Package 'spmodel'

September 10, 2026

Title Spatial Statistical Modeling and Prediction

Version 0.14.0

Description Fit, summarize, and predict for a variety of spatial statistical models applied to point-referenced and areal (lattice) data. Parameters are estimated using various methods. Additional modeling features include anisotropy, non-spatial random effects, partition factors, big data approaches, and more. Model-fit statistics are used to summarize, visualize, and compare models. Predictions at unobserved locations are readily obtainable. For additional details, see Dumelle et al. (2023) <doi:10.1371/journal.pone.0282524>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Imports graphics, generics, Matrix, sf, stats, tibble, parallel

Suggests rmarkdown, knitr, testthat (>= 3.0.0), ggplot2, ranger, statmod, pROC, emmeans (>= 1.4), estimability, numDeriv, GPvecchia, randomForest, xgboost, spsurvey

VignetteBuilder knitr

Config/testthat/edition 3

URL https://usepa.github.io/spmodel/

BugReports https://github.com/USEPA/spmodel/issues

NeedsCompilation no

Author Michael Dumelle [aut, cre] (ORCID: <https://orcid.org/0000-0002-3393-5529>), Matthew Heaton [ctb] (ORCID: <https://orcid.org/0000-0003-4654-9827>), Matt Higham [aut] (ORCID: <https://orcid.org/0009-0006-4217-625X>), Ryan A. Hill [ctb] (ORCID: <https://orcid.org/0000-0001-9583-0426>), Michael Mahon [ctb] (ORCID: <https://orcid.org/0000-0002-9436-2998>), Jay M. Ver Hoef [aut] (ORCID: <https://orcid.org/0000-0003-4302-6895>)

Maintainer Michael Dumelle <Dumelle.Michael@epa.gov>

Repository CRAN

Date/Publication 2026-09-10 18:40:02 UTC

Contents

`AICc` *Compute AICc of fitted model objects*

Description

Compute AICc for one or several fitted model objects for which a log-likelihood value can be obtained.

Usage

```
AICc(object, ..., k = 2)
```

Arguments

`object`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()` where `estmethod` is `"ml"` or `"reml"`.
`...`	Optionally more fitted model objects.
`k`	The penalty parameter, taken to be 2. Currently not allowed to differ from 2 (needed for generic consistency).

Details

When comparing models fit by maximum or restricted maximum likelihood, the smaller the AICc, the better the fit. The AICc contains a correction to AIC for small sample sizes. The theory of AICc requires that the log-likelihood has been maximized, and hence, no AICc methods exist for models where `estmethod` is not `"ml"` or `"reml"`. Additionally, AICc comparisons between `"ml"` and `"reml"` models are meaningless – comparisons should only be made within a set of models estimated using `"ml"` or a set of models estimated using `"reml"`. AICc comparisons for `"reml"` must use the same fixed effects. To vary the covariance parameters and fixed effects simultaneously, use `"ml"`.

Hoeting et al. (2006) study AIC and AICc in a spatial context, using the AIC definition $-2loglik + 2(estparams)$ and the AICc definition as $-2loglik + 2n(estparams)/(n - estparams - 1)$, where n is the sample size and $estparams$ is the number of estimated parameters. For `"ml"`, $estparams$ is the number of estimated covariance parameters plus the number of estimated fixed effects. For `"reml"`, $estparams$ is the number of estimated covariance parameters.

Value

If just one object is provided, a numeric value with the corresponding AICc.

If multiple objects are provided, a `data.frame` with rows corresponding to the objects and columns representing the number of parameters estimated (`df`) and the AICc.

See Also

`stats::AIC()` `stats::BIC()`

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
AICc(spmod)
AIC(spmod)
BIC(spmod)
```

---

anova.spmodel *Compute analysis of variance and likelihood ratio tests of fitted model objects*

Description

Compute analysis of variance tables for a fitted model object or a likelihood ratio test for two fitted model objects.

Usage

```
## S3 method for class 'splm'
anova(object, ..., test = TRUE, Terms, L, ddf)

## S3 method for class 'spautor'
anova(object, ..., test = TRUE, Terms, L, ddf)

## S3 method for class 'spglm'
anova(object, ..., test = TRUE, Terms, L, ddf)

## S3 method for class 'spgautor'
anova(object, ..., test = TRUE, Terms, L, ddf)

## S3 method for class 'anova.splm'
tidy(x, ...)

## S3 method for class 'anova.spautor'
tidy(x, ...)

## S3 method for class 'anova.spglm'
tidy(x, ...)

## S3 method for class 'anova.spgautor'
tidy(x, ...)
```

Arguments

`object`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.
`...`	An additional fitted model object.
`test`	A logical value indicating whether p-values from asymptotic Chi-squared hypothesis tests should be returned. Defaults to `TRUE`.
`Terms`	An optional character or integer vector that specifies terms in the model used to jointly compute test statistics and p-values (if `test = TRUE`) against a null hypothesis of zero. `Terms` is only used when a single fitted model object is passed to the function. If `Terms` is a character vector, it should contain the names of the fixed effect terms. If `Terms` is an integer vector, it should correspond to the order (starting at one) of the names of the fixed effect terms. The easiest way to obtain the names of all possible terms is to run `tidy(anova(object))$effects` (the integer representation matches the positions of this vector).
`L`	An optional numeric matrix or list specifying linear combinations of the coefficients in the model used to compute test statistics and p-values (if `test = TRUE`) for coefficient constraints corresponding to a null hypothesis of zero. `L` is only used when a single fitted model object is passed to the function. If `L` is a numeric matrix, its rows indicate coefficient constraints and its columns represent coefficients. Then a single hypothesis test is conducted against a null hypothesis of zero. If `L` is a list, each list element is a numeric matrix specified as above. Then

separate hypothesis tests are conducted. The easiest way to obtain all possible coefficients is to run `tidy(object)$term`.

`ddf` The denominator degrees of freedom used. `"asymptotic"` implements an asymptotic chi-squared test. `"satterthwaite"` impelments a Satterthwaite/Fai-Cornelius F-test. The default is `"satterthwaite"` when the sample size is less than or equal to 500 and `"asymptotic"` otherwise.

`x` An object from `anova(object)`.

Details

When one fitted model object is present, `anova()` performs a general linear hypothesis test corresponding to some hypothesis specified by a matrix of constraints. If `Terms` and `L` are not specified, each model term is tested against zero (which correspond to type III or marginal hypothesis tests from classical ANOVA). If `Terms` is specified and `L` is not specified, all terms are tested jointly against zero. When `L` is specified, the linear combinations of terms specified by `L` are jointly tested against zero.

When two fitted model objects are present, one must be a "reduced" model nested in a "full" model. Then `anova()` performs a likelihood ratio test.

Value

When `ddf` is `"asymptotic"`, `anova()` returns a data frame with degrees of freedom (`Df`), test statistics (`Chi2`), and p-values (`Pr(>Chi2)` if `test = TRUE`) corresponding to asymptotic Chi-squared hypothesis tests for each model term. When `ddf` is `"satterthwaite"`, `anova()` instead numerator degrees of freedom (`NumDF`), denominator degrees of freedom (`DenDF`), F statistics (`F value`), and p-values (`Pr(>F)` if `test = TRUE`) for each model term.

When two fitted model objects are present, `anova()` returns a data frame with the difference in degrees of freedom between the full and reduced model (`Df`), a test statistic (`Chi2`), and a p-value corresponding to the likelihood ratio test (`Pr(>Chi2)` if `test = TRUE`).

Whether one or two fitted model objects are provided, `tidy()` can be used to obtain tidy tibbles of the `anova(object)` output.

See Also

`satterthwaite()`

Examples

```
# one-model anova
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
anova(spmod)
tidy(anova(spmod))
# see terms
tidy(anova(spmod))$effects
tidy(anova(spmod, Terms = c("water", "tarp")))
# same as
```

```
tidy(anova(spmod, Terms = c(2, 3)))
# likelihood ratio test
lmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "none"
)
tidy(anova(spmod, lmod))
```

`augment.spmodel` *Augment data with information from fitted model objects*

Description

Augment accepts a fitted model object and a data set and adds information about each observation in the data set. New columns always begin with a `.` prefix to avoid overwriting columns in the original data set.

Augment behaves differently depending on whether the original data or new data requires augmenting. Typically, when augmenting the original data, only the fitted model object is specified, and when augmenting new data, the fitted model object and `newdata` is specified. When augmenting the original data, diagnostic statistics are augmented to each row in the data set. When augmenting new data, predictions and optional intervals or standard errors are augmented to each row in the new data set.

Usage

```
## S3 method for class 'splm'
augment(
  x,
  drop = TRUE,
  newdata = NULL,
  se_fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  local,
  ...
)

## S3 method for class 'spautor'
augment(
  x,
  drop = TRUE,
  newdata = NULL,
  se_fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  local,
  ...
```

```
)

## S3 method for class 'spglm'
augment(
  x,
  drop = TRUE,
  newdata = NULL,
  type.predict = c("link", "response"),
  type.residuals = c("deviance", "pearson", "response"),
  se_fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  newdata_size,
  level = 0.95,
  local = local,
  var_correct = TRUE,
  ...
)

## S3 method for class 'spgautor'
augment(
  x,
  drop = TRUE,
  newdata = NULL,
  type.predict = c("link", "response"),
  type.residuals = c("deviance", "pearson", "response"),
  se_fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  newdata_size,
  level = 0.95,
  local,
  var_correct = TRUE,
  ...
)
```

Arguments

x	A fitted model object from splm() or spautor().
drop	A logical indicating whether to drop extra variables in the fitted model object x when augmenting. The default for drop is TRUE. drop is ignored if augmenting newdata.
newdata	A data frame or tibble containing observations requiring prediction. All of the original explanatory variables used to create the fitted model object x must be present in newdata. Defaults to NULL, which indicates that nothing has been passed to newdata.
se_fit	Logical indicating whether or not a .se.fit column should be added to augmented output. Defaults to FALSE. When newdata is not supplied, .se.fit is the standard error of the fitted mean at the observed locations (on the link scale

	for spglm() and spgautor() model objects), matching predict(interval = "confidence").
interval	Character indicating the type of interval columns (.lower and .upper) to add to the augmented output. Defaults to "none".
level	Tolerance/confidence level. The default is 0.95.
local	A list or logical. If a list, specific list elements described in predict.spmodel() control the big data approximation behavior. If a logical, TRUE chooses default list elements for the list version of local as specified in predict.spmodel(). Defaults to FALSE, which performs exact computations.
...	Other arguments. Not used (needed for generic consistency).
type.predict	The scale (response or link) of fitted values and predictions obtained using spglm() or spgautor objects.
type.residuals	The residual type (deviance, pearson, or response) of fitted models from spglm() or spgautor objects. Ignored if newdata is specified.
newdata_size	The size value for each observation in newdata used when predicting for the binomial family.
var_correct	A logical indicating whether to return the corrected prediction variances when predicting via models fit using spglm() or spgautor(). The default is TRUE.

Details

augment() returns a tibble with the same class as data. That is, if data is an sf object, then the augmented object (obtained via augment(x)) will be an sf object as well. When augmenting newdata, the augmented object has the same class as data.

Missing response values from the original data can be augmented as if they were a newdata object by providing x$newdata to the newdata argument (where x is the name of the fitted model object). This is the only way to compute predictions for spautor() and spgautor() fitted model objects.

Value

When augmenting the original data set, a tibble with additional columns

- .fitted Fitted value
- .resid Response residual (the difference between observed and fitted values)
- .hat Leverage (diagonal of the hat matrix)
- .cooksd Cook's distance
- .std.resid Standardized residuals
- .se.fit Standard error of the fitted value.

When augmenting a new data set, a tibble with additional columns

- .fitted Predicted (or fitted) value
- .lower Lower bound on interval
- .upper Upper bound on interval
- .se.fit Standard error of the predicted (or fitted) value

Autoregressive models and prediction quantities

For `spautor()` and `spgautor()` model objects the prediction locations are part of the model: they determine the neighbor structure and hence the covariance of the observed data itself. `se_fit` and `interval` are therefore only defined at those locations, and `object$newdata` is automatically used when `se_fit` or `interval` is requested (and hence the rows returned by `augment()`) correspond to `object$newdata`. `splm()` and `spglm()` model objects have no such restriction, as their covariance does not depend on where predictions are made.

See Also

`tidy.spmodel()` `glance.spmodel()` `predict.spmodel()`

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
augment(spmod)
spmod_sulf <- splm(sulfate ~ 1, data = sulfate, spcov_type = "exponential")
augment(spmod_sulf)
augment(spmod_sulf, newdata = sulfate_preds)
# missingness in original data
spmod_seal <- spautor(log_trend ~ 1, data = seal, spcov_type = "car")
augment(spmod_seal)
augment(spmod_seal, newdata = spmod_seal$newdata)
```

AUROC *Area Under Receiver Operating Characteristic Curve*

Description

Compare area under the receiver operating characteristic curve (AUROC) for binary (e.g., logistic) models. The area under the ROC curve provides a measure of the model's classification accuracy averaged over all possible threshold values.

Usage

```
AUROC(object, ...)

## S3 method for class 'spglm'
AUROC(object, ...)

## S3 method for class 'spgautor'
AUROC(object, ...)
```

Arguments

`object`	A fitted model object from `spglm()` or `spgautor()`) where `family = "binomial"` and the response values are binary, representing a single success or failure for each datum.
`...`	Additional arguments to `pROC::auc()`.

Value

The area under the receiver operating characteristic curve.

References

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J. C., & Müller, M. (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC bioinformatics*, 12, 1-8.

Examples

```
spgmod <- spglm(presence ~ elev,
  family = "binomial", data = moose,
  spcov_type = "exponential"
)
AUROC(spgmod)
```

`caribou` *A caribou forage experiment*

Description

A caribou forage experiment.

Usage

```
caribou
```

Format

A `tibble` with 30 rows and 5 columns:

- water: A factor representing whether water was added. Takes values `N` (no water added) and `Y` (water added).
- tarp: A factor representing tarp cover. Takes values `clear` (a clear tarp), `shade` (a shade tarp), and `none` (no tarp).
- z: The percentage of nitrogen.
- x: The x-coordinate.
- y: The y-coordinate.

Source

These data were provided by Elizabeth Lenart of the Alaska Department of Fish and Game. The data were used in the publication listed in References.

References

Lenart, E.A., Bowyer, R.T., Ver Hoef, J.M. and Ruess, R.W. 2002. Climate Change and Caribou: Effects of Summer Weather on Forage. Canadian Journal of Zoology 80: 664-678.

`coef.spmodel` *Extract fitted model coefficients*

Description

coef extracts fitted model coefficients from fitted model objects. `coefficients` is an alias for it.

Usage

```
## S3 method for class 'splm'
coef(object, type = "fixed", ...)

## S3 method for class 'splm'
coefficients(object, type = "fixed", ...)

## S3 method for class 'spautor'
coef(object, type = "fixed", ...)

## S3 method for class 'spautor'
coefficients(object, type = "fixed", ...)

## S3 method for class 'spglm'
coef(object, type = "fixed", ...)

## S3 method for class 'spglm'
coefficients(object, type = "fixed", ...)

## S3 method for class 'spgautor'
coef(object, type = "fixed", ...)

## S3 method for class 'spgautor'
coefficients(object, type = "fixed", ...)
```

Arguments

`object`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.

type	"fixed" for fixed effect coefficients, "spcov" for spatial covariance parameter coefficients, or "randcov" for random effect variance coefficients. Defaults to "fixed". If type = "spcov", the coefficient vector is an spcov_params() object (which means that has class matching the spatial covariance function used).
...	Other arguments. Not used (needed for generic consistency).

Value

A named vector of coefficients.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
coef(spmod)
coefficients(spmod)
coef(spmod, type = "spcov")
```

---

conditional *Conditionally simulate from a model*

Description

Conditionally simulate prediction data from a model object.

Usage

```
conditional(object, ...)

## S3 method for class 'splm'
conditional(
  object,
  newdata,
  output = "newdata",
  samples = 1000,
  simulate_covparams = FALSE,
  ...
)

## S3 method for class 'spglm'
conditional(
  object,
  newdata,
  output = "newdata",
  type = c("link", "response", "new"),
  samples = 1000,
```

```
  newdata_size,
  ...
)
```

Arguments

object	A fitted model object.
...	Other arguments. Not used (needed for generic consistency).
newdata	A data frame or sf object in which to look for variables with which to predict. If a data frame, newdata must contain all variables used by formula(object) and all variables representing coordinates. If an sf object, newdata must contain all variables used by formula(object) and coordinates are obtained from the geometry of newdata. If omitted, missing data from the fitted model object are used.
output	The output type, which can be any subset of c("newdata", "beta", "object"). If simulate_covparams = TRUE, the output type can also be any subset of c("cov", "spcov", "randcov"). The default is "newdata". See Details for more.
samples	The number of conditional simulations. The default is 1,000.
simulate_covparams	For splm() model objects, whether to also simulate new covariance parameters for each sample. simulate_covparams requires object$vcov$cov to be specified during model fitting by selecting ddf = "satterthwaite". simulate_covparams = TRUE should generally not be used for sample sizes greater than 500 given its computational inefficiencies. The default is FALSE.
type	For spglm() model objects, the scale of the conditional simulations for newdata. When type = "link", the predicted means on the link scale are returned. When type = "response", the predicted means on the response scale are returned. When type = "new", a new observation is simulated from the appropriate response distribution with mean equal to the mean on the response scale and dispersion equal to the dispersion parameter from object. The default is "link".
newdata_size	The size value for each observation in newdata used when predicting for the binomial family, with a default value of 1.

Details

If "newdata" is in output, conditional simulations are returned for each row of newdata. If "beta" is in output, conditional simulations are returned for each fixed effect (i.e., element of coef(object). If "object" is in output, the observed data from object is returned once for each row of newdata. For example, c("newdata", "beta") returns the conditional simulations both for newdata and for the fixed effects. If "cov"/"spcov"/"randcov" is in output (only available when simulate_covparams = TRUE), the simulated covariance parameter draws themselves are returned.

Value

If output = "newdata", an a x b matrix of conditional simulations for each row in newdata, where a is the number of rows in newdata and b is the number of samples. If output = "beta", an p x b matrix of conditional simulations for each element in coef(object), where p is the number of

fixed effects and b is the number of samples. If `output = "object"`, an n x b matrix of observed data values, where n is the number of rows in `data` and b is the number of samples. If `output = "cov"/"spcov"/"randcov"` (`simulate_covparams = TRUE` only), a (covariance parameter) x b matrix of the of conditoinal simulations for each covariance parameter, where b is the number of samples. `"cov"` returns every covariance parameter, while `"spcov"/"randcov"` return just the spatial/random-effect covariance parameters, respectively.

If `output` has more than one element, a list is returned with the respetive elements named according to the relevant `output`. For example `output = c("newdata", "beta")` returns a list with elements `"newdata"` and `"beta"`, each containing the relevant output for `output = "newdata"` and `output = "beta"`, respectively.

Examples

```
set.seed(0)
spmod <- splm(sulfate ~ 1, data = sulfate, spcov_type = "exponential")
cond <- conditional(spmod, newdata = sulfate_preds)
predict(spmod, sulfate_preds[20, ], se.fit = TRUE)
c("fit_cond" = mean(cond[20, ]), "se.fit_cond" = sd(cond[20, ]))
hist(cond[20, ])
```

---

confint.spmodel *Confidence intervals for fitted model parameters*

Description

Computes confidence intervals for one or more parameters in a fitted model object.

Usage

```
## S3 method for class 'splm'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'spautor'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'spglm'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'spgautor'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.
`parm`	A specification of which parameters are to be given confidence intervals (a character vector of names). If missing, all parameters are considered.
`level`	The confidence level required. The default is `0.95`.
`...`	Other arguments. Not used (needed for generic consistency).

Value

Confidence intervals (two-sided and equal-tailed) for the fixed effect coefficients based on the confidence level specified by `level`. For `spglm()` or `spgautor()` fitted model objects, confidence intervals are on the link scale.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
confint(spmod)
confint(spmod, parm = "waterY", level = 0.90)
```

---

cooks.distance.spmodel

Compute Cook's distance

Description

Compute the Cook's distance for each observation from a fitted model object.

Usage

```
## S3 method for class 'splm'
cooks.distance(model, ...)

## S3 method for class 'spautor'
cooks.distance(model, ...)

## S3 method for class 'spglm'
cooks.distance(model, ...)

## S3 method for class 'spgautor'
cooks.distance(model, ...)
```

Arguments

`model`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.
`...`	Other arguments. Not used (needed for generic consistency).

Details

Cook's distance measures the influence of an observation on a fitted model object. If an observation is influential, its omission from the data noticeably impacts parameter estimates. The larger the Cook's distance, the larger the influence.

Value

A vector of Cook's distance values for each observation from the fitted model object.

See Also

augment.spmodel() hatvalues.spmodel() influence.spmodel() residuals.spmodel()

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
cooks.distance(spmod)
```

---

covmatrix *Create a covariance matrix*

Description

Create a covariance matrix from a fitted model object.

Usage

```
covmatrix(object, newdata, ...)

## S3 method for class 'splm'
covmatrix(object, newdata, cov_type, ...)

## S3 method for class 'spautor'
covmatrix(object, newdata, cov_type, ...)

## S3 method for class 'spglm'
covmatrix(object, newdata, cov_type, ...)

## S3 method for class 'spgautor'
covmatrix(object, newdata, cov_type, ...)
```

Arguments

object	A fitted model object (e.g., splm(), spautor(), spglm(), or spgautor()).
newdata	If omitted, the covariance matrix of the observed data is returned. If provided, newdata is a data frame or sf object that contains coordinate information required to construct the covariance between newdata and the observed data. If a data frame, newdata must contain variables that represent coordinates having the same name as the coordinates from the observed data used to fit object. If an sf object, coordinates are obtained from the geometry of newdata.

`...`	Other arguments. Not used (needed for generic consistency).
`cov_type`	The type of covariance matrix returned. If `newdata` is omitted or `cov_type` is `"obs.obs"`, the $n \times n$ covariance matrix of the observed data is returned, where n is the sample size used to fit `object`. If `newdata` is provided and `cov_type` is `"pred.obs"` (the default when `newdata` is provided), the $m \times n$ covariance matrix of the prediction and observed data is returned, where m is the number of elements in the prediction data. If `newdata` is provided and `cov_type` is `"obs.pred"`, the $n \times m$ covariance matrix of the observed and prediction data is returned. If `newdata` is provided and `cov_type` is `"pred.pred"`, the $m \times m$ covariance matrix of the prediction data is returned.

