Package ‘dsem’

February 13, 2025

Type Package

Title Fit Dynamic Structural Equation Models

Version 1.5.0

Date 2025-02-10

Imports TMB, Matrix, sem, igraph, utils, RTMB (>= 1.7.0), ggraph, ggplot2, grid, methods, stats, ggm

Depends R (>= 4.0.0),

Suggests knitr, AER, phylopath, rmarkdown, reshape, gridExtra, dynlm, MARSS, ggpubr, vars, testthat, DHARMa

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] (<`https://orcid.org/0000-0001-7415-1010`>), Maurice Goodman [ctb] (<`https://orcid.org/0000-0002-6874-2313`>), Wouter van der Bijl [ctb] (<`https://orcid.org/0000-0002-7366-1868`>, template for d-separation test)

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

Repository CRAN

Date/Publication 2025-02-13 03:40:02 UTC

Contents

`as_fitted_DAG` *Convert output from package dsem to phylopath*

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 `dsem`
`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` *Convert dsem to sem output*

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` *Bering Sea marine ecosystem*

Description

Data used to demonstrate and test ecosystem synthesis

Usage

```
data(bering_sea)
```

cAIC *Calculate conditional AIC*

Description

Calculates the conditional Akaike Information criterion (cAIC).

Usage

```
cAIC(object, what = c("cAIC", "EDF"))
```

Arguments

`object`	Output from `dsem`
`what`	Whether to return the cAIC or the effective degrees of freedom (EDF) for each group of random effects.

Details

cAIC is designed to optimize the expected out-of-sample predictive performance for new data that share the same random effects as the in-sample (fitted) data, e.g., spatial interpolation. In this sense, it should be a fast approximation to optimizing the model structure based on k-fold crossvalidation. By contrast, `AIC` calculates the marginal Akaike Information Criterion, which is designed to optimize expected predictive performance for new data that have new random effects, e.g., extrapolation, or inference about generative parameters.

cAIC also calculates as a byproduct the effective degrees of freedom, i.e., the number of fixed effects that would have an equivalent impact on model flexibility as a given random effect.

Both cAIC and EDF are calculated using Eq. 6 of Zheng Cadigan Thorson 2024.

Note that, for models that include profiled fixed effects, these profiles are turned off.

Value

Either the cAIC, or the effective degrees of freedom (EDF) by group of random effects

References

Deriving the general approximation to cAIC used here

Zheng, N., Cadigan, N., & Thorson, J. T. (2024). A note on numerical evaluation of conditional Akaike information for nonlinear mixed-effects models (arXiv:2411.14185). arXiv. doi:10.48550/arXiv.2411.14185

The utility of EDF to diagnose hierarchical model behavior

Thorson, J. T. (2024). Measuring complexity for hierarchical models using effective degrees of freedom. Ecology, 105(7), e4327 doi:10.1002/ecy.4327

`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 (>= 2) with permission from John Fox

Value

Tagged-list defining exogenous and endogenous variables

`convert_equations` *Convert equations notation*

Description

Converts equations to arrow-and-lag notation expected by dsem

Usage

```
convert_equations(equations)
```

Arguments

`equations`	Specification for time-series structural equation model structure including lagged or simultaneous effects. See Details section in `convert_equations` for more description

Details

The function modifies code copied from package ‘sem‘ under licence GPL (>= 2) with permission from John Fox.

For specifyEquations, each input line is either a regression equation or the specification of a variance or covariance. Regression equations are of the form y = par1*x1 + par2*x2 + ... + park*xk where y and the xs are variables in the model (either observed or latent), and the pars are parameters. If a parameter is given as a numeric value (e.g., 1) then it is treated as fixed. Note that no error variable is included in the equation; error variances are specified via either the covs argument, via V(y) = par (see immediately below), or are added automatically to the model when, as by default, endog.variances=TRUE. A regression equation may be split over more than one input by breaking at a +, so that + is either the last non-blank character on a line or the first non-blank character on the subsequent line.

