Package 'SSN2'

October 21, 2025

Title Spatial Modeling on Stream Networks

Version 0.4.0

Description Spatial statistical modeling and prediction for data on stream networks, including models based on in-stream distance (Ver Hoef, J.M. and Peterson, E.E., (2010) <DOI:10.1198/jasa.2009.ap08248>.) Models are created using moving average constructions. Spatial linear models, including explanatory variables, can be fit with (restricted) maximum likelihood. Mapping and other graphical functions are included.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 2.10)

Imports stats, sf, Matrix, generics, tibble, graphics, spmodel (>= 0.7.0), RSQLite, utils, withr, doParallel, filematrix, foreach, parallel, itertools, iterators

Suggests rmarkdown, knitr, testthat (>= 3.0.0), ggplot2, sp, statmod, pROC, emmeans (>= 1.4), estimability

Config/testthat/edition 3

VignetteBuilder knitr

URL https://usepa.github.io/SSN2/

BugReports https://github.com/USEPA/SSN2/issues

NeedsCompilation yes

Author Michael Dumelle [aut, cre] (ORCID: <https://orcid.org/0000-0002-3393-5529>), Jay M. Ver Hoef [aut], Erin Peterson [aut], Alan Pearse [ctb], Dan Isaak [ctb], Ryan A. Hill [ctb], Marc Weber [ctb]

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

Repository CRAN

Date/Publication 2025-10-21 05:10:29 UTC

Contents

`amongSitesBigDistMat` *Helper function to determining distance matrices among sites*

Description

Helper function to determining distance matrices among sites

Usage

```
amongSitesBigDistMat(
  ssn,
  pids,
  net.num,
  name = "obs",
  bin.table,
  workspace.name
)
```

Arguments

`ssn`	An SSN object.
`pids`	A list of pid values for prediction sites
`net.num`	Network number (netID) in character format.
`name`	The network name (obs or prediction name)
`bin.table`	A binaryID table for the network.
`workspace.name`	Name of new distance matrix file

Value

A distance matrix

anova.SSN2 *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 'ssn_lm'
anova(object, ..., test = TRUE, Terms, L)

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

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

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

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().
...	An additional fitted model object from ssn_lm() or ssn_glm() (for anova()).
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.
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 one fitted model object is present, `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 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.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
anova(ssn_mod)
tidy(anova(ssn_mod))
```

`augment.SSN2` *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` are 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 (confidence or prediction) or standard errors are augmented to each row in the new data set.

Usage

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

## S3 method for class 'ssn_glm'
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,
  ...
)
```

Arguments

`x`	A fitted model object from `ssn_lm()` or `ssn_glm()`.
`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 vector that contains the names of the prediction `sf` objects from the original `ssn.object` requiring prediction. All of the original explanatory variables used to create the fitted model object `x` must be present in each prediction `sf` object represented by `newdata`. Defaults to `NULL`, which indicates that nothing has been passed to `newdata` and augmenting occurs for the original data. The value `"ssn"` is shorthand for specifying all prediction `sf` objects.

se_fit	Logical indicating whether or not a .se.fit column should be added to augmented output. Passed to predict() and defaults to FALSE.
interval	Character indicating the type of confidence interval columns to add to the augmented newdata output. Passed to predict() and 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.ssn_lm() or predict.ssn_glm() control the big data approximation behavior. If a logical, TRUE chooses default list elements for the list version of local as specified in predict.ssn_lm() or predict.ssn_glm(). Defaults to FALSE, which performs exact computations.
...	Additional arguments to predict() when augmenting newdata.
type.predict	The scale (response or link) of fitted values and predictions obtained using ssn_glm() objects.
type.residuals	The residual type (deviance, pearson, or response) of fitted models from ssn_glm() 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 ssn_glm. The default is TRUE.

Details

augment() returns a tibble as an sf object.

Missing response values from the original data can be augmented as if they were a newdata object by providing ".missing" to the newdata argument.

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

When predictions for all prediction objects are desired, the output is a list where each element has a name that matches the prediction objects and values that are the predictions.

See Also

tidy.SSN2() glance.SSN2()

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, predpts = "CapeHorn", overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
augment(ssn_mod)
augment(ssn_mod, newdata = "CapeHorn")
```

---

coef.SSN2 *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 'ssn_lm'
coef(object, type = "fixed", ...)

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

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

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

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().

type	"fixed" for fixed effect coefficients, "tailup" for tailup covariance parameter coefficients, "taildown" for taildown covariance parameter coefficients, "euclid" for Euclidean covariance parameter coefficients, "nugget" for nugget covariance parameter coefficients, "dispersion" for the dispersion parameter coefficient (ssn_glm() objects), "randcov" for random effect variance coefficients, or "ssn" for all of the tailup, taildown, Euclidean, nugget, and dispersion (ssn_glm() objects) parameter coefficients. Defaults to "fixed".
...	Other arguments. Not used (needed for generic consistency).

Value

A named vector of coefficients.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
coef(ssn_mod)
coef(ssn_mod, type = "tailup")
coefficients(ssn_mod)
```

---

confint.SSN2 *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 'ssn_lm'
confint(object, parm, level = 0.95, ...)

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

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().
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

Gaussian-based confidence intervals (two-sided and equal-tailed) for the fixed effect coefficients based on the confidence level specified by level. For ssn_glm() objects, confidence intervals are on the link scale.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
confint(ssn_mod)
confint(ssn_mod, level = 0.9)
```

---

cooks.distance.SSN2 *Compute Cook's distance*

Description

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

Usage

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

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

Arguments

model	A fitted model object from ssn_lm() or ssn_glm().
...	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.SSN2() hatvalues.SSN2() influence.SSN2() residuals.SSN2()

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
cooks.distance(ssn_mod)
```

---

copy_lsn_to_temp *Copy LSN to temporary directory*

Description

Copies the LSN directory MiddleFork04.ssn to R's temporary directory so the examples in SSN2 do not write to the local library or any other places.

Usage

```
copy_lsn_to_temp()
```

Details

Copies the LSN directory MiddleFork04.ssn to R's temporary directory

Value

A copy of MiddleFork04.ssn residing in R's temporary directory

Examples

```
copy_lsn_to_temp()
# getwd()
# setwd(tempdir())
# getwd()
# if unix-alike, list temporary directory contents using: system('ls')
# if windows, list temporary directory contents using: shell('dir')
```

---

covmatrix.SSN2 *Create a covariance matrix*

Description

Create a covariance matrix from a fitted model object.

Usage

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

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

Arguments

object	A fitted model object (e.g., ssn_lm() or ssn_glm()).
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.
cov_type	The type of covariance matrix returned. If newdata is omitted, 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), the $m \times n$ covariance matrix of the predicted and observed data is returned, where m is the number of observations 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.
...	Other arguments. Not used (needed for generic consistency).

Value

A covariance matrix (see cov_type).

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, predpts = "CapeHorn", overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
covmatrix(ssn_mod)
covmatrix(ssn_mod, "CapeHorn")
```

---

create_netgeom *Create netgeom column in SSN object*

Description

Create netgeom column for edges, observed sites, and/or prediction sites in a Landscape Network (LSN).

Usage

```
create_netgeom(sf_data, type = NULL, overwrite = FALSE)
```

Arguments

sf_data	An sf object with LINESTING or POINT geometry created using link{lsn_to_ssn} (see Details).
type	Character string defining geometry type of sf_data. Default = NULL.
overwrite	Logical indicating whether existing data should be overwritten if present. Default = FALSE.

Details

Most users will not need to run create_netgeom themselves because it is called internally when lsn_to_ssn is run or an SSN is imported using link[SSN2]{ssn_import} found in the SSN2 package. For users who do wish to run create_netgeom, the sf_data object must represent edges, observed sites, or prediction sites in a SSN object created using link{lsn_to_ssn}.

The netgeom column contains information in character format used to describe the topology of the LSN. The format and content of the netgeom column differs depending on whether `sf_data` contains LINESTRING (edges) or POINT (observed or prediction sites) geometry. For edges, the netgeom format is:

- `'ENETWORK (netID, rid, upDist)'`

For observed or prediction sites, the netgeom format is:

- `'SNETWORK (netID, rid, upDist, ratio, pid, locID)'`

The rid, ratio, upDist, netID, pid, and locID columns must be present in `sf_data` before netgeom is added.

If `overwrite = TRUE` and a column named netgeom is present in `sf_data`, the data will be overwritten. Default = FALSE.

Value

An `sf` object containing the original data from `sf_data` and an additional column named netgeom.

Examples

```
## Create local temporary copy of MiddleFork04.ssn found in
## the SSN2 package. Only necessary for this example.
copy_lsn_to_temp()

# Import the SSN object with prediction points, pred1km
mf04<- ssn_import(
   paste0(tempdir(), "/MiddleFork04.ssn"),
   predpts = c("pred1km"),
   overwrite = TRUE
)

# Recalculate the netgeom column for the observed sites
sf_obs <- create_netgeom(
    mf04$obs,
    type = "POINT",
    overwrite = TRUE
)

# Recalculate the netgeom column for the edges
sf_edges <- create_netgeom(
    mf04$edges,
    type = "LINESTRING",
    overwrite = TRUE
)
```

deviance.SSN2 *Fitted model deviance*

Description

Returns the deviance of a fitted model object.

Usage

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

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

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().
...	Other arguments. Not used (needed for generic consistency).

Details

The deviance is twice the difference in log-likelihoods between the saturated (perfect-fit) model and the fitted model.

Value

The deviance.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_glm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  family = "Gamma",
  tailup_type = "exponential",
  additive = "afvArea"
)
deviance(ssn_mod)
```

`fitted.SSN2` *Extract model fitted values*

Description

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

Usage

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

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

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

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

Arguments

`object`	A fitted model object from `ssn_lm()` or `ssn_glm()`.
`type`	`"response"` for fitted values of the response, `"tailup"` for fitted values of the tailup random errors, `"taildown"` for fitted values of the taildown random errors, `"euclid"` for fitted values of the Euclidean random errors, `"nugget"` for fitted values of the nugget random errors, or `"randcov"` for fitted values of the random effects. If from `ssn_glm()`, `"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 `"tailup"`, `"taildown"`, `"euclid"`, or `"nugget"` the fitted values for each observation are (generally) the best linear unbiased predictors of the respective 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` `"tailup"`, `"taildown"`, `"euclid"`, `"nugget"`, and `"randcov"` can generally be used to check assumptions for each component of the fitted model object (e.g., check a Gaussian assumption).

If from `ssn_glm()`, when `type` is `"response"`, the fitted values for each observation are the standard fitted values on the inverse link scale: $g^{-1}(X\hat{\beta} + \nu)$, where $g(.)$ is a link function, β are the fixed effects, and ν are the spatial and random effects.

Value

The fitted values according to type.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
fitted(ssn_mod)
fitted.values(ssn_mod)
```

---

formula.SSN2 *Model formulae*

Description

Return formula used by a fitted model object.

Usage

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

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

Arguments

x	A fitted model object from ssn_lm() or ssn_glm().
...	Other arguments. Not used (needed for generic consistency).

Value

The formula used by a fitted model object.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
formula(ssn_mod)
```

`glance.SSN2` *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.

Usage

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

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

Arguments

`x`	A fitted model object from `ssn_lm()` or `ssn_glm()`.
`...`	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

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
glance(ssn_mod)
```

---

glances.SSN2 *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

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

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

Arguments

object	Fitted model object from ssn_lm() or ssn_glm().
...	Additional fitted model objects from ssn_lm() or ssn_glm().
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	Should sort_by be decreasing or not? 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

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

# tailup only
ssn_mod1 <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
# taildown only
ssn_mod2 <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  taildown_type = "exponential"
)
glances(ssn_mod1, ssn_mod2)
```

---

hatvalues.SSN2 *Compute leverage (hat) values*

Description

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

Usage

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

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

Arguments

`model`	A fitted model object from `ssn_lm()` or `ssn_glm()`.
`...`	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.SSN2() cooks.distance.SSN2() influence.SSN2() residuals.SSN2()

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
hatvalues(ssn_mod)
```

---

influence.SSN2 *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 'ssn_lm'
influence(model, ...)

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

Arguments

model	A fitted model object from ssn_lm() or ssn_glm().
...	Other arguments. Not used (needed for generic consistency).

Details

This function calls residuals.SSN2(), hatvalues.SSN2(), and cooks.distance.SSN2() and puts the results into a tibble. It is primarily used when calling augment.SSN2().

Value

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

See Also

augment.SSN2() cooks.distance.SSN2() hatvalues.SSN2() residuals.SSN2()

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
influence(ssn_mod)
```

---

labels.SSN2 *Find labels from object*

Description

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

Usage

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

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

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().
...	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

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
labels(ssn_mod)
```

---

logLik.SSN2 *Extract log-likelihood*

Description

Find the log-likelihood of a fitted model.

Usage

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

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

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().
...	Other arguments. Not used (needed for generic consistency).

Value

The log-likelihood.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
logLik(ssn_mod)
```

`loocv.SSN2` *Perform leave-one-out cross validation*

Description

Perform leave-one-out cross validation.

