

# Package ‘RSiena’

February 4, 2012

**Type** Package

**Title** Siena - Simulation Investigation for Empirical Network Analysis

**Version** 1.0.12.198

**Date** 2012-01-29

**Author** Various

**Depends** R (>= 2.10.0)

**Imports** Matrix

**Suggests** tcltk, snow, rlecuyer, network, codetools, lattice, MASS,parallel, xtable, tools

**SystemRequirements** GNU make, tcl/tk 8.5, Tktable

**Maintainer** RSiena developers <rsiena@stats.ox.ac.uk>

**Description** Fits models to longitudinal networks

**License** GPL (>= 2)

**LazyLoad** yes

**LazyData** yes

**BuildResaveData** no

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

**Repository** CRAN

**Date/Publication** 2012-02-04 14:24:13

**R topics documented:**

RSiena-package . . . . .	3
allEffects . . . . .	4
bayes . . . . .	5
coCovar . . . . .	7
coDyadCovar . . . . .	8
edit.sienaEffects . . . . .	9
effectsDocumentation . . . . .	10
getEffects . . . . .	11
includeEffects . . . . .	13
includeInteraction . . . . .	14
includeTimeDummy . . . . .	15
installGui . . . . .	17
iwlsm . . . . .	18
maxlikefn . . . . .	21
plot.sienaTimeTest . . . . .	22
print.sienaEffects . . . . .	24
print.sienaMeta . . . . .	25
print01Report . . . . .	27
s50 . . . . .	28
s501 . . . . .	29
s502 . . . . .	30
s503 . . . . .	30
s50a . . . . .	31
setEffect . . . . .	32
siena01Gui . . . . .	33
siena07 . . . . .	34
siena08 . . . . .	37
sienaCompositionChange . . . . .	40
sienaDataConstraint . . . . .	41
sienaDataCreate . . . . .	42
sienaDataCreateFromSession . . . . .	43
sienaFit.methods . . . . .	46
sienaGroupCreate . . . . .	47
sienaModelCreate . . . . .	50
sienaModelOptions . . . . .	53
sienaNet . . . . .	54
sienaNodeSet . . . . .	56
sienaTimeTest . . . . .	57
simstats0c . . . . .	59
summary.iwlsm . . . . .	62
tmp3 . . . . .	64
tmp4 . . . . .	64
updateTheta . . . . .	65
varCovar . . . . .	66
varDyadCovar . . . . .	67
xtable . . . . .	68

## Description

Fits models to longitudinal sets of networks.

## Details

Use `siena07` to fit a model.

Data objects can be created via the Gui displayed by `siena01Gui`, via `sienaDataCreateFromSession` or directly from matrices and vectors using `sienaNet`, `coCovar` etc.\ and finally `sienaDataCreate`.

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

Control of the fitting requires a *model* object, which can be created by `sienaModelCreate`.

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

Package:	RSiena
Type:	Package
Version:	1.0.12.198
Date:	2012-01-29
License:	GPL-2
LazyLoad:	yes

## Author(s)

Ruth Ripley, Krists Boitmanis.

Maintainer: RSiena developers <[rsiena@stats.ox.ac.uk](mailto:rsiena@stats.ox.ac.uk)>

## 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)
myModel <- sienaModelCreate(findiff=FALSE, fn=simstats0c, nsub=2, n3=100)
ans <- siena07(myModel, data=mydata, effects=myeff, batch=TRUE)
```

---

allEffects	<i>Internal data frame used to construct effect objects.</i>
------------	--

---

**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/>

---

 bayes

*A function for fitting Bayesian models*


---

**Description**

A function to fit a Bayesian model to Siena Data objects. Uses the function `maxlikec` for the MCMC part, the Bayesian part is performed in R.

**Usage**

```
bayes(data, effects, model, nwarm=100, nmain=100, nrunMHBatches=20,
      plotit=FALSE, nbrNodes=1, dfra=NULL, n=10,
      priorSigma=NULL, prevAns=NULL, clusterType=c("PSOCK", "FORK"),
      getDocumentation=FALSE)
```

**Arguments**

<code>data</code>	A <code>sienaData</code> object as returned by <code>sienaDataCreate</code> .
<code>effects</code>	List of <code>sienaEffects</code> objects as returned by <code>getEffects</code> .
<code>model</code>	Model object, as created by <code>sienaModelCreate</code> . Should contain all options required for the MCMC scheme, and a random seed if required.
<code>nwarm</code>	Number of iterations in the warm up phase.
<code>nmain</code>	Number of iterations in the main phase.
<code>nrunMHBatches</code>	Number of iterations of MCMC between Bayesian update steps.
<code>plotit</code>	Boolean: whether to plot parameters during the run
<code>nbrNodes</code>	Number of processes to be used. Cannot be more than the number of waves.
<code>dfra</code>	Hessian matrix. Optional
<code>n</code>	Number of iterations used to estimate Hessian
<code>priorSigma</code>	Covariance matrix to use for the prior distribution of the thetas.
<code>prevAns</code>	An object of class "sienaFit" as returned by <code>siena07</code> , 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, no estimate of the Hessian will be made. If not, any parameter estimates for effects which are included in the new model will be used as initial values, but an estimate of the Hessian will still be made. If the results used as <code>prevAns</code> are a reasonable starting point, this will increase the efficiency of the algorithm.

`clusterType` If using multiple processes, whether to use forking processes or not. (Only "PSOCK" can be used on Windows.)

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

### Details

This function wraps Bayesian sampling of parameters around calls to `maxlikec`. Unless a Hessian is supplied, one will be estimated from `n` batches of MH steps using the initial parameters. It is then transformed so the basic rate parameters are on a log scale. And inverted. It then attempts to scale the sampling covariance matrix to achieve about 40 out of 100 acceptances of Bayes proposals after single MH steps. Then a warming phase is done of `nwarm` Bayesian proposals each with 4 MH steps. Finally `nmain` repeats of (`nrunMHBatches` of `nrunMH` steps plus 1 Bayesian proposal) are performed. If `plotit` is TRUE, plots are produced at intervals during the run showing progress. They can be memory hogs, but the code may be useful for use on the returned values.

### Value

Returns a list containing, among other things:

`posteriorTot` ?

`posteriorMII` ?

`candidates` array of parameters

`acceptances` matrix of booleans: whether the corresponding change to the parameters was accepted. By group.

`MHacceptances` array of acceptances of the MH steps, by step type and group but summed over dependent variables.

`MHrejections` array of rejections of the ML steps

`MHproportions` array of proportions of the MH steps accepted

### 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)
```

```
mymodel <- sienaModelCreate(maxlike=TRUE)
ans <- bayes(mydata, myeff, mymodel)
```