Variances are specified in the form V(var) = par and covariances in the form C(var1, var2) = par, where the vars are variables (observed or unobserved) in the model. The symbols V and C may be in either lower- or upper-case. If par is a numeric value (e.g., 1) then it is treated as fixed. In conformity with the RAM model, a variance or covariance for an endogenous variable in the model is an error variance or covariance.

To set a start value for a free parameter, enclose the numeric start value in parentheses after the parameter name, as parameter(value).

`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,
  prior_negloglike = NULL,
  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 `make_dsem_ram` for more description
`tsdata`	time-series data, as outputted using `ts`

`family`	Character-vector listing the distribution used for each column of `tsdata`, where each element must be `fixed` (for no measurement error), `normal` for normal measurement error using an identity link, `gamma` for a gamma measurement error using a fixed CV and log-link, `bernoulli` for a Bernoulli measurement error using a logit-link, or `poisson` for a Poisson measurement error using a log-link. `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
`prior_negloglike`	A user-provided function that takes as input the vector of fixed effects out$obj$par returns the negative log-prior probability. For example `prior_negloglike = function(obj) -1 * dnorm( obj$par[1], mean=0, sd=0.1, log=TRUE)` specifies a normal prior probability for the for the first fixed effect with mean of zero and logsd of 0.1. NOTE: this implementation does not work well with `tmbstan` and is highly experimental. If using priors, considering using `dsemRTMB` instead. The option in `dsem` is mainly intended to validate its use in `dsemRTMB`. Note that the user must load RTMB using `library(RTMB)` prior to running the model.
`control`	Output from `dsem_control`, 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 μ_c for each variable. The mean and precision matrix therefore define a Gaussian Markov random field for $\mathbf{X}$:

$$\mathrm{vec}(\mathbf{X}) \sim \mathrm{MVN}(\mathrm{vec}(\mathbf{I_T} \otimes \mu), \mathbf{Q}^{-1})$$

Users can the specify a distribution for measurement errors (or assume that variables are measured without error) using argument `family`. This defines the link-function $g_c(.)$ and distribution $f_c(.)$ 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 μ_c, and distribution measurement errors θ_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, exogenoux variance parameters, and measurement error parameters) are estimated from the matrix of second derivatives of the log-marginal likelihod, 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`

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

run_time Total time to run model

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. (2024). Dynamic structural equation models synthesize ecosystem dynamics constrained by ecological mechanisms. Methods in Ecology and Evolution. doi:10.1111/2041210X.14289

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" )
```

`dsemRTMB` *Fit dynamic structural equation model*

Description

Fits a dynamic structural equation model

Usage

```
dsemRTMB(
  sem,
  tsdata,
  family = rep("fixed", ncol(tsdata)),
  estimate_delta0 = FALSE,
  log_prior = function(p) 0,
  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 `make_dsem_ram` for more description
`tsdata`	time-series data, as outputted using `ts`
`family`	Character-vector listing the distribution used for each column of `tsdata`, where each element must be `fixed` (for no measurement error), `normal` for normal

measurement error using an identity link, `gamma` for a gamma measurement error using a fixed CV and log-link, `bernoulli` for a Bernoulli measurement error using a logit-link, or `poisson` for a Poisson measurement error using a log-link. `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

`log_prior`
: A user-provided function that takes as input the list of parameters `out$obj$env$parList()` where `out` is the output from `dsemRTMB()`, and returns the log-prior probability. For example `log_prior = function(p) dnorm( p$beta_z[1], mean=0, sd=0.1, log=TRUE)` specifies a normal prior probability for the first path coefficient with mean of zero and sd of 0.1. Note that the user must load RTMB using `library(RTMB)` prior to running the model.

`control`
: Output from `dsem_control`, 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

`dsemRTMB` is interchangeable with `dsem`, but uses RTMB instead of TMB for estimation. Both are provided for comparison and real-world comparison.

