Package 'STEvaluationPaper'

May 31, 2019

Version 0.5-1

Date 2018-05-15

Title Evaluation procedures for forecasting with spatio-temporal data

Description Research compendium associated with paper ``Evaluation procedures for forecasting with spatio-temporal data", authored by Mariana Oliveira, Luis Torgo and Vitor Santos Costa. Presented at ECML-PKDD 2018.

Author Mariana Oliveira, Luis Torgo, Vitor Santos Costa

Maintainer Mariana Oliveira <mariana.r.oliveira@inesctec.pt>

Depends R (>= 3.1.0)

Imports assertthat (>= 0.2.0),
dplyr (>= 0.7.4),
foreach (>= 1.4.3),
Hmisc (>= 4.0-0),
lwgeom (>= 0.1-3),
sf (>= 0.6-0),
spdep (>= 0.6-8),
starma (>= 1.3),
stringr (>= 1.2.0),
tidyr (>= 0.6.0),
wavethresh (>= 4.6.8)

Suggests doParallel (>= 1.0.10),
DMwR2 (>= 0.0.2),
ggplot2 (>= 2.1.0),
ranger (>= 0.9.0),
uba (>= 0.7.8),
knitr (>= 1.6),
rmarkdown (>= 1.17)

URL https://github.com/mrfoliveira/Evaluation-procedures-for-forecasting-with-spatio-temporal-data/

License GPL (>=2)

Encoding latin1

RoxygenNote 6.1.1

VignetteBuilder knitr

R topics documented:

`add_ratios` *Add ratios between spatio-temporal neighborhood indicators*

Description

Add ratios between spatio-temporal neighborhood indicators

Usage

```
add_ratios(df, var, indStat = "mean")
```

Arguments

`df`	A data frame of spatio-temporal indicators. Column names should be of type <variable name>_<indStat>_<radius>.
`var`	A character string with the name of the variable with indicators to add ratios
`indStat`	The name of the summarizing stat that was used to calculate the indicators

Value

A data frame including the original data of `df` and additional ratios between the indicators of subsequent radiuses.

See Also

`get_st_indicators`

ae *Calculate vector of error values*

Description

Calculate vector of error values

Usage

```
ae(y, y_hat)

se(y, y_hat)
```

Arguments

y	a vector of true values
y_hat	a vector of predicted values

Value

a vector of error values

Functions

- ae: absolute error
- se: squared error

compressAllRes *Compress results from all (artificial) experiments*

Description

Compress results from all (artificial) experiments

Usage

```
compressAllRes(all.res, rmAllRaw = F)
```

Arguments

all.res	a list with multiple levels (model, grid size, series size and lists of full results of multiple experiments)
rmAllRaw	a boolean indicating whether the whole rawRes should be removed (defaults to FALSE). If TRUE, only "train" data will be removed from each set of results

Value

A multi-level list containing compressed results. Either all rawRes is removed, or `train` is substituted by a vector of the number of instances, time and location IDs in the training set, in both `out_estRes` and `in_estRes`.

See Also

`summarize_all_art_exps`, `run_all_experiments`

`compressRes` *Compress results from one experiment*

Description

Compress results from one experiment

Usage

```
compressRes(res, rmAllRaw = F)
```

Arguments

`res`	A list containing full results of one experiment (out_estRes and in_estRes)
`rmAllRaw`	a boolean indicating whether the whole rawRes should be removed (defaults to FALSE). If TRUE, only `train` data will be substituted by a vector of the number of instances, time and location IDs in the training set

Value

A list containing compressed results of one experiment. Either all rawRes is removed, or `train` substituted by a vector of the number of instances, time and location IDs in the training set in both `out_estRes` and `in_estRes`

See Also

`summarize_one_exp`, `run_one_experiment`

cv_folds *Cut into folds*

Description

Assigns rows of a data frame into folds for cross-validation.

Usage

```
cv_folds(x, nfolds)
```

Arguments

x	a data.frame
nfolds	number of folds

Value

a vector with the fold assignment of each row

data_list *Spatio-temporal data sets*

Description

Spatio-temporal data from multiple sources used for the experimental section in "Evaluation procedures for forecasting with spatio-temporal data".

Usage

```
data(data_list)
```

Format

An object of class `list`. Each slot in list contains another list with objects `stations` of class `sf`, and `df` of class `data.frame` with columns time, station, and value.

Source

MESA NCDC TCE COOK SAC RURAL BEIJ

References

- Sonja Pravilovic, Annalisa Appice, and Donato Malerba. Leveraging correlation across space and time to interpolate geophysical data via CoKriging. Int. J. Geogr. Inf. Sci., 32(1):191–212, 2018.
- Tomislav Hengl.GSIF: Global Soil Information Facilities, 2017.
- Caley K Gasch, Tomislav Hengl, Benedikt Graler, Hanna Meyer, Troy SMagney, and David J Brown. Spatio-temporal interpolation of soil water, temperature, and electrical conductivity in 3D+ T: The Cook Agronomy Farm data set.Spat. Stat., 14:70–90, 2015.
- Edzer Pebesma. spacetime: Spatio-Temporal Data in R. J. Stat. Softw., 51(7):1–30, 2012.
- Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. U-Air: When Urban Air QualityInference Meets Big Data. In Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., KDD '13, pages 1436–1444, New York, NY, USA, 2013.ACM.

`df2site_sf` *Create an sf object of available sites*

Description

Extracts the location information from a data frame and transforms into a `sf` object.

Usage

```
df2site_sf(df, site_id, lon, lat, crs)
```

Arguments

`df`	a data frame of the data set
`site_id`	the name of the column containing location IDs
`lon`	the name of the column containing the location's longitude
`lat`	the name of the column containing the location's latitude
`crs`	the code for the Coordinate Reference System

Value

a sf object, containing the geographic information for each location in `df`

See Also

`st_as_sf`

embed_series *Embed each time series in a spatio-temporal data set*

Description

Embed each time series in a spatio-temporal data set

Usage

```
embed_series(df, var, k, time = "time", station_id = "station")
```

Arguments

df	data frame
var	a character string, the name of the variable to embed
k	a numeric, the embed size
time	a character string, the column name identifying the time of observation
station_id	a character string, the column name identifying the location of observation

Value

A data frame with extra columns var_Tm1, var_Tm2, ..., var_Tm\(k-1\)

estimates *Estimate error using a chosen method*

Description

Estimate error using a chosen method

Usage

```
estimates(data, form, estimator = "kf_xval", est.pars = list(nfolds =
  10, fold.alloc.proc = "Trand_SPrand"), workflow = "simple_workflow",
  wf.pars = NULL, evaluator = "evaluate", eval.pars = NULL,
  seed = 1234)
```

Arguments

data	a data frame
form	a formula for learning
estimator	the name of an error estimator function
est.pars	a named list of arguments to feed to estimator
workflow	the name of the workflow to use for making predictions
wf.pars	a named list of arguments to feed to workflow
evaluator	the name of the function to use to calculate evaluation results
eval.pars	a named list of arguments to feed to evaluator
seed	a seed to set before performing estimates

Value

The results of evaluator after applying estimator to the learning task

---

evaluate *Evalute the results of a predictive workflow*

Description

Calculate evaluation metrics from the raw results of a workflow

Usage

```
evaluate(wfRes, eval.function = get("regressionMetrics",
  asNamespace("performanceEstimation")), .keptTrain = TRUE, ...)
```

