extra=TRUE, ...)
```

```
## S3 method for class 'sienaMeta'
plot(x,  ..., layout=c(2,2))
```

Arguments

object	An object of class sienaMeta
x	An object of class sienaMeta, or summary.sienaMeta as appropriate
file	Boolean: if TRUE, sends output to file named x$projname.out. If FALSE, output is to the terminal.
extra	Boolean: if TRUE, prints more information
layout	the vector giving number of rows and columns in the arrangement of the several panels in a rectangular array, possibly spanning multiple pages
...	For extra arguments (none used at present)

Value

The function print.sienaMeta prints details of the merged estimates of the meta-analysis, with test statistics.

The function summary.sienaMeta prints details as for the print method, but also details of the sienaFit objects included.

Output from either can be directed to a file by using the argument file. It will be appended to any existing file of the same name: projname.out where projname is the value of the argument to siena08.

The function plot.sienaMeta plots estimates against standard errors for each effect, with reference lines added at the two-sided significance threshold 0.05. It returns an object of class trellis, of the lattice.package. Effects for which a score test was requested are not plotted.

Author(s)

Ruth Ripley, Tom Snijders

References

T. A. B. Snijders and Chris Baerveldt. "Multilevel network study of the effects of delinquent behavior on friendship evolution". *Journal of Mathematical Sociology*, 27: 123–151, 2003.

See also the Siena manual and http://www.stats.ox.ac.uk/~snijders/siena/

Examples

```
## Not run:
# A meta-analysis for three groups does not make much sense
# for generalizing to a population of networks,
# but it the Fisher combinations of p-values are meaningful.
# But using three groups shows the idea.

Group1 <- sienaDependent(array(c(N3401, HN3401), dim=c(45, 45, 2)))
```

```
Group3 <- sienaDependent(array(c(N3403, HN3403), dim=c(37, 37, 2)))
Group4 <- sienaDependent(array(c(N3404, HN3404), dim=c(33, 33, 2)))
dataset.1 <- sienaDataCreate(Friends = Group1)
dataset.3 <- sienaDataCreate(Friends = Group3)
dataset.4 <- sienaDataCreate(Friends = Group4)
OneAlgorithm <- sienaAlgorithmCreate(projname = 'SingleGroups')
effects.1 <- getEffects(dataset.1)
effects.3 <- getEffects(dataset.3)
effects.4 <- getEffects(dataset.4)
effects.1 <- includeEffects(effects.1, transTrip)
effects.1 <- setEffect(effects.1, cycle3, fix=TRUE, test=TRUE)
effects.3 <- includeEffects(effects.3, transTrip)
effects.3 <- setEffect(effects.3, cycle3, fix=TRUE, test=TRUE)
effects.4 <- includeEffects(effects.4, transTrip)
effects.4 <- setEffect(effects.4, cycle3, fix=TRUE, test=TRUE)
ans.1 <- siena07(OneAlgorithm, data=dataset.1, effects=effects.1, batch=TRUE)
ans.3 <- siena07(OneAlgorithm, data=dataset.3, effects=effects.3, batch=TRUE)
ans.4 <- siena07(OneAlgorithm, data=dataset.4, effects=effects.4, batch=TRUE)
ans.1
ans.3
ans.4
(meta <- siena08(ans.1, ans.3, ans.4))
summary(meta)
plo <- plot(meta, layout = c(3,1))
plo
plo[3]

## End(Not run)
```

`print01Report` *Function to produce the Siena01 report from R objects*

Description

Prints a report of a Siena data object and its default effects.

Usage

```
print01Report(data, myeff, modelname = "Siena", session = NULL,
getDocumentation=FALSE)
```

Arguments

`data`	a Siena data object
`myeff`	a Siena Effects object
`modelname`	Character string used to name the output file "modelname.out"
`session`	Used to pass in a Siena01Gui() style session object so that data file names can be printed.
`getDocumentation`	Flag to allow documentation of internal functions, not for use by users.

Details

First deletes any file of the name "modelname.out", then prints a new one.

Value

No value returned.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

siena01Gui

Examples

```
mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
## Not run: print01Report(mydata, myeff)
```

---

s50 *Network data: excerpt from 'Teenage Friends and Lifestyle Study' data.*

Description

An excerpt of the network and alcohol consumption data for 50 randomly chosen girls from the Teenage Friends and Lifestyle Study data set. Useful as a small example of network and behaviour, for which models can be fitted quickly, and for which there are no missing values.

Format

Adjacency matrix for the network at time points 1, 2, 3; 50 by 3 matrix of alcohol consumption data for the three time points.

Source

http://www.stats.ox.ac.uk/~snijders/siena/s50_data.zip

References

West, P. and Sweeting, H. (1995) Background Rationale and Design of the West of Scotland 11-16 Study. Working Paper No. 52. MRC Medical Sociology Unit Glasgow.

See http://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm

See Also

s501, s502, s503, s50a

Examples

```
mynet <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mybeh <- sienaDependent(s50a, type='behavior')
mydata <- sienaDataCreate(mynet, mybeh)
```

---

s501 *Network 1 data: excerpt from 'Teenage Friends and Lifestyle Study' data.*

Description

First timepoint network data from an excerpt of 50 girls from the Teenage Friends and Lifestyle Study data set. Useful as a small example of network and behaviour, for which models can be fitted quickly.

Format

The adjacency matrix for the network at time point 1.

Source

http://www.stats.ox.ac.uk/~snijders/siena/s50_data.zip

References

West, P. and Sweeting, H. (1995) Background Rationale and Design of the West of Scotland 11-16 Study. Working Paper No. 52. MRC Medical Sociology Unit Glasgow.

See http://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm

See Also

s502, s503, s50a, s502

s502 *Network 2 data: excerpt from 'Teenage Friends and Lifestyle Study' data.*

Description

Second timepoint network data from an excerpt of 50 girls from the Teenage Friends and Lifestyle Study data set. Useful as a small example of network and behaviour, for which models can be fitted quickly.

Format

The adjacency matrix for the network at time point 2.

Source

http://www.stats.ox.ac.uk/~snijders/siena/s50_data.zip

References

West, P. and Sweeting, H. (1995) Background Rationale and Design of the West of Scotland 11-16 Study. Working Paper No. 52. MRC Medical Sociology Unit Glasgow.

See http://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm

See Also

s501, s503, s50a, s50

s503 *Network 3 data: excerpt from 'Teenage Friends and Lifestyle Study' data.*

Description

Second timepoint network data from an excerpt of 50 girls from the Teenage Friends and Lifestyle Study data set. Useful as a small example of network and behaviour, for which models can be fitted quickly.

Format

Adjacency matrix for the network at time point 3.

Source

http://www.stats.ox.ac.uk/~snijders/siena/s50_data.zip

References

West, P. and Sweeting, H. (1995) Background Rationale and Design of the West of Scotland 11-16 Study. Working Paper No. 52. MRC Medical Sociology Unit Glasgow.

See http://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm

See Also

s501, s502, s50a

Examples

```
mynet <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mybeh <- sienaDependent(s50a, type='behavior')
mydata <- sienaDataCreate(mynet, mybeh)
```

---

s50a — *Alcohol use data: excerpt from 'Teenage Friends and Lifestyle Study' data*

Description

Data from an excerpt of 50 girls from the Teenage Friends and Lifestyle Study data set. Useful as a small example of network and behaviour, for which models can be fitted quickly.

Format

A matrix of variables relating to the use of alcohol for the actors in the network. Three columns, one for each time point. Coding is 1–5, high values indicating higher consumption.

Source

http://www.stats.ox.ac.uk/~snijders/siena/s50_data.zip

References

West, P. and Sweeting, H. (1995) Background Rationale and Design of the West of Scotland 11-16 Study. Working Paper No. 52. MRC Medical Sociology Unit Glasgow.

See http://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm

See Also

s501, s502, s503

Examples

```
mynet <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mybeh <- sienaDependent(s50a, type='behavior')
mydata <- sienaDataCreate(mynet, mybeh)
```

setEffect *Function to set various columns in an effects object in a Siena model*

Description

This function provides an interface to change various columns of a selected row of a Siena effects object

Usage

```
setEffect(myeff, shortName, parameter = 0, fix = FALSE,
test = FALSE, initialValue = 0, timeDummy = ",", include = TRUE,
name = myeff$name[1], type = "eval", interaction1 = "",
interaction2 = "", period=1, group=1, character=FALSE)
```

Arguments

myeff	a Siena effects object as created by getEffects
shortName	A short name (all with or all without quotes) to identify the effect which should be changed.
parameter	Integer value required. Default 0.
fix	Boolean required. Default FALSE.
test	Boolean required. Default FALSE.
initialValue	Initial value required. Default 0.
timeDummy	string: Comma delimited string of which periods to dummy. Alternatively, use includeTimeDummy.
include	Boolean. default TRUE, but can be switched to FALSE to turn off an effect.
name	Name of network for which effects are being included. Defaults to the first in the effects object.
type	Character string indicating the type of the effect to be changed : currently "rate", "eval", "endow", or "creation". Default "eval".
interaction1	Name of siena object where needed to completely identify the effect e.g. covariate name or behavior variable name.
interaction2	Name of siena object where needed to completely identify the effect e.g. covariate name or behavior variable name.
period	Number of period if basic rate. Use numbering within groups.
group	Number of group if basic rate.
character	Boolean: is the short name a character string or not

Details

The arguments should identify the effects completely. The parm column will be set to the parameter value requested. The include column will be set to the include value requested (TRUE or FALSE)

Value

An updated version of the input effects object, with one row updated. Details of the row altered will be printed.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

getEffects

Examples

```
mynet <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mybeh <- sienaDependent(s50a, type="behavior")
mydata <- sienaDataCreate(mynet, mybeh)
myeff <- getEffects(mydata)
myeff <- setEffect(myeff, Rate, initialValue=1.5, name="mybeh",
                   type="rate", period=2)
myeff <- setEffect(myeff, outInv, 3) ## parameter
myeff
```

---

siena01Gui *User interface*

Description

Gui to allow entry of the data for a siena model fit.

Usage

```
siena01Gui(getDocumentation=FALSE)
```

Arguments

getDocumentation	Flag to allow documentation of internal functions, not for use by users.

Details

This function provides a graphical user interface for fitting Siena models. Note that this is less flexible and has fewer possibilities than creating data objects from matrices and vectors using `sienaDependent`, `coCovar` etc., and finally `sienaDataCreate`.

It can be run from within an R session, but is also called directly by `sienascript` (Linux or Mac) (the Windows interface via `siena.exe` has now been removed). It allows entry of details of the data files required, either using the gui or by loading a session file. The `Apply` button causes a call to `sienaDataCreateFromSession` followed by a display of the `sienaModelOptions` screen.

The required format for the column entries is described on the help page for `sienaDataCreateFromSession`, as this function can also be called directly.

The entries for the table can be loaded from a file by using the buttons `Load new session from file` or `Continue session from file`. The former will create a new report file and produce a descriptive report. The latter will use an existing report file and omit the descriptive report.

Alternatively, use `Add` and `Remove` buttons to enter the file names, and adjust the other columns to describe your data (see help page for `sienaDataCreateFromSession`).

The `Save to file` button will save the entries in the table to a session file.

The `Clear` button will empty the table.

The `Apply` button will prompt to save the session, then create the data objects and display the `sienaModelOptions` screen.

Exit by using the menu `File/Quit` or by closing the Window.

Value

None, although various objects made will still be in the directory if you are using this within an R session.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

`sienaDataCreateFromSession`, `sienaDataCreate`.

Examples

