

# Package: dsem (via r-universe)

August 22, 2024

**Type** Package

**Title** Fit Dynamic Structural Equation Models

**Version** 1.3.0

**Date** 2024-07-18

**Imports** TMB, Matrix, sem, igraph, methods

**Depends** R (>= 4.0.0),

**Suggests** knitr, AER, phylopath, rmarkdown, reshape, gridExtra, dynlm,  
MARSS, ggplot2, ggpubr, ggraph, grid, vars, testthat

**Enhances** rstan, tmbstan

**LinkingTo** TMB, RcppEigen

**Description** Applies dynamic structural equation models to time-series data with generic and simplified specification for simultaneous and lagged effects. Methods are described in Thorson et al. (2024) ``Dynamic structural equation models synthesize ecosystem dynamics constrained by ecological mechanisms."`

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**LazyData** true

**URL** <https://james-thorson-noaa.github.io/dsem/>

**BugReports** <https://github.com/James-Thorson-NOAA/dsem/issues>

**NeedsCompilation** yes

**Author** James Thorson [aut, cre]

**Maintainer** James Thorson <James.Thorson@noaa.gov>

**Repository** CRAN

**Date/Publication** 2024-07-22 08:00:02 UTC

## Contents

as_fitted_DAG . . . . .	2
as_sem . . . . .	3
bering_sea . . . . .	3
classify_variables . . . . .	4
dsem . . . . .	4
dsem_control . . . . .	7
isle_royale . . . . .	9
list_parameters . . . . .	9
logLik.dsem . . . . .	10
make_dfa . . . . .	10
make_dsem_ram . . . . .	11
parse_path . . . . .	14
plot.dsem . . . . .	15
predict.dsem . . . . .	16
print.dsem . . . . .	16
residuals.dsem . . . . .	17
sea_otter . . . . .	17
simulate.dsem . . . . .	18
summary.dsem . . . . .	19
TMBAIC . . . . .	20
vcov.dsem . . . . .	20
<b>Index</b>	<b>22</b>

---

as_fitted_DAG	<i>Convert output from package dsem to phylopath</i>
---------------	--

---

### Description

Convert dsem to phylopath output

### Usage

```
as_fitted_DAG(
  fit,
  lag = 0,
  what = c("Estimate", "Std_Error", "p_value"),
  direction = 1
)
```

### Arguments

fit	Output from <a href="#">dsem</a>
lag	which lag to output
what	whether to output estimates what="Estimate", standard errors what="Std_Error" or p-values what="Std_Error"

direction      whether to include one-sided arrows direction=1, or both one- and two-sided arrows direction=c(1,2)

### Value

Convert output to format supplied by [est\\_DAG](#)

---

as_sem	<i>Convert dsem to sem output</i>
--------	-----------------------------------

---

### Description

Convert output from package dsem to sem

### Usage

```
as_sem(object, lag = 0)
```

### Arguments

object      Output from [dsem](#)  
lag      what lag to extract and visualize

### Value

Convert output to format supplied by [sem](#)

---

bering_sea	<i>Bering Sea marine ecosystem</i>
------------	------------------------------------

---

### Description

Data used to demonstrate and test ecosystem synthesis

### Usage

```
data(bering_sea)
```

---

`classify_variables`      *Classify variables path*

---

**Description**

`classify_variables` is copied from `sem:::classifyVariables`

**Usage**

```
classify_variables(model)
```

**Arguments**

`model`                  SEM model

**Details**

Copied from package ‘sem’ under licence GPL ( $\geq 2$ ) with permission from John Fox

**Value**

Tagged-list defining exogenous and endogenous variables

---

`dsem`                      *Fit dynamic structural equation model*

---

**Description**

Fits a dynamic structural equation model

**Usage**

```
dsem(  
  sem,  
  tsdata,  
  family = rep("fixed", ncol(tsdata)),  
  estimate_delta0 = FALSE,  
  control = dsem_control(),  
  covs = colnames(tsdata)  
)
```

