Package: SSN2 (via r-universe)

November 21, 2026

Title Spatial Modeling on Stream Networks

Version 0.5.0

Description Fit, summarize, and predict for spatial linear and generalized linear models on stream networks. Stream covariance models use moving average constructions based on in-stream distance. Parameters are estimated using likelihood methods. Additional modeling features include Euclidean covariance, anisotropy, non-spatial random effects, partition factors, and big data methods. Model-fit statistics are used to summarize, visualize, and compare models, and predictions at unobserved locations are readily obtainable. For additional details, see Ver Hoef and Peterson (2010) <DOI:10.1198/jasa.2009.ap08248> and Dumelle et al. (2024) <DOI:10.21105/joss.06389>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Depends R (>= 2.10)

Imports stats, sf, Matrix, generics, tibble, graphics, spmodel (>= 0.14.0), RSQLite, utils, withr, doParallel, filematrix, foreach, parallel, itertools, iterators

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

Config/testthat/edition 3

VignetteBuilder knitr

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

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

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 https://cran.r-universe.dev

Date/Publication 2026-11-21 00:20:02 UTC

RemoteUrl https://github.com/cran/SSN2

RemoteRef HEAD

RemoteSha 066c70876c076e7daf54430f42402bfc38fd0d37

Contents

anova.SSN2 . . . 3
augment.SSN2 . . . 5
coef.SSN2 . . . 8
conditional.SSN2 . . . 10
confint.SSN2 . . . 12
cooks.distance.SSN2 . . . 13
copy_lsn_to_temp . . . 14
covmatrix.SSN2 . . . 15
create_netgeom . . . 16
deviance.SSN2 . . . 17
fitted.SSN2 . . . 18
formula.SSN2 . . . 20
glance.SSN2 . . . 21
glances.SSN2 . . . 22
hatvalues.SSN2 . . . 23
influence.SSN2 . . . 24
kcv.SSN2 . . . 25
labels.SSN2 . . . 28
logLik.SSN2 . . . 29
loocv.SSN2 . . . 30
mf04p . . . 32
MiddleFork04.ssn . . . 32
model.frame.SSN2 . . . 35
model.matrix.SSN2 . . . 36
nobs.SSN2 . . . 37
plot.SSN2 . . . 38
plot.Torgegram . . . 39
predict.ssn_decorrelate . . . 40
print.SSN . . . 44
print.ssn_decorrelate . . . 45
pseudoR2.SSN2 . . . 47
residuals.SSN2 . . . 48
satterthwaite.SSN2 . . . 49

ssn_create_bigdist . . . 51
ssn_create_distmat . . . 53
ssn_decorrelate . . . 56
ssn_decorrelate_data . . . 62
ssn_decorrelate_grid . . . 65
ssn_decorrelate_newdata . . . 67
ssn_get_data . . . 69
ssn_get_netgeom . . . 70
ssn_get_stream_distmat . . . 71
ssn_glm . . . 74
ssn_import . . . 82
ssn_import_predpts . . . 84
ssn_initial . . . 86
ssn_lm . . . 90
ssn_lmRF . . . 96
ssn_names . . . 98
ssn_params . . . 98
ssn_put_data . . . 100
ssn_recorrelate_newdata . . . 101
ssn_simulate . . . 102
ssn_split_predpts . . . 106
ssn_subset . . . 109
SSN_to_SSN2 . . . 110
ssn_update_path . . . 111
ssn_write . . . 112
summary.SSN . . . 113
summary.SSN2 . . . 113
tidy.SSN2 . . . 114
Torgegram . . . 115
varcomp.SSN2 . . . 118
vcov.SSN2 . . . 119

Index **121**

`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, ddf)

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

## 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.
test	A logical value indicating whether p-values from asymptotic Chi-squared hypothesis tests should be returned. Defaults to TRUE.
Terms	An optional character or integer vector that specifies terms in the model used to jointly compute test statistics and p-values (if test = TRUE) against a null hypothesis of zero. Terms is only used when a single fitted model object is passed to the function. If Terms is a character vector, it should contain the names of the fixed effect terms. If Terms is an integer vector, it should correspond to the order (starting at one) of the names of the fixed effect terms. The easiest way to obtain the names of all possible terms is to run tidy(anova(object))$effects (the integer representation matches the positions of this vector).
L	An optional numeric matrix or list specifying linear combinations of the coefficients in the model used to compute test statistics and p-values (if test = TRUE) for coefficient constraints corresponding to a null hypothesis of zero. L is only used when a single fitted model object is passed to the function. If L is a numeric matrix, its rows indicate coefficient constraints and its columns represent coefficients. Then a single hypothesis test is conducted against a null hypothesis of zero. If L is a list, each list element is a numeric matrix specified as above. Then separate hypothesis tests are conducted. The easiest way to obtain all possible coefficients is to run tidy(object)$term.
ddf	The denominator degrees of freedom used. "asymptotic" implements an asymptotic chi-squared test. "satterthwaite" implements a Satterthwaite/Fai-Cornelius F-test. The default is "satterthwaite" when the sample size is less than or equal to 500 and "asymptotic" otherwise. Currently in SSN2, only numerically approximated Satterthwaite degrees of freedom are available.
x	An object from anova(object).

Details

When one fitted model object is present, anova() performs a general linear hypothesis test corresponding to some hypothesis specified by a matrix of constraints. If Terms and L are not specified,

each model term is tested against zero (which correspond to type III or marginal hypothesis tests from classical ANOVA). If `Terms` is specified and `L` is not specified, all terms are tested jointly against zero. When `L` is specified, the linear combinations of terms specified by `L` are jointly tested against zero.

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

Value

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

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

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

See Also

`satterthwaite()`

Examples

```
# 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` is specified. When augmenting the original data, diagnostic statistics are augmented to each row in the data set. When augmenting new data, predictions and optional intervals or standard errors are augmented to each row in the new data set.

Usage

```
## S3 method for class '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,
  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 "all" 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. Defaults to FALSE. When newdata is not supplied, .se.fit is the standard error of the fitted mean at the observed locations (on the link scale for ssn_glm() model objects), matching predict(interval = "confidence").
interval	Character indicating the type of interval columns (.lower and .upper) to add to the augmented output. Defaults to "none".
level	Tolerance/confidence level. The default is 0.95.
local	A list or logical. If a list, specific list elements described in predict.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()` `predict.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". If type is "tailup", "taildown", "euclid", or "nugget", the coefficient vector is a tailup_params(), taildown_params(), euclid_params(), or nugget_params() object (respectively), which has class matching the corresponding covariance function used.
...	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)
```

`conditional.SSN2` *Conditionally simulate from a model*

Description

Conditionally simulate prediction data from a model object.

Usage

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

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

Arguments

`object`	A fitted model object.
`newdata`	A character string naming a single prediction data set (accessible via `object$ssn.object$preds`). May be omitted when `"newdata"` is not requested; otherwise, omission uses the `.missing` prediction set, if present. `"all"`/multiple prediction sets are not supported.
`output`	The output type, which can be any subset of `c("newdata", "beta", "object")` (`"all"` is shorthand for this full vector). If `simulate_covparams = TRUE`, the output type can also be any subset of `c("cov", "ssn", "tailup", "taildown", "euclid", "nugget", "randcov")`. For generalized linear models, `"latent"` returns simulated observed-site link values, including offsets. `"random"` returns realized random effects. The default is `"newdata"`. See Details for more.
`samples`	The number of conditional simulations. The default is `1,000`.

simulate_covparams
: For ssn_lm() model objects, whether to also simulate new covariance parameters for each sample. simulate_covparams requires object$vcov$cov to be specified during model fitting by selecting ddf = "satterthwaite". simulate_covparams = TRUE should generally not be used for sample sizes greater than 500 given its computational inefficiencies. The default is FALSE.

...
: Other arguments. Not used (needed for generic consistency).

type
: For ssn_glm() model objects, the scale of the conditional simulations for newdata. When type = "link", the predicted means on the link scale are returned. When type = "response", the predicted means on the response scale are returned. When type = "new", a new observation is simulated from the appropriate response distribution with mean equal to the mean on the response scale and dispersion equal to the dispersion parameter from object. The default is "link".

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

Details

If "newdata" is in output, conditional simulations are returned for each row of the named prediction set. If "beta" is in output, conditional simulations are returned for each fixed effect (i.e., element of coef(object)). If "object" is in output, observed responses (ssn_lm()) or fitted link values (including offsets) (ssn_glm()) are returned in observed-row order, repeated across simulation columns. If "latent" is requested, the actual observed link-scale draws are returned, with offsets included. type affects only newdata. For example, c("newdata", "beta") returns the conditional simulations both for newdata and for the fixed effects. If "cov"/"ssn"/"tailup"/"taildown"/ "euclid"/"nugget"/"randcov" is in output (only available when simulate_covparams = TRUE), the simulated covariance parameter draws themselves are returned.

Value

If output = "newdata", an a x b matrix of conditional simulations for each row in newdata, where a is the number of rows in newdata and b is the number of samples. If output = "beta", an p x b matrix of conditional simulations for each element in coef(object), where p is the number of fixed effects and b is the number of samples. If output = "object", an n x b matrix of observed responses (ssn_lm()) or fitted link values (including offsets) (ssn_glm()), where n is the number of observed rows and b is the number of samples. output = "latent" returns an n x b matrix of simulated observed link values (including offsets). output = "random" returns a q x b matrix of realized random-effect coefficients, separated by partition factor (when relevant). If output = "cov"/"ssn"/"tailup"/"taildown"/ "euclid"/"nugget"/"randcov" (simulate_covparams = TRUE only), a (covariance parameter) x b matrix of conditional simulations for each covariance parameter is returned, where b is the number of samples. "cov" returns every free covariance parameter, while "ssn" returns the tailup, taildown, Euclidean, and nugget subset. "tailup"/"taildown"/"euclid"/"nugget"/ "randcov" return just the corresponding covariance parameters.

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

Examples