```
## Not run: siena01Gui()
```

siena07 *Function to estimate parameters in a Siena model*

Description

Estimates parameters in a Siena model using method of moments, based on direct simulation, conditional or otherwise; or using Maximum Likelihood by MCMC simulation. Estimation is done using a Robbins-Monro algorithm. Note that the data and particular model to be used must be passed in using named arguments as the ..., and the specification for the algorithm must be passed on as x, which is a sienaAlgorithm object as produced by sienaAlgorithmCreate (see examples).

Usage

```
siena07(x, batch=FALSE, verbose=FALSE, silent=FALSE,
        useCluster=FALSE, nbrNodes=2, initC=TRUE,
        clusterString=rep("localhost", nbrNodes), tt=NULL,
        parallelTesting=FALSE, clusterIter=!x$maxlike,
        clusterType=c("PSOCK", "FORK"), ...)
```

Arguments

x	A control object, of class sienaAlgorithm
batch	Desired interface: FALSE gives a gui (graphical user interface implemented as a tcl/tk screen), FALSE gives a small amount of printout to the console.
verbose	Produces various output to the console if TRUE.
silent	Produces no output to the console if TRUE, even if batch mode.
useCluster	Boolean: whether to use a cluster of processes (useful if multiple processors are available).
nbrNodes	Number of processes to use if useCluster is TRUE.
initC	Boolean: set to TRUE if the simulation will use C routines (currently always needed). Only for use if using multiple processors, to ensure all copies are initialised correctly. Ignored otherwise, so is set to TRUE by default.
clusterString	Definitions of clusters. Default set up to use the local machine only.
tt	A tcltk toplevel window. Used if called from the model options screen.
parallelTesting	Boolean. If TRUE, sets up random numbers to parallel those in Siena 3.
clusterIter	Boolean. If TRUE, multiple processes execute complete iterations at each call. If FALSE, multiple processes execute a single wave at each call.
clusterType	Either "PSOCK" or "FORK". On Windows, must be "PSOCK". On a single non-Windows machine may be "FORK", and subprocesses will be formed by forking. If "PSOCK", subprocesses are formed using R scripts.

`...` Arguments for the simulation function, see `simstats0c`: usually, `data` and `effects`; possibly also `prevAns` if a previous reasonable provisional estimate was obtained for a similar model; possibly also `returnDeps` if the simulated dependent variables (networks, behaviour) should be returned.

Details

Runs a Robbins-Monro algorithm for parameter estimation using the three-phase implementation in Snijders (2001) and Snijders, Steglich and Schweinberger (2007), with (if `findiff=TRUE` in the `sienaAlgorithm` object) derivative estimation as in Schweinberger and Snijders (2007). Phase 1 does a few iterations to estimate the derivative matrix of the targets with respect to the parameter vector. Phase 2 does the estimation. Phase 3 runs a simulation to estimate standard errors and check convergence of the model. The simulation function is called once for each iteration in these phases and also once to initialise the model fitting and once to complete it. Unless in batch mode, displays a tcl/tk screen to allow interruption and to show progress.

Value

Returns an object of class `sienaFit`, some parts of which are:

`OK`	Boolean indicating successful termination
`termination`	Character string, values: "OK", "Error", or "UserInterrupt". "UserInterrupt" indicates that the user asked for early termination before phase 3.
`f`	Various characteristics of the data and model definition.
`theta`	Fitted value of theta.
`covtheta`	Estimated covariance matrix of theta; this is not available if the sienaAlgorithm object x was produced with simOnly=TRUE.
`dfra`	Matrix of estimated derivatives.
`sf`	Matrix of deviations from target in phase 3.
`sf2`	Array of statistics from simulations in phase 3.
`targets`	Observed statistics.
`targets2`	Observed statistics by wave, starting with the second wave .
`ssc`	Score function contributions for each wave for each simulation in phase 3. Zero if finite difference method is used
`sims`	If `returnDeps` is `TRUE`: list of simulated dependent variables (networks, behaviour). Networks are given as a list of edgelists, one for each period.
`Phase3nits`	Number of iterations actually performed in phase 3.

Writes text output to the file named "projname.out", where projname is defined in the sienaAlgorithm object `x`.

Author(s)

Ruth Ripley

References

- Schweinberger, Michael, and Snijders, Tom A.B. (2007). Markov models for digraph panel data: Monte Carlo-based derivative estimation. *Computational Statistics and Data Analysis* 51, 4465-4483.
- Snijders, Tom A.B. (2001). The statistical evaluation of social network dynamics. *Sociological Methodology*, 31, 361-395.
- Snijders, Tom A.B., Steglich, Christian E.G., and Schweinberger, Michael (2007). Modeling the co-evolution of networks and behavior. Pp. 41-71 in *Longitudinal models in the behavioral and related sciences*, edited by Kees van Montfort, Han Oud and Albert Satorra; Lawrence Erlbaum.
- Steglich, Christian E. G., Snijders, Tom A. B., and Pearson, Michael A. (2010). Dynamic networks and behavior: Separating selection from influence. Sociological Methodology, 40, 329-393.
- Further see http://www.stats.ox.ac.uk/~snijders/siena/ .

See Also

There are print, summary and xtable methods for `sienaFit` objects: `xtable`, `print.sienaFit`

Examples

```
myalgorithm <- sienaAlgorithmCreate(nsub=2, n3=100)
# nsub=2 and n3=100 is used here for having a brief computation, not for practice.
mynet1 <- sienaDependent(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
ans <- siena07(myalgorithm, data=mydata, effects=myeff, batch=TRUE)

# or for conditional estimation
## Not run:
myalgorithm$condname <- 'mynet1'
myalgorithm$cconditional <- TRUE
ans <- siena07(myalgorithm, data=mydata, effects=myeff, batch=TRUE)

## End(Not run)

# or if a previous 'on track' result ans was obtained
## Not run:
ans1 <- siena07(myalgorithm, data=mydata, effects=myeff, prevAns=ans)

## End(Not run)
```

`siena08` *Function to perform a meta analysis of a collection of Siena fits.*

Description

Estimates a meta analysis based on a collection of Siena fits.

Usage

```
siena08(..., projname = "sienaMeta", bound = 5, alpha = 0.05, maxit=20)
```

Arguments

...	names of Siena fit objects, returned from siena07. They will be renamed if entered in format newname=oldname.
projname	Base name of report file if required
bound	Upper limit of standard error for inclusion in the meta analysis.
alpha	1 minus confidence level of confidence intervals.
maxit	Number of iterations of iterated least squares procedure.

Details

A meta analysis is performed as described in the Siena manual, section 'Meta-analysis of Siena results'. This consists of three parts: an iterated weighted least squares modification of the method described in the reference below; maximum likelihood estimates and confidence intervals based on profile likelihoods under normality assumptions; and Fisher combinations of left-sided and right-sided p-values. These are produced for all effects separately.

Note that the corresponding effects must have the same effect name in each model fit. This implies that at least covariates and behavior variables must have the same name in each model fit.

Value

An object of class sienaMeta. There are print, summary and plot methods for this class,

An object of class sienaMeta is a list containing at least the following. (Items cor.est to ns appear once for each effect.)

cor.est	Spearman rank correlation coefficient between estimates and their standard errors.
cor.pval	p-value for above
regfit	Part of the result of the fit of iwlsm.
regsummary	The summary of the fit, which includes the coefficient table.
Tsq	test statistic for effect zero in every model
pTsq	p-value for above
tratio	test statistics that mean effect is 0
ptratio	p-value for above
Qstat	Test statistic for variance of effects is zero
pttilde	p-value for above
cjplus	Test statistic for at least one theta strictly greater than 0
cjminus	Test statistic for at least one theta strictly less than 0
cjplusp	p-value for cjplus
cjminusp	p-value for cjminus

mu.ml	ML estimate of population mean
mu.ml.se	standard error of ML estimate of population mean
sigma.ml	ML estimate of population standard deviation
mu.confint	confidence interval for population mean based on profile likelihood
sigma.confint	confidence interval for population standard deviation based on profile likelihood
n1	Number of fits on which the meta analysis is based
scoreplus	Test statistic for combination of right one-sided p-values from score tests
scoreminus	Test statistic for combination of left one-sided p-values from score tests
scoreplusp	p-value for scoreplus
scoreminusp	p-value for scoreminus
ns	Number of fits on which the score test analysis is based
thetadf	Data frame containing the coefficients, standard errors and score test results
projname	Name for any output file to be produced by the print method
bound	Estimates with standard error above this value were excluded from the calculations
scores	Object of class by indicating, for each effect in the models, whether score test information was present.

Author(s)

Ruth Ripley, Tom Snijders

References

T. A. B. Snijders and Chris Baerveldt. Multilevel network study of the effects of delinquent behavior on friendship evolution. *Journal of Mathematical Sociology*, 27: 123–151, 2003.

See also the manual and http://www.stats.ox.ac.uk/~snijders/siena/

See Also

sienaMeta, iwlsm, siena07

Examples

```
## Not run:
# A meta-analysis for three groups does not make much sense
# for generalizing to a population of networks,
# but it the Fisher combinations of p-values are meaningful.
# But using three groups shows the idea.

Group1 <- sienaDependent(array(c(N3401, HN3401), dim=c(45, 45, 2)))
Group3 <- sienaDependent(array(c(N3403, HN3403), dim=c(37, 37, 2)))
Group4 <- sienaDependent(array(c(N3404, HN3404), dim=c(33, 33, 2)))
dataset.1 <- sienaDataCreate(Friends = Group1)
dataset.3 <- sienaDataCreate(Friends = Group3)
dataset.4 <- sienaDataCreate(Friends = Group4)
```

```
OneAlgorithm <- sienaAlgorithmCreate(projname = 'SingleGroups')
effects.1 <- getEffects(dataset.1)
effects.3 <- getEffects(dataset.3)
effects.4 <- getEffects(dataset.4)
effects.1 <- includeEffects(effects.1, transTrip)
effects.1 <- setEffect(effects.1, cycle3, fix=TRUE, test=TRUE)
effects.3 <- includeEffects(effects.3, transTrip)
effects.3 <- setEffect(effects.3, cycle3, fix=TRUE, test=TRUE)
effects.4 <- includeEffects(effects.4, transTrip)
effects.4 <- setEffect(effects.4, cycle3, fix=TRUE, test=TRUE)
ans.1 <- siena07(OneAlgorithm, data=dataset.1, effects=effects.1, batch=TRUE)
ans.3 <- siena07(OneAlgorithm, data=dataset.3, effects=effects.3, batch=TRUE)
ans.4 <- siena07(OneAlgorithm, data=dataset.4, effects=effects.4, batch=TRUE)
ans.1
ans.3
ans.4
(meta <- siena08(ans.1, ans.3, ans.4))

