Package ‘tinyVAST’

July 22, 2025

Type Package

Title Multivariate Spatio-Temporal Models using Structural Equations

Version 1.2.0

Date 2025-07-19

Description Fits a wide variety of multivariate spatio-temporal models with simultaneous and lagged interactions among variables (including vector autoregressive spatio-temporal ('VAST') dynamics) for areal, continuous, or network spatial domains. It includes time-variable, space-variable, and space-time-variable interactions using dynamic structural equation models ('DSEM') as expressive interface, and the 'mgcv' package to specify splines via the formula interface. See Thorson et al. (2024) <doi:10.48550/arXiv.2401.10193> for more details.

License GPL-3

Depends R (>= 4.1.0)

Imports corpcor, fmesher, igraph, Matrix (>= 1.3.0), methods, utils, mgcv, sem, sf, sfnetworks, TMB (>= 1.9.17), units, checkmate, abind, sdmTMB, dsem (>= 1.6.0), insight, cv

Suggests ggplot2, knitr, lattice, mvtnorm, pdp, rmarkdown, rnaturalearth, rnaturalearthdata, testthat, tweedie, viridisLite, visreg, plyr, DHARMa, glmmTMB, tibble,

LinkingTo RcppEigen, TMB

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

URL https://vast-lib.github.io/tinyVAST/

BugReports https://github.com/vast-lib/tinyVAST/issues

NeedsCompilation yes

Author James T. Thorson [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7415-1010>),
Sean C. Anderson [aut] (ORCID: <https://orcid.org/0000-0001-9563-1937>)

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

Repository CRAN

Date/Publication 2025-07-19 23:40:03 UTC

Contents

`add_predictions` *Add predictions to data-list*

Description

Given user-provided `newdata`, expand the object `tmb_data` to include predictions corresponding to those new observations

Usage

```
add_predictions(object, newdata, remove_origdata = FALSE)
```

Arguments

`object`	Output from `tinyVAST()`.
`newdata`	New data-frame of independent variables used to predict the response.
`remove_origdata`	Whether to remove original-data to allow faster evaluation. `remove_origdata=TRUE` eliminates information about the distribution for random effects, and cannot be combined with epsilon bias-correction. WARNING: feature is experimental and subject to change.

Value

the object `fit$tmb_inputs$tmb_data` representing data used during fitting, but with updated values for slots associated with predictions, where this updated object can be recompiled by TMB to provide predictions

---

`bering_sea` *Survey domain for the eastern and northern Bering Sea surveys*

Description

Shapefile defining the spatial domain for the eastern and northern Bering Sea bottom trawl surveys.

Usage

```
data(bering_sea)
```

bering_sea_pollock_ages

Survey catch-rates at age for Alaska pollock in the Eastern and Northern Bering Sea

Description

Data used to demonstrate and test model-based age expansion, using density= dependence corrected survey catch rates after first=stage expansion from the bottom trawl survey for ages 1-15, conducted by by the Alaska Fisheries Science Center, including annual surveys in the eastern Bering Sea 1982-2019 and 2021-2023, as well as the northern Bering Sea in 1982/85/88/91 and 2010/17/18/19/21/22/23.

Usage

```
data(bering_sea_pollock_ages)
```

bering_sea_pollock_vast

Estimated proportion-at-age for Alaska pollock using VAST

Description

Estimated proporrtion-at-age for Alaska pollock using the package VAST, for comparison with output using tinyVAST.

Usage

```
data(bering_sea_pollock_vast)
```

cAIC

Calculate conditional AIC

Description

Calculates the conditional Akaike Information criterion (cAIC).

Usage

```
cAIC(object)
```

Arguments

`object`	Output from `tinyVAST()`.

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 cross-validation.

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.

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

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

Value

cAIC value

References

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

Examples

```
data( red_snapper )
red_snapper = droplevels(subset(red_snapper, Data_type=="Biomass_KG"))

# Define mesh
mesh = fmesher::fm_mesh_2d( red_snapper[,c('Lon','Lat')],
                            cutoff = 1 )

# define formula with a catchability covariate for gear
formula = Response_variable ~ factor(Year) + offset(log(AreaSwept_km2))

# make variable column
red_snapper$var = "logdens"
# fit using tinyVAST
fit = tinyVAST( data = red_snapper,
                formula = formula,
                space_term = "logdens <-> logdens, sd_space",
                space_columns = c("Lon",'Lat'),
                spatial_domain = mesh,
                family = tweedie(link="log"),
                variable_column = "var",
                control = tinyVASTcontrol( getsd = FALSE,
                                           profile = "alpha_j" ) )

cAIC(fit) # conditional AIC
AIC(fit) # marginal AIC
```

classify_variables *Classify variables path*

Description

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

Usage

```
classify_variables(model)
```

Arguments

`model`	syntax for structural equation model

Details

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

Value

Tagged-list defining exogenous and endogenous variables

conditional_gmrf *Conditional simulation from a GMRF*

Description

Generates samples from a Gaussian Markov random field (GMRF) conditional upon fixed values for some elements.

Usage

```
conditional_gmrf(
  Q,
  observed_idx,
  x_obs,
  n_sims = 1,
  what = c("simulate", "predict")
)
```

Arguments

`Q`	precision for a zero-centered GMRF.
`observed_idx`	integer vector listing rows of `Q` corresponding to fixed measurements
`x_obs`	numeric vector with fixed values for indices `observed_idx`
`n_sims`	integer listing number of simulated values
`what`	Whether to simulate from the conditional GMRF, or predict the mean and precision

Value

A matrix with `n_sims` columns and a row for every row of `Q` not in `observed_idx`, with simulations for those rows

---

`condition_and_density` *Condition and density example*

Description

Data used to demonstrate and test a bivariate model for morphometric condition (i.e., residuals in a weight-at-length relationship) and density for fishes, using the same example as was provided as a wiki example for VAST. Data are from doi:10.3354/meps13213

Usage

```
data(condition_and_density)
```

`deviance_explained` *Calculate deviance explained*

Description

`deviance_explained` fits a null model, calculates the deviance relative to a saturated model for both the original and the null model, and uses these to calculate the proportion of deviance explained.