Arguments

wfRes	a data frame (or list of data frames) containing the results of a predictive workflow with columns trues and preds containing the real and predicted values, respectively
eval.function	the function to be used to calculate error metrics from wfRes
.keptTrain	a Boolean indicating whether .keepTrain was set to TRUE in calls to estimation methods. Only useful if evaluation metrics need training data.
...	parameters to pass to eval.function

Value

The results (or a list of results) of eval.function applied to the data frame (or list of data frames) in wfRes

exp_c *Exponential function*

Description

Calculate exp(-x/C).A list of matrices like the ones returned by consecutive use of spdep::dnearneigh and spdep::nblag where a value higher than 0 implies that the row and column locations are neighbours

Usage

```
exp_c(x, C = 10000)
```

Arguments

x	A vector
C	A constant

Value

A vector exp(-x/C)

generate_coef *Generate coefficients for a STARMA stationary process*

Description

Generate coefficients for a STARMA stationary process

Usage

```
generate_coef(coef_specs = list(c_10 = c(-2, 2), c_11 = c(-2, 2), c_20 =
  c(-1, 1), c_21 = 0), type = "STAR", ndigits = 3)
```

Arguments

coef_specs	A named list of coefficient specifications for the models. Each slot in the list should itself contain a named list with slots c_10, c_11, c_20, c_21 containing either a number which will be used directly for the coefficient of that order in the AR and/or the MA components of the model, or a vector specifying an interval whithin which a coefficient will be randomly generated.
type	A vector of the types of STARMA models to be used for data generation. Should be STAR,STARMA,NL_STAR or STMA.
ndigits	Number of digits to keep when rounding coefficients

Value

A vector of stationary coefficients generated with names `phi_10`, `phi_11`, `phi_20`, `phi_21`, `theta_10`, `theta_11`, `theta_20`, `theta_21` and `FUN`.

See Also

`starma_stat_check`

`generate_grid` *Create a regular grid using spdep package*

Description

Create a regular grid with a certain number of locations, returning each location's position and neighbour matrices of order 0 and 1.

Usage

```
generate_grid(Nsites, grid.h = ceiling(sqrt(Nsites)),
  grid.w = ceiling(sqrt(Nsites)))
```

Arguments

`Nsites`	Number of locations in the grid
`grid.h`	Height of the grid (in number of locations). Defaults to `sqrt(Nsites)`
`grid.w`	Width of the grid (in number of locations). Defaults to `sqrt(Nsites)`

Value

A list with slots sites – a data frame with columns id, x and y –, and klist – a list of neighbour matrices of order 0 and 1 where the values are calculated using functions from `spdep` package.

See Also

`dnearneigh`, `nblag`, `nb2mat`

`generate_multiple_datasets` *Generate multiple artificial datasets using STARMA on regular grids*

Description

This function generates multiple regular grids and simulates time series for each location according to one or more STARMA model specifications.

Usage

```
generate_multiple_datasets(Nsites, Ntimes, mtypes, coef_specs, ncoefs,
  trash = 0, grid.h = ceiling(sqrt(Nsites)),
  grid.w = ceiling(sqrt(Nsites)), init_seed = 1234, mid_seed = NULL,
  sim_seed = NULL)
```

Arguments

`Nsites`	A vector containing the number of locations in the regular grids to be generated
`Ntimes`	A vector containing the number of time series points to be generated for each location
`mtypes`	A vector of the types of STARMA models to be used for data generation. Should be `STAR`,`STARMA`,`NL_STAR` or `STMA`.
`coef_specs`	A named list of coefficient specifications for the models. Each slot in the list should itself contain a named list with slots `c_10`, `c_11`, `c_20`, `c_21` containing either a number which will be used directly for the coefficient of that order in the AR and/or the MA components of the model, or a vector specifying an interval whithin which a coefficient will be randomly generated. Coefficients are only randomly generated for the first pair of `Nsites` and `Ntimes`, and then are re-used for different grid and time series sizes.
`ncoefs`	A vector with the number of data sets that should be generated according to each of the specifications in the `coef_specs` list.
`trash`	A number of initial time series points to be discarded at each location.
`grid.h`	Height of the grid (in number of locations). Defaults to `sqrt(Nsites)`.
`grid.w`	Width of the grid (in number of locations). Defaults to `sqrt(Nsites)`.
`init_seed`	Seed to set at the begining (before coefficient generation). Default is `1234`.
`mid_seed`	Seed to set between grid/time series size change. Default is `NULL`.
`sim_seed`	Seed to feed to `generate_one_dataset`. Default is `NULL`.

Value

A list containing a list for each size in `Nsites`. Each of the sublists contains a list of datasets generated by `generate_one_dataset` for each time series size specified in `Ntimes`. Each of these sublists will have `sum(length(mtypes) x ncoefs)` datasets of each grid and time series size. In total, `length(Nsites) x length(Ntimes) x sum(length(mtypes) x ncoefs)` spatio-temporal datasets are generated.

See Also

generate_one_dataset

---

generate_one_dataset *Generate an artificial dataset using STARMA on a regular grid*

Description

This function generates a regular grid and simulates a time series for each location according to a STARMA model specification.

Usage

```
generate_one_dataset(Nsites, Ntimes, coef_spec, gen_coef = TRUE,
  mtype = NULL, trash = 0, ndigits = 3, seed = NULL,
  grid.h = ceiling(sqrt(Nsites)), grid.w = ceiling(sqrt(Nsites)))
```

Arguments

Nsites	Number of locations in the grid
Ntimes	Number of points in each time series to generate
coef_spec	Coefficient specifications. Should be a named list with slots c_10, c_11, c_20, c_21 containing either a number which will be used directly for the coefficient of that order in the AR and/or the MA components of the model, or a vector specifying an interval whithin which a coefficient will be randomly generated.
gen_coef	A Boolean indicating whether coef_spec should be used to generate coefficients (TRUE: default) or whether coef_spec already contains a proper list of coefficients
mtype	Type of STARMA model (should only be provided if gen_coef is TRUE. One of STAR,STARMA,NL_STAR or STMA.
trash	A number of initial time series points to be discarded at each location.
ndigits	Number of digits to keep when rounding coefficients.
seed	Seed to feed to generate_stdata. Default is NULL.
grid.h	Height of the grid (in number of locations). Defaults to sqrt(Nsites)
grid.w	Width of the grid (in number of locations). Defaults to sqrt(Nsites)

Value

A list with three slots: data, a matrix of spatio-temporal data (columns correspond to locations, rows to time-stamps); coef, a vector of the coefficients actually used to generate the data; and grid, a list containing the positions of location and the neighbour matrices.

@seealso generate_stdata, generate_coef, generate_grid

generate_stdata *Simulate a STARMA stationary process*

Description

Simulate a STARMA stationary process

Usage

```
generate_stdata(Ntimes, klist, coef, scale = FALSE, trash = 100,
  seed = NULL)
```

Arguments

Ntimes	Length of time series to generate
klist	A list of matrices like the ones returned by consecutive use of spdep::dnearneigh and spdep::nblag where a value higher than 0 implies that the row and column locations are neighbours
coef	A named vector of stationary stationary coefficients with names phi_10, phi_11, phi_20, phi_21, theta_10, theta_11, theta_20, theta_21 and FUN.
scale	A boolean indicating whether the data should be scaled around 0.
trash	Number of initial time series points to discard
seed	Seed to feed to starma_sim. Default is NULL.