```
# 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_create_distmat(mf04p, predpts = "CapeHorn", overwrite = TRUE, among_predpts = TRUE)

ssn_mod <- ssn_lm(
  formula = Summer_mn ~ ELEV_DEM,
  ssn.object = mf04p,
  tailup_type = "exponential",
  additive = "afvArea"
)
set.seed(1)
draws <- conditional(ssn_mod, "CapeHorn")
head(draws) # rows are prediction locations; columns are simulations
```

---

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

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 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()
# 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 character string naming the prediction data set (accessible via `object$ssn.object$preds`) for which the covariance is desired. Note that the prediction data must be in the original SSN object used to fit `object`.
`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

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

Examples

```
# 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, overwrite = FALSE)
```

Arguments

sf_data	An sf object with LINESTRING or POINT geometry created using ssn_assemble() from SSNbler (see Details).
type	Character string defining geometry type of sf_data: "POINT" or "LINESTRING".
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 ssn_assemble() is run or an SSN is imported using 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 ssn_assemble() from SSNbler.

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

```
# 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()

# 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).

Random effects with the same grouping level in different partition_factor partitions are reported separately. Their names include the grouping level and partition. Computational blocks used for local fitting do not create distinct random effects: block contributions to the same effect are averaged.

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.

See Also

stats::AIC() AICc() stats::BIC() logLik.SSN2() deviance.SSN2() pseudoR2() tidy.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"
)
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	A fitted model object from ssn_lm() or ssn_glm().
...	Additional fitted model objects.
sort_by	Sort by a glance statistic (i.e., the name of a column output from glance() or the order of model input (sort_by = "order"). The default is "AICc".
decreasing	Whether sort_by should sort by decreasing order? The default is FALSE.
warning	Whether a warning is displayed when model comparisons violate certain rules. The default is TRUE.

Value

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

Examples

```
# 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)
```

---

kcv.SSN2 *Perform k-fold cross validation*

Description

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

Usage

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

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

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().
k	The number of folds. Must be a whole number at least 2 and no more than the sample size (the number of non-missing observations in object). The default is 5. If k equals the sample size, kcv() is equivalent to (and calls) loocv(). Ignored when folds_index is supplied.
cv_predict	A logical indicating whether the k-fold cross validation fitted values should be returned. Defaults to FALSE. If object is from ssn_glm(), the fitted values returned are on the link scale.
se.fit	A logical indicating whether the k-fold cross validation prediction standard errors should be returned. Defaults to FALSE. If object is from ssn_glm(), the standard errors correspond to the fitted values returned on the link scale.
local	A list or logical. If a list, specific list elements described in predict.SSN2() control the big data approximation behavior. If a logical, TRUE chooses default list elements for the list version of local as specified in predict.SSN2(). Defaults to FALSE, which performs exact computations.
interval	For ssn_lm() objects, whether to append empirical coverage at level. The default, "none", retains the legacy 80, 90, and 95 percent coverage columns.

level	Prediction interval level used when interval = "prediction". The default is 0.95.
folds_index	An optional vector, the same length as the number of non-missing observations in object and in the same order, assigning each observation to a fold (at least two distinct values required). When supplied, this fold assignment is used as-is, implying k is ignored and no random fold assignment is performed. The default is NULL, which randomly assigns k (approximately) equally sized folds.
...	Other arguments. Not used (needed for generic consistency).
type	The scale (response or link) of predictions obtained when cv_predict = TRUE and using ssn_lm() or ssn_glm objects.
delta	A logical indicating whether to return delta method standard errors on the response scale when se.fit = TRUE and type = "response". The default is FALSE.

Details

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

When object is from ssn_lm(), setting interval = "prediction" additionally reports the empirical coverage of the level (e.g. 95\ the proportion of held-out observations whose true value falls within fit +/- qnorm(1 - (1 - level) / 2) * se.fit (the same normal-quantile interval spmodel::predict.spmodel() uses by default). This is only available for ssn_lm() objects, since ssn_glm() have no observed-scale latent mean to compare against.

Value

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

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")
mf <- ssn_import(temp_path, overwrite = TRUE)
fit <- ssn_lm(Summer_mn ~ ELEV_DEM, mf,
  tailup_type = "exponential", additive = "afvArea"
)
set.seed(1)
kcv(fit, k = 3)
```

`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 with options for computationally efficient approximations for big data.

Usage

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

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

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. If object is from ssn_glm(), the fitted values returned are on the link scale.
se.fit	A logical indicating whether the leave-one-out prediction standard errors should be returned. Defaults to FALSE. If object is from ssn_glm(), the standard errors correspond to the fitted values returned on the link scale.
local	A list or logical. If a list, specific list elements described in predict.SSN2() control the big data approximation behavior. If a logical, TRUE chooses default list elements for the list version of local as specified in predict.SSN2(). Defaults to FALSE, which performs exact computations.

interval	Whether to report coverage at a requested prediction interval level. "none" preserves the legacy coverage columns; "prediction" additionally reports coverage at level. Only available for ssn_lm() objects.
level	The requested prediction interval coverage level. Defaults to 0.95.
...	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.
delta	A logical indicating whether to return delta method standard errors on the response scale when se.fit = TRUE and type = "response". The default is FALSE.

Details

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

Value

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

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 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 Fork 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)
```

---

nobs.SSN2 *Extract the number of observations*

Description

Find the number of observations used for model fitting. This exists so that generic R functions that call `nobs()` (e.g., `stats::nobs()`) work on `SSN2` fitted model objects. Could alternatively rename object$n as object$nobs and rely on stats::nobs.default.

Usage

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

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

Arguments

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

Value

The number of observations used for modeling (i.e., `object$n`).

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"
)
nobs(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 the empirical semivariogram or empirical autocovariance diagnostic pairs summarized in a fitted Torgegram object.

Usage

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

Arguments

x	A Torgegram object from Torgegram().
type	The diagnostic pair type. 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.ssn_decorrelate

Model predictions (Kriging)

Description

Predicted values and intervals based on a fitted model object.