## End(Not run)
```

`sienaAlgorithmCreate` *Function to create an object containing the algorithm specifications for parameter estimation in RSiena*

Description

Creates an object with specifications for the algorithm for parameter estimation in RSiena. `sienaAlgorithmCreate()` and `sienaModelCreate()` are identical functions; the second name was used from the start of the `RSiena` package, but the first name indicates more precisely the purpose of this function.

Usage

```
sienaAlgorithmCreate(fn, projname = "Siena", MaxDegree = 0,
     useStdInits = FALSE, n3 = 1000, nsub = 4,
     dolby=TRUE, maxlike = FALSE, diagonalize=1.0*!maxlike,
     condvarno = 0, condname = "", firstg = 0.2,
     cond = NA, findiff = FALSE, seed = NULL, pridg=0.05,
     prcdg=0.05, prper=0.2, pripr=0.3, prdpr=0.3, prirms=0.05,
     prdrms=0.05, maximumPermutationLength=40,
     minimumPermutationLength=2, initialPermutationLength=20,
     modelType=1, mult=5, simOnly=FALSE)

sienaModelCreate(fn, projname = "Siena", MaxDegree = 0,
     useStdInits = FALSE, n3 = 1000, nsub = 4,
     dolby=TRUE, maxlike = FALSE, diagonalize=1.0*!maxlike,
     condvarno = 0, condname = "", firstg = 0.2,
     cond = NA, findiff = FALSE, seed = NULL, pridg=0.05,
```

```
prcdg=0.05, prper=0.2, pripr=0.3, prdpr=0.3, prirms=0.05,
prdrms=0.05, maximumPermutationLength=40,
minimumPermutationLength=2, initialPermutationLength=20,
modelType=1, mult=5, simOnly=FALSE)
```

Arguments

fn	Function to do one simulation in the Robbins-Monro algorithm. Not to be touched.
projname	Character string name of project; the output file will be called projname.out. No embedded spaces!!!
MaxDegree	Named vector of maximum degree values for corresponding networks. Allows to restrict the model to networks with degrees not higher than this maximum.
useStdInits	Boolean. If TRUE, the initial values in the effects object will be ignored and default values used instead. If FALSE, the initial values in the effects object will be used.
n3	Number of iterations in phase 3.
nsub	Number of subphases in phase 2.
dolby	Boolean. Should there be noise reduction by regression on augmented data score. In most cases dolby=TRUE yields better convergence; if convergence is problematic, however, dolby=FALSE may be tried. Just use whatever works best.
maxlike	Whether to use maximum likelihood method or Method of Moments estimation.
diagonalize	Number between 0 and 1 (bounds included), values outside this interval will be truncated; for diagonalize=0 the complete estimated derivative matrix will be used for updates in the Robbins-Monro procedure; for diagonalize=1 only the diagonal entries will be used; for values between 0 and 1, the weighted average will be used with weight diagonalize for the diagonalized matrix. Has no effect for ML estimation. Higher values are more stable, lower values potentially more efficient. Default: for ML estimation, diagonalize=0; for MoM estimation, diagonalize = 1.0.
condvarno	If cond (conditional simulation), the sequential number of the network or behavior variable on which to condition.
condname	If conditional, the name of the dependent variable on which to condition. Use one or other of condname or condvarno to specify the variable.
firstg	Initial value of scaling ('gain') parameter for updates in the Robbins-Monro procedure.
cond	Boolean. Only relevant for Method of Moments simulation/estimation. If TRUE, use conditional simulation; if FALSE, unconditional simulation. If missing, decision is deferred until siena07, when it is set to TRUE if there is only one dependent variable, FALSE otherwise.
findiff	Boolean: If TRUE, estimate derivatives using finite differences. If FALSE, use scores.
seed	Integer. Starting value of random seed. Not used if parallel testing.

pridg	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
prcdg	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
prper	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
pripr	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
prdpr	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
prirms	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
prdrms	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
maximumPermutationLength	Maximum length of permutation in steps in ML estimation
minimumPermutationLength	Minimum length of permutation in steps in ML estimation
initialPermutationLength	Initial length of permutation in steps in ML estimation
modelType	Type of model to be fitted: 1=directed, 2:6 for symmetric networks: 2=forcing, 3=Initiative model, 4=Pairwise forcing model, 5=Pairwise mutual model, 6=Pairwise joint model
mult	Multiplication factor for maximum likelihood. Number of steps per iteration is set to this multiple of the total distance between the observations at start and finish of the wave. Decreasing mult below a certain value has no further effect.
simOnly	Logical: If TRUE, then the calculation of the covariance matrix and standard errors of the estimates at the end of Phase 3 of the estimation algorithm in function siena07 is skipped. This is suitable if nsub=0 and siena07 is used only for the purpose of simulation.

Details

Model specification is done via this object for siena07. This function creates an object with the elements required to control the Robbins-Monro algorithm. Those not available as arguments can be changed manually where desired.

Value

Returns an object of class sienaAlgorithm containing:

projname	String value of name of project.
useStdInits	Boolean, see above.
checktime	Boolean, set to TRUE: report time in the phases or not.
n3	number of iterations in Phase 3
firstg	Initial value of the scaling ('gain') parameter in the Robbins-Monro algorithm.
maxrat	Value used to control the maximum size of the jumps.
maxmaxrat	Value used to control the maximum size of the jumps.
maxlike	Boolean: is FRAN using maximum likelihood?
FRANname	Name of simulation function FRAN. Is derived by sienaModelCreate from fn and maxlike.

`cconditional`	Boolean: is FRAN using conditional estimation?
`condvarno`	Number of dependent variable on which to condition.
`condname`	Name of dependent variable on which to condition.
`FinDiff.method`	Boolean: are derivatives calculated using finite differences?
`nsub`	Number of subphases in phase 2.
`diag`	Boolean: use only the diagonal of the derivative matrix?
`modelType`	Type of model to be fitted: 1=directed, 2:6 for symmetric networks: 2=forcing, 3=Initiative model, 4=Pairwise forcing model, 5=Pairwise mutual model, 6=Pairwise joint model
`MaxDegree`	Named vector of maximum degree values, or NULL.
`randomSeed`	Integer. Starting value of random seed. Not present unless given in call.
`pridg`	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
`prcdg`	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
`prper`	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
`pripr`	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
`prdpr`	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
`prirms`	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
`prdrms`	Real number. Probability used in Metropolis-Hastings routine in ML estimation.
`maximumPermutationLength`	Maximum length of permutation in steps in ML estimation
`minimumPermutationLength`	Minimum length of permutation in steps in ML estimation
`initialPermutationLength`	Initial length of permutation in steps in ML estimation
`mult`	Multiplication factor for maximum likelihood. Number of steps per iteration is set to this multiple of the total distance between the observations at start and finish of the wave.
`simOnly`	Logical, indicating whether output of covariance matrix by `siena07` is to be skipped.

Author(s)

Ruth Ripley and Tom A.B. Snijders

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

`siena07`, `simstats0c`.

Examples

```
myAlgorithm <- sienaAlgorithmCreate(projname="NetworkDyn")
StdAlgorithm <- sienaAlgorithmCreate(projname="NetworkDyn", useStdInits=TRUE)
CondAlgorithm <- sienaAlgorithmCreate(projname="NetworkDyn", condvarno=1, cond=TRUE)
Max10Algorithm <- sienaAlgorithmCreate(projname="NetworkDyn", MaxDegree=c(mynet=10))
# where mynet is the name of the network object created by sienaDependent().
```

---

sienaCompositionChange

Functions to create a Siena composition change object

Description

Used to create a list of events describing the changes over time of a Siena actor set

Usage

```
sienaCompositionChange(changelist, nodeSet = "Actors", option = 1)
sienaCompositionChangeFromFile(filename, nodeSet = "Actors",
    fileobj=NULL, option = 1)
```

Arguments

changelist	A list with an entry for each actor in the node set. Each entry a vector of numbers (may be as characters) indicating intervals during which the corresponding actor was present.
filename	Name of file containing change information. One line per actor, each line a series of space delimited numbers indicating intervals.
fileobj	The result of readLines on filename.
nodeSet	Character string containing the name of a Siena node set. If the entire data set contains more than one node set, then the node sets must be specified in all data objects.
option	Integer controlling the processing of the tie variables for the actors not currently present. Values (default is 1)

1. 0 before entry, final value carried forward after leaving
2. 0 before entry, missing after (final value carried forward, but treated as missing)
3. missing whenever not in the network. Previous values will be used where available, but always treated as missing values.
4. Convert to structural zeros (not available at present).

Details

Intervals are treated as closed at each end.
For data sets including a composition change object, estimation by Method of Moments is forced to be unconditional, overriding the specification in the sienaAlgorithm object.

Value

An object of class "compositionChange", a list of numeric vectors, with attributes:

NodeSet	Name of node set
Option	Option

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

sienaNodeSet

Examples

```
clist <- list(c(1, 3), c(1.4, 2.5))
#or
clist <- list(c('1', '3'), c('1.4', '2.5'))

compChange <- sienaCompositionChange(clist)

## Not run:
filedata <- c("1 3", "1.4 2.5")
write.table(filedata, "cc.dat",row.names=FALSE, col.names=FALSE,
        quote=FALSE)
## file will be
## 1 3
## 1.4 2.5
compChange <- sienaCompositionChangeFromFile("cc.dat")

## End(Not run)
```

`sienaDataConstraint` *Function to change the values of the constraints between networks.*

Description

This function allows the user to change the constraints of "higher", "disjoint" and "atLeastOne" for a specified pair of networks in a Siena data object.

Usage

```
sienaDataConstraint(x, net1, net2,
          type = c("higher", "disjoint", "atLeastOne"), value = FALSE)
```

Arguments

`x`	Siena Data Object; maybe a group object?
`net1`	name of first network
`net2`	name of second network
`type`	one of "higher", "disjoint", "atleastOne". Default is "higher".
`value`	Boolean giving the value.

Details

The value of the appropriate attribute is set to the value requested.

Value

Updated Siena data object.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

`sienaDataCreate`

Examples

```
nowFriends <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
ever <- array(c(s501, s502, s503), dim=c(50, 50, 3))
ever[,,2] <- pmax(ever[,,1], ever[,,2])
ever[,,3] <- pmax(ever[,,2], ever[,,3])
everFriends <- sienaDependent(ever)
# Note: this data set serves to illustrate this function,
# but it is not an appropriate data set for estimation by siena07,
# because everFriends (for the three waves together) depends deterministically
# on nowFriends (for the three waves together).
nowOrEver <- sienaDataCreate(nowFriends, everFriends)
attr(nowOrEver, "higher")
nowOrEver
nowOrEver.unconstrained <-
   sienaDataConstraint(nowOrEver, everFriends, nowFriends, "higher", FALSE)
nowOrEver.unconstrained
attr(nowOrEver.unconstrained, "higher")
```

---

sienaDataCreate *Function to create a Siena data object*

Description

Creates a Siena data object from input networks, covariates, and composition change objects.

Usage