Value

If `newdata` is omitted, the covariance matrix of the observed data, which has dimension n x n, where n is the sample size used to fit `object`. If `newdata` is provided, the covariance matrix between the unobserved (new) data and the observed data, which has dimension m x n, where m is the number of new observations and n is the sample size used to fit `object`.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
covmatrix(spmod)
```

---

decorrelate *Apply the Spatial Decorrelation Transformation for Machine Learning Models*

Description

Apply the spatial decorrelation transformation for point-referenced data, allowing for random effects, anisotropy, partition factors, and big data methods.

Usage

```
decorrelate(
  formula,
  data,
  spcov_type,
  spcov_params,
  xcoord,
  ycoord,
  algorithm = "ranger",
  statistic = "RMSPE",
  training,
  evaluate_test,
```

```
  anisotropy = FALSE,
  random,
  randcov_params,
  partition_factor,
  ordering,
  local,
  grid,
  dense_grid,
  ...
)

## S3 method for class 'decorrelate_grid'
tidy(x, sort_by, decreasing, ...)
```

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms on the right, separated by + operators. . on the right-hand side represents every variable in data except the response and the x-coordinate/y-coordinate columns (xcoord/ycoord, or, for an sf object, the geometry column), which are never included via . (though they may still be given explicitly).
data	A data frame or sf object object that contains the variables in fixed, random, and partition_factor as well as geographical information. If an sf object is provided with POINT geometries, the x-coordinates and y-coordinates are used directly. If an sf object is provided with POLYGON geometries, the x-coordinates and y-coordinates are taken as the centroids of each polygon.
spcov_type	The spatial covariance type. Available options include "exponential", "spherical", "gaussian", "triangular", "circular", "cubic", "pentaspherical", "cosine", "wave", "jbessel", "gravity", "rquad", "magnetic", "matern", "cauchy", "pexponential", and "none". Parameterizations of each spatial covariance type are available in Details. Multiple spatial covariance types can be provided as a character vector, and then decorrelate() is called iteratively for each element and a list is returned for each model fit. The default for spcov_type is "exponential". When spcov_type is specified, all spatial covariance parameters are estimated. spcov_type is ignored if spcov_params is provided.
spcov_params	An object from spcov_params() that contains the spatial covariance parameters used by the spatial decorrelation transformation.
xcoord	The name of the column in data representing the x-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
ycoord	The name of the column in data representing the y-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
algorithm	The machine learning algorithm applied. Available options include "ranger", "randomForest", and "xgboost". "ranger" specifies a random forest via ranger::ranger(). "randomForest" specifies a random forest via randomForest::randomForest(). "xgboost" specifies a boosted decision tree ensemble via xgboost::xgboost().

`statistic` The statistic used to evaluate fit in the test data. Available options include `"bias"` (mean bias), `"MSPE"` (mean-squared-prediction error), `"RMSPE"` (root-mean-squared-prediction error) and `"cor2"` (the predictive R-squared; i.e., the squared correlation between observations and predictions).

`training` An list controlling how the training and test data are assigned when evaluating test data performance. The following arguments detail this process:

- `method`: The method used to evaluate test data performance. `"split"` will split `data` up into distinct training and test sets proportionally based on `p`. `"cv"` will split `data` up via k-fold cross validation based on `folds`, the number of folds.
- `p`: The proportion (a numeric vector between zero and one) of observations in `data` that should be assigned to the training data. The default is 0.8, which means that 80% of the observations are assigned to the training data and 20% to the test data. Ignored if `training_index` or `test_index` are provided.
- `replicate`: The number of times to replicate `"split"` with different random training and test assignments.
- `folds`: The number of folds to use in cross-validation. Requires `method = "cv"`. Ignored if `folds_index` is specified. The default is 5, matching the 80/20 default split above.
- `training_index`: A numeric vector that specifies which rows (i.e., indices) of `data` should be assigned to the training data. If omitted, defaults to the rows which are not already included in `test_index`.
- `test_index`: A numeric vector that specifies which rows (i.e., indices) of `data` should be assigned to the test data. If omitted, defaults to the rows which are not already included in `training_index`.
- `folds_index`: A numeric vector that specifies which rows of `data` are associated with each cross-validation fold. Requires `method = "cv"`.

If omitted, `training` is transformed into `list(method = "split", p = 0.8, replicate = 1)`.

`evaluate_test` A logical indicating whether a grid should be constructed and evaluated when spatial decorrelation parameters are known (i.e., `spcov_params` is specified, and, if random effects are included, `randcov_params` is specified). If `TRUE`, constructs and evalutes the grid after assigning observations to test and training data sets. If any parameters (spatial or random effects) are estimated via a grid search, `evaluate_test` is set to `TRUE`.

`anisotropy` A logical indicating whether (geometric) anisotropy should be modeled. Not required if the `rotate` and `scale` parameters in `spcov_params()` are 0 and 1, respectively. When `anisotropy` is `TRUE`, computational times can significantly increase. The default is `FALSE`.

`random` A one-sided linear formula describing the random effect structure of the model. Terms are specified to the right of the `~ operator`. Each term has the structure `x1 + ... + xn | g1/.../gm`, where `x1 + ... + xn` specifies the model for the random effects and `g1/.../gm` is the grouping structure. Separate terms are separated by `+` and must generally be wrapped in parentheses. Random intercepts are added to each model implicitly when at least one other variable is

defined. If a random intercept is not desired, this must be explicitly defined (e.g., x1 + ... + xn - 1 | g1/.../gm). If only a random intercept is desired for a grouping structure, the random intercept must be specified as 1 | g1/.../gm. Note that g1/.../gm is shorthand for (1 | g1/.../gm). If only random intercepts are desired and the shorthand notation is used, parentheses can be omitted.

randcov_params An object from randcov_params() that contains the random effect variances used by the spatial decorrelation transformation.

partition_factor

A one-sided linear formula with a single term specifying the partition factor. The partition factor assumes observations from different levels of the partition factor are uncorrelated.

ordering The data ordering applied. Available options include "grts", "maxmin", "middleout", "outsidein", "coordinate", "random", and "none". "grts" applies ordering using a spatially balanced GRTS sample via spsurvey::grts(). "maxmin" applies maximum minimum distance ordering via GPvecchia::order_maxmin_exact(). "middleout" applies middle out ordering via GPvecchia::order_middleout(). "outsidein" applies middle out ordering via GPvecchia::order_outsidein(). "coordinate" applies middle out ordering via GPvecchia::order_coordinate(..., coordinate = c(1, 2)), which orders from bottom-left to top-right of the spatial domain. "random" applies a completely random ordering. "none" applies no random ordering. The default is "maxmin" unless there are multiple observations at a single location, in which case the default is "grts".

local A optional logical or list controlling the big data approximation. If omitted, local is set to TRUE or FALSE based on the sample size (the number of non-missing observations in data) – if the sample size exceeds 5,000, local is set to TRUE. Otherwise it is set to FALSE. If local is FALSE, no big data approximation is implemented. If a list is provided, the following arguments detail the big data approximation:

- method: The big data approximation method. If method = "all", all observations are used and size is ignored. If method = "distance", the size data observations closest (in terms of Euclidean distance) to the observation requiring prediction are used. If method = "covariance", the size data observations with the highest covariance with the observation requiring prediction are used. If random effects and partition factors are not used in estimation and the spatial covariance function is monotone decreasing, "distance" and "covariance" are equivalent. The default is "covariance".
- size: The number of data observations to use when method is "distance" or "covariance". The default is 30.
- parallel: If TRUE, parallel processing via the parallel package is automatically used. This can significantly speed up computations even when method = "all" (i.e., no big data approximation is used), as predictions are spread out over multiple cores. The default is FALSE.
- ncores: If parallel = TRUE, the number of cores to parallelize over. The default is the number of available cores on your machine.

When local is a list, at least one list element must be provided to initialize default arguments for the other list elements. If local is TRUE, defaults for

	`local` are chosen such that `local` is transformed into `list(size = 30, method = "covariance", parallel = FALSE)`.
`grid`	An explicit grid of parameter values by which to evaluate fit. The names of `grid` must contain all the names returned by `decorrelate_grid(formula, data, ...)`.
`dense_grid`	If `grid` is not provided, `dense_grid` is a logical which controls the density of the constructed grid to be evaluated. If `dense_grid` is `TRUE`, a denser grid is used. If `dense_grid` is `FALSE`, a sparser grid is used. By default, `dense_grid` is `FALSE` when the sample size is greater than 5,000 and `TRUE` otherwise.
`...`	Other arguments to the functions called by `algorithm`.
`x`	An object from `object$grid`.
`sort_by`	Sort by a specific row in `x`. Fit statistics are `"bias"`, `"MSPE"`, `"RMSPE"`, and `"cor2"`. The default is `"MSPE"`.
`decreasing`	Whether `sort_by` should sort by decreasing order? If `sort_by = "cor2"`, the default is `TRUE`; otherwise it is `FALSE`.

Details

The spatial decorrelation transformation is a preprocessing transformation that reduces the impacts of spatial dependence (i.e., covariance, correlation) on machine learning models. Predictions are made on the decorrelated scale and then recorrelated to account for spatial dependence. See Heaton et al., 2025 for details.

`spcov_type` Details: The correlation matrix R controls the spatial dependence structure among observations. Parametric forms for R are given below, where $\eta = h/range$ for h distance between observations:

- exponential: $exp(-\eta)$
- spherical: $(1 - 1.5\eta + 0.5\eta^3) * I(h <= range)$
- gaussian: $exp(-\eta^2)$
- triangular: $(1 - \eta) * I(h <= range)$
- circular: $(1 - (2/\pi) * (m * sqrt(1 - m^2) + sin^{-1}(m))) * I(h <= range), m = min(\eta, 1)$
- cubic: $(1 - 7\eta^2 + 8.75\eta^3 - 3.5\eta^5 + 0.75\eta^7) * I(h <= range)$
- pentaspherical: $(1 - 1.875\eta + 1.25\eta^3 - 0.375\eta^5) * I(h <= range)$
- cosine: $cos(\eta)$
- wave: $sin(\eta)/\eta * I(h > 0) + I(h = 0)$
- jbessel: $Bj(h * range)$, Bj is Bessel-J function
- gravity: $(1 + \eta^2)^{-0.5}$
- rquad: $(1 + \eta^2)^{-1}$
- magnetic: $(1 + \eta^2)^{-1.5}$
- matern: $2^{1-extra}/\Gamma(extra) * \alpha^{extra} * Bk(\alpha, extra)$, $\alpha = (2extra)^{0.5} * \eta$, Bk is Bessel-K function with order $1/5 \leq extra \leq 5$
- cauchy: $(1 + \eta^2)^{-extra}$, $extra > 0$

- pexponential: $exp(h^{extra}/range)$, $0 < extra \leq 2$
- none: 0

All spatial covariance functions are valid in one spatial dimension. All spatial covariance functions except `triangular` and `cosine` are valid in two dimensions. An alias for `none` is `ie`.

`anisotropy` Details: By default, all spatial covariance parameters except `rotate` and `scale` as well as all random effect variance parameters are assumed unknown, requiring estimation. If either `rotate` or `scale` are given initial values other than 0 and 1 (respectively) in `spcov_params()`, `anisotropy` is implicitly set to `TRUE`. (Geometric) Anisotropy is modeled by transforming a covariance function that decays differently in different directions to one that decays equally in all directions via rotation and scaling of the original coordinates. The rotation is controlled by the `rotate` parameter in $[0, \pi]$ radians. The scaling is controlled by the `scale` parameter in $[0, 1]$. The anisotropy correction involves first a rotation of the coordinates clockwise by `rotate` and then a scaling of the coordinates' minor axis by the reciprocal of `scale`. The spatial covariance is then computed using these transformed coordinates.

`random` Details: If random effects are used (the estimation method must be `"reml"` or `"ml"`), the model can be written as $y = X\beta + Z1u1 + ...Zjuj + \tau + \epsilon$, where each Z is a random effects design matrix and each u is a random effect.

`partition_factor` Details: The partition factor can be represented in matrix form as P, where elements of P equal one for observations in the same level of the partition factor and zero otherwise. The covariance matrix involving only the spatial and random effects components is then multiplied element-wise (Hadmard product) by P, yielding the final covariance matrix.

`training` Details: When `replicate` or the number of cross validation folds is at least two, there are separate grids evaluated for each replication (or fold). Statistics in each grid are averaged across replications (or folds) to determine a final grid ranked by `statistic`.

`local` Details: The big data approximation works by leveraging the conditional nature of the spatial decorrelation transformation via the Vecchia approximation. The Vecchia approximation enables efficient computation of the conditional distribution by considering only the `size` most relevant observations in the ordering (rather than using all the observations).

Observations with `NA` response values are removed for model fitting, but their values can be predicted afterwards by running `predict(object)`.

Value

A list with many elements that store information about the fitted model object:

- `algorithm`: The machine learning algorithm used.
- `decorrelate_data`: The output of `decorrelate_data()` applied to `data`.
- `fit`: The fitted machine learning model object applied to the decorrelated data.
- `grid`: If used, the grid of spatial decorrelation parameters evaluated and their corresponding metrics when applied to the test data.
- `newdata`: The rows of `data` that have `NA` response values and are stored as prediction data.
- `training`: If used, the observations assigned to each training and test data set.
- `test`: If used, a list with the lowest (absolute) mean bias (bias), mean-squared-prediction error (MSPE), root-mean-squared-prediction error (RMSPE), and predictive R-squared (cor2).

References

Matthew J. Heaton, Andrew Millane, and Jake S. Rhodes. 2025. A Scalable Spatial Decorrelation Preprocessing Approach for Machine and Deep Learning. *Journal of Data Science*. 1-15, DOI 10.6339/25-JDS1210

Examples

```
decorr <- decorrelate(log_cond ~ temp, data = lake, spcov_type = "exponential")
tidy(decorr$grid)
```

`decorrelate_data` *Apply the Spatial Decorrelation Transformation to a Data Object*

Description

Apply the spatial decorrelation transformation to a data object. This object contains the transformed explanatory and response variables which can be used to fit a machine learning model. This object also contains information needed to decorrelate prediction data.

Usage

```
decorrelate_data(
  formula,
  data,
  spcov_params,
  xcoord,
  ycoord,
  randcov_params,
  partition_factor,
  ordering,
  local,
  ...
)
```

Arguments

`formula`	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the `~` operator and the terms on the right, separated by `+` operators. `.` on the right-hand side represents every variable in `data` except the response and the x-coordinate/y-coordinate columns (`xcoord`/`ycoord`, or, for an `sf` object, the geometry column), which are never included via `.` (though they may still be given explicitly).
`data`	A data frame or `sf` object object that contains the variables in `fixed`, `random`, and `partition_factor` as well as geographical information. If an `sf` object is provided with `POINT` geometries, the x-coordinates and y-coordinates are used directly. If an `sf` object is provided with `POLYGON` geometries, the x-coordinates and y-coordinates are taken as the centroids of each polygon.

spcov_params An object from spcov_params() that contains the spatial covariance parameters used by the spatial decorrelation transformation.

xcoord The name of the column in data representing the x-coordinate. Can be quoted or unquoted. Not required if data is an sf object.

ycoord The name of the column in data representing the y-coordinate. Can be quoted or unquoted. Not required if data is an sf object.

randcov_params An object from randcov_params() that contains the random effect variances used by the spatial decorrelation transformation.

partition_factor
A one-sided linear formula with a single term specifying the partition factor. The partition factor assumes observations from different levels of the partition factor are uncorrelated.

ordering The data ordering applied. Available options include "grts", "maxmin", "middleout", "outsidein", "coordinate", "random", and "none". "grts" applies ordering using a spatially balanced GRTS sample via spsurvey::grts(). "maxmin" applies maximum minimum distance ordering via GPvecchia::order_maxmin_exact(). "middleout" applies middle out ordering via GPvecchia::order_middleout(). "outsidein" applies middle out ordering via GPvecchia::order_outsidein(). "coordinate" applies middle out ordering via GPvecchia::order_coordinate(..., coordinate = c(1, 2)), which orders from bottom-left to top-right of the spatial domain. "random" applies a completely random ordering. "none" applies no random ordering. The default is "maxmin" unless there are multiple observations at a single location, in which case the default is "grts".

local A optional logical or list controlling the big data approximation. If omitted, local is set to TRUE or FALSE based on the sample size (the number of non-missing observations in data) – if the sample size exceeds 5,000, local is set to TRUE. Otherwise it is set to FALSE. If local is FALSE, no big data approximation is implemented. If a list is provided, the following arguments detail the big data approximation:

- method: The big data approximation method. If method = "all", all observations are used and size is ignored. If method = "distance", the size data observations closest (in terms of Euclidean distance) to the observation requiring prediction are used. If method = "covariance", the size data observations with the highest covariance with the observation requiring prediction are used. If random effects and partition factors are not used in estimation and the spatial covariance function is monotone decreasing, "distance" and "covariance" are equivalent. The default is "covariance".
- size: The number of data observations to use when method is "distance" or "covariance". The default is 30.
- parallel: If TRUE, parallel processing via the parallel package is automatically used. This can significantly speed up computations even when method = "all" (i.e., no big data approximation is used), as predictions are spread out over multiple cores. The default is FALSE.
- ncores: If parallel = TRUE, the number of cores to parallelize over. The default is the number of available cores on your machine.

When `local` is a list, at least one list element must be provided to initialize default arguments for the other list elements. If `local` is `TRUE`, defaults for `local` are chosen such that `local` is transformed into `list(size = 30, method = "covariance", parallel = FALSE)`.

`...` Other arguments to the functions called by `algorithm`.

Details

The spatial decorrelation transformation is a preprocessing transformation that reduces the impacts of spatial dependence (i.e., covariance, correlation) on machine learning models. See `decorrelate()` and Heaton et al., 2025 for more details.

Value

A list with many elements that store information about the fitted model object. Importantly, the list contains the following elements:

- `X`: The original fixed effects design matrix (of explanatory variables)
- `y`: The original response variable
- `tX`: The spatially decorrelated transformed fixed effects design matrix
- `ty`: The spatially decorrelated transformed response variable

References

Matthew J. Heaton, Andrew Millane, and Jake S. Rhodes. 2025. A Scalable Spatial Decorrelation Preprocessing Approach for Machine and Deep Learning. *Journal of Data Science*. 1-15, DOI 10.6339/25-JDS1210

See Also

`decorrelate()` `spcov_params()` `randcov_params()`

Examples

```
params <- spcov_params("exponential", de = 1, ie = 0.2, range = 1e5)
decorr <- decorrelate_data(log_cond ~ temp, data = lake, spcov_params = params)
head(cbind(decorr$X, decorr$tX))
head(cbind(decorr$y, decorr$ty))
```

---

decorrelate_grid *Create a Spatial Decorrelation Transformation Grid*

Description

Create a spatial decorrelation transformation grid of initial parameters to be evaluated via a grid search.

Usage

```
decorrelate_grid(
  formula,
  data,
  spcov_type,
  spcov_params,
  xcoord,
  ycoord,
  anisotropy = FALSE,
  random,
  randcov_params,
  dense_grid
)
```

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms on the right, separated by + operators. . on the right-hand side represents every variable in data except the response and the x-coordinate/y-coordinate columns (xcoord/ycoord, or, for an sf object, the geometry column), which are never included via . (though they may still be given explicitly).
data	A data frame or sf object object that contains the variables in fixed, random, and partition_factor as well as geographical information. If an sf object is provided with POINT geometries, the x-coordinates and y-coordinates are used directly. If an sf object is provided with POLYGON geometries, the x-coordinates and y-coordinates are taken as the centroids of each polygon.
spcov_type	The spatial covariance type. Available options include "exponential", "spherical", "gaussian", "triangular", "circular", "cubic", "pentaspherical", "cosine", "wave", "jbessel", "gravity", "rquad", "magnetic", "matern", "cauchy", "pexponential", and "none". Parameterizations of each spatial covariance type are available in Details. Multiple spatial covariance types can be provided as a character vector, and then decorrelate() is called iteratively for each element and a list is returned for each model fit. The default for spcov_type is "exponential". When spcov_type is specified, all spatial covariance parameters are estimated. spcov_type is ignored if spcov_params is provided.
spcov_params	An object from spcov_params() that contains the spatial covariance parameters used by the spatial decorrelation transformation.
xcoord	The name of the column in data representing the x-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
ycoord	The name of the column in data representing the y-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
anisotropy	A logical indicating whether (geometric) anisotropy should be modeled. Not required if the rotate and scale parameters in spcov_params() are 0 and 1, respectively. When anisotropy is TRUE, computational times can significantly increase. The default is FALSE.

random	A one-sided linear formula describing the random effect structure of the model. Terms are specified to the right of the ~ operator. Each term has the structure x1 + ... + xn | g1/.../gm, where x1 + ... + xn specifies the model for the random effects and g1/.../gm is the grouping structure. Separate terms are separated by + and must generally be wrapped in parentheses. Random intercepts are added to each model implicitly when at least one other variable is defined. If a random intercept is not desired, this must be explicitly defined (e.g., x1 + ... + xn - 1 | g1/.../gm). If only a random intercept is desired for a grouping structure, the random intercept must be specified as 1 | g1/.../gm. Note that g1/.../gm is shorthand for (1 | g1/.../gm). If only random intercepts are desired and the shorthand notation is used, parentheses can be omitted.
randcov_params	An object from randcov_params() that contains the random effect variances used by the spatial decorrelation transformation.
dense_grid	A logical which controls the density of the constructed grid to be evaluated. If dense_grid is TRUE, a denser grid is used. If dense_grid is FALSE, a sparser grid is used. By default, dense_grid is FALSE when the sample size is greater than 5,000 and TRUE otherwise.

Value

A grid of spatial decorrelation parameters stored as a data.frame.

Examples

```
decorrelate_grid(log_cond ~ temp, data = lake, spcov_type = "exponential")
```

---

decorrelate_newdata *Apply the Spatial Decorrelation Transformation to a Newdata Object for Prediction*

Description

Apply the spatial decorrelation transformation to a newdata object. This object contains explanatory variables that are transformed for prediction accoring to some spatial decorrelation transformation.

Usage

```
decorrelate_newdata(object, newdata, local, ...)
```

Arguments

object	A decorrelate_data() object.
newdata	A data frame or sf object in which to look for variables with which to predict. If a data frame, newdata must contain all variables used by formula(object) and all variables representing coordinates. If an sf object, newdata must contain all variables used by formula(object) and coordinates are obtained from the geometry of newdata. If omitted, missing data from the fitted model object are used.

`local` A optional logical or list controlling the big data approximation. If omitted, `local` is set to `TRUE` or `FALSE` based on the sample size (the number of non-missing observations in `data`) – if the sample size exceeds 5,000, `local` is set to `TRUE`. Otherwise it is set to `FALSE`. If `local` is `FALSE`, no big data approximation is implemented. If a list is provided, the following arguments detail the big data approximation:

- `method`: The big data approximation method. If `method = "all"`, all observations are used and `size` is ignored. If `method = "distance"`, the `size` data observations closest (in terms of Euclidean distance) to the observation requiring prediction are used. If `method = "covariance"`, the `size` data observations with the highest covariance with the observation requiring prediction are used. If random effects and partition factors are not used in estimation and the spatial covariance function is monotone decreasing, `"distance"` and `"covariance"` are equivalent. The default is `"covariance"`.
- `size`: The number of data observations to use when `method` is `"distance"` or `"covariance"`. The default is 30.
- `parallel`: If `TRUE`, parallel processing via the parallel package is automatically used. This can significantly speed up computations even when `method = "all"` (i.e., no big data approximation is used), as predictions are spread out over multiple cores. The default is `FALSE`.
- `ncores`: If `parallel = TRUE`, the number of cores to parallelize over. The default is the number of available cores on your machine.

When `local` is a list, at least one list element must be provided to initialize default arguments for the other list elements. If `local` is `TRUE`, defaults for `local` are chosen such that `local` is transformed into `list(size = 30, method = "covariance", parallel = FALSE)`.

... Other arguments.

Value

A list with many elements that store information about the fitted model object. Importantly, the list contains the following element:

- `X_newdata`: The original fixed effects design matrix (of explanatory variables) for the prediction data.
- `tX_newdata`: The spatially decorrelated fixed effects design matrix for the prediction data.

Examples

```
params <- spcov_params("exponential", de = 1, ie = 0.2, range = 1e5)
decorr <- decorrelate_data(log_cond ~ temp, data = lake, spcov_params = params)
decorr_newdata <- decorrelate_newdata(decorr, newdata = lake_preds)
head(decorr_newdata$tX_newdata)
```

`deviance.spmodel` *Fitted model deviance*

Description

Returns the deviance of a fitted model object.

Usage

```
## S3 method for class 'splm'
deviance(object, ...)