Value

A list with two slots: coef, a vector of the coefficients actually used to generate the data; data, a matrix of spatio-temporal data (columns correspond to locations, rows to time-stamps).

@seealso starma_sim

get_all_neib_vals *Get the spatio-temporal neighbourhoods of all observations*

Description

Get the spatio-temporal neighbourhoods of all observations

Usage

```
get_all_neib_vals(df, max_radius, t_dist_mat, s_dist_mat, alpha, vars,
  time_id, site_id, parallel = FALSE, nsplits = 4)
```

Arguments

df	a data frame of observations
max_radius	the maximum spatio-temporal distance allowed to be included in a neighbourhood
t_dist_mat	a matrix of normalized temporal distances between time-stamps (rownames and colnames should be a concatenation of "TIME_" and the time-stamp)
s_dist_mat	a matrix of normalized spatial distances between locations (rownames and colnames should be a concatenation of "SITE_" and the location IDs)
alpha	a weighting factor for the spatio-temporal distance
vars	Vector of character strings indicating the columns whose values should be retrieved
time_id	the name of the column containing time-stamps
site_id	the name of the column containing location IDs
parallel	Boolean indicating whether the code should run in parallel. Default is FALSE
nsplits	Number of subsets of rows to split the data frame into so they can be processed in parallel

Details

The spatio-temporal distance is defined as

$$D_{i,j} = d_{i,j} x \alpha + t_{i,j} x (1 - \alpha)$$

where d_i,j is the spatial distance between locations, t_i,j is the temporal distance between timestapms and α is a weighting factor. Note that the `radius` should always be a number between zero and `min(alpha,alpha-1)`. Also note that if `alpha` is set to 1, then instead of a cone, the neighbourhood will have the shape of a cylinder.

Value

A data frame where each row describes a neighbour, with the first two columns containing the location ID and time-stamp of the central observation, followed by two columns with the neighbouring location ID and time-stamp, a colunm containing the spatio-temporal distance between the two, and a final column containing the values of the variables in df at the neighbouring time and location.

References

Ohashi, Orlando, and Luis Torgo. "Wind speed forecasting using spatio-temporal indicators." ECAI. 2012.

See Also

`get_st_neighbours`

`get_full_indicators` *Get time series embeds and spatio-temporal indicators*

Description

Get time series embeds and spatio-temporal indicators

Usage

```
get_full_indicators(df, stations, k, betas, alpha = 0.5, var = "value",
  stats = c("mean", "weighted.mean", "sd"), ratios2add = c(TRUE, TRUE,
  FALSE), neib_type = "cone", parallel = FALSE, nsplits = 1,
  time_id = "time", site_id = "station")
```

Arguments

`df`	A data frame containing spatio-temporal information
`stations`	An `sf` object containing geographical information on the location of `df`
`k`	A numeric indicating the temporal embed size \(number\)
`betas`	A vector of values defining the maximum spatio-temporal distance allowed for an observation to be considered within a spatio-temporal neighbourhood
`alpha`	a weighting factor for the spatio-temporal distance
`var`	The name of the variable to summarize into indicators
`stats`	A vector containing the names of functions that are to be used to calculate summarizing statistics
`ratios2add`	A vector of Boolean values indicating, for each statistic in `stats` whether ratios between neighorhoods of subsequent sizes should be included as extra columns
`neib_type`	the type of neighborhood to consider. Can be • `cone` (default) - a cone with the center of its base at the observation (spatial radius growing with time) • `reversed` - a cone with its peak at the observation (spatial radius shrinking with time)
`parallel`	Boolean indicating whether the code should run in parallel. Default is FALSE
`nsplits`	Number of subsets of rows to split the data frame into so they can be processed in parallel
`time_id`	The name of the column containing time-stamps in `df`
`site_id`	The name of the column containing location IDs in `df`

Value

A data frame that contains extra columns <var>_Tm1, <var>_Tm2, ..., <var>_Tm<k-1> with previous observations for each location, summary statistics of the values of `var` found within the spatio-temporal neighbourhoods of the one or more `radiuses` of each pair (location ID, time-stamp) and ratios between them

get_spatial_dist_mat *Calculate spatial distance matrix*

Description

A function that calculates the geographical distance matrix between the locations of an `sf` object.

Usage

```
get_spatial_dist_mat(sites_sf, site_id)
```

Arguments

`sites_sf`	an `sf` object with the geograhic information of the locations (as returned by `df2site_sf`)
`site_id`	the column name of the location ID

Value

a matrix of distances. Row and column names are a concatenation of "SITE_" and the location IDs.

See Also

`df2site_sf`

get_st_indicator *Get a spatio-temporal indicator from neighbourhood data frame*

Description

Calculate a spatio-temporal indicator of a certain variable within a spatio-temporal neighborhood of a certain radius.

Usage

```
get_st_indicator(all_neib_vals, stat, radius, ind_name, var,
  time_id = "time", site_id = "site")
```

Arguments

all_neib_vals	a data frame containing information on observations spatio-temporal distance to neighbours and variable values at the neighbouring locations and times.
stat	the name of a function that calculates a statistic (e.g., "mean"). If the stat is "weighted.mean" then the inverse of the spatio-temporal distance is used to weight the values of observations in the spatio-temporal neighbourhood
radius	a value defining the maximum spatio-temporal distance allowed for an observation to be considered within a spatio-temporal neighbourhood
ind_name	the name of the indicator column
var	the name of the variable to summarize into an indicator
time_id	the name of the column containing time-stamps
site_id	the name of the column containing location IDs

Details

The spatio-temporal distance is defined as

$$D_{i,j} = d_{i,j} x \alpha + t_{i,j} x (1 - \alpha)$$

where d_i,j is the spatial distance between locations, t_i,j is the temporal distance between time-stapms and α is a weighting factor. Note that `radius` should always be a number between zero and `min(alpha,alpha-1)`, so the border conditions apply. Also note that if `alpha` is set to 1, then instead of a cone, the neighbourhood will have the shape of a cylinder.

Value

A data frame that contains a summary statistic of the values found within the spatio-temporal neighbourhood of a certain radius of each pair (location ID, time-stamp) in `all_neib_vals`

References

Ohashi, Orlando, and Luis Torgo. "Wind speed forecasting using spatio-temporal indicators." ECAI. 2012.

`get_st_indicators` *Get spatio-temporal indicators from a data frame containing spatio-temporal information*

Description

Calculate spatio-temporal indicators of one or more variables within a spatio-temporal neighborhood of one or more maximum radius (in terms of spatio-temporal distance).

