Package ‘blockCV’

May 7, 2026

Type Package

Title Spatial and Environmental Blocking for K-Fold and LOO Cross-Validation

Version 3.2-0

Date 2025-08-20

URL https://github.com/rvalavi/blockCV

BugReports https://github.com/rvalavi/blockCV/issues

Maintainer Roozbeh Valavi <valavi.r@gmail.com>

Description Creating spatially or environmentally separated folds for cross-validation to provide a robust error estimation in spatially structured environments; Investigating and visualising the effective range of spatial autocorrelation in continuous raster covariates and point samples to find an initial realistic distance band to separate training and testing datasets spatially described in Valavi, R. et al. (2019) <doi:10.1111/2041-210X.13107>.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 3.5.0)

Imports sf (>= 1.0), sp, terra (>= 1.6-41), ggplot2 (>= 3.3.6), cowplot, automap (>= 1.0-16), Rcpp (>= 1.0.2)

Suggests shiny (>= 1.7), tmap (>= 2.0), biomod2, gstat, methods, knitr, rmarkdown, testthat (>= 3.0.0)

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/testthat/edition 3

LinkingTo Rcpp

NeedsCompilation yes

Author Roozbeh Valavi [aut, cre] (ORCID: <https://orcid.org/0000-0003-2495-5277>),
Jane Elith [aut],
José Lahoz-Monfort [aut],
Ian Flint [aut],
Gurutzeta Guillera-Arroita [aut]

Repository CRAN
Date/Publication 2025-08-21 13:40:07 UTC

Contents

blockCV . . . 2
buffering . . . 3
cv_block_size . . . 4
cv_buffer . . . 5
cv_cluster . . . 7
cv_nndm . . . 9
cv_plot . . . 12
cv_similarity . . . 13
cv_spatial . . . 15
cv_spatial_autocor . . . 19
envBlock . . . 22
foldExplorer . . . 23
rangeExplorer . . . 23
spatialAutoRange . . . 24
spatialBlock . . . 25

Index **28**

`blockCV` *blockCV: Spatial and Environmental Blocking for K-Fold and LOO Cross-Validation*

Description

Simple random selection of training and testing folds in the structured environment leads to an underestimation of error in the evaluation of spatial predictions and may result in inappropriate model selection (Telford and Birks, 2009; Roberts et al., 2017). The use of spatial and environmental blocks to separate training and testing sets has been suggested as a good strategy for realistic error estimation in datasets with dependence structures, and more generally as a robust method for estimating the predictive performance of models used to predict mapped distributions (Roberts et al., 2017). The package `blockCV` offers a range of functions for generating train and test folds for **k-fold** and **leave-one-out (LOO)** cross-validation (CV). It allows for separation of data spatially and environmentally, with various options for block construction. Additionally, it includes a function for assessing the level of spatial autocorrelation in response or raster covariates, to aid in selecting an appropriate distance band for data separation. The `blockCV` package is suitable for the evaluation of a variety of spatial modelling applications, including classification of remote sensing imagery, soil mapping, and species distribution modelling (SDM). It also provides support for different SDM scenarios, including presence-absence and presence-background species data, rare and common species, and raster data for predictor variables.

Author(s)

Roozbeh Valavi, Jane Elith, José Lahoz-Monfort, Ian Flint, and Gurutzeta Guillera-Arroita

References

Valavi, R., Elith, J., Lahoz-Monfort, J. J., & Guillera-Arroita, G. (2019). blockCV: An R package for generating spatially or environmentally separated folds for k-fold cross-validation of species distribution models. Methods in Ecology and Evolution, 10(2), 225-232. doi:10.1111/2041-210X.13107.

See Also

cv_spatial, cv_cluster, cv_buffer, and cv_nndm for blocking strategies.

---

buffering *Use distance (buffer) around records to separate train and test folds*

Description

This function is deprecated and will be removed in future updates! Please use cv_buffer instead!