Usage

```
## S3 method for class 'ssn_lm'
loocv(object, cv_predict = FALSE, se.fit = FALSE, ...)

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

Arguments

`object`	A fitted model object from `ssn_lm()` or `ssn_glm()`.
`cv_predict`	A logical indicating whether the leave-one-out fitted values should be returned. Defaults to `FALSE`.
`se.fit`	A logical indicating whether the leave-one-out prediction standard errors should be returned. Defaults to `FALSE`.
`...`	Other arguments. Not used (needed for generic consistency).
`type`	The scale (`response` or `link`) of predictions obtained when `cv_predict = TRUE` and using `ssn_glm()` objects.

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 model-fit statistics are computed across all observations. Currently only computationally feasible for observed data matrices smaller than approximately 10,000.

Value

If `cv_predict = FALSE` and `se.fit = FALSE`, a tibble indicating several leave-one-out cross validation error metrics. If `cv_predict = TRUE` or `se.fit = TRUE`, a list with elements: `stats`, a tibble indicating several leave-one-out cross validation metrics; `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`).

If an `ssn_lm` object, the cross validation error metrics are:

- bias: The average difference between the predicted value and true value
- std.bias: The average standardized difference between the predicted value and true value
- MSPE: The average squared difference between the predicted value and true value
- RMSPE: The root average squared difference between the predicted value and true value
- std.MSPE: The average standardized squared difference between the predicted value and true value
- RAV: The root of the average estimated variance of the predicted value
- cor2: The squared correlation between the predicted and true values
- cover.80: Coverage rates of 80% prediction intervals built for the true values
- cover.90: Coverage rates of 90% prediction intervals built for the true values
- cover.95: Coverage rates of 95% prediction intervals built for the true values

If an `ssn_glm` object, the cross validation error metrics are:

- bias: The average difference between the predicted value and true value
- MSPE: The average squared difference between the predicted value and true value
- RMSPE: The root average squared difference between the predicted value and true value
- RAV: The root of the average estimated variance of the predicted value (on the link scale)

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
```

```
  tailup_type = "exponential",
  additive = "afvArea"
)
loocv(ssn_mod)
```

`mf04p` *Imported SSN object from the MiddleFork04.ssn data folder*

Description

The MiddleFork04.ssn data folder contains the spatial, attribute, and topological information needed to construct a spatial stream network object using the SSN2 package. `mf04p` was created using `ssn_import()`.

Usage

```
mf04p
```

Format

An object of class `SSN` of length 4.

See Also

MiddleFork04.ssn for details about the contents of `mf04p`. `ssn_import()` to convert a .ssn object to an `SSN` object in R. ssn_create_distmat for details about the distance matrix file structure.

`MiddleFork04.ssn` *MiddleFork04.ssn: Middle Fork 2004 stream temperature dataset*

Description

The `MiddleFork04.ssn` data folder contains the spatial, attribute, and topological information needed to construct an SSN object using the `SSN2` package.

Details

The `MiddleFork04.ssn` folder contains five geopackages:

- edges: geopackage with LINESTRING geometry representing the stream network
- sites: geopackage with POINT geometry representing the observed site locations
- pred1km: geopackage with POINT geometry prediction site locations at approximately 1km intervals throughout the stream network
- CapeHorn: geopackage with POINT geometry representing prediction site locations on the Cape Horn River

The MiddleFork04.ssn includes two text files, netID1.txt and netID2.txt, which contain the topological information for the two stream networks in the Middle Fork 2004 dataset.

The distance folder contains four folders that store the hydrologic distance matrices for each of the point datasets (obs, CapeHorn, and pred1km). See ssn_create_distmat() for a detailed description of the distance matrix file structure.

Attribute data is also stored within each of the spatial datasets. The column names are defined as follows:

edges:

- rid: Reach identifier
- COMID: Common identifier of an NHD feature or relationship
- GNIS_Name: Feature name as found in the Geographic Names Information System
- REACHCODE: Unique identifier for a reach. The first 8 digits contain the identifier for the HUC8 and the last 6 digits are a unique within-HUC8 identifier for the reach
- FTYPE: three-digit integer used to classify hydrography features in the NHD and define subtypes
- FCODE: Numeric code that contains the feature type and its attributes as found in the NHDFCode lookup table
- AREAWTMAP: Area weighted mean annual precipitation (mm) at the lowermost location on the edge
- SLOPE: Slope of the edge (cm/cm)
- rcaAreaKm2: Reach contributing area (km2) for each edge feature. The RCA represents the land area that drains directly to the edge feature
- h2oAreaKm2: Watershed area (km2) for the lowermost location on the edge feature
- areaPI: Segment proportional influence value, calculated using watershed area (h2oAreaKm2)
- afvArea: Additive function value, calculated using areaPI
- upDist: Distance from the stream outlet (most downstream location in the the stream network) to the uppermost location on the line segment
- Length: Length of line segment (m)
- netID: Network identifier

sites:

- rid: Reach identifier of the edge the site resides on
- pid: Point identifier
- STREAMNAME: Stream name
- COMID: Common identifier of an NHD feature or relationship
- AREAWTMAP: Area weighted mean annual precipitation (mm) at lowermost location on the line segment where the site resides
- SLOPE: Slope of the line segment (cm/cm) where the site resides
- ELEV_DEM: Elevation at the site based on a 30m DEM
- Source: Source of the data - relates to the ID field of the source table

- Summer_mn: Overall summer mean temperature (C) of the deployment
- MaxOver20: Binary variable: 1 represents the maximum summer temperature was greater than 20C and 0 indicates that it was less than 20C
- C16: Number of times daily stream temperature exceeded 16C
- C20: Number of times daily stream temperature exceeded 20C
- C24: Number of times daily stream temperature exceeded 24C
- FlowCMS: Average stream flow (cubic meters per sec) for August, by year, from 1950-2010 across 9 USGS gauges in the region
- AirMEANc: Average mean air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain
- AirMWMTc: Average maximum air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain. MWMT = maximum 7-day moving average of the maximum daily temperature (i.e. maximum of all the 7-day maximums)
- rcaAreaKm2: Reach contributing area (km2) for each edge feature the site resides on. The RCA represents the land area that drains directly to the edge feature
- h2oAreaKm2: Watershed area (km2) for the lowermost location on the edge feature the site resides on
- ratio: Site ratio value; provides the proportional distance from the downstream node of the edge to the site location
- upDist: Distance upstream from the stream outlet (m)
- afvArea: Additive function value calculated using watershed area (h2oAreaKm2)
- locID: Location identifier
- netID: Stream network identifier
- snapdist: Distance (m) that site was moved when it was snapped to the edges

`pred1km` and `CapeHorn`:

- rid: Reach identifier of the edge the site resides on
- pid: Point identifier
- COMID: Common identifier of an NHD feature or relationship
- GNIS_Name: Feature name of the edge the site resides on, as found in the Geographic Names Information System
- AREAWTMAP: Area weighted mean annual precipitation (mm) at lowermost location on the line segment where the site resides
- SLOPE: Slope of the line segment (cm/cm) where the site resides
- ELEV_DEM: Elevation at the site based on a 30m DEM
- rcaAreaKm2: Reach contributing area (km2) for each edge feature the site resides on. The RCA represents the land area that drains directly to the edge feature
- h2oAreaKm2: Watershed area (km2) for the lowermost location on the edge feature the site resides on
- ratio: Site ratio value; provides the proportional distance along the edge to the site location

- upDist: Distance upstream from the stream outlet (m)
- afvArea: Additive function value calculated using watershed area (h2oAreaKm2)
- locID: Location identifier
- netID: Stream network identifier
- snapdist: Distance (m) that site was moved when it was snapped to the edges
- FlowCMS: Average stream flow (cubic meters per sec) for August, by year, from 1950-2010 across 9 USGS gauges in the region
- AirMEANc: Average mean air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain
- AirMWMTc: Average maximum air temperature (C) from July 15 - August 31, from 1980-2009 across 10 COOP air stations within the domain. MWMT = maximum 7-day moving average of the maximum daily temperature(i.e. maximum of all the 7-day maximums)

Source

`edges` are a modified version of the United States National Hydrography Dataset (http://nhd.usgs.gov/). `sites`, `pred1km` and `CapeHorn` are unpublished United States Forest Service data.

See Also

mf04p for the Middle For 04 data as an `SSN` object.

`model.frame.SSN2` *Extract the model frame from a fitted model object*

Description

Extract the model frame from a fitted model object.

Usage

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

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

Arguments

`formula`	A fitted model object from `ssn_lm()` or `ssn_glm()`.
`...`	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

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
model.frame(ssn_mod)
```

---

model.matrix.SSN2 *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 'ssn_lm'
model.matrix(object, ...)

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

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().
...	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

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
model.matrix(ssn_mod)
```

---

plot.SSN2 *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, and Cook's distance vs leverage.

Usage

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

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

Arguments

x	A fitted model object from ssn_lm() or ssn_glm().
which	An integer vector taking on values between 1 and 6, 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 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

Value

No return value. Function called for plotting side effects.

See Also

`plot.Torgegram()`

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
plot(ssn_mod, which = 1)
```

`plot.Torgegram` *Plot Torgegram*

Description

Plot Torgegram

Usage

```
## S3 method for class 'Torgegram'
plot(x, type, separate = FALSE, ...)
```

Arguments

`x`	A Torgegram object from `Torgegram()`.
`type`	The type of semivariogram. Can take character values that are a subset of objects in `x`. The default is `names(x)`.
`separate`	When `type` is length greater than one, whether each `type` be placed in a separate plot. The default is `FALSE`.
`...`	Other arguments passed to other methods.

Value

No return value. Function called for plotting side effects.

See Also

plot.SSN2

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

tg <- Torgegram(Summer_mn ~ 1, mf04p)
plot(tg)
```

---

predict.SSN2 *Model predictions (Kriging)*

Description

Predicted values and intervals based on a fitted model object.

Usage

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

## S3 method for class 'ssn_glm'
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,
  ...
)
```

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().
newdata	A character vector that indicates the name of the prediction data set for which predictions are desired (accessible via object$ssn.object$preds). Note that the prediction data must be in the original SSN object used to fit the model. If newdata is omitted, predictions for all prediction data sets are returned. Note that the name ".missing" indicates the prediction data set that contains the missing observations in the data used to fit the model.
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).
level	Tolerance/confidence level. The default is 0.95.
type	The scale (response or link) of predictions obtained using ssn_glm objects.
block	A logical indicating whether a block prediction over the entire region in newdata should be returned. The default is FALSE, which returns point predictions for each location in newdata. Currently only available for model fit using ssn_lm() or models fit using ssn_glm() where family is "gaussian".
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 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). 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 = "covariance"`, the `size` data observations having the average highest covariance with the prediction locations are used. The default is `"covariance"`. Only used with models fit using `ssn_lm()`.
- `size`: The number of data observations to use when `method` is `"distance"` or `"covariance"`. The default is 4000. Only used with models fit using `ssn_lm()`.
- `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 = 4000, method = "covariance", parallel = FALSE)`.

`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. Not used (needed for generic consistency).

`dispersion` The dispersion of assumed when computing the prediction standard errors for `ssn_glm()` 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 `ssn_glm`. The default is `TRUE`.

`newdata_size` The `size` value for each observation in `newdata` used when predicting for the binomial family.

Details

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.

Value

If `se.fit` is `FALSE`, `predict.ssn()` 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

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, predpts = "pred1km", overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
predict(ssn_mod, "pred1km")
```

---

print.SSN *Print SSN object*

Description

Print information about the data found in an SSN object.

Usage

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

Arguments

`x`	An SSN object.
`...`	Other arguments. Not used (needed for generic consistency).

Value

Print summary to console

print.SSN2 *Print values*

Description

Print fitted model objects and summaries.

Usage

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

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

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

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

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

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

Arguments

x	A fitted model object from ssn_lm(), a fitted model object from ssn_glm(), or output from summary(x) or 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.