This implementation conditions upon the maximum likelihood estimate of fixed effects and the empirical Bayes ("plug-in") prediction of random effects. It can be described as "conditional deviance explained". A state-space model that estimates measurement error variance approaching zero (i.e., collapses to a process-error-only model) will have a conditional deviance explained that approaches 1.0

Usage

```
deviance_explained(x, null_formula, null_delta_formula = ~1)
```

Arguments

x
: output from \code{tinyVAST()}

null_formula
: formula for the null model. If missing, it uses null_formula = response ~ 1. For multivariate models, it might make sense to use null_formula = response ~ category

null_delta_formula
: formula for the null model for the delta component. If missing, it uses null_formula = response ~ 1. For multivariate models, it might make sense to use null_delta_formula = response ~ category

Value

the proportion of conditional deviance explained.

Families *Additional families*

Description

Additional families compatible with tinyVAST().

Usage

```
delta_lognormal(link1, link2 = "log", type = c("standard", "poisson-link"))

delta_gamma(link1, link2 = "log", type = c("standard", "poisson-link"))
```

Arguments

link1
: Link for first part of delta/hurdle model.

link2
: Link for second part of delta/hurdle model.

type
: Delta/hurdle family type. "standard" for a classic hurdle model. "poisson-link" for a Poisson-link delta model (Thorson 2018).

link
: Link.

Value

A list with elements common to standard R family objects including family, link, linkfun, and linkinv. Delta/hurdle model families also have elements delta (logical) and type (standard vs. Poisson-link).

References

Poisson-link delta families:

Thorson, J.T. 2018. Three problems with the conventional delta-model for biomass sampling data, and a computationally efficient alternative. Canadian Journal of Fisheries and Aquatic Sciences, 75(9), 1369-1382. doi:10.1139/cjfas20170266

Poisson-link delta families:

Thorson, J.T. 2018. Three problems with the conventional delta-model for biomass sampling data, and a computationally efficient alternative. Canadian Journal of Fisheries and Aquatic Sciences, 75(9), 1369-1382. doi:10.1139/cjfas20170266

`GetResponse.tinyVAST` *Get response*

Description

S3 generic from package cv, used for crossvalidation

Usage

```
## S3 method for class 'tinyVAST'
GetResponse(model, ...)
```

Arguments

`model`	output from `tinyVAST()`
`...`	not used

`get_data.tinyVAST` *Get data*

Description

S3 generic from package insight, used for crossvalidation

Usage

```
## S3 method for class 'tinyVAST'
get_data(x, ...)
```

Arguments

`x`	output from `tinyVAST()`
`...`	not used

integrate_output *Integration for target variable*

Description

Calculates an estimator for a derived quantity by summing across multiple predictions. This can be used to approximate an integral when estimating area-expanded abundance, abundance-weighting a covariate to calculate distribution shifts, and/or weighting one model variable by another.

Usage

```
integrate_output(
  object,
  newdata,
  area,
  type = rep(1, nrow(newdata)),
  weighting_index,
  covariate,
  getsd = TRUE,
  bias.correct = TRUE,
  apply.epsilon = FALSE,
  intern = FALSE
)
```

Arguments

object	Output from `tinyVAST()`.
newdata	New data-frame of independent variables used to predict the response, where a total value is calculated by combining across these individual predictions. If these locations are randomly drawn from a specified spatial domain, then `integrate_output` applies midpoint integration to approximate the total over that area. If locations are drawn sysmatically from a domain, then `integrate_output` is applying a midpoint approximation to the integral.
area	vector of values used for area-weighted expansion of estimated density surface for each row of `newdata` with length of `nrow(newdata)`.

type	Integer-vector indicating what type of expansion to apply to each row of `newdata`, with length of `nrow(newdata)`.
	`type=1` Area-weighting: weight predictor by argument `area`
	`type=2` Abundance-weighted covariate: weight `covariate` by proportion of total in each row of `newdata`
	`type=3` Abundance-weighted variable: weight predictor by proportion of total in a prior row of `newdata`. This option is used to weight a prediction for one category based on predicted proportional density of another category, e.g., to calculate abundance-weighted condition in a bivariate model.

type=4 Abundance-expanded variable: weight predictor by density in a prior row of newdata. This option is used to weight a prediction for one category based on predicted density of another category, e.g., to calculate abundance-expanded consumption in a bivariate model.

type=0 Exclude from weighting: give weight of zero for a given row of newdata. Including a row of newdata with type=0 is useful, e.g., when calculating abundance at that location, where the eventual index uses abundance as weighting term but without otherwise using the predicted density in calculating a total value.

weighting_index
: integer-vector used to indicate a previous row that is used to calculate a weighted average that is then applied to the given row of newdata. Only used for when type=3.

covariate
: numeric-vector used to provide a covariate that is used in expansion, e.g., to provide positional coordinates when calculating the abundance-weighted centroid with respect to that coordinate. Only used for when type=2.

getsd
: logical indicating whether to get the standard error, where getsd=FALSE is faster during initial exploration

bias.correct
: logical indicating if bias correction should be applied using standard methods in TMB::sdreport()

apply.epsilon
: Apply epsilon bias correction using a manual calculation rather than using the conventional method in TMB::sdreport? See details for more information.

intern
: Do Laplace approximation on C++ side? Passed to TMB::MakeADFun().

Details

Analysts will often want to calculate some value by combining the predicted response at multiple locations, and potentially from multiple variables in a multivariate analysis. This arises in a univariate model, e.g., when calculating the integral under a predicted density function, which is approximated using a midpoint or Monte Carlo approximation by calculating the linear predictors at each location newdata, applying the inverse-link-trainsformation, and calling this predicted response mu_g. Total abundance is then be approximated by multiplying mu_g by the area associated with each midpoint or Monte Carlo approximation point (supplied by argument area), and summing across these area-expanded values.

In more complicated cases, an analyst can then use covariate to calculate the weighted average of a covariate for each midpoint location. For example, if the covariate is positional coordinates or depth/elevation, then type=2 measures shifts in the average habitat utilization with respect to that covariate. Alternatively, an analyst fitting a multivariate model might weight one variable based on another using weighting_index, e.g., to calculate abundance-weighted average condition, or predator-expanded stomach contents.

