Package: dsem (via r-universe)

October 28, 2026

Type Package

Title Dynamic Structural Equation Models

Version 3.0.0

Date 2026-06-10

Imports TMB (>= 1.9.18), Matrix, igraph, utils, RTMB (>= 1.7.0), ggraph, ggplot2, grid, methods, stats, checkmate

Depends R (>= 4.0.0),

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

Enhances rstan, tmbstan

LinkingTo TMB (>= 1.9.18), 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

VignetteBuilder knitr

LazyData true

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

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

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author James Thorson [aut, cre] (ORCID: <https://orcid.org/0000-0001-7415-1010>), Maurice Goodman [ctb] (ORCID: <https://orcid.org/0000-0002-6874-2313>), Wouter van der Bijl [ctb] (ORCID: <https://orcid.org/0000-0002-7366-1868>, template for d-separation test), Giovanni M. Marchetti [ctr] (creator of ggm, from which functions are copied to avoid dependency on package graph not on CRAN)

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

Repository https://cran.r-universe.dev

Date/Publication 2026-06-16 11:38:45 UTC

RemoteUrl https://github.com/cran/dsem

RemoteRef HEAD

RemoteSha 1b6b302d6d163d769e886d2776de346f6eb629e2

Contents

as_fitted_DAG . . . 3
bering_sea . . . 3
cAIC . . . 4
classify_variables . . . 5
convert_equations . . . 5
dsem . . . 6
dsem_control . . . 10
dsemRTMB . . . 13
fixed . . . 14
gaussian_fixed_sd . . . 14
hare_lynx . . . 15
isle_royale . . . 15
lake_washington . . . 16
list_parameters . . . 16
logLik.dsem . . . 17
lognormal . . . 17
loo_residuals . . . 18
make_dfa . . . 19
make_dsem_ram . . . 19
make_matrices . . . 23
make_msv . . . 24
paramesium_didinium . . . 25
parse_path . . . 25
partition_variance . . . 26
pdo_departure_bay . . . 27
plot.dsem . . . 28
predict.dsem . . . 29
print.dsem . . . 29
read_model . . . 30
residuals.dsem . . . 31
rk4sys . . . 31
sea_otter . . . 32
simulate.dsem . . . 32
stepwise_selection . . . 33
summary.dsem . . . 35
test_dsep . . . 36
TMBAIC . . . 38

total_effect 39
tweedie 40
vcov.dsem 41

Index **42**

`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`

`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 = Map(function(.) fixed(), colnames(tsdata)),
  estimate_delta0 = FALSE,
  estimate_mu = NULL,
  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`, with `NA` for missing values.

`family` A named list of families, each returning a class `family`, including `fixed()`, `gaussian()`, `binomial()`, `Gamma()`, `poisson()`, `lognormal()`, `tweedie()`, or `gaussian_fixed_sd()` with names that match levels of `colnames(tsdata)` to allow different families by variable. Family `fixed()` specifies that states are known (i.e., measurements for that variable have no error). Other families allow users to supply a link function including `identity`, `log`, `logit`, or `cloglog`. For example `family = list(y = binomial("logit"), x = fixed())` would specify logit-linked Bernoulli distribution for variable `tsdata$y` and a fixed (no measurement error) distribution for `tsdata$x`. For many variables, it is convenient to do e.g., `family = Map(function(.) gaussian(), colnames(tsdata))` rather than writing them all manually.

`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

`estimate_mu` character-vector listing columns of `tsdata` for which to estimate the mean, which is subtracted off of `tsdata` prior to evaluating interactions among parameters. The default `estimate_mu = NULL` estimates the mean for every column with at least one value that is not `NA` (i.e., does *not* estimate the mean for latent variables). If you want to have no `mu` parameters, use `estimate_mu = vector()`.

`prior_negloglike`
A user-provided function that takes as input the vector of fixed effects outobjpar 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. 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 $\mathbf{X}$:

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

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).