---

pseudoR2.SSN2 *Compute a pseudo r-squared*

Description

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

Usage

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

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

Arguments

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

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

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
pseudoR2(ssn_mod)
```

---

residuals.SSN2 *Extract fitted model residuals*

Description

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

Usage

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

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

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

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

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

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

Arguments

`object`	A fitted model object from `ssn_lm()` or `ssn_glm()`.

type	"response" for response residuals, "pearson" for Pearson residuals, or "standardized" for standardized residuals. For ssn_lm() fitted model objects, the default is "response". For ssn_glm() 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 ssn_lm() or ssn_glm().

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.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
residuals(ssn_mod)
resid(ssn_mod)
rstandard(ssn_mod)
```

---

ssn_create_bigdist *Calculate Hydrologic Distances for an* SSN *object*

Description

Creates a collection of (non-symmetric) matrices containing pairwise downstream hydrologic distances between sites in an SSN object

Usage

```
ssn_create_bigdist(
  ssn.object,
  predpts = NULL,
  overwrite = FALSE,
  among_predpts = FALSE,
  only_predpts = FALSE,
  no_cores = 1,
  verbose = TRUE
)
```

Arguments

ssn.object	An SSN object
predpts	name of prediction points in an SSN object. When a vector with length greater than one, each name is iterated upon. Default is NULL.
overwrite	Logical. If TRUE, overwrite existing distance matrices. Defaults to FALSE.
among_predpts	Logical. If TRUE, compute the pairwise distances between the prediction sites. Defaults to FALSE.
only_predpts	Logical. If TRUE, only compute distances for prediction sites. Defaults to FALSE.
no_cores	Number of cores to use in computation of the distance matrices. Also, the number of chunks to split the dataset into during computation.
verbose	Logical. If TRUE, warning messages (if they exist) are printed to the console. Defaults to TRUE.

Details

A distance matrix that contains the hydrologic distance between any two sites in SSN object is needed to fit a spatial statistical model using the tail-up and tail-down autocovariance functions described in Ver Hoef and Peterson (2010). These models are implemented in R via ssn_lm and ssn_glm in theSSN2 package. The hydrologic distance information needed to model the covariance between flow-connected (i.e. water flows from one location to the other) and flow-unconnected (i.e. water does not flow from one location to the other, but they reside on the same network) locations differs. The total hydrologic distance is a directionless measure; it represents the hydrologic distance between two sites, ignoring flow direction. The hydrologic distance from each site to a common downstream stream junction is used when creating models for flow-unconnected pairs, which we term downstream hydrologic distance. In contrast, the total hydrologic distance is used for modeling flow-connected pairs, which we term total hydrologic distance.

A downstream hydrologic distance matrix provides enough information to meet the data requirements for both the tail-up and tail-down models. When two locations are flow-connected, the downstream hydrologic distance from the upstream location to the downstream location is greater than zero, but it is zero in the other direction. When two locations are flow-unconnected the downstream hydrologic distance will be greater than zero in both directions. A site's downstream hydrologic distance to itself is equal to zero. The format of the downstream hydrologic distance matrix is efficient because distance information needed to fit both the tail-up and tail-down models is only stored once. As an example, a matrix containing the total hydrologic distance between sites is easily calculated by adding the downstream distance matrix to its transpose.

The downstream hydrologic distances are calculated based on the binaryIDs and stored as matrices. The matrices are stored in a directory named 'distance', which is created by the `ssn_create_distmat` function within the .ssn directory. The distance directory will always contain at least one directory named 'obs', which contains a number of .RData files, one for each network that has observed sites residing on it. The naming convention for the files is based on the netID number (e.g. dist.net1.RData). Each matrix in the 'obs' folder contains the information to form a square matrix, which contains the downstream hydrologic distance between each pair of observed sites on the network. Direction is preserved, with columns representing the FROM site and rows representing the TO site. Row and column names correspond to the pid attribute for each site.

If the argument `predpts` is specified in the call to the function, the downstream hydrologic distances between the observed and prediction sites will also be computed. A new directory is created within the distance directory, with the name corresponding to the names attribute for the preds (e.g. `attributes(ssn.object$preds)$names`). A sequence of .RData files is created within this directory, similar to the structure for the observed sites, except that two objects are stored for each network that contains *both* observed and prediction sites. The letters `a` and `b` are used in the naming convention to distinguish between the two objects (e.g. dist.net1.a and dist.net1.b). The matrices that these objects represent are not necessarily square. In matrices of type `a`, rows correspond to observed locations and columns to prediction locations. In contrast, rows correspond to prediction locations and columns to observed locations in matrices of type `b`. Direction is also preserved, with columns representing the FROM site and rows representing the TO site in both object types. Again, row and column names correspond to the pid attribute for each site.

If `among_predpts = TRUE`, the downstream hydrologic distances will also be computed between prediction sites, for each network. Again these are stored within the distance directory with the name corresponding to the prediction points dataset. The naming convention for these prediction to prediction site distance matrices is the same as the distance matrices stored in the 'obs' directory (e.g. dist.net1.RData). These extra distance matrices are needed to perform block Kriging using `predict.ssn_lm`.

If `only_predpts = TRUE`, the downstream hydrologic distances will not be calculated between observed sites themselves. Pairwise distances will only be calculated for observed and prediction locations and. Pairwise distances between prediction locations will also be calculated if `among_predpts = TRUE`.

Value

The `ssn_create_distmat` function creates a collection of hierarchical directories in the `ssn$path` directory, which store the pairwise distances between sites associated with the `SSN` object. See details section for additional information.

Examples

```
## Copy the MiddleForke04.ssn data to a local temporary directory.
## Only needed for this example.
copy_lsn_to_temp()
## Import SSN data
mf04p <- ssn_import(paste0(tempdir(), "/MiddleFork04.ssn"),
  predpts = c("pred1km.gpkg", "CapeHorn"),
  overwrite = TRUE
)
```

```
## Create distance matrices for observations and one set of prediction sites
## Include hydrologic distance matrices among prediction sites.
ssn_create_distmat(mf04p,
  predpts = "pred1km", overwrite = TRUE,
  among_predpts = TRUE
)

## Create distance matrices for an additional set of prediction points.
## Distance matrices for observations and pred1km prediction sites are
## not recalculated.
ssn_create_distmat(mf04p,
  predpts = "CapeHorn", overwrite = TRUE,
  among_predpts = TRUE, only_predpts = TRUE
)
```

---

ssn_create_distmat *Calculate Hydrologic Distances for an* SSN *object*

Description

Creates a collection of (non-symmetric) matrices containing pairwise downstream hydrologic distances between sites in an SSN object

Usage

```
ssn_create_distmat(
  ssn.object,
  predpts = NULL,
  overwrite = FALSE,
  among_predpts = FALSE,
  only_predpts = FALSE
)
```

Arguments

ssn.object	An SSN object
predpts	name of prediction points in an SSN object. When a vector with length greater than one, each name is iterated upon. Default is NULL.
overwrite	Logical. If TRUE, overwrite existing distance matrices. Defaults to FALSE.
among_predpts	Logical. If TRUE, compute the pairwise distances between the prediction sites. Defaults to FALSE.
only_predpts	Logical. If TRUE, only compute distances for prediction sites. Defaults to FALSE.

Details

A distance matrix that contains the hydrologic distance between any two sites in SSN object is needed to fit a spatial statistical model using the tail-up and tail-down autocovariance functions described in Ver Hoef and Peterson (2010). These models are implemented in R via ssn_lm and ssn_glm in theSSN2 package. The hydrologic distance information needed to model the covariance between flow-connected (i.e. water flows from one location to the other) and flow-unconnected (i.e. water does not flow from one location to the other, but they reside on the same network) locations differs. The total hydrologic distance is a directionless measure; it represents the hydrologic distance between two sites, ignoring flow direction. The hydrologic distance from each site to a common downstream stream junction is used when creating models for flow-unconnected pairs, which we term downstream hydrologic distance. In contrast, the total hydrologic distance is used for modeling flow-connected pairs, which we term total hydrologic distance.

A downstream hydrologic distance matrix provides enough information to meet the data requirements for both the tail-up and tail-down models. When two locations are flow-connected, the downstream hydrologic distance from the upstream location to the downstream location is greater than zero, but it is zero in the other direction. When two locations are flow-unconnected the downstream hydrologic distance will be greater than zero in both directions. A site's downstream hydrologic distance to itself is equal to zero. The format of the downstream hydrologic distance matrix is efficient because distance information needed to fit both the tail-up and tail-down models is only stored once. As an example, a matrix containing the total hydrologic distance between sites is easily calculated by adding the downstream distance matrix to its transpose.

The downstream hydrologic distances are calculated based on the binaryIDs and stored as matrices. The matrices are stored in a directory named 'distance', which is created by the ssn_create_distmat function within the .ssn directory. The distance directory will always contain at least one directory named 'obs', which contains a number of .RData files, one for each network that has observed sites residing on it. The naming convention for the files is based on the netID number (e.g. dist.net1.RData). Each matrix in the 'obs' folder contains the information to form a square matrix, which contains the downstream hydrologic distance between each pair of observed sites on the network. Direction is preserved, with columns representing the FROM site and rows representing the TO site. Row and column names correspond to the pid attribute for each site.

If the argument predpts is specified in the call to the function, the downstream hydrologic distances between the observed and prediction sites will also be computed. A new directory is created within the distance directory, with the name corresponding to the names attribute for the preds (e.g. attributes(ssn.object$preds)$names). A sequence of .RData files is created within this directory, similar to the structure for the observed sites, except that two objects are stored for each network that contains *both* observed and prediction sites. The letters a and b are used in the naming convention to distinguish between the two objects (e.g. dist.net1.a and dist.net1.b). The matrices that these objects represent are not necessarily square. In matrices of type a, rows correspond to observed locations and columns to prediction locations. In contrast, rows correspond to prediction locations and columns to observed locations in matrices of type b. Direction is also preserved, with columns representing the FROM site and rows representing the TO site in both object types. Again, row and column names correspond to the pid attribute for each site.

If among_predpts = TRUE, the downstream hydrologic distances will also be computed between prediction sites, for each network. Again these are stored within the distance directory with the name corresponding to the prediction points dataset. The naming convention for these prediction to prediction site distance matrices is the same as the distance matrices stored in the 'obs' directory (e.g. dist.net1.RData). These extra distance matrices are needed to perform block Kriging using

predict.ssn_lm.

If only_predpts = TRUE, the downstream hydrologic distances will not be calculated between observed sites themselves. Pairwise distances will only be calculated for observed and prediction locations and. Pairwise distances between prediction locations will also be calculated if among_predpts = TRUE.

Value

The ssn_create_distmat function creates a collection of hierarchical directories in the ssn$path directory, which store the pairwise distances between sites associated with the SSN object. See details section for additional information.

Examples

```
## Copy the MiddleForke04.ssn data to a local temporary directory.
## Only needed for this example.
copy_lsn_to_temp()
## Import SSN data
mf04p <- ssn_import(paste0(tempdir(), "/MiddleFork04.ssn"),
  predpts = c("pred1km.gpkg", "CapeHorn"),
  overwrite = TRUE
)

## Create distance matrices for observations and one set of prediction sites
## Include hydrologic distance matrices among prediction sites.
ssn_create_distmat(mf04p,
  predpts = "pred1km", overwrite = TRUE,
  among_predpts = TRUE
)

## Create distance matrices for an additional set of prediction points.
## Distance matrices for observations and pred1km prediction sites are
## not recalculated.
ssn_create_distmat(mf04p,
  predpts = "CapeHorn", overwrite = TRUE,
  among_predpts = TRUE, only_predpts = TRUE
)
```

---

ssn_get_data *Get a data.frame from an SSN, ssn_lm, or ssn_glm object*

Description

The ssn_get_data function extracts an sf data.frame for the observation or prediction data from an SSN, ssn_lm, or ssn_glm object.

Usage

```
ssn_get_data(x, name = "obs")
```

Arguments

x	An object of class SSN, ssn_lm, or ssn_glm.
name	the internal name of the dataset in the object x. For observed values, this will always be "obs", the default.

Details

The internal name for observed data in objects of class SSN is "obs" and it is the default. If another name is specified, it must represent a prediction data set in the SSN, ssn_lm, or ssn_glm object. For SSN objects, these names are obtained using the call names(x$preds). For all other object classes, the names are obtained using the call names(x$ssn.object$preds).

Value

An sf data.frame

See Also

ssn_put_data()

Examples

```
## Extract observed data from an SSN object
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, predpts = "pred1km", overwrite = TRUE)

obs.df <- ssn_get_data(mf04p)
dim(obs.df)

## Extract prediction data from an SSN object
names(mf04p$preds)
pred1km.df <- ssn_get_data(mf04p, name = "pred1km")
names(pred1km.df)

## extract observed data from an ssn_lm object
ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
obs.mod.df <- ssn_get_data(ssn_mod)
summary(obs.mod.df)
```

ssn_get_netgeom *Extract netgeom column*

Description

Extract topological information from netgeom column

Usage

```
ssn_get_netgeom(x, netvars = "all", reformat = FALSE)
```

Arguments

x	An sf data.frame found in an SSN object or the netgeom column as a vector
netvars	Network coordinate variables to return. Default is "all". For edges, valid column names include: "NetworkID", "SegmentID", and "DistanceUpstream". For point datasets, valid column names include "NetworkID", "SegmentID", "DistanceUpstream", "ratio", "pid", and "locID".
reformat	Convert network coordinate variables from character to numeric.