```
sienaDataCreate(..., nodeSets=NULL, getDocumentation=FALSE)
```

Arguments

`...`	objects of class `sienaDependent`, `coCovar`,`varCovar`,`coDyadCovar`, `varDyadCovar`, `sienaCompositionChange`
`nodeSets`	list of Siena node sets. Default is the single node set named 'Actors', length equal to the number of rows in the first object of class "sienaDependent". If the entire data set contains more than one node set, then the node sets must have been specified in the creation of all data objects.
`getDocumentation`	Flag to allow documentation of internal functions, not for use by users.

Details

Checks that the objects fit, that there is at least one network, and adds various attributes to each dependent variable describing the data. If there is more than one nodeSet they must all be specified.

Value

An object of class "siena" which is designed to be used in a siena model fit. The components of the object are.

`nodeSets`	List of node sets involved
`observations`	Integer indicating number of waves of data
`depvars`	List of networks and behavior variables
`cCovars`	List of constant covariates
`vCovars`	List of changing covariates
`dycCovars`	List of constant dyadic covariates
`dyvCovars`	List of changing dyadic covariates
`compositionChange`	List of composition change objects corresponding to the node sets

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

`sienaDependent`, `coCovar`, `varCovar`, `coDyadCovar`, `varDyadCovar`, `sienaCompositionChange`, `sienaGroupCreate`

Examples

```
mynet <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mybeh <- sienaDependent(s50a, type='behavior')
mydata <- sienaDataCreate(mynet, mybeh)
```

---

sienaDataCreateFromSession

Creates a Siena data object from a Siena session file

Description

Reads in a Siena session from file or `siena01Gui()` and creates a Siena data or group object.

Usage

```
sienaDataCreateFromSession(filename = NULL, session = NULL,
    modelName = "Siena", edited = NULL, files = NULL,
    getDocumentation=FALSE)
```

Arguments

`filename`	Input session file
`session`	Input session (from `siena01Gui`)
`modelName`	Character string of project name
`edited`	Boolean, indicates whether a file has been edited and therefore should not be re-read. Used internally by `siena01Gui`.
`files`	List of data files, used internally by `siena01Gui`
`getDocumentation`	Flag to allow documentation of internal functions, not for use by users.

Details

Allows creation of data objects of class "Siena" direct from data files rather than from the various Siena network and covariate objects. Is always called by `siena01Gui` but can also be used directly.

The columns of the gui screen should have the format described below. If a session file is used for input, it should have columns with exactly the same names and in exactly the same order as those below with a row of column headings and no row numbers.

Group Used to identify the groups when using the multi-group option described in the Manual. Must not contain embedded blanks, and should be identical for all rows which relate to the same group.

Name Network files or dyadic covariates should use the same name for each file of the set. Other files should have unique names, a list of space separated ones for constant covariates.

File Name in `siena01Gui`, usually entered by using a file selection box, after clicking `Add`.

Format Only relevant for networks or dyadic covariates. Can be matrix, a single Pajek network (.net) or a Siena network file (and edgelist with three or four columns: from, to, value, wave (optional)). Not tested for dyadic covariates yet!

Period(s) Only relevant for networks and dyadic covariates. All other files cover all the relevant periods. Indicates the order of the network and dyadic covariate files. Should range from 1 to n within each group. Enter multiple integers with spaces between for Siena network multi-wave files. Use the value 1 or blank for other files which cover multiple periods.

ActorSet If you have more than one set of nodes, use this column to indicate which is relevant to each file. Should not contain embedded blanks.

Type Indicate here what type of data the file contains. Options are "network", "behavior", "constant covariate", "changing covariate", "constant dyadic covariate", "changing dyadic covariate", "exogenous event".

Selected Yes or No. Only files with Yes will be included in the model.

Missing Values Enter any values which indicate missingness, with spaces between different entries.

Nonzero Codes Enter any values which indicate ties, with spaces between different entries.

NbrOfActors For Siena network files, enter the number of actors here.

If using a file for input, it should be of one of the following types:

Extension	Type
`.csv`	Comma separated
`.dat` or `.prn`	Space delimited
`.txt`	Tab delimited

Network and covariate files should be text files with a row for each node. The numbers should be separated by spaces or tabs. Exogenous events should be specified by a file with a row for each node. Each row should be consist of a set of pairs of numbers which indicate the periods during which the corresponding actor was present. e.g.

```
1 3
1.5 3
1 1.4 2.3 3
2.4 3
```

would describe a network with 4 nodes, and 3 observations. Actor 1 is present all the time, actor 2 joins at time 1.5, actor 3 leaves and time 1.4 then rejoins at time 2.3, actor 4 joins at time 2.4. All intervals are treated as closed.

Value

A list with the following components:

`OK`	Boolean, TRUE indicating success
`mydata`	A Siena data or group object, of class `siena` or `sienaGroup`
`myeff`	Effects object associated with `mydata`

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

`sienaDataCreate`, `siena`

Examples

```
## Not run:
tmp <- sienaDataCreateFromSession("sienaFreshman.csv")
mydata <- tmp$mydata
myeff <- tmp$myeff

## End(Not run)
```

`sienaDependent` *Function to create a dependent variable for a Siena model*

Description

Creates a Siena dependent variable: either a network, created from a matrix or array or list of sparse matrix of triples; or a behavior variable, created from a matrix. `sienaDependent()` and `sienaNet()` are identical functions; the second name was used from the start of the `RSiena` package, but the first name indicates more precisely the purpose of this function.

Usage

```
sienaDependent(netarray, type=c("oneMode", "bipartite", "behavior"),
nodeSet="Actors", sparse=is.list(netarray), allowOnly=TRUE)

sienaNet(netarray, type=c("oneMode", "bipartite", "behavior"),
nodeSet="Actors", sparse=is.list(netarray), allowOnly=TRUE)
```

Arguments

`netarray`	matrix (type="behavior" only) or (for the other types) array of values or list of sparse matrices of type "dgTMatrix"
`type`	type of network, default "oneMode"
`nodeSet`	character string naming the appropriate node set. A vector containing 2 character strings for a bipartite network: "rows" first, then "columns".
`sparse`	logical: TRUE indicates the data is in sparse matrix format, FALSE otherwise
`allowOnly`	logical: If TRUE, it will be detected when between any two consecutive waves the changes are non-decreasing or non-increasing, and if this is the case, this will also be a constraint for the simulations between these two waves. This is done by means of the internal parameters `uponly` and `downonly`. If FALSE, the parameters `uponly` and `downonly` always are set to FALSE, and changes in dependent variables will not be constrained to be non-decreasing or non-increasing. For normal operation, TRUE is the appropriate option.

Details

Adds attributes so that the array or list of matrices can be used in a Siena model fit.

Value

An object of class "sienaDependent". An array or (networks only) a list of sparse matrices with attributes:

`netdims`	Dimensions of the network or behavior variable: senders, receivers (1 for behavior), periods
`type`	oneMode, bipartite or behavior

sparse	Boolean: whether the network is given as a list of sparse matrices or not
nodeSet	Character string with name(s) of node set(s)
allowOnly	The value of the allowOnly parameter

Author(s)

Ruth Ripley and Tom A.B. Snijders

References

See http://www.stats.ox.ac.uk/siena/

See Also

sienaDataCreate, sienaDataConstraint

Examples

```
mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mybeh <- sienaDependent(s50a, type="behavior")
## note that the following example works although the node sets do not yet exist!
mynet3 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)),
       type="bipartite", nodeSet=c("senders", "receivers"))
## sparse matrix input - create some RSiena edgelists first
library(Matrix)
tmps501 <- as(Matrix(s501), "dgTMatrix")
tmps502 <- as(Matrix(s502), "dgTMatrix")
tmps503 <- as(Matrix(s503), "dgTMatrix")
mymat1 <- cbind(tmps501@i + 1, tmps501@j + 1, 1, 1)
mymat2 <- cbind(tmps502@i + 1, tmps502@j + 1, 1, 2)
mymat3 <- cbind(tmps503@i + 1, tmps503@j + 1, 1, 3)
mymat <- rbind(mymat1, mymat2, mymat3)
library(Matrix)
## mymat includes all 3 waves
mymatlist <- by(mymat, mymat[, 4], function(x)
    spMatrix(50, 50, x[, 1], x[, 2], x[, 3]))
mynet4 <- sienaDependent(mymatlist)
## or alternatively
mymat1 <- mymat[mymat[, 4] == 1, ]
mymat2 <- mymat[mymat[, 4] == 2, ]
mymat3 <- mymat[mymat[, 4] == 3, ]
mymat1s <- spMatrix(50, 50, mymat1[, 1], mymat1[, 2], mymat1[, 3])
mymat2s <- spMatrix(50, 50, mymat2[, 1], mymat2[, 2], mymat2[, 3])
mymat3s <- spMatrix(50, 50, mymat3[, 1], mymat3[, 2], mymat3[, 3])
mynet4 <- sienaDependent(list(mymat1s, mymat2s, mymat3s))
```

sienaFit.methods *Methods for processing sienaFit objects, produced by* siena07.

Description

print, summary, and xtable methods for sienaFit objects.

Usage

```
## S3 method for class 'sienaFit'
print(x, tstat=TRUE, ...)

## S3 method for class 'sienaFit'
summary(object,  ...)

## S3 method for class 'summary.sienaFit'
print(x,  ...)

## S3 method for class 'sienaFit'
xtable(x, caption = NULL, label = NULL, align = NULL,
               digits = NULL, display = NULL, ...)

siena.table(x, type='tex', file=paste(deparse(substitute(x)), '.', type,sep=""),
            vertLine=TRUE, tstatPrint=FALSE, sig=FALSE, d=3)
```

Arguments

object	An object of class sienaFit, produced by siena07.
x	An object of class sienaFit, or summary.sienaFit as appropriate
tstat	Boolean: add the t-statistics for convergence to the report
type	Type of output to produce; must be either 'tex' or 'html'
file	Name of the file; defaults to the name of the sienaFit object
vertLine	Boolean: add vertical lines separating the columns in siena.table
tstatPrint	Boolean: add a column of significance t values (parameter estimate/standard error estimate) to siena.table
sig	Boolean: adds symbols (daggers and asterisks) indicating significance levels for the parameter estimates to siena.table
d	The number of decimals places used in siena.table
caption	See documentation for xtable
label	See documentation for xtable
align	See documentation for xtable
digits	See documentation for xtable
display	See documentation for xtable
...	Add extra parameters for print.xtable here. e.g. type, file

Value

The function `print.sienaFit` prints a table containing estimated parameter values, standard errors and (optionally) t-statistics for convergence.

The function `summary.sienaFit` prints a table containing estimated parameter values, standard errors and t-statistics for convergence together with the covariance matrix of the estimates, the derivative matrix of expected statistics `X` by parameters, and the covariance matrix of the expected statistics `X`.

The function `xtable.sienaFit` creates an object of class `xtable.sienaFit` which inherits from class `xtable` and passes an extra arguments to the `print.xtable`.

The function `siena.table` outputs a latex or html table of the estimates and standards errors of a `sienaFit` object. The table will be written to the current directory and has a footnote reporting the maximum of the convergence t ratios.

Author(s)

Ruth Ripley, Charlotte Greenan

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

`xtable`, `print.xtable`, `siena07`

Examples