In practice, spatial integration in a multivariate model requires two passes through the rows of newdata when calculating a total value. In the following, we write equations using C++ indexing conventions such that indexing starts with 0, to match the way that integrate_output expects indices to be supplied. Given inverse-link-transformed predictor μ_g, function argument type as $type_g$ function argument area as a_g, function argument covariate as x_g, function argument weighting_index as \eqn{ h_g } function argument weighting_index as \eqn{ h_g } the first pass calculates:

$$\nu_g = \mu_g a_g$$

where the total value from this first pass is calculated as:

$$\nu^* = \sum_{g=0}^{G-1} \nu_g$$

The second pass then applies a further weighting, which depends upon $type_g$, and potentially upon x_g and h_g.

If $type_g = 0$ then $\phi_g = 0$

If $type_g = 1$ then $\phi_g = \nu_g$

If $type_g = 2$ then $\phi_g = x_g \frac{\nu_g}{\nu^*}$

If $type_g = 3$ then $\phi_g = \frac{\nu_{h_g}}{\nu^*} \mu_g$

If $type_g = 4$ then $\phi_g = \nu_{h_g} \mu_g$

Finally, the total value from this second pass is calculated as:

$$\phi^* = \sum_{g=0}^{G-1} \phi_g$$

and ϕ^* is outputted by `integrate_output`, along with a standard error and potentially using the epsilon bias-correction estimator to correct for skewness and retransformation bias.

Standard bias-correction using `bias.correct=TRUE` can be slow, and in some cases it might be faster to do `apply.epsilon=TRUE` and `intern=TRUE`. However, that option is somewhat experimental, and a user might want to confirm that the two options give identical results. Similarly, using `bias.correct=TRUE` will still calculate the standard-error, whereas using `apply.epsilon=TRUE` and `intern=TRUE` will not.

Value

A vector containing the plug-in estimate, standard error, the epsilon bias-corrected estimate if available, and the standard error for the bias-corrected estimator. Depending upon settings, one or more of these will be `NA` values, and the function can be repeatedly called to get multiple estimators and/or statistics.

`logLik.tinyVAST` *Extract the (marginal) log-likelihood of a tinyVAST model*

Description

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

Usage

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

Arguments

object	output from tinyVAST
...	not used

Value

object of class logLik with attributes

val	log-likelihood
df	number of parameters

---

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

Arguments

dsem	dynamic structural equation model structure, passed to either specifyModel or specifyEquations and then parsed to control the set of path coefficients and variance-covariance parameters
times	A character vector listing the set of times in order
variables	A character vector listing the set of variables
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. For confirmatory factor analysis models specified via cfa, covs defaults to all of the factors in the

model, thus specifying all variances and covariances among these factors. *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.

`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 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:

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

Usefully, it is also easy to compute the inverse-covariance (precision) matrix $\mathbf{Q} = \mathbf{V}^{-1}$:

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

Example: univariate and first-order autoregressive model

This simultaneous autoregressive (SAR) process across variables and times allows the user to specify both simultaneous 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 `dsem = X -> X, 1, rho; 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 $\mathbf{P}$:

```
\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 $\mathbf{\Gamma}$:

```
\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^1 & \rho^2 \
    \rho^2 & \rho^1 & 1 & \rho^1 \
    \rho^3 & \rho^2 & \rho^1 & 1\
    \end{bmatrix} }
```

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
dsem = "
  X -> X, 1, rho
  X <-> X, 0, sigma
"
make_dsem_ram( dsem=dsem, variables="X", times=1:4 )

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

# Bivariate VAR
dsem = "
  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( dsem=dsem, 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)
dsem = "
  factor -> X, 0, loadings1
  factor -> Y, 0, loadings2
  factor -> factor, 1, NA, 1
  X <-> X, 0, NA, 0           # No additional variance
  Y <-> Y, 0, NA, 0           # No additional variance
"
make_dsem_ram( dsem=dsem, variables=c("X","Y","factor"), times=1:4 )

# ARIMA(1,1,0)
dsem = "
  factor -> factor, 1, rho1 # AR1 component
  X -> X, 1, NA, 1          # Integrated component
  factor -> X, 0, NA, 1
```

```
  X <-> X, 0, NA, 0         # No additional variance
"
make_dsem_ram( dsem=dsem, variables=c("X","factor"), times=1:4 )

# ARIMA(0,0,1)
dsem = "
  factor -> X, 0, NA, 1
  factor -> X, 1, rho1     # MA1 component
  X <-> X, 0, NA, 0        # No additional variance
"
make_dsem_ram( dsem=dsem, variables=c("X","factor"), times=1:4 )
```

---

make_eof_ram   *Make a RAM (Reticular Action Model)*

Description

make_eof_ram converts SEM arrow notation to ram describing SEM parameters

Usage

```
make_eof_ram(
  times,
  variables,
  n_eof,
  remove_na = TRUE,
  standard_deviations = "unequal"
)
```

Arguments

times	A character vector listing the set of times in order
variables	A character vector listing the set of variables
n_eof	Number of EOF modes of variability to estimate
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
standard_deviations	One of "equal", "unequal", or a numeric vector indicating fixed values.

Value

A reticular action module (RAM) describing dependencies

Examples

```
# Two EOFs for two variables
make_eof_ram( times = 2010:2020, variables = c("pollock","cod"), n_eof=2 )
```

`make_sem_ram` *Make a RAM (Reticular Action Model) from a SEM (structural equation model)*

Description

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

Usage

```
make_sem_ram(sem, variables, quiet = FALSE, covs = variables)
```

Arguments

`sem`	structural equation model structure, passed to either `specifyModel` or `specifyEquations` and then parsed to control the set of path coefficients and variance-covariance parameters
`variables`	A character vector listing the set of variables
`quiet`	if `FALSE`, the default, then the number of input lines is reported and a message is printed suggesting that `specifyEquations` or `cfa` be used.
`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. For confirmatory factor analysis models specified via `cfa`, `covs` defaults to all of the factors in the model, thus specifying all variances and covariances among these factors. *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.