---

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

### 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 `SienaData.create`.

### Author(s)

Ruth Ripley

### References

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

### See Also

[sienaDataCreate](#)

### 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.
sparse	Boolean: whether a sparse matrix or not
type	OneMode or Bipartite: whether the matrix refers to a one Mode or a bipartite 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 SienaData.create.

### Author(s)

Ruth Ripley

### References

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

### See Also

[sienaDataCreate](#)

### 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/>

## Examples

```
mynet1 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))  
mynet2 <- sienaNet(s50a, type='behavior')  
mycovar <- coCovar(rnorm(50))  
mydyadcovar <- coDyadCovar(matrix(as.numeric(rnorm(2500)) > 2), nrow=50))  
mydata <- sienaDataCreate(mynet1, mynet2, 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(type = "html", display = type == "html",  
filename = "effects")
```

### Arguments

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	Basename of the files to be written.

### Details

The allEffects object is reformatted into a table, either latex or html.

### Value

Nothing returned. Output files are created.

### Author(s)

Ruth Ripley

### References

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

### Examples

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

---

getEffects	<i>Function to create a Siena effects object</i>
------------	--

---

### Description

Creates a basic list of effects for each of the 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 lines for user defined network interactions.
behNintn	Number of lines for user defined behavior interactions.
getDocumentation	Flag to allow documentation of internal functions, not for use by users.

### Details

Considers all the elements of the input siena data object and creates lists 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.

### 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 in interaction, if any
interaction2	third variable in interaction, if any
type	"eval", "endow", or "rate"
basicRate	boolean: whether a basic rate parameter
include	boolean: include or not
randomEffects	boolean: random or fixed effect
fix	boolean: fix value or not
test	boolean: test required 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
parm	parameter values
functionType	"objective", "rate"
period	period for basic rate parameters
rateType	"Structural", "covariate"
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" effects
effectFn	here NULL, but could be replaced by a function later
statisticFn	here NULL, but could be replaced by a function later
netType	"oneMode", "behavior", "bipartite"
groupName	name of relevant data object
group	sequential number of relevant 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.

### Author(s)

Ruth Ripley

### References

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

### See Also

[sienaDataCreate](#)

### Examples

```

mynet1 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mynet2 <- sienaNet(s50a, type='behavior')
mycovar <- coCovar(rnorm(50))
mydyadcovar <- coDyadCovar(matrix(as.numeric(rnorm(2500) > 2), nrow=50))
mydata <- sienaDataCreate(mynet1, mynet2, mycovar, mydyadcovar)
myeff <- getEffects(mydata)

```

---

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 <a href="#">getEffects</a>
...	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.
interaction1	Name of siena object where needed to completely identify the effects e.g. co-variate name or behavior variable name.
interaction2	Name of siena object where needed to completely identify the effects e.g. co-variate name or behavior variable name.
character	Boolean: are the effect names character strings or not

### Details

The arguments should identify the effects completely. The include column will be set 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](#)**Examples**

```

mynet1 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mynet2 <- sienaNet(s50a, type="behavior")
mydata <- sienaDataCreate(mynet1, mynet2)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, balance)
myeff <- includeEffects(myeff, avAlt, name="mynet2", interaction1="mynet1")

```

---

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

---

**Description**

This function provides an interface to allow easy update of an unspecified interaction row 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 <a href="#">getEffects</a>
...	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 network 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. co-variate name or behavior variable name. Trailing blanks may be omitted.
interaction2	Vector of siena objects where needed to completely identify the effect e.g. co-variate 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.