Details

When an SSN object is generated using the importSSN function, a text column named "netgeom" is added to the edges, observed sites, and prediction sites (if they exist) data.frames. The netgeom column contains data used to describe how edge and site features relate to one another in topological space. For edges, netgeom values contain the "ENETWORK" prefix, with 3 space delimited values in parentheses: "ENETWORK (NetworkID SegmentID DistanceUpstream)". For point datasets (observed and prediction sites), the values contain the "SNETWORK" prefix, followed by 6 space delimited values in parentheses: "SNETWORK (NetworkID SegmentID DistanceUpstream ratio pid locID)". The ssn_get_netgeom function extracts and converts these values from text to numeric, returning either a data.frame (default) or vector containing the variables requested via netvars.

Value

If more than one column is requested using netvars, the function returns a data.frame (default). If only one column is requested, the result is a vector.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_get_netgeom(mf04p$obs)
ssn_get_netgeom(mf04p$edges, "DistanceUpstream")
```

ssn_get_stream_distmat *Get stream distance matrices from an* SSN *object*

Description

Extracts the stream network distance matrices for the observation or prediction data from an SSN object.

Usage

```
ssn_get_stream_distmat(x, name = "obs")
```

Arguments

x	An SSN object
name	Internal name of the dataset in the object x. For observed values, this will always be "obs", the default. To get a stream network distance matrix for a prediction data set, the name of the dataset must be given, in quotes.

Details

ssn_get_stream_distmat() extracts the stream network distance matrices created using ssn_create_distmat, which are stored as .Rdata files. The internal name for observed data in objects of class SSN is "obs" and it is the default. If another name is specified, it must represent a prediction data set in the SSN object. For SSN objects, these names are obtained using the call names(x$preds).

Note that these are not traditional symmetric distance matrices. First, distances in an SSN object represent stream distance, or hydrologic distance, which is the distance between two locations when movement is restricted to the branching stream network. Another important difference is the distance matrices for SSN objects contain the *downstream only* stream distance between two locations, making them asymmetric. This asymmetry provides a way to store two types of spatial relationships based on stream distance:

- Flow-connected: Water flows from an upstream site to a downstream site.
- Flow-unconnected: Two sites reside on the same stream network, but do not share flow.

For example, if two sites are flow-connected the downstream distance from the upstream site to the downstream site is > 0, while the downstream distance between the downstream site and the upstream site = 0. For flow-unconnected sites, the downstream distance represents the distance from each site to the closest downstream junction and will be > 0 in both directions. Direction is preserved, with columns representing the FROM site and rows representing the TO site. Row and column names correspond to the unique point identifier "pid" for each site. From this matrix, it is also possible to get total stream distance (downstream + upstream) between any two sites on the same network (see examples for additional details).

Stream distances are only calculated within a network and so the asymmetric matrices are also stored by network. For observation data, a single square matrix of distances is returned for each

network, with the names based on the netID value (e.g. "dist.net1", "dist.net2", etc.). However, two distance matrices ("a" and "b") are required to store the downstream only distance between observed and prediction sites. The label "a" represents the downstream stream distance *from* prediction sites *to* observation sites, and the label "b" represents the distance *from* observation sites *to* predictions sites. Thus, the list of prediction matrices are labeled "dist.net1.a" for the downstream only distance from prediction sites in the columns, to observation sites in the rows, for the first network. A prediction matrix labeled "dist.net1.b" contains downstream distances *from* observation sites in the columns *to* prediction sites in the rows, for the first network. The downstream only distance matrices for observations and predictions will be rectangular, unless the number of observation and prediction locations are equal. If the argument `amongPreds = TRUE` was used in the function `ssn_create_distmat`, then the distance between prediction sites themselves is also returned, using the same labelling convention as for among observation sites. That is, the matrices for each network will be labeled "dist.net1", "dist.net2", etc., for the first and second network, etc.

Value

A `list` of asymmetric downstream only stream distance matrices, by network.

References

Ver Hoef, J.M. and Peterson, E.E. (2010) A moving average approach to spatial statistical models of stream networks. The Journal of the American Statistical Association, **105(489)**, 22–24

See Also

`ssn_create_distmat()`

Examples

```
## For this example only, copy MiddleFork04.ssn directory to R's
## temporary directory
copy_lsn_to_temp()
## Create an SSN object with prediction sites
mf04p <- ssn_import(paste0(tempdir(), "/MiddleFork04.ssn"),
  predpts = "pred1km", overwrite = TRUE
)

## Create distance matrices for obs x obs, obs x preds, and preds x
## preds
## Not run:
ssn_create_distmat(mf04p,
  predpts = "pred1km", among_predpts = TRUE,
  overwrite = TRUE
)

## End(Not run)

## Check names of prediction datasets
names(mf04p$preds)

## Get list of stream distance matrices for observations
```

```
dist_obs <- ssn_get_stream_distmat(mf04p)
## Display structure of list and names of the matrices
str(dist_obs)
names(dist_obs)
## Look at first 5 rows and columns in asymmetric
## downstream only distance matrix for netID == 1
dist_obs$dist.net1[1:5, 1:5]

## Create symmetric total stream distance matrix between
## observations
strdist_2 <- dist_obs$dist.net2 + t(dist_obs$dist.net2)
strdist_2[5:10, 5:10]

## Get maximum downstream only distance between
## observations on netID == 2
a.mat <- pmax(dist_obs$dist.net2, t(dist_obs$dist.net2))
a.mat[5:10, 5:10]

## Get minimum downstream only distance between observations. If
## minimum distance == 0, sites are flow-connected
b.mat <- pmin(dist_obs$dist.net2, t(dist_obs$dist.net2))
b.mat[5:10, 5:10]

## Get distance matrices for pred1km
dist_pred1km <- ssn_get_stream_distmat(mf04p, name = "pred1km")
str(dist_pred1km)
names(dist_pred1km)
## Look at first 5 rows and columns of downstream only distances
## FROM prediction sites TO observed sites on netID == 1
dist_pred1km$dist.net1.a[1:5, 1:5]

## Look at downstream only stream distances among prediction
## sites in pred1km on netID == 1. This is useful for block
## prediction
dist_pred1km$dist.net1[1:5, 1:5]
```

`ssn_glm` *Fitting Generalized Linear Models for Spatial Stream Networks*

Description

This function works on spatial stream network objects to fit generalized linear models with spatially autocorrelated errors using likelihood methods, allowing for non-spatial random effects, anisotropy, partition factors, big data methods, and more. The spatial formulation is described in Ver Hoef and Peterson (2010) and Peterson and Ver Hoef (2010).

Usage

```
ssn_glm(
  formula,
```

```
  ssn.object,
  family,
  tailup_type = "none",
  taildown_type = "none",
  euclid_type = "none",
  nugget_type = "nugget",
  tailup_initial,
  taildown_initial,
  euclid_initial,
  nugget_initial,
  dispersion_initial,
  additive,
  estmethod = "reml",
  anisotropy = FALSE,
  random,
  randcov_initial,
  partition_factor,
  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.
ssn.object	A spatial stream network object with class SSN.
family	The generalized linear model family for use with ssn_glm(). Available options include "Gaussian", "poisson", "nbinomial" (negative binomial), "binomial", "beta", "Gamma", and "invgauss". When family is "Gaussian", arguments are passed to and evaluated by ssn_lm(). 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.
tailup_type	The tailup covariance function type. Available options include "linear", "spherical", "exponential", "mariah", "epa", "gaussian", and "none". Parameterizations are described in Details.
taildown_type	The taildown covariance function type. Available options include "linear", "spherical", "exponential", "mariah", "epa", "gaussian", and "none". Parameterizations are described in Details.
euclid_type	The euclidean covariance function type. Available options include "spherical", "exponential", "gaussian", "cosine", "cubic", "pentaspherical", "wave", "jbessel", "gravity", "rquad", "magnetic", and "none". Parameterizations are described in Details.
nugget_type	The nugget covariance function type. Available options include "nugget" or "none". Parameterizations are described in Details.
tailup_initial	An object from tailup_initial() specifying initial and/or known values for the tailup covariance parameters.

taildown_initial
: An object from taildown_initial() specifying initial and/or known values for the taildown covariance parameters.

euclid_initial
: An object from euclid_initial() specifying initial and/or known values for the euclidean covariance parameters.

nugget_initial
: An object from nugget_initial() specifying initial and/or known values for the nugget covariance parameters.

dispersion_initial
: An object from dispersion_initial() specifying initial and/or known values for the tailup covariance parameters.

additive
: The name of the variable in ssn.object that is used to define spatial weights. Can be quoted or unquoted. For the tailup covariance functions, these additive weights are used for branching. Technical details that describe the role of the additive variable in the tailup covariance function are available in Ver Hoef and Peterson (2010).

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. See spmodel::randcov_initial().

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. local can only be used when big data distance matrices have been created using ssn_create_bigdist() and is most beneficial when the sample size is at least 5,000 ssn_lm() or 3,000 ssn_glm(). 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 200.
- 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 = 200, method = "kmeans", var_adjust = "theoretical", parallel = FALSE).

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

Details

The generalized linear model for spatial stream networks can be written as $g(\mu) = \eta = X\beta + zu + zd + ze + n$, where μ is the expectation of the response given the random errors, y, $g()$ is a function that links the mean and η (and is called a link function), X is the fixed effects design matrix, β are the fixed effects, zu is tailup random error, zd is taildown random error, and ze is Euclidean random error, and n is nugget random error.

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
- Gaussian: $-\infty < y < \infty$
- 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
- Gaussian: $g(\mu) = \eta$ (identity link)
- 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: $Var(y)$
- Gaussian: σ^2
- 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).

In the generalized linear model context, the tailup, taildown, Euclidean, and nugget covariance affect the modeled mean of an observation (conditional on these effects). On the link scale, the tailup random errors capture spatial covariance moving downstream (and depend on downstream

distance), the taildown random errors capture spatial covariance moving upstream (and depend on upstream) distance, the Euclidean random errors capture spatial covariance that depends on Euclidean distance, and the nugget random errors captures variability independent of spatial locations. η is modeled using a spatial covariance function expressed as $de(zu) * R(zu) + de(zd) * R(zd) + de(ze) * R(ze) + nugget * I$. $de(zu)$, $de(zu)$, and $de(zd)$ represent the tailup, taildown, and Euclidean variances, respectively. $R(zu)$, $R(zd)$, and $R(ze)$ represent the tailup, taildown, and Euclidean correlation matrices, respectively. Each correlation matrix depends on a range parameter that controls the distance-decay behavior of the correlation. $nugget$ represents the nugget variance and I represents an identity matrix.

`tailup_type` Details: Let D be a matrix of hydrologic distances, W be a diagonal matrix of weights from `additive`, $r = D/range$, and I be an identity matrix. Then parametric forms for flow-connected elements of $R(zu)$ are given below:

- linear: $(1 - r) * (r <= 1) * W$
- spherical: $(1 - 1.5r + 0.5r^3) * (r <= 1) * W$
- exponential: $exp(-r) * W$
- mariah: $log(90r + 1)/90r * (D > 0) + 1 * (D = 0) * W$
- epa: $(D - range)^2 * F * (r <= 1) * W/16range^5$
- gaussian: $2exp(-r^2) * (1 - pnorm(r * 2^{1/2})) * W$
- none: I * W

Details describing the `F` matrix in the `epa` covariance are given in Garreta et al. (2010). Flow-unconnected elements of $R(zu)$ are assumed uncorrelated. Observations on different networks are also assumed uncorrelated.

`taildown_type` Details: Let D be a matrix of hydrologic distances, $r = D/range$, and I be an identity matrix. Then parametric forms for flow-connected elements of $R(zd)$ are given below:

- linear: $(1 - r) * (r <= 1)$
- spherical: $(1 - 1.5r + 0.5r^3) * (r <= 1)$
- exponential: $exp(-r)$
- mariah: $log(90r + 1)/90r * (D > 0) + 1 * (D = 0)$
- epa: $(D - range)^2 * F1 * (r <= 1)/16range^5$
- gaussian: 0
- none: I

Now let A be a matrix that contains the shorter of the two distances between two sites and the common downstream junction, $r1 = A/range$, B be a matrix that contains the longer of the two distances between two sites and the common downstream junction, $r2 = B/range$, and I be an identity matrix. Then parametric forms for flow-unconnected elements of $R(zd)$ are given below:

- linear: $(1 - r2) * (r2 <= 1)$
- spherical: $(1 - 1.5r1 + 0.5r2) * (1 - r2)^2 * (r2 <= 1)$
- exponential: 0
- mariah: $(log(90r1+1) - log(90r2+1))/(90r1 - 90r2) * (A = /B) + (1/(90r1+1)) * (A = B)$
- epa: $(B - range)^2 * F2 * (r2 <= 1)/16range^5$

- gaussian: $2exp(-(B-A)/range) * (1 - pnorm(r * 2^{1/2})) * W$
- none: I

Details describing the `F1` and `F2` matrices in the `epa` covariance are given in Garreta et al. (2010). Observations on different networks are assumed uncorrelated.