Latent variables

Any column $\mathbf{x}_c$ of `tsdata` that includes only `NA` values represents a latent variable, and all others are called manifest variables. The identifiability criteria for latent variables can be complicated. To explain, we ignore lagged effects (only simultaneous paths) and classify three types of latent variables:

factor latent variables: any latent variable $\mathbf{F}$ that includes paths out from it to manifest variables, but has no paths from manifest variables into $\mathbf{F}$ is a factor variable. These are identifable by fixing their SD (i.e., at one), and using a trimmed Cholesky parameterization (i.e., each successive factor includes fewer paths to manifest variables). See the DFA vignette for an example. Factor latent variables can be used to represent residual covariance while also estimating the source of that covariance explicitly

intermediate latent variables: Any latent variable $\mathbf{Y}$ that includes paths in from some manifest variables $\mathbf{X}$ and some paths out to manifest variables $\mathbf{Z}$ is an intermediate latent variable. In general, the at least one path in or out must be fixed a priori (e.g., at one) to identify the scale of the intermediate LV. These intermediate latent variables can represent ecological concepts that serve as intermediate link between different manifest variables

composite latent variables: Any latent variable $\mathbf{C}$ that includes paths in from some manifest variables $\mathbf{X}$ and no paths out to manifest variables is a composite latent variable. In general, you must fix all paths to composite variables a priori, and must also fix the SD a priori (e.g., at zero). These composite variables allow DSEM to estimate a response with standard errors that integrates across multiple manifest variables

As stated, these criteria do not involve paths from one to another latent variable. These are also possible, but involve more complicated identifiability criteria.

When to do (ot not do) model selection

In general, DSEM can be used for predictive modelling and/or structural causal modelling.

For predictive modelling, DSEM provides an expressive interface to specify any number of fixed effects and use these to represent the covariance among variables and over time. The predictive

error is expected to decrease when using a parsimonious model, and model selection might be appropriate using either `stepwise_selection` or some manual rule for dropping coefficients that are not statistically significant using a likelihood ratio or Wald test.

However, structural causal modelling (SCM) is necessary for models to be transferable to new environments (patterns of colinearity), or for counterfactual analysis. In general, SCM does not involve using parsimony as a basis for model selection. Instead, SCM structure should be defined based on ecological knowledge, and models can be further elaborated using tests of directional separation (see `test_dsep`).

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

Introducing moderated DSEM (e.g., nonstationarity and varying paths):

Thorson, J. T., Kristensen, K. (In press). Ecological examples of nonstationarity, nonlinearity, and statistical interactions in dynamic structural equation models. Methods in Ecology and Evolution. https://ecoevorxiv.org/repository/view/11418/

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,
  build_model = TRUE,
  gmrf_parameterization = c("gmrf_project", "full", "project", "mvn_project"),
  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,
  project_k = NULL,
  suppress_nlminb_warnings = TRUE,
  stabilize_Q = FALSE,
  logscale_moderating_variance = FALSE
)
```

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;
build_model	Boolean indicating whether to return inputs to MakeADFun
gmrf_parameterization	Parameterization to use for the Gaussian Markov random field, where the default full constructs a precision matrix that must be full rank, project constructs a full-rank and IID precision for variables over time and then projects this using the inverse-cholesky of the precision (which allows a rank-deficient GMRF, but does not allow family = "fixed"), mvn_project uses the dense variance for the full-rank component of the distribution and then projects values to the rank-deficient component (which allows family = "fixed" in any set of set of variables that is estimable, but is much slower), and gmrf_project uses a full-rank precision for all variables that have non-zero exogenous variance, and then projects to the remaining variables that have zero exogenous variance (where this is fast but cannot allow family = "fixed" combined with measurements for those variables that have zero exogenous variance)
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.

`project_k` logical-vector only used when `gmrf_parameterization=="mvn_project"`, with length `dim(tsdata)` indicating which state-values should be projected determinstically while ignoring measurements. This is useful e.g., for determinstic composite variables. If `project_k=NULL` (the default), then it projects any variable with zero exogenous variance. However, this then ignores any measurements for those variables. Manual specification can be used to confidition upon measurements for determinstic variables, but identifiability conditions are then hard to determine automatically.

`suppress_nlminb_warnings` whether to suppress uniformative warnings from `nlminb` arising when a function evaluation is NA, which are then replaced with Inf and avoided during estimation

`stabilize_Q` add `stability_eps = 1e-10` to stabilize precision (experimental)

`logscale_moderating_variance` When users supply a moderating variable for an exogenous variance (i.e., double-headed arrow), whether to exponentiate the variable prior to

Value

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

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, with NA for missing values.
family	A named list of families, each returning a class family, including fixed(), gaussian(), binomial(), Gamma(), poisson(), lognormal(), tweedie(), or gaussian_fixed_sd() with names that match levels of colnames(tsdata) to allow different families by variable. Family fixed() specifies that states are known (i.e., measurements for that variable have no error). Other families allow users to supply a link function including identity, log, logit, or cloglog. For example family = list(y = binomial("logit"), x = fixed()) would specify logit-linked Bernoulli distribution for variable tsdata$y and a fixed (no measurement error) distribution for tsdata$x. For many variables, it is convenient to do e.g., family = Map(function(.) gaussian(), colnames(tsdata)) rather than writing them all manually.
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 outobjenv$parList() where out is the output from dsemRTMB(), and returns the log-prior probability. For example log_prior = function(p) dnorm(p$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. See ?dsem for more details

Value

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

---

fixed *Family for data that are known without error*

Description

Allows using family = fixed() to specify data that have no measurement error

Usage