Usage

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

## 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", "weight"),
  block = FALSE,
  local,
  terms = NULL,
  na.action = na.fail,
  ...
```

```
)

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

## S3 method for class 'ssn_lmRF'
predict(
  object,
  newdata,
  local,
  se.fit = FALSE,
  interval = "none",
  type = "response",
  block = FALSE,
  ...
)
```

Arguments

`object`	A fitted model object from `ssn_lm()`, `ssn_glm()`, `ssn_lmRF()`, or `ssn_decorrelate()`.
`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.
`local`	A optional logical or list controlling the big data approximation. If omitted, `local` is set to `TRUE` or `FALSE` based on the observed data sample size (i.e., sample size of the fitted model object) – if the sample size exceeds 10,000, `local` is set to `TRUE`, otherwise it is set to `FALSE`. This default behavior occurs because for point prediction the main computational burden of the big data approximation depends almost exclusively on the observed data sample size, not the number of

predictions desired (which we feel is not intuitive at first glance). For block prediction (block = TRUE) the density of the prediction grid also matters, so local is additionally set to approximate the theoretical solution when nrow(newdata) exceeds 10,000 (see below). If local is FALSE, no big data approximation is implemented. If a list is provided, the following arguments detail the big data approximation:

- method: The big data approximation method. If method = "all", all observations are used and size is ignored. If method = "covariance", the size data observations having the largest absolute covariance with each prediction location are used. The default is "covariance".
- size: The number of data observations to use when method is "covariance". The default is 200.
- 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.
- chunk_size: Maximum number of prediction rows per covariance-construction chunk (default 1000). Relevant primarily with prediction for large newdata.

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 = "covariance", chunk_size = 1000, parallel = FALSE).

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

- method and size act on the observed data exactly as when block is FALSE (method takes "all" or "covariance"); the default method is "covariance" with size 4000. This default size is much larger than when block is FALSE because block prediction averages covariances and explanatory variables before prediction.
- method_new controls the big data prediction grid density. method_new = "basis" (the default) approximates the block variance using a basis of size_new well-spread grid nodes. method_new = "subset" approximates the block mean and variance by subsampling newdata so that its size is only size_new. The default size_new is 4000.
- ordering chooses the size_new nodes from newdata. The options are "pid", "maxmin", "grts", "random", "none", "middleout", "outsidein", "coordinate". The default is "pid".
- parallel and ncores parallelize the covariance-chunk calculations. Their defaults are the same as for point prediction.
- chunk_size: Maximum number of prediction rows per covariance-construction chunk (default 1000). Relevant primarily with prediction for large newdata.

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

size = 4000, method_new = "basis", size_new = 4000, ordering = "pid", chunk_size = 1000).

...
: Other arguments. Only used for models fit using ssn_lmRF() where ... indicates other arguments to ranger::predict.ranger().

se.fit
: A logical indicating if standard errors are returned. The default is FALSE.

scale
: A numeric constant by which to scale the regular standard errors and intervals. Similar to but slightly different than scale for stats::predict.lm(), because predictions form a spatial model may have different residual variances for each observation in newdata. The default is NULL, which returns the regular standard errors and intervals.

df
: Degrees of freedom to use for confidence or prediction intervals (ignored if scale is not specified). The default is Inf.

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

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

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

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

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.

dispersion
: The dispersion 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.

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

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

Details

For ssn_lm and ssn_glm objects, the (empirical) best linear unbiased predictions (i.e., Kriging predictions) at each site are returned when interval is "none" or "prediction" alongside standard

errors. Prediction intervals are also returned if `interval` is `"prediction"`. When `interval` is `"confidence"`, the estimated mean is returned alongside standard errors and confidence intervals for the mean.

For `ssn_lmRF` objects, random forest spatial residual model predictions combine the random forest prediction with the (empirical) best linear unbiased prediction for the residual. This approach is called random forest regression Kriging.

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

Value

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

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

For `ssn_lmRF` and `ssn_decorrelate` objects, a vector of predictions.

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.ssn_decorrelate *Print values*

Description

Print fitted model objects and summaries.

Usage

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

## 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"),
  ...
)

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

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

Arguments

x	A fitted model object from ssn_lm(), ssn_glm(), ssn_decorrelate(), or ssn_lmRF(), or output from summary(x) or anova(x).
digits	The number of significant digits to use when printing.
...	Other arguments passed to or from other methods.
signif.stars	Logical. If TRUE, significance stars are printed for each coefficient

Value

Printed fitted model objects and summaries with formatting.

Examples

```
# 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)
fit <- ssn_lm(
  Summer_mn ~ ELEV_DEM, mf04p, tailup_type = "exponential",
  additive = "afvArea"
)
print(fit)
```

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 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 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.

See Also

`augment.SSN2()` `cooks.distance.SSN2()` `hatvalues.SSN2()` `influence.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"
)
residuals(ssn_mod)
resid(ssn_mod)
residuals(ssn_mod, type = "pearson")
residuals(ssn_mod, type = "standardized")
rstandard(ssn_mod)
```

---

satterthwaite.SSN2 *Compute Satterthwaite denominator degrees of freedom*

Description

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

Usage

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

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

Arguments

object	A fitted model object from ssn_lm() fit without local. Not supported for ssn_glm() objects.
method	The method by which to compute gradients. Must be supplied as "numeric" for numerical differentiation, the only supported method.
...	Other arguments. Not used (needed for generic consistency).

Details

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

Value

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

References

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

See Also

ssn_lm() anova.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"
)
satterthwaite(ssn_mod, method = "numeric")
```

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 the SSN2 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_bigdist` function within the .ssn directory. The distance directory will always contain at least one directory named 'obs', which contains a number of .bmat 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.bmat). 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 .bmat 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.bmat). 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. Pairwise distances between prediction locations will also be calculated if `among_predpts = TRUE`.

Value

The `ssn_create_bigdist` 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 MiddleFork04.ssn data to a local temporary directory.
## Only needed for this example.
copy_lsn_to_temp()
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
## Import SSN data
mf04p <- ssn_import(temp_path,
  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_bigdist(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_bigdist(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 can be fitted with ssn_lm and ssn_glm in the SSN2 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. 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 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")
## Import SSN data
mf04p <- ssn_import(temp_path,
  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_decorrelate *Apply the Spatial Decorrelation Transformation for Machine Learning Models*

Description

Apply the spatial decorrelation transformation for spatial stream network data, allowing for random effects, anisotropy, partition factors, and big data methods.

Usage

```
ssn_decorrelate(
  formula,
  ssn.object,
  tailup_type = "none",
  taildown_type = "none",
  euclid_type = "none",
  nugget_type = "nugget",
  tailup_params,
  taildown_params,
  euclid_params,
  nugget_params,
  additive,
  algorithm = "ranger",
  statistic = "RMSPE",
  training,
  evaluate_test,
  anisotropy = FALSE,
  random,
  randcov_params,
  partition_factor,
  ordering,
  local,
  grid,
  dense_grid,
  ...
)

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

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms on the right, separated by + operators. . on the right-hand side represents every variable in ssn.object$obs except the response and geometry column.

`ssn.object` A spatial stream network object with class `SSN`.

`tailup_type`, `taildown_type`, `euclid_type`, `nugget_type`
Covariance types specified as in `ssn_lm()`.

`tailup_params`, `taildown_params`, `euclid_params`, `nugget_params`
Known covariance parameter objects to be used in the spatial decorrelation transformation. See `tailup_params()`, `taildown_params()`, `euclid_params()`, and `nugget_params()`.

`additive` Additive-function variable for the tail-up covariance, as in `ssn_lm()`.

`algorithm` The machine learning algorithm applied. Available options include `"ranger"`, `"randomForest"`, and `"xgboost"`. `"ranger"` specifies a random forest via `ranger::ranger()`. `"randomForest"` specifies a random forest via `randomForest::randomForest()`. `"xgboost"` specifies a boosted decision tree ensemble via `xgboost::xgboost()`.

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

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

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

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

`evaluate_test` A logical indicating whether a grid should be constructed and evaluated when spatial decorrelation parameters are known (i.e., `*_params` are all specified, and, if random effects are included, `randcov_params` is specified). If `TRUE`, constructs and evaluates the grid after assigning observations to test and training

data sets. If any parameters (spatial or random effects) are estimated via a grid search, `evaluate_test` is set to `TRUE`.

`anisotropy` A logical indicating whether Euclidean (geometric) anisotropy should be modeled. Not required if `euclid_params` is provided with a nonzero `rotate` or a `scale` 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_params` An object from `spmodel::randcov_params()` that contains the random effect variances used by the spatial decorrelation transformation.

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

`ordering` The data ordering applied. Available options include `"pid"`, `"grts"`, `"maxmin"`, `"middleout"`, `"outsidein"`, `"coordinate"`, `"random"`, and `"none"`. `"pid"` applies ordering by point ID from the network geometry, with network ID used to break ties. `"grts"` applies ordering using a spatially balanced GRTS sample via `spsurvey::grts()`. `"maxmin"` applies maximum minimum distance ordering via `GPvecchia::order_maxmin_exact()`. `"middleout"` applies middle out ordering via `GPvecchia::order_middleout()`. `"outsidein"` applies outside in ordering via `GPvecchia::order_outsidein()`. `"coordinate"` applies coordinate ordering via `GPvecchia::order_coordinate(..., coordinate = c(1, 2))`, which orders from bottom-left to top-right of the spatial domain. `"random"` applies a completely random ordering. `"none"` applies no random ordering. The default is `"pid"`. `"maxmin"`, `"middleout"`, `"outsidein"`, and `"coordinate"` require **GPvecchia**; `"grts"` requires **spsurvey**.