## Arguments

sem	Specification for time-series structural equation model structure including lagged or simultaneous effects. See Details section in <a href="#">make_dsem_ram</a> for more description
tsdata	time-series data, as outputted using <a href="#">ts</a>
family	Character-vector listing the distribution used for each column of tsdata, where each element must be fixed or normal. family="fixed" is default behavior and assumes that a given variable is measured exactly. Other options correspond to different specifications of measurement error.
estimate_delta0	Boolean indicating whether to estimate deviations from equilibrium in initial year as fixed effects, or alternatively to assume that dynamics start at some stochastic draw away from the stationary distribution
control	Output from <a href="#">dsem_control</a> , used to define user settings, and see documentation for that function for details.
covs	optional: a character vector of one or more elements, with each element giving a string of variable names, separated by commas. Variances and covariances among all variables in each such string are added to the model. Warning: covs="x1, x2" and covs=c("x1", "x2") are not equivalent: covs="x1, x2" specifies the variance of x1, the variance of x2, and their covariance, while covs=c("x1", "x2") specifies the variance of x1 and the variance of x2 but not their covariance. These same covariances can be added manually via argument 'sem', but using argument 'covs' might save time for models with many variables.

## Details

A DSEM involves (at a minimum):

**Time series** a matrix  $\mathbf{X}$  where column  $\mathbf{x}_c$  for variable  $c$  is a time-series;

**Path diagram** a user-supplied specification for the path coefficients, which define the precision (inverse covariance)  $\mathbf{Q}$  for a matrix of state-variables and see [make\\_dsem\\_ram](#) for more details on the math involved.

The model also estimates the time-series mean  $\mu_c$  for each variable. The mean and precision matrix therefore define a Gaussian Markov random field for  $\mathbf{X}$ :

$$\text{vec}(\mathbf{X}) \sim \text{MVN}(\text{vec}(\mathbf{I}_T \otimes \boldsymbol{\mu}), \mathbf{Q}^{-1})$$

Users can specify a distribution for measurement errors (or assume that variables are measured without error) using argument family. This defines the link-function  $g_c(\cdot)$  and distribution  $f_c(\cdot)$  for each time-series  $c$ :

$$y_{t,c} \sim f_c(g_c^{-1}(x_{t,c}), \theta_c)$$

dsem then estimates all specified coefficients, time-series means  $\mu_c$ , and distribution measurement errors  $\theta_c$  via maximizing a log-marginal likelihood, while also estimating state-variables  $x_{t,c}$ .

summary.dsem then assembles estimates and standard errors in an easy-to-read format. Standard errors for fixed effects (path coefficients, exogenous variance parameters, and measurement error parameters) are estimated from the matrix of second derivatives of the log-marginal likelihood, and standard errors for random effects (i.e., missing or state-space variables) are estimated from a generalization of this method (see [sdreport](#) for details).

## Value

An object (list) of class 'dsem'. Elements include:

**obj** TMB object from [MakeADFun](#)

**ram** RAM parsed by `make_dsem_ram`

**model** SEM structure parsed by `make_dsem_ram` as intermediate description of model linkages

**tmb\_inputs** The list of inputs passed to [MakeADFun](#)

**opt** The output from `nlminb`

**sdrep** The output from [sdreport](#)

**internal** Objects useful for package function, i.e., all arguments passed during the call

## References

**\*\*Introducing the package, its features, and comparison with other software (to cite when using dsem):\*\***

Thorson, J. T., Andrews, A., Essington, T., Large, S. (In review). Dynamic structural equation models synthesize ecosystem dynamics constrained by ecological mechanisms.