```
fixed()
```

---

gaussian_fixed_sd *Gaussian with known standard deviation for measurement errors*

Description

Allows using family = gaussian_fixed_sd(link, sd) to specify data that have known measurement error

Usage

```
gaussian_fixed_sd(link, sd)
```

Arguments

link	Link function
sd	vector of known standard deviations (repeated for length of time-series, so be careful)

hare_lynx *Lynx-Hare dynamics*

Description

Data used to demonstrate and test nonlinear dynamics

Usage

```
data(hare_lynx)
```

Details

records of pelts for Canada Lynx and their prey snowshoe hare from Hudson Bay 1900-1920, extracted from Gotelli (2008 Fig. 6.16) and originating elsewhere (Elton & Nicholson, 1942; MacLulich, 1937)

References

Gotelli, N. J. (2008). A Primer of Ecology, Fourth Edition by Nicholas J. Gotelli. Sinauer Associates, 2008.

Elton, C., & Nicholson, M. (1942). The Ten-Year Cycle in Numbers of the Lynx in Canada. Journal of Animal Ecology, 11(2), 215-244. doi:10.2307/1358

MacLulich, D. A. (1937). Fluctuations in the Numbers of the Varying Hare (Lepus Americanus). University of Toronto Press. doi:10.3138/9781487583064

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

lake_washington *Lake washington plankton*

Description

Data used to demonstrate and test statistical interactions

Usage

```
data(lake_washington)
```

Details

measurements of Temperature (W_t in Celcius), Cryptomonas (resource, C_t in log-abundance), Daphnia (consumer, D_t in log-abundance), and Leptodora (predator, L_t in log-abundance) in Lake Washington from 1962-1994, T=396 .

References

Hampton, S. E., Scheuerell, M. D., & Schindler, D. E. (2006). Coalescence in the Lake Washington story: Interaction strengths in a planktonic food web. Limnology and Oceanography, 51(5), 2042-2051. doi:10.4319/lo.2006.51.5.2042

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.

lognormal *Lognormal distribution*

Description

Lognormal distribution

Usage

```
lognormal(link)
```

Arguments

link	Link function

Details

Central tendancy meanlog is the median of log-response i.e., does not include bias correction term $sigma^2/2$

`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 approach 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

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

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:

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

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	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:

```
\deqn{ \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:

```
\deqn{ \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:

```
\deqn{ \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 $t >> 1$:

```
\deqn{ \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 )
```

`make_matrices` *Make path matrices*

Description

Constructs path matrices for dynamic structural equation model (DSEM) using a vector of parameters and specification of the DSEM

Usage

```
make_matrices(beta_p, model, times, variables)
```

Arguments

`beta_p`	vector parameters.