## S3 method for class 'spautor'
deviance(object, ...)

## S3 method for class 'spglm'
deviance(object, ...)

## S3 method for class 'spgautor'
deviance(object, ...)
```

Arguments

`object`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`, where `estmethod` is `"ml"` or `"reml"`.
`...`	Other arguments. Not used (needed for generic consistency).

Details

For objects estimated using `"ml"` or `"reml"`, the deviance is twice the difference in log-likelihoods between the saturated (perfect-fit) model and the fitted model.

Value

The deviance.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
deviance(spmod)
```

`dispersion_initial` *Create a dispersion parameter initial object*

Description

Create a dispersion parameter initial object that specifies initial and/or known values to use while estimating the dispersion parameter with `spglm()` or `spgautor()`.

Usage

```
dispersion_initial(family, dispersion, known)
```

Arguments

`family`	The generalized linear model family describing the distribution of the response variable to be used. `"poisson"`, `"nbinomial"`, `"binomial"`, `"beta"`, `"Gamma"`, and `"inverse.gaussian"`.
`dispersion`	The value of the dispersion parameter.
`known`	A character vector indicating whether the dispersion parameter is to be assumed known. The value `"dispersion"` or `"given"` is assumes the dispersion parameter is known.

Details

The `dispersion_initial` list is later passed to `spglm()` or `spgautor()`.

The variance function of an individual y (given μ) for each generalized linear model family is given below:

- family: $Var(y)$
- poisson: $\mu\phi$
- nbinomial: $\mu + \mu^2/\phi$
- binomial: $n\mu(1-\mu)\phi$
- beta: $\mu(1-\mu)/(1+\phi)$
- Gamma: μ^2/ϕ
- inverse.gaussian: μ^2/ϕ

The parameter ϕ is a dispersion parameter that influences $Var(y)$. For the `poisson` and `binomial` families, ϕ is always one. Note that this inverse Gaussian parameterization is different than a standard inverse Gaussian parameterization, which has variance μ^3/λ. Setting $\phi = \lambda/\mu$ yields our parameterization, which is preferred for computational stability. Also note that the dispersion parameter is often defined in the literature as $V(\mu)\phi$, where $V(\mu)$ is the variance function of the mean. We do not use this parameterization, which is important to recognize while interpreting dispersion parameter estimates using `spglm()` or `spgautor()`. For more on generalized linear model constructions, see McCullagh and Nelder (1989).

Value

A list with two elements: `initial` and `is_known`. `initial` is a named numeric vector indicating the dispersion parameters with a specified initial and/or known value. `is_known` is a named numeric vector indicating whether the dispersion parameters in `initial` is known or not. The class of the list matches the value given to the `family` argument.

References

McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models*. London: Chapman and Hall.

Examples

```
# known dispersion value 1
dispersion_initial("nbinomial", dispersion = 1, known = "dispersion")
```

`dispersion_params` *Create a dispersion parameter object*

Description

Create a dispersion parameter object for use with other functions.

Usage

```
dispersion_params(family, dispersion)
```

Arguments

`family`	The generalized linear model family describing the distribution of the response variable to be used. `"poisson"`, `"nbinomial"`, `"binomial"`, `"beta"`, `"Gamma"`, and `"inverse.gaussian"`.
`dispersion`	The value of the dispersion parameter.

Details

The variance function of an individual y (given μ) for each generalized linear model family is given below:

- family: $Var(y)$
- poisson: $\mu\phi$
- nbinomial: $\mu + \mu^2/\phi$
- binomial: $n\mu(1-\mu)\phi$
- beta: $\mu(1-\mu)/(1+\phi)$
- Gamma: μ^2/ϕ
- inverse.gaussian: μ^2/ϕ

The parameter ϕ is a dispersion parameter that influences $Var(y)$. For the `poisson` and `binomial` families, ϕ is always one. Note that this inverse Gaussian parameterization is different than a standard inverse Gaussian parameterization, which has variance μ^3/λ. Setting $\phi = \lambda/\mu$ yields our parameterization, which is preferred for computational stability. Also note that the dispersion parameter is often defined in the literature as $V(\mu)\phi$, where $V(\mu)$ is the variance function of the mean. We do not use this parameterization, which is important to recognize while interpreting dispersion parameter estimates using `spglm()` or `spgautor()`. For more on generalized linear model constructions, see McCullagh and Nelder (1989).

Value

A named numeric vector with class `family` containing the dispersion.

References

McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models*. London: Chapman and Hall.

Examples

```
dispersion_params("beta", dispersion = 1)
```

`eacf` *Compute the empirical autocovariance*

Description

Compute the empirical autocovariance (i.e., empirical covariance) for varying bin sizes and cutoff values.

Usage

```
eacf(
  formula,
  data,
  xcoord,
  ycoord,
  cloud = FALSE,
  bins = 15,
  cutoff,
  dist_matrix,
  partition_factor
)

## S3 method for class 'eacf'
plot(x, ...)
```

Arguments

formula	A formula describing the fixed effect structure. . on the right-hand side represents every variable in data except the response and the x-coordinate/y-coordinate columns (xcoord/ycoord, or, for an sf object, the geometry column), which are never included via . (though they may still be given explicitly).
data	A data frame or sf object containing the variables in formula and geographic information.
xcoord	Name of the variable in data representing the x-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
ycoord	Name of the variable in data representing the y-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
cloud	A logical indicating whether the empirical autocovariance should be summarized by distance class or not. When cloud = FALSE (the default), pairwise autocovariances are binned and averaged within distance classes. When cloud = TRUE, all pairwise autocovariances and distances are returned (this is known as the "cloud" autocovariance).
bins	The number of equally spaced bins. The default is 15. Ignored if cloud = TRUE.
cutoff	The maximum distance considered. The default is half the diagonal of the bounding box from the coordinates.
dist_matrix	A distance matrix to be used instead of providing coordinate names.
partition_factor	An optional formula specifying the partition factor. If specified, autocovariances are only computed for observations sharing the same level of the partition factor.
x	An object from eacf().
...	Other arguments passed to other methods.

Details

The empirical autocovariance (i.e., empirical covariance) is a tool used to visualize and model spatial dependence by estimating the semivariance of a process at varying distances. For a constant-mean process, the autocovariance at distance h is denoted $Cov(h)$ and defined as $Cov(z1, z2)$. Under second-order stationarity, $Cov(h) = Cov(0) - \gamma(h)$, where $gamma(h)$ is the semivariance function at distance h. Typically the residuals from an ordinary least squares fit defined by formula are second-order stationary with mean zero. These residuals are used to compute the empirical autocovariance At a distance h, the empirical autocovariance is $1/N(h) \sum(r1 \times r2)$, where $N(h)$ is the number of (unique) pairs in the set of observations whose distance separation is h and r1 and r2 are residuals corresponding to observations whose distance separation is h. In spmodel, these distance bins actually contain observations whose distance separation is h +- c, where c is a constant determined implicitly by bins. Typically, only observations whose distance separation is below some cutoff are used to compute the empirical semivariogram (this cutoff is determined by cutoff).

Value

If cloud = FALSE, a tibble (data.frame) with distance bins (bins), the average distance (dist), the average autocovariance (acov), and the number of (unique) pairs (np). If cloud = TRUE, a tibble (data.frame) with distance (dist) and autocovariance (acov) for each unique pair.

Examples

```
eacf(sulfate ~ 1, sulfate)
plot(eacf(sulfate ~ 1, sulfate))
```

---

esv *Compute the empirical semivariogram*

Description

Compute the empirical semivariogram for varying bin sizes and cutoff values.

Usage

```
esv(
  formula,
  data,
  xcoord,
  ycoord,
  cloud = FALSE,
  robust = FALSE,
  bins = 15,
  cutoff,
  dist_matrix,
  partition_factor
)

## S3 method for class 'esv'
plot(x, ...)
```

Arguments

formula	A formula describing the fixed effect structure. . on the right-hand side represents every variable in data except the response and the x-coordinate/y-coordinate columns (xcoord/ycoord, or, for an sf object, the geometry column), which are never included via . (though they may still be given explicitly).
data	A data frame or sf object containing the variables in formula and geographic information.
xcoord	Name of the variable in data representing the x-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
ycoord	Name of the variable in data representing the y-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
cloud	A logical indicating whether the empirical semivariogram should be summarized by distance class or not. When cloud = FALSE (the default), pairwise semivariances are binned and averaged within distance classes. When cloud = TRUE, all pairwise semivariances and distances are returned (this is known as the "cloud" semivariogram).

robust	A logical indicating whether the robust semivariogram (Cressie and Hawkins, 1980) is used. The default is FALSE.
bins	The number of equally spaced bins. The default is 15. Ignored if cloud = TRUE.
cutoff	The maximum distance considered. The default is half the diagonal of the bounding box from the coordinates.
dist_matrix	A distance matrix to be used instead of providing coordinate names.
partition_factor	An optional formula specifying the partition factor. If specified, semivariances are only computed for observations sharing the same level of the partition factor.
x	An object from esv().
...	Other arguments passed to other methods.

Details

The empirical semivariogram is a tool used to visualize and model spatial dependence by estimating the semivariance of a process at varying distances. For a constant-mean process, the semivariance at distance h is denoted $\gamma(h)$ and defined as $0.5 * Var(z1 - z2)$. Under second-order stationarity, $\gamma(h) = Cov(0) - Cov(h)$, where $Cov(h)$ is the covariance function at distance h. Typically the residuals from an ordinary least squares fit defined by formula are second-order stationary with mean zero. These residuals are used to compute the empirical semivariogram. At a distance h, the empirical semivariance is $1/N(h) \sum (r1 - r2)^2$, where $N(h)$ is the number of (unique) pairs in the set of observations whose distance separation is h and r1 and r2 are residuals corresponding to observations whose distance separation is h. The robust version is described by Cressie and Hawkins (1980). In spmodel, these distance bins actually contain observations whose distance separation is h +- c, where c is a constant determined implicitly by bins. Typically, only observations whose distance separation is below some cutoff are used to compute the empirical semivariogram (this cutoff is determined by cutoff).

When using splm() with estmethod as "sv-wls", the empirical semivariogram is calculated internally and used to estimate spatial covariance parameters.

Value

If cloud = FALSE, a tibble (data.frame) with distance bins (bins), the average distance (dist), the average semivariance (gamma), and the number of (unique) pairs (np). If cloud = TRUE, a tibble (data.frame) with distance (dist) and semivariance (gamma) for each unique pair.

References

Cressie, N & Hawkins, D.M. 1980. Robust estimation of the variogram. *Journal of the International Association for Mathematical Geology*, **12**, 115-125.

Examples

```
esv(sulfate ~ 1, sulfate)
plot(esv(sulfate ~ 1, sulfate))
```

fc_borders *Four Corners State Borders*

Description

State borders for the four corners states in the United States: Arizona, Colorado, New Mexico, Utah.

Usage

```
fc_borders
```

Format

An `sf` object with 4 rows and 4 columns:

- NAME: State name.
- STUSPS: State postal code.
- GEO.ID: State GEO.ID from the United States Census Bureau.
- geometry: `POINT` geometry representing coordinates in a NAD83 projection (EPSG: 5070). Distances between points are in meters.

Source

The data source is the United States Census Bureau TIGER/Line Shapefiles.

fitted.spmodel *Extract model fitted values*

Description

Extract fitted values from fitted model objects. `fitted.values` is an alias.

Usage

```
## S3 method for class 'splm'
fitted(object, type = "response", ...)

## S3 method for class 'splm'
fitted.values(object, type = "response", ...)

## S3 method for class 'spautor'
fitted(object, type = "response", ...)

## S3 method for class 'spautor'
```

```
fitted.values(object, type = "response", ...)

## S3 method for class 'spglm'
fitted(object, type = "response", ...)

## S3 method for class 'spglm'
fitted.values(object, type = "response", ...)

## S3 method for class 'spgautor'
fitted(object, type = "response", ...)

## S3 method for class 'spgautor'
fitted.values(object, type = "response", ...)
```

Arguments

object	A fitted model object from splm(), spautor(), spglm(), or spgautor().
type	"response" for fitted values of the response, "spcov" for fitted values of the spatial random errors, or "randcov" for fitted values of the random effects. If from spglm() or spgautor(), "link" for fitted values on the link scale. The default is "response".
...	Other arguments. Not used (needed for generic consistency).

Details

When type is "response", the fitted values for each observation are the standard fitted values $X\hat{\beta}$. When type is "spcov" the fitted values for each observation are (generally) the best linear unbiased predictors of the spatial dependent and spatial independent random error. When type is "randcov", the fitted values for each level of each random effect are (generally) the best linear unbiased predictors of the corresponding random effect. The fitted values for type "spcov" and "randcov" can generally be used to check assumptions for each component of the fitted model object (e.g., check a Gaussian assumption).

Value

The fitted values according to type.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
fitted(spmod)
fitted.values(spmod)
fitted(spmod, type = "spcov")
```

`formula.spmodel` *Model formulae*

Description

Return formula used by a fitted model object.

Usage

```
## S3 method for class 'splm'
formula(x, ...)

## S3 method for class 'spautor'
formula(x, ...)

## S3 method for class 'spglm'
formula(x, ...)

## S3 method for class 'spgautor'
formula(x, ...)
```

Arguments

x	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.
...	Other arguments. Not used (needed for generic consistency).

Value

The formula used by a fitted model object.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
formula(spmod)
```

`glance.spmodel` *Glance at a fitted model object*

Description

Returns a row of model summaries from a fitted model object. Glance returns the same number of columns for all models and estimation methods. If a particular summary is undefined for a model or estimation method (e.g., likelihood statistics for estimation methods `"sv-wls"` or `"sv-cl"` of `splm()` objects), `NA` is returned for that summary.

Usage

```
## S3 method for class 'splm'
glance(x, ...)

## S3 method for class 'spautor'
glance(x, ...)

## S3 method for class 'spglm'
glance(x, ...)

## S3 method for class 'spgautor'
glance(x, ...)
```

Arguments

`x`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.
`...`	Other arguments. Not used (needed for generic consistency).

Value

A single-row tibble with columns

- `n` The sample size.
- `p` The number of fixed effects.
- `npar` The number of estimated covariance parameters.
- `value` The optimized value of the fitting function.
- `AIC` The AIC.
- `AICc` The AICc.
- `BIC` The BIC.
- `logLik` The log-likelihood.
- `deviance` The deviance.
- `pseudo.r.squared` The pseudo r-squared.

See Also

stats::AIC() AICc() stats::BIC() logLik.spmodel() deviance.spmodel() pseudoR2() tidy.spmodel() augment.spmodel()

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
glance(spmod)
```

---

glances *Glance at many fitted model objects*

Description

glances() repeatedly calls glance() on several fitted model objects and binds the output together, sorted by a column of interest.

Usage

```
glances(object, ...)

## S3 method for class 'splm'
glances(object, ..., sort_by = "AICc", decreasing = FALSE, warning = TRUE)

## S3 method for class 'spautor'
glances(object, ..., sort_by = "AICc", decreasing = FALSE, warning = TRUE)

## S3 method for class 'splm_list'
glances(object, ..., sort_by = "AICc", decreasing = FALSE, warning = TRUE)

## S3 method for class 'spautor_list'
glances(object, ..., sort_by = "AICc", decreasing = FALSE, warning = TRUE)

## S3 method for class 'spglm'
glances(object, ..., sort_by = "AICc", decreasing = FALSE, warning = TRUE)

## S3 method for class 'spgautor'
glances(object, ..., sort_by = "AICc", decreasing = FALSE, warning = TRUE)

## S3 method for class 'spglm_list'
glances(object, ..., sort_by = "AICc", decreasing = FALSE, warning = TRUE)

## S3 method for class 'spgautor_list'
glances(object, ..., sort_by = "AICc", decreasing = FALSE, warning = TRUE)
```

Arguments

object	A fitted model object from splm(), spautor(), spglm(), or spgautor().
...	Additional fitted model objects. Ignored if object has class splm_list, spautor_list, spglm_list, or spgautor_list.
sort_by	Sort by a glance statistic (i.e., the name of a column output from glance() or the order of model input (sort_by = "order"). The default is "AICc".
decreasing	Whether sort_by should sort by decreasing order? The default is FALSE.
warning	Whether a warning is displayed when model comparisons violate certain rules. The default is TRUE.

Value

A tibble where each row represents the output of glance() for each fitted model object.

Examples

```
lmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "none"
)
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
glances(lmod, spmod)
glances(lmod, spmod, sort_by = "logLik", decreasing = TRUE)
```

---

hatvalues.spmodel *Compute leverage (hat) values*

Description

Compute the leverage (hat) value for each observation from a fitted model object.

Usage

```
## S3 method for class 'splm'
hatvalues(model, ...)

## S3 method for class 'spautor'
hatvalues(model, ...)

## S3 method for class 'spglm'
hatvalues(model, ...)

## S3 method for class 'spgautor'
hatvalues(model, ...)
```

Arguments

model	A fitted model object from splm(), spautor(), spglm(), or spgautor().
...	Other arguments. Not used (needed for generic consistency).

Details

Leverage values measure how far an observation's explanatory variables are relative to the average of the explanatory variables. In other words, observations with high leverage are typically considered to have an extreme or unusual combination of explanatory variables. Leverage values are the diagonal of the hat (projection) matrix. The larger the hat value, the larger the leverage.

Value

A vector of leverage (hat) values for each observation from the fitted model object.

See Also

augment.spmodel() cooks.distance.spmodel() influence.spmodel() residuals.spmodel()

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
hatvalues(spmod)
```

---

influence.spmodel *Regression diagnostics*

Description

Provides basic quantities which are used in forming a wide variety of diagnostics for checking the quality of fitted model objects.

Usage

```
## S3 method for class 'splm'
influence(model, ...)

## S3 method for class 'spautor'
influence(model, ...)

## S3 method for class 'spglm'
influence(model, ...)

## S3 method for class 'spgautor'
influence(model, ...)
```

Arguments

model	A fitted model object from splm(), spautor(), spglm(), or spgautor().
...	Other arguments. Not used (needed for generic consistency).

Details

This function calls residuals.spmodel(), hatvalues.spmodel(), and cooks.distance.spmodel() and puts the results into a tibble. It is primarily used when calling augment.spmodel().

Value

A tibble with residuals (.resid), leverage values (.hat), cook's distance (.cooksd), and standardized residuals (.std.resid).

See Also

augment.spmodel() cooks.distance.spmodel() hatvalues.spmodel() residuals.spmodel()

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
influence(spmod)
```

---

kcv *Perform k-fold cross validation*

Description

Perform k-fold cross validation with options for computationally efficient approximations for big data. Generalizes loocv() (leave-one-out cross validation) to leaving out k folds of (approximately) equal size instead of single observations.

Usage

```
kcv(object, ...)