Value

An S3-class `"sem_ram"` containing:

`model` Output from `specifyEquations` or `specifyModel` that defines paths and parameters

`ram` reticular action module (RAM) describing dependencies

parse_path — *Parse path*

Description

`parse_path` is copied from `sem::parse.path`

Usage

```
parse_path(path)
```

Arguments

`path`	character string indicating a one-headed or two-headed path in a structural equation model

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

predict.tinyVAST — *Predict using vector autoregressive spatio-temporal model*

Description

Predicts values given new covariates using a **tinyVAST** model

Usage

```
## S3 method for class 'tinyVAST'
predict(
  object,
  newdata,
  remove_origdata = FALSE,
 what = c("mu_g", "p_g", "palpha_g", "pgamma_g", "pepsilon_g", "pomega_g", "pdelta_g",
  "pxi_g", "p2_g", "palpha2_g", "pgamma2_g", "pepsilon2_g", "pomega2_g", "pdelta2_g",
    "pxi2_g"),
  se.fit = FALSE,
  bias.correct = FALSE,
  ...
)
```

Arguments

object	Output from tinyVAST().
newdata	New data-frame of independent variables used to predict the response.
remove_origdata	Whether to eliminate the original data from the TMB object, thereby speeding up the TMB object construction. However, this also eliminates information about random-effect variance, and is not appropriate when requesting predictive standard errors or epsilon bias-correction.
what	What REPORTed object to output, where mu_g is the inverse-linked transformed predictor including both linear components, p_g is the first linear predictor, palpha_g is the first predictor from fixed covariates in formula, pgamma_g is the first predictor from random covariates in formula (e.g., splines), pomega_g is the first predictor from spatial variation, pepsilon_g is the first predictor from spatio-temporal variation, pxi_g is the first predictor from spatially varying coefficients, p2_g is the second linear predictor, palpha2_g is the second predictor from fixed covariates in formula, pgamma2_g is the second predictor from random covariates in formula (e.g., splines), pomega2_g is the second predictor from spatial variation, pepsilon2_g is the second predictor from spatio-temporal variation, and pxi2_g is the second predictor from spatially varying coefficients.
se.fit	Calculate standard errors?
bias.correct	whether to epsilon bias-correct the predicted value
...	Not used.

Value

Either a vector with the prediction for each row of newdata, or a named list with the prediction and standard error (when se.fit = TRUE).

---

print.tinyVAST *print summary of tinyVAST model*

Description

print summary of tinyVAST model

Usage

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

Arguments

x	output from tinyVAST
...	not used

Value

invisibly returns a named list of key model outputs and summary statements

---

`project` *Project tinyVAST to future times (EXPERIMENTAL)*

Description

Projects a fitted model forward in time.

Usage

```
project(
  object,
  extra_times,
  newdata,
  what = "mu_g",
  future_var = TRUE,
  past_var = FALSE,
  parm_var = FALSE
)
```

Arguments

`object`	fitted model from `tinyVAST(.)`
`extra_times`	a vector of extra times, matching values in `newdata`
`newdata`	data frame including new values for `time_variable`
`what`	What REPORTed object to output, where `mu_g` is the inverse-linked transformed predictor including both linear components, `p_g` is the first linear predictor, `palpha_g` is the first predictor from fixed covariates in `formula`, `pgamma_g` is the first predictor from random covariates in `formula` (e.g., splines), `pomega_g` is the first predictor from spatial variation, `pepsilon_g` is the first predictor from spatio-temporal variation, `pxi_g` is the first predictor from spatially varying coefficients, `p2_g` is the second linear predictor, `palpha2_g` is the second predictor from fixed covariates in `formula`, `pgamma2_g` is the second predictor from random covariates in `formula` (e.g., splines), `pomega2_g` is the second predictor from spatial variation, `pepsilon2_g` is the second predictor from spatio-temporal variation, and `pxi2_g` is the second predictor from spatially varying coefficients.
`future_var`	logical indicating whether to simulate future process errors from GMRFs, or just compute the predictive mean
`past_var`	logical indicating whether to re-simulate past process errors from predictive distribution of random effects, thus changing the boundary condition of the forecast
`parm_var`	logical indicating whether to re-sample fixed effects from their predictive distribution, thus changing the GMRF for future process errors

Value

A vector of values corresponding to rows in newdata

Examples

```
# Convert to long-form
set.seed(123)
n_obs = 100
rho = 0.9
sigma_x = 0.2
sigma_y = 0.1
x = rnorm(n_obs, mean=0, sd = sigma_x)
for(i in 2:length(x)) x[i] = rho * x[i-1] + x[i]
y = x + rnorm( length(x), mean = 0, sd = sigma_y )
data = data.frame( "val" = y, "var" = "y", "time" = seq_along(y) )

# Define AR2 time_term
time_term = "
  y -> y, 1, rho1
  y -> y, 2, rho2
  y <-> y, 0, sd
"

# fit model
mytiny = tinyVAST(
  time_term = time_term,
  data = data,
  times = unique(data$t),
  variables = "y",
  formula = val ~ 1,
  control = tinyVASTcontrol( getJointPrecision = TRUE )
)

# Deterministic projection
extra_times = length(x) + 1:100
n_sims = 10
newdata = data.frame( "time" = c(seq_along(x),extra_times), "var" = "y" )
Y = project(
  mytiny,
  newdata = newdata,
  extra_times = extra_times,
  future_var = FALSE
)
plot( x = seq_along(Y),
      y = Y,
      type = "l", lty = "solid", col = "black" )

# Stochastic projection with future process errors
## Not run:
extra_times = length(x) + 1:100
n_sims = 10
newdata = data.frame( "time" = c(seq_along(x),extra_times), "var" = "y" )
```

```
Y = NULL
for(i in seq_len(n_sims) ){
  tmp = project(
    mytiny,
    newdata = newdata,
    extra_times = extra_times,
    future_var = TRUE,
    past_var = TRUE,
    parm_var = TRUE
  )
  Y = cbind(Y, tmp)
}
matplot( x = row(Y),
         y = Y,
         type = "l", lty = "solid", col = "black" )

## End(Not run)
```

red_snapper *Presence/absence, count, and biomass data for red snapper*

Description

Data used to demonstrate and test analysis using multiple data types

Usage

```
data(red_snapper)
```

red_snapper_shapefile *Shapefile for red snapper analysis*

Description

Spatial extent used for red snapper analysis, derived from Chap-7 of doi:10.1201/9781003410294

Usage

```
data(red_snapper_shapefile)
```

`reload_model` *Reload a previously fitted model*

Description

`reload_model` allows a user to save a fitted model, reload it in a new R terminal, and then relink the DLLs so that it functions as expected.