**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. 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](#)

**Examples**

```
myNet1 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))
myNet2 <- sienaNet(s50a, type="behavior")
mydata <- sienaDataCreate(myNet1, myNet2)
myeff <- getEffects(mydata)
myeff <- includeInteraction(myeff, transTrip, egoX,
  interaction1=c("", "myNet2"))
```

---

includeTimeDummy

*Functions to include time dummy effects in a Siena model*

---

**Description**

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

**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 <a href="#">getEffects</a>
...	short names to identify the effects which should be included or excluded.
timeDummy	character string. Either "all" or the periods for which to create dummies, space delimited.
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.
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 arguments should identify the effects completely. The include column will be set 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)

Josh Lospinoso

### References

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

### See Also

[sienaTimeTest](#), [getEffects](#), [siena07](#)

### Examples

```
## Estimate a restricted model
mymodel <- sienaModelCreate(fn=simstats0c, nsub=2, n3=100)
mynet1 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, balance)
ans <- siena07(mymodel, data=mydata, effects=myeff, batch=TRUE)

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

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

## Re-assess the time heterogeneity
```

```
tt2 <- sienaTimeTest(ans2)

## And so on..

## A demonstration of RateX heterogeneity. Note that rate
## interactions are not implemented in general, just for
## Rate x cCovar.
## Not run:
mymodel <- sienaModelCreate(fn=simstats0c, nsub=2, n3=100)
mynet1 <- sienaNet(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(mymodel, data=mydata, effects=myeff, batch=TRUE)

## 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 $y \sim x_1 + x_2 + \dots$
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 passed 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 <code>na.omit</code> , and can be changed by <code>options(na.action=)</code> .
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?

<code>y.ret</code>	should the response be returned in the object?
<code>contrasts</code>	optional contrast specifications: see <a href="#">lm</a> .
<code>w</code>	(optional) initial down-weighting for each case. Will not work at present.
<code>init</code>	(optional) initial values for the coefficients OR a method to find initial values OR the result of a fit with a <code>coef</code> component. Known methods are "ls" (the default) for an initial least-squares fit using weights <code>w*weights</code> , and "lts" for an unweighted least-trimmed squares fit with 200 samples. Probably not functioning.
<code>psi</code>	the psi function is specified by this argument. It must give (possibly by name) a function <code>g(x, ..., deriv, w)</code> that for <code>deriv=0</code> returns <code>psi(x)/x</code> and for <code>deriv=1</code> returns some value. Extra arguments may be passed in via <code>...</code>
<code>scale.est</code>	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").
<code>k2</code>	tuning constant used for Huber proposal 2 scale estimation.
<code>maxit</code>	the limit on the number of IWLS iterations.
<code>acc</code>	the accuracy for the stopping criterion.
<code>test.vec</code>	the stopping criterion is based on changes in this vector.
<code>...</code>	additional arguments to be passed to <code>iwlsm.default</code> or to the <code>psi</code> function.
<code>lqs.control</code>	An optional list of control values for <a href="#">lqs</a> .
<code>u</code>	numeric vector of evaluation points.
<code>k</code>	tuning constant. Not used.
<code>deriv</code>	0 or 1: compute values of the psi function or of its first derivative. (Latter not used).
<code>sj2</code>	Estimated variance of the responses
<code>hh</code>	Diagonal values of the hat matrix

## Details

This function is very slightly adapted from `r1m` 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

<code>s</code>	the robust scale estimate used
<code>w</code>	the weights used in the IWLS process
<code>psi</code>	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.
mymodel <- sienaModelCreate(fn=simstats0c, nsub=2, n3=100)
mynet1 <- sienaNet(array(c(s501, s502), dim=c(50, 50, 2)))
mynet2 <- sienaNet(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(mymodel, data=mydata1, effects=myeff1, batch=TRUE)
ans2 <- siena07(mymodel, 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 `sienaModelCreate()`

**Value**

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

<code>fra</code>	Simulated scores
<code>dff</code>	2nd deriv, not phase 2
<code>OK</code>	could be set to <code>FALSE</code> 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 <- sienaNet(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(mynet1)
myeff<- getEffects(mydata)
mymodel<- sienaModelCreate(fn = maxlikefn, nsub=2, n3=100, maxlike=TRUE)
ans<- siena07(mymodel, data=mydata, effects=myeff, batch=TRUE)

## End(Not run)
```

---

`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 a 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:
mymodel <- sienaModelCreate(fn=simstats0c, nsub=4, n3=500)
mynet1 <- sienaNet(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(mymodel, data=mydata, effects=myeff, batch=FALSE)
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](#)

**Examples**

```

mynet1 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mynet2 <- sienaNet(s50a, type='behavior')
mycovar <- coCovar(rnorm(50))
mydyadcovar <- coDyadCovar(matrix(as.numeric(rnorm(2500) > 2), nrow=50))
mydata <- sienaDataCreate(mynet1, mynet2, 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 <code>sienaMeta</code>
x	An object of class <code>sienaMeta</code> , or <code>summary.sienaMeta</code> as appropriate
file	Boolean: if TRUE, sends output to file named <code>x\$projname.out</code> . 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 <http://www.stats.ox.ac.uk/~snijders/siena/>

**Examples**