## S3 method for class 'splm'
kcv(
  object,
  k = 5,
  cv_predict = FALSE,
  se.fit = FALSE,
  local,
  interval = c("none", "prediction"),
```

```
  level = 0.95,
  folds_index,
  ...
)

## S3 method for class 'spautor'
kcv(
  object,
  k = 5,
  cv_predict = FALSE,
  se.fit = FALSE,
  local,
  interval = c("none", "prediction"),
  level = 0.95,
  folds_index,
  ...
)

## S3 method for class 'spglm'
kcv(
  object,
  k = 5,
  cv_predict = FALSE,
  type = c("link", "response"),
  se.fit = FALSE,
  delta = FALSE,
  local,
  folds_index,
  ...
)

## S3 method for class 'spgautor'
kcv(
  object,
  k,
  cv_predict = FALSE,
  type = c("link", "response"),
  se.fit = FALSE,
  delta = FALSE,
  local,
  folds_index,
  ...
)
```

Arguments

object	A fitted model object from splm(), spautor(), spglm(), or spgautor().
...	Other arguments. Not used (needed for generic consistency).

k	The number of folds. Must be a whole number at least 2 and no more than the sample size (the number of non-missing observations in object). The default is 5. If k equals the sample size, kcv() is equivalent to (and calls) loocv(). Ignored when folds_index is supplied.
cv_predict	A logical indicating whether the k-fold cross validation fitted values should be returned. Defaults to FALSE. If object is from spglm() or spgautor(), the fitted values returned are on the link scale.
se.fit	A logical indicating whether the k-fold cross validation prediction standard errors should be returned. Defaults to FALSE. If object is from spglm() or spgautor(), the standard errors correspond to the fitted values returned on the link scale.
local	A list or logical. If a list, specific list elements described in predict.spmodel() control the big data approximation behavior. If a logical, TRUE chooses default list elements for the list version of local as specified in predict.spmodel(). Defaults to FALSE, which performs exact computations.
interval	Whether to also report empirical k-fold cross validation prediction interval coverage in the returned fit statistics. "none" (the default) omits it; "prediction" reports it (see Details). Only available for splm()/spautor() objects.
level	The prediction interval level (e.g. 0.95) used to compute prediction interval coverage when interval = "prediction". Ignored otherwise. The default is 0.95.
folds_index	An optional vector, the same length as the number of non-missing observations in object and in the same order, assigning each observation to a fold (at least two distinct values required). When supplied, this fold assignment is used as-is, implying k is ignored and no random fold assignment is performed. The default is NULL, which randomly assigns k (approximately) equally sized folds.
type	The scale (response or link) of predictions obtained when cv_predict = TRUE and using spglm() or spgautor objects.
delta	A logical indicating whether to return delta method standard errors on the response scale when se.fit = TRUE and type = "response". The default is FALSE.

Details

Observations are randomly partitioned into k folds of (approximately) equal size. Each fold is held out from the data set in turn and the remaining data are used to make predictions for the held-out fold. This is compared to the true values of the held-out observations and several fit statistics are (sometimes optionally) computed: bias, mean-squared-prediction error (MSPE), root-mean-squared-prediction error (RMSPE), and the squared correlation (cor2) between the observed data and k-fold cross validation predictions (regarded as a prediction version of r-squared appropriate for comparing across spatial and nonspatial models), , and prediction interval coverage (cover.XX). Generally, bias should be near zero and prediction interval coverage at the intended level for well-fitting models. The lower the MSPE and RMSPE, the better the model fit (according to the k-fold cross validation criterion). The higher the cor2, the better the model fit (according to the k-fold cross validation criterion). cor2 and cover.XX are not returned when object was fit using spglm() or spgautor() because we do not observe the underlying latent mean.

When object is from splm() or spautor(), setting interval = "prediction" additionally reports the empirical coverage of the level (e.g. 95\ the proportion of held-out observations whose

true value falls within `fit +/- qnorm(1 - (1 - level) / 2) * se.fit` (the same normal-quantile interval `predict.spmodel()` uses by default). This is only available for `splm()`/`spautor()` objects, since `spglm()`/`spgautor()` have no observed-scale latent mean to compare against.

Value

If `cv_predict = FALSE` and `se.fit = FALSE`, a fit statistics tibble (with bias, MSPE, RMSPE, and cor2; see Details). If `cv_predict = TRUE` or `se.fit = TRUE`, a list with elements: `stats`, a fit statistics tibble (with bias, MSPE, RMSPE, and cor2; see Details); `cv_predict`, a numeric vector with k-fold cross validation predictions for each observation (if `cv_predict = TRUE`); and `se.fit`, a numeric vector with k-fold cross validation prediction standard errors for each observation (if `se.fit = TRUE`). When `object` is from `splm()` or `spautor()` and `interval = "prediction"`, the fit statistics tibble also has a `cover.XX` column (e.g. `cover.95` for `level = 0.95`; see Details).

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
kcv(spmod)
kcv(spmod, k = 10, cv_predict = TRUE, se.fit = TRUE)
```

---

labels.spmodel *Find labels from object*

Description

Find a suitable set of labels from a fitted model object.

Usage

```
## S3 method for class 'splm'
labels(object, ...)

## S3 method for class 'spautor'
labels(object, ...)

## S3 method for class 'spglm'
labels(object, ...)

## S3 method for class 'spgautor'
labels(object, ...)
```

Arguments

`object`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.
`...`	Other arguments. Not used (needed for generic consistency).

Value

A character vector containing the terms used for the fixed effects from a fitted model object.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
labels(spmod)
```

---

lake *National Lakes Assessment Data*

Description

Lake data collected as part of the United States Environmental Protection Agency's 2012 and 2017 National Lakes Assessment and LakeCat.

Usage

```
lake
```

Format

An `sf` object with 102 rows and 10 columns:

- comid: A common identifier from NHDPlusV2.
- log_cond: The natural logarithm of lake conductivity.
- cond: The lake conductivity.
- state: The US state: One of Arizona (AZ), Colorado (CO), Nevada (NV), Utah (UT).
- temp: Lake catchment 30-year average temperature (in degrees Celsius).
- precip: Lake watershed 30-year average precipitation (in centimeters).
- elev: Lake elevation (in meters).
- origin: Lake origin (human-made or natural).
- year: A factor representing year (2012 or 2017).
- geometry: `POINT` geometry representing coordinates in a NAD83 projection (EPSG: 5070). Distances between points are in meters.

lake_preds *Lakes Prediction Data*

Description

Lake prediction data collected as part of the United States Environmental Protection Agency's 2012 and 2017 National Lakes Assessment and LakeCat.

Usage

```
lake_preds
```

Format

An `sf` object with 10 rows and 8 columns:

- comid: A common identifier from NHDPlusV2.
- state: The US state: One of Arizona (AZ), Nevada (NV), Utah (UT).
- temp: Lake catchment 30-year average temperature (in degrees Celsius).
- precip: Lake watershed 30-year average precipitation (in centimeters).
- elev: Lake elevation (in meters).
- origin: Lake origin (human-made or natural).
- year: A factor representing year (2012 or 2017).
- geometry: `POINT` geometry representing coordinates in a NAD83 projection (EPSG: 5070). Distances between points are in meters.

logLik.spmodel *Extract log-likelihood*

Description

Find the log-likelihood of a fitted model when `estmethod` is `"ml"` or `"reml"`.

Usage

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

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

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

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

Arguments

object	A fitted model object from splm(), spautor(), spglm(), or spgautor() where estmethod is "ml" or "reml".
...	Other arguments. Not used (needed for generic consistency).

Value

The log-likelihood.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
logLik(spmod)
```

---

loocv *Perform leave-one-out cross validation*

Description

Perform leave-one-out cross validation with options for computationally efficient approximations for big data.

Usage

```
loocv(object, ...)

## S3 method for class 'splm'
loocv(
  object,
  cv_predict = FALSE,
  se.fit = FALSE,
  local,
  interval = c("none", "prediction"),
  level = 0.95,
  ...
)

## S3 method for class 'spautor'
loocv(
  object,
  cv_predict = FALSE,
  se.fit = FALSE,
  local,
  interval = c("none", "prediction"),
```

```
  level = 0.95,
  ...
)

## S3 method for class 'spglm'
loocv(
  object,
  cv_predict = FALSE,
  type = c("link", "response"),
  se.fit = FALSE,
  delta = FALSE,
  local,
  ...
)

## S3 method for class 'spgautor'
loocv(
  object,
  cv_predict = FALSE,
  type = c("link", "response"),
  se.fit = FALSE,
  delta = FALSE,
  local,
  ...
)
```

Arguments

`object`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.
`...`	Other arguments. Not used (needed for generic consistency).
`cv_predict`	A logical indicating whether the leave-one-out fitted values should be returned. Defaults to `FALSE`. If `object` is from `spglm()` or `spgautor()`, the fitted values returned are on the link scale.
`se.fit`	A logical indicating whether the leave-one-out prediction standard errors should be returned. Defaults to `FALSE`. If `object` is from `spglm()` or `spgautor()`, the standard errors correspond to the fitted values returned on the link scale.
`local`	A list or logical. If a list, specific list elements described in `predict.spmodel()` control the big data approximation behavior. If a logical, `TRUE` chooses default list elements for the list version of `local` as specified in `predict.spmodel()`. Defaults to `FALSE`, which performs exact computations.
`interval`	Whether to also report empirical leave-one-out prediction interval coverage in the returned fit statistics. `"none"` (the default) omits it; `"prediction"` reports it (see Details). Only available for `splm()`/`spautor()` objects.
`level`	The prediction interval level (e.g. 0.95) used to compute prediction interval coverage when `interval = "prediction"`. Ignored otherwise. The default is `0.95`.

type	The scale (response or link) of predictions obtained when cv_predict = TRUE and using spglm() or spgautor objects.
delta	A logical indicating whether to return delta method standard errors on the response scale when se.fit = TRUE and type = "response". The default is FALSE.

Details

Each observation is held-out from the data set and the remaining data are used to make a prediction for the held-out observation. This is compared to the true value of the observation and several fit statistics are (sometimes optionally) computed: bias, mean-squared-prediction error (MSPE), root-mean-squared-prediction error (RMSPE), and the squared correlation (cor2) between the observed data and leave-one-out predictions (regarded as a prediction version of r-squared appropriate for comparing across spatial and nonspatial models), and prediction interval coverage (cover.XX). Generally, bias should be near zero and prediction interval coverage at the intended level for well-fitting models. The lower the MSPE and RMSPE, the better the model fit (according to the leave-out-out criterion). The higher the cor2, the better the model fit (according to the leave-out-out criterion). cor2 and cover.XX are not returned when object was fit using spglm() or spgautor() because we do not observe the underlying latent mean.

Value

If cv_predict = FALSE and se.fit = FALSE, a fit statistics tibble (with bias, MSPE, RMSPE, and cor2; see Details). If cv_predict = TRUE or se.fit = TRUE, a list with elements: stats, a fit statistics tibble (with bias, MSPE, RMSPE, and cor2; see Details); cv_predict, a numeric vector with leave-one-out predictions for each observation (if cv_predict = TRUE); and se.fit, a numeric vector with leave-one-out prediction standard errors for each observation (if se.fit = TRUE). When object is from splm() or spautor() and interval = "prediction", the fit statistics tibble also has a cover.XX column (e.g. cover.95 for level = 0.95; see Details).

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
loocv(spmod)
loocv(spmod, cv_predict = TRUE, se.fit = TRUE)
```

---

model.frame.spmodel *Extract the model frame from a fitted model object*

Description

Extract the model frame from a fitted model object.

Usage

```
## S3 method for class 'splm'
model.frame(formula, ...)

## S3 method for class 'spautor'
model.frame(formula, ...)

## S3 method for class 'spglm'
model.frame(formula, ...)

## S3 method for class 'spgautor'
model.frame(formula, ...)
```

Arguments

formula	A fitted model object from splm(), spautor(), spglm(), or spgautor().
...	Other arguments. Not used (needed for generic consistency).

Value

A model frame that contains the variables used by the formula for the fitted model object.

See Also

stats::model.frame()

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
model.frame(spmod)
```

---

model.matrix.spmodel *Extract the model matrix from a fitted model object*

Description

Extract the model matrix (X) from a fitted model object.

Usage

```
## S3 method for class 'splm'
model.matrix(object, ...)

## S3 method for class 'spautor'
model.matrix(object, ...)

## S3 method for class 'spglm'
model.matrix(object, ...)

## S3 method for class 'spgautor'
model.matrix(object, ...)
```

Arguments

object	A fitted model object from splm(), spautor(), spglm(), or spgautor().
...	Other arguments. Not used (needed for generic consistency).

Value

The model matrix (of the fixed effects), whose rows represent observations and whose columns represent explanatory variables corresponding to each fixed effect.

See Also

stats::model.matrix()

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
model.matrix(spmod)
```

---

moose *Moose counts and presence in Alaska, USA*

Description

Moose counts and presence in Alaska, USA.

Usage

```
moose
```

Format

An sf object with 218 rows and 5 columns.

- elev: The elevation.
- strat: A factor representing strata (used for sampling). Can take values L and M.
- count: The count (number) of moose observed.
- presence: A binary factor representing whether no moose were observed (value 0) or at least one moose was observed (value 1).
- geometry: POINT geometry representing coordinates in an Alaska Albers projection (EPSG: 3338). Distances between points are in meters.

Source

Alaska Department of Fish and Game, Division of Wildlife Conservation has released this data set under the CC0 license.

---

moose_preds — *Locations at which to predict moose counts and presence in Alaska, USA*

Description

Locations at which to predict moose counts and presence in Alaska, USA.

Usage

```
moose_preds
```

Format

An sf object with 100 rows and 3 columns.

- elev: The elevation.
- strat: A factor representing strata (used for sampling). Can take values L and M.
- geometry: POINT geometry representing coordinates in an Alaska Albers projection (EPSG: 3338). Distances between points are in meters.

Source

Alaska Department of Fish and Game, Division of Wildlife Conservation has released this data set under the CC0 license.

moss *Heavy metals in mosses near a mining road in Alaska, USA*

Description

Heavy metals in mosses near a mining road in Alaska, USA.

Usage

```
moss
```

Format

An `sf` object with 365 rows and 12 columns:

- sample: A factor with a sample identifier. Some samples were replicated in the field or laboratory. As a result, there are 318 unique sample identifiers.
- field_dup: A factor representing field duplicate. Takes values `1` and `2`.
- lab_rep: A factor representing laboratory replicate. Takes values `1` and `2`.
- year: A factor representing year. Takes values `2001` and `2006`.
- sideroad: A factor representing direction relative to the haul road. Takes values `N` (north of the haul road) and `S` (south of the haul road).
- log_dist2road: The log of distance (in meters) to the haul road.
- dist2road: The distance (in meters) to the haul road.
- log_Zn: The log of zinc concentration in moss tissue (mg/kg).
- Zn: The zinc concentration in moss tissue (mg/kg).
- geometry: `POINT` geometry representing coordinates in an Alaska Albers projection (EPSG: 3338). Distances between points are in meters.

Source

Data were obtained from Peter Neitlich and Linda Hasselbach of the National Park Service. Data were used in the publications listed in References.

References

Neitlich, P.N., Ver Hoef, J.M., Berryman, S. D., Mines, A., Geiser, L.H., Hasselbach, L.M., and Shiel, A. E. 2017. Trends in Spatial Patterns of Heavy Metal Deposition on National Park Service Lands Along the Red Dog Mine Haul Road, Alaska, 2001-2006. PLOS ONE 12(5):e0177936 DOI:10.1371/journal.pone.0177936

Hasselbach, L., Ver Hoef, J.M., Ford, J., Neitlich, P., Berryman, S., Wolk B. and Bohle, T. 2005. Spatial Patterns of Cadmium, Lead and Zinc Deposition on National Park Service Lands in the Vicinity of Red Dog Mine, Alaska. Science of the Total Environment 348: 211-230.

`plot.spmodel` *Plot fitted model diagnostics*

Description

Plot fitted model diagnostics such as residuals vs fitted values, quantile-quantile, scale-location, Cook's distance, residuals vs leverage, Cook's distance vs leverage, a fitted spatial covariance function, and a fitted anisotropic level curve of equal correlation.

Usage

```
## S3 method for class 'splm'
plot(x, which, ...)

## S3 method for class 'spautor'
plot(x, which, ...)

## S3 method for class 'spglm'
plot(x, which, ...)

## S3 method for class 'spgautor'
plot(x, which, ...)
```

Arguments

`x`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.
`which`	An integer vector taking on values between 1 and 8, which indicates the plots to return. Available plots are described in Details. If `which` has length greater than one, additional plots are stepped through in order using `<Return>`. The default for `splm()` and `spglm()` fitted model objects is `which = c(1, 2, 7)`. The default for `spautor()` and `spgautor()` fitted model objects is `which = c(1, 2)`.
`...`	Other arguments passed to other methods.

Details

For all fitted model objects,, the values of `which` make the corresponding plot:

- 1: Standardized residuals vs fitted values (of the response)
- 2: Normal quantile-quantile plot of standardized residuals
- 3: Scale-location plot of standardized residuals
- 4: Cook's distance
- 5: Standardized residuals vs leverage
- 6: Cook's distance vs leverage

For `splm()` and `spglm()` fitted model objects, there are two additional values of `which`:

- 7: Fitted spatial covariance function vs distance
- 8: Fitted anisotropic (or isotropic) level curve of equal correlation

Value

No return value. Function called for plotting side effects.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
plot(spmod)
plot(spmod, which = c(1, 2, 4, 6))
```

---

predict.spmodel *Model predictions (Kriging)*

Description

Predicted values and intervals based on a fitted model object.

Usage

```
## S3 method for class 'splm'
predict(
  object,
  newdata,
  se.fit = FALSE,
  scale = NULL,
  df = Inf,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  type = c("response", "terms", "weight"),
  block = FALSE,
  local,
  terms = NULL,
  na.action = na.fail,
  ...
)

## S3 method for class 'spautor'
predict(
  object,
  newdata,
  se.fit = FALSE,
  scale = NULL,
  df = Inf,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
```

```
  type = c("response", "terms", "weight"),
  local,
  terms = NULL,
  na.action = na.fail,
  ...
)

## S3 method for class 'splm_list'
predict(
  object,
  newdata,
  se.fit = FALSE,
  scale = NULL,
  df = Inf,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  type = c("response", "terms"),
  local,
  terms = NULL,
  na.action = na.fail,
  ...
)

## S3 method for class 'spautor_list'
predict(
  object,
  newdata,
  se.fit = FALSE,
  scale = NULL,
  df = Inf,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  type = c("response", "terms"),
  local,
  terms = NULL,
  na.action = na.fail,
  ...
)

## S3 method for class 'splmRF'
predict(object, newdata, local, ...)

## S3 method for class 'spautorRF'
predict(object, newdata, local, ...)

## S3 method for class 'splmRF_list'
predict(object, newdata, local, ...)
```

```
## S3 method for class 'spautorRF_list'
predict(object, newdata, local, ...)

## S3 method for class 'spglm'
predict(
  object,
  newdata,
  type = c("link", "response", "terms", "weight"),
  se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  dispersion = NULL,
  terms = NULL,
  local,
  var_correct = TRUE,
  delta = FALSE,
  newdata_size,
  na.action = na.fail,
  ...
)

## S3 method for class 'spgautor'
predict(
  object,
  newdata,
  type = c("link", "response", "terms", "weight"),
  se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  dispersion = NULL,
  terms = NULL,
  local,
  var_correct = TRUE,
  delta = FALSE,
  newdata_size,
  na.action = na.fail,
  ...
)

## S3 method for class 'spglm_list'
predict(
  object,
  newdata,
  type = c("link", "response", "terms"),
  se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  dispersion = NULL,
```

```
  terms = NULL,
  local,
  var_correct = TRUE,
  newdata_size,
  na.action = na.fail,
  ...
)

## S3 method for class 'spgautor_list'
predict(
  object,
  newdata,
  type = c("link", "response", "terms"),
  se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  dispersion = NULL,
  terms = NULL,
  local,
  var_correct = TRUE,
  newdata_size,
  na.action = na.fail,
  ...
)

## S3 method for class 'decorrelate'
predict(object, newdata, local, ...)

## S3 method for class 'decorrelate_list'
predict(object, newdata, local, ...)
```

Arguments

object	A fitted model object.
newdata	A data frame or sf object in which to look for variables with which to predict. If a data frame, newdata must contain all variables used by formula(object) and all variables representing coordinates. If an sf object, newdata must contain all variables used by formula(object) and coordinates are obtained from the geometry of newdata. If omitted, missing data from the fitted model object are used.
se.fit	A logical indicating if standard errors are returned. The default is FALSE.
scale	A numeric constant by which to scale the regular standard errors and intervals. Similar to but slightly different than scale for stats::predict.lm(), because predictions form a spatial model may have different residual variances for each observation in newdata. The default is NULL, which returns the regular standard errors and intervals.
df	Degrees of freedom to use for confidence or prediction intervals (ignored if scale is not specified). The default is Inf.

`interval` Type of interval calculation. The default is `"none"`. Other options are `"confidence"` (for confidence intervals) and `"prediction"` (for prediction intervals). When `interval` is `"none"` or `"prediction"`, predictions are returned (and when requested, their corresponding uncertainties). When `interval` is `"confidence"`, mean estimates are returned (and when requested, their corresponding uncertainties). This `"none"` behavior differs from that of `lm()`, as `lm()` returns confidence uncertainties (in `.$se.fit`).

`level` Tolerance/confidence level. The default is `0.95`.

`type` The prediction type, either on the response scale, link scale (only for `spglm()` or `spgautor()` model objects), terms scale, or prediction (i.e., Kriging) weight scale.

`block` A logical indicating whether a block prediction over the entire region in `newdata` should be returned. When `block` is `TRUE`, `newdata` should be a dense grid of prediction locations that span the entire region. The default is `FALSE`, which returns point predictions for each location on `newdata`.

`local` A optional logical or list controlling the big data approximation. If omitted, `local` is set to `TRUE` or `FALSE` based on the observed data sample size (i.e., sample size of the fitted model object) – if the sample size exceeds 10,000, `local` is set to `TRUE`, otherwise it is set to `FALSE`. This default behavior occurs because for point prediction the main computational burden of the big data approximation depends almost exclusively on the observed data sample size, not the number of predictions desired (which we feel is not intuitive at first glance). For block prediction (`block = TRUE`) the density of the prediction grid also matters, so `local` is additionally set to approximate the theoretical solution when `nrow(newdata)` exceeds 10,000 (see below). If `local` is `FALSE`, no big data approximation is implemented. If a list is provided, the following arguments detail the big data approximation:

- `method`: The big data approximation method. If `method = "all"`, all observations are used and `size` is ignored. If `method = "distance"`, the `size` data observations closest (in terms of Euclidean distance) to the observation requiring prediction are used. If `method = "covariance"`, the `size` data observations with the highest covariance with the observation requiring prediction are used. If random effects and partition factors are not used in estimation and the spatial covariance function is monotone decreasing, `"distance"` and `"covariance"` are equivalent. The default is `"covariance"`. Only used with models fit using `splm()` or `spglm()`.
- `size`: The number of data observations to use when `method` is `"distance"` or `"covariance"`. The default is 100. Only used with models fit using `splm()` or `spglm()`.
- `parallel`: If `TRUE`, parallel processing via the parallel package is automatically used. This can significantly speed up computations even when `method = "all"` (i.e., no big data approximation is used), as predictions are spread out over multiple cores. The default is `FALSE`.
- `ncores`: If `parallel = TRUE`, the number of cores to parallelize over. The default is the number of available cores on your machine.
- `byrow_threshold`: Only relevant when a random effect or partition factor is used and `method` is `"distance"` or `"covariance"`. In this scenario,

when the observed sample size times the number of predictions is less than `byrow_threshold`, computations are performed once for all predictions simultaneously (which is generally faster but uses more memory). Otherwise, computations are performed separately for each prediction, one row at a time (which is generally slower but uses less memory). The default is `10000^2`. Only used with models fit using `splm()` or `spglm()`.