Usage

```
get_st_indicators(df, stations_sf, radiuses = c(0.1), stats = c("mean",
  "sd"), alpha = 0.5, neib_type = "cone", time_id = "time",
  site_id = "site_id", vars = c("value"), parallel = FALSE,
  nsplits = 4)
```

Arguments

df	A data frame containing spatio-temporal information
stations_sf	An sf object containing geographical information on the location of df
radiuses	A vector of values defining the maximum spatio-temporal distance allowed for an observation to be considered within a spatio-temporal neighbourhood
stats	A vector containing the names of functions that are to be used to calculate summarizing statistics
alpha	a weighting factor for the spatio-temporal distance
neib_type	the type of neighborhood to consider. Can be • cone (default) - a cone with the center of its base at the observation (spatial radius growing with time) • reversed - a cone with its peak at the observation (spatial radius shrinking with time)
time_id	The name of the column containing time-stamps in df
site_id	The name of the column containing location IDs in df
vars	The name of the variables to summarize into indicators
parallel	Boolean indicating whether the code should run in parallel. Default is FALSE
nsplits	Number of subsets of rows to split the data frame into so they can be processed in parallel

Details

The spatio-temporal distance is defined as

$$D_{i,j} = d_{i,j} x \alpha + t_{i,j} x (1 - \alpha)$$

where d_i,j is the spatial distance between locations, t_i,j is the temporal distance between time-stapms and α is a weighting factor.

Value

A data frame that contains summary statistics of the values of vars found within the spatio-temporal neighbourhoods of the one or more radiuses of each pair (location ID, time-stamp) in df

References

Ohashi, Orlando, and Luis Torgo. "Wind speed forecasting using spatio-temporal indicators." ECAI. 2012.

`get_st_neighbours` *Get spatio-temporal neighbourhood*

Description

A function that calculates the observations that are within a spatio-temporal neighbourhood of a certain radius of a time and location.

Usage

```
get_st_neighbours(site, time, radius, t_dist_mat, s_dist_mat, alpha,
  time_id = "time", site_id = "site_id")
```

Arguments

`site`	a location ID
`time`	a time-stamp
`radius`	a radius of spatio-temporal distance
`t_dist_mat`	a matrix of normalized temporal distances between time-stamps (rownames and colnames should be a concatenation of "TIME_" and the time-stamp)
`s_dist_mat`	a matrix of normalized spatial distances between locations (rownames and colnames should be a concatenation of "SITE_" and the location IDs)
`alpha`	a weighting factor for the spatio-temporal distance
`time_id`	the name to give to the column of time-stamps (Default: time)
`site_id`	the name to give to the column of location IDs (Default: site_id)

Details

The spatio-temporal distance is defined as

$$D_{i,j} = d_{i,j} x \alpha + t_{i,j} x (1 - \alpha)$$

where d_i,j is the spatial distance between locations, t_i,j is the temporal distance between time-stapms and α is a weighting factor.

Note that `radius` should always be a number between zero and `min(alpha,alpha-1)`. Also note that if `alpha` is set to 1, then instead of a cone, the neighbourhood will have the shape of a cylinder.

Value

A data frame where each row describes a neighbour, with the first two columns containing the location ID and time-stamp of the central observation, followed by two columns with the neighbouring location ID and time-stamp, and a final colunm containing the spatio-temporal distance between the two.

References

Ohashi, Orlando, and Luis Torgo. "Wind speed forecasting using spatio-temporal indicators." ECAI. 2012.

get_time_dist_mat *Calculate temporal distance matrix*

Description

A function that calculates a distance matrix of time-stamps

Usage

```
get_time_dist_mat(times, origin = min(times))
```

Arguments

times	A vector of time-stamps
origin	A date to use as origin for difftime

Value

a matrix of distances. Row and column names are a concatenation of "TIME_" and the time-stamp.

grid_neibs_ord1 *Get directional neighbours from a regular grid*

Description

Given a matrix detailing the position of location in a regular grid, returns a matrix specifying the IDs of immediate neighbours to the right, left, top and bottom of each location.

Usage

```
grid_neibs_ord1(sites, klist)
```

Arguments

sites	a matrix or data.frame with site ID equal to row number and two columns, corresponding to position in x and y
klist	a matrix like the ones returned by consecutive use of spdep::dnearneigh and spdep::nblag where a value higher than 0 implies that the row and column locations are neighbours

Value

A matrix with nrow(sites) rows and four columns containing the ID of a location's neighbour in each cardinal direction (top, right, bottom and left).

identity *Identity function*

Description

Return the identity function.

Usage

```
identity(x)
```

Arguments

x	A vector

Value

The vector x

kf_xval *Cross-validation*

Description

Performs a cross-validation experiment where folds can be allocated in different ways considering time and/or space

Usage

```
kf_xval(data, nfolds, FUN, form, fold.alloc.proc = "Trand_SPrand",
  alloc.pars = NULL, time = "time", site_id = "site",
  .keepTrain = TRUE, ...)
```

Arguments

data	full dataset
nfolds	number of folds for the data set to be separated into. If you would like to set the number of time and space folds separately, nfolds should be set to NULL and t.nfolds and sp.nfolds should be fed as a list to alloc.pars (only available when using fold.alloc.proc set to Tblock_SPchecker, Tblock_SPcontig or Tblock_SPrand).
FUN	function with arguments • train training set • test testing set • time column name of time-stamps

- site_id column name of location identifiers
- form a formula for model learning
- ... other arguments

form a formula for model learning

fold.alloc.proc
name of fold allocation function. Should be one of

- Trand_SPrand – each fold contains completely random observations. The default
- Tall_SPcontig - each fold includes all time and a contiguous block of space
- Tall_SPrand - each fold includes all time and random locations in space
- Tall_SPchecker - each fold includes all time and a set of systematically assigned (checkered) part of space
- Tblock_SPall - each fold includes a block of contiguous time for all locations
- Trand_SPall - each fold includes random time-snapshots of of all locations
- Tblock_SPchecker - each fold includes a block of contiguous time for a systematically assigned (checkered) part of space
- Tblock_SPcontig - each fold includes a block of contiguous time for a block of spatially contiguous locations
- Tblock_SPrand - each fold includes a block of contiguous time for a randomly assigned part of space

alloc.pars parameters to pass onto fold.alloc.proc

time column name of time-stamp in data. Default is "time"

site_id column name of location identifier in data. Default is "site_id"

.keepTrain if TRUE (default), instead of the results of FUN being directly returned, a list is created with both the results and a data.frame with the time and site identifiers of the observations used in the training step.

... other arguments to FUN

Value

If keepTrain is TRUE, a list where each slot corresponds to one repetition or fold, containing a list with slots results containing the results of FUN, and train containing a data.frame with the time and site_id identifiers of the observations used in the training step. Usually, the results of FUN is a data.frame with location identifier site_id, time-stamp time, true values trues and the workflow's predictions preds.

lag_multiple_datasets *Create a spatio-temporal embed of multiple datasets*

Description

Transform multiple spatio-temporal data sets in a list of lists into a list of lists containing data frames that have a spatio-temporal embed. Only data for "interior" points in the grid is kept. That is, points that do not have four immediate neighbours are filtered out.

Usage

```
lag_multiple_datasets(data_list, LAG_use, SLAGS,
  min_time = rep(max(LAG_use), length(LAG_use)))
```

Arguments

data_list	A multi-level list as the ones generated by `generate_multiple_datasets`. The first level corresponds to a certain grid size; the second level to a certain time series size and the third level contains multiple lists containing datasets (that might have been generated from different STARMA specifications).
LAG_use	A vector of the temporal lag orders to use.
SLAGS	A list containing vectors of length of the corresponing temporal lag size. Each of the vectors contains values of 0 meaning no neighbour columns are used at that temporal lag, and/or values of 1 meaning that the values of immediate neighbours are included for that temporal lag.
min_time	A vector of the minimum time-stamps to be included in the datasets. Defaults to `rep(max(LAG_use),length(LAG_use)`.