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

- `method`: The big data approximation method. If `method = "all"`, all earlier observations in the ordering are used and `size` is ignored. If `method = "covariance"`, the `size` data observations with the highest absolute covariance among earlier observations are used. The default is `"covariance"`.

- `size`: The number of data observations to use when `method` is `"covariance"`. The default is 30.
- `parallel`: If `TRUE`, parallel processing via the **parallel** package is 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 = 30, method = "covariance", parallel = FALSE)`.

`grid` An explicit grid of parameter values by which to evaluate fit. The names of `grid` must contain all the names returned by `ssn_decorrelate_grid(formula, ssn.object, ...)`.

`dense_grid` If `grid` is not provided, `dense_grid` is a logical which controls the density of the constructed grid to be evaluated. If `dense_grid` is `TRUE`, a denser grid is used. If `dense_grid` is `FALSE`, a sparser grid is used. By default, `dense_grid` is `FALSE`.

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

`x` A fitted model from `ssn_decorrelate()`.

Details

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

`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: $2exp(-r^2) * (1 - pnorm(r * 2^{1/2})) * W$
- wave: $\rho_c(D) * W$, see `tailup_initial()`.
- 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
- wave: $\rho_c(D)$, see `taildown_initial()`.
- 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: $2exp(-(B - A)/range) * (1 - pnorm(r * 2^{1/2})) * W$
- wave: $\rho_u(A, B)$, see `taildown_initial()`.
- 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: $(1 - 1.875r + 1.25r^3 - 0.375r^5) * (r <= 1)$
- circular: $1 - (2/\pi) * (r * sqrt(1 - r^2) + \arcsin(r))$ for $0 \leq r \leq 1$, and zero for $r > 1$
- 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: $(1 + r^2)^{-1.5}$
- matern: $2^{1-extra} eta^{extra} K_{extra}(eta)/Gamma(extra)$, where $eta = \sqrt{2extra}D/range$
- cauchy: $(1 + r^2)^{-extra}$

- pexponential: $exp(-D^{extra}/range)$
- none: I

The powered-exponential range has units of distance raised to `extra`.

`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` as well as all random effect variance parameters are assumed unknown, requiring estimation. If either `rotate` or `scale` are given values other than 0 and 1 (respectively) in `euclid_params()`, `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 $y = X\beta + Z1u1 + ...Zjuj + zu + zd + ze + n$, 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 (Hadamard product) by P, yielding the final covariance matrix.

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

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

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

Value

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

- `algorithm`: The machine learning algorithm used.

- decorrelate_data: The output of ssn_decorrelate_data() applied to ssn.object.
- fit: The fitted machine learning model object applied to the decorrelated data.
- grid: If used, the grid of spatial decorrelation parameters evaluated and their corresponding metrics when applied to the test data.
- newdata: The name ".missing" when observations with NA response values are stored as prediction data, or NULL otherwise.
- training: If used, the observations assigned to each training and test data set.
- test: If used, a list with mean bias (bias), mean-squared-prediction error (MSPE), root-mean-squared-prediction error (RMSPE), and predictive R-squared (cor2) for the candidate selected by statistic.

References

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

Examples

```
# 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_create_distmat(mf04p, predpts = "pred1km", overwrite = TRUE)
fit <- ssn_decorrelate(
  Summer_mn ~ ELEV_DEM, mf04p, tailup_type = "exponential",
  taildown_type = "exponential", euclid_type = "exponential",
  additive = "afvArea", dense_grid = FALSE
)
tidy(fit$grid)
head(predict(fit, "pred1km"))
```

---

ssn_decorrelate_data *Apply the Spatial Decorrelation Transformation to a Data Object*

Description

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

Usage

```
ssn_decorrelate_data(
  formula,
  ssn.object,
  tailup_params,
  taildown_params,
  euclid_params,
  nugget_params,
  additive,
  randcov_params,
  partition_factor,
  ordering,
  local,
  ...
)
```

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms on the right, separated by + operators. . on the right-hand side represents every variable in ssn.object$obs except the response and geometry column.
ssn.object	A spatial stream network object with class SSN.
tailup_params	A tailup_params() object giving the known tailup covariance parameters, or omitted/NULL for no tailup covariance (equivalent to tailup_type = "none").
taildown_params	A taildown_params() object giving the known taildown covariance parameters, or omitted/NULL for no taildown covariance (equivalent to taildown_type = "none").
euclid_params	A euclid_params() object giving the known Euclidean covariance parameters, or omitted/NULL for no Euclidean covariance (equivalent to euclid_type = "none"). Anisotropy is modeled automatically whenever euclid_params's rotate is nonzero or its scale is not one; there is no separate anisotropy argument.
nugget_params	A nugget_params() object giving the known nugget covariance parameter. Required (there is no default): every active covariance parameter, including the nugget, must be supplied and known.
additive	Additive-function variable for the tail-up covariance, as in ssn_lm().
randcov_params	A spmodel::randcov_params() object giving the known random-effect variances, or omitted/NULL for no random effects. The random effect grouping structure is derived automatically from its names; there is no separate random argument.
partition_factor	A one-sided linear formula with a single term specifying the partition factor. The partition factor assumes observations from different levels of the partition factor are uncorrelated.

`ordering` The data ordering applied. Available options include `"pid"`, `"grts"`, `"maxmin"`, `"middleout"`, `"outsidein"`, `"coordinate"`, `"random"`, and `"none"`. `"pid"` applies ordering by point ID from the network geometry, with network ID used to break ties. `"grts"` applies ordering using a spatially balanced GRTS sample via `spsurvey::grts()`. `"maxmin"` applies maximum minimum distance ordering via `GPvecchia::order_maxmin_exact()`. `"middleout"` applies middle out ordering via `GPvecchia::order_middleout()`. `"outsidein"` applies outside in ordering via `GPvecchia::order_outsidein()`. `"coordinate"` applies coordinate ordering via `GPvecchia::order_coordinate(..., coordinate = c(1, 2))`, which orders from bottom-left to top-right of the spatial domain. `"random"` applies a completely random ordering. `"none"` applies no random ordering. The default is `"pid"`. `"maxmin"`, `"middleout"`, `"outsidein"`, and `"coordinate"` require **GPvecchia**; `"grts"` requires **spsurvey**.

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

- `method`: The big data approximation method. If `method = "all"`, all earlier observations in the ordering are used and `size` is ignored. If `method = "covariance"`, the `size` data observations with the highest absolute covariance among earlier observations are used. The default is `"covariance"`.
- `size`: The number of data observations to use when `method` is `"covariance"`. The default is 30.
- `parallel`: If `TRUE`, parallel processing via the **parallel** package is 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 = 30, method = "covariance", parallel = FALSE)`.

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

Details

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

Value

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

- `X`: The original fixed effects design matrix (of explanatory variables)

- y: The original response variable
- tX: The spatially decorrelated transformed fixed effects design matrix
- ty: The spatially decorrelated transformed response variable

References

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

See Also

ssn_decorrelate(), ssn_decorrelate_newdata(), ssn_recorrelate_newdata()

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)
transformed <- ssn_decorrelate_data(
  Summer_mn ~ ELEV_DEM, mf04p, additive = "afvArea",
  tailup_params = tailup_params("exponential", de = 1, range = 10000),
  nugget_params = nugget_params("nugget", nugget = 0.1)
)
head(cbind(response = transformed$y, transformed = transformed$ty))
```

---

ssn_decorrelate_grid *Create a Spatial Decorrelation Transformation Grid*

Description

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

Usage

```
ssn_decorrelate_grid(
  formula,
  ssn.object,
  tailup_type = "none",
  taildown_type = "none",
  euclid_type = "none",
  nugget_type = "nugget",
  tailup_params,
```

```
  taildown_params,
  euclid_params,
  nugget_params,
  additive,
  anisotropy = FALSE,
  random,
  randcov_params,
  dense_grid
)

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

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

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms on the right, separated by + operators. . on the right-hand side represents every variable in ssn.object$obs except the response and geometry column.
ssn.object	A spatial stream network object with class SSN.
tailup_type, taildown_type, euclid_type, nugget_type	Covariance types specified as in ssn_lm().
tailup_params, taildown_params, euclid_params, nugget_params	Known covariance parameter objects to be used in the spatial decorrelation transformation. See tailup_params(), taildown_params(), euclid_params(), and nugget_params().
additive	Additive-function variable for the tail-up covariance, as in ssn_lm().
anisotropy	A logical indicating whether Euclidean (geometric) anisotropy should be modeled. Not required if euclid_params is provided with a nonzero rotate or a scale 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_params	An object from spmodel::randcov_params() that contains the random effect variances used by the spatial decorrelation transformation.

dense_grid	A logical which controls the density of the constructed grid to be evaluated. If dense_grid is TRUE, a denser grid is used. If dense_grid is FALSE, a sparser grid is used. By default, dense_grid is FALSE.
x	An object from object$grid.
...	Other arguments to the functions called by algorithm.
sort_by	Sort by a specific row in x. Fit statistics are "bias", "MSPE", "RMSPE", and "cor2". The default is "MSPE".
decreasing	Whether sort_by should sort by decreasing order? If sort_by = "cor2", the default is TRUE; otherwise it is FALSE.