When `local` is a list, at least one list element must be provided to initialize default arguments for the other list elements. If `local` is `TRUE`, defaults for `local` are chosen such that `local` is transformed into `list(size = 100, method = "covariance", parallel = FALSE)`.

If `block` is `TRUE`, `local` controls two separate big data approximations, one for the observed data and one for the prediction grid (`newdata`):

- `method` and `size` act on the observed data exactly as when `block` is `FALSE` (`method` takes `"all"`, `"covariance"`, or `"distance"`); the default `method` is `"covariance"` with `size` `4000`. This default `size` is much larger than when `block` is `FALSE` because block prediction averages covariances and explanatory variables before prediction.
- `method_new` controls the big data prediction grid density. `method_new = "basis"` (the default) approximates the block variance using a basis of `size_new` well-spread grid nodes. `method_new = "subset"` approximates the block mean and variance by subsampling `newdata` so that its size is only `size_new`. The default `size_new` is `4000`.
- `ordering` chooses the `size_new` nodes from `newdata` and takes the same values as the `ordering` argument of `sprnorm()`/`conditional()` under `approximation = "vecchia"` (`"maxmin"`, `"grts"`, `"random"`, `"none"`, `"middleout"`, `"outsidein"`, `"coordinate"`). The default is `"grts"`.
- `parallel` and `ncores` parallelize the variance calculation when `method_new = "basis"`.

When `local` is a list, at least one list element must be provided to initialize default arguments for the other list elements. If `local` is `TRUE`, defaults for `local` are chosen such that `local` is transformed into `list(method = "covariance", size = 4000, method_new = "basis", size_new = 4000, ordering = "grts")`.

`terms` — If `type` is `"terms"`, the type of terms to be returned, specified via either numeric position or name. The default is all terms are included.

`na.action` — Missing (`NA`) values in `newdata` will return an error and should be removed before proceeding.

`...` — Other arguments. Only used for models fit using `splmRF()` or `spautorRF()` where `...` indicates other arguments to `ranger::predict.ranger()`.

`dispersion` — The dispersion of assumed when computing the prediction standard errors for `spglm()` or `spgautor()` model objects when `family` is `"nbinomial"`, `"beta"`, `"Gamma"`, or `"inverse.gaussian"`. If omitted, the model object dispersion parameter is used.

`var_correct` — A logical indicating whether to return the corrected prediction variances when predicting via models fit using `spglm()` or `spgautor()`. The default is `TRUE`.

`delta` — A logical indicating whether to return delta method standard errors on the response scale when `se.fit = TRUE` and `type = "response"`. The default is `FALSE`.

`newdata_size` The `size` value for each observation in `newdata` used when predicting for the binomial family, with a default value of 1.

Details

For `splm` and `spautor` objects, the (empirical) best linear unbiased predictions (i.e., Kriging predictions) at each site are returned when `interval` is `"none"` or `"prediction"` alongside standard errors. Prediction intervals are also returned if `interval` is `"prediction"`. When `interval` is `"confidence"`, the estimated mean is returned alongside standard errors and confidence intervals for the mean. For `splm_list` and `spautor_list` objects, predictions and associated intervals and standard errors are returned for each list element.

For `splmRF` or `spautorRF` objects, random forest spatial residual model predictions are computed by combining the random forest prediction with the (empirical) best linear unbiased prediction for the residual. This approach is called random forest regression Kriging. For `splmRF_list` or `spautorRF` objects, predictions are returned for each list element.

For `decorrelate` objects, the spatial decorrelation transformation predictions recorrelated to the original scale. For `decorrelate_list` objects, predictions are returned for each list element.

Value

For `splm` or `spautor` objects, if `se.fit` is `FALSE`, `predict()` returns a vector of predictions or a matrix of predictions with column names `fit`, `lwr`, and `upr` if `interval` is `"confidence"` or `"prediction"`. If `se.fit` is `TRUE`, a list with the following components is returned:

- `fit`: vector or matrix as above
- `se.fit`: standard error of each fit

For `splm_list` or `spautor_list` objects, a list that contains relevant quantities for each list element.

For `splmRF` or `spautorRF` objects, a vector of predictions. For `splmRF_list` or `spautorRF_list` objects, a list that contains relevant quantities for each list element.

For `decorrelate` objects, a vector of predictions. For `decorrelate_list` objects, a list that contains relevant quantities for each list element.

References

Fox, E.W., Ver Hoef, J. M., & Olsen, A. R. (2020). Comparing spatial regression to random forests for large environmental data sets. *PloS one*, 15(3), e0229509.

Examples

```
spmod <- splm(sulfate ~ 1, data = sulfate, spcov_type = "exponential")
predict(spmod, sulfate_preds)
predict(spmod, sulfate_preds, interval = "prediction")
augment(spmod, newdata = sulfate_preds, interval = "prediction")

sulfate$var <- rnorm(NROW(sulfate)) # add noise variable
sulfate_preds$var <- rnorm(NROW(sulfate_preds)) # add noise variable
sprfmod <- splmRF(sulfate ~ var, data = sulfate, spcov_type = "exponential")
```

```
predict(sprfmod, sulfate_preds)


spgmod <- spglm(presence ~ elev * strat,
  family = "binomial",
  data = moose,
  spcov_type = "exponential"
)
predict(spgmod, moose_preds)
predict(spgmod, moose_preds, interval = "prediction")
augment(spgmod, newdata = moose_preds, interval = "prediction")

decorr <- decorrelate(log_cond ~ temp, data = lake, spcov_type = "exponential")
predict(decorr, newdata = lake_preds)
```

`print.spmodel` *Print values*

Description

Print fitted model objects and summaries.

Usage

```
## S3 method for class 'splm'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'spautor'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.splm'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'summary.spautor'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'anova.splm'
print(
```

```
  x,
  digits = max(getOption("digits") - 2L, 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'anova.spautor'
print(
  x,
  digits = max(getOption("digits") - 2L, 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'decorrelate'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'spglm'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'spgautor'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.spglm'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'summary.spgautor'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'anova.spglm'
print(
  x,
  digits = max(getOption("digits") - 2L, 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'anova.spgautor'
```

```
print(
  x,
  digits = max(getOption("digits") - 2L, 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'splmRF'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.splmRF'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'spautorRF'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.spautorRF'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

x	A fitted model object from splm(), spautor(), spglm(), spgautor(), splmRF(), spautorRF() or output from summary(x) or anova(x).
digits	The number of significant digits to use when printing.
...	Other arguments passed to or from other methods.
signif.stars	Logical. If TRUE, significance stars are printed for each coefficient

Value

Printed fitted model objects and summaries with formatting.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
print(spmod)
print(summary(spmod))
```

```
print(anova(spmod))

sulfate$var <- rnorm(NROW(sulfate)) # add noise variable
sprfmod <- splmRF(sulfate ~ var, data = sulfate, spcov_type = "exponential")
print(sprfmod)


sprfmod <- spautorRF(log_trend ~ stock, data = seal, spcov_type = "car")
print(sprfmod)
```

---

pseudoR2 *Compute a pseudo r-squared*

Description

Compute a pseudo r-squared for a fitted model object.

Usage

```
pseudoR2(object, ...)

## S3 method for class 'splm'
pseudoR2(object, adjust = FALSE, ...)

## S3 method for class 'spautor'
pseudoR2(object, adjust = FALSE, ...)

## S3 method for class 'spglm'
pseudoR2(object, adjust = FALSE, ...)

## S3 method for class 'spgautor'
pseudoR2(object, adjust = FALSE, ...)
```

Arguments

object	A fitted model object from splm(), spautor(), spglm(), or spgautor().
...	Other arguments. Not used (needed for generic consistency).
adjust	A logical indicating whether the pseudo r-squared should be adjusted to account for the number of explanatory variables. The default is FALSE.

Details

Several pseudo r-squared statistics exist for in the literature. We define this pseudo r-squared as one minus the ratio of the deviance of a full model relative to the deviance of a null (intercept only) model. This pseudo r-squared can be viewed as a generalization of the classical r-squared definition seen as one minus the ratio of error sums of squares from the full model relative to the error sums of squares from the null model. If adjusted, the adjustment is analogous to the the classical r-squared adjustment.

Value

The pseudo r-squared as a numeric vector.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
pseudoR2(spmod)
```

---

randcov_initial *Create a random effects covariance parameter initial object*

Description

Create a random effects (co)variance parameter initial object that specifies initial and/or known values to use while estimating random effect variances with modeling functions.

Usage

```
randcov_initial(..., known)
```

Arguments

`...`	Arguments to `randcov_params()`.
`known`	A character vector indicating which random effect variances are to be assumed known. The value `"given"` is shorthand for assuming all random effect variances given to `randcov_initial()` are assumed known.

Details

A random effect is specified as Zu, where Z is the random effects design matrix and u is the random effect. The covariance of Zu is $\sigma 2 Z Z^T$, where $\sigma 2$ is the random effect variance, and Z^T is the transpose of Z.

Value

A list with two elements: `initial` and `is_known`. `initial` is a named numeric vector indicating the random effect variances with specified initial and/or known values. `is_known` is a named logical vector indicating whether the random effect variances in `initial` are known or not.

Examples

```
randcov_initial(group = 1)
randcov_initial(group = 1, known = "group")
```

randcov_params *Create a random effects covariance parameter object*

Description

Create a random effects covariance parameter object for use with other functions.

Usage

```
randcov_params(..., nm)
```

Arguments

...	A named vector (or vectors) whose names represent the name of each random effect and whose values represent the variance of each random effect. If unnamed, nm is used to set names.
nm	A character vector of names to assign to ....

Details

Names of the random effects should match eligible names given to random in modeling functions. While with the random argument to these functions, an intercept is implicitly assumed, with randcov_params, an intercept must be explicitly specified. That is, while with random, x | group is shorthand for (1 | group) + (x | group), with randcov_params, x | group implies just x | group, which means that if 1 | group is also desired, it must be explicitly specified.

Value

A named numeric vector of random effect covariance parameters.

Examples

```
randcov_params(group = 1, subgroup = 2)
randcov_params(1, 2, nm = c("group", "subgroup"))
# same as
randcov_params("1 | group" = 1, "1 | subgroup" = 2)
```

recorrelate_newdata *Recorrelate Machine Learning Predictions*

Description

Recorrelate machine learning predictions according to a spatial decorrelation transformation.

Usage

```
recorrelate_newdata(object, ty_newdata)
```

Arguments

object	A decorrelate_newdata() object.
ty_newdata	Predictions from the machine learning model trained on the spatially decorrelated data and applied to spatially decorrelated newdata.

Value

A vector of predictions

Examples

```
params <- spcov_params("exponential", de = 1, ie = 0.2, range = 1e5)
decorr <- decorrelate_data(log_cond ~ temp, data = lake, spcov_params = params)
fit <- ranger::ranger(x = decorr$tX, y = decorr$ty)
decorr_newdata <- decorrelate_newdata(decorr, newdata = lake_preds)
rfpreds <- predict(fit, data = decorr_newdata$tX_newdata)$predictions
recorrelate_newdata(decorr_newdata, rfpreds)
```

residuals.spmodel *Extract fitted model residuals*

Description

Extract residuals from a fitted model object. resid is an alias.

Usage

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

## S3 method for class 'splm'
resid(object, type = "response", ...)

## S3 method for class 'splm'
```

```
rstandard(model, ...)

## S3 method for class 'spautor'
residuals(object, type = "response", ...)

## S3 method for class 'spautor'
resid(object, type = "response", ...)

## S3 method for class 'spautor'
rstandard(model, ...)

## S3 method for class 'spglm'
residuals(object, type = "deviance", ...)

## S3 method for class 'spglm'
resid(object, type = "deviance", ...)

## S3 method for class 'spglm'
rstandard(model, ...)

## S3 method for class 'spgautor'
residuals(object, type = "deviance", ...)

## S3 method for class 'spgautor'
resid(object, type = "deviance", ...)

## S3 method for class 'spgautor'
rstandard(model, ...)
```

Arguments

`object`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.
`type`	`"response"` for response residuals, `"pearson"` for Pearson residuals, or `"standardized"` for standardized residuals. For `splm()` and `spautor()` fitted model objects, the default is `"response"`. For `spglm()` and `spgautor()` fitted model objects, deviance residuals are also available (`"deviance"`) and are the default residual type.
`...`	Other arguments. Not used (needed for generic consistency).
`model`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.

Details

The response residuals are taken as the response minus the fitted values for the response: $y - X\hat{\beta}$. The Pearson residuals are the response residuals pre-multiplied by their inverse square root. The standardized residuals are Pearson residuals divided by the square root of one minus the leverage (hat) value. The standardized residuals are often used to check model assumptions, as they have mean zero and variance approximately one.

`rstandard()` is an alias for `residuals(model, type = "standardized")`.

Value

The residuals as a numeric vector.

See Also

augment.spmodel() cooks.distance.spmodel() hatvalues.spmodel() influence.spmodel()

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
residuals(spmod)
resid(spmod)
residuals(spmod, type = "pearson")
residuals(spmod, type = "standardized")
rstandard(spmod)
```

---

satterthwaite.splm *Compute Satterthwaite denominator degrees of freedom*

Description

Compute Satterthwaite denominator degrees of freedom t-based (rather than asymptotic z-based) fixed effect inference in small samples.

Usage

```
## S3 method for class 'splm'
satterthwaite(object, method, ...)

## S3 method for class 'spautor'
satterthwaite(object, method, ...)

satterthwaite(object, ...)
```

Arguments

object	A fitted model object from splm() or spautor().
method	The method by which to compute gradients. "numeric" for numerical differentiation and "closed" for closed form solutions. The default "closed" for "exponential", "gaussian", "spherical", "none", and "ie" spatial covariance functions (without anisotropy) and "numeric" otherwise.
...	Other arguments. Not used (needed for generic consistency).

Details

Satterthwaite degrees of freedom are generally more appropriate than asymptotic degrees of freedom for small samples. They can be computationally costly for sample sizes exceeding 500; however, for sample sizes this large, they Satterthwaite and asymptotic degrees of freedom should yield very similar inferences.

Value

A named numeric vector of Satterthwaite degrees of freedom for each fixed effect.

References

Rencher, Alvin C. and Schaalje, G. Bruce (2008). Linear Models in Statistics, Second Edition. John Wiley & Sons.

See Also

`splm()` `spautor()` `anova.spmodel()`

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y, estmethod = "reml"
)
satterthwaite(spmod)
```

---

seal

Estimated harbor-seal trends from abundance data in southeast Alaska, USA

Description

Estimated harbor-seal trends from abundance data in southeast Alaska, USA.

Usage

```
seal
```

Format

A `sf` object with 149 rows and 3 columns:

- log_trend: The log of the estimated harbor-seal trends from abundance data.
- trend: The estimated harbor-seal trends from abundance data.

- stock: A seal stock factor with two levels: 8 and 10. The factor levels indicate the type of seal stock (i.e., type of seal). Stocks 8 and 10 are two distinct stocks (out of 13 total stocks) in southeast Alaska.
- geometry: `POLYGON` geometry representing polygons in an Alaska Albers projection (EPSG: 3338).

Source

These data were collected by the Polar Ecosystem Program of the Marine Mammal Laboratory of the Alaska Fisheries Science Center of NOAA Fisheries. The data were used in the publication listed in References.

References

Ver Hoef, J.M., Peterson, E. E., Hooten, M. B., Hanks, E. M., and Fortin, M.-J. 2018. Spatial Autoregressive Models for Statistical Inference from Ecological Data. Ecological Monographs, 88: 36-59. DOI: 10.1002/ecm.1283.

`spautor` *Fit spatial autoregressive models*

Description

Fit spatial linear models for areal data (i.e., spatial autoregressive models) using a variety of estimation methods, allowing for random effects, partition factors, and row standardization.

Usage

```
spautor(
  formula,
  data,
  spcov_type,
  spcov_initial,
  estmethod = "reml",
  random,
  randcov_initial,
  partition_factor,
  W,
  row_st = TRUE,
  M,
  range_positive = TRUE,
  cutoff,
  ddf,
  ...
)
```

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms, separated by + operators, on the right.
data	A data frame or sf object that contains the variables in fixed, random, and partition_factor, as well as potentially geographical information.
spcov_type	The spatial covariance type. Available options include "car", "sar", and "none". Parameterizations of each spatial covariance type are available in Details. When spcov_type is specified, relevant spatial covariance parameters are assumed unknown, requiring estimation. spcov_type is not required (and is ignored) if spcov_initial is provided. Multiple values can be provided in a character vector. Then spautor() is called iteratively for each element and a list is returned for each model fit. The default for spcov_type is "car". When spcov_type is "none", splm() is called.
spcov_initial	An object from spcov_initial() specifying initial and/or known values for the spatial covariance parameters. Not required if spcov_type is provided. Multiple spcov_initial() objects can be provided in a list. Then spautor() is called iteratively for each element and a list is returned for each model fit.
estmethod	The estimation method. Available options include "reml" for restricted maximum likelihood and "ml" for maximum likelihood The default is "reml".
random	A one-sided linear formula describing the random effect structure of the model. Terms are specified to the right of the ~ operator. Each term has the structure x1 + ... + xn | g1/.../gm, where x1 + ... + xn specifies the model for the random effects and g1/.../gm is the grouping structure. Separate terms are separated by + and must generally be wrapped in parentheses. Random intercepts are added to each model implicitly when at least one other variable is defined. If a random intercept is not desired, this must be explicitly defined (e.g., x1 + ... + xn - 1 | g1/.../gm). If only a random intercept is desired for a grouping structure, the random intercept must be specified as 1 | g1/.../gm. Note that g1/.../gm is shorthand for (1 | g1/.../gm). If only random intercepts are desired and the shorthand notation is used, parentheses can be omitted.
randcov_initial	An optional object specifying initial and/or known values for the random effect variances.
partition_factor	A one-sided linear formula with a single term specifying the partition factor. The partition factor assumes observations from different levels of the partition factor are uncorrelated.
W	Weight matrix specifying the neighboring structure used. Not required if data is an sf object with POLYGON geometry, as W is calculated internally using queen contiguity. If calculated internally, W is computed using sf::st_intersects(). Also not required if data is an sf object with POINT geometry as long as cutoff is specified.
row_st	A logical indicating whether row standardization be performed on W. The default is TRUE.

M	M matrix satisfying the car symmetry condition. The car symmetry condition states that $(I - range * W)^{-1}M$ is symmetric, where I is an identity matrix, $range$ is a constant that controls the spatial dependence, W is the weights matrix, and $^{-1}$ represents the inverse operator. M is required for car models when W is provided and row_st is FALSE. When M, is required, the default is the identity matrix. M must be diagonal or given as a vector or one-column matrix assumed to be the diagonal.
range_positive	Whether the range should be constrained to be positive. The default is TRUE.
cutoff	The numeric, distance-based cutoff used to determine W when W is not specified. For an sf object with POINT geometry, two locations are considered neighbors if the distance between them is less than or equal to cutoff.
ddf	The denominator degrees of freedom to be used for eventual hypothesis testing (e.g., confint(), summary(), or tidy() calls on the fitted model object). "asymptotic" assumes infinite degrees of freedom, implying z-tests. "satterthwaite" impelments Satterthwaite degrees of freeom, implying t-tests that are generally more appropriate for small samples. The default is "satterthwaite" when the sample size is less than or equal to 500 and "asymptotic" otherwise.
...	Other arguments to stats::optim().

Details

The spatial linear model for areal data (i.e., spatial autoregressive model) can be written as $y = X\beta + \tau + \epsilon$, where X is the fixed effects design matrix, β are the fixed effects, τ is random error that is spatially dependent, and ϵ is random error that is spatially independent. Together, τ and ϵ are modeled using a spatial covariance function, expressed as $de * R + ie * I$, where de is the dependent error variance, R is a matrix that controls the spatial dependence structure among observations, ie is the independent error variance, and I is an identity matrix. Note that de and ie must be non-negative while $range$ must be between the reciprocal of the maximum eigenvalue of W and the reciprocal of the minimum eigenvalue of W.

spcov_type Details: Parametric forms for R are given below:

- car: $(I - range * W)^{-1}M$, weights matrix W, symmetry condition matrix M
- sar: $[(I - range * W)(I - range * W)^T]^{-1}$, weights matrix W, ^{T} indicates matrix transpose
- none: 0

If there are observations with no neighbors, they are given a unique variance parameter called extra, which must be non-negative.

estmethod Details: The various estimation methods are

- reml: Maximize the restricted log-likelihood.
- ml: Maximize the log-likelihood.

By default, all spatial covariance parameters except ie as well as all random effect variance parameters are assumed unknown, requiring estimation. ie is assumed zero and known by default (in contrast to models fit using splm(), where ie is assumed unknown by default). To change this default behavior, specify spcov_initial (an NA value for ie in spcov_initial to assume ie is unknown, requiring estimation).

`random` Details: If random effects are used, the model can be written as $y = X\beta + Z1u1 + ...Zjuj + \tau + \epsilon$, where each Z is a random effects design matrix and each u is a random effect.

`partition_factor` Details: The partition factor can be represented in matrix form as P, where elements of P equal one for observations in the same level of the partition factor and zero otherwise. The covariance matrix involving only the spatial and random effects components is then multiplied element-wise (Hadmard product) by P, yielding the final covariance matrix.

Observations with `NA` response values are removed for model fitting, but their values can be predicted afterwards by running `predict(object)`. This is the only way to perform prediction for `spautor()` models (i.e., the prediction locations must be known prior to estimation).