## Examples

```
# Define model
sem = "
  # Link, lag, param_name
  cprofits -> consumption, 0, a1
  cprofits -> consumption, 1, a2
  pwage -> consumption, 0, a3
  gwage -> consumption, 0, a3
  cprofits -> invest, 0, b1
  cprofits -> invest, 1, b2
  capital -> invest, 0, b3
  gnp -> pwage, 0, c2
  gnp -> pwage, 1, c3
  time -> pwage, 0, c1
"

# Load data
data(KleinI, package="AER")
TS = ts(data.frame(KleinI, "time"=time(KleinI) - 1931))
tsdata = TS[,c("time", "gnp", "pwage", "cprofits", "consumption",
              "gwage", "invest", "capital")]

# Fit model
```

```

fit = dsem( sem=sem,
           tsdata = tsdata,
           estimate_delta0 = TRUE,
           control = dsem_control(quiet=TRUE) )
summary( fit )
plot( fit )
plot( fit, edge_label="value" )

```

---

dsem\_control

*Detailed control for dsem structure*


---

## Description

Define a list of control parameters. Note that the format of this input is likely to change more rapidly than that of [dsem](#)

## Usage

```

dsem_control(
  nlminb_loops = 1,
  newton_loops = 1,
  trace = 0,
  eval.max = 1000,
  iter.max = 1000,
  getsd = TRUE,
  quiet = FALSE,
  run_model = TRUE,
  gmrf_parameterization = c("separable", "projection"),
  constant_variance = c("conditional", "marginal", "diagonal"),
  use_REML = TRUE,
  profile = NULL,
  parameters = NULL,
  map = NULL,
  getJointPrecision = FALSE,
  extra_convergence_checks = TRUE
)

```

## Arguments

nlminb_loops	Integer number of times to call <a href="#">nlminb</a> .
newton_loops	Integer number of Newton steps to do after running <a href="#">nlminb</a> .
trace	Parameter values are printed every ‘trace’ iteration for the outer optimizer. Passed to ‘control’ in <a href="#">nlminb</a> .
eval.max	Maximum number of evaluations of the objective function allowed. Passed to ‘control’ in <a href="#">nlminb</a> .
iter.max	Maximum number of iterations allowed. Passed to ‘control’ in <a href="#">nlminb</a> .

getsd	Boolean indicating whether to call <a href="#">sdreport</a>
quiet	Boolean indicating whether to run model printing messages to terminal or not;
run_model	Boolean indicating whether to estimate parameters (the default), or instead to return the model inputs and compiled TMB object without running;
gmr_f_parameterization	Parameterization to use for the Gaussian Markov random field, where the default ‘separable’ constructs a precision matrix that must be full rank, and the alternative ‘projection’ constructs a full-rank and IID precision for variables over time, and then projects this using the inverse-cholesky of the precision, where this projection can be rank-deficient.
constant_variance	Whether to specify a constant conditional variance $\Gamma\Gamma^t$ using the default <code>constant_variance="conditional"</code> which results in a changing marginal variance along the specified causal graph when lagged paths are present. Alternatively, the user can specify a constant marginal variance using <code>constant_variance="diagonal"</code> or <code>constant_variance="marginal"</code> , such that $\Gamma$ and $\mathbf{I} - \mathbf{P}$ are rescaled to achieve this constraint. All options are equivalent when the model includes no lags (only simultaneous effects) and no covariances (no two-headed arrows). "diagonal" and "marginal" are equivalent when the model includes no covariances. Given some exogenous covariance, <code>constant_variance = "marginal"</code> preserves the conditional correlation and has changing conditional variance, while <code>constant_variance = "conditional"</code> has changing conditional correlation along the causal graph.
use_REML	Boolean indicating whether to treat non-variance fixed effects as random, either to mitigate bias in estimated variance parameters or improve efficiency for parameter estimation given correlated fixed and random effects
profile	Parameters to profile out of the likelihood (this subset will be appended to random with Laplace approximation disabled).
parameters	list of fixed and random effects, e.g., as constructed by <code>dsem</code> and then modified by hand (only helpful for advanced users to change starting values or restart at intended values)
map	list of fixed and mirrored parameters, constructed by <code>dsem</code> by default but available to override this default and then pass to <a href="#">MakeADFun</a>
getJointPrecision	whether to get the joint precision matrix. Passed to <a href="#">sdreport</a> .
extra_convergence_checks	Boolean indicating whether to run extra checks on model convergence.