Value

A list of lists similar to data_list, but with the data sets having spatio-temporal embeds.

See Also

`generate_multiple_datasets`, `st_lag_neib_ord1`

mse *Calculate error metrics*

Description

Calculate error metrics

Usage

```
mse(y, y_hat, na.rm = TRUE)

rmse(y, y_hat, na.rm = TRUE)

mae(y, y_hat, na.rm = TRUE)

nmae(y, y_hat, y_train = NULL, statFUN = stats::median, na.rm = TRUE)

nmse(y, y_hat, y_train = NULL, statFUN = mean, na.rm = TRUE)

nrmse(y, y_hat, y_train = NULL, statFUN = mean, na.rm = TRUE)
```

Arguments

y	a vector of true values
y_hat	a vector of predicted values
na.rm	boolean indicating whether NAs should be removed. Default is TRUE
y_train	a vector of training values
statFUN	summary statistic to use for normalization. Default is median for nmae and mean for nmse.

Value

one error value

Functions

- mse: mean squared error
- rmse: mean squared error
- mae: mean absolute error
- nmae: normalized mean absolute error
- nmse: normalized mean squared error
- nrmse: normalized root mean squared error

---

nd_kf_xval *Non-dependent cross-validation*

Description

Performs a cross-validation experiment where folds can be allocated in different ways considering time and/or space and a certain buffer around the testing set time and/or space is removed from the training set.

Usage

```
nd_kf_xval(data, nfolds, FUN, form, fold.alloc.proc = "Trand_SPrand",
  alloc.pars = NULL, t.buffer = NULL, s.buffer = NULL,
  s.dists = NULL, t.dists = NULL, time = "time", site_id = "site",
  .keepTrain = TRUE, ...)
```

Arguments

data	full dataset
nfolds	number of folds for the data set to be separated into. If you would like to set the number of time and space folds separately, nfolds should be set to NULL and t.nfolds and sp.nfolds should be fed as a list to alloc.pars (only available when using fold.alloc.proc set to Tblock_SPchecker, Tblock_SPcontig or Tblock_SPrand).
FUN	function with arguments • train training set • test testing set • time column name of time-stamps • site_id column name of location identifiers • form a formula for model learning • ... other arguments
form	a formula for model learning
fold.alloc.proc	name of fold allocation function. Should be one of • Trand_SPrand – each fold contains completely random observations. The default • Tall_SPcontig - each fold includes all time and a contiguous block of space • Tall_SPrand - each fold includes all time and random locations in space • Tblock_SPrand - each fold includes a block of contiguous time for a randomly assigned part of space • Tblock_SPall - each fold includes a block of contiguous time for all locations
alloc.pars	parameters to pass onto fold.alloc.proc
t.buffer	numeric value with the distance of the temporal buffer between training and test sets. For each instance in the test set, instances that have a temporal distance of t.buffer or less at the same point in space are removed from the training set.
s.buffer	numeric value with the maximum distance of the spatial buffer between training and test sets. For each instance in the test set, instances that have a spatial distance of s.buffer or less at the same point in time are removed from the training set.
s.dists	a matrix of the distances between the spatial IDs in data. The column names and row names should be of type "SITE_<site_id>"

t.dists	a matrix of the distances between the time-stamps in data. The column names and row names should be of type "TIME_<time>"
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site_id"
.keepTrain	if TRUE (default), instead of the results of FUN being directly returned, a list is created with both the results and a data.frame with the time and site identifiers of the observations used in the training step.
...	other arguments to FUN

Value

If keepTrain is TRUE, a list where each slot corresponds to one repetition or fold, containing a list with slots results containing the results of FUN, and train containing a data.frame with the time and site_id identifiers of the observations used in the training step. Usually, the results of FUN is a data.frame with location identifier site_id, time-stamp time, true values trues and the workflow's predictions preds.

---

norm_scale *Feature scaling*

Description

Normalize values to be within the range between [0,1].

Usage

```
norm_scale(x)
```

Arguments

x	a vector of values

Value

a scaled vector

prequential_eval *Prequential evaluation*

Description

Performs an evaluation procedure where training and test sets can be allocated in different ways, while always respecting the ordering provided by time (models are trained in the past and tested in the relative future).

Usage

```
prequential_eval(data, nfolds, FUN, form, window = "growing",
  fold.alloc.proc = "Tblock_SPall", alloc.pars = NULL,
  removeSP = FALSE, time = "time", site_id = "site",
  .keepTrain = TRUE, ...)
```

Arguments

data	full dataset
nfolds	number of folds for the data set to be separated into. If you would like to set the number of time and space folds separately, nfolds should be set to NULL and t.nfolds and sp.nfolds should be fed as a list to alloc.pars (only available when using fold.alloc.proc set to Tblock_SPchecker, Tblock_SPcontig or Tblock_SPrand).
FUN	function with arguments • train training set • test testing set • time column name of time-stamps • site_id column name of location identifiers • form a formula for model learning • ... other arguments
form	a formula for model learning
window	type of blocked-time window ordering considered. Should be one of • growing - for each time block being tested, all previous time blocks are used for training • sliding - for each time block being tested, the immediately previous time blocks are used for training
fold.alloc.proc	name of fold allocation function. Should be one of • Tblock_SPall - each fold includes a block of contiguous time for all locations • Tblock_SPchecker - each fold includes a block of contiguous time for a systematically assigned (checkered) part of space

	• Tblock_SPcontig - each fold includes a block of contiguous time for a block of spatially contiguous locations • Tblock_SPrand - each fold includes a block of contiguous time for a randomly assigned part of space
alloc.pars	parameters to pass onto fold.alloc.proc
removeSP	argument that determines whether spatio-temporal blocks including the space being used for testing should be removed from the training set. Default is FALSE, meaning the information is not removed
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site_id"
.keepTrain	if TRUE (default), instead of the results of FUN being directly returned, a list is created with both the results and a data.frame with the time and site identifiers of the observations used in the training step.
...	other arguments to FUN

Value

If keepTrain is TRUE, a list where each slot corresponds to one repetition or fold, containing a list with slots results containing the results of FUN, and train containing a data.frame with the time and site_id identifiers of the observations used in the training step. Usually, the results of FUN is a data.frame with location identifier site_id, time-stamp time, true values trues and the workflow's predictions preds.