Value

A list with many elements that store information about the fitted model object. If `spcov_type` or `spcov_initial` are length one, the list has class `spautor`. Many generic functions that summarize model fit are available for `spautor` objects, including `AIC`, `AICc`, `anova`, `augment`, `BIC`, `coef`, `cooks.distance`, `covmatrix`, `deviance`, `fitted`, `formula`, `glance`, `glances`, `hatvalues`, `influence`, `labels`, `logLik`, `loocv`, `model.frame`, `model.matrix`, `plot`, `predict`, `print`, `pseudoR2`, `satterthwaite`, `summary`, `terms`, `tidy`, `update`, `varcomp`, and `vcov`. If `spcov_type` or `spcov_initial` are length greater than one, the list has class `spautor_list` and each element in the list has class `spautor`. `glances` can be used to summarize `spautor_list` objects, and the aforementioned `spautor` generics can be used on each individual list element (model fit).

Note

This function does not perform any internal scaling. If optimization is not stable due to large extremely large variances, scale relevant variables so they have variance 1 before optimization.

Examples

```
spmod <- spautor(log_trend ~ 1, data = seal, spcov_type = "car")
summary(spmod)
```

---

spautorRF *Fit random forest spatial residual models*

Description

Fit random forest residual spatial linear models for areal data (i.e., spatial autoregressive models) using random forest to fit the mean and a spatial linear model to fit the residuals. The spatial linear model fit to the residuals can incorporate a variety of estimation methods, allowing for random effects, partition factors, and row standardization.

Usage

```
spautorRF(formula, data, ...)
```

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms on the right, separated by + operators.
data	A data frame or sf object object that contains the variables in fixed, random, and partition_factor as well as geographical information. If an sf object is provided with POINT geometries, the x-coordinates and y-coordinates are used directly. If an sf object is provided with POLYGON geometries, the x-coordinates and y-coordinates are taken as the centroids of each polygon.
...	Additional named arguments to ranger::ranger() or spautor().

Details

The random forest residual spatial linear model is described by Fox et al. (2020). A random forest model is fit to the mean portion of the model specified by formula using ranger::ranger(). Residuals are computed and used as the response variable in an intercept-only spatial linear model fit using spautor(). This model object is intended for use with predict() to perform prediction, also called random forest regression Kriging.

Value

A list with several elements to be used with predict(). These elements include the function call (named call), the random forest object fit to the mean (named ranger), the spatial linear model object fit to the residuals (named spautor or spautor_list), and an object can contain data for locations at which to predict (called newdata). The newdata object contains the set of observations in data whose response variable is NA. If spcov_type or spcov_initial (which are passed to spautor()) are length one, the list has class spautorRF and the spatial linear model object fit to the residuals is called spautor, which has class spautor. If spcov_type or spcov_initial are length greater than one, the list has class spautorRF_list and the spatial linear model object fit to the residuals is called spautor_list, which has class spautor_list. and contains several objects, each with class spautor.

References

Fox, E.W., Ver Hoef, J. M., & Olsen, A. R. (2020). Comparing spatial regression to random forests for large environmental data sets. *PloS one*, 15(3), e0229509.

Examples

```
sprfmod <- spautorRF(log_trend ~ stock, data = seal, spcov_type = "car")
predict(sprfmod)
```

spcov_initial *Create a spatial covariance parameter initial object*

Description

Create a spatial covariance parameter initial object that specifies initial and/or known values to use while estimating spatial covariance parameters with splm(), spglm(), spautor(), or spgautor().

Usage

```
spcov_initial(spcov_type, de, ie, range, extra, rotate, scale, known)
```

Arguments

spcov_type	The spatial covariance function type. Available options include "exponential", "spherical", "gaussian", "triangular", "circular", "cubic", "pentaspherical", "cosine", "wave", "jbessel", "gravity", "rquad", "magnetic", "matern", "cauchy", "pexponential", "car", "sar", and "none".
de	The spatially dependent (correlated) random error variance. Commonly referred to as a partial sill.
ie	The spatially independent (uncorrelated) random error variance. Commonly referred to as a nugget.
range	The correlation parameter.
extra	An extra covariance parameter used when spcov_type is "matern", "cauchy", "pexponential", "car", or "sar".
rotate	Anisotropy rotation parameter (from 0 to π radians). Not used if spcov_type is "car" or "sar".
scale	Anisotropy scale parameter (from 0 to 1). Not used if spcov_type is "car" or "sar".
known	A character vector indicating which spatial covariance parameters are to be assumed known. The value "given" is shorthand for assuming all spatial covariance parameters given to spcov_initial() are assumed known.

Details

The spcov_initial list is later passed to splm(), spglm(), spautor(), or spgautor(). NA values can be given for ie, rotate, and scale, which lets these functions find initial values for parameters that are sometimes otherwise assumed known (e.g., rotate and scale with splm() and spglm() and ie with spautor() and spgautor()). The spatial covariance functions can be generally expressed as $de * R + ie * I$, where de is de above, R is a matrix that controls the spatial dependence structure among observations, h, ie is ie above, and I is and identity matrix. Note that de and ie must be non-negative while $range$ must be positive, except when spcov_type is car or sar, in which case $range$ must be between the reciprocal of the maximum eigenvalue of W and the reciprocal of the minimum eigenvalue of W. Parametric forms for R are given below, where $\eta = h/range$:

- exponential: $exp(-\eta)$
- spherical: $(1 - 1.5\eta + 0.5\eta^3) * I(h <= range)$
- gaussian: $exp(-\eta^2)$
- triangular: $(1 - \eta) * I(h <= range)$
- circular: $(1 - (2/\pi) * (m * sqrt(1 - m^2) + sin^{-1}(m))) * I(h <= range), m = min(\eta, 1)$
- cubic: $(1 - 7\eta^2 + 8.75\eta^3 - 3.5\eta^5 + 0.75\eta^7) * I(h <= range)$
- pentaspherical: $(1 - 1.875\eta + 1.25\eta^3 - 0.375\eta^5) * I(h <= range)$
- cosine: $cos(\eta)$
- wave: $sin(\eta)/\eta * I(h > 0) + I(h = 0)$
- jbessel: $Bj(h * range)$, Bj is Bessel-J function
- gravity: $(1 + \eta^2)^{-0.5}$
- rquad: $(1 + \eta^2)^{-1}$
- magnetic: $(1 + \eta^2)^{-1.5}$
- matern: $2^{1-extra}/\Gamma(extra) * \alpha^{extra} * Bk(\alpha, extra)$, $\alpha = (2extra)^{0.5} * \eta$, Bk is Bessel-K function wit order $1/5 \leq extra \leq 5$
- cauchy: $(1 + \eta^2)^{-extra}$, $extra > 0$
- pexponential: $exp(h^{extra}/range)$, $0 < extra \leq 2$
- car: $(I - range * W)^{-1} * M$, weights matrix W, symmetry condition matrix M, observations with no neighbors are given a unique variance parameter called $extra$, $extra \geq 0$.
- sar: $[(I - range * W)(I - range * W)^T]^{-1}$, weights matrix W, ^{T} indicates matrix transpose, observations with no neighbors are given a unique variance parameter called $extra$, $extra \geq 0$.
- none: 0

All spatial covariance functions are valid in one spatial dimension. All spatial covariance functions except `triangular` and `cosine` are valid in two dimensions. An alias for `none` is `ie`.

When the spatial covariance function is `car` or `sar`, `extra` represents the variance parameter for the observations in `W` without at least one neighbor (other than itself) – these are called unconnected observations. `extra` is only used if there is at least one unconnected observation.

Value

A list with two elements: `initial` and `is_known`. `initial` is a named numeric vector indicating the spatial covariance parameters with specified initial and/or known values. `is_known` is a named numeric vector indicating whether the spatial covariance parameters in `initial` are known or not. The class of the list matches the value given to the `spcov_type` argument.

Examples

```
# known de value 1 and initial range value 0.4
spcov_initial("exponential", de = 1, range = 0.4, known = c("de"))
# known ie value 0 and known range value 1
spcov_initial("gaussian", ie = 0, range = 1, known = c("given"))
# ie given NA
spcov_initial("car", ie = NA)
```

spcov_params *Create a spatial covariance parameter object*

Description

Create a spatial covariance parameter object for use with other functions.

Usage

```
spcov_params(spcov_type, de, ie, range, extra, rotate = 0, scale = 1)
```

Arguments

spcov_type	The spatial covariance function type. Available options include "exponential", "spherical", "gaussian", "triangular", "circular", "cubic", "pentaspherical", "cosine", "wave", "jbessel", "gravity", "rquad", "magnetic", "matern", "cauchy", "pexponential", "car", "sar", "none", and "ie" (an alias for "none").
de	The spatially dependent (correlated) random error variance. Commonly referred to as a partial sill.
ie	The spatially independent (uncorrelated) random error variance. Commonly referred to as a nugget.
range	The correlation parameter.
extra	An extra covariance parameter used when spcov_type is "matern", "cauchy", "pexponential", "car", or "sar".
rotate	Anisotropy rotation parameter (from 0 to π radians). A value of 0 (the default) implies no rotation. Not used if spcov_type is "car" or "sar".
scale	Anisotropy scale parameter (from 0 to 1). A value of 1 (the default) implies no scaling. Not used if spcov_type is "car" or "sar".

Details

Generally, all arguments to spcov_params must be specified, though default arguments are often chosen based on spcov_type. When spcov_type is car or sar, ie is assumed to be 0 unless specified otherwise. For full parameterizations of all spatial covariance functions, see spcov_initial().

Value

A named numeric vector of spatial covariance parameters with class spcov_type.

Examples

```
spcov_params("exponential", de = 1, ie = 1, range = 1)
```

spgautor *Fit spatial generalized autoregressive models*

Description

Fit spatial generalized linear models for areal data (i.e., spatial generalized autoregressive models) using a variety of estimation methods, allowing for random effects, partition factors, and row standardization.

Usage

```
spgautor(
  formula,
  family,
  data,
  spcov_type,
  spcov_initial,
  dispersion_initial,
  estmethod = "reml",
  random,
  randcov_initial,
  partition_factor,
  W,
  row_st = TRUE,
  M,
  range_positive = TRUE,
  cutoff,
  ...
)
```

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms, separated by + operators, on the right.
family	The generalized linear model family describing the distribution of the response variable to be used. Available options "poisson", "nbinomial", "binomial", "beta", "Gamma", and "inverse.gaussian". Can be quoted or unquoted. Note that the family argument only takes a single value, rather than the list structure used by stats::glm. See Details for more.
data	A data frame or sf object that contains the variables in fixed, random, and partition_factor, as well as potentially geographical information.
spcov_type	The spatial covariance type. Available options include "car", "sar", "none", and "ie". Parameterizations of each spatial covariance type are available in Details. When spcov_type is specified, relevant spatial covariance parameters are assumed unknown, requiring estimation. spcov_type is not required (and

is ignored) if spcov_initial is provided. Multiple values can be provided in a character vector. Then spgautor() is called iteratively for each element and a list is returned for each model fit. The default for spcov_type is "car". When spcov_type is "none" or "ie", splm() is called.

spcov_initial An object from spcov_initial() specifying initial and/or known values for the spatial covariance parameters. Not required if spcov_type is provided. Multiple spcov_initial() objects can be provided in a list. Then spgautor() is called iteratively for each element and a list is returned for each model fit.

dispersion_initial
An object from dispersion_initial() specifying initial and/or known values for the dispersion parameter for the "nbinomial", "beta", "Gamma", and "inverse.gaussian" families. family is ignored if dispersion_initial is provided.

estmethod The estimation method. Available options include "reml" for restricted maximum likelihood and "ml" for maximum likelihood The default is "reml".

random A one-sided linear formula describing the random effect structure of the model. Terms are specified to the right of the ~ operator. Each term has the structure x1 + ... + xn | g1/.../gm, where x1 + ... + xn specifies the model for the random effects and g1/.../gm is the grouping structure. Separate terms are separated by + and must generally be wrapped in parentheses. Random intercepts are added to each model implicitly when at least one other variable is defined. If a random intercept is not desired, this must be explicitly defined (e.g., x1 + ... + xn - 1 | g1/.../gm). If only a random intercept is desired for a grouping structure, the random intercept must be specified as 1 | g1/.../gm. Note that g1/.../gm is shorthand for (1 | g1/.../gm). If only random intercepts are desired and the shorthand notation is used, parentheses can be omitted.

randcov_initial
An optional object specifying initial and/or known values for the random effect variances.

partition_factor
A one-sided linear formula with a single term specifying the partition factor. The partition factor assumes observations from different levels of the partition factor are uncorrelated.

W Weight matrix specifying the neighboring structure used. Not required if data is an sf object wtih POLYGON geometry, as W is calculated internally using queen contiguity. If calculated internally, W is computed using sf::st_intersects(). Also not required if data is an sf object with POINT geometry as long as cutoff is specified.

row_st A logical indicating whether row standardization be performed on W. The default is TRUE.

M M matrix satisfying the car symmetry condition. The car symmetry condition states that $(I - range * W)^{-1}M$ is symmetric, where I is an identity matrix, $range$ is a constant that controls the spatial dependence, W is the weights matrix, and $^{-1}$ represents the inverse operator. M is required for car models when W is provided and row_st is FALSE. When M, is required, the default is the identity matrix. M must be diagonal or given as a vector or one-column matrix assumed to be the diagonal.

range_positive	Whether the range should be constrained to be positive. The default is TRUE.
cutoff	The numeric, distance-based cutoff used to determine W when W is not specified. For an sf object with POINT geometry, two locations are considered neighbors if the distance between them is less than or equal to cutoff.
...	Other arguments to stats::optim().

Details

The spatial generalized linear model for areal data (i.e., spatial generalized autoregressive model) can be written as $g(\mu) = \eta = X\beta + \tau + \epsilon$, where μ is the expectation of the response (y) given the random errors, $g(.)$ is called a link function which links together the μ and η, X is the fixed effects design matrix, β are the fixed effects, τ is random error that is spatially dependent, and ϵ is random error that is spatially independent.

There are six generalized linear model families available: poisson assumes y is a Poisson random variable nbinomial assumes y is a negative binomial random variable, binomial assumes y is a binomial random variable, beta assumes y is a beta random variable, Gamma assumes y is a gamma random variable, and inverse.gaussian assumes y is an inverse Gaussian random variable.

The supports for y for each family are given below:

- family: support of y
- poisson: $0 \leq y$; y an integer
- nbinomial: $0 \leq y$; y an integer
- binomial: $0 \leq y$; y an integer
- beta: $0 < y < 1$
- Gamma: $0 < y$
- inverse.gaussian: $0 < y$

The generalized linear model families and the parameterizations of their link functions are given below:

- family: link function
- poisson: $g(\mu) = log(\eta)$ (log link)
- nbinomial: $g(\mu) = log(\eta)$ (log link)
- binomial: $g(\mu) = log(\eta/(1-\eta))$ (logit link)
- beta: $g(\mu) = log(\eta/(1-\eta))$ (logit link)
- Gamma: $g(\mu) = log(\eta)$ (log link)
- inverse.gaussian: $g(\mu) = log(\eta)$ (log link)

The variance function of an individual y (given μ) for each generalized linear model family is given below:

- family: $Var(y)$
- poisson: $\mu\phi$
- nbinomial: $\mu + \mu^2/\phi$
- binomial: $n\mu(1-\mu)\phi$

- beta: $\mu(1-\mu)/(1+\phi)$
- Gamma: μ^2/ϕ
- inverse.gaussian: μ^2/ϕ

The parameter ϕ is a dispersion parameter that influences $Var(y)$. For the `poisson` and `binomial` families, ϕ is always one. Note that this inverse Gaussian parameterization is different than a standard inverse Gaussian parameterization, which has variance μ^3/λ. Setting $\phi = \lambda/\mu$ yields our parameterization, which is preferred for computational stability. Also note that the dispersion parameter is often defined in the literature as $V(\mu)\phi$, where $V(\mu)$ is the variance function of the mean. We do not use this parameterization, which is important to recognize while interpreting dispersion parameter estimates. For more on generalized linear model constructions, see McCullagh and Nelder (1989).

Together, τ and ϵ are modeled using a spatial covariance function, expressed as $de * R + ie * I$, where de is the dependent error variance, R is a matrix that controls the spatial dependence structure among observations, ie is the independent error variance, and I is an identity matrix. Note that de and ie must be non-negative while $range$ must be between the reciprocal of the maximum eigenvalue of `W` and the reciprocal of the minimum eigenvalue of `W`. Recall that τ and ϵ are modeled on the link scale, not the inverse link (response) scale. Random effects are also modeled on the link scale.

`spcov_type` Details: Parametric forms for R are given below:

- car: $(I - range * W)^{-1}M$, weights matrix W, symmetry condition matrix M
- sar: $[(I - range * W)(I - range * W)^T]^{-1}$, weights matrix W, ^{T} indicates matrix transpose
- none: 0
- ie: 0 (see `spglm()` for differences compared to none)

If there are observations with no neighbors, they are given a unique variance parameter called `extra`, which must be non-negative.

`estmethod` Details: The various estimation methods are

- `reml`: Maximize the restricted log-likelihood.
- `ml`: Maximize the log-likelihood.

Note that the likelihood being optimized is obtained using the Laplace approximation.

By default, all spatial covariance parameters except `ie` as well as all random effect variance parameters are assumed unknown, requiring estimation. `ie` is assumed zero and known by default (in contrast to models fit using `spglm()`, where `ie` is assumed unknown by default). To change this default behavior, specify `spcov_initial` (an `NA` value for `ie` in `spcov_initial` to assume `ie` is unknown, requiring estimation).

`random` Details: If random effects are used, the model can be written as $y = X\beta + Z1u1 + ...Zjuj + \tau + \epsilon$, where each Z is a random effects design matrix and each u is a random effect.

`partition_factor` Details: The partition factor can be represented in matrix form as P, where elements of P equal one for observations in the same level of the partition factor and zero otherwise. The covariance matrix involving only the spatial and random effects components is then multiplied element-wise (Hadmard product) by P, yielding the final covariance matrix.

Observations with `NA` response values are removed for model fitting, but their values can be predicted afterwards by running `predict(object)`. This is the only way to perform prediction for `spgautor()` models (i.e., the prediction locations must be known prior to estimation).

Value

A list with many elements that store information about the fitted model object. If spcov_type or spcov_initial are length one, the list has class spgautor. Many generic functions that summarize model fit are available for spgautor objects, including AIC, AICc, anova, augment, AUROC, BIC, coef, cooks.distance, covmatrix, deviance, fitted, formula, glance, glances, hatvalues, influence, labels, logLik, loocv, model.frame, model.matrix, plot, predict, print, pseudoR2, summary, terms, tidy, update, varcomp, and vcov. If spcov_type or spcov_initial are length greater than one, the list has class spgautor_list and each element in the list has class spgautor. glances can be used to summarize spgautor_list objects, and the aforementioned spgautor generics can be used on each individual list element (model fit).

Note

This function does not perform any internal scaling. If optimization is not stable due to large extremely large variances, scale relevant variables so they have variance 1 before optimization.

References

McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models*. London: Chapman and Hall.

Examples

```
spgmod <- spgautor(I(log_trend^2) ~ 1, family = "Gamma", data = seal, spcov_type = "car")
summary(spgmod)
```

---

spglm *Fit spatial generalized linear models*

Description

Fit spatial generalized linear models for point-referenced data (i.e., generalized geostatistical models) using a variety of estimation methods, allowing for random effects, anisotropy, partition factors, and big data methods.

Usage

```
spglm(
  formula,
  family,
  data,
  spcov_type,
  xcoord,
  ycoord,
  spcov_initial,
  dispersion_initial,
  estmethod = "reml",
  anisotropy = FALSE,
```

```
  random,
  randcov_initial,
  partition_factor,
  local,
  range_constrain,
  ...
)
```

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms on the right, separated by + operators.
family	The generalized linear model family describing the distribution of the response variable to be used. Available options "poisson", "nbinomial", "binomial", "beta", "Gamma", and "inverse.gaussian". Can be quoted or unquoted. Note that the family argument only takes a single value, rather than the list structure used by stats::glm. See Details for more.
data	A data frame or sf object object that contains the variables in fixed, random, and partition_factor as well as geographical information. If an sf object is provided with POINT geometries, the x-coordinates and y-coordinates are used directly. If an sf object is provided with POLYGON geometries, the x-coordinates and y-coordinates are taken as the centroids of each polygon.
spcov_type	The spatial covariance type. Available options include "exponential", "spherical", "gaussian", "triangular", "circular", "cubic", "pentaspherical", "cosine", "wave", "jbessel", "gravity", "rquad", "magnetic", "matern", "cauchy", "pexponential", "ie", and "none". Parameterizations of each spatial covariance type are available in Details. Multiple spatial covariance types can be provided as a character vector, and then spglm() is called iteratively for each element and a list is returned for each model fit. The default for spcov_type is "exponential". When spcov_type is specified, all unknown spatial covariance parameters are estimated. spcov_type is ignored if spcov_initial is provided.
xcoord	The name of the column in data representing the x-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
ycoord	The name of the column in data representing the y-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
spcov_initial	An object from spcov_initial() specifying initial and/or known values for the spatial covariance parameters. Multiple spcov_initial() objects can be provided in a list. Then spglm() is called iteratively for each element and a list is returned for each model fit.
dispersion_initial	An object from dispersion_initial() specifying initial and/or known values for the dispersion parameter for the "nbinomial", "beta", "Gamma", and "inverse.gaussian" families. family is ignored if dispersion_initial is provided.

estmethod The estimation method. Available options include "reml" for restricted maximum likelihood and "ml" for maximum likelihood. The default is "reml".

anisotropy A logical indicating whether (geometric) anisotropy should be modeled. Not required if spcov_initial is provided with 1) rotate assumed unknown or assumed known and non-zero or 2) scale assumed unknown or assumed known and less than one. When anisotropy is TRUE, computational times can significantly increase. The default is FALSE.

random A one-sided linear formula describing the random effect structure of the model. Terms are specified to the right of the ~ operator. Each term has the structure x1 + ... + xn | g1/.../gm, where x1 + ... + xn specifies the model for the random effects and g1/.../gm is the grouping structure. Separate terms are separated by + and must generally be wrapped in parentheses. Random intercepts are added to each model implicitly when at least one other variable is defined. If a random intercept is not desired, this must be explicitly defined (e.g., x1 + ... + xn - 1 | g1/.../gm). If only a random intercept is desired for a grouping structure, the random intercept must be specified as 1 | g1/.../gm. Note that g1/.../gm is shorthand for (1 | g1/.../gm). If only random intercepts are desired and the shorthand notation is used, parentheses can be omitted.

randcov_initial
An optional object specifying initial and/or known values for the random effect variances.

partition_factor
A one-sided linear formula with a single term specifying the partition factor. The partition factor assumes observations from different levels of the partition factor are uncorrelated.

local An optional logical or list controlling the big data approximation. If omitted, local is set to TRUE or FALSE based on the sample size (the number of non-missing observations in data) – if the sample size exceeds 3,000, local is set to TRUE. Otherwise it is set to FALSE. If FALSE, no big data approximation is implemented. If a list is provided, the following arguments detail the big data approximation:

- index: The group indexes. Observations in different levels of index are assumed to be uncorrelated for the purposes of estimation. If index is not provided, it is determined by specifying method and either size or groups.
- method: The big data approximation method used to determine index. Ignored if index is provided. If method = "random", observations are randomly assigned to index based on size. If method = "kmeans", observations assigned to index based on k-means clustering on the coordinates with groups clusters. The default is "kmeans". Note that both methods have a random component, which means that you may get different results from separate model fitting calls. To ensure consistent results, specify index or set a seed via base::set.seed().
- size: The number of observations in each index group when method is "random". If the number of observations is not divisible by size, some levels get size - 1 observations. The default is 100.
- groups: The number of index groups. If method is "random", size is $ceiling(n/groups)$, where n is the sample size. Automatically determined

if `size` is specified. If `method` is `"kmeans"`, `groups` is the number of clusters.