model	matrix or data.frame with the following columns, and one row per one-headed or two-headed arrow in the dynamic structural model: **direction** whether a path coefficient is one-headed (1) or two-headed (2) **lag** whether the lag associated with a given coefficient **start** starting value, used when parameter=0 **parameter** The parameter number from beta_p associated with a given path **first** The variable at the tail of a given path **second** The variable at the head of a given path
times	integer-vector of times to use when defining matrices
variables	character-vector listing variables

Details

When length(times) is T and length(variables) is J, make_matrices returns matrices of dimension $TJ \times TJ$ representing paths among $vec(\mathbf{X})$ where matrix $\mathbf{X}$ has dimension $T \times J$ and vec stacks columns into a single long vector

Value

A named list of matrices including:

P_kk The matrix of interactions, i.e., one-headed arrows

G_kk The matrix of exogenous covariance, i.e., two-headed arrows

---

make_msv *Make text for multivariate stochastic volatility model*

Description

Make the text string for a multivariate stochastic volatility model using arrow-lag-slope notation for DSEM.

Usage

```
make_msv(
  variables,
  n_factors,
  factor_names = paste0("F", seq_len(n_factors)),
  collapse_text = TRUE
)
```

Arguments

variables	Character string of variables (i.e., column names of tsdata).
n_factors	Number of stochastic volatility factors.
factor_names	Optional character-vector of factor names, which must match NA columns in tsdata.
collapse_text	whether to collapse text into long character string

Value

A text string to be passed to dsem

`paramesium_didinium` *Paramesium-Didinium dynamics*

Description

Data used to demonstrate and test nonlinear dynamics

Usage

```
data(paramesium_didinium)
```

Details

records of Paramesium aurelia and Didinium nasutum in a microcosm experiment at 0.5 Cerophyll concentration measured every 12 hours over 35 days, i.e., T=71 (Veilleux, 1976 Fig. 11a), as previously digitized (Jost & Ellner, 2000 Fig. 1).

References

Veilleux, B. G. (1979). An Analysis of the Predatory Interaction Between Paramecium and Didinium. Journal of Animal Ecology, 48(3), 787-803. doi:10.2307/4195

Jost, C., & Ellner, S. P. (2000). Testing for predator dependence in predator-prey dynamics: A non-parametric approach. Proceedings of the Royal Society of London. Series B: Biological Sciences. doi:10.1098/rspb.2000.1186

`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

---

`partition_variance` *Partition variance in one variable due to another (EXPERIMENTAL)*

Description

Calculate the proportion of variance for a response variable that is attributed to another set of predictor variables, calculated across lags from from 0 (simultaneous effects) to a user-specified maximum lag.

Usage

```
partition_variance(object, which_response, n_times = 10)
```

Arguments

`object`	Output from `dsem`
`which_response`	string matching colnames from `tsdata` identifying response variable
`n_times`	Number of lags over which to calculate total effects

Details

This function calculates the variance for each variable and lag, and then recalculates it when setting exogenous variance to zero for all variables except `which_pred`. It then calculates the ratio of the diagonal of these two. This represents the proportion of variance in the full model that is attributable to one or more variables.

This function is under development and may still change or be removed.

Value

A list with two elements:

total_variance A matrix of the total variance for each variable (column) and each time from 1 to `n_times`

proportion_variance_explained A matrix of the proportion of variance explained for variable `which_response` by each model variable (column) and each time from 1 to `n_times`

Note that in a model with lagged effects, the total_variance and variance_explained will vary for each time (row), and the analyst might want to either choose a time for which the value has stabilized.