## Value

An S3 object of class "dsem\_control" that specifies detailed model settings, allowing user specification while also specifying default values

---

`isle_royale`*Isle Royale wolf and moose*

---

**Description**

Data used to demonstrate and test cross-lagged (vector autoregressive) models

**Usage**

```
data(isle_royale)
```

**Details**

Data extracted from file "Data\_wolves\_moose\_Isle\_Royale\_June2019.csv" available at <https://isleroyalewolf.org/data/data/home.html> and obtained 2023-06-23. Reproduced with permission from John Vucetich, and generated by the Wolves and Moose of Isle Royale project.

**References**

Vucetich, JA and Peterson RO. 2012. The population biology of Isle Royale wolves and moose: an overview. <https://www.isleroyalewolf.org>

---

`list_parameters`*List fixed and random effects*

---

**Description**

`list_parameters` lists all fixed and random effects

**Usage**

```
list_parameters(Obj, verbose = TRUE)
```

**Arguments**

<code>Obj</code>	Compiled TMB object
<code>verbose</code>	Boolean, whether to print messages to terminal

**Value**

Tagged-list of fixed and random effects, returned invisibly and printed to screen

---

logLik.dsem	<i>Marginal log-likelihood</i>
-------------	--------------------------------

---

**Description**

Extract the (marginal) log-likelihood of a dsem model

**Usage**

```
## S3 method for class 'dsem'
logLik(object, ...)
```

**Arguments**

object	Output from <a href="#">dsem</a>
...	Not used

**Value**

object of class logLik with attributes

val	log-likelihood
df	number of parameters

Returns an object of class logLik. This has attributes "df" (degrees of freedom) giving the number of (estimated) fixed effects in the model, and "val" (value) giving the marginal log-likelihood. This class then allows AIC to work as expected.

---

make_dfa	<i>Make text for dynamic factor analysis</i>
----------	--

---

**Description**

Make the text string for a dynamic factor analysis expressed using arrow-and-lag notation for DSEM.

**Usage**

```
make_dfa(variables, n_factors, factor_names = paste0("F", seq_len(n_factors)))
```

**Arguments**

variables	Character string of variables (i.e., column names of tsdata).
n_factors	Number of factors.
factor_names	Optional character-vector of factor names, which must match NA columns in tsdata.

**Value**

A text string to be passed to [dsem](#)

---

make_dsem_ram	<i>Make a RAM (Reticular Action Model)</i>
---------------	--

---

**Description**

make\_dsem\_ram converts SEM arrow notation to ram describing SEM parameters

**Usage**

```
make_dsem_ram(
  sem,
  times,
  variables,
  covs = NULL,
  quiet = FALSE,
  remove_na = TRUE
)
```

**Arguments**

sem	Specification for time-series structural equation model structure including lagged or simultaneous effects. See Details section in <a href="#">make_dsem_ram</a> for more description
times	A character vector listing the set of times in order
variables	A character vector listing the set of variables
covs	A character vector listing variables for which to estimate a standard deviation
quiet	Boolean indicating whether to print messages to terminal
remove_na	Boolean indicating whether to remove NA values from RAM (default) or not. remove_NA=FALSE might be useful for exploration and diagnostics for advanced users

**Details****RAM specification using arrow-and-lag notation**

Each line of the RAM specification for [make\\_dsem\\_ram](#) consists of four (unquoted) entries, separated by commas:

- 1. Arrow specification:** This is a simple formula, of the form  $A \rightarrow B$  or, equivalently,  $B \leftarrow A$  for a regression coefficient (i.e., a single-headed or directional arrow);  $A \leftrightarrow A$  for a variance or  $A \leftrightarrow B$  for a covariance (i.e., a double-headed or bidirectional arrow). Here, A and B are variable names in the model. If a name does not correspond to an observed variable, then it is assumed to be a latent variable. Spaces can appear freely in an arrow specification, and there can be any number of hyphens in the arrows, including zero: Thus, e.g.,  $A \rightarrow B$ ,  $A \dashrightarrow B$ , and  $A > B$  are all legitimate and equivalent.