`euclid_type` Details: Let D be a matrix of Euclidean distances, $r = D/range$, and I be an identity matrix. Then parametric forms for elements of $R(ze)$ are given below:

- exponential: $exp(-r)$
- spherical: $(1 - 1.5r + 0.5r^3) * (r <= 1)$
- gaussian: $exp(-r^2)$
- cubic: $(1 - 7r^2 + 8.75r^3 - 3.5r^5 + 0.75r^7) * (r <= 1)$
- pentaspherical: $(1 - 1.875r + 1.25r^3 - 0.375r^5) * (r <= 1)$
- cosine: $cos(r)$
- wave: $sin(r) * (h > 0)/r + (h = 0)$
- jbessel: $Bj(h * range)$, Bj is Bessel-J function
- gravity: $(1 + r^2)^{-0.5}$
- rquad: $(1 + r^2)^{-1}$
- magnetic: $(1 + r^2)^{-1.5}$
- none: I

`nugget_type` Details: Let I be an identity matrix and 0 be the zero matrix. Then parametric forms for elements the nugget variance are given below:

- nugget: I
- none: 0

In short, the nugget effect is modeled when `nugget_type` is `"nugget"` and omitted when `nugget_type` is `"none"`.

`estmethod` Details: The various estimation methods are

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

`anisotropy` Details: By default, all Euclidean covariance parameters except `rotate` and `scale` 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 `euclid_initial()`, `anisotropy` is implicitly set to `TRUE`. (Geometric) Anisotropy is modeled by transforming a Euclidean covariance function that decays differently in different directions to one that decays equally in all directions via rotation and scaling of the original Euclidean 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 Euclidean 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 $g(\mu) = \eta = X\beta + W1\gamma 1 + ...Wj\gamma j + zu + zd + ze + n$, 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.

Other Details: 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 and has class ssn_glm. Many generic functions that summarize model fit are available for ssn_glm objects, including AIC, AICc, anova, augment, 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.

This fitted model list contains the following elements:

- additive: The name of the additive function value column.
- anisotropy: Whether euclidean anisotropy was modeled.
- call: The function call.
- coefficients: Model coefficients.
- contrasts: Any user-supplied contrasts.
- cooks_distance: Cook's distance values.
- crs: The geographic coordinate reference system.
- deviance: The model deviance.
- diagtol: A tolerance value that may be added to the diagonal of covariance matrices to encourage decomposition stability.
- estmethod: The estimation method.
- euclid_max: The maximum euclidean distance.
- family: The generalized linear model family
- fitted: Fitted values.
- formula: The model formula.
- hatvalues: The hat (leverage) values.
- is_known: An object that identifies which parameters are known.
- local_index: An index identifier used internally for sorting.
- missing_index: Which rows in the "obs" object had missing responses.
- n: The sample size.
- npar: The number of estimated covariance parameters.
- observed_index: Which rows in the "obs" object had observed responses.
- optim: The optimization output.

- p: The number of fixed effects.
- partition_factor: The partition factor formula.
- pseudoR2: The pseudo R-squared.
- random: The random effect formula.
- residuals: The residuals.
- sf_column_name: The name of the geometry columns ssn.object
- size: The size of the binomial trials if relevant.
- ssn.object: An updated ssn.object.
- tail_max: The maximum stream distance.
- terms: The model terms.
- vcov: Variance-covariance matrices
- xlevels: The levels of factors in the model matrix.
- y: The response.

These list elements are meant to be used with various generic functions (e.g., residuals() that operate on the model object. While possible to access elements of the fitted model list directly, we strongly advise against doing so when there is a generic available to return the element of interest. For example, we strongly recommend using residuals() to obtain model residuals instead of accessing the fitted model list directly via object$residuals.

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

Garreta, V., Monestiez, P. and Ver Hoef, J.M. (2010) Spatial modelling and prediction on river networks: up model, down model, or hybrid? *Environmetrics* **21(5)**, 439–456.

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

Peterson, E.E. and Ver Hoef, J.M. (2010) A mixed-model moving-average approach to geostatistical modeling in stream networks. *Ecology* **91(3)**, 644–651.

Ver Hoef, J.M. and Peterson, E.E. (2010) A moving average approach for spatial statistical models of stream networks (with discussion). *Journal of the American Statistical Association* **105**, 6–18. DOI: 10.1198/jasa.2009.ap08248. Rejoinder pgs. 22–24.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)
```

```
ssn_gmod <- ssn_glm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  family = "Gamma",
  tailup_type = "exponential",
  additive = "afvArea"
)
summary(ssn_gmod)
```

---

ssn_import *Import* SSN *object*

Description

This function reads spatial data from a .ssn folder and creates an SSN object.

Usage

```
ssn_import(
  path,
  include_obs = TRUE,
  predpts = NULL,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

path	Filepath to the .ssn directory. See details.
include_obs	default = TRUE. Logical indicating whether observed sites should be included in the SSN object.
predpts	Vector of prediction site dataset names found within the .ssn folder. See details.
overwrite	default = FALSE. If TRUE, overwrite existing binaryID.db files and netgeom column(s) if they exist in the edges, observed sites (if include_obs = TRUE), and prediction site datasets (if they exist).
verbose	default = TRUE. If FALSE, warning messages will not be printed to the console.

Details

The ssn_import function imports spatial data (shapefile or GeoPackage format) from a .ssn folder generated using the SSNbler package function SSNbler::lsn_to_ssn. The .ssn folder contains all of the spatial, topological and attribute data needed to fit a spatial statistical stream network model to streams data. This includes:

- An edges dataset with LINESTRING geometry representing the stream network.

- A sites dataset with POINT geometry where observed data were collected on the stream network.
- Prediction sites dataset(s) representing locations where predictions will be made.
- netID.dat file(s) for each distinct network, which stores the topological relationships of the line features in edges.

A more detailed description of the .ssn directory and its contents is provided in Peterson and Ver Hoef (2014).

The `ssn_import` imports the edges, observed sites (optional), and prediction sites (optional) as `sf data.frame` objects. A new column named 'netgeom' is created to store important data representing topological relationships in a spatial stream network model. These data are stored in character format, which is less likely to be inadvertently changed by users. See `create_netgeom` for a more detailed description of the format and contents of 'netgeom'.

The information contained in the netID text files is imported into an SQLite database, binaryID.db, which is stored in the .ssn directory. This information is used internally by `ssn_create_distmat`, `ssn_lm` and `ssn_glm` to calculate the data necessary to fit a spatial statistical model to stream network data. If `overwrite = TRUE` (`overwrite = FALSE` is the default) and a binaryID.db file already exists within the .ssn directory, it will be overwritten when the `SSN` object is created. If a 'netgeom' column exists in any of the input datasets (e.g. edges, observed sites, predictions sites) and `overwrite = TRUE`, it will be overwriten.

At a minimum, an `SSN` object must always contain streams, which are referred to as edges. The `SSN` object would also typically contain a set of observed sites, where measurements have been collected. Only one observed dataset is permitted in an `SSN` object. When `include_obs=FALSE`, an `SSN` object is created without observations. This option provides flexibility for users who would like to simulate data on a set of artificial sites on an existing stream network. Note that observation sites must be included in the `SSN` object in order to fit models using `ssn_lm` or `ssn_glm`. The `SSN` object may contain multiple sets of prediction points (or none), which are stored as separate datasets in the .ssn directory. If `predpts` is a named vector, the names of the `preds` list in the `SSN` object correspond to the vector names. Otherwise, they are set to the basename of the prediction site dataset file(s) specified in `predpts`. The `ssn_import_predpts` function allows users to import additional sets of prediction sites to a an existing `SSN` object.

Value

`ssn_import` returns an object of class SSN, which is a list with four elements containing:

- `edges`: An `sf data.frame` containing the stream network, with an additional 'netgeom' column.
- `obs`: An sf data.frame containing observed site locations, with an additional 'netgeom' column. NA if `include_obs = FALSE`.
- `preds`: A list of sf data.frames containing prediction site locations. An empty list is returned if `predpts` is not provided.
- path: The local filepath for the .ssn directory associated with the `SSN` object.

References

Peterson, E., and Ver Hoef, J.M. (2014) STARS: An ArcGIS toolset used to calculate the spatial information needed to fit spatial statistical stream network models to stream network data. *Journal of Statistical Software* **56(2)**, 1–17.

Examples

```
## Create local temporary copy of MiddleFork04.ssn found in
# SSN2/lsndata folder. Only necessary for this example.
copy_lsn_to_temp()

## Import SSN object with no prediction sites
mf04 <- ssn_import(paste0(tempdir(), "/MiddleFork04.ssn"),
  overwrite = TRUE
)

## Import SSN object with 3 sets of prediction sites
mf04p <- ssn_import(paste0(tempdir(), "/MiddleFork04.ssn"),
  predpts = c(
    "pred1km",
    "CapeHorn"
  ),
  overwrite = TRUE
)
```

---

ssn_import_predpts *Import prediction points into an SSN, ssn_lm, or ssn_glm object*

Description

A shapefile of prediction points found in the .ssn directory are imported into an existing object of class SSN, ssn_lm, or ssn_glm.

Usage

```
ssn_import_predpts(x, predpts, overwrite = FALSE, verbose = TRUE)
```

Arguments

x	An object of classSSN, ssn_lm, or ssn_glm.
predpts	Name of the prediction point dataset to import in character format. See details.
overwrite	default = FALSE. If TRUE, overwrite existing netgeom column(s) if they exist in the prediction site dataset.
verbose	default = TRUE. If FALSE, warning messages will not be printed to the console.

Details

ssn_import_predpts imports one set of prediction points residing in the .ssn directory into an existing SSN, ssn_lm, or ssn_glm object. The prediction dataset must be in shapefile or geopackage format (.shp or .gpkg, respectively) and reside in the ssn.object$path directory. The path for an SSN object can be updated using ssn_update_path() prior to importing prediction datasets. The argument predpts accepts the name of the prediction point dataset, with or without the file extension. If it is passed as a named vector (of length 1), then the name provided is used as the prediction dataset name in the SSN object prediction sites list (e.g. names(ssn.obj$preds)). Otherwise, the file basename is used in the names attribute. See ssn_import for a detailed description of the prediction dataset format within the SSN class object.

When the prediction dataset is imported, a new column named netgeom is created. If this column already exists it is overwritten. Please see create_netgeom for a detailed description of the netgeom column and the information it contains.

The prediction dataset specified in predpts must contain the spatial, topological and attribute information needed to make predictions using an ssn_lm or ssn_glm object. This information is generated using the SSNbler package, which makes use of the functionality found in the sf and igraph packages to process streams data in vector format.

Value

an object of class SSN, ssn_lm, or ssn_glm which contains the new prediction dataset.

Examples

```
## Create local temporary copy of MiddleFork04.ssn found in
# SSN2/lsndata folder. Only necessary for this example.
copy_lsn_to_temp()

## Import SSN object with no prediction sites
mf04p <- ssn_import(paste0(tempdir(), "/MiddleFork04.ssn"),
  overwrite = TRUE
)

## Import pred1km prediction dataset into SSN object and assign the
## name preds1
mf04p <- ssn_import(paste0(tempdir(), "/MiddleFork04.ssn"),
    overwrite = TRUE)
mf04p <- ssn_import_predpts(mf04p, predpts = c(preds1 = "pred1km"))
names(mf04p$preds)

## Import CapeHorn prediction dataset into a ssn_glm object, using
## the default file basename as the name
ssn_gmod <- ssn_glm(Summer_mn ~ netID, mf04p,
  family = "Gamma",
  tailup_type = "exponential", additive = "afvArea"
)
ssn_gmod <- ssn_import_predpts(ssn_gmod, predpts = "CapeHorn")
names(ssn_gmod$ssn.object$preds)
```

ssn_initial *Create a covariance parameter initial object*

Description

Create a covariance parameter initial object that specifies initial and/or known values to use while estimating specific covariance parameters with `ssn_lm()` or `ssn_glm()`. See `spmodel::randcov_initial()` for documentation regarding random effect covariance parameter initial objects.

Usage

```
tailup_initial(tailup_type, de, range, known)

taildown_initial(taildown_type, de, range, known)

euclid_initial(euclid_type, de, range, rotate, scale, known)

nugget_initial(nugget_type, nugget, known)
```

Arguments

`tailup_type`	The tailup covariance function type. Available options include `"linear"`, `"spherical"`, `"exponential"`, `"mariah"`, `"epa"`, and `"none"`.
`de`	The spatially dependent (correlated) random error variance. Commonly referred to as a partial sill.
`range`	The correlation parameter.
`known`	A character vector indicating which covariance parameters are to be assumed known. The value `"given"` is shorthand for assuming all covariance parameters given to `*_initial()` are assumed known.
`taildown_type`	The taildown covariance function type. Available options include `"linear"`, `"spherical"`, `"exponential"`, `"mariah"`, `"epa"`, and `"none"`.
`euclid_type`	The euclidean covariance function type. Available options include `"spherical"`, `"exponential"`, `"gaussian"`, `"cosine"`, `"cubic"`, `"pentaspherical"`, `"wave"`, `"jbessel"`, `"gravity"`, `"rquad"`, `"magnetic"`, and `"none"`.
`rotate`	Anisotropy rotation parameter (from 0 to π radians) for the euclidean portion of the covariance. A value of 0 (the default) implies no rotation.
`scale`	Anisotropy scale parameter (from 0 to 1) for the euclidean portion of the covariance. A value of 1 (the default) implies no scaling.
`nugget_type`	The nugget covariance function type. Available options include `"nugget"` or `"none"`.
`nugget`	The spatially independent (not correlated) random error variance. Commonly referred to as a nugget.