Usage

```
reload_model(x, check_gradient = TRUE)
```

Arguments

`x`	Output from `tinyVAST`, potentially with DLLs not linked
`check_gradient`	Whether to check the gradients of the reloaded model

Value

Output from `tinyVAST` with DLLs relinked

`residuals.tinyVAST` *Calculate deviance or response residuals for tinyVAST*

Description

Calculate residuals

Usage

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

Arguments

`object`	Output from `tinyVAST()`
`type`	which type of residuals to compute (only option is `"deviance"` or `"response"` for now)
`...`	Note used

Value

a vector residuals, associated with each row of `data` supplied during fitting

rmvnorm_prec — *Multivariate Normal Random Deviates using Sparse Precision*

Description

This function provides a random number generator for the multivariate normal distribution with mean equal to `mu` and sparse precision matrix `prec`.

Usage

```
rmvnorm_prec(prec, n = 1, mu = rep(0, nrow(prec)))
```

Arguments

`prec`	sparse precision (inverse-covariance) matrix.
`n`	number of observations.
`mu`	mean vector.

Value

a matrix with dimension `length(mu)` by `n`, containing realized draws from the specified mean and precision

rotate_pca — *Rotate factors to match Principal-Components Analysis*

Description

Rotate lower-triangle loadings matrix to order factors from largest to smallest variance.

Usage

```
rotate_pca(
  L_tf,
  x_sf = matrix(0, nrow = 0, ncol = ncol(L_tf)),
  order = c("none", "increasing", "decreasing")
)
```

Arguments

`L_tf`	Loadings matrix with dimension $T \times F$.
`x_sf`	Spatial response with dimensions $S \times F$.
`order`	Options for resolving label-switching via reflecting each factor to achieve a given order across dimension T.

Value

List containing the rotated loadings `L_tf`, the inverse-rotated response matrix `x_sf`, and the rotation `H`

`salmon_returns` *North Pacific salmon returns*

Description

Data used to demonstrate and test multivariate second-order autoregressive models using a simultaneous autoregressive (SAR) process across regions. Data are from doi:10.1002/mcf2.10023

Usage

```
data(salmon_returns)
```

`sample_variable` *Sample from predictive distribution of a variable*

Description

`sample_variable` samples from the joint distribution of random and fixed effects to approximate the predictive distribution for a variable

Using `sample_fixed=TRUE` (the default) in `sample_variable` propagates variance in both fixed and random effects, while using `sample_fixed=FALSE` does not. Sampling fixed effects will sometimes cause numerical under- or overflow (i.e., output values of `NA`) in cases when variance parameters are estimated imprecisely. In these cases, the multivariate normal approximation being used is a poor representation of the tail probabilities, and results in some samples with implausibly high (or negative) variances, such that the associated random effects then have implausibly high magnitude.

Usage

```
sample_variable(
  object,
  newdata = NULL,
  variable_name = "mu_i",
  n_samples = 100,
  sample_fixed = TRUE,
  seed = 123456
)
```

Arguments

object	output from \code{tinyVAST()}
newdata	data frame of new data, used to sample model components for predictions e.g., mu_g
variable_name	name of variable available in report using Obj$report() or parameters using ObjenvparList()
n_samples	number of samples from the joint predictive distribution for fixed and random effects. Default is 100, which is slow.
sample_fixed	whether to sample fixed and random effects, sample_fixed=TRUE as by default, or just sample random effects, sample_fixed=FALSE
seed	integer used to set random-number seed when sampling variables, as passed to set.seed(.)

Value

A matrix with a row for each data supplied during fitting, and n_samples columns, where each column in a vector of samples for a requested quantity given sampled uncertainty in fixed and/or random effects

Examples

```
set.seed(101)
x = runif(n = 100, min = 0, max = 2*pi)
y = 1 + sin(x) + 0.1 * rnorm(100)

# Do fit with getJointPrecision=TRUE
fit = tinyVAST( formula = y ~ s(x),
                data = data.frame(x=x,y=y) )

# samples from distribution for the mean
# excluding fixed effects due to CRAN checks
samples = sample_variable(fit, sample_fixed = FALSE)
```

---

sea_ice *Arctic September sea ice concentrations*

Description

Data used to demonstrate and test empirical orthogonal function generalized linear latent variable model (EOF-GLLVM)

Usage

```
data(sea_ice)
```

sfnetwork_evaluator *Construct projection matrix for stream network*

Description

Make sparse matrix to project from stream-network nodes to user-supplied points

Usage

```
sfnetwork_evaluator(stream, loc, tolerance = 0.01)
```

Arguments

stream	**sfnetworks** object representing stream network
loc	**sf** object representing points to which are being projected
tolerance	error-check tolerance

Value

the sparse interpolation matrix, with rows for each row of `data` supplied during fitting and columns for each spatial random effect.

sfnetwork_mesh *Make mesh for stream network*

Description

make an object representing spatial information required to specify a stream-network spatial domain, similar in usage to `link[fmesher]{fm_mesh_2d}` for a 2-dimensional continuous domain

Usage

```
sfnetwork_mesh(stream)
```

Arguments

stream	**sfnetworks** object representing stream network

Value

An object (list) of class `sfnetwork_mesh`. Elements include:

N The number of random effects used to represent the network

table a table containing a description of parent nodes (from), childen nodes (to), and the distance separating them

stream copy of the stream network object passed as argument `simulate.tinyVAST` *Simulate new data from a fitted model*

Description

`simulate.tinyVAST` is an S3 method for producing a matrix of simulations from a fitted model. It can be used with the **DHARMa** package among other uses. Code is modified from the version in sdmTMB

Usage

```
## S3 method for class 'tinyVAST'
simulate(
  object,
  nsim = 1L,
  seed = sample.int(1e+06, 1L),
  type = c("mle-eb", "mle-mvn"),
  ...
)
```

Arguments

`object`	output from `tinyVAST()`
`nsim`	how many simulations to do
`seed`	random seed
`type`	How parameters should be treated. `"mle-eb"`: fixed effects are at their maximum likelihood (MLE) estimates and random effects are at their empirical Bayes (EB) estimates. `"mle-mvn"`: fixed effects are at their MLEs but random effects are taken from a single approximate sample. This latter option is a suggested approach if these simulations will be used for goodness of fit testing (e.g., with the DHARMa package).
`...`	not used

Value

A matrix with row for each row of `data` in the fitted model and `nsim` columns, containing new samples from the fitted model.