Value

A grid of spatial decorrelation parameters stored as a data.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)
grid <- ssn_decorrelate_grid(
  Summer_mn ~ ELEV_DEM, mf04p, tailup_type = "exponential",
  additive = "afvArea", dense_grid = FALSE
)
tidy(grid)
# Supply edited rows as grid = custom_grid to ssn_decorrelate().
custom_grid <- grid[1:2, ]
custom_grid$tailup_range <- c(5000, 10000)
```

---

ssn_decorrelate_newdata

Apply the Spatial Decorrelation Transformation to a Newdata Object for Prediction

---

Description

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

Usage

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

Arguments

object	An ssn_decorrelate_data() object.
newdata	A character string naming a single prediction data set (accessible via object$covariance_fit$ssn.obje Note that the prediction data must be in the original SSN object used to fit the model. If omitted, missing data from the fitted model object are used via the .missing prediction set, if present. Otherwise, newdata must be supplied.
local	An optional logical or list controlling the big data approximation. If omitted, the settings in object$local are used. 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 with the highest absolute covariance with the observation requiring prediction are used. The default is "covariance". • size: The number of data observations to use when method is "covariance". The default is 30. • parallel: If TRUE, parallel processing via the **parallel** package is 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 = 30, method = "covariance", parallel = FALSE).
...	Other arguments.

Value

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

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

Examples

```
# 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_create_distmat(mf04p, predpts = "CapeHorn", overwrite = TRUE)
transformed <- ssn_decorrelate_data(
```

```
  Summer_mn ~ ELEV_DEM, mf04p, additive = "afvArea",
  tailup_params = tailup_params("exponential", de = 1, range = 10000),
  nugget_params = nugget_params("nugget", nugget = 0.1)
)
new_transformed <- ssn_decorrelate_newdata(transformed, "CapeHorn")
head(new_transformed$tX_newdata)
```

---

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* prediction 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 `among_predpts = TRUE` was used in the function `ssn_create_distmat`, then the distance between prediction sites themselves is also returned, using the same labeling 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

```
# 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")
## Create an SSN object with prediction sites
mf04p <- ssn_import(temp_path,
  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
b.mat <- pmax(dist_obs$dist.net2, t(dist_obs$dist.net2))
b.mat[5:10, 5:10]

## Get minimum downstream only distance between observations. If
## minimum distance == 0, sites are flow-connected
a.mat <- pmin(dist_obs$dist.net2, t(dist_obs$dist.net2))
a.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 *Fit spatial generalized linear models for stream networks*

Description

Fit spatial generalized linear models for stream network data using likelihood methods, allowing for non-spatial random effects, anisotropy, partition factors, and big data methods. 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,
  range_constrain,
  ...
)
```

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms on the right, separated by + operators. . on the right-hand side represents every variable in data except the geometry column.
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 "inverse.gaussian". 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", "wave", and "none". Parameterizations are described in Details.
taildown_type	The taildown covariance function type. Available options include "linear", "spherical", "exponential", "mariah", "epa", "gaussian", "wave", and "none". Parameterizations are described in Details.
euclid_type	The euclidean covariance function type. Available options include "spherical", "exponential", "gaussian", "circular", "cubic", "pentaspherical", "wave", "jbessel", "gravity", "rquad", "magnetic", "matern", "cauchy", "pexponential", 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 dispersion parameter for the "nbinomial", "beta", "Gamma", and "inverse.gaussian" families. family is ignored if dispersion_initial is provided.
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 euclid_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 variable named must be a factor, ordered factor, or character. 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)`.

`range_constrain`

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

`...`

Other arguments to `spmodel::esv()` or `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 $de(zu) * R(zu) + de(zd) * R(zd) + de(ze) * R(ze) + nugget * I$. $de(zu)$, $de(ze)$, 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. Random effects are also modeled on the link scale.

`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: $2exp(-r^2) * (1 - pnorm(r * 2^{1/2})) * W$
- wave: $\rho_c(D) * W$, see `tailup_initial()`.
- 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
- wave: $\rho_c(D)$, see `taildown_initial()`.
- 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: $2exp(-(B - A)/range) * (1 - pnorm(r * 2^{1/2})) * W$
- wave: $\rho_u(A, B)$, see `taildown_initial()`.
- 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: $(1 - 1.875r + 1.25r^3 - 0.375r^5) * (r <= 1)$
- circular: $1 - (2/\pi) * (r * sqrt(1 - r^2) + \arcsin(r))$ for $0 \leq r \leq 1$, and zero for $r > 1$
- 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: $(1 + r^2)^{-1.5}$
- matern: $2^{1-extra} eta^{extra} K_{extra}(eta)/Gamma(extra)$, where $eta = \sqrt{2extra} D/range$
- cauchy: $(1 + r^2)^{-extra}$
- pexponential: $exp(-D^{extra}/range)$
- none: I

The powered-exponential range has units of distance raised to extra.

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.

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

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 $g(\mu) = \eta = X\beta + Z1u1 + ...Zjuj + zu + zd + ze + n$, 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 (Hadamard 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_glm()` 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. The list has class `ssn_glm`. Many generic functions that summarize model fit are available for `ssn_glm` objects, including `AIC`, `AICc`, `anova`, `augment`, `AUROC`, `BIC`, `coef`, `cooks.distance`, `covmatrix`, `deviance`, `fitted`, `formula`, `glance`, `glances`, `hatvalues`, `influence`, `labels`, `logLik`, `loocv`, `model.frame`, `model.matrix`, `plot`, `predict`, `print`, `pseudoR2`, `summary`, `terms`, `tidy`, `update`, `varcomp`, and `vcov`.

Note

This function does not perform any internal scaling. If optimization is not stable due to 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, prediction sites) and overwrite = TRUE, it will be overwritten.

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 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

```
# 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")

## Import SSN object with no prediction sites
mf04 <- ssn_import(temp_path,
  overwrite = TRUE
)

## Import SSN object with 3 sets of prediction sites
mf04p <- ssn_import(temp_path,
  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

```
# 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")

## Import SSN object with no prediction sites
mf04p <- ssn_import(temp_path,
  overwrite = TRUE
)

## Import pred1km prediction dataset into SSN object and assign the
## name preds1
mf04p <- ssn_import(temp_path,
    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, extra)

nugget_initial(nugget_type, nugget, known)
```

Arguments

`tailup_type`	The tailup covariance function type. Available options include `"linear"`, `"spherical"`, `"exponential"`, `"mariah"`, `"epa"`, `"gaussian"`, `"wave"`, 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"`, `"gaussian"`, `"wave"`, and `"none"`.

euclid_type	The Euclidean covariance function type. Available options include "spherical", "exponential", "gaussian", "circular", "cubic", "pentaspherical", "wave", "jbessel", "gravity", "rquad", "magnetic", "matern", "cauchy", "pexponential", 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.
extra	An optional initial or known Matérn smoothness, Cauchy shape, or powered-exponential exponent. Omit it to initialize the parameter by the covariance search.
nugget_type	The nugget covariance function type. Available options include "nugget" or "none".
nugget	The spatially independent (uncorrelated) 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 rotate and scale, which lets these functions find initial values for parameters that are otherwise assumed known by default (rotate = 0 and scale = 1 when anisotropy is not modeled with ssn_lm() or 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: $2exp(-r^2) * (1 - pnorm(r * 2^{1/2})) * W$
- wave: $\rho_c(D) * W$, see Stream wave covariance below.
- 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
- wave: $\rho_c(D)$, see Stream wave covariance below.
- 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: $2exp(-(B - A)/range) * (1 - pnorm(r * 2^{1/2})) * W$
- wave: $\rho_u(A, B)$, see Stream wave covariance below.
- 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: $(1 - 1.875r + 1.25r^3 - 0.375r^5) * (r <= 1)$
- circular: $1 - (2/\pi) * (r * sqrt(1 - r^2) + \arcsin(r))$ for $0 \leq r \leq 1$, and zero for $r > 1$
- 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: $(1 + r^2)^{-1.5}$
- matern: $2^{1-extra} eta^{extra} K_{extra}(eta)/Gamma(extra)$, where $eta = \sqrt{2extra}D/range$
- cauchy: $(1 + r^2)^{-extra}$
- pexponential: $exp(-D^{extra}/range)$
- none: I

The powered-exponential range has units of distance raised to `extra`.