```
myalgorithm <- sienaAlgorithmCreate(nsub=2, n3=100)
mynet1 <- sienaDependent(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
ans <- siena07(myalgorithm, data=mydata, effects=myeff, batch=TRUE)
ans
summary(ans)
xtable(ans, type='html', file='ans.html')
siena.table(ans, type='html', tstat=TRUE, d=2)
```

---

sienaGOF *Functions to assess goodness of fit for SAOMs*

Description

The function `sienaGOF` assesses goodness of fit for a model specification as represented by an estimated `sienaFit` object. This is done by simulations of auxiliary statistics, that differ from the statistics used for estimating the parameters. The auxiliary statistics must be given explicitly. The fit is good if the average values of the auxiliary statistics over many simulation runs are close to the values observed in the data. A Monte Carlo test based on the Mahalanobis distance is used to

calculate frequentist p-values. Plotting functions can be used to diagnose bad fit. There are basic functions for calculating auxiliary statistics available out of the box, and the user is also permitted to create custom functions.

Usage

```
sienaGOF(sienaFitObject, auxiliaryFunction,
         period=NULL, verbose=FALSE, join=TRUE, twoTailed=FALSE,
         cluster=NULL, robust=FALSE, groupName="Data1",
         varName, ...)
## S3 method for class 'sienaGOF'
plot(x, center=FALSE, scale=FALSE, violin=TRUE, key=NULL,
          perc=.05, period=1, main=main, ylab=ylab, ...)
```

Arguments

sienaFitObject	Results from a call to siena07 with returnDeps = TRUE.
auxiliaryFunction	Function to be used to calculate the auxiliary statistics; this can be a user-defined function, e.g. depending on the sna or igraph packages. See Examples and sienaGOF-auxiliary for more information on the signature of this function. The basic signature is function(index, data, sims, period, groupName, varName, ...), where index is the index of the simulated network, or NULL if the observed variable is needed; data is the observed data object from which the relevant variables are extracted; sims is the list of simulations returned from siena07; period is the index of the period; and ... are further arguments (like levls in the examples below and in sienaGOF-auxiliary).
period	Vector of period(s) to be used (may run from 1 to number of waves - 1). Has an effect only if join=FALSE.
verbose	Whether to print intermediate results. This may give some peace of mind to the user because calculations can take some time.
join	Boolean: should sienaGOF do tests on all of the periods individually (FALSE), or sum across periods (TRUE)?
twoTailed	Whether to use two tails for calculating p-values on the Monte Carlo test. Recommended for advanced users only, as it is probably only applicable in rare cases.
cluster	Optionally, a snow cluster to execute the auxiliary function calculations on.
robust	Whether to use robust estimation of the covariance matrix.
groupName	Name of group; relevant for multi-group data sets.
varName	Name of dependent variable.
x	Result from a call to sienaGOF.
center	Whether to center the statistics by median during plotting.
scale	Whether to scale the statistics by range during plotting.
violin	Use violin plots (vs. box plots only)?

key	Keys in the plot for the levels of the auxiliary statistic (as given by parameter levls in the examples).
perc	1 minus confidence level for the confidence bands (two sided).
main	Main title of the plot.
ylab	The y-axis label for the plot.
...	Other arguments.

Details

This function is used to assess the goodness of fit of a stochastic actor oriented model for an arbitrarily defined multidimensional auxiliary statistic. The auxiliary statistics are calculated for the simulated dependent variables in Phase 3 of the estimation algorithm, returned in sienaFitObject because of having used returnDeps = TRUE in the call to siena07. These statistics should be chosen to represent features of the network that are not explicitly fit by the estimation procedure but can be considered important properties that the model at hand should represent well. Some examples are:

- Outdegree distribution
- Indegree distribution
- Distribution of the dependent behavior variable (if any).
- Distribution of geodesic distances
- Triad census
- Edgewise homophily counts
- Edgewise shared partner counts
- Statistics depending on the combination of network and behavioral variables.

The function is written so that the user can easily define other functions to capture some other relevant aspects of the network, behaviors, etc. This is further illustrated in the help page sienaGOF-auxiliary.

We recommend the following heuristic approach to model checking:

1. Check convergence of the estimation.
2. Assess time heterogeneity by sienaTimeTest and if there is evidence for time heterogeneity either modify the base effects or include time dummy terms.
3. Assess goodness of fit (primarily using join=TRUE) on auxiliary statistics, and if necessary refine the model.

The print function will display some useful information to help with model selection if some effects are set to FIX and TEST on the effects object. A rough estimator for the Mahalanobis distance that would be obtained at each proposed specification is given in the output. This can help guide model selection. This estimator is called the modified Mahalanobis distance (MMD). See Lospinoso (2012), the manual, or the references for more information.

The following functions are pre-fabricated for ease of use, and can be passed in as the auxiliaryFunction with no extra effort; see sienaGOF-auxiliary and the examples below.

- IndegreeDistribution
- OutdegreeDistribution
- BehaviorDistribution

Value

`sienaGOF` returns a result of class `sienaGOF`; this is a list of elements of class `sienaGofTest`; if `join=TRUE`, the list has length 1; if `join=FALSE`, each list element corresponds to a period analyzed; the list elements are themselves lists again, including the following elements:

* **Observations** The observed values for the auxiliary statistics.
* **Simulations** The simulated auxiliary statistics.
* **ObservedTestStat** The observed Mahalobis distance in the data.
* **SimulatedTestStat** The Mahalobis distance for the simulations.
* **TwoTailed** Whether the p-value corresponds to a one- or two-tailed Monte Carlo test.
* **p** The p-value for the observed Mahalanobis distance in the permutation distribution of the simulated Mahalanobis distances.
* **Rank** Rank of the covariance matrix of the simulated auxiliary statistics.

Author(s)

Josh Lospinoso, modifications by Ruth Ripley and Tom Snijders

References

- See http://www.stats.ox.ac.uk/~snijders/siena/ for general information on RSiena.
- Lospinoso, J.A. and Snijders, T.A.B., "Goodness of fit for Stochastic Actor Oriented Models." Presentation given at Sunbelt XXXI, St. Pete's Beach, Fl. 2011.
- Lospinoso, J.A. (2012). "Statistical Models for Social Network Dynamics." Ph.D. Thesis. University of Oxford: U.K.

See Also

`siena07`, `sienaGOF-auxiliary`, `sienaTimeTest`

Examples

```
## Not run:
   mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
   mynet2 <- sienaDependent(array(c(s503, s502, s501), dim=c(50, 50, 3)))
   mybeh <- sienaDependent(s50a, type='behavior')
   mydata <- sienaDataCreate(mynet1, mynet2, mybeh)
   myeff <- getEffects(mydata)
   myeff <- includeEffects(myeff, transTrip)
   myeff <- includeEffects(myeff, recip, name="mynet2")
   myeff <- setEffect(myeff, cycle3, fix=TRUE, test=TRUE, include=TRUE)
   myeff  <- setEffect(myeff, nbrDist2, fix=TRUE, test=TRUE, include=TRUE)
   myeff <- setEffect(myeff, transTies, fix=TRUE, test=TRUE, include=TRUE)
   myalgorithm <- sienaAlgorithmCreate(n3=200) # Shorter phase 3, just for example.
   ans <- siena07(myalgorithm, data=mydata, effects=myeff, returnDeps=TRUE)
   gofi <- sienaGOF(ans, IndegreeDistribution, verbose=TRUE, join=TRUE,
                    varName="mynet1")
   gofi
```

```
plot(gofi)

gofi2 <- sienaGOF(ans, IndegreeDistribution, verbose=TRUE, join=TRUE,
                  varName="mynet2")
gofi2
plot(gofi2)

gofb <- sienaGOF(ans, BehaviorDistribution, varName = "mybeh",
                 verbose=TRUE, join=TRUE)
plot(gofb)

gofo <- sienaGOF(ans, OutdegreeDistribution, verbose=TRUE, join=TRUE,
             varName="mynet1", cumulative=FALSE)
# cumulative is an example of "...".
gofo
plot(gofo)

## End(Not run)
```

---

sienaGOF-auxiliary *Auxiliary functions for goodness of fit assessment by* sienaGOF

Description

The functions given here are auxiliary to function sienaGOF which assesses goodness of fit for actor-oriented models.

The auxiliary functions are, first, some functions of networks or behavior (i.e., statistics) for which the simulated values for the fitted model are compared to the observed value; second, some extraction functions to extract the observed and simulated networks and/or behavior from the sienaFit object produced by siena07 with returnDeps=TRUE.

These functions are exported here mainly to enable users to write their own versions. At the end of this help page some non-exported functions are listed. These are not exported because they depend on packages that are not in the R base distribution; and to show templates for readers wishing to contruct their own functions.

Usage

```
OutdegreeDistribution(i, obsData, sims, period, groupName, varName,
         levls=0:8, cumulative=TRUE)

IndegreeDistribution(i, obsData, sims, period, groupName, varName,
         levls=0:8, cumulative=TRUE)

BehaviorDistribution(i, obsData, sims, period, groupName, varName,
         levls=NULL, cumulative=TRUE)

sparseMatrixExtraction(i, obsData, sims, period, groupName, varName)
```

```
networkExtraction(i, obsData, sims, period, groupName, varName)

behaviorExtraction(i, obsData, sims, period, groupName, varName)
```

Arguments

i	Index number of simulation to be extracted, ranging from 1 to length(sims); if NULL, the data observation will be extracted.
obsData	The observed data set to which the model was fitted; normally this is x$f where x is the sienaFit object for which the fit is being assessed.
sims	The simulated data sets to be compared with the observed data; normally this is x$sims where x is the sienaFit object for which the fit is being assessed.
period	Period for which data and simulations are used (may run from 1 to number of waves - 1).
groupName	Name of group; relevant for multi-group data sets; defaults in sienaGOF to "Data1".
varName	Name of dependent variable.
levls	Levels used as values of the auxiliary statistic. For BehaviorDistribution, this defaults to the observed range of values.
cumulative	Are the distributions to be considered as raw or cumulative (<=) distributions?

Details

The statistics should be chosen to represent features of the network that are not explicitly fit by the estimation procedure but can be considered important properties that the model at hand should represent well. The three given here are far from a complete set; they will be supplemented in due time by statistics depending on networks and behavior jointly.

The method signature for the auxiliary statistics generally is `function(i, obsData, sims, period, groupName, varName, ...)`. For constructing new auxiliary statistics, it is helpful to study the code of OutdegreeDistribution, IndegreeDistribution, and BehaviorDistribution and of the example functions below.

Value

OutdegreeDistribution returns a named vector, the distribution of the observed or simulated outdegrees for the values in levls.

IndegreeDistribution returns a named vector, the distribution of the observed or simulated indegrees for the values in levls.

BehaviorDistribution returns a named vector, the distribution of the observed or simulated behavioral variable for the values in levls.

sparseMatrixExtraction returns the simulated network as a dgCMatrix; this is the "standard" class for sparse numeric matrices in the Matrix package. See the help file for dgCMatrix-class. Tie variables for ordered pairs with a missing value for wave=period or period+1 are zeroed; note that this also is done in RSiena for calculation of target statistics.
To treat the objects returned by this function as regular matrices, it is necessary to attach the Matrix package in your session.

networkExtraction returns the network as an edge list of class "network" according to the network package (used for package sna). Tie variables for ordered pairs with a missing value for wave=period or period+1 are zeroed; note that this also is done in RSiena for calculation of target statistics.

behaviorExtraction returns the dependent behavior variable as an integer vector. Values for actors with a missing value for wave=period or period+1 are transformed to NA.

Author(s)

Josh Lospinoso, Tom Snijders

References

- See http://www.stats.ox.ac.uk/~snijders/siena/ for general information on RSiena.
- Lospinoso, J.A. and Snijders, T.A.B., "Goodness of fit for Stochastic Actor Oriented Models." Presentation given at Sunbelt XXXI, St. Pete's Beach, Fl. 2011.
- Lospinoso, J.A. (2012). "Statistical Models for Social Network Dynamics." Ph.D. Thesis. University of Oxford: U.K.

See Also

siena07, sienaGOF

Examples