```
## Not run:
mymodel <- sienaModelCreate(fn=simstats0c, nsub=2, n3=100)
mynet1 <- sienaNet(array(c(s501, s502), dim=c(50, 50, 2)))
mynet2 <- sienaNet(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)
myeff2 <- setEffect(myeff2, transTrip)
myeff1 <- setEffect(myeff1, cycle3, fix=TRUE, test=TRUE)
myeff2 <- setEffect(myeff2, cycle3, fix=TRUE, test=TRUE)
ans1 <- siena07(mymodel, data=mydata1, effects=myeff1, batch=TRUE)
ans2 <- siena07(mymodel, data=mydata2, effects=myeff2, batch=TRUE)
meta <- siena08(ans1, ans2)
meta
summary(meta)
plo <- plot(meta)
plo
plo <- plot(meta, layout = c(1,1))
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**

```
mymodel <- sienaModelCreate(findiff=TRUE, fn=simstats0c)
mynet1 <- sienaNet(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](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](http://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm)

## See Also

[s501](#), [s502](#), [s503](#), [s50a](#)

## Examples

```
mynet <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))  
mybeh <- sienaNet(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](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](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](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](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](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](http://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm)

## See Also

[s501](#), [s502](#), [s50a](#)

## Examples

```
mynet <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))  
mybeh <- sienaNet(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](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](http://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm)

## See Also

[s501](#), [s502](#), [s503](#)

## Examples

```
mynet <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))  
mybeh <- sienaNet(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 <a href="#">getEffects</a>
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 <a href="#">includeTimeDummy</a> .
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" or "endow". 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**

```
myNet1 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))
myNet2 <- sienaNet(s50a, type="behavior")
mydata <- sienaDataCreate(myNet1, myNet2)
myeff <- getEffects(mydata)
myeff <- setEffect(myeff, outInv, 3) ##parameter
```

---

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. 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](#)

### 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 on 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 [sienaModel](#) object as produced by [sienaModelCreate](#) (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 <a href="#">sienaModel</a>
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. If FALSE, multiple processes execute a single wave at each call.
...	Arguments for the simulation function, see <a href="#">simstats0c</a> : usually, data and effects; possibly also prevAns if a previous reasonable provisional estimate was obtained for a similar model.

**Details**

Runs a Robbins-Monro algorithm for parameter estimation using the three-phase implementation in Snijders (2001) and Snijders, Steglich and Schweinberger (2007), with 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.
theta	Fitted value of theta.
covtheta	Estimated covariance matrix of theta.
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 for each wave.
ssc	Score function contributions for each wave for each simulation in phase 3. Zero if finite difference method is used
sims	List of simulated networks. Each a list of edgelist, 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 sienaModel object x.

**Author(s)**

Ruth Ripley

**References**

- Schweinberger, M., and Snijders, T.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., The statistical evaluation of social network dynamics. Pp. 361-395 in *Sociological Methodology - 2001*, edited by M.E. Sobel and M.P. Becker. Boston and London: Basil Blackwell.
- Snijders, Tom A.B., Steglich, Christian E.G., and Schweinberger, Michael, 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, 2007.
- Further see <http://www.stats.ox.ac.uk/~snijders/siena/> .

**See Also**

There are print, summary and xtable methods for sienaFit objects [sienaModelCreate](#), [print.sienaFit](#)

## Examples

```

mymodel <- sienaModelCreate(fn=simstats0c, nsub=2, n3=100)
# nsub=2 and n3=100 is used here for having a brief computation, not for practice.
mynet1 <- sienaNet(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
ans <- siena07(mymodel, data=mydata, effects=myeff, batch=TRUE)

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

## End(Not run)

# or if a previous 'on track' result ans was obtained
## Not run:
ans1 <- siena07(mymodel, 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 <a href="#">siena07</a> . 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.

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.)

<code>cor.est</code>	Spearman rank correlation coefficient between estimates and their standard errors.
<code>cor.pval</code>	p-value for above
<code>regfit</code>	Part of the result of the fit of <code>iwlsm</code> .
<code>regsummary</code>	The summary of the fit, which includes the coefficient table.
<code>Tsq</code>	test statistic for effect zero in every model
<code>pTsq</code>	p-value for above
<code>tratio</code>	test statistics that mean effect is 0
<code>ptratio</code>	p-value for above
<code>Qstat</code>	Test statistic for variance of effects is zero
<code>ptilde</code>	p-value for above
<code>cjplus</code>	Test statistic for at least one theta strictly greater than 0
<code>cjminus</code>	Test statistic for at least one theta strictly less than 0
<code>cjplusp</code>	p-value for <code>cjplus</code>
<code>cjminusp</code>	p-value for <code>cjminus</code>
<code>mu.ml</code>	ML estimate of population mean
<code>mu.ml.se</code>	standard error of ML estimate of population mean
<code>sigma.ml</code>	ML estimate of population standard deviation
<code>mu.confint</code>	confidence interval for population mean based on profile likelihood
<code>sigma.confint</code>	confidence interval for population standard deviation based on profile likelihood
<code>n1</code>	Number of fits on which the meta analysis is based
<code>scoreplus</code>	Test statistic for combination of right one-sided p-values from score tests
<code>scoreminus</code>	Test statistic for combination of left one-sided p-values from score tests
<code>scoreplusp</code>	p-value for <code>scoreplus</code>
<code>scoreminusp</code>	p-value for <code>scoreminus</code>

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

### See Also

[sienaMeta](#), [siena07](#)

### Examples