Value

An object (list) of class ‘dsem‘, fitted using RTMB

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 = dsemRTMB( sem=sem,
            tsdata = tsdata,
            estimate_delta0 = TRUE,
            control = dsem_control(quiet=TRUE) )
```

`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,
  lower = -Inf,
  upper = Inf
)
```

Arguments

`nlminb_loops`	Integer number of times to call `nlminb`.
`newton_loops`	Integer number of Newton steps to do after running `nlminb`.
`trace`	Parameter values are printed every ‘trace‘ iteration for the outer optimizer. Passed to ‘control‘ in `nlminb`.
`eval.max`	Maximum number of evaluations of the objective function allowed. Passed to ‘control‘ in `nlminb`.
`iter.max`	Maximum number of iterations allowed. Passed to ‘control‘ in `nlminb`.
`getsd`	Boolean indicating whether to call `sdreport`
`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;
`gmrf_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 $\mathbf{\Gamma}\mathbf{\Gamma}^t$ using the default `constant_variance="conditi` 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 `constant_variance="diagonal"` or `constant_variance="marginal"`, such that $\mathbf{\Gamma}$ 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, `constant_variance = "diagonal"` preserves the conditional correlation and has changing conditional variance, while `constant_variance = "marginal"` has changing conditional correlation along the causal graph.
`use_REML`	Boolean indicating whether to treat non-variance fixed effects as random, either to motigate 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 `dsem` 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 `dsem` by default but available to override this default and then pass to `MakeADFun`
`getJointPrecision`	whether to get the joint precision matrix. Passed to `sdreport`.
`extra_convergence_checks`	Boolean indicating whether to run extra checks on model convergence.

`lower`	vectors of lower bounds, replicated to be as long as start and passed to `nlminb`. If unspecified, all parameters are assumed to be unconstrained.
`upper`	vectors of upper bounds, replicated to be as long as start and passed to `nlminb`. If unspecified, all parameters are assumed to be unconstrained.

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://www.isleroyalewolf.org` 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

`Obj`	Compiled TMB object
`verbose`	Boolean, whether to print messages to terminal

Value

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

---

`logLik.dsem` *Marginal log-likelihood*

Description

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

Usage

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

Arguments

`object`	Output from `dsem`
`...`	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, abd "val" (value) giving the marginal log-likelihood. This class then allows `AIC` to work as expected.

`loo_residuals` *Calculate leave-one-out residuals*

Description

Calculates quantile residuals using the predictive distribution from a jacknife (i.e., leave-one-out predictive distribution)

Usage

```
loo_residuals(
  object,
  nsim = 100,
  what = c("quantiles", "samples", "loo"),
  track_progress = TRUE,
  ...
)
```

Arguments

`object`	Output from `dsem`
`nsim`	Number of simulations to use if `family!="fixed"` for some variable, such that simulation residuals are required.
`what`	whether to return quantile residuals, or samples from the leave-one-out predictive distribution of data, or a table of leave-one-out predictions and standard errors for the latent state
`track_progress`	whether to track runtimes on terminal
`...`	Not used

Details

Conditional quantile residuals cannot be calculated when using `family = "fixed"`, because state-variables are fixed at available measurements and hence the conditional distribution is a Dirac delta function. One alternative is to use leave-one-out residuals, where we calculate the predictive distribution for each state value when dropping the associated observation, and then either use that as the predictive distribution, or sample from that predictive distribution and then calculate a standard quantile distribution for a given non-fixed family. This appraoch is followed here. It is currently only implemented when all variables follow `family = "fixed"`, but could be generalized to a mix of families upon request.