```
## Not run:
   ### For use out of the box:

   mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
   mybeh <- sienaDependent(s50a, type='behavior')
   mydata <- sienaDataCreate(mynet1, mybeh)
   myeff <- getEffects(mydata)
   myeff <- includeEffects(myeff, transTies, cycle3)
   # Shorter phases 2 and 3, just for example:
   myalgorithm <- sienaAlgorithmCreate(nsub=2, n3=300)
   ans <- siena07(myalgorithm, data=mydata, effects=myeff, returnDeps=TRUE)

   OutdegreeDistribution(NULL, ans$f, ans$sims, period=1, groupName="Data1",
                     levls=0:7, varName="mynet1")
   IndegreeDistribution(5, ans$f, ans$sims, period=1, groupName="Data1",
                     varName="mynet1")
   BehaviorDistribution(20, ans$f, ans$sims, period=1, groupName="Data1",
                     varName="mybeh")
   sparseMatrixExtraction(50, ans$f, ans$sims, period=1, groupName="Data1",
                     varName="mynet1")
   networkExtraction(100, ans$f, ans$sims, period=1, groupName="Data1",
                     varName="mynet1")
   behaviorExtraction(200, ans$f, ans$sims, period=1, groupName="Data1",
                     varName="mybeh")

   gofi <- sienaGOF(ans, IndegreeDistribution, verbose=TRUE, join=TRUE,
```

```
                varName="mynet1")
gofi
plot(gofi)

(gofo <- sienaGOF(ans, OutdegreeDistribution, verbose=TRUE, join=TRUE,
                varName="mynet1", cumulative=FALSE))
# cumulative is an example of "...".
plot(gofo)

(gofb <- sienaGOF(ans, BehaviorDistribution, varName = "mybeh",
                verbose=TRUE, join=TRUE, cumulative=FALSE))
plot(gofb)

### Here come some useful functions for building your own auxiliary statistics:
### First an extraction function.

# igraphNetworkExtraction extracts simulated and observed networks
# from the results of a siena07 run.
# It returns the network as an edge list of class "graph"
# according to the igraph package.
# Ties for ordered pairs with a missing value for wave=period or period+1
# are zeroed;
# note that this also is done in RSiena for calculation of target statistics.
igraphNetworkExtraction <- function(i, data, sims, period, groupName, varName){
  require(igraph)
  dimsOfDepVar<- attr(data[[groupName]]$depvars[[varName]], "netdims")
  missings <- is.na(data[[groupName]]$depvars[[varName]][,,period]) |
              is.na(data[[groupName]]$depvars[[varName]][,,period+1])
  if (is.null(i)) {
# sienaGOF wants the observation:
    original <- data[[groupName]]$depvars[[varName]][,,period+1]
    original[missings] <- 0
    returnValue <- graph.adjacency(original)
  }
  else
  {
    missings <- graph.adjacency(missings)
#sienaGOF wants the i-th simulation:
    returnValue <- graph.difference(
             graph.edgelist(sims[[i]][[groupName]][[varName]][[period]][,1:2]),
             missings)
  }
  returnValue
}

### Then some auxiliary statistics.

# GeodesicDistribution calculates the distribution of directed
# geodesic distances; see ?sna::geodist
# The default for \code{levls} reflects that geodesic distances larger than 5
# do not differ appreciably with respect to interpretation.
# Note that the levels of the result are named;
# these names are used in the \code{plot} method.
```

```
  GeodesicDistribution <- function (i, data, sims, period, groupName,
                           varName, levls=c(1:5,Inf), cumulative=TRUE, ...) {
    x <- networkExtraction(i, data, sims, period, groupName, varName)
    require(sna)
    a <- sna::geodist(x)$gdist
    if (cumulative)
    {
      gdi <- sapply(levls, function(i){ sum(a<=i) })
    }
else
    {
      gdi <- sapply(levls, function(i){ sum(a==i) })
    }
    names(gdi) <- as.character(levls)
    gdi
  }

  # Holland and Leinhardt Triad Census; see ?sna::triad.census.
  TriadCensus <- function(i, data, sims, wave, groupName, varName, levls=1:16){
      unloadNamespace("igraph") # to avoid package clashes
      require(sna)
      require(network)
      x <- networkExtraction(i, data, sims, wave, groupName, varName)
      tc <- sna::triad.census(x)[1,levls]
      # names are transferred automatically
      tc
  }

 # Distribution of Burt's constraint values; see ?igraph::constraint
 # the maximum finite value is 9/8 (see Buskens and van de Rijt, AJS 2008).
 ConstraintDistribution <- function (i, data, sims, period, groupName, varName,
                           levls=c(seq(0,1.125,by=0.125)), cumulative=TRUE){
    require(igraph)
    x <- igraphNetworkExtraction(i, data, sims, period, groupName, varName)
    a <- igraph::constraint(x)
    a[is.na(a)] <- Inf
    lel <- length(levls)
    if (cumulative)
    {
      cdi <- sapply(2:lel, function(i){sum(a<=levls[i])})
    }
    else
    {
      cdi <- sapply(2:lel, function(i){
                    sum(a<=levls[i]) - sum(a <= levls[i-1])})
    }
    names(cdi) <- as.character(levls[2:lel])
    cdi
   }

 ## Finally some examples of the three auxiliary statistics constructed above.

  mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
```

```
mybeh <- sienaDependent(s50a, type='behavior')
mydata <- sienaDataCreate(mynet1, mybeh)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, cycle3, nbrDist2)
myeff <- includeEffects(myeff, outdeg, name="mybeh",
         interaction1="mynet1")
myeff <- includeEffects(myeff,  outdeg, name="mybeh",
         interaction1="mynet1")
# Shorter phases 2 and 3, just for example:
myalgorithm <- sienaAlgorithmCreate(nsub=2, n3=300)
(ans2 <- siena07(myalgorithm, data=mydata, effects=myeff, returnDeps=TRUE))

gofc <- sienaGOF(ans2, ConstraintDistribution, varName="mynet1",
        verbose=TRUE, join=TRUE)
plot(gofc)

goftc <- sienaGOF(ans2, TriadCensus, varName="mynet1", verbose=TRUE, join=TRUE)
plot(goftc, center=TRUE, scale=TRUE)
# For this type of auxiliary statistics
# it is advisable in the plot to center and scale.
# note the keys at the x-axis; these names are given by sna::triad.census

gofgd <- sienaGOF(ans2, GeodesicDistribution, varName="mynet1",
         verbose=TRUE, join=TRUE, cumulative=FALSE)
plot(gofgd)
# and without infinite distances:
gofgdd <- sienaGOF(ans2, GeodesicDistribution, varName="mynet1",
          verbose=TRUE, join=TRUE, levls=1:7, cumulative=FALSE)
plot(gofgdd)

## End(Not run)
```

`sienaGroupCreate` *Function to group together several Siena data objects*

Description

Creates an object of class "sienaGroup" from a list of Siena data objects.

Usage

```
sienaGroupCreate(objlist, singleOK = FALSE, getDocumentation=FALSE)
```

Arguments

`objlist`	List of objects of class "siena"
`singleOK`	Boolean: is it OK to only have one object?
`getDocumentation`	Flag to allow documentation of internal functions, not for use by users.

Details

This function creates a Siena group object from several Siena data objects, all of which use networks, covariates and actor sets with the same names. The variables must correspond exactly between all data objects; the numbers of waves may differ. It can be used as data input to `siena07` for the multigroup option. Also used internally for convenience with a single Siena data object.

Value

An object of class "sienaGroup"; this is a list containing the input objects, with attributes:

`netnames`	names of the dependent variables in each set
`symmetric`	vector of booleans, one for each dependent variable. TRUE if all occurrences of the network are symmetric.
`structural`	vector of booleans, indicating whether structurally fixed values occur in this network
`allUpOnly`	vector of booleans, indicating whether changes are all upwards in all the occurrences of this network
`allDownOnly`	similar to previous, but for downward changes
`anyUpOnly`	vector of booleans, indicating whether changes are all upwards in any of the occurrences of this network
`anyDownOnly`	similar to previous, but for downward changes
`types`	vector of network types of the dependent variables
`observations`	Total number of periods to process
`periodNos`	Sequence of numbers of periods which are not skipped in multigroup processing
`netnodeSets`	list of names of the node sets corresponding to the dependent variables
`cCovars`	names of the constant covariates, if any
`vCovars`	names of the changing covariates, if any
`dycCovars`	names of the constant dyadic covariates, if any
`dyvCovars`	names of the changing dyadic covariates, if any
`ccnodeSets`	list of the names of the node sets corresponding to the constant covariates
`cvnodeSets`	list of the names of the node sets corresponding to the changing covariates
`dycnodeSets`	list of the names of the node sets corresponding to the constant dyadic covariates
`dyvcnodeSets`	list of the names of the node sets corresponding to the changing dyadic covariates
`compositionChange`	boolean: any composition change at all?
`exooptions`	named vector of composition change options for the node sets
`names`	Either from the input objects or "Data1", "Data2" etc
`class`	"sienaGroup" inheriting from "siena"
`balmean`	vector of means for balance calculations
`bRange`	vector of difference between maximum and minimum values for behavior variables, NA for other dependent variables

`behRange`	matrix of maximum and minimum values for behavior variables, NA for other dependent variables
`bSim`	vector of similarity means for behavior variables, NA for other dependent variables
`bPoszvar`	vector of booleans indicating positive variance for behavior variables. NA for other dependent variables
`bMoreThan2`	vector of booleans indicating whether the behavior variables take more than 2 distinct values
`cCovarPoszvar`	vector of booleans indicating positive variance for constant covariates
`cCovarMoreThan2`	vector of booleans indicating whether the constant covariates take more than 2 distinct values
`cCovarRange`	vector of difference between maximum and minimum values for constant covariates
`cCovarRange2`	matrix of maximum and minimum values for constant covariates
`cCovarSim`	vector of similarity means for constant covariates
`cCovarMean`	vector of means for constant covariates
`vCovarRange`	vector of difference between maximum and minimum values for changing covariates
`vCovarSim`	vector of similarity means for changing covariates
`vCovarMoreThan2`	vector of booleans indicating whether the changing covariates take more than 2 distinct values
`vCovarPoszvar`	vector of booleans indicating positive variance for changing covariates
`vCovarMean`	vector of means for changing covariates
`dycCovarMean`	vector of means for constant dyadic covariates
`dycCovarRange`	vector of ranges for constant dyadic covariates
`dycCovarRange2`	matrix of maximum and minimum values for constant dyadic covariates
`dyvCovarRange`	vector of ranges for changing dyadic covariates
`dyvCovarMean`	vector of means for changing dyadic covariates
`anyMissing`	vector of booleans, one for each dependent variable, indicating the presence of any missing values
`netRanges`	matrix of maximum and minimum values for dependent networks, NA for behavior variables

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

sienaDataCreate

Examples