```
## Not run:
mymodel <- sienaModelCreate(fn=simstats0c, nsub=2, n3=100)
mynet1 <- sienaNet(array(c(s501, s502), dim=c(50, 50, 2)))
mynet2 <- sienaNet(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(mymodel, data=mydata1, effects=myeff1, batch=TRUE)
ans2 <- siena07(mymodel, data=mydata2, effects=myeff2, batch=TRUE)
meta <- siena08(ans1, ans2)

## End(Not run)
```

---

 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.
option	Integer controlling the processing of the network entries 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.

## 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**

```
myNet1 <- sienaNet(array(c(s501,s502,s503),dim=c(50,50,3)))
myNet2 <- sienaNet(s50a,type='behavior')
mydata <- sienaDataCreate(myNet1, myNet2)
mydata <- sienaDataConstraint(mydata, myNet2, myNet1, "higher", FALSE)
```

---

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 "sienaNet", "coCovar", "varCovar", "coDyadCovar", "varDyadCovar", "compositionChange"
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 "sienaNet"
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**

[sienaNet](#), [coCovar](#), [varCovar](#), [coDyadCovar](#), [varDyadCovar](#),  
[sienaCompositionChange](#)

**Examples**

```
myNet1 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))
myNet2 <- sienaNet(s50a, type='behavior')
mydata <- sienaDataCreate(myNet1, myNet2)
```

---

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 <a href="#">siena</a> or <a href="#">sienaGroup</a>
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)
```

---

sienaFit.methods      *Methods for processing sienaFit objects*

---

## 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, ...)
```

## Arguments

object	An object of class sienaFit
x	An object of class sienaFit, or summary.sienaFit as appropriate
tstat	Boolean: add the t-statistics for convergence to the report
caption	See documentation for <a href="#">xtable</a>
label	See documentation for <a href="#">xtable</a>
align	See documentation for <a href="#">xtable</a>
digits	See documentation for <a href="#">xtable</a>
display	See documentation for <a href="#">xtable</a>
...	Add extra parameters for <a href="#">print.xtable</a> 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`.

**Author(s)**

Ruth Ripley

**References**See <http://www.stats.ox.ac.uk/~snijders/siena/>**See Also**[xtable](#), [print.xtable](#), [siena07](#)**Examples**

```
mymodel <- sienaModelCreate(fn=simstats0c, nsub=2, n3=100)
mynet1 <- sienaNet(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
ans <- siena07(mymodel, data=mydata, effects=myeff, batch=TRUE)
ans
summary(ans)
xtable(ans, type='html', file='ans.html')
```

---

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 the same networks, covariates and actor sets (although possibly different subsets of the actor set in each object). 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**

...	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?
exoptions	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**

```
## not very useful, but will work
mynet1 <- sienaNet(array(c(s501,s502),dim=c(50,50,2)))
mydata <- sienaDataCreate(mynet1)
mygroup <- sienaGroupCreate(list(mydata, mydata))
```

---

sienaModelCreate	<i>Function to create an object containing the algorithm specifications for parameter estimation in RSiena</i>
------------------	--

---

### Description

Creates an object with specifications for the algorithm for parameter estimation in RSiena.

### Usage

```
sienaModelCreate(fn, projname = "Siena", MaxDegree = 0,
  useStdInits = FALSE, n3 = 1000, nsub = 4, maxlike = FALSE,
  diag = !maxlike, condvarno = 0, condname = "", firstg = 0.2,
  cond = NA, findiff = FALSE, seed = NULL, prldg=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)
```

### 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.
maxlike	Whether to use maximum likelihood method or Method of Moments estimation.
diag	Boolean: if FALSE, use the complete estimated derivative matrix in the Robbins-Monro procedure; if TRUE just use the diagonal entries.
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. If TRUE, use conditional simulation. If missing, decision is deferred until <a href="#">siena07</a> , 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.

### 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 a model object of class "sienaModel" 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.

**Author(s)**

Ruth Ripley

**References**

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

**See Also**

This is for use in [siena07](#).

## Examples

```
mymodel <- sienaModelCreate(projname="NetworkDyn")
StdModel <- sienaModelCreate(projname="NetworkDyn",useStdInits=TRUE)
CondModel <- sienaModelCreate(projname="NetworkDyn",condvarno=1,cond=TRUE)
Max10Model <- sienaModelCreate(projname="NetworkDyn",MaxDegree=c(mynet=10))
# where mynet is the name of the network object created by sienaNet().
```

---

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 bottom 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](#)

---

sienaNet

*Function to create a Siena network object*

---

### Description

Creates a Siena network object from a matrix or array or list of sparse matrix of triples.

### Usage

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

### Arguments

netarray	matrix (type="behavior" only) or 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: set to TRUE if the data is in sparse matrix format, FALSE otherwise

**Details**

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

**Value**

An object of class "sienaNet". 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 a list of sparse matrices or not
nodeSet	Character string with name(s) of node set(s)

**Author(s)**

Ruth Ripley

**References**

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

**See Also**

[sienaDataCreate](#)

**Examples**