Examples

```
set.seed(101)
x = seq(0, 2*pi, length=100)
y = sin(x) + 0.1*rnorm(length(x))
fit = tinyVAST( data=data.frame(x=x,y=y), formula = y ~ s(x) )
sims = simulate(fit, nsim=100, type="mle-mvn")

if(requireNamespace("DHARMa")){
```

```
  # simulate new data conditional on fixed effects
  # and sampling random effects from their predictive distribution
  y_iz = simulate(fit, nsim=500, type="mle-mvn")

  # Visualize using DHARMa
  res = DHARMa::createDHARMa( simulatedResponse = y_iz,
                      observedResponse = y,
                      fittedPredictedResponse = fitted(fit) )
  plot(res)
}
```

`simulate_sfnetwork` *Simulate GMRF for stream network*

Description

Simulate values from a GMRF using a tail-down (flow-unconnected) exponential model on a stream network

Usage

```
simulate_sfnetwork(sfnetwork_mesh, theta, n = 1, what = c("samples", "Q"))
```

Arguments

`sfnetwork_mesh`	Output from `sfnetwork_mesh`
`theta`	Decorrelation rate
`n`	number of simulated GMRFs
`what`	Whether to return the simulated GMRF or its precision matrix

Value

a matrix of simulated values for a Gaussian Markov random field arising from a stream-network spatial domain, with row for each spatial random effect and `n` columns, using the sparse precision matrix defined in Charsley et al. (2023)

References

Charsley, A. R., Gruss, A., Thorson, J. T., Rudd, M. B., Crow, S. K., David, B., Williams, E. K., & Hoyle, S. D. (2023). Catchment-scale stream network spatio-temporal models, applied to the freshwater stages of a diadromous fish species, longfin eel (Anguilla dieffenbachii). Fisheries Research, 259, 106583. doi:10.1016/j.fishres.2022.106583

summary.tinyVAST *summarize tinyVAST*

Description

summarize parameters from a fitted tinyVAST

Usage

```
## S3 method for class 'tinyVAST'
summary(
  object,
  what = c("space_term", "time_term", "spacetime_term", "fixed"),
  predictor = c("one", "two"),
  ...
)
```

Arguments

`object`	Output from `tinyVAST()`
`what`	What component to summarize, whether `space_term`, `spacetime_term`, or `fixed` for the fixed effects included in the GAM formula
`predictor`	whether to get the 1st or 2nd linear predictor (the latter is only applicable in delta models)
`...`	Not used

Details

`tinyVAST` includes three components:

Space-variable interaction a separable Gaussian Markov random field (GMRF) constructed from a structural equation model (SEM) and a spatial variable

Space-variable-time interaction a separable GMRF constructed from a a dynamic SEM (a non-separable time-variable interaction) and a spatial variable

Additive variation a generalized additive model (GAM), representing exogenous covariates

Each of these are summarized and interpreted differently, and `summary.tinyVAST` facilitates this.

Regarding the DSEM componennt, tinyVAST includes an "arrow and lag" notation, which specifies the set of path coefficients and exogenous variance parameters to be estimated. Function `tinyVAST` then estimates the maximum likelihood value for those coefficients and parameters by maximizing the log-marginal likelihood.

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

A data-frame containing the estimate (and standard errors, two-sided Wald-test z-value, and associated p-value if the standard errors are available) for model parameters, including the fixed-effects specified via `formula`, or the path coefficients for the spatial SEM specified via `space_term`, the dynamic SEM specified via `time_term`, or the spatial dynamic SEM specified via `spacetime_term`

`term_covariance` *Extract covariance*

Description

Extract the covariance resulting from a specified path structure and estimated parameters for a SEM or DSEM term in tinyVAST

Usage

```
term_covariance(
  object,
  what = c("space_term", "time_term", "spacetime_term"),
  pred = c("one", "two"),
  n_times = NULL
)
```

Arguments

`object`	Output from `tinyVAST`
`what`	Which SEM or DSEM term to extract
`pred`	Extract the term `what` for which linear predictor
`n_times`	The number of times to include when calculating covariance for a DSEM component, i.e., `time_term` or `spacetime_term`. If missing, the default is to use the one more than the maximum specified lag (e.g., `n_times=2` by default when the maximum `lag=1`)

Details

tinyVAST constructs the covariance from specified path structure and estimated parameters

Value

The covariance matrix among variables

Examples

```
# Extract covariance for spatial factor analysis (too slow for CRAN)

# Simulate settings
set.seed(101)
theta_xy = 0.4
n_x = n_y = 10
n_c = 3           # Number of species
n_f = 1           # Number of factors
rho = 0.8
resid_sd = 0.5

# Simulate GMRFs
R_s = exp(-theta_xy * abs(outer(1:n_x, 1:n_y, FUN="-")) )
R_ss = kronecker(X=R_s, Y=R_s)
delta_fs = mvtnorm::rmvnorm(n_c, sigma=R_ss )

# Simulate loadings for two factors
L_cf = matrix( rnorm(n_c^2), nrow=n_c )
L_cf[,seq(from=n_f+1, to=n_c)] = 0
L_cf = L_cf + resid_sd * diag(n_c)

# Simulate correlated densities
d_cs = L_cf %*% delta_fs

# Shape into longform data-frame and add error
Data = data.frame( expand.grid(species=1:n_c, x=1:n_x, y=1:n_y),
                   "var"="logn", "z"=exp(as.vector(d_cs)) )
Data$n = rnorm( n=nrow(Data), mean=Data$z, sd=1 )

# make mesh
mesh = fmesher::fm_mesh_2d( Data[,c('x','y')] )

# Specify factor model with two factors and additional independent variance with shared SD
sem = "
  # Loadings matrix
  f1 -> 1, l1
  f1 -> 2, l2
  f1 -> 3, l3

  # Factor variance = 1
  f1 <-> f1, NA, 1

  # Shared residual variance
  1 <-> 1, sd, 1
  2 <-> 2, sd, 1
  3 <-> 3, sd, 1
"

# fit model
out = tinyVAST( space_term = sem,
           data = Data,
```

```
            formula = n ~ 0 + factor(species),
            spatial_domain = mesh,
            variables = c( "f1", "f2", 1:n_c ),
            space_columns = c("x","y"),
            variable_column = "species",
            time_column = "time",
            distribution_column = "dist" )

# Extract covariance among species and factors, where
# estimated covariance is obtained by ignoring factors
V = term_covariance( out, what = "space_term", pred = "one" )
```