2. **Lag (using positive values):** An integer specifying whether the linkage is simultaneous ( $\text{Lag}=\theta$ ) or lagged (e.g.,  $X \rightarrow Y, 1$ ,  $X_{t_1} \rightarrow Y_{t_2}$  indicates that  $X$  in time  $T$  affects  $Y$  in time  $T+1$ ), where only one-headed arrows can be lagged. Using positive values to indicate lags then matches the notational convention used in package **dynlm**.
3. **Parameter name:** The name of the regression coefficient, variance, or covariance specified by the arrow. Assigning the same name to two or more arrows results in an equality constraint. Specifying the parameter name as NA produces a fixed parameter.
4. **Value:** start value for a free parameter or value of a fixed parameter. If given as NA (or simply omitted), the model is provide a default starting value.

Lines may end in a comment following #. The function extends code copied from package ‘sem’ under licence GPL ( $\geq 2$ ) with permission from John Fox.

### Simultaneous autoregressive process for simultaneous and lagged effects

This text then specifies linkages in a multivariate time-series model for variables  $\mathbf{X}$  with dimensions  $T \times C$  for  $T$  times and  $C$  variables. `make_dsem_ram` then parses this text to build a path matrix  $\mathbf{P}$  with dimensions  $TC \times TC$ , where element  $\rho_{k_2, k_1}$  represents the impact of  $x_{t_1, c_1}$  on  $x_{t_2, c_2}$ , where  $k_1 = Tc_1 + t_1$  and  $k_2 = Tc_2 + t_2$ . This path matrix defines a simultaneous equation

$$\text{vec}(\mathbf{X}) = \mathbf{P}\text{vec}(\mathbf{X}) + \text{vec}(\mathbf{\Delta})$$

where  $\mathbf{\Delta}$  is a matrix of exogenous errors with covariance  $\mathbf{V} = \mathbf{\Gamma}\mathbf{\Gamma}^t$ , where  $\mathbf{\Gamma}$  is the Cholesky of exogenous covariance. This simultaneous autoregressive (SAR) process then results in  $\mathbf{X}$  having covariance:

$$\text{Cov}(\mathbf{X}) = (\mathbf{I} - \mathbf{P})^{-1}\mathbf{\Gamma}\mathbf{\Gamma}^t((\mathbf{I} - \mathbf{P})^{-1})^t$$

Usefully, computing the inverse-covariance (precision) matrix  $\mathbf{Q} = \mathbf{V}^{-1}$  does not require inverting  $(\mathbf{I} - \mathbf{P})$ :

$$\mathbf{Q} = (\mathbf{\Gamma}^{-1}(\mathbf{I} - \mathbf{P}))^t\mathbf{\Gamma}^{-1}(\mathbf{I} - \mathbf{P})$$

### Example: univariate first-order autoregressive model

This simultaneous autoregressive (SAR) process across variables and times allows the user to specify both simutanous effects (effects among variables within year  $T$ ) and lagged effects (effects among variables among years  $T$ ). As one example, consider a univariate and first-order autoregressive process where  $T = 4$ . with independent errors. This is specified by passing `sem = "X -> X, 1, rho \n X <-> X, 0, sigma"` to `make_dsem_ram`. This is then parsed to a RAM:

heads	to	from	parameter	start
	1	2	1	<NA>
	1	3	2	<NA>
	1	4	3	<NA>
	2	1	1	<NA>
	2	2	2	<NA>
	2	3	3	<NA>
	2	4	4	<NA>

Rows of this RAM where heads=1 are then interpreted to construct the path matrix  $\mathbf{P}$ , where column "from" in the RAM indicates column number in the matrix, column "to" in the RAM indicates row number in the matrix:

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \rho & 0 & 0 & 0 \\ 0 & \rho & 0 & 0 \\ 0 & 0 & \rho & 0 \end{bmatrix}$$