- `var_adjust`: The approach for adjusting the variance-covariance matrix of the fixed effects. `"none"` for no adjustment, `"theoretical"` for the theoretically-correct adjustment, `"pooled"` for the pooled adjustment, and `"empirical"` for the empirical adjustment. The default is `"theoretical"` for samples sizes up to 100,000 and `"none"` for samples sizes exceeding 100,000.
- `parallel`: If `TRUE`, parallel processing via the parallel package is automatically used. The default is `FALSE`.
- `ncores`: If `parallel = TRUE`, the number of cores to parallelize over. The default is the number of available cores on your machine.

When `local` is a list, at least one list element must be provided to initialize default arguments for the other list elements. If `local` is `TRUE`, defaults for `local` are chosen such that `local` is transformed into `list(size = 100, method = "kmeans", var_adjust = "theoretical", parallel = FALSE)`.

`range_constrain`

An optional logical that indicates whether the range should be constrained to enhance numerical stability. If `range_constrain = TRUE`, the maximum possible range value is 4 times the maximum distance in the domain. If `range_constrain = FALSE`, then maximum possible range is unbounded. The default is `FALSE`. Note that if `range_constrain = TRUE` and the value of `range` in `spcov_initial` is larger than `range_constrain`, then `range_constrain` is set to `FALSE`.

`...` Other arguments to `esv()` or `stats::optim()`.

Details

The spatial generalized linear model for point-referenced data (i.e., generalized geostatistical model) can be written as $g(\mu) = \eta = X\beta + \tau + \epsilon$, where μ is the expectation of the response (y) given the random errors, $g(.)$ is called a link function which links together the μ and η, X is the fixed effects design matrix, β are the fixed effects, τ is random error that is spatially dependent, and ϵ is random error that is spatially independent.

There are six generalized linear model families available: `poisson` assumes y is a Poisson random variable `nbinomial` assumes y is a negative binomial random variable, `binomial` assumes y is a binomial random variable, `beta` assumes y is a beta random variable, `Gamma` assumes y is a gamma random variable, and `inverse.gaussian` assumes y is an inverse Gaussian random variable.

The supports for y for each family are given below:

- family: support of y
- poisson: $0 \leq y$; y an integer
- nbinomial: $0 \leq y$; y an integer
- binomial: $0 \leq y$; y an integer
- beta: $0 < y < 1$
- Gamma: $0 < y$
- inverse.gaussian: $0 < y$

The generalized linear model families and the parameterizations of their link functions are given below:

- family: link function
- poisson: $g(\mu) = log(\eta)$ (log link)
- nbinomial: $g(\mu) = log(\eta)$ (log link)
- binomial: $g(\mu) = log(\eta/(1-\eta))$ (logit link)
- beta: $g(\mu) = log(\eta/(1-\eta))$ (logit link)
- Gamma: $g(\mu) = log(\eta)$ (log link)
- inverse.gaussian: $g(\mu) = log(\eta)$ (log link)

The variance function of an individual y (given μ) for each generalized linear model family is given below:

- family: $Var(y)$
- poisson: $\mu\phi$
- nbinomial: $\mu + \mu^2/\phi$
- binomial: $n\mu(1-\mu)\phi$
- beta: $\mu(1-\mu)/(1+\phi)$
- Gamma: μ^2/ϕ
- inverse.gaussian: μ^2/ϕ

The parameter ϕ is a dispersion parameter that influences $Var(y)$. For the `poisson` and `binomial` families, ϕ is always one. Note that this inverse Gaussian parameterization is different than a standard inverse Gaussian parameterization, which has variance μ^3/λ. Setting $\phi = \lambda/\mu$ yields our parameterization, which is preferred for computational stability. Also note that the dispersion parameter is often defined in the literature as $V(\mu)\phi$, where $V(\mu)$ is the variance function of the mean. We do not use this parameterization, which is important to recognize while interpreting dispersion estimates. For more on generalized linear model constructions, see McCullagh and Nelder (1989).

Together, τ and ϵ are modeled using a spatial covariance function, expressed as $de*R+ie*I$, where de is the dependent error variance, R is a correlation matrix that controls the spatial dependence structure among observations, ie is the independent error variance, and I is an identity matrix. Recall that τ and ϵ are modeled on the link scale, not the inverse link (response) scale. Random effects are also modeled on the link scale.

`spcov_type` Details: Parametric forms for R are given below, where $\eta = h/range$ for h distance between observations:

- exponential: $exp(-\eta)$
- spherical: $(1 - 1.5\eta + 0.5\eta^3) * I(h <= range)$
- gaussian: $exp(-\eta^2)$
- triangular: $(1-\eta) * I(h <= range)$
- circular: $(1 - (2/\pi) * (m * sqrt(1-m^2) + sin^{-1}(m))) * I(h <= range), m = min(\eta, 1)$
- cubic: $(1 - 7\eta^2 + 8.75\eta^3 - 3.5\eta^5 + 0.75\eta^7) * I(h <= range)$
- pentaspherical: $(1 - 1.875\eta + 1.25\eta^3 - 0.375\eta^5) * I(h <= range)$

- cosine: $cos(\eta)$
- wave: $sin(\eta)/\eta * I(h > 0) + I(h = 0)$
- jbessel: $Bj(h * range)$, Bj is Bessel-J function
- gravity: $(1 + \eta^2)^{-0.5}$
- rquad: $(1 + \eta^2)^{-1}$
- magnetic: $(1 + \eta^2)^{-1.5}$
- matern: $2^{1-extra}/\Gamma(extra) * \alpha^{extra} * Bk(\alpha, extra)$, $\alpha = (2extra)^{0.5} * \eta$, Bk is Bessel-K function with order $1/5 \leq extra \leq 5$
- cauchy: $(1 + \eta^2)^{-extra}$, $extra > 0$
- pexponential: $exp(h^{extra}/range)$, $0 < extra \leq 2$
- none: 0
- ie: 0 (see below for differences compared to none)

There is a slight difference between `none` and `ie` (the `spcov_type`) from above. `none` fixes the `"ie"` covariance parameter at zero, which should yield a model that has very similar parameter estimates as one from stats::glm. Note that `spglm()` uses a different likelihood, so direct likelihood-based comparisons (e.g., `AIC`) should not be made to compare an `spglm()` model to a `glm()` one (though deviance can be used). `ie` allows the `"ie"` covariance parameter to vary. If the `"ie"` covariance parameter is estimated near zero, then `none` and `ie` yield similar models.

All spatial covariance functions are valid in one spatial dimension. All spatial covariance functions except `triangular` and `cosine` are valid in two dimensions.

`estmethod` Details: The various estimation methods are

- `reml`: Maximize the restricted log-likelihood.
- `ml`: Maximize the log-likelihood.

Note that the likelihood being optimized is obtained using the Laplace approximation.

`anisotropy` Details: By default, all spatial covariance parameters except `rotate` and `scale` as well as all random effect variance parameters are assumed unknown, requiring estimation. If either `rotate` or `scale` are given initial values other than 0 and 1 (respectively) or are assumed unknown in `spcov_initial()`, `anisotropy` is implicitly set to `TRUE`. (Geometric) Anisotropy is modeled by transforming a covariance function that decays differently in different directions to one that decays equally in all directions via rotation and scaling of the original coordinates. The rotation is controlled by the `rotate` parameter in $[0, \pi]$ radians. The scaling is controlled by the `scale` parameter in $[0, 1]$. The anisotropy correction involves first a rotation of the coordinates clockwise by `rotate` and then a scaling of the coordinates' minor axis by the reciprocal of `scale`. The spatial covariance is then computed using these transformed coordinates.

`random` Details: If random effects are used, the model can be written as $y = X\beta + Z1u1 + ...Zjuj + \tau + \epsilon$, where each Z is a random effects design matrix and each u is a random effect.

`partition_factor` Details: The partition factor can be represented in matrix form as P, where elements of P equal one for observations in the same level of the partition factor and zero otherwise. The covariance matrix involving only the spatial and random effects components is then multiplied element-wise (Hadmard product) by P, yielding the final covariance matrix.

`local` Details: The big data approximation works by sorting observations into different levels of an index variable. Observations in different levels of the index variable are assumed to be uncorrelated

for the purposes of model fitting. Sparse matrix methods are then implemented for significant computational gains. Parallelization generally further speeds up computations when data sizes are larger than a few thousand. Both the "random" and "kmeans" values of method in local have random components. That means you may get slightly different results when using the big data approximation and rerunning spglm() with the same code. For consistent results, either set a seed via base::set.seed() or specify index to local.

Observations with NA response values are removed for model fitting, but their values can be predicted afterwards by running predict(object).

Value

A list with many elements that store information about the fitted model object. If spcov_type or spcov_initial are length one, the list has class spglm. Many generic functions that summarize model fit are available for spglm objects, including AIC, AICc, anova, augment, AUROC, BIC, coef, cooks.distance, covmatrix, deviance, fitted, formula, glance, glances, hatvalues, influence, labels, logLik, loocv, model.frame, model.matrix, plot, predict, print, pseudoR2, summary, terms, tidy, update, varcomp, and vcov. If spcov_type or spcov_initial are length greater than one, the list has class spglm_list and each element in the list has class spglm. glances can be used to summarize spglm_list objects, and the aforementioned spglm generics can be used on each individual list element (model fit).

Note

This function does not perform any internal scaling. If optimization is not stable due to large extremely large variances, scale relevant variables so they have variance 1 before optimization.

References

McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models*. London: Chapman and Hall.

Examples

```
spgmod <- spglm(presence ~ elev,
  family = "binomial", data = moose,
  spcov_type = "exponential"
)
summary(spgmod)
```

---

splm *Fit spatial linear models*

Description

Fit spatial linear models for point-referenced data (i.e., geostatistical models) using a variety of estimation methods, allowing for random effects, anisotropy, partition factors, and big data methods.

Usage

```
splm(
  formula,
  data,
  spcov_type,
  xcoord,
  ycoord,
  spcov_initial,
  estmethod = "reml",
  weights = "cressie",
  anisotropy = FALSE,
  random,
  randcov_initial,
  partition_factor,
  local,
  range_constrain,
  ddf,
  ...
)
```

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms on the right, separated by + operators.
data	A data frame or sf object object that contains the variables in fixed, random, and partition_factor as well as geographical information. If an sf object is provided with POINT geometries, the x-coordinates and y-coordinates are used directly. If an sf object is provided with POLYGON geometries, the x-coordinates and y-coordinates are taken as the centroids of each polygon.
spcov_type	The spatial covariance type. Available options include "exponential", "spherical", "gaussian", "triangular", "circular", "cubic", "pentaspherical", "cosine", "wave", "jbessel", "gravity", "rquad", "magnetic", "matern", "cauchy", "pexponential", and "none". Parameterizations of each spatial covariance type are available in Details. Multiple spatial covariance types can be provided as a character vector, and then splm() is called iteratively for each element and a list is returned for each model fit. The default for spcov_type is "exponential". When spcov_type is specified, all unknown spatial covariance parameters are estimated. spcov_type is ignored if spcov_initial is provided.
xcoord	The name of the column in data representing the x-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
ycoord	The name of the column in data representing the y-coordinate. Can be quoted or unquoted. Not required if data is an sf object.
spcov_initial	An object from spcov_initial() specifying initial and/or known values for the spatial covariance parameters. Multiple spcov_initial() objects can be

provided in a list. Then splm() is called iteratively for each element and a list is returned for each model fit.

estmethod — The estimation method. Available options include "reml" for restricted maximum likelihood, "ml" for maximum likelihood, "sv-wls" for semivariogram weighted least squares, and "sv-cl" for semivariogram composite likelihood. The default is "reml".

weights — Weights to use when estmethod is "sv-wls". Available options include "cressie", "cressie-dr", "cressie-nopairs", "cressie-dr-nopairs", "pairs", "pairs-invd", "pairs-invrd", and "ols". Parameterizations for each weight are available in Details. The default is "cressie".

anisotropy — A logical indicating whether (geometric) anisotropy should be modeled. Not required if spcov_initial is provided with 1) rotate assumed unknown or assumed known and non-zero or 2) scale assumed unknown or assumed known and less than one. When anisotropy is TRUE, computational times can significantly increase. The default is FALSE.

random — A one-sided linear formula describing the random effect structure of the model. Terms are specified to the right of the ~ operator. Each term has the structure x1 + ... + xn | g1/.../gm, where x1 + ... + xn specifies the model for the random effects and g1/.../gm is the grouping structure. Separate terms are separated by + and must generally be wrapped in parentheses. Random intercepts are added to each model implicitly when at least one other variable is defined. If a random intercept is not desired, this must be explicitly defined (e.g., x1 + ... + xn - 1 | g1/.../gm). If only a random intercept is desired for a grouping structure, the random intercept must be specified as 1 | g1/.../gm. Note that g1/.../gm is shorthand for (1 | g1/.../gm). If only random intercepts are desired and the shorthand notation is used, parentheses can be omitted.

randcov_initial — An optional object specifying initial and/or known values for the random effect variances.

partition_factor — A one-sided linear formula with a single term specifying the partition factor. The partition factor assumes observations from different levels of the partition factor are uncorrelated.

local — An optional logical or list controlling the big data approximation. If omitted, local is set to TRUE or FALSE based on the sample size (the number of non-missing observations in data) – if the sample size exceeds 5,000, local is set to TRUE. Otherwise it is set to FALSE. local is also set to FALSE when spcov_type is "none" and there are no random effects specified via random. If FALSE, no big data approximation is implemented. If a list is provided, the following arguments detail the big data approximation:

- index: The group indexes. Observations in different levels of index are assumed to be uncorrelated for the purposes of estimation. If index is not provided, it is determined by specifying method and either size or groups.
- method: The big data approximation method used to determine index. Ignored if index is provided. If method = "random", observations are randomly assigned to index based on size. If method = "kmeans", observations assigned to index based on k-means clustering on the coordinates

with groups clusters. The default is "kmeans". Note that both methods have a random component, which means that you may get different results from separate model fitting calls. To ensure consistent results, specify index or set a seed via base::set.seed().

- size: The number of observations in each index group when method is "random". If the number of observations is not divisible by size, some levels get size - 1 observations. The default is 100.
- groups: The number of index groups. If method is "random", size is $ceiling(n/groups)$, where n is the sample size. Automatically determined if size is specified. If method is "kmeans", groups is the number of clusters.
- var_adjust: The approach for adjusting the variance-covariance matrix of the fixed effects. "none" for no adjustment, "theoretical" for the theoretically-correct adjustment, "pooled" for the pooled adjustment, and "empirical" for the empirical adjustment. The default is "theoretical" for samples sizes up to 100,000 and "none" for samples sizes exceeding 100,000.
- parallel: If TRUE, parallel processing via the parallel package is automatically used. The default is FALSE.
- ncores: If parallel = TRUE, the number of cores to parallelize over. The default is the number of available cores on your machine.

When local is a list, at least one list element must be provided to initialize default arguments for the other list elements. If local is TRUE, defaults for local are chosen such that local is transformed into list(size = 100, method = "kmeans", var_adjust = "theoretical", parallel = FALSE).

range_constrain

An optional logical that indicates whether the range should be constrained to enhance numerical stability. If range_constrain = TRUE, the maximum possible range value is 4 times the maximum distance in the domain. If range_constrain = FALSE, then maximum possible range is unbounded. The default is FALSE. Note that if range_constrain = TRUE and the value of range in spcov_initial is larger than range_constrain, then range_constrain is set to FALSE.

ddf

The denominator degrees of freedom to be used for eventual hypothesis testing (e.g., confint(), summary(), or tidy() calls on the fitted model object). "asymptotic" assumes infinite degrees of freedom, implying z-tests. "satterthwaite" impelments Satterthwaite degrees of freeom, implying t-tests that are generally more appropriate for small samples. The default is "satterthwaite" when the sample size is less than or equal to 500 and "asymptotic" otherwise.

...

Other arguments to esv() or stats::optim().

Details

The spatial linear model for point-referenced data (i.e., geostatistical model) can be written as $y = X\beta + \tau + \epsilon$, where X is the fixed effects design matrix, β are the fixed effects, τ is random error that is spatially dependent, and ϵ is random error that is spatially independent. Together, τ and ϵ are modeled using a spatial covariance function, expressed as $de * R + ie * I$, where de is the

dependent error variance, R is a correlation matrix that controls the spatial dependence structure among observations, ie is the independent error variance, and I is an identity matrix.

`spcov_type` Details: Parametric forms for R are given below, where $\eta = h/range$ for h distance between observations:

- exponential: $exp(-\eta)$
- spherical: $(1 - 1.5\eta + 0.5\eta^3) * I(h <= range)$
- gaussian: $exp(-\eta^2)$
- triangular: $(1 - \eta) * I(h <= range)$
- circular: $(1 - (2/\pi) * (m * sqrt(1 - m^2) + sin^{-1}(m))) * I(h <= range), m = min(\eta, 1)$
- cubic: $(1 - 7\eta^2 + 8.75\eta^3 - 3.5\eta^5 + 0.75\eta^7) * I(h <= range)$
- pentaspherical: $(1 - 1.875\eta + 1.25\eta^3 - 0.375\eta^5) * I(h <= range)$
- cosine: $cos(\eta)$
- wave: $sin(\eta)/\eta * I(h > 0) + I(h = 0)$
- jbessel: $Bj(h * range)$, Bj is Bessel-J function
- gravity: $(1 + \eta^2)^{-0.5}$
- rquad: $(1 + \eta^2)^{-1}$
- magnetic: $(1 + \eta^2)^{-1.5}$
- matern: $2^{1-extra}/\Gamma(extra) * \alpha^{extra} * Bk(\alpha, extra)$, $\alpha = (2extra)^{0.5} * \eta$, Bk is Bessel-K function with order $1/5 \leq extra \leq 5$
- cauchy: $(1 + \eta^2)^{-extra}$, $extra > 0$
- pexponential: $exp(h^{extra}/range)$, $0 < extra \leq 2$
- none: 0

All spatial covariance functions are valid in one spatial dimension. All spatial covariance functions except `triangular` and `cosine` are valid in two dimensions. An alias for `none` is `ie`.

`estmethod` Details: The various estimation methods are

- `reml`: Maximize the restricted log-likelihood.
- `ml`: Maximize the log-likelihood.
- `sv-wls`: Minimize the semivariogram weighted least squares loss.
- `sv-cl`: Minimize the semivariogram composite likelihood loss.

`anisotropy` Details: By default, all spatial covariance parameters except `rotate` and `scale` as well as all random effect variance parameters are assumed unknown, requiring estimation. If either `rotate` or `scale` are given initial values other than 0 and 1 (respectively) or are assumed unknown in `spcov_initial()`, `anisotropy` is implicitly set to `TRUE`. (Geometric) Anisotropy is modeled by transforming a covariance function that decays differently in different directions to one that decays equally in all directions via rotation and scaling of the original coordinates. The rotation is controlled by the `rotate` parameter in $[0, \pi]$ radians. The scaling is controlled by the `scale` parameter in $[0, 1]$. The anisotropy correction involves first a rotation of the coordinates clockwise by `rotate` and then a scaling of the coordinates' minor axis by the reciprocal of `scale`. The spatial covariance is then computed using these transformed coordinates.

random Details: If random effects are used (the estimation method must be "reml" or "ml"), the model can be written as $y = X\beta + Z1u1 + ...Zjuj + \tau + \epsilon$, where each Z is a random effects design matrix and each u is a random effect.

partition_factor Details: The partition factor can be represented in matrix form as P, where elements of P equal one for observations in the same level of the partition factor and zero otherwise. The covariance matrix involving only the spatial and random effects components is then multiplied element-wise (Hadmard product) by P, yielding the final covariance matrix.

local Details: The big data approximation works by sorting observations into different levels of an index variable. Observations in different levels of the index variable are assumed to be uncorrelated for the purposes of model fitting. Sparse matrix methods are then implemented for significant computational gains. Parallelization generally further speeds up computations when data sizes are larger than a few thousand. Both the "random" and "kmeans" values of method in local have random components. That means you may get slightly different results when using the big data approximation and rerunning splm() with the same code. For consistent results, either set a seed via base::set.seed() or specify index to local.

Observations with NA response values are removed for model fitting, but their values can be predicted afterwards by running predict(object).

Value

A list with many elements that store information about the fitted model object. If spcov_type or spcov_initial are length one, the list has class splm. Many generic functions that summarize model fit are available for splm objects, including AIC, AICc, anova, augment, BIC, coef, cooks.distance, covmatrix, deviance, fitted, formula, glance, glances, hatvalues, influence, labels, logLik, loocv, model.frame, model.matrix, plot, predict, print, pseudoR2, satterthwaite, summary, terms, tidy, update, varcomp, and vcov. If spcov_type or spcov_initial are length greater than one, the list has class splm_list and each element in the list has class splm. glances can be used to summarize splm_list objects, and the aforementioned splm generics can be used on each individual list element (model fit).