```

mynet1 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mynet2 <- sienaNet(s50a, type="behavior")
## note that the following example works although the node sets do not exist!
mynet3 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)),
  type="bipartite", nodeSet=c("senders", "receivers"))
## sparse matrix input - create some RSiena edgelist first
library(Matrix)
tmp1 <- as(Matrix(s501), "dgTMatrix")
tmp2 <- as(Matrix(s502), "dgTMatrix")
tmp3 <- as(Matrix(s503), "dgTMatrix")
mymat1 <- cbind(tmp1@i + 1, tmp1@j + 1, 1, 1)
mymat2 <- cbind(tmp2@i + 1, tmp2@j + 1, 1, 2)
mymat3 <- cbind(tmp3@i + 1, tmp3@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 <- sienaNet(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 <- sienaNet(list(mymat1s, mymat2s, mymat3s))
```

---

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.

### 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")
```

**Description**

Takes a `sienaFit` object from a `siena07` estimation and tests the addition of dummy parameters at waves  $m=2 \dots (M-1)$  through the score type test of Schweinberger (2007b). Tests for joint significance, parameter-wise significance, individual significance, and one step estimation of the unrestricted model parameters are returned in a list.

After assessing time heterogeneity, effects objects can be modified via the `timeDummy` column. Simply type the periods which you would like to have time dummied separated by commas into this column. Using the `timeDummy` column within the effects object, you may specify which time dummy interacted parameters are to be estimated.

If you wish to use this function with `sienaFit` objects which 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 to `siena07`.

**Usage**

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

**Arguments**

<code>sienaFit</code>	A <code>sienaFit</code> object returned by <code>siena07</code> .
<code>effects</code>	Optional vector of effect numbers to test. Use the number on the print of the <code>sienaFit</code> object.
<code>condition</code>	Whether to orthogonalize individual score tests against base effects and un-estimated dummy terms or just base effects.

**Details**

This test follows the score type test of Schweinberger (2007b) as implemented by Lospinoso et. al. (2010) by using statistics already calculated at each wave to populate vectors of partitioned moment functions corresponding to a restricted model (the model that has been fit by `sienaFit` Object) and an unrestricted model (which contains dummies for  $m=2 \dots (M-1)$ ). A covariance matrix of these statistics is calculated, and a delta matrix is constructed through the score functions' outer products with these statistics. Through an orthogonalization and a Delta method of approximation, the variance-covariance structure of the dummy statistics is calculated, and appropriate statistical tests can be used.

If it is determined that a time heterogeneity occurs for any number of time periods or effects (or any combination therein), the `timeDummy` column provides facilities within a `sienaFit` object to quickly adjust the model and re-estimate in the usual way. The `includeTimeDummy` function can be used to add the desired dummies entries to the effects object.

**Value**

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

JointTest        A chi<sup>2</sup> test for joint significance of the dummies.  
 ParameterTest   A chi<sup>2</sup> test for joint significance across dummies for each separate effect.  
 IndividualTest   A matrix displaying initial estimates, one step estimates, and a p-value for H0: the unrestricted parameters are equal to zero.

**Author(s)**

Josh Lospinoso

**References**

See <http://www.stats.ox.ac.uk/~snijders/siena/> for general information on RSiena.  
 Lospinoso, J.A., Schweinberger, M., Snijders, T.A.B, and Ripley, R.M. "Assessing and Accounting for Time Heterogeneity in Stochastic Actor Oriented Models". Advances in Data Analysis and Computation. Special Issue on Social Networks. Submitted. Available from <http://www.stats.ox.ac.uk/~lospinos/>.  
 M. Schweinberger and T.A.B. Snijders. Markov models for digraph panel data: Monte carlo-based derivative estimation. Comput. Stat. Data Anal., 51(9):4465-4483, 2007. ISSN 0167-9473.  
 M. Schweinberger. Chapter 4: Statistical Modeling of Network Panel Data: Goodness of Fit. PhD thesis, University of Groningen, 2007.

**See Also**

[siena07](#), [plot.sienaTimeTest](#), [includeTimeDummy](#)

**Examples**

```
## Estimate a restricted model
mymodel <- sienaModelCreate(fn=simstats0c, nsub=2, n3=100)
mynet1 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mydata <- sienaDataCreate(mynet1)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, balance)
ans <- siena07(mymodel, data=mydata, effects=myeff, batch=TRUE)

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

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

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

```

## And so on..

## A demonstration of the plotting facilities, on a larger dataset:
## Not run:
mymodel <- sienaModelCreate(fn=simstats0c, nsub=4, n3=500)
mynet1 <- sienaNet(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(mymodel, data=mydata, effects=myeff, batch=FALSE)
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:
mymodel <- sienaModelCreate(fn=simstats0c, nsub=2, n3=100)
mynet1 <- sienaNet(array(c(s501, s502, s503), dim=c(50, 50, 3)))
mycccov <- coCovar(s50a[,1])
mydata <- sienaDataCreate(mynet1, mycccov)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip, balance)
myeff <- includeTimeDummy(myeff, RateX, type="rate",
interaction1="mycccov")
ans <- siena07(mymodel, 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