While rows where heads=2 are interpreted to construct the Cholesky of exogenous covariance  $\mathbf{\Gamma}$  and column "parameter" in the RAM associates each nonzero element of those two matrices with an element of a vector of estimated parameters:

$$\mathbf{\Gamma} = \begin{bmatrix} \sigma & 0 & 0 & 0 \\ 0 & \sigma & 0 & 0 \\ 0 & 0 & \sigma & 0 \\ 0 & 0 & 0 & \sigma \end{bmatrix}$$

with two estimated parameters  $\beta = (\rho, \sigma)$ . This then results in covariance:

$$\text{Cov}(\mathbf{X}) = \sigma^2 \begin{bmatrix} 1 & \rho^1 & \rho^2 & \rho^3 \\ \rho^1 & 1 + \rho^2 & \rho^1(1 + \rho^2) & \rho^2(1 + \rho^2) \\ \rho^2 & \rho^1(1 + \rho^2) & 1 + \rho^2 + \rho^4 & \rho^1(1 + \rho^2 + \rho^4) \\ \rho^3 & \rho^2(1 + \rho^2) & \rho^1(1 + \rho^2 + \rho^4) & 1 + \rho^2 + \rho^4 + \rho^6 \end{bmatrix}$$

Which converges on the stationary covariance for an AR1 process for times  $t \gg 1$ :

$$\text{Cov}(\mathbf{X}) = \frac{\sigma^2}{1 + \rho^2} \begin{bmatrix} 1 & \rho^1 & \rho^2 & \rho^3 \\ \rho^1 & 1 & \rho^1 & \rho^2 \\ \rho^2 & \rho^1 & 1 & \rho^1 \\ \rho^3 & \rho^2 & \rho^1 & 1 \end{bmatrix}$$

except having a lower pointwise variance for the initial times, which arises as a "boundary effect".

Similarly, the arrow-and-lag notation can be used to specify a SAR representing a conventional structural equation model (SEM), cross-lagged (a.k.a. vector autoregressive) models (VAR), dynamic factor analysis (DFA), or many other time-series models.

## Value

A reticular action module (RAM) describing dependencies

## Examples

```
# Univariate AR1
sem = "
  X -> X, 1, rho
  X <-> X, 0, sigma
"
make_dsem_ram( sem=sem, variables="X", times=1:4 )
```

```

# Univariate AR2
sem = "
  X -> X, 1, rho1
  X -> X, 2, rho2
  X <-> X, 0, sigma
"
make_dsem_ram( sem=sem, variables="X", times=1:4 )

# Bivariate VAR
sem = "
  X -> X, 1, XtoX
  X -> Y, 1, XtoY
  Y -> X, 1, YtoX
  Y -> Y, 1, YtoY
  X <-> X, 0, sdX
  Y <-> Y, 0, sdY
"
make_dsem_ram( sem=sem, variables=c("X","Y"), times=1:4 )

# Dynamic factor analysis with one factor and two manifest variables
# (specifies a random-walk for the factor, and miniscule residual SD)
sem = "
  factor -> X, 0, loadings1
  factor -> Y, 0, loadings2
  factor -> factor, 1, NA, 1
  X <-> X, 0, NA, 0.01      # Fix at negligible value
  Y <-> Y, 0, NA, 0.01      # Fix at negligible value
"
make_dsem_ram( sem=sem, variables=c("X","Y","factor"), times=1:4 )

# ARIMA(1,1,0)
sem = "
  factor -> factor, 1, rho1 # AR1 component
  X -> X, 1, NA, 1         # Integrated component
  factor -> X, 0, NA, 1
  X <-> X, 0, NA, 0.01     # Fix at negligible value
"
make_dsem_ram( sem=sem, variables=c("X","factor"), times=1:4 )

# ARIMA(0,0,1)
sem = "
  factor -> X, 0, NA, 1
  factor -> X, 1, rho1     # MA1 component
  X <-> X, 0, NA, 0.01     # Fix at negligible value
"
make_dsem_ram( sem=sem, variables=c("X","factor"), times=1:4 )

```

**Description**

parse\_path is copied from sem:::parse.path

**Usage**

```
parse_path(path)
```

**Arguments**

path                    text to parse

**Details**

Copied from package 'sem' under licence GPL (>= 2) with permission from John Fox

**Value**

Tagged-list defining variables and direction for a specified path coefficient

---

plot.dsem	<i>Simulate dsem</i>
-----------	----------------------

---

**Description**

Plot from a fitted dsem model

**Usage**

```
## S3 method for class 'dsem'
plot(x, y, edge_label = c("name", "value"), digits = 2, ...)
```

**Arguments**

x                    Output from [dsem](#)  
y                    Not used  
edge\_label          Whether to plot parameter names or estimated values  
digits               integer indicating the number of decimal places to be used  
...                   arguments passed to [plot.igraph](#)

**Details**

This function coerces output from a graph and then plots the graph.