---

realSumRes2Tab *Transform a multi-level list of summarized results into a table*

Description

Transform a multi-level list of summarized results into a table

Usage

```
realSumRes2Tab(sumRes, statFUN = mean, na.rm = FALSE)
```

Arguments

sumRes	A multi-level list of summarized results where the first level corresponds to learning model used in the experiment, the second level contains results for each data set
statFUN	a function to summarize the evaluation metrics. Default is mean
na.rm	whether to remove NAs in function statFUN

Value

A data frame containing columns identifying the learning model, data set "gold standard"/"real" error/ (that of the out-set), name of error estimator and estimated error (on the in-set), in long format

`regMetrics` *Calculate regression metrics*

Description

Calculate MAE, RMSE and utility-based regression evaluation metrics

Usage

```
regMetrics(trues, preds, y_train = NULL, norm = FALSE,
  aeStatFUN = stats::median, seStatFUN = mean, util = FALSE,
  util.parms = NULL)
```

Arguments

`trues`	a vector of true values
`preds`	a vector of predicted values
`y_train`	a vector of training values
`norm`	a Boolean indicating whether to calculate normalized regression metrics
`aeStatFUN`	a function to calculate a summary of y_train for absolute error normalization. Default is median
`seStatFUN`	a function to calculate a summary of y_train for squared error normalization. Default is mean
`util`	a Boolean indicating whether to calculate utility-based regression metrics
`util.parms`	a named list of parameters to use for calculating utility-based regression metrics. Should contain slots • `phi.parms` - the result of function `phi.control` • `phi.control` - if `phi.parms` is undefined, then `phi.control` can be provided with a list of named arguments to feed function `phi.control` using `y_train`. Default is `list(method = "extremes",extr.type="high")` • `loss.parms` - the results of function `uba::loss.control` • `p` - Default is 0.5 • `thr` - Relevance threshold. Default is 1 • `beta` - Beta for F-measure. Default is 1

Value

a named vector of calculated metrics

responseValues *Get response values of a dataset from a formula*

Description

Get response values of a dataset from a formula

Usage

```
responseValues(formula, data, na = NULL)
```

Arguments

formula	learning formula
data	data set to get the target values from
na	what action to perform if NAs are present. Default is na.fail

Value

A vector of the target values.

run_all_experiments *Run multiple experiments for different grid and time series sizes*

Description

Run multiple experiments for different grid and time series sizes

Usage

```
run_all_experiments(models, nested_data_list, in_set_perc, form,
  in_estimators, in_est.pars, out_estimator = "t_oos",
  out_est.pars = list(tr.perc = in_set_perc),
  workflow = "simple_workflow", wf.pars = NULL,
  evaluator = "evaluate", eval.pars = NULL, seed = 1234,
  site_id = "site", time = "time", .compress = FALSE,
  .progress = NULL, .verbose = TRUE, .saveMem = FALSE)
```

Arguments

models	vector of names of models to use for learning
nested_data_list	a nested list of data sets: top level identifies grid size, second level identifies time series size, and third level contains lists of multiple data sets
in_set_perc	a fraction of the data to be used as in-set
form	a learning formula
in_estimators	a vector of names of estimator functions to use on the in-set data
in_est.pars	a named list of the same length as in_estimators containing lists of arguments to feed to each estimator apllied to the in-set data
out_estimator	the name of the estimator function to use on the out-set data
out_est.pars	a list containing arguments to feed to the estimator apllied to the out-set data
workflow	the name of the function implementing a workflow
wf.pars	a list of arguments to feed to workflow
evaluator	the name of the function to calculate evaluation metrics
eval.pars	a list of arguments to feed to evaluator
seed	a seed to set at the start of the experiment
site_id	the name of the data column containing location identifiers
time	the name of the data column containing time-stamps
.compress	a Boolean indicating whether to compress results
.progress	a file name to save temporary results
.verbose	a Boolean indicating whether progress should be reported on during experiments
.saveMem	a Boolean indicating whether partial results should be discarded to save memory. Only possible if .progress is TRUE. Default is FALSE.

Value

A nested list: top top level identifies grid size, second level identifies time series size, and third level contains lists with two slots: out_estRes containing the results of the out-set estimators, and in_estRes containing a list of results of each estimator used on in-set data

See Also

run_multiple_experiments

`run_multiple_experiments` *Run multiple in-set/out-set error estimation experiments*

Description

Run multiple in-set/out-set error estimation experiments

Usage

```
run_multiple_experiments(data_list, in_set_perc, form, in_estimators,
  in_est.pars, out_estimator = "t_oos", out_est.pars = list(tr.perc =
  in_set_perc), workflow = "simple_workflow", wf.pars = NULL,
  evaluator = "evaluate", eval.pars = NULL, seed = 1234,
  site_id = "site", time = "time")
```

Arguments

`data_list`	a list of data frames
`in_set_perc`	a fraction of the data to be used as in-set
`form`	a learning formula
`in_estimators`	a vector of names of estimator functions to use on the in-set data
`in_est.pars`	a named list of the same length as `in_estimators` containing lists of arguments to feed to each estimator apllied to the in-set data
`out_estimator`	the name of the estimator function to use on the out-set data
`out_est.pars`	a list containing arguments to feed to the estimator apllied to the out-set data
`workflow`	the name of the function implementing a workflow
`wf.pars`	a list of arguments to feed to `workflow`
`evaluator`	the name of the function to calculate evaluation metrics
`eval.pars`	a list of arguments to feed to `evaluator`
`seed`	a seed to set at the start of the experiment
`site_id`	the name of the `data` column containing location identifiers
`time`	the name of the `data` column containing time-stamps

Value

A list containing, for each data in `data_list`, a list with two slots: `out_estRes` containing the results of the out-set estimators, and `in_estRes` containing a list of results of each estimator used on in-set data

See Also

run_one_experiment

`run_one_experiment` *Run one in-set/out-set error estimation experiment*

Description

Run one in-set/out-set error estimation experiment

Usage

```
run_one_experiment(data, in_set_perc, form, in_estimators, in_est.pars,
  out_estimator = "t_oos", out_est.pars = list(tr.perc = in_set_perc),
  workflow = "simple_workflow", wf.pars = NULL,
  evaluator = "evaluate", eval.pars = NULL, seed = 1234,
  site_id = "site", time = "time")
```

Arguments

`data`	a data frame
`in_set_perc`	a fraction of the data to be used as in-set
`form`	a learning formula
`in_estimators`	a vector of names of estimator functions to use on the in-set data
`in_est.pars`	a named list of the same length as `in_estimators` containing lists of arguments to feed to each estimator apllied to the in-set data
`out_estimator`	the name of the estimator function to use on the out-set data
`out_est.pars`	a list containing arguments to feed to the estimator apllied to the out-set data
`workflow`	the name of the function implementing a workflow
`wf.pars`	a list of arguments to feed to `workflow`
`evaluator`	the name of the function to calculate evaluation metrics
`eval.pars`	a list of arguments to feed to `evaluator`
`seed`	a seed to set at the start of the experiment
`site_id`	the name of the `data` column containing location identifiers
`time`	the name of the `data` column containing time-stamps

Value

A list with two slots: `out_estRes` containing the results of the out-set estimators, and `in_estRes` containing a list of results of each estimator used on in-set data

shuffle *Shuffle values/rows*

Description

Shuffle the values or rows of a vector or data frame

Usage

```
shuffle(x)
```

Arguments

x	a vector or data frame

Value

a vector or data frame

simple_workflow *A simple learning and prediction workflow*

Description

A simple learning and prediction workflow

Usage

```
simple_workflow(train, test, form, model = "lm", handleNAs = NULL,
  min_train = 2, nORp = 0.2, time = "time", site_id = "site", ...)
```

Arguments

train	a data frame for training
test	a data frame for testing
form	a formula describing the model to learn
model	the name of the algorithm to use
handleNAs	string indicating how to deal with NAs. If "centralImput", training observations with at least 80% of non-NA columns, will have their NAs substituted by the mean value and testing observatiosn will have their NAs filled in with mean value regardless.
min_train	a minimum number of observations that must be left to train a model. If there are not enough observations, predictions will be NA. Default is 2.

nORp	a maximum number or fraction of columns with missing values above which a row will be removed from train before learning the model. Only works if handleNAs was set to centralImputation. Default is 0.2.
time	the name of the column in train and test containing time-stamps
site_id	the name of the column in train and test containing location IDs
...	other parameters to feed to model

Value

a data frame containing time-stamps, location IDs, true values and predicted values

---

sp_checker *Assign the locations of a regular grid to folds following a checkered pattern*

Description

Systematically assigns the locations of a data frame into folds which are checkered across space for cross-validation. Assumes the sites are sorted (e.g., left to right, bottom to top).