`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 relevant spatial covariance type.

Stream wave covariance

Stream `"wave"` uses the one-sided moving-average kernel $k_\theta(x) = \sin(x/\theta)/(x/\theta)$ for $x > 0$, with $k_\theta(0) = 1$ and `range` equal to $\theta > 0$. The flow-connected correlations are

$$\rho_c(h) = \frac{2}{\pi\theta}\int_0^\infty k_\theta(u)k_\theta(u+h)\,du$$

for (hydrologic) distance h. The flow-unconnected correlations are

$$\rho_u(a,b) = \frac{2}{\pi\theta}\int_a^\infty k_\theta(u)k_\theta(u+b-a)\,du$$

, where $0 \leq a \leq b$ are the shorter and longer distances to their common downstream junction.

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"))
euclid_initial("matern", de = 2, range = 4, extra = 1, known = "extra")
dispersion_initial("nbinomial", dispersion = 5)
```

ssn_lm *Fit spatial linear models for stream networks*

Description

Fit spatial linear models for stream-network data using likelihood methods, allowing for non-spatial random effects, anisotropy, partition factors, and big data methods. 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,
  range_constrain,
  ddf,
  ...
)
```

Arguments

formula	A two-sided linear formula describing the fixed effect structure of the model, with the response to the left of the ~ operator and the terms on the right, separated by + operators. . on the right-hand side represents every variable in data except the geometry column.
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", "wave", and "none". Parameterizations are described in Details.
taildown_type	The taildown covariance function type. Available options include "linear", "spherical", "exponential", "mariah", "epa", "gaussian", "wave", and "none". Parameterizations are described in Details.

`euclid_type` The euclidean covariance function type. Available options include `"spherical"`, `"exponential"`, `"gaussian"`, `"circular"`, `"cubic"`, `"pentaspherical"`, `"wave"`, `"jbessel"`, `"gravity"`, `"rquad"`, `"magnetic"`, `"matern"`, `"cauchy"`, `"pexponential"`, 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 `euclid_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 variable named must be a factor, ordered factor, or character. 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)`.

`range_constrain`

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

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

... — Other arguments to `spmodel::esv()` or `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 $de(zu) * R(zu) + de(zd) * R(zd) + de(ze) * R(ze) + nugget * I$. $de(zu)$, $de(ze)$, 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: $2exp(-r^2) * (1 - pnorm(r * 2^{1/2})) * W$
- wave: $\rho_c(D) * W$, see `tailup_initial()`.
- 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
- wave: $\rho_c(D)$, see `taildown_initial()`.
- 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: $2exp(-(B - A)/range) * (1 - pnorm(r * 2^{1/2})) * W$
- wave: $\rho_u(A, B)$, see `taildown_initial()`.
- 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: $(1 - 1.875r + 1.25r^3 - 0.375r^5) * (r <= 1)$
- circular: $1 - (2/\pi) * (r * sqrt(1 - r^2) + \arcsin(r))$ for $0 \leq r \leq 1$, and zero for $r > 1$
- 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: $(1 + r^2)^{-1.5}$
- matern: $2^{1-extra}eta^{extra}K_{extra}(eta)/Gamma(extra)$, where $eta = \sqrt{2extra}D/range$
- cauchy: $(1 + r^2)^{-extra}$
- pexponential: $exp(-D^{extra}/range)$
- none: I

The powered-exponential range has units of distance raised to `extra`.

`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` as well as all random effect variance parameters are assumed unknown, requiring estimation. If either `rotate` or `scale` are given initial values other than 0 and 1 (respectively) or are assumed unknown in `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 $y = X\beta + Z1u1 + ...Zjuj + zu + zd + ze + n$, 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 (Hadamard 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. The list has class `ssn_lm`. Many generic functions that summarize model fit are available for `ssn_lm` objects, including `AIC`, `AICc`, `anova`, `augment`, `BIC`, `coef`, `cooks.distance`, `covmatrix`, `deviance`, `fitted`, `formula`, `glance`, `glances`, `hatvalues`, `influence`, `labels`, `logLik`, `loocv`, `model.frame`, `model.matrix`, `plot`, `predict`, `print`, `pseudoR2`, `satterthwaite`, `summary`, `terms`, `tidy`, `update`, `varcomp`, and `vcov`.

Note

This function does not perform any internal scaling. If optimization is not stable due to 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_lmRF *Fit a random forest stream-network residual model*

Description

Fit random forest spatial residual models for stream network data using random forest to fit the mean and a spatial linear model to fit the residuals. The spatial linear model fit to the residuals can incorporate variety of estimation methods, allowing for random effects, anisotropy, partition factors, and big data methods.

Usage

```
ssn_lmRF(formula, ssn.object, ...)
```

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.
...	Additional named arguments to ranger::ranger() or ssn_lm().

Details

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

Value

A list with several elements to be used with predict(). These elements include the function call (named call), the random forest object fit to the mean (named ranger), the spatial stream network model object fit to the residuals (named ssn_lm), and an object can contain data for locations at which to predict (called newdata). The newdata object contains the set of observations in data whose response variable is NA.

Note

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

References

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

Examples

```
if (requireNamespace("ranger", quietly = TRUE)) {
# 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")
mf <- ssn_import(temp_path,
  predpts = "CapeHorn", overwrite = TRUE
)
ssn_create_distmat(mf, predpts = "CapeHorn", overwrite = TRUE)
rf_fit <- ssn_lmRF(
  Summer_mn ~ ELEV_DEM, mf, tailup_type = "exponential",
  additive = "afvArea", num.trees = 50, num.threads = 1, seed = 1
)
print(rf_fit)
```

```
head(predict(rf_fit, "CapeHorn"))
}
```

---

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 use 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, extra)

nugget_params(nugget_type, nugget)
```

Arguments

tailup_type	The tailup covariance function type. Available options include "linear", "spherical", "exponential", "mariah", "epa", "gaussian", "wave", 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", "gaussian", "wave", and "none".
euclid_type	The Euclidean covariance function type. Available options include "spherical", "exponential", "gaussian", "circular", "cubic", "pentaspherical", "wave", "jbessel", "gravity", "rquad", "magnetic", "matern", "cauchy", "pexponential", 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.
extra	An extra covariance parameter used when spcov_type is "matern", "cauchy", "pexponential".
nugget_type	The nugget covariance function type. Available options include "nugget" or "none".
nugget	The spatially independent (uncorrelated) random error variance. Commonly referred to as a nugget.

Details

Generally, all arguments to tailup_params(), taildown_params(), euclid_params(), and nugget_params() must be specified, though default arguments are chosen when the relevant _type is "none". For full parameterizations of all tailup, taildown, Euclidean, and nugget covariance functions, see tailup_initial(), taildown_initial(), euclid_initial(), and nugget_initial().

Value

A named numeric vector of covariance parameters 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)
euclid_params("matern", de = 1, range = 20, extra = 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 data 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_data argument specifies whether 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_data is set to TRUE, especially if the new 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 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
## first row to NA
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_recorrelate_newdata

Recorrelate Machine Learning Predictions

Description

Recorrelate machine learning predictions according to a spatial decorrelation transformation.

Usage

```
ssn_recorrelate_newdata(object, ty_newdata)
```

Arguments

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

Value

A vector of predictions

Examples

```
# 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_create_distmat(mf04p, predpts = "CapeHorn", overwrite = TRUE)
transformed <- ssn_decorrelate_data(
  Summer_mn ~ ELEV_DEM, mf04p, additive = "afvArea",
  tailup_params = tailup_params("exponential", de = 1, range = 10000),
```

```
  nugget_params = nugget_params("nugget", nugget = 0.1)
)
learner <- lm.fit(transformed$tX, transformed$ty)
new_transformed <- ssn_decorrelate_newdata(transformed, "CapeHorn")
predictions <- ssn_recorrelate_newdata(
  new_transformed, as.numeric(new_transformed$tX_newdata %*% learner$coefficients)
)
head(predictions)
```

---

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 ssn.object$obs. The default is 0. For non-Gaussian families, this is the mean of the latent normal random variable on the link scale. See Details for more.
samples	The number of independent samples to generate. The default is 1.
dispersion	The dispersion value for the beta, gamma, negative binomial, and inverse Gaussian distributions. The default is 1.
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.
...	Additional arguments passed to ssn_rnorm() when using ssn_simulate() or another simulation function.

Details

Random variables are simulated via the product of the covariance matrix's square (Cholesky) root and independent standard normal random variables with mean 0 and variance 1.

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_simulate("poisson", mf04p, "obs", tailup, taildown, euclid, nugget, additive = "afvArea")
```