```
Group1 <- sienaDependent(array(c(N3401, HN3401), dim=c(45, 45, 2)))
Group3 <- sienaDependent(array(c(N3403, HN3403), dim=c(37, 37, 2)))
Group4 <- sienaDependent(array(c(N3404, HN3404), dim=c(33, 33, 2)))
Group6 <- sienaDependent(array(c(N3406, HN3406), dim=c(36, 36, 2)))
dataset.1 <- sienaDataCreate(Friends = Group1)
dataset.3 <- sienaDataCreate(Friends = Group3)
dataset.4 <- sienaDataCreate(Friends = Group4)
dataset.6 <- sienaDataCreate(Friends = Group6)
FourGroups <- sienaGroupCreate(list(dataset.1, dataset.3, dataset.4, dataset.6))
```

---

sienaModelOptions *Function to allow entry of model options*

Description

Displays a Gui with model options, and allows editing of effects plus running of Siena07

Details

Called from the Apply function in siena01Gui. An internal function of siena01Gui.

Various parameters can be set on the upper part of the screen:

Estimation Method 0 for Unconditional fitting, 1 for Conditional. If there are multiple dependent variables, a list will be displayed from which to choose.

Standard starting value If checked, the estimation will ignore the initial values of the parameters in the effects object, and use the default ones.

Specify random seed If you wish your run to be repeatable, check this box and then choose any integer as the seed.

Number of processors If checked, a box will appear for you to select the number of processors to be used. All processes will run on the same machine.

Initial value of gain parameter A parameter to control the Robbins-Monro algorithm. Will be multiplied by the number of processors before use.

Number of phase 2 subphases Default 4. To omit phase 2, set this to 0.

Derivative method 0 for finite differences, 1 for score function. Default 1.

Number of phase 3 iterations Default 1000.

If you wish to restrict the degree of the simulations, enter the value in the table on the bottom left.

Desired effects can be selected by using the botton Edit effects. Change the Include column to a 1 to select, 0 to deselect.

Initial values can be specified in the initialValues column.

If it is desired to fix a parameter, set the `fix` column to 1.

To request a test, set both the `test` and `fix` columns to 1 and specify the value against which to test in the initialValue column.

Some effects have parameter values: these can be specified in the `parm` column.

Check the included effects by using the `Show included effects` button.

The model can be fitted by using the `Estimate` button.

The data objects can be saved to an R data set using `Save to file`.

The results object can be saved to an R data set using `Save results`.

The `Display Results` button is a toggle and should display or remove the display of the results file.

`Exit Model Options` allows you to return to the previous screen.

Value

None

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

siena01Gui, siena07

---

sienaNodeSet *Function to create a node set*

Description

Creates a Siena node set which can be used as the nodes in a siena network.

Usage

```
sienaNodeSet(n, nodeSetName="Actors", names=NULL)
```

Arguments

n	integer, size of set.
nodeSetName	character string naming the node set.
names	optional character string vector of length n of the names of the nodes.

Details

This function is important for data sets having more than one node set, but not otherwise.

Value

Returns a Siena node set, an integer vector, possibly with names, plus the attributes, class equal to 'sienaNodeSet', and nodeSetName equal to the argument nodeSetName.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

Examples

```
students <- sienaNodeSet(50, "student")
```

---

sienaTimeTest *Functions to assess and account for time heterogeneity of parameters*

Description

Takes a `sienaFit` object from a `siena07` estimation and tests for time heterogeneity by the addition of interactions with time dummy variables at waves `m=2...(M-1)`. The test used is the score-type test of Schweinberger (2012). Tests for joint significance, parameter-wise significance, period-wise significance, individual significance, and one-step estimates of the unrestricted model parameters are returned in a list.

Usage

```
sienaTimeTest(sienaFit, effects=NULL, excludedEffects=NULL, condition=FALSE)
```

Arguments

`sienaFit`	A `sienaFit` object returned by `siena07`.
`effects`	Optional vector of effect numbers to test. Use the number on the print of the sienaFit object.
`excludedEffects`	Optional vector of effect numbers for which time heterogeneity is not to be tested. Use the number on the print of the sienaFit object.
`condition`	Whether to orthogonalize effect-wise score-type tests and individual significance tests against estimated effects and un-estimated dummy terms, or just against estimated effects.

Details

This test follows the score type test of Schweinberger (2012) as elaborated by Lospinoso et al. (2011) by using statistics already calculated at each wave to obtain vectors of partitioned moment functions corresponding to a restricted model (the model in the `sienaFit` object; used as null hypothesis) and an unrestricted model (which contains dummies for waves `m=2...(M-1)`; used as alternative hypothesis).

`condition=TRUE` leads to a rough-and-easy approximation to controlling the mentioned tests also for the unestimated effects.

After assessing time heterogeneity, effects objects can be modified by adding numbers of all or some periods to the `timeDummy` column. This is facilitated by the `includeTimeDummy` function. For an effects object in which the `timeDummy` column of some of the included effects includes some or all period numbers, interactions of those effects with time dummies for the indicated periods will also be estimated.

An alternative to the use of `includeTimeDummy` is to define time-dependent actor covariates (dummy variables or other functions of wave number that are the same for all actors), include these in the data set through `sienaAlgorithmCreate`, and include interactions of other effects with ego effects of these time-dependent actor covariates by `includeInteraction`. Using `includeTimeDummy` is easier; using self-defined interactions with time-dependent variables gives more control.

If you wish to use this function with `sienaFit` objects that use the finite differences method of derivative estimation, or which use maximum likelihood estimation, you must request the derivatives to be returned by wave using the `byWave=TRUE` option for `siena07`.

Effects leading to dummy interactions that are collinear with the model originally fitted, after excluding the effects mentioned, will be automatically excluded from the time heterogeneity testing.

Value

`sienaTimeTest` Returns a list containing many items, including the following:

`JointTest`	A chi-squared test for joint significance of the dummies.
`EffectTest`	A chi-squared test for joint significance across dummies for each separate effect.
`GroupTest`	A chi-squared test for joint significance across dummies; if `sienaFit` is a fit for a multi-group object then these refer to each group; else they refer to ecah period.
`IndividualTest`	A matrix displaying initial estimates, one-step estimates, and p-values for the individual interactions.

Author(s)

Josh Lospinoso, Tom Snijders

References

See http://www.stats.ox.ac.uk/~snijders/siena/ for general information on RSiena.

J.A. Lospinoso, M. Schweinberger, T.A.B. Snijders, and R.M. Ripley (2011). Assessing and Accounting for Time Heterogeneity in Stochastic Actor Oriented Models. *Advances in Data Analysis and Computation*, 5:147-176.

M. Schweinberger (2012). Statistical modeling of network panel data: Goodness-of-fit. *British Journal of Statistical and Mathematical Psychology*, 65:263-281.

See Also

siena07, plot.sienaTimeTest, includeTimeDummy

Examples

```
## Not run:
## Estimate a restricted model
myalgorithm <- sienaAlgorithmCreate(nsub=2, n3=100)
mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, balance)
ans <- siena07(myalgorithm, data=mydata, effects=myeff, batch=TRUE)

## Conduct the score-type test to assess whether heterogeneity is present.
tt <- sienaTimeTest(ans)
summary(tt)

## Suppose that we wish to include two time dummies.
## Add them in the following way:
myeff <- includeTimeDummy(myeff, recip, balance, timeDummy="2")
ans2 <- siena07(myalgorithm, data=mydata, effects=myeff, batch=TRUE)

## Re-assess the time heterogeneity
tt2 <- sienaTimeTest(ans2)

## And so on..

## End(Not run)

## A demonstration of the plotting facilities, on a larger dataset:
## (Of course pasting these identical sets of three waves after each other
##  in a sequence of six is not really meaningful.)
## Not run:
myalgorithm <- sienaAlgorithmCreate(nsub=4, n3=1000)
mynet1 <- sienaDependent(array(c(s501, s502, s503, s501, s503, s502), dim=c(50, 50, 6)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, balance)
myeff <- includeTimeDummy(myeff, recip, timeDummy="2,3,5")
myeff <- includeTimeDummy(myeff, balance, timeDummy="4")
myeff <- includeTimeDummy(myeff, density, timeDummy="all")
ansp <- siena07(myalgorithm, data=mydata, effects=myeff, batch=TRUE)
ttp <- sienaTimeTest(ansp)

## Pairwise plots show
plot(ttp, pairwise=TRUE)

## Time test plots show
```

```
plot(ttp, effects=1:4, dims=c(2,2))

## End(Not run)
## A demonstration of RateX heterogeneity. Note that rate
## interactions are not implemented in general, just for
## Rate x cCovar.
## Not run:
myalgorithm <- sienaAlgorithmCreate(nsub=4, n3=1000)
mynet1 <- sienaDependent(array(c(s501, s502, s503), dim=c(50, 50, 3)))
myccov <- coCovar(s50a[,1])
mydata <- sienaDataCreate(mynet1, myccov)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, balance)
myeff <- includeTimeDummy(myeff, RateX, type="rate",
            interaction1="myccov")
ans <- siena07(myalgorithm, data=mydata, effects=myeff, batch=TRUE)

## End(Not run)
```

---

simstats0c *Versions of FRAN*

Description

The functions to be called as 'FRAN' by siena07. They call compiled C++.

Usage