`tinyVAST` *Fit vector autoregressive spatio-temporal model*

Description

Fits a vector autoregressive spatio-temporal (VAST) model using a minimal feature-set and a widely used interface.

Usage

```
tinyVAST(
  formula,
  data,
  time_term = NULL,
  space_term = NULL,
  spacetime_term = NULL,
  family = gaussian(),
  space_columns = c("x", "y"),
  spatial_domain = NULL,
  time_column = "time",
  times = NULL,
  variable_column = "var",
  variables = NULL,
  distribution_column = "dist",
  delta_options = list(formula = ~1),
  spatial_varying = NULL,
  weights = NULL,
  control = tinyVASTcontrol()
)
```

Arguments

`formula` Formula with response on left-hand-side and predictors on right-hand-side, parsed by `mgcv` and hence allowing `s(.)` for splines or `offset(.)` for an offset.

data
: Data-frame of predictor, response, and offset variables. Also includes variables that specify space, time, variables, and the distribution for samples, as identified by arguments variable_column, time_column, space_columns, and distribution_column.

time_term
: Specification for time-series structural equation model structure for constructing a time-variable interaction that defines a time-varying intercept for each variable (i.e., applies uniformly across space). time_term=NULL disables the space-variable interaction; see make_dsem_ram() for notation.

space_term
: Specification for structural equation model structure for constructing a space-variable interaction. space_term=NULL disables the space-variable interaction; see make_sem_ram() for notation.

spacetime_term
: Specification for time-series structural equation model structure including lagged or simultaneous effects for constructing a time-variable interaction, which is then combined in a separable process with the spatial correlation to form a space-time-variable interaction (i.e., the interaction occurs locally at each site). spacetime_term=NULL disables the space-variable interaction; see make_dsem_ram() or make_eof_ram().

family
: A function returning a class family, including gaussian(), lognormal(), tweedie(), binomial(), Gamma(), poisson(), nbinom1(), or nbinom2(). Alternatively, can be a named list of these functions, with names that match levels of data$distribution_column to allow different families by row of data. Delta model families are possible, and see Families for delta-model options. For binomial family options, see 'Binomial families' in the Details section below.

space_columns
: A string or character vector that indicates the column(s) of data indicating the location of each sample. When spatial_domain is an igraph object, space_columns is a string with with levels matching the names of vertices of that object. When spatial_domain is an fmesher or sfnetwork object, space_columns is a character vector indicating columns of data with coordinates for each sample.

spatial_domain
: Object that represents spatial relationships, either using fmesher::fm_mesh_2d() to apply the SPDE method, igraph::make_empty_graph() for independent time-series, igraph::make_graph() to apply a simultaneous autoregressive (SAR) process to a user-supplied graph, sfnetwork_mesh() for stream networks, or class sfc_GEOMETRY e.g constructed using sf::st_make_grid to apply a SAR to an areal model with adjacency based on the geometry of the object, or NULL to specify a single site. If using igraph then the graph must have vertex names V(graph)$name that match levels of data[,'space_columns']

time_column
: A character string indicating the column of data listing the time-interval for each sample, from the set of times in argument times.

times
: A integer vector listing the set of times in order. If times=NULL, then it is filled in as the vector of integers from the minimum to maximum value of data$time. Alternatively, it could be the minimum value of data$time through future years, such that the model can forecast those future years.

variable_column
: A character string indicating the column of data listing the variable for each sample, from the set of times in argument variables.

variables
: A character vector listing the set of variables. if variables=NULL, then it is filled in as the unique values from data$variable_columns.

`distribution_column`
: A character string indicating the column of `data` listing the distribution for each sample, from the set of names in argument `family`. if `variables=NULL`, then it is filled in as the unique values from `data$variables`.

`delta_options`
: a named list with slots for `formula`, `space_term`, and `spacetime_term`. These specify options for the second linear predictor of a delta model, and are only used (or estimable) when a `delta family` is used for some samples.

`spatial_varying`
: a formula specifying spatially varying coefficients.

`weights`
: A numeric vector representing optional likelihood weights for the data likelihood. Weights do not have to sum to one and are not internally modified. Thee weights argument needs to be a vector and not a name of the variable in the data frame.

`control`
: Output from `tinyVASTcontrol()`, used to define user settings.

Details

`tinyVAST` includes several basic inputs that specify the model structure:

- `formula` specifies covariates and splines in a Generalized Additive Model;
- `time_term` specifies interactions among variables and over time that are constant across space, constructing the time-variable interaction.
- `space_term` specifies interactions among variables and over time that occur based on the variable values at each location, constructing the space-variable interaction.
- `spacetime_term` specifies interactions among variables and over time, constructing the space-time-variable interaction.