Examples

```
# Simulate linear model
x = rnorm(100)
y = 1 + 1 * x + rnorm(100)
data = data.frame(x=x, y=y)

# Fit as DSEM
fit = dsem( sem = "x -> y, 0, beta",
            tsdata = ts(data),
            control = dsem_control(quiet=TRUE) )

# Apply
partition_variance( fit,
                    which_response = "y",
                    n_times = 10 )
```

---

pdo_departure_bay *Pacific Decadal Oscillation and Departure Bay temperatures*

Description

Data used to demonstrate and test nonstationary dynamics

Usage

```
data(pdo_departure_bay)
```

Details

records of the Pacific Decadal Oscillation by year (row) and month (Jan-Dec, columns), downloaded from JISAO on Nov. 6, 2018, as analyzed previously by Thorson et al. (2020)

References

Thorson, J. T., Ciannelli, L., & Litzow, M. A. (2020). Defining indices of ecosystem variability using biological samples of fish communities: A generalization of empirical orthogonal functions. Progress in Oceanography, 181, 102244. doi:10.1016/j.pocean.2019.102244

`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.

rk4sys *Classical Runge-Kutta for system of equations*

Description

Classical Runge-Kutta of order 4.

Usage

```
rk4sys(f, a, b, y0, n, Pars, ...)
```

Arguments

f	function in the differential equation $y' = f(x, y)$; defined as a function $R \times R^m \to R^m$, where m is the number of equations.
a	starting time for the interval to integrate
b	ending time for the interval to integrate.
y0	starting values at time a
n	the number of steps from a to b.
Pars	named list of parameters passed to f
...	additional inputs to function f

Details

Classical Runge-Kutta of order 4 for (systems of) ordinary differential equations with fixed step size. Copied from pracma under GPL-3, with small modifications to work with RTMB

Value

List with components x for grid points between a and b and y an n-by-m matrix with solutions for variables in columns, i.e. each row contains one time stamp.

---

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,
  fill_missing = 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)
fill_missing	whether to fill in simulate all data (including values that are missing in the original data set)
...	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,
  options_initial = c(),
  quiet = FALSE,
  criterion = AIC,
  ...
)
```

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
options_initial	character-vector containing some (possible empty) subset of model_options, where stepwise selection begins with that set of model options included.
quiet	whether to avoid displaying progress to terminal
criterion	function that computes the information criterion to be minimized, typically using AIC. However, users can instead supply a function that computes CIC using test_dsep and desired settings, presumably including a set.seed if missing data are being imputed
...	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. See link{dsem} for further discussion of model selection.

Value

An object (list) that includes:

model the string with the selected SEM model

step a list, where each list element shows the models fitted during one step in the stepwise algorithm, from first to last step. Each step then lists a table, where each table row is a single fitted model, the first column is the AIC for that model, and the subsequent columns show whether each variable is included (1) or not (0)

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"),
  seed = 123456,
  order = NULL,
  impute_data = c("by_test", "single", "none")
)
```

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.
seed	random number seed used when simulating imputed data, so that results are reproducible.
order	an optional character vector providing the order for variables to be tested when defining the directed acyclic graph for use in d-sep testing
impute_data	whether to independently impute missing data for each conditional independence test, or to use imputed values from the original fit. The data are imputed separately for each conditional independence test, so that they are uncorrelated as expected when combining them using Fisher's method. Preliminary testing suggests that using imputed data improves test performance

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, resulting from a pulse experiment (i.e., an exogenous and temporary change in a single variable in time `t=0`) or a press experiment (i.e., an exogenous and permanent change in a single variable starting in time `t=0` and continuing for `n_lags` times), representing the estimated effect of a change in any 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, type = c("pulse", "press"))
```

Arguments

`object`	Output from `dsem`
`n_lags`	Number of lags over which to calculate total effects
`type`	Whether a pulse or press experiment is intended. A pulse experiment answers the question: `What happens if a variable is changed for only a single time-interval?" A pres` happens if a variable is permanently changed starting in a given time-interval?

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. $(\mathbf{I} - \mathbf{P})^{-1}\delta$ calculates the effect of a perturbation represented by vector δ with length $n_{\text{lags}} \times n_{\text{J}}$ where n_{J} is the number of variables. $(\mathbf{I} - \mathbf{P})^{-1}\delta$ calculates the total effect of a given variable (from) upon any other variable (to) either in the same time ($t = 0$), or subsequent times ($t \geq 1$), where $\delta = \mathbf{i}_{\text{T}} \otimes \mathbf{i}_{\text{J}}$, where $\mathbf{i}_{\text{J}}$ is one for the `from` variable and zero otherwise. For a pulse experiment, $\mathbf{i}_{\text{T}}$ is one at $t = 0$ and zero for other times. For a press experiment, $\mathbf{i}_{\text{T}}$ is one for all times.

We compute and list the total effect at each time from time `t=0` to `t=n_lags-1`. For press experiments, this includes transient values as the the total effect approaches its asymptotic value (if this exists) as t approaches infinity. If the analyst wants an asymptotic effect from a press experiment, we recommend using a high lag (e.g., `n_lags = 100`) and then confirming that it has reached it's asymptote (i.e., the total effect is almost identical for the last and next-to-last lag), and then reporting the value for that last lag.

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)

Examples