```
simstats0c(z, x, data=NULL, effects=NULL, fromFiniteDiff=FALSE,
           returnDeps=FALSE, returnChains=FALSE, byWave=FALSE,
           returnDataFrame=FALSE, returnLoglik=FALSE)
maxlikec(z, x, data=NULL, effects=NULL,
           returnChains=FALSE, byGroup = FALSE, byWave=FALSE,
           returnDataFrame=FALSE, returnLoglik=FALSE,
           onlyLoglik=FALSE)
initializeFRAN(z, x, data, effects, prevAns = NULL, initC,
           profileData = FALSE, returnDeps = FALSE, returnChains =
           FALSE, byGroup = FALSE, returnDataFrame = FALSE,
           byWave = FALSE, returnLoglik = FALSE,  onlyLoglik = FALSE)
terminateFRAN(z, x)
```

Arguments

z	Control object, passed in automatically in siena07.
x	A sienaAlgorithm object, passed in automatically in siena07.
data	A sienaData object as returned by sienaDataCreate.
effects	A sienaEffects object as returned by getEffects.

fromFiniteDiff	Boolean used during calculation of derivatives by finite differences. Not for user use.
returnDeps	Boolean. Whether to return the simulated networks in Phase 3.
returnChains	Boolean. Whether to return the chains.
byWave	Boolean. Whether to return the finite difference or maximum likelihood derivatives by wave (uses a great deal of memory). Only necessary for sienaTimeTest
byGroup	Boolean. For internal use: allows different thetas for each group to be used in sienaBayes.
returnDataFrame	Boolean. Whether to return the chains as lists or data frames.
returnLoglik	Boolean. Whether to return the log likelihood of the simulated chain.
onlyLoglik	Boolean: whether to return just the likelihood for the simulated chain, plus details of steps accepted and rejected.
prevAns	An object of class "sienaFit" as returned by siena07, from which scaling information (derivative matrix and standard deviation of the deviations) will be extracted along with the latest version of the parameters which will be used as the initial values, unless the model requests the use of standard initial values. If the previous model is exactly the same as the current one, Phase 1 will be omitted. If not, any parameter estimates for effects which are included in the new model will be used as initial values, but phase 1 will still be carried out. If the results used as prevAns are a reasonable starting point, this will increase the efficiency of the algorithm.
initC	If TRUE, call is to setup the data and model in C++. For use with multiple processes only.
profileData	Boolean to force dumping of the data for profiling with sienaProfile.exe.

Details

The name of simstats0c or maxlikec should be used for the element FRAN of the model object, the former when using estimation by forward simulation, the latter for maximum likelihood estimation. The arguments with no defaults must be passed in on the call to siena07. initializeFRAN and terminateFRAN are called in both cases.

Value

simstats0c returns a list containing:

fra	Simulated statistics.
sc	Scores with which to calculate the derivative (not phase 2 or if using finite differences or maximum likelihood).
dff	Contributions to the derivative if finite differences
ntim	For conditional processing, time taken.
feasible	Currently set to TRUE.
OK	Could be set to FALSE if serious error has occurred.

sims	A list of simulation results, one for each period. Each list consists of a list for each data object, each of which consists of a list for each network, each of which consists of a list for each period, each component of which is an edgelist in matrix form (the columns are from, to, value) (or vector for behavior variables). Only if returnDeps is TRUE.

maxlikec returns a list containing:

fra	Simulated scores.
dff	Simulated Hessians: stored as lower triangular matrices
ntim	NULL, compatibility only
feasible	Currently set to TRUE.
OK	Could be set to FALSE if serious error has occurred.
dff	Simulated Hessian
sims	NULL, for compatibility only
chain	A list of sampled chains, one for each period. Each list consists of a list for each data object, each of which consists of a list for each network, each of which consists of a list for each period, each component of which is a list or a data frame depending on the value of returnDataFrame. Only if returnChainss is TRUE.
accepts	Number of accepted MH steps by dependent variable (permute steps are counted under first dependent variable)
rejects	Number of rejected MH steps by dependent variable (permute steps are counted under first dependent variable)
aborts	Number of aborted MH steps counted under first dependent variable.
loglik	Loglikelihood of the simulations. Only if returnLoglik is TRUE. If onlyLoglik is TRUE, only loglik, accepts,rejects and aborts are returned.

initializeFRAN and terminateFRAN return the control object z.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

siena07

Examples

```
mynet1 <- sienaNet(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip)
myalgorithm <- sienaAlgorithmCreate(fn=simstats0c, nsub=2, n3=100)
ans <- siena07(myalgorithm, data=mydata, effects=myeff,  batch=TRUE)
```

`summary.iwlsm` *Summary method for Iterative Weighted Least Squares Models*

Description

`summary` method for objects of class `"iwlsm"`

Usage

```
## S3 method for class 'iwlsm'
summary(object, method = c("XtX", "XtWX"),
 correlation = FALSE, ...)
```

Arguments

`object`	the fitted model. This is assumed to be the result of some fit that produces an object inheriting from the class `iwlsm`, in the sense that the components returned by the `iwlsm` function will be available.
`method`	Should the weighted (by the IWLS weights) or unweighted cross-products matrix be used?
`correlation`	logical. Should correlations be computed (and printed)?
`...`	arguments passed to or from other methods.

Details

This function is a method for the generic function `summary()` for class `"iwlsm"`. It can be invoked by calling `summary(x)` for an object `x` of the appropriate class, or directly by calling `summary.iwlsm(x)` regardless of the class of the object.

Value

If printing takes place, only a null value is returned. Otherwise, a list is returned with the following components. Printing always takes place if this function is invoked automatically as a method for the `summary` function.

`correlation`	The computed correlation coefficient matrix for the coefficients in the model.

`cov.unscaled`	The unscaled covariance matrix; i.e, a matrix such that multiplying it by an estimate of the error variance produces an estimated covariance matrix for the coefficients.
`sigma`	The scale estimate.
`stddev`	A scale estimate used for the standard errors.
`df`	The number of degrees of freedom for the model and for residuals.
`coefficients`	A matrix with three columns, containing the coefficients, their standard errors and the corresponding t statistic.
`terms`	The terms object used in fitting this model.

Author(s)

Adapted by Ruth Ripley

References

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.
See also http://www.stats.ox.ac.uk/~snijders/siena/

See Also

summary

Examples

```
## Not run:
##not enough data here for a sensible example, but shows the idea.
myalgorithm <- sienaAlgorithmCreate(nsub=2, n3=100)
mynet1 <- sienaDependent(array(c(s501, s502), dim=c(50, 50, 2)))
mynet2 <- sienaDependent(array(c(s502, s503), dim=c(50, 50, 2)))
mydata1 <- sienaDataCreate(mynet1)
mydata2 <- sienaDataCreate(mynet2)
myeff1 <- getEffects(mydata1)
myeff2 <- getEffects(mydata2)
myeff1 <- setEffect(myeff1, transTrip, fix=TRUE, test=TRUE)
myeff2 <- setEffect(myeff2, transTrip, fix=TRUE, test=TRUE)
myeff1 <- setEffect(myeff1, cycle3, fix=TRUE, test=TRUE)
myeff2 <- setEffect(myeff2, cycle3, fix=TRUE, test=TRUE)
ans1 <- siena07(myalgorithm, data=mydata1, effects=myeff1, batch=TRUE)
ans2 <- siena07(myalgorithm, data=mydata2, effects=myeff2, batch=TRUE)
meta <- siena08(ans1, ans2)
metadf <- split(meta$thetadf, meta$thetadf$effects)[[1]]
metalm <- iwlsm(theta ~ tconv, metadf, ses=se^2)
summary(metalm)

## End(Not run)
```

tmp3 *van de Bunt's Freshman dataset, time point 3*

Description

Third timepoint of van de Bunt's freshman dataset.

Format

Adjacency matrix for the network at time point 3.

Source

vrnd32t3.dat from http://www.stats.ox.ac.uk/~snijders/siena/vdBunt_data.zip

References

Van de Bunt, G.G., M.A.J. van Duijn, and T.A.B. Snijders (1999). Friendship networks through time: An actor-oriented statistical network model. *Computational and Mathematical Organization Theory*, 5, 167-192.
Also see http://www.stats.ox.ac.uk/~snijders/siena/vdBunt_data.htm.

See Also

tmp4

tmp4 *van de Bunt's Freshman dataset, time point 4*

Description

Fourth timepoint of van de Bunt's freshman dataset

Format

Adjacency matrix for the network at time point 4.

Source

vrnd32t4.dat from http://www.stats.ox.ac.uk/~snijders/siena/vdBunt_data.zip

References

Van de Bunt, G.G., M.A.J. van Duijn, and T.A.B. Snijders (1999). Friendship networks through time: An actor-oriented statistical network model. *Computational and Mathematical Organization Theory*, 5, 167-192.
Also see http://www.stats.ox.ac.uk/~snijders/siena/vdBunt_data.htm.

See Also

tmp3

---

updateTheta *Function to update the initial values of theta.*

Description

This function copies the final values of any matching selected effects from a sienaFit object to a Siena effects object.

Usage

```
updateTheta(effects, prevAns)
```

Arguments

effects	Object of class sienaEffects
prevAns	Object of class sienaFit as returned by siena07.

Details

The initial values of any selected effects in the input effects object which match an effect estimated in prevAns will be updated. If the previous run was conditional, the estimated rate parameters for the dependent variable on which the run was conditioned are added to the final value of theta.

Value

The effects object with initial value column updated.

Note

Using this function explicitly before calling siena07 rather than using it via the argument prevAns of siena07 will not permit the use of the previous derivative matrix. This will be inefficient if the new model has the same selected effects as the previous one, and the initial values in the prevAns object are close to the final estimates.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

siena07, getEffects

Examples

```
mynet1 <- sienaDependent(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip)
myalgorithm <- sienaAlgorithmCreate(nsub=2, n3=100)
ans <- siena07(myalgorithm, data=mydata, effects=myeff,  batch=TRUE)
myeff <- updateTheta(myeff, ans)
```

---

varCovar *Function to create a changing covariate object.*

Description

This function creates a changing covariate object from a matrix.

Usage

```
varCovar(val, nodeSet='Actors')
```

Arguments

val	Matrix of covariate values, one row for each actor, one column for each period.
nodeSet	Character string containing the name of the associated node set. If the entire data set contains more than one node set, then the node sets must be specified in all data objects.

Details

When part of a Siena data object, the covariate is assumed to be associated with node set nodeSet of the Siena data object.

Value

Returns the covariate as an object of class 'varCovar', in which form it can be used as an argument to SienaData.create.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

sienaDataCreate, coCovar, coDyadCovar, varDyadCovar

Examples

```
myvarCovar <- varCovar(s50a)
```

`varDyadCovar` *Function to create a changing dyadic covariate object.*

Description

This function creates a changing dyadic covariate object from an array.

Usage

```
varDyadCovar(val, nodeSets=c("Actors", "Actors"),
    sparse=is.list(val), type=c("oneMode", "bipartite"))
```

Arguments

`val`	Array of covariate values, third dimension is the time. Alternatively, a list of sparse matrices of type "dgTMatrix".
`nodeSets`	Names (character string) of the associated node sets. If the entire data set contains more than one node set, then the node sets must be specified in all data objects.
`sparse`	Boolean: whether sparse matrices or not.
`type`	oneMode or bipartite: whether the matrix refers to a one-mode or a bipartite (two-mode) network.

Details

When part of a Siena data object, the covariate is assumed to be associated with the node sets named NodeSets of the Siena data object. The names of the associated node sets will only be checked when the Siena data object is created.

Value

Returns the covariate as an object of class 'varDyadCovar', in which form it can be used as an argument to `SienaDataCreate`.

Author(s)

Ruth Ripley

References

See http://www.stats.ox.ac.uk/~snijders/siena/

See Also

sienaDataCreate, coDyadCovar, coCovar, varCovar

Examples

```
mydyadvar <- varDyadCovar(array(c(s501, s502), dim=c(50, 50, 2)))
```

---

xtable *Access xtable in package xtable*

Description

Dummy function to allow access to xtable in package xtable

Usage

```
xtable(x, ...)
```

Arguments

x	sienaFit object
...	Other arguments for xtable.sienaFit

Value

Value returned from xtable.sienaFit

Author(s)

Ruth Ripley

References

http://www.stats.ox.ac.uk/~snijders/siena/

See Also

xtable.sienaFit

Examples

```
## The function is currently defined as
function (x, ...)
{
    xtable::xtable(x, ...)
  }

myalgorithm <- sienaAlgorithmCreate(nsub=2, n3=100)
mynet1 <- sienaDependent(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
ans <- siena07(myalgorithm, data=mydata, effects=myeff, batch=TRUE)
ans
summary(ans)
xtable(ans, type='html', file='ans.html')
```

Index