---

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

Description

The ssn_split_predpts 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 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() ## Only needed for this example
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
ssn <- ssn_import(temp_path,
  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 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() ## Only needed for this example
temp_path <- paste0(tempdir(), "/MiddleFork04.ssn")
mf04p <- ssn_import(temp_path,
  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 a 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_lm()`/`ssn_glm()` functions 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

```
# 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")
## Import SSN object with prediction sites
mf04p <- ssn_import(temp_path,
  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 an 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, ...)

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

Arguments

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

Details

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

Value

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

See Also

print.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 an empirical stream-network diagnostic*

Description

Compute the empirical semivariogram or autocovariance 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,
  eacf = FALSE
)
```

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 selected diagnostic should be summarized by distance class or not. When cloud = FALSE (the default), pairwise values are binned and averaged within distance classes. When cloud = TRUE, all selected pairwise values and distances are returned.
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. Ignored if cloud = TRUE.
cutoff	The maximum distance considered. The default is half the maximum observed distance for each selected pair type.
partition_factor	An optional formula specifying the partition factor. If specified, diagnostic values are only computed for observations sharing the same level of the partition factor.
eacf	A logical indicating whether to compute the empirical autocovariance rather than the empirical semivariogram. The default is FALSE. The same pairs, distances, bins, cutoff, and partition restrictions are used in either case. When TRUE, robust = TRUE is not available.

Details

With eacf = FALSE, the Torgegram is an empirical semivariogram 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 $1/(2N(h))\sum(r1 - r2)^2$, 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).

With eacf = TRUE, the Torgegram is an empirical autocovariance used to visualize and model spatial dependence by estimating the autocovariance of a process at varying distances. For a constant-mean process, the autocovariance at distance h is denoted $Cov(h)$ and defined as $Cov(z1, z2)$. Under second-order stationarity, $Cov(h) = Cov(0) - \gamma(h)$, where $\gamma(h)$ is the semivariance function at distance h. Typically the residuals from an ordinary least squares fit defined by formula are second-order stationary with mean zero. These residuals are used to compute the empirical autocovariance. At a distance h, the empirical autocovariance is $1/N(h)\sum(r1 \times r2)$, where $N(h)$ is the number of (unique) pairs in the set of observations whose distance separation is h and r1 and r2 are residuals corresponding to observations whose distance separation is h.

In SSN2, the Torgegram 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 either diagnostic (this cutoff is determined by cutoff).

Value

With eacf = FALSE, a list with elements corresponding 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. With eacf = TRUE, the empirical autocovariance (acov) replaces gamma; the other columns are unchanged. When cloud = TRUE, each element contains dist and either gamma or acov, with one row for each unique pair.

References

Cressie, N & Hawkins, D.M. 1980. Robust estimation of the variogram. *Journal of the International Association for Mathematical Geology*, **12**, 115-125. 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)
Torgegram(Summer_mn ~ 1, mf04p, eacf = TRUE)
```

`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).

Details

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

- `tailup_de`: the tailup (downstream-moving) random error variance.
- `taildown_de`: the taildown (upstream-moving) random error variance.
- `euclid_de`: the Euclidean random error variance.
- `nugget`: the nugget (spatially independent) random error variance.
- random effects: if `object` was fit with a `random` argument, one additional variance parameter per named random effect term (e.g., a random intercept's grouping variable), representing the variance attributable to that grouping.

The proportion of variability explained by the fixed effects is the pseudo R-squared returned by `pseudoR2()`. The remaining `1 - pseudoR2` proportion is then split among `tailup_de`, `taildown_de`, `euclid_de`, `nugget`, and any random effect variances, in proportion to their share of the total variance (the sum of `tailup_de`, `taildown_de`, `euclid_de`, `nugget`, and all random effect variances).

Value

A tibble that partitions the total variability by the fixed effects and each variance parameter (see Details). 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, type = "fixed", ...)

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

Arguments

object	A fitted model object from ssn_lm() or ssn_glm().
type	For type = "fixed" (the default), the variance-covariance matrix of the fixed effects. If Satterthwaite degrees of freedom were calculated, type = "cov" returns the variance-covariance matrix of the covariance parameters. In SSN2, type = "ssn" returns the tailup, taildown, Euclidean, and nugget subset; "tailup", "taildown", "euclid", and "nugget" return their individual subsets. type = "randcov" returns the variance-covariance matrix of just the random effects (if relevant).
...	Other arguments. Not used (needed for generic consistency).
var_correct	A logical indicating whether to return the corrected variance-covariance matrix for models fit using ssn_glm() (when family is different from "Gaussian"). The default is TRUE.

Value

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

Examples

```
# 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

∗ **datasets**
    mf04p, 32

AICc(), *22*
anova.SSN2, 3
anova.SSN2(), *50*
anova.ssn_glm *(*anova.SSN2*)*, 3
anova.ssn_lm *(*anova.SSN2*)*, 3
augment.SSN2, 5
augment.SSN2(), *14*, *22*, *24, 25*, *49*, *115*
augment.ssn_glm *(*augment.SSN2*)*, 5
augment.ssn_lm *(*augment.SSN2*)*, 5

coef.SSN2, 8
coef.ssn_glm *(*coef.SSN2*)*, 8
coef.ssn_lm *(*coef.SSN2*)*, 8
coefficients.ssn_glm *(*coef.SSN2*)*, 8
coefficients.ssn_lm *(*coef.SSN2*)*, 8
conditional.SSN2, 10
conditional.ssn_glm *(*conditional.SSN2*)*, 10
conditional.ssn_lm *(*conditional.SSN2*)*, 10
confint.SSN2, 12
confint.ssn_glm *(*confint.SSN2*)*, 12
confint.ssn_lm *(*confint.SSN2*)*, 12
cooks.distance.SSN2, 13
cooks.distance.SSN2(), *24, 25*, *49*
cooks.distance.ssn_glm *(*cooks.distance.SSN2*)*, 13
cooks.distance.ssn_lm *(*cooks.distance.SSN2*)*, 13
copy_lsn_to_temp, 14
covmatrix.SSN2, 15
covmatrix.ssn_glm *(*covmatrix.SSN2*)*, 15
covmatrix.ssn_lm *(*covmatrix.SSN2*)*, 15
create_netgeom, 16, *83*, *85*

deviance.SSN2, 17
deviance.SSN2(), *22*
deviance.ssn_glm *(*deviance.SSN2*)*, 17
deviance.ssn_lm *(*deviance.SSN2*)*, 17
dispersion_initial(), *75*

euclid_initial *(*ssn_initial*)*, 86
euclid_initial(), *75*, *80*, *91*, *95*, *99*
euclid_params *(*ssn_params*)*, 98
euclid_params(), *9*, *57*, *61*, *63*, *66*, *105*

fitted.SSN2, 18
fitted.ssn_glm *(*fitted.SSN2*)*, 18
fitted.ssn_lm *(*fitted.SSN2*)*, 18
fitted.values.ssn_glm *(*fitted.SSN2*)*, 18
fitted.values.ssn_lm *(*fitted.SSN2*)*, 18
formula.SSN2, 20
formula.ssn_glm *(*formula.SSN2*)*, 20
formula.ssn_lm *(*formula.SSN2*)*, 20

glance.SSN2, 21
glance.SSN2(), *8*, *115*
glance.ssn_glm *(*glance.SSN2*)*, 21
glance.ssn_lm *(*glance.SSN2*)*, 21
glances.SSN2, 22
glances.ssn_glm *(*glances.SSN2*)*, 22
glances.ssn_lm *(*glances.SSN2*)*, 22

hatvalues.SSN2, 23
hatvalues.SSN2(), *14*, *25*, *49*
hatvalues.ssn_glm *(*hatvalues.SSN2*)*, 23
hatvalues.ssn_lm *(*hatvalues.SSN2*)*, 23

influence.SSN2, 24
influence.SSN2(), *14*, *24*, *49*
influence.ssn_glm *(*influence.SSN2*)*, 24
influence.ssn_lm *(*influence.SSN2*)*, 24