<code>z</code>	Control object, passed in automatically in <a href="#">siena07</a> .
<code>x</code>	Model object, passed in automatically in <a href="#">siena07</a> .
<code>data</code>	A <code>sienaData</code> object as returned by <a href="#">sienaDataCreate</a> .
<code>effects</code>	A <code>sienaEffects</code> object as returned by <a href="#">getEffects</a> .
<code>fromFiniteDiff</code>	Boolean used during calculation of derivatives by finite differences. Not for user use.
<code>returnDeps</code>	Boolean. Whether to return the simulated networks in Phase 3.
<code>returnChains</code>	Boolean. Whether to return the chains.
<code>byWave</code>	Boolean. Whether to return the finite difference or maximum likelihood derivatives by wave (uses a great deal of memory). Only necessary for <a href="#">sienaTimeTest</a>
<code>byGroup</code>	Boolean. For internal use: allows different thetas for each group to be used in <a href="#">bayes</a> .
<code>returnDataFrame</code>	Boolean. Whether to return the chains as lists or data frames.
<code>returnLoglik</code>	Boolean. Whether to return the log likelihood of the simulated chain.
<code>onlyLoglik</code>	Boolean: whether to return just the likelihood for the simulated chain, plus details of steps accepted and rejected.
<code>prevAns</code>	An object of class "sienaFit" as returned by <a href="#">siena07</a> , 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 <code>prevAns</code> are a reasonable starting point, this will increase the efficiency of the algorithm.
<code>initC</code>	If TRUE, call is to setup the data and model in C++. For use with multiple processes only.
<code>profileData</code>	Boolean to force dumping of the data for profiling with <code>sienaProfile.exe</code> .

## 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. `initializeFRAN` is also called in `bayes`

## Value

`simstats0c` returns a list containing:

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

`maxlikec` returns a list containing:

<code>fra</code>	Simulated scores.
<code>dff</code>	Simulated Hessians: stored as lower triangular matrices
<code>ntim</code>	<code>NULL</code> , compatibility only
<code>feasible</code>	Currently set to <code>TRUE</code> .
<code>OK</code>	Could be set to <code>FALSE</code> if serious error has occurred.
<code>dff</code>	Simulated Hessian
<code>sims</code>	<code>NULL</code> , for compatibility only
<code>chain</code>	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 <code>returnDataFrame</code> . Only if <code>returnChainss</code> is <code>TRUE</code> .
<code>accepts</code>	Number of accepted MH steps by dependent variable (permute steps are counted under first dependent variable)
<code>rejects</code>	Number of rejected MH steps by dependent variable (permute steps are counted under first dependent variable)
<code>aborts</code>	Number of aborted MH steps counted under first dependent variable.
<code>loglik</code>	Loglikelihood of the simulations. Only if <code>returnLoglik</code> is <code>TRUE</code> . If <code>onlyLoglik</code> is <code>TRUE</code> , only <code>loglik</code> , <code>accepts</code> , <code>rejects</code> and <code>aborts</code> 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)
mymodel <- sienaModelCreate(nsub=2, n3=100)
ans <- siena07(mymodel, 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.

<code>correlation</code>	The computed correlation coefficient matrix for the coefficients in the model.
<code>cov.unscaled</code>	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.
<code>sigma</code>	The scale estimate.
<code>stddev</code>	A scale estimate used for the standard errors.
<code>df</code>	The number of degrees of freedom for the model and for residuals.
<code>coefficients</code>	A matrix with three columns, containing the coefficients, their standard errors and the corresponding t statistic.
<code>terms</code>	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.
mymodel <- sienaModelCreate(fn=simstats0c, nsub=2, n3=100)
mynet1 <- sienaNet(array(c(s501, s502), dim=c(50, 50, 2)))
mynet2 <- sienaNet(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(mymodel, data=mydata1, effects=myeff1, batch=TRUE)
ans2 <- siena07(mymodel, data=mydata2, effects=myeff2, batch=TRUE)
meta <- siena08(ans1, ans2)
metadf <- split(meta$thetadf, meta$thetadf$effects)[[1]]
metalm <- iwls(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 Stocnet

### References

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

---

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

see siena web page

### References

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

---

updateTheta	<i>Function to update the initial values of theta.</i>
-------------	--

---

### 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

<code>effects</code>	Object of class <code>sienaEffects</code>
<code>prevAns</code>	Object of class <code>sienaFit</code> as returned by <code>siena07</code> .

### Details

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. The initial values of any selected effects in the input effects object which match an effect estimated in `prevAns` will be updated.

### 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.

### Author(s)

Ruth Ripley

### References

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

### See Also

[siena07](#), [getEffects](#)

## Examples

```
myNet1 <- sienaNet(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(myNet1)
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip)
myModel <- sienaModelCreate(nsub=2, n3=100)
ans <- siena07(myModel, 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

## 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](#)

## Examples

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

---

varDyadCovar	<i>Function to create a changing dyadic covariate object.</i>
--------------	---

---

## 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
sparse	Boolean: whether sparse matrices or not.
type	OneMode or Bipartite: whether the matrix refers to a one Mode or a bipartite 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 SienaData.create.

## Author(s)

Ruth Ripley

## References

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

## See Also

[sienaDataCreate](#)

**Examples**

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

---

xtable	<i>Access xtable in package xtable</i>
--------	--

---

**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/>

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

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

mymodel <- sienaModelCreate(fn=simstats0c, nsub=2, n3=100)
mynet1 <- sienaNet(array(c(tmp3, tmp4), dim=c(32, 32, 2)))
mydata <- sienaDataCreate(mynet1)
```

```
myeff <- getEffects(mydata)
ans <- siena07(mymodel, data=mydata, effects=myeff, batch=TRUE)
ans
summary(ans)
xtable(ans, type='html', file='ans.html')
```