Details

Create an initial object for use with `ssn_lm()` or `ssn_glm()`. `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 `ssn_lm()` and `ssn_glm()`. Parametric forms for each spatial covariance type are presented below.

`tailup_type` Details: Let D be a matrix of hydrologic distances, W be a diagonal matrix of weights from `additive`, $r = D/range$, and I be an identity matrix. Then parametric forms for flow-connected elements of $R(zu)$ are given below:

- linear: $(1 - r) * (r <= 1) * W$
- spherical: $(1 - 1.5r + 0.5r^3) * (r <= 1) * W$
- exponential: $exp(-r) * W$
- mariah: $log(90r + 1)/90r * (D > 0) + 1 * (D = 0) * W$
- epa: $(D - range)^2 * F * (r <= 1) * W/16range^5$
- gaussian: $2exp(-r^2) * (1 - pnorm(r * 2^{1/2})) * W$
- none: I * W

Details describing the `F` matrix in the `epa` covariance are given in Garreta et al. (2010). Flow-unconnected elements of $R(zu)$ are assumed uncorrelated. Observations on different networks are also assumed uncorrelated.

`taildown_type` Details: Let D be a matrix of hydrologic distances, $r = D/range$, and I be an identity matrix. Then parametric forms for flow-connected elements of $R(zd)$ are given below:

- linear: $(1 - r) * (r <= 1)$
- spherical: $(1 - 1.5r + 0.5r^3) * (r <= 1)$
- exponential: $exp(-r)$
- mariah: $log(90r + 1)/90r * (D > 0) + 1 * (D = 0)$
- epa: $(D - range)^2 * F1 * (r <= 1)/16range^5$
- gaussian: 0
- none: I

Now let A be a matrix that contains the shorter of the two distances between two sites and the common downstream junction, $r1 = A/range$, B be a matrix that contains the longer of the two distances between two sites and the common downstream junction, $r2 = B/range$, and I be an identity matrix. Then parametric forms for flow-unconnected elements of $R(zd)$ are given below:

- linear: $(1 - r2) * (r2 <= 1)$
- spherical: $(1 - 1.5r1 + 0.5r2) * (1 - r2)^2 * (r2 <= 1)$
- exponential: 0
- mariah: $(log(90r1+1)-log(90r2+1))/(90r1-90r2)*(A = /B)+(1/(90r1+1))*(A = B)$
- epa: $(B - range)^2 * F2 * (r2 <= 1)/16range^5$
- gaussian: $2exp(-(B - A)/range) * (1 - pnorm(r * 2^{1/2})) * W$
- none: I

Details describing the `F1` and `F2` matrices in the `epa` covariance are given in Garreta et al. (2010). Observations on different networks are assumed uncorrelated.

`euclid_type` Details: Let D be a matrix of Euclidean distances, $r = D/range$, and I be an identity matrix. Then parametric forms for elements of $R(ze)$ are given below:

- exponential: $exp(-r)$
- spherical: $(1 - 1.5r + 0.5r^3) * (r <= 1)$
- gaussian: $exp(-r^2)$
- cubic: $(1 - 7r^2 + 8.75r^3 - 3.5r^5 + 0.75r^7) * (r <= 1)$
- pentaspherical: $(1 - 1.875r + 1.25r^3 - 0.375r^5) * (r <= 1)$
- cosine: $cos(r)$
- wave: $sin(r) * (h > 0)/r + (h = 0)$
- jbessel: $Bj(h * range)$, Bj is Bessel-J function
- gravity: $(1 + r^2)^{-0.5}$
- rquad: $(1 + r^2)^{-1}$
- magnetic: $(1 + r^2)^{-1.5}$
- none: I

`nugget_type` Details: Let I be an identity matrix and 0 be the zero matrix. Then parametric forms for elements the nugget variance are given below:

- nugget: I
- none: 0

In short, the nugget effect is modeled when `nugget_type` is `"nugget"` and omitted when `nugget_type` is `"none"`.

Dispersion and random effect initial objects are specified via `spmodel::dispersion_initial()` and `spmodel::randcov_initial()`, respectively.

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 the relevant spatial covariance type.

References

Peterson, E.E. and Ver Hoef, J.M. (2010) A mixed-model moving-average approach to geostatistical modeling in stream networks. *Ecology* **91(3)**, 644–651.

Ver Hoef, J.M. and Peterson, E.E. (2010) A moving average approach for spatial statistical models of stream networks (with discussion). *Journal of the American Statistical Association* **105**, 6–18. DOI: 10.1198/jasa.2009.ap08248. Rejoinder pgs. 22–24.

Examples

```
tailup_initial("exponential", de = 1, range = 20, known = "range")
tailup_initial("exponential", de = 1, range = 20, known = "given")
euclid_initial("spherical", de = 2, range = 4, scale = 0.8, known = c("range", "scale"))
dispersion_initial("nbinomial", dispersion = 5)
```

---

ssn_lm *Fitting Linear Models for Spatial Stream Networks*

Description

This function works on spatial stream network objects to fit linear models with spatially autocorrelated errors using likelihood methods, allowing for non-spatial random effects, anisotropy, partition factors, big data methods, and more. The spatial formulation is described in Ver Hoef and Peterson (2010) and Peterson and Ver Hoef (2010).

Usage

```
ssn_lm(
  formula,
  ssn.object,
  tailup_type = "none",
  taildown_type = "none",
  euclid_type = "none",
  nugget_type = "nugget",
  tailup_initial,
  taildown_initial,
  euclid_initial,
  nugget_initial,
  additive,
  estmethod = "reml",
  anisotropy = FALSE,
  random,
  randcov_initial,
  partition_factor,
  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.
ssn.object	A spatial stream network object with class SSN.

`tailup_type` The tailup covariance function type. Available options include `"linear"`, `"spherical"`, `"exponential"`, `"mariah"`, `"epa"`, `"gaussian"`, and `"none"`. Parameterizations are described in Details.

`taildown_type` The taildown covariance function type. Available options include `"linear"`, `"spherical"`, `"exponential"`, `"mariah"`, `"epa"`, `"gaussian"`, and `"none"`. Parameterizations are described in Details.

`euclid_type` The euclidean covariance function type. Available options include `"spherical"`, `"exponential"`, `"gaussian"`, `"cosine"`, `"cubic"`, `"pentaspherical"`, `"wave"`, `"jbessel"`, `"gravity"`, `"rquad"`, `"magnetic"`, and `"none"`. Parameterizations are described in Details.

`nugget_type` The nugget covariance function type. Available options include `"nugget"` or `"none"`. Parameterizations are described in Details.

`tailup_initial` An object from `tailup_initial()` specifying initial and/or known values for the tailup covariance parameters.

`taildown_initial` An object from `taildown_initial()` specifying initial and/or known values for the taildown covariance parameters.

`euclid_initial` An object from `euclid_initial()` specifying initial and/or known values for the euclidean covariance parameters.

`nugget_initial` An object from `nugget_initial()` specifying initial and/or known values for the nugget covariance parameters.

`additive` The name of the variable in `ssn.object` that is used to define spatial weights. Can be quoted or unquoted. For the tailup covariance functions, these additive weights are used for branching. Technical details that describe the role of the additive variable in the tailup covariance function are available in Ver Hoef and Peterson (2010).

`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. See spmodel::randcov_initial().

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. local can only be used when big data distance matrices have been created using ssn_create_bigdist() and is most beneficial when the sample size is at least 5,000 ssn_lm() or 3,000 ssn_glm(). 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 200.
- 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 = 200, method = "kmeans", var_adjust = "theoretical", parallel = FALSE).

...

Other arguments to stats::optim().

Details

The linear model for spatial stream networks can be written as $y = X\beta + zu + zd + ze + n$, where X is the fixed effects design matrix, β are the fixed effects, zu is tailup random error, zd is taildown random error, and ze is Euclidean random error, and n is nugget random error. The tailup random errors capture spatial covariance moving downstream (and depend on downstream distance), the taildown random errors capture spatial covariance moving upstream (and depend on upstream) distance, the Euclidean random errors capture spatial covariance that depends on Euclidean distance, and the nugget random errors captures variability independent of spatial locations. The response y is modeled using a spatial covariance function expressed as $de(zu) * R(zu) + de(zd) * R(zd) + de(ze) * R(ze) + nugget * I$. $de(zu)$, $de(zu)$, and $de(zd)$ represent the tailup, taildown, and Euclidean variances, respectively. $R(zu)$, $R(zd)$, and $R(ze)$ represent the tailup, taildown, and Euclidean correlation matrices, respectively. Each correlation matrix depends on a range parameter that controls the distance-decay behavior of the correlation. $nugget$ represents the nugget variance and I represents an identity matrix.

`tailup_type` Details: Let D be a matrix of hydrologic distances, W be a diagonal matrix of weights from `additive`, $r = D/range$, and I be an identity matrix. Then parametric forms for flow-connected elements of $R(zu)$ are given below:

- linear: $(1 - r) * (r <= 1) * W$
- spherical: $(1 - 1.5r + 0.5r^3) * (r <= 1) * W$
- exponential: $exp(-r) * W$
- mariah: $log(90r + 1)/90r * (D > 0) + 1 * (D = 0) * W$
- epa: $(D - range)^2 * F * (r <= 1) * W/16range^5$
- gaussian: $2exp(-r^2) * (1 - pnorm(r * 2^{1/2})) * W$
- none: I * W

Details describing the `F` matrix in the `epa` covariance are given in Garreta et al. (2010). Flow-unconnected elements of $R(zu)$ are assumed uncorrelated. Observations on different networks are also assumed uncorrelated.

`taildown_type` Details: Let D be a matrix of hydrologic distances, $r = D/range$, and I be an identity matrix. Then parametric forms for flow-connected elements of $R(zd)$ are given below:

- linear: $(1 - r) * (r <= 1)$
- spherical: $(1 - 1.5r + 0.5r^3) * (r <= 1)$
- exponential: $exp(-r)$
- mariah: $log(90r + 1)/90r * (D > 0) + 1 * (D = 0)$
- epa: $(D - range)^2 * F1 * (r <= 1)/16range^5$
- gaussian: 0
- none: I

Now let A be a matrix that contains the shorter of the two distances between two sites and the common downstream junction, $r1 = A/range$, B be a matrix that contains the longer of the two distances between two sites and the common downstream junction, $r2 = B/range$, and I be an identity matrix. Then parametric forms for flow-unconnected elements of $R(zd)$ are given below:

- linear: $(1 - r2) * (r2 <= 1)$

- spherical: $(1 - 1.5r1 + 0.5r2) * (1 - r2)^2 * (r2 <= 1)$
- exponential: 0
- mariah: $(log(90r1+1) - log(90r2+1))/(90r1 - 90r2) * (A = /B) + (1/(90r1+1)) * (A = B)$
- epa: $(B - range)^2 * F2 * (r2 <= 1)/16range^5$
- gaussian: $2exp(-(B - A)/range) * (1 - pnorm(r * 2^{1/2})) * W$
- none: I

Details describing the `F1` and `F2` matrices in the `epa` covariance are given in Garreta et al. (2010). Observations on different networks are assumed uncorrelated.

`euclid_type` Details: Let D be a matrix of Euclidean distances, $r = D/range$, and I be an identity matrix. Then parametric forms for elements of $R(ze)$ are given below:

- exponential: $exp(-r)$
- spherical: $(1 - 1.5r + 0.5r^3) * (r <= 1)$
- gaussian: $exp(-r^2)$
- cubic: $(1 - 7r^2 + 8.75r^3 - 3.5r^5 + 0.75r^7) * (r <= 1)$
- pentaspherical: $(1 - 1.875r + 1.25r^3 - 0.375r^5) * (r <= 1)$
- cosine: $cos(r)$
- wave: $sin(r) * (h > 0)/r + (h = 0)$
- jbessel: $Bj(h * range)$, Bj is Bessel-J function
- gravity: $(1 + r^2)^{-0.5}$
- rquad: $(1 + r^2)^{-1}$
- magnetic: $(1 + r^2)^{-1.5}$
- none: I

`nugget_type` Details: Let I be an identity matrix and 0 be the zero matrix. Then parametric forms for elements the nugget variance are given below:

- nugget: I
- none: 0

In short, the nugget effect is modeled when `nugget_type` is `"nugget"` and omitted when `nugget_type` is `"none"`.

`estmethod` Details: The various estimation methods are

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

`anisotropy` Details: By default, all Euclidean covariance parameters except `rotate` and `scale` 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 `euclid_initial()`, `anisotropy` is implicitly set to `TRUE`. (Geometric) Anisotropy is modeled by transforming a Euclidean covariance function that decays differently in different directions to one that decays equally in all directions via rotation and scaling of the original Euclidean 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 Euclidean covariance is then computed using these transformed coordinates.

`random` Details: If random effects are used, the model can be written as $y = X\beta + W1\gamma 1 + ...Wj\gamma j + zu + zd + ze + n$, 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 `ssn_lm()` with the same code. For consistent results, either set a seed via `base::set.seed()` or specify `index` to `local`.

Other Details: 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 and has class `ssn_lm`. Many generic functions that summarize model fit are available for `ssn_lm` objects, including `AIC`, `AICc`, `anova`, `augment`, `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`.

This fitted model list contains the following elements:

- `additive`: The name of the additive function value column.
- `anisotropy`: Whether euclidean anisotropy was modeled.
- `call`: The function call.
- `coefficients`: Model coefficients.
- `contrasts`: Any user-supplied contrasts.
- `cooks_distance`: Cook's distance values.
- `crs`: The geographic coordinate reference system.
- `deviance`: The model deviance.
- `diagtol`: A tolerance value that may be added to the diagonal of covariance matrices to encourage decomposition stability.
- `estmethod`: The estimation method.
- `euclid_max`: The maximum euclidean distance.
- `fitted`: Fitted values.
- `formula`: The model formula.

- hatvalues: The hat (leverage) values.
- is_known: An object that identifies which parameters are known.
- local_index: An index identifier used internally for sorting.
- missing_index: Which rows in the "obs" object had missing responses.
- n: The sample size.
- npar: The number of estimated covariance parameters.
- observed_index: Which rows in the "obs" object had observed responses.
- optim: The optimization output.
- p: The number of fixed effects.
- partition_factor: The partition factor formula.
- pseudoR2: The pseudo R-squared.
- random: The random effect formula.
- residuals: The residuals.
- sf_column_name: The name of the geometry columns ssn.object
- ssn.object: An updated ssn.object.
- tail_max: The maximum stream distance.
- terms: The model terms.
- vcov: Variance-covariance matrices
- xlevels: The levels of factors in the model matrix.

These list elements are meant to be used with various generic functions (e.g., residuals() that operate on the model object. While possible to access elements of the fitted model list directly, we strongly advise against doing so when there is a generic available to return the element of interest. For example, we strongly recommend using residuals() to obtain model residuals instead of accessing the fitted model list directly via object$residuals.

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

Garreta, V., Monestiez, P. and Ver Hoef, J.M. (2010) Spatial modelling and prediction on river networks: up model, down model, or hybrid? *Environmetrics* **21(5)**, 439–456.

Peterson, E.E. and Ver Hoef, J.M. (2010) A mixed-model moving-average approach to geostatistical modeling in stream networks. *Ecology* **91(3)**, 644–651.

Ver Hoef, J.M. and Peterson, E.E. (2010) A moving average approach for spatial statistical models of stream networks (with discussion). *Journal of the American Statistical Association* **105**, 6–18. DOI: 10.1198/jasa.2009.ap08248. Rejoinder pgs. 22–24.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
summary(ssn_mod)
```