Usage

```
sp_checker(nfolds, nsites, grid.h = sqrt(nsites),
  grid.w = sqrt(nsites))
```

Arguments

nfolds	number of folds to divide the space into
nsites	number of locations in the regular grid
grid.h	height of the grid (in number of sites). Default is sqrt(nfolds)
grid.w	width of the grid (in number of sites). Default is sqrt(nfolds)

Value

a vector with the fold assignment of each location

sp_contig — *Assign the locations of a regular grid to folds in contiguous square blocks.*

Description

Assigns the locations of a data frame into contiguous blocks folds for cross-validation. Assumes the sites are sorted (e.g., left to right, bottom to top). WARNING: Works well for perfect squares that can be divided into `nfolds` perfect squares ONLY.

Usage

```
sp_contig(nfolds, nsites, grid.h = sqrt(nsites), grid.w = sqrt(nsites))
```

Arguments

`nfolds`	number of folds to divide the space into
`nsites`	number of locations in the regular grid
`grid.h`	height of the grid (in number of sites). Default is `sqrt(nfolds)`
`grid.w`	width of the grid (in number of sites). Default is `sqrt(nfolds)`

Value

a vector with the fold assignment of each location

starma_sim — *Simulate spatio-temporal data using a STARMA model*

Description

Generate a spatio-teporal dataset according to a STARMA model.

Usage

```
starma_sim(model, klist, n, rand.gen = stats::rnorm, innov = NULL,
  seed = NULL, FUN = function(x) {     x }, ...)
```

Arguments

`model`	A list with components `ar` and `ma`. Each component contains a matrix of the coefficients where the first index corresponds to the row and the second index to the column.
`klist`	A list of matrices like the ones returned by consecutive use of `spdep::dnearneigh` and `spdep::nblag` where a value higher than 0 implies that the row and column locations are neighbours

n	The length of the time series to be generated
rand.gen	The random generator to be used. Defaults to stats::rnorm
innov	A matrix of initial innovations. Defaults to matrix(rand.gen(n*ncol(klist[[1]]),...),n,ncol(kli
seed	A seed to set before generating the dataset. Defaults to NULL
FUN	A (possibly non-linear) function to apply to the matrix during STAR data generation.
...	Other parameters (?)

Value

A matrix where each column contains the data for a location, and each row contains the data for a time-stamp.

References

See example on page 3 of https://cran.r-project.org/web/packages/starma/starma.pdf

---

starma_stat_check *Check a STARMA model for stationarity*

Description

Checks the coefficients of a STARMA models for stationarity.

Usage

```
starma_stat_check(model)
```

Arguments

model	A list with components ar and ma. Each component contains a matrix of the coefficients as specified in this functions' details.

Details

The stationarity constraints for a STAR model are the following:

- STAR(2_11)

$$-phi_20 + abs(phi_21) < 1$$

$$abs(phi_10 + phi_11) < 1 - phi_20 - phi_21$$

$$abs(phi_10 - phi_11) < 1 - phi_20 + phi_21$$

- STAR(2_10), meaning $phi_21 = 0$

$$phi_20 > -1$$

$$abs(phi_10) + abs(phi_11) < 1 - phi_20$$

- STAR(2_00), meaning $phi_2 1 = 0 \& phi_1 1 = 0$

$$phi_2 0 > -1$$

$$abs(phi_1 0) < 1 - phi_2 0$$

- STAR(1_1), meaning $phi_2 0 = 0 \& phi_2 1 = 0$

$$abs(phi_1 0) + abs(phi_1 1) < 1$$

- STAR(1_0), meaning $phi = 20 = 0 \& phi_2 1 = 0 \& phi_1 1 = 0$

$$abs(phi_1 0) < 1$$

where `phi_10` = `model$ar[1,1]`, `phi_11` = `model$ar[1,2]`, `phi_20` = `model$ar[2,1]`, and `phi_21` = `model$ar[2,2]`

STMA models have the same constraints for theta as the STAR contraints for phi.

Value

`TRUE` if the model is stationary. `FALSE`, otherwise.

References

http://www.tandfonline.com/doi/abs/10.1080/03610918008812173

`st_lag_neib_ord1` *Create a spatio-temporal embed of data*

Description

Given spatio-temporal data, transform it into a data frame where each row has its own value as target variable and, as predictors, lagged values recorded in the past at its own location and/or in the past at neighbouring locations.

Usage

```
st_lag_neib_ord1(data, neibs, p, slags)
```

Arguments

`data`	a matrix of spatio-temporal data where each row contains the variable information for a time point and each column corresponds to a certain location
`neibs`	a data frame with information about the neighbours, like the one returned by `grid_neibs_ord1`
`p`	a number for the order of the temporal lag for values at a specific location
`slags`	a vector with the length of the order of temporal lags for values at immediately neighbouring locations. A value of 0 means no neighbours are used at that temporal lag, a value of 1 means that the values of immediate neighbours are included for that temporal lag.

Value

a data frame with columns time, site, target, as many self_lag columns as p, and as many top, bottom, right and left neighbour value lags as the sum of slags.

See Also

grid_neibs_ord1

---

summarize_all_art_exps

Summarize the results of all (artificial) in-set/out-set experiments

---

Description

Summarize the results of all (artificial) in-set/out-set experiments

Usage

```
summarize_all_art_exps(all.res, statFUN, na.rm)
```

Arguments

all.res	a multi-level list with where the first level corresponds to learning model used in the experiment, the second level contains a list for each grid size of artificial data set, the third level contains a list for each time series size. Inside there is a list for each generated set, the list containing a list of results for each lag embed order
statFUN	a function to summarize the evaluation metrics. Default is mean
na.rm	whether to remove NAs in function statFUN

Value

A data frame containing columns identifying the learning model, grid size, time series size, type of STARMA used to generate the data, order of STARMA used to generate, number of iteration of the generation process with those settings, lag embed order, gold standard error (that of the out-set), name of error estimator and estimated error (on the in-set)

See Also

summarize_multiple_exp

summarize_multiple_exp *Summarize the results of multiple in-set/out-set experiment*

Description

Summarize the results of multiple in-set/out-set experiment

Usage

```
summarize_multiple_exp(multi_exp_res, statFUN = mean, na.rm = FALSE)
```

Arguments

multi_exp_res	a list containing, for each experiment, a list containing two slots: out_estRes containing a named vector of metrics estimated in out-set data, and in_estRes containing a list of data frames where each column corresponds to a metric and each row to a repetition/iteration of an estimator used on in-set data
statFUN	a function to summarize the evaluation metrics. Default is mean
na.rm	whether to remove NAs in function statFUN

Value

A list of data frames of the summarized results of one experiment – each with a first column containing a summary (e.g., the mean) of metrics measured in the out-set data and further columns containing summaries of metrics estimated in the in-set data