# Index

## \*Topic **classes**

- coCovar, 7
- coDyadCovar, 8
- getEffects, 11
- includeEffects, 13
- includeInteraction, 14
- setEffect, 32
- sienaCompositionChange, 40
- sienaDataConstraint, 41
- sienaDataCreate, 42
- sienaDataCreateFromSession, 43
- sienaGroupCreate, 47
- sienaModelCreate, 50
- sienaNet, 54
- sienaNodeSet, 56
- varCovar, 66
- varDyadCovar, 67

## \*Topic **datasets**

- allEffects, 4
- s50, 28
- s501, 29
- s502, 30
- s503, 30
- s50a, 31
- tmp3, 64
- tmp4, 64

## \*Topic **methods**

- edit.sienaEffects, 9
- print.sienaEffects, 24
- print.sienaMeta, 25

## \*Topic **method**

- sienaFit.methods, 46
- xtable, 68

## \*Topic **misc**

- effectsDocumentation, 10
- installGui, 17
- siena01Gui, 33
- sienaModelOptions, 53

## \*Topic **models**

- bayes, 5
- includeTimeDummy, 15
- iwlsm, 18
- plot.sienaTimeTest, 22
- siena07, 34
- siena08, 37
- sienaTimeTest, 57
- simstats0c, 59
- summary.iwlsm, 62
- updateTheta, 65

## \*Topic **package**

- RSiena-package, 3

## \*Topic **print**

- print01Report, 27

allEffects, 4

bayes, 5, 60, 61

coCovar, 3, 7, 43

coDyadCovar, 8, 43

edit.sienaEffects, 9

effectsDocumentation, 10

getEffects, 3, 5, 11, 13–16, 32, 33, 60, 65

includeEffects, 13

includeInteraction, 14

includeTimeDummy, 15, 32, 58

initializeFRAN (simstats0c), 59

installGui, 17

iwlsm, 18, 38

lattice, 26

lm, 19

lqs, 19

maxlikec, 5, 6

maxlikec (simstats0c), 59

maxlikefn, 21

- model.create (sienaModelCreate), 50
- na.omit, 18
- options, 18
- plot.sienaMeta (print.sienaMeta), 25
- plot.sienaTimeTest, 22, 58
- predict.iwls (iwls), 18
- print.iwls (iwls), 18
- print.sienaEffects, 24
- print.sienaFit, 36
- print.sienaFit (sienaFit.methods), 46
- print.sienaMeta, 25
- print.summary.iwls (summary.iwls), 62
- print.summary.sienaEffects
  - (print.sienaEffects), 24
- print.summary.sienaFit
  - (sienaFit.methods), 46
- print.summary.sienaMeta
  - (print.sienaMeta), 25
- print.xtable, 46, 47
- print.xtable.sienaFit
  - (sienaFit.methods), 46
- print01Report, 27
- psi.iwls (iwls), 18
- RSiena (RSiena-package), 3
- RSiena-package, 3
- s50, 28, 30
- s501, 29, 29–31
- s502, 29, 30, 31
- s503, 29, 30, 30, 31
- s50a, 29, 30, 31, 31
- setEffect, 32
- siena, 45
- siena (sienaDataCreate), 42
- siena01Gui, 3, 28, 33, 53, 54
- siena07, 3, 5, 6, 16, 22, 24, 34, 37, 39, 47, 50–52, 54, 57–62, 65
- siena08, 20, 37
- sienaCompositionChange, 40, 43
- sienaCompositionChangeFromFile
  - (sienaCompositionChange), 40
- sienaDataConstraint, 41
- sienaDataCreate, 3, 5, 7, 8, 12, 42, 42, 45, 49, 55, 60, 66, 67
- sienaDataCreateFromSession, 3, 33, 34, 43
- sienaEffects, 65
- sienaEffects (getEffects), 11
- sienaFit, 20, 36, 65, 68
- sienaFit (sienaFit.methods), 46
- sienaFit.methods, 46
- sienaGroup, 45
- sienaGroup (sienaGroupCreate), 47
- sienaGroupCreate, 47
- sienaGroupEffects (getEffects), 11
- sienaMeta, 20, 38, 39
- sienaMeta (print.sienaMeta), 25
- sienaModel, 34, 35
- sienaModel (sienaModelCreate), 50
- sienaModelCreate, 3, 5, 34, 36, 50
- sienaModelOptions, 33, 34, 53
- sienaNet, 3, 43, 54
- sienaNodeSet, 41, 56
- sienaTimeTest, 16, 22, 24, 25, 57, 60
- simstats0c, 35, 59
- summary, 63
- summary.iwls, 62
- summary.sienaEffects
  - (print.sienaEffects), 24
- summary.sienaFit (sienaFit.methods), 46
- summary.sienaMeta (print.sienaMeta), 25
- terminateFRAN (simstats0c), 59
- tmp3, 64
- tmp4, 64
- updateTheta, 65
- varCovar, 43, 66
- varDyadCovar, 43, 67
- xtable, 46, 47, 68
- xtable.sienaFit, 68
- xtable.sienaFit (sienaFit.methods), 46
- xyplot, 24