---

ssn_names *Return names of data in an SSN object*

Description

Extract and print names from the `edges`, `sites` and `preds` elements of an SSN object.

Usage

```
ssn_names(ssn.object)
```

Arguments

`ssn.object`	An `SSN` object.

Value

Print variable names to console

---

ssn_params *Create covariance parameter objects.*

Description

Create a covariance parameter object for us with other functions. See `spmodel::randcov_params()` for documentation regarding random effect covariance parameter objects.

Usage

```
tailup_params(tailup_type, de, range)

taildown_params(taildown_type, de, range)

euclid_params(euclid_type, de, range, rotate, scale)

nugget_params(nugget_type, nugget)
```

Arguments

tailup_type	The tailup covariance function type. Available options include "linear", "spherical", "exponential", "mariah", "epa", and "none".
de	The spatially dependent (correlated) random error variance. Commonly referred to as a partial sill.
range	The correlation parameter.
taildown_type	The taildown covariance function type. Available options include "linear", "spherical", "exponential", "mariah", "epa", and "none".
euclid_type	The euclidean covariance function type. Available options include "spherical", "exponential", "gaussian", "cosine", "cubic", "pentaspherical", "wave", "jbessel", "gravity", "rquad", "magnetic", and "none".
rotate	Anisotropy rotation parameter (from 0 to π radians) for the euclidean portion of the covariance. A value of 0 (the default) implies no rotation.
scale	Anisotropy scale parameter (from 0 to 1) for the euclidean portion of the covariance. A value of 1 (the default) implies no scaling.
nugget_type	The nugget covariance function type. Available options include "nugget" or "none".
nugget	The spatially independent (not correlated) random error variance. Commonly referred to as a nugget.

Value

A parameter object with class that matches the relevant type argument.

References

Peterson, E.E. and Ver Hoef, J.M. (2010) A mixed-model moving-average approach to geostatistical modeling in stream networks. *Ecology* **91(3)**, 644–651.

Ver Hoef, J.M. and Peterson, E.E. (2010) A moving average approach for spatial statistical models of stream networks (with discussion). *Journal of the American Statistical Association* **105**, 6–18. DOI: 10.1198/jasa.2009.ap08248. Rejoinder pgs. 22–24.

Examples

```
tailup_params("exponential", de = 1, range = 20)
taildown_params("exponential", de = 1, range = 20)
euclid_params("exponential", de = 1, range = 20, rotate = 0, scale = 1)
nugget_params("nugget", nugget = 1)
```

---

ssn_put_data *Put an sf data.frame in an SSN object*

Description

The ssn_put_data function puts an sf data.frame representing observation or prediction data into an SSN, ssn_lm, or ssn_glm object.

Usage

```
ssn_put_data(data, x, name = "obs", resize_data = FALSE)
```

Arguments

data	sf data.frame with point geometry.
x	An object of class SSN, ssn_lm, or ssn_glm.
name	the internal name of the data set in the object x. For observed data, this will always be "obs", the default.
resize_data	Logical. Indicates whether sf_df can have a different number of features than the current data.frame in the object. Default is FALSE.

Details

The internal name for observed data in objects of class SSN, ssn_lm, and ssn_glm is "obs" and it is the default. If another name is specified, it must represent a prediction dataset in the object. For SSN objects, these names are obtained using the call names(x$preds). For all other object classes, the names are obtained using the call names(x$ssn.object$preds).

The resize_sf_data argument specifies whether sf_data can have a different number of features (i.e., rows) than the sf data.frame it is replacing. Care should be taken when resize_df is set to TRUE, especially if the new sf_data has more features than the existing sf data.frame. In these cases, the user is responsible for ensuring that the additional features have the correct spatial, topological, and attribute data to accurately represent spatial relationships in the SSN object.

Value

Returns an object of the same class as x, which contains the sf data.frame sf_data.

See Also

ssn_get_data()

Examples

```
data(mf04p)
## Extract observation data.frame from SSN object
obs.df <- ssn_get_data(mf04p)
## Create a new column for summer mean temperature and set Value in
obs.df$Value <- obs.df$Summer_mn
obs.df$Value[1] <- NA

## Put the modified sf data.frame into the SSN object
mf04p <- ssn_put_data(obs.df, mf04p)
head(ssn_get_data(mf04p)[, c("Summer_mn", "Value")])
```

---

ssn_simulate *Simulate random variables on a stream network*

Description

Simulate random variables on a stream network with a specific mean and covariance structure. Designed to use `ssn_simulate()`, but individual simulation functions for each response distribution also exist.

Usage

```
ssn_simulate(
  family = "Gaussian",
  ssn.object,
  network = "obs",
  tailup_params,
  taildown_params,
  euclid_params,
  nugget_params,
  additive,
  mean = 0,
  samples = 1,
  dispersion = 1,
  size = 1,
  randcov_params,
  partition_factor,
  ...
)

ssn_rbeta(
  ssn.object,
  network = "obs",
  tailup_params,
  taildown_params,
  euclid_params,
```

```
  nugget_params,
  dispersion = 1,
  mean = 0,
  samples = 1,
  additive,
  randcov_params,
  partition_factor,
  ...
)

ssn_rbinom(
  ssn.object,
  network = "obs",
  tailup_params,
  taildown_params,
  euclid_params,
  nugget_params,
  mean = 0,
  size = 1,
  samples = 1,
  additive,
  randcov_params,
  partition_factor,
  ...
)

ssn_rgamma(
  ssn.object,
  network = "obs",
  tailup_params,
  taildown_params,
  euclid_params,
  nugget_params,
  dispersion = 1,
  mean = 0,
  samples = 1,
  additive,
  randcov_params,
  partition_factor,
  ...
)

ssn_rinvgauss(
  ssn.object,
  network = "obs",
  tailup_params,
  taildown_params,
  euclid_params,
```

```
  nugget_params,
  dispersion = 1,
  mean = 0,
  samples = 1,
  additive,
  randcov_params,
  partition_factor,
  ...
)

ssn_rnbinom(
  ssn.object,
  network = "obs",
  tailup_params,
  taildown_params,
  euclid_params,
  nugget_params,
  dispersion = 1,
  mean = 0,
  samples = 1,
  additive,
  randcov_params,
  partition_factor,
  ...
)

ssn_rnorm(
  ssn.object,
  network = "obs",
  tailup_params,
  taildown_params,
  euclid_params,
  nugget_params,
  mean = 0,
  samples = 1,
  additive,
  randcov_params,
  partition_factor,
  ...
)

ssn_rpois(
  ssn.object,
  network = "obs",
  tailup_params,
  taildown_params,
  euclid_params,
  nugget_params,
```

```
  mean = 0,
  samples = 1,
  additive,
  randcov_params,
  partition_factor,
  ...
)
```

Arguments

family	The response distribution family. The default is "Gaussian".
ssn.object	A spatial stream network object with class SSN. Random variables are simulated for each row of ssn.object$obs.
network	The spatial stream network to simulate on. Currently only allowed to be "obs" for the ssn.object$obs object.
tailup_params	An object from tailup_params() specifying the tailup covariance parameters.
taildown_params	An object from taildown_params() specifying the taildown covariance parameters.
euclid_params	An object from euclid_params() specifying the Euclidean covariance parameters.
nugget_params	An object from nugget_params() specifying the nugget covariance parameters.
additive	The name of the variable in ssn.object that is used to define spatial weights. Can be quoted or unquoted. For the tailup covariance functions, these additive weights are used for branching. Technical details that describe the role of the additive variable in the tailup covariance function are available in Ver Hoef and Peterson (2010).
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.
dispersion	The dispersion value (if relevant).
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.
randcov_params	A spmodel::randcov_params() object.
partition_factor	A formula indicating the partition factor.
...	Other arguments. Not used (needed for generic consistency).

Details

Random variables are simulated via the product of the covariance matrix's square (Cholesky) root and independent standard normal random variables on the link scale, which are then used to simulate a relevant variable on the response scale according to family. Computing the square root is a significant computational burden and likely unfeasible for sample sizes much past 10,000. Because

this square root only needs to be computed once, however, it is nearly the sample computational cost to call `ssn_rnorm()` for any value of `samples`.