**Value**

Invisibly returns the output from [graph\\_from\\_data\\_frame](#) which was passed to [plot.igraph](#) for plotting.

---

predict.dsem	<i>predictions using dsem</i>
--------------	-------------------------------

---

**Description**

Predict variables given new (counterfactual) values of data, or for future or past times

**Usage**

```
## S3 method for class 'dsem'
predict(object, newdata = NULL, type = c("link", "response"), ...)
```

**Arguments**

object	Output from <a href="#">dsem</a>
newdata	optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted data are used to create predictions. If desiring predictions after the fitted data, the user must append rows with NAs for those future times. Similarly, if desiring predictions given counterfactual values for time-series data, then those individual observations can be edited while keeping other observations at their original fitted values.
type	the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a Poisson-distributed variable the default predictions are of log-intensity and type = "response" gives the predicted intensity.
...	Not used

**Value**

A matrix of predicted values with dimensions and order corresponding to argument newdata is provided, or tsdata if not. Predictions are provided on either link or response scale, and are generated by re-optimizing random effects condition on MLE for fixed effects, given those new data.

---

print.dsem	<i>Print fitted dsem object</i>
------------	---------------------------------

---

**Description**

Prints output from fitted dsem model

**Usage**

```
## S3 method for class 'dsem'
print(x, ...)
```

**Arguments**

x	Output from <a href="#">dsem</a>
...	Not used

**Value**

No return value, called to provide clean terminal output when calling fitted object in terminal.

---

residuals.dsem	<i>Calculate residuals</i>
----------------	----------------------------

---

**Description**

Calculate deviance or response residuals for dsem

**Usage**

```
## S3 method for class 'dsem'
residuals(object, type = c("deviance", "response"), ...)
```

**Arguments**

object	Output from <a href="#">dsem</a>
type	which type of residuals to compute (only option is "deviance" or "response" for now)
...	Not used

**Value**

A matrix of residuals, with same order and dimensions as argument `tsdata` that was passed to `dsem`.

---

sea_otter	<i>Sea otter trophic cascade</i>
-----------	----------------------------------

---

**Description**

Data used to demonstrate and test trophic cascades options

**Usage**

```
data(sea_otter)
```

---

simulate.dsem	<i>Simulate dsem</i>
---------------	----------------------

---

### Description

Simulate from a fitted dsem model

### Usage

```
## S3 method for class 'dsem'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  variance = c("none", "random", "both"),
  resimulate_gmrf = FALSE,
  ...
)
```

### Arguments

object	Output from <a href="#">dsem</a>
nsim	number of simulated data sets
seed	random seed
variance	whether to ignore uncertainty in fixed and random effects, include estimation uncertainty in random effects, or include estimation uncertainty in both fixed and random effects
resimulate_gmrf	whether to resimulate the GMRF based on estimated or simulated random effects (determined by argument variance)
...	Not used

### Details

This function conducts a parametric bootstrap, i.e., simulates new data conditional upon estimated values for fixed and random effects. The user can optionally simulate new random effects conditional upon their estimated covariance, or simulate new fixed and random effects conditional upon their imprecision.

Note that `simulate` will have no effect on states  $x_{tj}$  for which there is a measurement and when those measurements are fitted using `family="fixed"`, unless `resimulate_gmrf=TRUE`. In this latter case, the GMRF is resimulated given estimated path coefficients

### Value

Simulated data, either from `obj$simulate` where `obj` is the compiled TMB object, first simulating a new GMRF and then calling `obj$simulate`.