Value

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

`make_dfa` *Make text for dynamic factor analysis*

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` *Make a RAM (Reticular Action Model)*

Description

`make_dsem_ram` converts SEM arrow notation to `ram` describing SEM parameters

Usage

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

Arguments

`sem`	Specification for time-series structural equation model structure including lagged or simultaneous effects. See Details section in `make_dsem_ram` 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 -> B` or, equivalently, `B <- A` for a regression coefficient (i.e., a single-headed or directional arrow); `A <-> A` for a variance or `A <-> 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->B`, `A --> B`, and `A>B` are all legitimate and equivalent.

2. **Lag (using positive values):** An integer specifying whether the linkage is simultaneous (`lag=0`) or lagged (e.g., `X -> Y, 1, XtoY` 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 (>= 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 ρ_{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

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

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

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

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	paarameter	start
1	2	1	1	<NA>
1	3	2	1	<NA>
1	4	3	1	<NA>
2	1	1	2	<NA>
2	2	2	2	<NA>
2	3	3	2	<NA>
2	4	4	2	<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:

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

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

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

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 )
```

`parse_path` *Parse path*

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 *Simulate dsem*

Description

Plot from a fitted dsem model

Usage

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

Arguments

x	Output from dsem
y	Not used
edge_label	Whether to plot parameter names, estimated values, or estimated values along with stars indicating significance at 0.05, 0.01, or 0.001 levels (based on two-sided Wald tests)
digits	integer indicating the number of decimal places to be used
style	Whether to make a graph using igraph or ggraph
...	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` *predictions using dsem*

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 `dsem`
`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` *Print fitted dsem object*

Description

Prints output from fitted dsem model

Usage

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

Arguments

`x`	Output from `dsem`
`...`	Not used

Value

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

`read_model` *Make a RAM (Reticular Action Model)*

Description

`read_model` converts SEM arrow notation to `model` describing SEM parameters

Usage

```
read_model(sem, times, variables, covs = NULL, quiet = FALSE)
```

Arguments

`sem`	Specification for time-series structural equation model structure including lagged or simultaneous effects. See Details section in `make_dsem_ram` 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

Details

See `make_dsem_ram` for details

`residuals.dsem` *Calculate residuals*

Description

Calculate deviance or response residuals for dsem

Usage

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

Arguments

`object`	Output from `dsem`
`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` *Sea otter trophic cascade*

Description

Data used to demonstrate and test trophic cascades options

Usage

```
data(sea_otter)
```

simulate.dsem *Simulate dsem*

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 `dsem`
`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`.

`stepwise_selection` *Simulate dsem*

Description

Plot from a fitted `dsem` model

Usage

```
stepwise_selection(model_options, model_shared, quiet = FALSE, ...)
```

Arguments

`model_options`	character-vector containing sem elements that could be included or dropped depending upon their parsimony
`model_shared`	character-vector containing sem elements that must be included regardless of parsimony
`quiet`	whether to avoid displaying progress to terminal
`...`	arguments passed to `dsem`, other than `sem` e.g., `tsdata`, `family` etc.

Details

This function conducts stepwise (i.e., forwards and backwards) model selection using marginal AIC, while forcing some model elements to be included and selecting among others.

Value

An object (list) that includes:

model the string with the selected SEM model

record a list showing the AIC and whether each `model_options` is included or not

Examples

```
# Simulate x -> y -> z
set.seed(101)
x = rnorm(100)
y = 0.5*x + rnorm(100)
z = 1*y + rnorm(100)
tsdata = ts(data.frame(x=x, y=y, z=z))

# define candidates
model_options = c(
  "y -> z, 0, y_to_z",
  "x -> z, 0, x_to_z"
)
# define paths that are required
model_shared = "
```

```
  x -> y, 0, x_to_y
"

# Do selection
step = stepwise_selection(
  model_options = model_options,
  model_shared = model_shared,
  tsdata = tsdata,
  quiet = TRUE
)

# Check selected model
cat(step$model)
```

summary.dsem *summarize dsem*

Description

summarize parameters from a fitted dynamic structural equation model

Usage

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

Arguments

object	Output from dsem
...	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

---

`test_dsep` *Test d-separation*

Description

Calculate the p-value for a test of d-separation **(Experimental)**

Usage