These inputs require defining the *domain* of the model. This includes:

- `spatial_domain` specifies spatial domain, with determines spatial correlations
- `times` specifies the temporal domain, i.e., sequence of time-steps
- `variables` specifies the set of variables, i.e., the variables that will be modeled

The default `spacetime_term=NULL` and `space_term=NULL` turns off all multivariate and temporal indexing, such that `spatial_domain` is then ignored, and the model collapses to a generalized additive model using `gam`. To specify a univariate spatial model, the user must specify `spatial_domain` and either `space_term=""` or `spacetime_term=""`, where the latter two are then parsed to include a single exogenous variance for the single variable

Model type

Generalized additive model
Dynamic structural equation model (including vector autoregressive, dynamic factor analysis, ARIMA, and structural equation
Univariate spatio-temporal model, or multiple independence spatio-temporal variables
Multivariate spatial model including interactions
Vector autoregressive spatio-temporal model (i.e., lag-1 interactions among variables)

Model building notes

- `binomial familes`: A binomial family can be specified in only one way: the response is the observed proportion (proportion = successes / trials), and the 'weights' argument is used to specify the Binomial size (trials, N) parameter (`proportion ~ ..., weights = N`).
- `factor models`: If a factor model is desired, the factor(s) must be named and included in the `variables`. The factor is then modeled for `space_term`, `time_term`, and `spacetime_term` and it's variance must be fixed a priori for any term where it is not being used.

Value

An object (list) of class `tinyVAST`. Elements include:

data Data-frame supplied during model fitting

spatial_domain the spatial domain supplied during fitting

formula the formula specified during model fitting

obj The TMB object from `MakeADFun`

opt The output from `nlminb`

opt The report from `obj$report()`

sdrep The output from `sdreport`

tmb_inputs The list of inputs passed to `MakeADFun`

call A record of the function call

run_time Total time to run model

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

deviance_explained output from `deviance_explained`

See Also

Details section of `make_dsem_ram()` for a summary of the math involved with constructing the DSEM, and doi:10.1111/2041210X.14289 for more background on math and inference

doi:10.48550/arXiv.2401.10193 for more details on how GAM, SEM, and DSEM components are combined from a statistical and software-user perspective

`summary.tinyVAST()` to visualize parameter estimates related to SEM and DSEM model components

Examples

```
# Simulate a seperable two-dimensional AR1 spatial process
n_x = n_y = 25
n_w = 10
R_xx = exp(-0.4 * abs(outer(1:n_x, 1:n_x, FUN="-")) )
R_yy = exp(-0.4 * abs(outer(1:n_y, 1:n_y, FUN="-")) )
z = mvtnorm::rmvnorm(1, sigma=kronecker(R_xx,R_yy) )

# Simulate nuissance parameter z from oscillatory (day-night) process
w = sample(1:n_w, replace=TRUE, size=length(z))
Data = data.frame( expand.grid(x=1:n_x, y=1:n_y), w=w, z=as.vector(z) + cos(w/n_w*2*pi))
Data$n = Data$z + rnorm(nrow(Data), sd=1)
```

```
# Add columns for multivariate and/or temporal dimensions
Data$var = "n"

# make SPDE mesh for spatial term
mesh = fmesher::fm_mesh_2d( Data[,c('x','y')], n=100 )

# fit model with cyclic confounder as GAM term
out = tinyVAST( data = Data,
                formula = n ~ s(w),
                spatial_domain = mesh,
                space_term = "n <-> n, sd_n" )

# Run crossvalidation (too slow for CRAN)

CV = cv::cv( out, k = 4 )
CV
```

---

tinyVASTcontrol *Control parameters for tinyVAST*

Description

Control parameters for tinyVAST

Usage

```
tinyVASTcontrol(
  nlminb_loops = 1,
  newton_loops = 0,
  eval.max = 1000,
  iter.max = 1000,
  getsd = TRUE,
  silent = getOption("tinyVAST.silent", TRUE),
  trace = getOption("tinyVAST.trace", 0),
  verbose = getOption("tinyVAST.verbose", FALSE),
  profile = c(),
  tmb_par = NULL,
  tmb_map = NULL,
  gmrf_parameterization = c("separable", "projection"),
  reml = FALSE,
  getJointPrecision = FALSE,
  calculate_deviance_explained = TRUE,
  run_model = TRUE,
  suppress_nlminb_warnings = TRUE,
  suppress_user_warnings = FALSE,
```

```
  get_rsr = FALSE,
  extra_reporting = FALSE,
  use_anisotropy = FALSE,
  sar_adjacency = "queen"
)
```

Arguments

nlminb_loops
: Integer number of times to call stats::nlminb().

newton_loops
: Integer number of Newton steps to do after running stats::nlminb().

eval.max
: Maximum number of evaluations of the objective function allowed. Passed to control in stats::nlminb().

iter.max
: Maximum number of iterations allowed. Passed to control in stats::nlminb().

getsd
: Boolean indicating whether to call TMB::sdreport()

silent
: Disable terminal output for inner optimizer?

trace
: Parameter values are printed every trace iteration for the outer optimizer. Passed to control in stats::nlminb().

verbose
: Output additional messages about model steps during fitting?

profile
: Parameters to profile out of the likelihood (this subset will be appended to random with Laplace approximation disabled).

tmb_par
: list of parameters for starting values, with shape identical to tinyVAST(...)$internal$parlist

tmb_map
: input passed to TMB::MakeADFun as argument map, over-writing the version tinyVAST(...)tmb_inputstmb_map and allowing detailed control over estimated parameters (advanced feature)

gmrf_parameterization
: Parameterization to use for the Gaussian Markov random field, where the default separable constructs a full-rank and separable precision matrix, 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 allows for rank-deficient covariance.

reml
: Logical: use REML (restricted maximum likelihood) estimation rather than maximum likelihood? Internally, this adds the fixed effects to the list of random effects to integrate over.

getJointPrecision
: whether to get the joint precision matrix. Passed to sdreport.

calculate_deviance_explained
: whether to calculate proportion of deviance explained. See deviance_explained()

run_model
: whether to run the model of export TMB objects prior to compilation (useful for debugging)

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

suppress_user_warnings
: whether to suppress warnings from package author regarding dangerous or non-standard options

get_rsr	Experimental option, whether to report restricted spatial regression (RSR) adjusted estimator for covariate responses
extra_reporting	Whether to report a much larger set of quantities via objenvreport()
use_anisotropy	Whether to estimate two parameters representing geometric anisotropy
sar_adjacency	Whether to use queen or rook adjacency when defining a Simultaneous Autoregressive spatial precision from a sfc_GEOMETRY (default is queen)

Value

An object (list) of class tinyVASTcontrol, containing either default or updated values supplied by the user for model settings

---

vcov.tinyVAST *Extract Variance-Covariance Matrix*

Description

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

Usage

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

Arguments

object	output from tinyVAST()
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