See Also

run_multiple_experiments, summarize_one_exp

summarize_one_exp *Summarize the results of one in-set/out-set experiment*

Description

Summarize the results of one in-set/out-set experiment

Usage

```
summarize_one_exp(one_exp_res, statFUN = mean, na.rm = FALSE)
```

Arguments

one_exp_res	a list containing two slots: out_estRes containing a named vector of metrics estimated in out-set data, and in_estRes containing a list of data frames where each column corresponds to a metric and each row to a repetition/iteration of an estimator used on in-set data
statFUN	a function to summarize the evaluation metrics. Default is mean
na.rm	whether to remove NAs in function statFUN

Value

A data frame with a first column containing a summary (e.g., the mean) of metrics measured in the out-set data and further columns containing summaries of metrics estimated in the in-set data

See Also

run_one_experiment

---

sumRes2Tab *Transform a multi-level list of summarized results into a table*

Description

Transform a multi-level list of summarized results into a table

Usage

```
sumRes2Tab(sumRes)
```

Arguments

sumRes	A multi-level list of summarized results where the first level corresponds to learning model used in the experiment, the second level contains a list for each grid size of artificial data set, the third level contains a list for each time series size, and the next level contains a data frame with the results obtained in the out-set (gold-standard or "real" error) as well as estimated errors for different estimators (in wide format)

Value

A data frame containing columns identifying the learning model, grid size, time series size, type of STARMA used to generate the data, order of STARMA used to generate, number of iteration of the generation process with those settings, lag embed order, gold standard error (that of the out-set), name of error estimator and estimated error (on the in-set), in long format

Tall_SPchecker *Systematic spatial CV*

Description

Fold allocation of k-fold CV using:

- all time
- systematically assigned (checkered) individual locations

Usage

```
Tall_SPchecker(data, nfolds, time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold numbers (from 1 to nfolds) of each row in data

Tall_SPcontig *Spatially blocked CV*

Description

Fold allocation of k-fold CV using:

- all time
- contiguously blocked locations

Usage

```
Tall_SPcontig(data, nfolds, time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold numbers (from 1 to nfolds) of each row in data

---

Tall_SPrand *Spatial CV*

Description

Fold allocation of k-fold CV using:

- all time
- shuffled individual locations

Usage

```
Tall_SPrand(data, nfolds, time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold numbers (from 1 to nfolds) of each row in data

Tblock_SPall *Temporally blocked CV*

Description

Fold allocation of k-fold CV using:

- blocked time
- all locations

Usage

```
Tblock_SPall(data, nfolds, time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold numbers (from 1 to nfolds) of each row in data

Tblock_SPchecker *Temporal blocked and systematic spatial CV*

Description

Fold allocation of k-fold CV using:

- blocked time
- systematically assigned (checkered) individual locations

Usage

```
Tblock_SPchecker(data, nfolds, t.nfolds = round(sqrt(nfolds)),
  sp.nfolds = round(sqrt(nfolds)), time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
t.nfolds	number of folds across time. Default is sqrt(nfolds)
sp.nfolds	number of folds across space. Default is sqrt(nfolds)
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold identifiers of each row in data. The fold identifier is composed of the concatenation of time-fold number (from 1 to t.nfolds) and space-fold number (from 1 to sp.nfolds), separated by "_".

Tblock_SPcontig *Temporal blocked and contiguously-blocked spatial CV*

Description

Fold allocation of k-fold CV using:

- blocked time
- contiguously blocked locations

Usage

```
Tblock_SPcontig(data, nfolds, t.nfolds = round(sqrt(nfolds)),
  sp.nfolds = round(sqrt(nfolds)), time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
t.nfolds	number of folds across time. Default is sqrt(nfolds)
sp.nfolds	number of folds across space. Default is sqrt(nfolds)
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold identifiers of each row in data. The fold identifier is composed of the concatenation of time-fold number (from 1 to t.nfolds) and space-fold number (from 1 to sp.nfolds), separated by "_".

Tblock_SPrand *Temporal blocked and randomly assigned spatial CV*

Description

Fold allocation of k-fold CV using:

- blocked time
- randomly assigned locations

Usage

```
Tblock_SPrand(data, nfolds, t.nfolds = round(sqrt(nfolds)),
  sp.nfolds = round(sqrt(nfolds)), time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
t.nfolds	number of folds across time. Default is sqrt(nfolds)
sp.nfolds	number of folds across space. Default is sqrt(nfolds)
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold identifiers of each row in data. The fold identifier is composed of the concatenation of time-fold number (from 1 to t.nfolds) and space-fold number (from 1 to sp.nfolds), separated by "_".

Trand_SPall *Temporal CV*

Description

Fold allocation of k-fold CV using:

- shuffled time
- all locations

Usage

```
Trand_SPall(data, nfolds, time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold numbers (from 1 to nfolds) of each row in data

Trand_SPrand *Classic k-fold CV*

Description

Fold allocation of classic k-fold CV:

- shuffled time
- shuffled locations

Usage

```
Trand_SPrand(data, nfolds, time = "time", site_id = "site")
```

Arguments

data	full dataset
nfolds	number of folds
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site"

Value

A list with slots:

- data, possibly re-ordered
- f, a vector with the fold numbers (from 1 to nfolds) of each row in data

---

t_oos *Time-wise holdout*

Description

Performs one holdout experiment.

Usage

```
t_oos(data, tr.perc, FUN, form, time = "time", site_id = "site",
  .keepTrain = TRUE, ...)
```

Arguments

data	full dataset
tr.perc	percentage of data used for training. Remaining will be used for testing
FUN	function with arguments • train training set • test testing set • time column name of time-stamps • site_id column name of location identifiers • form a formula for model learning • ... other arguments
form	a formula for model learning
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site_id"
.keepTrain	if TRUE (default), instead of the results of FUN being directly returned, a list is created with both the results and a data.frame with the time and site identifiers of the observations used in the training step.
...	other arguments to FUN

Value

The results of FUN. Usually, a data.frame with location identifier site_id, time-stamp time, true values trues and the workflow's predictions preds.

---

t_oos_mc *Time-wise Monte Carlo*

Description

Performs a time-wise Monte Carlo experiment where split points are randomly chosen and a window of previous observations are used for training, with a window of following observations used for testing.

Usage

```
t_oos_mc(data, tr.perc, ts.perc, nreps, FUN, form, time = "time",
  site_id = "site", .keepTrain = TRUE, ...)
```

Arguments

data	full dataset
tr.perc	percentage of data used for training. Remaining will be used for testing
ts.perc	percentage of data used for testing
nreps	number of repetitions/split-points in experiment
FUN	function with arguments • train training set • test testing set • time column name of time-stamps • site_id column name of location identifiers • form a formula for model learning • ... other arguments
form	a formula for model learning
time	column name of time-stamp in data. Default is "time"
site_id	column name of location identifier in data. Default is "site_id"
.keepTrain	if TRUE (default), instead of the results of FUN being directly returned, a list is created with both the results and a data.frame with the time and site identifiers of the observations used in the training step.
...	other arguments to FUN

Value

If keepTrain is TRUE, a list where each slot corresponds to one repetition or fold, containing a list with slots results containing the results of FUN, and train containing a data.frame with the time and site_id identifiers of the observations used in the training step. Usually, the results of FUN is a data.frame with location identifier site_id, time-stamp time, true values trues and the workflow's predictions preds.

Index