```
### EXAMPLE 1
# Define linear model with slope of 0.5
sem = "
  # from, to, lag, name, starting_value
  x -> y, 0, slope, 0.5
"
# Build DSEM with specified value for path coefficients
mod = dsem(
  sem = sem,
  tsdata = ts(data.frame(x=rep(0,20),y=rep(0,20))),
  control = dsem_control( run_model = FALSE )
)

# Show that total effect of X on Y from pulse experiment is 0.5 but does not propagate over time
pulse = total_effect(mod, n_lags = 2, type = "pulse")
subset( pulse, from=="x" & to=="y")


### EXAMPLE 2
# Define linear model with slope of 0.5 and autocorrelated response
sem = "
  x -> y, 0, slope, 0.5
  y -> y, 1, ar_y, 0.8
"
mod = dsem(
  sem = sem,
  tsdata = ts(data.frame(x=rep(0,20),y=rep(0,20))),
  control = dsem_control( run_model = FALSE )
)

# Show that total effect of X on Y from pulse experiment  is 0.5 with decay of 0.8 for each time
pulse = total_effect(mod, n_lags = 4, type = "pulse")
subset( pulse, from=="x" & to=="y")

# Show that total effect of X on Y from press experiment  asymptotes at 2.5
press = total_effect(mod, n_lags = 50, type = "press")
subset( press, from=="x" & to=="y")
```

---

tweedie *Tweedie distribution*

Description

Tweedie distribution

Usage

```
tweedie(link)
```

Arguments

link	Link function

---

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

∗ **data**
    bering_sea, 3
    hare_lynx, 15
    isle_royale, 15
    lake_washington, 16
    paramesium_didinium, 25
    pdo_departure_bay, 27
    sea_otter, 32

as_fitted_DAG, 3

bering_sea, 3
binomial(), *7*, *13*

cAIC, 4
classify_variables, 5
convert_equations, 5, *5*

dsem, *3, 4*, 6, *10*, *14*, *17–19*, *25, 26*, *28–31*, *33–36*, *39*
dsem_control, *7*, 10, *14*
dsemRTMB, 13

est_DAG, *3*

fixed, 14
fixed(), *7*, *13*

Gamma(), *7*, *13*
gaussian(), *7*, *13*
gaussian_fixed_sd, 14
gaussian_fixed_sd(), *7*, *13*
graph_from_data_frame, *28*

hare_lynx, 15

isle_royale, 15

lake_washington, 16
list_parameters, 16
logLik.dsem, 17
lognormal, 17
lognormal(), *7*, *13*
loo_residuals, 18

make_dfa, 19
make_dsem_ram, *6, 7*, *13*, 19, *20*, *30*
make_matrices, 23
make_msv, 24
MakeADFun, *9*, *12*

nlminb, *9*, *11, 12*

paramesium_didinium, 25
parse_path, 25
partition_variance, 26
pdo_departure_bay, 27
plot.dsem, 28
plot.igraph, *28*
poisson(), *7*, *13*
predict.dsem, 29
print.dsem, 29

read_model, 30
residuals.dsem, 31
rk4sys, 31

sdreport, *8, 9*, *11, 12*
sea_otter, 32
simulate.dsem, 32
stepwise_selection, *9*, 33
summary.dsem, 35

test_dsep, *9*, *34*, 36
TMBAIC, 38
total_effect, 39
ts, *7*, *13*
tweedie, 40
tweedie(), *7*, *13*

vcov.dsem, 41