If not using `ssn_simulate()`, individual simulation functions for each response distribution do exist:

- `ssn_rnorm()`: Simulate from a Gaussian distribution
- `ssn_rpois()`: Simulate from a Poisson distribution
- `ssn_rnbinom()`: Simulate from a negative binomial distribution
- `ssn_rbinom()`: Simulate from a binomial distribution
- `ssn_rbeta()`: Simulate from a beta distribution
- `ssn_rgamma()`: Simulate from a gamma distribution
- `ssn_rinvgauss()`: Simulate from an inverse Gaussian distribution

Value

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

References

Ver Hoef, J.M. and Peterson, E.E. (2010) A moving average approach for spatial statistical models of stream networks (with discussion). *Journal of the American Statistical Association* **105**, 6–18. DOI: 10.1198/jasa.2009.ap08248. Rejoinder pgs. 22–24.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

tailup <- tailup_params("exponential", de = 0.1, range = 200)
taildown <- taildown_params("exponential", de = 0.4, range = 300)
euclid <- euclid_params("spherical", de = 0.2, range = 1000, rotate = 0, scale = 1)
nugget <- nugget_params("nugget", nugget = 0.1)
ssn_simulate("gaussian", mf04p, "obs", tailup, taildown, euclid, nugget, additive = "afvArea")
```

ssn_split_predpts *Split a prediction dataset in an* SSN *object*

Description

The splitPrediction function is used to split prediction sets in an SSN object into smaller prediction sets. It returns a SSN object with additional prediction sets based on equal interval splits, a factor, integer, character or logical column stored within the prediction set, or a logical expression.

Usage

```
ssn_split_predpts(
  ssn,
  predpts,
  size_predpts,
  by,
  subset,
  id_predpts,
  keep = TRUE,
  drop_levels = FALSE,
  overwrite = FALSE
)
```

Arguments

ssn	An SSN object.
predpts	A character string representing the name of the prediction dataset.
size_predpts	numeric value representing the size of the new prediction sets. The existing prediction set is split equally to produce multiple prediction sets of this size
by	character string representing the column name of type factor, integer, character or logical that the split will be based on
subset	logical expression indicating which elements or rows to keep; missing values are taken as FALSE
id_predpts	character string representing the new prediction dataset name. This value is only specified when the subset method is used
keep	logical value indicating whether the original prediction dataset should be retained in the SSN object. Default is TRUE
drop_levels	logical value indicating whether empty factor levels should be dropped in the by column when the new prediction dataset(s) are created. Default is FALSE
overwrite	logical indicating whether the new prediction dataset geopackage should be deleted in the .ssn directory if it already exists. Default = FALSE

Details

Three methods have been provided to split prediction sets: size, by, and subset. The size method is used to split the existing prediction set into multiple equally-sized prediction sets using the `size_predpts` argument. Note that the final prediction set may be smaller in size than the others if the total number of predictions is not evenly divisible by `size_predpts`. The by method is used if the prediction set is to be split into multiple new prediction sets based on an existing column of type factor, integer, character, or logical specified using the argument `by`. The subset method is used to create one new prediction set based on a logical expression defined in `subset`.

When more than one prediction dataset is created the prediction dataset names will be appended with a hyphen and prediction dataset number if more than one prediction dataset is created. For example, when "preds" is split using `size_predpts`, the new names will be "preds-1", "preds-2", and so forth.

When `keep=FALSE`, the prediction dataset is removed from the `SSN` object stored in memory, but is not deleted from the .ssn directory specified in `ssn$path`.

Note that, only one method may be specified when the `ssn_split_predpts` function is called. The distance matrices for the new prediction datasets must be created using the `ssn_create_distmat` before predictions can be made.

Value

returns the `SSN` specified in `ssn`, with one or more new prediction sets. Geopackages of the new prediction sets are written to the .ssn directory designated in ssn$path.

Examples

```
## Import SSN object
copy_lsn_to_temp() ## Only needed for this example
ssn <- ssn_import(paste0(tempdir(), "/MiddleFork04.ssn"),
  predpts = c("pred1km", "CapeHorn"),
  overwrite = TRUE
)

## Split predictions based on 'size' method
ssn1 <- ssn_split_predpts(ssn, "CapeHorn",
  size_predpts = 200,
  keep = FALSE, overwrite = TRUE
)
names(ssn1$preds)
nrow(ssn1$preds[["CapeHorn-1"]])

## Split predictions using 'by' method
ssn$preds$pred1km$net.fac <- as.factor(ssn$preds$pred1km$netID)
ssn2 <- ssn_split_predpts(ssn, "pred1km",
  by = "net.fac",
  overwrite = TRUE
)
names(ssn2$preds)

## Split predictions using 'subset' method
```

```
ssn3 <- ssn_split_predpts(ssn, "pred1km",
  subset = ratio > 0.5,
  id_predpts = "RATIO_05", overwrite = TRUE
)
names(ssn3$preds)
```

---

ssn_subset *Subset an* SSN *object*

Description

Returns an SSN object that has been subset based on a logical expression.

Usage

```
ssn_subset(ssn, path, subset, clip = FALSE, overwrite = FALSE)
```

Arguments

ssn	An SSN object.
path	The filepath to the .ssn folder, in string format, where the subset SSN will be saved.
subset	A logical expression indicating which features to keep.
clip	If TRUE, create a subset of the edges and prediction sites, based on the same logical expression used to subset the observed sites. Default = FALSE.
overwrite	If TRUE, overwrite the folder specified in path if it exists. Default = FALSE.

Details

This function creates a subset of the original SSN object based on a logical expression defined in the subset argument. The subset argument is treated as an expression within ssn_subset() and so the full argument is not a string; although values in factor or character format will still require quotes (see examples). If clip = TRUE, then the columns referred to in subset must be present in the edges and all of the prediction datasets (if present in the SSN object). Note that features with missing values in the subset expression are treated as false and are not included in the subset SSN object.

Once the subset SSN object has been written to the local directory, it is re-imported using ssn_import. During this process, the binaryID.db is recreated. If distance matrices exist in the original SSN object, they are not copied or recalculated for the new SSN object. Users will need to run the ssn_create_distmat to create the distance matrices before fitting models to the data in the subset SSN.

Value

an object of class SSN, which is stored locally in the .ssn directory specified in path. It also creates and stores an SQLite database, binaryID.db, within the .ssn directory.

Examples

```
## Import SSN object
copy_lsn_to_temp() ## Only needed for this example
mf04p <- ssn_import(paste0(tempdir(), "/MiddleFork04.ssn"),
  predpts = "pred1km",
  overwrite = TRUE
)

## Subset SSN observations, edges, and prediction sites on network 1
ssn.sub1 <- ssn_subset(mf04p,
  path = paste0(tempdir(), "/subset1.ssn"),
  subset = netID == 1, clip = TRUE,
  overwrite = TRUE
)

## Subset SSN observations, removing two sites
ssn.sub2 <- ssn_subset(mf04p,
  path = paste0(tempdir(), "/subset2.ssn"),
  subset = !COMID %in% c("23519461", "23519365"),
  overwrite = TRUE
)
```

SSN_to_SSN2 *Convert object from* SpatialStreamNetwork *class to* SSN *class*

Description

Convert an S4 SpatialStreamNetwork object created in the SSN package to an S3 SSN object used in the SSN2 package.

Usage

```
SSN_to_SSN2(object)
```

Arguments

object	A SpatialStreamNetwork object

Details

SSN_to_SSN2() has been made available to help users migrate from the SSN package to the updated SSN2 package. It is used to convert existing S4 SpatialStreamNetwork objects *stored in saved workspaces* to the S3 SSN class object used in the SSN2 package. Note that ssn_import is used to create an S3 SSN object from data stored locally in a .ssn directory.

Value

An S3 SSN class object.

ssn_update_path *Update path in an SSN object*

Description

Update the local path in an existing SSN object based on an user defined file.

Usage

```
ssn_update_path(x, path, verbose = FALSE)
```

Arguments

x	An SSN, ssn_lm or ssn_glm object.
path	Filepath to the .ssn folder associated with the SSN object.
verbose	A logical that indicates if the new path should be printed to the console.

Details

At times, it may be necessary to move a .ssn directory, which is linked to an SSN object in an R workspace. If the .ssn directory is moved, the path must be updated before using the ssn_glmssn function and other functions that read/write to the .ssn directory. The ssn_update_path is a helper function that serves this purpose.

Value

An SSN object with a new path list element.

Examples

```
## Use mf04p SSN object provided in SSN2
data(mf04p)

## For examples only, make sure mf04p has the correct path
## If you use ssn_import(), the path will be correct
newpath <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_update_path(mf04p, newpath)
```

ssn_write *write an SSN object*

Description

This function writes an SSN object to a local .ssn directory

Usage

```
ssn_write(ssn, path, overwrite = FALSE, copy_dist = FALSE, import = FALSE)
```

Arguments

ssn	An SSN object.
path	filepath to the local .ssn directory to write to.
overwrite	If TRUE, overwrite existing files in file (if it exists). Defaults to FALSE.
copy_dist	If TRUE, copy distance matrices to file (if they exist). Defaults to FALSE.
import	If TRUE, import and return the SSN object after writing to file. Defaults to FALSE.

Value

ssn_write creates an .ssn directory that contains the spatial, topological, and attribute information stored in the original SSN object. Spatial datasets found in the SSN object (e.g. edges, obs, and prediction sites) are saved in GeoPackage format. When import = TRUE, the SSN object is imported and returned.

Examples

```
## For examples only, copy MiddleFork04.ssn directory to R's
# temporary directory
copy_lsn_to_temp()
## Import SSN object with prediction sites
mf04p <- ssn_import(paste0(tempdir(), "/MiddleFork04.ssn"),
  predpts = "pred1km",
  overwrite = TRUE
)

## Write SSN to new .ssn directory
ssn_write(mf04p,
  path = paste0(tempdir(), "/tempSSN.ssn"),
  overwrite = TRUE
)

## Write SSN to .ssn directory and return SSN object
tempSSN <- ssn_write(mf04p, path = paste0(
  tempdir(),
  "/tempSSN.ssn"
), overwrite = TRUE, import = TRUE)
```

summary.SSN *Summarize an SSN object*

Description

Summarize data found in an SSN object.

Usage

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

Arguments

object	An SSN object.
...	Other arguments. Not used (needed for generic consistency).

Details

`summary.SSN()` creates a summary of a SSN object intended to be printed using `print()`. This summary contains information about the number of observed and prediction locations, as well as the column names found in their respective sf data.frames.

Value

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

summary.SSN2 *Summarize a fitted model object*

Description

Summarize a fitted model object.

Usage

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

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

Arguments

object	A fitted model object from `ssn_lm()` or `ssn_glm()`.
...	Other arguments. Not used (needed for generic consistency).

Details

summary.ssn() 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.SSN2

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
summary(ssn_mod)
```

---

tidy.SSN2 *Tidy a fitted model object*

Description

Tidy a fitted model object into a summarized tibble.

Usage

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

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

Arguments

x	A fitted model object from ssn_lm() or ssn_glm().
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), "tailup" (tailup covariance parameters), "taildown" (taildown covariance parameters), "euclid" (Euclidean covariance parameters), "nugget" (nugget covariance parameter), "dispersion" (dispersion parameter if relevant), "ssn" for all of "tailup", "taildown", "euclid", "nugget", and "dispersion", 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.SSN2() augment.SSN2()

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
tidy(ssn_mod)
```

Torgegram *Compute the empirical semivariogram*

Description

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

Usage

```
Torgegram(
  formula,
  ssn.object,
  type = c("flowcon", "flowuncon"),
  cloud = FALSE,
  robust = FALSE,
  bins = 15,
  cutoff,
  partition_factor
)
```

Arguments

formula	A formula describing the fixed effect structure.
ssn.object	A spatial stream network object with class SSN.
type	The Torgegram type. A vector with possible values "flowcon" for flow-connected distances, "flowuncon" for flow-unconnected distances, and "euclid" for Euclidean distances. The default is to show both flow-connected and flow-unconnected distances.
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 for each type. The default is FALSE.
bins	The number of equally spaced bins. The default is 15.
cutoff	The maximum distance considered. The default is half the diagonal of the bounding box from the coordinates.
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.

Details

The Torgegram is an empirical semivariogram is a tool used to visualize and model spatial dependence by estimating the semivariance of a process at varying distances separately for flow-connected, flow-unconnected, and Euclidean 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 SSN2, 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

A list with elements correspond to type. Each element is data frame with distance bins (bins), the average distance (dist), the semivariance (gamma), and the number of (unique) pairs (np) for the respective type.

References

Cressie, N & Hawkins, D.M. 1980. Robust estimation of the variogram. *Journal of the International Association for Mathematical Geology*, **12**, 115-125. Zimmerman, D. L., & Ver Hoef, J. M. (2017). The Torgegram for fluvial variography: characterizing spatial dependence on stream networks. *Journal of Computational and Graphical Statistics*, **26(2)**, 253–264.

See Also

plot.Torgegram()

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

tg <- Torgegram(Summer_mn ~ 1, mf04p)
plot(tg)
```

---

varcomp.SSN2 *Variability component comparison*

Description

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

Usage

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

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

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().
...	Other arguments. Not used (needed for generic consistency).

Value

A tibble that partitions the the total variability by the fixed effects and each variance parameter. The proportion of variability explained by the fixed effects is the pseudo R-squared obtained by psuedoR2(). The remaining proportion is spread accordingly among each variance parameter: "tailup_de", "taildown_de", "euclid_de", "nugget", and if random effects are used, each named random effect. For ssn_glm(), models, only the variances on the link scale are considered (i.e., the variance function of the response is omitted).

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
varcomp(ssn_mod)
```

---

vcov.SSN2 *Calculate variance-covariance matrix for a fitted model object*

Description

Calculate variance-covariance matrix for a fitted model object.

Usage

```
## S3 method for class 'ssn_lm'
vcov(object, ...)

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

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().
...	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 ssn_glm() (when family is different from "Gaussian"). The default is TRUE.

Value

The variance-covariance matrix of coefficients obtained via coef(). Currently, only the variance-covariance matrix of the fixed effects is supported.

Examples

```
# Copy the mf04p .ssn data to a local directory and read it into R
# When modeling with your .ssn object, you will load it using the relevant
# path to the .ssn data on your machine
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path, overwrite = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
vcov(ssn_mod)
```

Index