Usage

```
buffering(
  speciesData,
  species = NULL,
  theRange,
  spDataType = "PA",
  addBG = TRUE,
  progress = TRUE
)
```

Arguments

speciesData	A simple features (sf) or SpatialPoints object containing species data (response variable).
species	Character. Indicating the name of the field in which species data (binary response i.e. 0 and 1) is stored. If speceis = NULL the presence and absence data (response variable) will be treated the same and only training and testing records will be counted. This can be used for multi-class responses such as land cover classes for remote sensing image classification, but it is not necessary. *Do not use this argument when the response variable is continuous or count data*.
theRange	Numeric value of the specified range by which the training and testing datasets are separated. This distance should be in ***metres*** no matter what the coordinate system is. The range can be explored by spatialAutoRange.
spDataType	Character input indicating the type of species data. It can take two values, **PA** for *presence-absence* data and **PB** for *presence-background* data, when species argument is not NULL. See the details section for more information on these two approaches.
addBG	Logical. Add background points to the test set when spDataType = "PB".
progress	Logical. If TRUE a progress bar will be shown.

See Also

cv_buffer

---

cv_block_size *Explore spatial block size*

Description

This function assists selection of block size. It allows the user to visualise the blocks interactively, viewing the impact of block size on number and arrangement of blocks in the landscape (and optionally on the distribution of species data in those blocks). Slide to the selected block size, and click *Apply Changes* to change the block size.

Usage

```
cv_block_size(r, x = NULL, column = NULL, min_size = NULL, max_size = NULL)
```

Arguments

r	a terra SpatRaster object (optional). If provided, its extent will be used to specify the blocks. It also supports *stars*, *raster*, or path to a raster file on disk.
x	a simple features (sf) or SpatialPoints object of spatial sample data. If r is supplied, this is only added to the plot. Otherwise, the extent of x is used for creating the blocks.
column	character (optional). Indicating the name of the column in which response variable (e.g. species data as a binary response i.e. 0s and 1s) is stored to be shown on the plot.
min_size	numeric; the minimum size of the blocks (in metres) to explore.
max_size	numeric; the maximum size of the blocks (in metres) to explore.

Value

an interactive shiny session

Examples

```
if(interactive()){
library(blockCV)

# import presence-absence species data
points <- read.csv(system.file("extdata/", "species.csv", package = "blockCV"))
pa_data <- sf::st_as_sf(points, coords = c("x", "y"), crs = 7845)

# manually choose the size of spatial blocks
cv_block_size(x = pa_data,
              column = "occ",
              min_size = 2e5,
```

```
                max_size = 9e5)

}
```

---

cv_buffer *Use buffer around records to separate train and test folds (a.k.a. buffered/spatial leave-one-out)*

Description

This function generates spatially separated train and test folds by considering buffers of the specified distance (`size` parameter) around each observation point. This approach is a form of *leave-one-out* cross-validation. Each fold is generated by excluding nearby observations around each testing point within the specified distance (ideally the range of spatial autocorrelation, see `cv_spatial_autocor`). In this method, the testing set never directly abuts a training sample (e.g. presence or absence; 0s and 1s). For more information see the details section.

Usage

```
cv_buffer(
  x,
  column = NULL,
  size,
  presence_bg = FALSE,
  add_bg = FALSE,
  progress = TRUE,
  report = TRUE
)
```

Arguments

`x`	a simple features (sf) or SpatialPoints object of spatial sample data (e.g., species data or ground truth sample for image classification).
`column`	character; indicating the name of the column in which response variable (e.g. species data as a binary response i.e. 0s and 1s) is stored. This is required when `presence_bg = TRUE`, otherwise optional.
`size`	numeric value of the specified range by which training/testing data are separated. This distance should be in **metres**. The range could be explored by `cv_spatial_autocor`.
`presence_bg`	logical; whether to treat data as species presence-background data. For all other data types (presence-absence, continuous, count or multi-class responses), this option should be `FALSE`.
`add_bg`	logical; add background points to the test set when `presence_bg = TRUE`. We do not recommend this according to Radosavljevic & Anderson (2014). Keep it `FALSE`, unless you mean to add the background pints to testing points.

progress	logical; whether to shows a progress bar.
report	logical; whether to generate print summary of records in each fold; for very big datasets, set to FALSE for faster calculation.

Details

When working with presence-background (presence and pseudo-absence) species distribution data (should be specified by presence_bg = TRUE argument), only presence records are used for specifying the folds (recommended). Consider a target presence point. The buffer is defined around this target point, using the specified range (size). By default, the testing fold comprises only the target presence point (all background points within the buffer are also added when add_bg = TRUE). Any non-target presence points inside the buffer are excluded. All points (presence and background) outside of buffer are used for the training set. The methods cycles through all the *presence* data, so the number of folds is equal to the number of presence points in the dataset.