---

summary.dsem	<i>summarize dsem</i>
--------------	-----------------------

---

## Description

summarize parameters from a fitted dynamic structural equation model

## Usage

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

## Arguments

object	Output from <a href="#">dsem</a>
...	Not used

## Details

A DSEM is specified using "arrow and lag" notation, which specifies the set of path coefficients and exogenous variance parameters to be estimated. Function `dsem` then estimates the maximum likelihood value for those coefficients and parameters by maximizing the log-marginal likelihood. Standard errors for parameters are calculated from the matrix of second derivatives of this log-marginal likelihood (the "Hessian matrix").

However, many users will want to associate individual parameters and standard errors with the path coefficients that were specified using the "arrow and lag" notation. This task is complicated in models where some path coefficients or variance parameters are specified to share a single value a priori, or were assigned a name of NA and hence assumed to have a fixed value a priori (such that these coefficients or parameters have an assigned value but no standard error). The `summary` function therefore compiles the MLE for coefficients (including duplicating values for any path coefficients that assigned the same value) and standard error estimates, and outputs those in a table that associates them with the user-supplied path and parameter names. It also outputs the z-score and a p-value arising from a two-sided Wald test (i.e. comparing the estimate divided by standard error against a standard normal distribution).

## Value

Returns a data.frame summarizing estimated path coefficients, containing columns:

**path** The parsed path coefficient

**lag** The lag, where e.g. 1 means the predictor in time t effects the response in time t+1

**name** Parameter name

**start** Start value if supplied, and NA otherwise

**parameter** Parameter number

**first** Variable in path treated as predictor

**second** Variable in path treated as response  
**direction** Whether the path is one-headed or two-headed  
**Estimate** Maximum likelihood estimate  
**Std\_Error** Estimated standard error from the Hessian matrix  
**z\_value** Estimate divided by Std\_Error  
**p\_value** P-value associated with z\_value using a two-sided Wald test

---

TMBAIC	<i>Calculate marginal AIC for a fitted model</i>
--------	--

---

### Description

TMBAIC calculates AIC for a given model fit

### Usage

```
TMBAIC(opt, k = 2, n = Inf)
```

### Arguments

opt	the output from <code>nlminb</code> or <code>optim</code>
k	the penalty on additional fixed effects (default=2, for AIC)
n	the sample size, for use in AICc calculation (default=Inf, for which AICc=AIC)

### Value

AIC, where a parsimonious model has a AIC relative to other candidate models

---

vcov.dsem	<i>Extract Variance-Covariance Matrix</i>
-----------	---

---

### Description

extract the covariance of fixed effects, or both fixed and random effects.

### Usage

```
## S3 method for class 'dsem'
vcov(object, which = c("fixed", "random", "both"), ...)
```

### Arguments

object	output from <code>dsem</code>
which	whether to extract the covariance among fixed effects, random effects, or both
...	ignored, for method compatibility

**Value**

A square matrix containing the estimated covariances among the parameter estimates in the model. The dimensions depend upon the argument *which*, to determine whether fixed, random effects, or both are outputted.

# Index

- \* **data**
  - bering\_sea, 3
  - isle\_royale, 9
  - sea\_otter, 17
- as\_fitted\_DAG, 2
- as\_sem, 3
- bering\_sea, 3
- classify\_variables, 4
- dsem, 2, 3, 4, 7, 10, 11, 15–19
- dsem\_control, 5, 7
- est\_DAG, 3
- graph\_from\_data\_frame, 15
- isle\_royale, 9
- list\_parameters, 9
- logLik.dsem, 10
- make\_dfa, 10
- make\_dsem\_ram, 5, 11, 11
- MakeADFun, 6, 8
- nlminb, 6, 7
- parse\_path, 14
- plot.dsem, 15
- plot.igraph, 15
- predict.dsem, 16
- print.dsem, 16
- residuals.dsem, 17
- sdreport, 6, 8
- sea\_otter, 17
- sem, 3
- simulate.dsem, 18
- summary.dsem, 19
- TMBAIC, 20
- ts, 5
- vcov.dsem, 20