```
test_dsep(
  object,
  n_time = NULL,
  n_burnin = NULL,
  what = c("pvalue", "CIC", "all"),
  test = c("wald", "lr")
)
```

Arguments

`object`	object from `dsem`
`n_time`	how many times to include when defining the set of conditional independence relationships. If missing, this value is taken from the maximum lag that's included in the model plus one.
`n_burnin`	how many times to include prior to `seq_len(n_time)` when identifying the conditioning set that must be included when defining conditional independence relationships.

`what`	whether to just get the p-value, an information criterion based on the conditional independence test, or a named list with these two and other intermediate calculations (used for diagnosing test behavior)
`test`	whether to test each conditional-independence relationship using a (univariate) wald test or a (multivariate) likelihood ratio test. The likelihood-ratio test might be more accurate given estimation covariance and also faster (does not require standard errors), but also is not used by phylopath and therefore less supported by previous d-dsep testing applications.

Details

A user-specified SEM implies a set of conditional independence relationships among variables, which can be fitted individually, extracting the slope and associated p-value, and then combining these p-values to define a model-wide (omnibus) p-value for the hypothesis that a given data set arises from the specified model. This test is modified from package:phylopath. However it is unclear exactly how to define the set of conditional-independence assumptions in a model with temporal autocorrelation, and the test was not developed for uses when data are missing. At the time of writing, the function is hightly experimental.

Note that the method is not currently designed to deal with two-headed arrows among variables (i.e., exogenous covariance).

Value

A p-value representing the weight of evidence that the data arises from the specified model, where a low p-value indicates significant evidence for rejecting this hypothesis.

References

Shipley, B. (2000). A new inferential test for path models based on directed acyclic graphs. Structural Equation Modeling, 7(2), 206-218. doi:10.1207/S15328007SEM0702_4

Examples

```
# Simulate data set
set.seed(101)
a = rnorm( 100 )
b = 0.5*a + rnorm(100)
c = 1*a + rnorm(100)
d = 1*b - 0.5*c + rnorm(100)
tsdata = ts(data.frame(a=a, b=b, c=c, d=d))

# fit wrong model
wrong = dsem(
  tsdata = tsdata,
  sem = "
    a -> d, 0, a_to_d
    b -> d, 0, b_to_d
    c -> d, 0, c_to_d
  "
)
```

```
test_dsep( wrong )

# fit right model
right = dsem(
  tsdata = tsdata,
  sem = "
    a -> b, 0, a_to_b
    a -> c, 0, a_to_c
    b -> d, 0, b_to_d
    c -> d, 0, c_to_d
  "
)
test_dsep( right )
```

`TMBAIC` *Calculate marginal AIC for a fitted model*

Description

`TMBAIC` calculates AIC for a given model fit

Usage

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

Arguments

`opt`	the output from `nlminb` or `optim`
`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

---

`total_effect` *Calculate total effects*

Description

Calculate a data frame of total effects, representing the estimated effect of every variable on every other variable and any time-lag from 0 (simultaneous effects) to a user-specified maximum lag.

Usage

```
total_effect(object, n_lags = 4)
```

Arguments

`object`	Output from `dsem`
`n_lags`	Number of lags over which to calculate total effects

Details

Total effects are taken from the Leontief matrix $(\mathbf{I} - \mathbf{P})^{-1}$, where $\mathbf{P}$ is the path matrix across variables and times.

Value

A data frame listing the time-lag (lag), variable that is undergoing some exogenous change (from), and the variable being impacted (to), along with the total effect (total_effect) including direct and indirect pathways, and the partial "direct" effect (direct_effect)

`vcov.dsem` *Extract Variance-Covariance Matrix*

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 `dsem`
`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 dependend upon the argument `which`, to determine whether fixed, random effects, or both are outputted.

Index