kcv.SSN2, 25
kcv.ssn_glm *(*kcv.SSN2*)*, 25
kcv.ssn_lm *(*kcv.SSN2*)*, 25

labels.SSN2, 28

labels.ssn_glm *(labels.SSN2)*, 28
labels.ssn_lm *(labels.SSN2)*, 28
list, *72*
logLik.SSN2, 29
logLik.SSN2(), *22*
logLik.ssn_glm *(logLik.SSN2)*, 29
logLik.ssn_lm *(logLik.SSN2)*, 29
loocv(), *25*, *26*
loocv.SSN2, 30
loocv.ssn_glm *(loocv.SSN2)*, 30
loocv.ssn_lm *(loocv.SSN2)*, 30

mf04p, 32, *35*
MiddleFork04.ssn, 32, *32*
model.frame.SSN2, 35
model.frame.ssn_glm *(model.frame.SSN2)*, 35
model.frame.ssn_lm *(model.frame.SSN2)*, 35
model.matrix.SSN2, 36
model.matrix.ssn_glm *(model.matrix.SSN2)*, 36
model.matrix.ssn_lm *(model.matrix.SSN2)*, 36

nobs.SSN2, 37
nobs.ssn_glm *(nobs.SSN2)*, 37
nobs.ssn_lm *(nobs.SSN2)*, 37
nugget_initial *(ssn_initial)*, 86
nugget_initial(), *75*, *91*, *99*
nugget_params *(ssn_params)*, 98
nugget_params(), *9*, *57*, *63*, *66*, *105*

plot.SSN2, 38, *40*
plot.ssn_glm *(plot.SSN2)*, 38
plot.ssn_lm *(plot.SSN2)*, 38
plot.Torgegram, 39
plot.Torgegram(), *39*, *117*
predict.SSN2 *(predict.ssn_decorrelate)*, 40
predict.SSN2(), *8*, *26*, *30*
predict.ssn_decorrelate, 40
predict.ssn_glm *(predict.ssn_decorrelate)*, 40
predict.ssn_glm(), *7*
predict.ssn_lm, *52*, *55*
predict.ssn_lm *(predict.ssn_decorrelate)*, 40
predict.ssn_lm(), *7*
predict.ssn_lmRF *(predict.ssn_decorrelate)*, 40
print.anova.ssn_glm *(print.ssn_decorrelate)*, 45
print.anova.ssn_lm *(print.ssn_decorrelate)*, 45
print.SSN, 44
print.SSN2, *114*
print.SSN2 *(print.ssn_decorrelate)*, 45
print.ssn_decorrelate, 45
print.ssn_decorrelate_grid *(ssn_decorrelate_grid)*, 65
print.ssn_glm *(print.ssn_decorrelate)*, 45
print.ssn_lm *(print.ssn_decorrelate)*, 45
print.ssn_lmRF *(print.ssn_decorrelate)*, 45
print.summary.ssn_glm *(print.ssn_decorrelate)*, 45
print.summary.ssn_lm *(print.ssn_decorrelate)*, 45
print.summary.ssn_lmRF *(print.ssn_decorrelate)*, 45
pseudoR2(), *22*, *118*
pseudoR2.SSN2, 47
pseudoR2.ssn_glm *(pseudoR2.SSN2)*, 47
pseudoR2.ssn_lm *(pseudoR2.SSN2)*, 47

randomForest::randomForest(), *57*
ranger::ranger(), *57*, *97*
resid.ssn_glm *(residuals.SSN2)*, 48
resid.ssn_lm *(residuals.SSN2)*, 48
residuals.SSN2, 48
residuals.SSN2(), *14*, *24*, *25*
residuals.ssn_glm *(residuals.SSN2)*, 48
residuals.ssn_lm *(residuals.SSN2)*, 48
rstandard.ssn_glm *(residuals.SSN2)*, 48
rstandard.ssn_lm *(residuals.SSN2)*, 48

satterthwaite, *95*
satterthwaite(), *5*
satterthwaite.SSN2, 49
satterthwaite.ssn_glm *(satterthwaite.SSN2)*, 49
satterthwaite.ssn_lm *(satterthwaite.SSN2)*, 49
spmodel::dispersion_initial(), *89*
spmodel::esv(), *77*, *93*
spmodel::predict.spmodel(), *27*

spmodel::randcov_initial(), *76*, *86*, *89*, *91*
spmodel::randcov_params(), *58*, *63*, *66*, *98*, *105*
ssn_create_bigdist, 51
ssn_create_bigdist(), *76*, *92*
ssn_create_distmat, *32*, 53, *72*, *83*, *109*
ssn_create_distmat(), *33*, *73*
ssn_decorrelate, 56
ssn_decorrelate(), *41*, *46*, *59*, *64, 65*
ssn_decorrelate_data, 62
ssn_decorrelate_data(), *62*, *68*
ssn_decorrelate_grid, 65
ssn_decorrelate_newdata, 67
ssn_decorrelate_newdata(), *65*, *101*
ssn_get_data, 69
ssn_get_data(), *100*
ssn_get_netgeom, 70
ssn_get_stream_distmat, 71
ssn_glm, 74, *83*
ssn_glm(), *4*, *6*, *9*, *12, 13*, *15*, *18–23*, *25, 26*, *28–30*, *35–38*, *41*, *46–48*, *50*, *86, 87*, *114, 115*, *118, 119*
ssn_import, *16*, 82, *85*, *109, 110*
ssn_import(), *32*
ssn_import_predpts, *83*, 84
ssn_initial, 86
ssn_lm, *83*, 90
ssn_lm(), *4*, *6*, *9*, *12, 13*, *15*, *18–23*, *25, 26*, *28–30*, *35–38*, *41*, *46–48*, *50*, *57*, *63*, *66*, *75*, *86, 87*, *97*, *114, 115*, *118, 119*
ssn_lmRF, 96
ssn_lmRF(), *41*, *46*, *114*
ssn_names, 98
ssn_params, 98
ssn_put_data, 100
ssn_put_data(), *69*
ssn_rbeta *(*ssn_simulate*)*, 102
ssn_rbinom *(*ssn_simulate*)*, 102
ssn_recorrelate_newdata, 101
ssn_recorrelate_newdata(), *65*
ssn_rgamma *(*ssn_simulate*)*, 102
ssn_rinvgauss *(*ssn_simulate*)*, 102
ssn_rnbinom *(*ssn_simulate*)*, 102
ssn_rnorm *(*ssn_simulate*)*, 102
ssn_rnorm(), *105*
ssn_rpois *(*ssn_simulate*)*, 102
ssn_simulate, 102
ssn_split_predpts, 106
ssn_subset, 109
SSN_to_SSN2, 110
ssn_update_path, 111
ssn_write, 112
stats::AIC(), *22*
stats::BIC(), *22*
stats::glm, *75*
stats::model.frame(), *36*
stats::model.matrix(), *36*
stats::predict.lm(), *43*
summary.SSN, 113
summary.SSN2, 113
summary.ssn_glm *(*summary.SSN2*)*, 113
summary.ssn_lm *(*summary.SSN2*)*, 113
summary.ssn_lmRF *(*summary.SSN2*)*, 113

taildown_initial *(*ssn_initial*)*, 86
taildown_initial(), *60*, *75*, *79*, *91*, *94*, *99*
taildown_params *(*ssn_params*)*, 98
taildown_params(), *9*, *57*, *63*, *66*, *105*
tailup_initial *(*ssn_initial*)*, 86
tailup_initial(), *59*, *75*, *79*, *91*, *93*, *99*
tailup_params *(*ssn_params*)*, 98
tailup_params(), *9*, *57*, *63*, *66*, *105*
tidy.anova.ssn_glm *(*anova.SSN2*)*, 3
tidy.anova.ssn_lm *(*anova.SSN2*)*, 3
tidy.SSN2, 114
tidy.SSN2(), *8*, *22*
tidy.ssn_decorrelate *(*ssn_decorrelate*)*, 56
tidy.ssn_decorrelate_grid *(*ssn_decorrelate_grid*)*, 65
tidy.ssn_glm *(*tidy.SSN2*)*, 114
tidy.ssn_lm *(*tidy.SSN2*)*, 114
Torgegram, 115
Torgegram(), *39*

varcomp.SSN2, 118
varcomp.ssn_glm *(*varcomp.SSN2*)*, 118
varcomp.ssn_lm *(*varcomp.SSN2*)*, 118
vcov.SSN2, 119
vcov.ssn_glm *(*vcov.SSN2*)*, 119
vcov.ssn_lm *(*vcov.SSN2*)*, 119

xgboost::xgboost(), *57*