Note

This function does not perform any internal scaling. If optimization is not stable due to large extremely large variances, scale relevant variables so they have variance 1 before optimization.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
summary(spmod)
```

splmRF *Fit random forest spatial residual models*

Description

Fit random forest spatial residual models for point-referenced data (i.e., geostatistical models) using random forest to fit the mean and a spatial linear model to fit the residuals. The spatial linear model fit to the residuals can incorporate variety of estimation methods, allowing for random effects, anisotropy, partition factors, and big data methods.

Usage

```
splmRF(formula, data, ...)
```

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms on the right, separated by + operators.
data	A data frame or sf object object that contains the variables in fixed, random, and partition_factor as well as geographical information. If an sf object is provided with POINT geometries, the x-coordinates and y-coordinates are used directly. If an sf object is provided with POLYGON geometries, the x-coordinates and y-coordinates are taken as the centroids of each polygon.
...	Additional named arguments to ranger::ranger() or splm().

Details

The random forest residual spatial linear model is described by Fox et al. (2020). A random forest model is fit to the mean portion of the model specified by formula using ranger::ranger(). Residuals are computed and used as the response variable in an intercept-only spatial linear model fit using splm(). This model object is intended for use with predict() to perform prediction, also called random forest regression Kriging.

Value

A list with several elements to be used with predict(). These elements include the function call (named call), the random forest object fit to the mean (named ranger), the spatial linear model object fit to the residuals (named splm or splm_list), and an object can contain data for locations at which to predict (called newdata). The newdata object contains the set of observations in data whose response variable is NA. If spcov_type or spcov_initial (which are passed to splm()) are length one, the list has class splmRF and the spatial linear model object fit to the residuals is called splm, which has class splm. If spcov_type or spcov_initial are length greater than one, the list has class splmRF_list and the spatial linear model object fit to the residuals is called splm_list, which has class splm_list. and contains several objects, each with class splm.

An splmRF object to be used with predict(). There are three elements: ranger, the output from fitting the mean model with ranger::ranger(); splm, the output from fitting the spatial linear model to the ranger residuals; and newdata, the newdata object, if relevant.

Note

This function does not perform any internal scaling. If optimization is not stable due to large extremely large variances, scale relevant variables so they have variance 1 before optimization.

References

Fox, E.W., Ver Hoef, J. M., & Olsen, A. R. (2020). Comparing spatial regression to random forests for large environmental data sets. *PloS one*, 15(3), e0229509.

Examples

```
sprfmod <- splmRF(log_cond ~ temp + precip, data = lake, spcov_type = "exponential")
predict(sprfmod, lake_preds)
```

---

sprbeta *Simulate a spatial beta random variable*

Description

Simulate a spatial beta random variable with a specific mean and covariance structure.

Usage

```
sprbeta(
  spcov_params,
  dispersion = 1,
  mean = 0,
  samples = 1,
  data,
  randcov_params,
  partition_factor,
  ...
)
```

Arguments

spcov_params	An spcov_params() object.
dispersion	The dispersion value.
mean	A numeric vector representing the mean. mean must have length 1 (in which case it is recycled) or length equal to the number of rows in data. The default is 0.
samples	The number of independent samples to generate. The default is 1.
data	A data frame or sf object containing spatial information.
randcov_params	A randcov_params() object.
partition_factor	A formula indicating the partition factor.
...	Additional arguments passed to sprnorm().

Details

The values of `spcov_params`, `mean`, and `randcov_params` are assumed to be on the link scale. They are used to simulate a latent normal (Gaussian) response variable using `sprnorm()`. This latent variable is the conditional mean used with `dispersion` to simulate a beta random variable.

Value

If `samples` is 1, a vector of random variables for each row of `data` is returned. If `samples` is greater than one, a matrix of random variables is returned, where the rows correspond to each row of `data` and the columns correspond to independent samples.

Examples

```
spcov_params_val <- spcov_params("exponential", de = 0.2, ie = 0.1, range = 1)
sprbeta(spcov_params_val, data = caribou, xcoord = x, ycoord = y)
sprbeta(spcov_params_val, samples = 5, data = caribou, xcoord = x, ycoord = y)
```

---

sprbinom *Simulate a spatial binomial random variable*

Description

Simulate a spatial binomial random variable with a specific mean and covariance structure.

Usage

```
sprbinom(
  spcov_params,
  mean = 0,
  size = 1,
  samples = 1,
  data,
  randcov_params,
  partition_factor,
  ...
)
```

Arguments

`spcov_params`	An `spcov_params()` object.
`mean`	A numeric vector representing the mean. `mean` must have length 1 (in which case it is recycled) or length equal to the number of rows in `data`. The default is `0`.
`size`	A numeric vector representing the sample size for each binomial trial. The default is `1`, which corresponds to a Bernoulli trial for each observation.
`samples`	The number of independent samples to generate. The default is `1`.

data	A data frame or sf object containing spatial information.
randcov_params	A randcov_params() object.
partition_factor	A formula indicating the partition factor.
...	Additional arguments passed to sprnorm().

Details

The values of spcov_params, mean, and randcov_params are assumed to be on the link scale. They are used to simulate a latent normal (Gaussian) response variable using sprnorm(). This latent variable is the conditional mean used with dispersion to simulate a binomial random variable.

Value

If samples is 1, a vector of random variables for each row of data is returned. If samples is greater than one, a matrix of random variables is returned, where the rows correspond to each row of data and the columns correspond to independent samples.

Examples

```
spcov_params_val <- spcov_params("exponential", de = 0.2, ie = 0.1, range = 1)
sprbinom(spcov_params_val, data = caribou, xcoord = x, ycoord = y)
sprbinom(spcov_params_val, samples = 5, data = caribou, xcoord = x, ycoord = y)
```

---

sprgamma *Simulate a spatial gamma random variable*

Description

Simulate a spatial gamma random variable with a specific mean and covariance structure.

Usage

```
sprgamma(
  spcov_params,
  dispersion = 1,
  mean = 0,
  samples = 1,
  data,
  randcov_params,
  partition_factor,
  ...
)
```

Arguments

spcov_params	An spcov_params() object.
dispersion	The dispersion value.
mean	A numeric vector representing the mean. mean must have length 1 (in which case it is recycled) or length equal to the number of rows in data. The default is 0.
samples	The number of independent samples to generate. The default is 1.
data	A data frame or sf object containing spatial information.
randcov_params	A randcov_params() object.
partition_factor	A formula indicating the partition factor.
...	Additional arguments passed to sprnorm().

Details

The values of spcov_params, mean, and randcov_params are assumed to be on the link scale. They are used to simulate a latent normal (Gaussian) response variable using sprnorm(). This latent variable is the conditional mean used with dispersion to simulate a gamma random variable.

Value

If samples is 1, a vector of random variables for each row of data is returned. If samples is greater than one, a matrix of random variables is returned, where the rows correspond to each row of data and the columns correspond to independent samples.

Examples

```
spcov_params_val <- spcov_params("exponential", de = 0.2, ie = 0.1, range = 1)
sprgamma(spcov_params_val, data = caribou, xcoord = x, ycoord = y)
sprgamma(spcov_params_val, samples = 5, data = caribou, xcoord = x, ycoord = y)
```

---

sprinvgauss *Simulate a spatial inverse gaussian random variable*

Description

Simulate a spatial inverse gaussian random variable with a specific mean and covariance structure.

Usage

```
sprinvgauss(
  spcov_params,
  dispersion = 1,
  mean = 0,
  samples = 1,
```

```
  data,
  randcov_params,
  partition_factor,
  ...
)
```

Arguments

spcov_params	An spcov_params() object.
dispersion	The dispersion value.
mean	A numeric vector representing the mean. mean must have length 1 (in which case it is recycled) or length equal to the number of rows in data. The default is 0.
samples	The number of independent samples to generate. The default is 1.
data	A data frame or sf object containing spatial information.
randcov_params	A randcov_params() object.
partition_factor	A formula indicating the partition factor.
...	Additional arguments passed to sprnorm().

Details

The values of spcov_params, mean, and randcov_params are assumed to be on the link scale. They are used to simulate a latent normal (Gaussian) response variable using sprnorm(). This latent variable is the conditional mean used with dispersion to simulate a inverse gaussian random variable.

Value

If samples is 1, a vector of random variables for each row of data is returned. If samples is greater than one, a matrix of random variables is returned, where the rows correspond to each row of data and the columns correspond to independent samples.

Examples

```
spcov_params_val <- spcov_params("exponential", de = 0.2, ie = 0.1, range = 1)
sprinvgauss(spcov_params_val, data = caribou, xcoord = x, ycoord = y)
sprinvgauss(spcov_params_val, samples = 5, data = caribou, xcoord = x, ycoord = y)
```

sprnbinom *Simulate a spatial negative binomial random variable*

Description

Simulate a spatial negative binomial random variable with a specific mean and covariance structure.

Usage

```
sprnbinom(
  spcov_params,
  dispersion = 1,
  mean = 0,
  samples = 1,
  data,
  randcov_params,
  partition_factor,
  ...
)
```

Arguments

spcov_params	An spcov_params() object.
dispersion	The dispersion value.
mean	A numeric vector representing the mean. mean must have length 1 (in which case it is recycled) or length equal to the number of rows in data. The default is 0.
samples	The number of independent samples to generate. The default is 1.
data	A data frame or sf object containing spatial information.
randcov_params	A randcov_params() object.
partition_factor	A formula indicating the partition factor.
...	Additional arguments passed to sprnorm().

Details

The values of spcov_params, mean, and randcov_params are assumed to be on the link scale. They are used to simulate a latent normal (Gaussian) response variable using sprnorm(). This latent variable is the conditional mean used with dispersion to simulate a negative binomial random variable.

Value

If samples is 1, a vector of random variables for each row of data is returned. If samples is greater than one, a matrix of random variables is returned, where the rows correspond to each row of data and the columns correspond to independent samples.

Examples

```
spcov_params_val <- spcov_params("exponential", de = 0.2, ie = 0.1, range = 1)
sprnbinom(spcov_params_val, data = caribou, xcoord = x, ycoord = y)
sprnbinom(spcov_params_val, samples = 5, data = caribou, xcoord = x, ycoord = y)
```

---

sprnorm *Simulate a spatial normal (Gaussian) random variable*

Description

Simulate a spatial normal (Gaussian) random variable with a specific mean and covariance structure.

Usage

```
sprnorm(
  spcov_params,
  mean = 0,
  samples = 1,
  data,
  randcov_params,
  partition_factor,
  ...
)

## S3 method for class 'exponential'
sprnorm(
  spcov_params,
  mean = 0,
  samples = 1,
  data,
  randcov_params,
  partition_factor,
  xcoord,
  ycoord,
  ...
)

## S3 method for class 'none'
sprnorm(
  spcov_params,
  mean = 0,
  samples = 1,
  data,
  randcov_params,
  partition_factor,
  ...
)
```

```
## S3 method for class 'ie'
sprnorm(
  spcov_params,
  mean = 0,
  samples = 1,
  data,
  randcov_params,
  partition_factor,
  ...
)

## S3 method for class 'car'
sprnorm(
  spcov_params,
  mean = 0,
  samples = 1,
  data,
  randcov_params,
  partition_factor,
  W,
  row_st = TRUE,
  M,
  ...
)
```

Arguments

spcov_params	An spcov_params() object.
mean	A numeric vector representing the mean. mean must have length 1 (in which case it is recycled) or length equal to the number of rows in data. The default is 0.
samples	The number of independent samples to generate. The default is 1.
data	A data frame or sf object containing spatial information.
randcov_params	A randcov_params() object.
partition_factor	A formula indicating the partition factor.
...	Other arguments. Not used (needed for generic consistency).
xcoord	Name of the column in data representing the x-coordinate. Can be quoted or unquoted. Not required if data are an sf object.
ycoord	Name of the column in data representing the y-coordinate. Can be quoted or unquoted. Not required if data are an sf object.
W	Weight matrix specifying the neighboring structure used for car and sar models. Not required if data are an sf polygon object and W should be calculated internally (using queen contiguity).

row_st	A logical indicating whether row standardization be performed on W. The default is TRUE.
M	M matrix satisfying the car symmetry condition. The car symmetry condition states that $(I - range * W)^{-1}M$ is symmetric, where I is an identity matrix, $range$ is a constant that controls the spatial dependence, W is the weights matrix, and $^{-1}$ represents the inverse operator. M is required for car models when W is provided and row_st is FALSE. When M, is required, the default is the identity matrix.

Details

Random variables are simulated via the product of the covariance matrix's square (Cholesky) root and independent standard normal random variables with mean 0 and variance 1. It is nearly the sample computational cost to call sprnorm() for any value of samples.

Only methods for the exponential and car covariance functions are documented here, but methods exist for all other spatial covariance functions defined in spcov_initial(). Syntax for the exponential method is the same as syntax for ie, spherical, gaussian, triangular, circular, cubic, pentaspherical, cosine, wave, jbessel, gravity, rquad, magnetic, matern, cauchy, and pexponential methods. Syntax for the car method is the same as syntax for the sar method. The extra parameter for car and sar models is ignored when all observations have neighbors.

Value

If samples is 1, a vector of random variables for each row of data is returned. If samples is greater than one, a matrix of random variables is returned, where the rows correspond to each row of data and the columns correspond to independent samples.

Examples

```
spcov_params_val <- spcov_params("exponential", de = 1, ie = 1, range = 1)
sprnorm(spcov_params_val, data = caribou, xcoord = x, ycoord = y)
sprnorm(spcov_params_val, mean = 1:30, samples = 5, data = caribou, xcoord = x, ycoord = y)
```

---

sprpois *Simulate a spatial Poisson random variable*

Description

Simulate a spatial Poisson random variable with a specific mean and covariance structure.

Usage

```
sprpois(
  spcov_params,
  mean = 0,
  samples = 1,
  data,
```

```
  randcov_params,
  partition_factor,
  ...
)
```

Arguments

spcov_params	An spcov_params() object.
mean	A numeric vector representing the mean. mean must have length 1 (in which case it is recycled) or length equal to the number of rows in data. The default is 0.
samples	The number of independent samples to generate. The default is 1.
data	A data frame or sf object containing spatial information.
randcov_params	A randcov_params() object.
partition_factor	A formula indicating the partition factor.
...	Additional arguments passed to sprnorm().

Details

The values of spcov_params, mean, and randcov_params are assumed to be on the link scale. They are used to simulate a latent normal (Gaussian) response variable using sprnorm(). This latent variable is the conditional mean used with dispersion to simulate a Poisson random variable.

Value

If samples is 1, a vector of random variables for each row of data is returned. If samples is greater than one, a matrix of random variables is returned, where the rows correspond to each row of data and the columns correspond to independent samples.

Examples

```
spcov_params_val <- spcov_params("exponential", de = 0.2, ie = 0.1, range = 1)
sprpois(spcov_params_val, data = caribou, xcoord = x, ycoord = y)
sprpois(spcov_params_val, samples = 5, data = caribou, xcoord = x, ycoord = y)
```

---

sulfate *Sulfate atmospheric deposition in the conterminous USA*

Description

Sulfate atmospheric deposition in the conterminous USA.

Usage

```
sulfate
```

Format

An sf object with 197 rows and 2 columns.

- sulfate: Total wet deposition sulfate in kilograms per hectare.
- geometry: POINT geometry representing coordinates in a Conus Albers projection (EPSG: 5070). Distances between points are in meters.

Source

These data were used in the publication listed in References. Data were downloaded from the National Atmospheric Deposition Program National Trends Network.

References

Zimmerman, D.L. (1994). Statistical analysis of spatial data. Pages 375-402 in *Statistical Methods for Physical Science*, J. Stanford and S. Vardeman (eds.), Academic Press: New York.

---

sulfate_preds *Locations at which to predict sulfate atmospheric deposition in the conterminous USA*

Description

Locations at which to predict sulfate atmospheric deposition in the conterminous USA.

Usage

```
sulfate_preds
```

Format

An sf object with 197 rows and 1 column.

- geometry: POINT geometry representing coordinates in a Conus Albers projection (EPSG: 5070).

Source

These data were used in the publication listed in References. Data were downloaded from the National Atmospheric Deposition Program National Trends Network.

References

Zimmerman, D.L. (1994). Statistical analysis of spatial data. Pages 375-402 in *Statistical Methods for Physical Science*, J. Stanford and S. Vardeman (eds.), Academic Press: New York.

summary.spmodel *Summarize a fitted model object*

Description

Summarize a fitted model object.

Usage

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

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

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

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

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

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

Arguments

object	A fitted model object from splm(), spautor(), spglm(), spgautor(), splmRF(), or spautorRF().
...	Other arguments. Not used (needed for generic consistency).

Details

summary() creates a summary of a fitted model object intended to be printed using print(). This summary contains useful information like the original function call, residuals, a coefficients table, a pseudo r-squared, and estimated covariance parameters.

Value

A list with several fitted model quantities used to create informative summaries when printing.

See Also

print.spmodel()

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
summary(spmod)

sprfmod <- splmRF(log_cond ~ temp + precip, data = lake, spcov_type = "exponential")
summary(sprfmod)


sprfmod <- spautorRF(log_trend ~ stock, data = seal, spcov_type = "car")
summary(sprfmod)
```

---

texas *Texas Turnout Data*

Description

Texas voter turnout data collected during the United States 1980 Presidential election.

Usage

```
texas
```

Format

An `sf` object with 254 rows and 5 columns:

- FIPS: Federal Information Processing System (FIPS) county codes.
- turnout: Proportion of eligible voters who voted.
- log_income: The natural logarithm of average per capita (in dollars) income.
- income: The average per capita (in dollars) income.
- geometry: `POINT` geometry representing coordinates in a NAD83 projection (EPSG: 5070). Distances between points are in meters.

Source

The data source is the `elect80` data set in the `spData` R package.

`tidy.spmodel` *Tidy a fitted model object*

Description

Tidy a fitted model object into a summarized tibble.

Usage

```
## S3 method for class 'splm'
tidy(x, conf.int = FALSE, conf.level = 0.95, effects = "fixed", ...)

## S3 method for class 'spautor'
tidy(x, conf.int = FALSE, conf.level = 0.95, effects = "fixed", ...)

## S3 method for class 'spglm'
tidy(x, conf.int = FALSE, conf.level = 0.95, effects = "fixed", ...)

## S3 method for class 'spgautor'
tidy(x, conf.int = FALSE, conf.level = 0.95, effects = "fixed", ...)
```

Arguments

`x`	A fitted model object from `splm()`, `spautor()`, `spglm()`, or `spgautor()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. The default is `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int` is `TRUE`. Must be strictly greater than 0 and less than 1. The default is 0.95, which corresponds to a 95 percent confidence interval.
`effects`	The type of effects to tidy. Available options are `"fixed"` (fixed effects), `"spcov"` (spatial covariance parameters), and `"randcov"` (random effect variances). The default is `"fixed"`.
`...`	Other arguments. Not used (needed for generic consistency).

Value

A tidy tibble of summary information `effects`.

See Also

`glance.spmodel()` `augment.spmodel()`

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
tidy(spmod)
tidy(spmod, effects = "spcov")
```

`varcomp` *Variability component comparison*

Description

Compare the proportion of total variability explained by the fixed effects and each variance parameter.

Usage

```
varcomp(object, ...)

## S3 method for class 'splm'
varcomp(object, ...)

## S3 method for class 'spautor'
varcomp(object, ...)

## S3 method for class 'spglm'
varcomp(object, ...)

## S3 method for class 'spgautor'
varcomp(object, ...)
```

Arguments

`object`	A fitted model object (e.g., from `splm()`, `spautor()`, `spglm()`, or `spgautor()`).
`...`	Other arguments. Not used (needed for generic consistency).

Details

The total variability in the response is decomposed into a portion explained by the fixed effects and a portion explained by each variance parameter in the fitted covariance structure:

- `de`: the spatially dependent (correlated) random error variance, commonly referred to as a partial sill.
- `ie`: the spatially independent (uncorrelated) random error variance, commonly referred to as a nugget.

- random effects: if `object` was fit with a `random` argument, one additional variance parameter per named random effect term (e.g., a random intercept's grouping variable), representing the variance attributable to that grouping.

See `spcov_params()` abd `spcov_initial()` for more on de/ie and `splm()` (or `spglm()`) for more on random effects. The proportion of variability explained by the fixed effects is the pseudo R-squared returned by `pseudoR2()`. The remaining `1 - pseudoR2` proportion is then split among de, ie, and any random effect variances, in proportion to their share of the total variance (the sum of de, ie, and all random effect variances).

Value

A tibble that partitions the total variability by the fixed effects and each variance parameter (see Details). If `spautor()` objects have unconnected sites, a list is returned with three elements: `"connected"` for a variability comparison among the connected sites; `"unconnected"` for a variability comparison among the unconnected sites; and `"ratio"` for the ratio of the variance of the connected sites relative to the variance of the unconnected sites.

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
varcomp(spmod)
```

---

vcov.spmodel *Calculate variance-covariance matrix for a fitted model object*

Description

Calculate variance-covariance matrix for a fitted model object.

Usage

```
## S3 method for class 'splm'
vcov(object, type = "fixed", ...)

## S3 method for class 'spautor'
vcov(object, type = "fixed", ...)

## S3 method for class 'spglm'
vcov(object, var_correct = TRUE, ...)

## S3 method for class 'spgautor'
vcov(object, var_correct = TRUE, ...)
```

Arguments

object	A fitted model object from splm(), spautor(), spglm(), or spgautor().
type	For type = "fixed" (the default), the variance-covariance matrix of the fixed effects. If Satterthwaite degrees of freedom were calculated, type = "cov" returns the variance-covariance matrix of the covariance parameters, type = "spcov" returns the variance-covariance matrix of just the spatial variance-covariance parameters, and type = "randcov" returns the variance-covariance matrix of just the random effects (if relevant).
...	Other arguments. Not used (needed for generic consistency).
var_correct	A logical indicating whether to return the corrected variance-covariance matrix for models fit using spglm() or spgautor(). The default is TRUE.

Value

The variance-covariance matrix of fixed effect coefficients obtained via coef(..., type = "fixed") or the variance-covariance matrix of estimated covariance parameters (when available).

Examples

```
spmod <- splm(z ~ water + tarp,
  data = caribou,
  spcov_type = "exponential", xcoord = x, ycoord = y
)
vcov(spmod)
```

Index