For presence-absence data (and all other types of data), folds are created based on all records, both presences and absences. As above, a target observation (presence or absence) forms a test point, all presence and absence points other than the target point within the buffer are ignored, and the training set comprises all presences and absences outside the buffer. Apart from the folds, the number of *training-presence*, *training-absence*, *testing-presence* and *testing-absence* records is stored and returned in the records table. If column = NULL and presence_bg = FALSE, the procedure is like presence-absence data. All other data types (continuous, count or multi-class responses) should be done by presence_bg = FALSE.

Value

An object of class S3. A list of objects including:

- folds_list - a list containing the folds. Each fold has two vectors with the training (first) and testing (second) indices
- k - number of the folds
- size - the defined range of spatial autocorrelation)
- column - the name of the column if provided
- presence_bg - whether this was treated as presence-background data
- records - a table with the number of points in each category of training and testing

References

Radosavljevic, A., & Anderson, R. P. (2014). Making better Maxent models of species distributions: Complexity, overfitting and evaluation. Journal of Biogeography, 41, 629–643. https://doi.org/10.1111/jbi.12227

See Also

cv_nndm, cv_spatial, and cv_spatial_autocor

Examples

```
library(blockCV)

# import presence-absence species data
points <- read.csv(system.file("extdata/", "species.csv", package = "blockCV"))
# make an sf object from data.frame
pa_data <- sf::st_as_sf(points, coords = c("x", "y"), crs = 7845)

bloo <- cv_buffer(x = pa_data,
                  column = "occ",
                  size = 350000, # size in metres no matter the CRS
                  presence_bg = FALSE)
```

`cv_cluster` *Use environmental or spatial clustering to separate train and test folds*

Description

This function uses clustering methods to specify sets of similar environmental conditions based on the input covariates, or cluster of spatial coordinates of the sample data. Sample data (i.e. species data) corresponding to any of these groups or clusters are assigned to a fold. Clustering is done using `kmeans` for both approaches. The only requirement is `x` that leads to a clustering of the confidantes of sample data. Otherwise, by providing `r`, environmental clustering is done.

Usage

```
cv_cluster(
  x,
  column = NULL,
  r = NULL,
  k = 5L,
  scale = TRUE,
  raster_cluster = FALSE,
  num_sample = 10000L,
  biomod2 = TRUE,
  report = TRUE,
  ...
)
```

Arguments

`x`	a simple features (sf) or SpatialPoints object of spatial sample data (e.g., species data or ground truth sample for image classification).
`column`	character (optional). Indicating the name of the column in which response variable (e.g. species data as a binary response i.e. 0s and 1s) is stored. This is only used to see whether all the folds contain all the classes in the final report.

`r`	a terra SpatRaster object of covariates to identify environmental groups. If provided, clustering will be done in environmental space rather than spatial coordinates of sample points.
`k`	integer value. The number of desired folds for cross-validation. The default is `k = 5`.
`scale`	logical; whether to scale the input rasters (recommended) for clustering.
`raster_cluster`	logical; if `TRUE`, the clustering is done over the entire raster layer, otherwise it will be over the extracted raster values of the sample points. See details for more information.
`num_sample`	integer; the number of samples from raster layers to build the clusters (when `raster_cluster = FALSE`).
`biomod2`	logical. Creates a matrix of folds that can be directly used in the **biomod2** package as a *CV.user.table* for cross-validation.
`report`	logical; whether to print the report of the records per fold.
`...`	additional arguments for `stats::kmeans` function, e.g. `algorithm = "MacQueen"`.

Details

As k-means algorithms use Euclidean distance to estimate clusters, the input raster covariates should be quantitative variables. Since variables with wider ranges of values might dominate the clusters and bias the environmental clustering (Hastie et al., 2009), all the input rasters are first scaled and centred (`scale = TRUE`) within the function.

If `raster_cluster = TRUE`, the clustering is done in the raster space. In this approach the clusters will be consistent throughout the region and different sample datasets in the same region (for comparison). However, this may result in a cluster(s) that covers none of the species records (the spatial location of response samples), especially when species data is not dispersed throughout the region or the number of clusters (k or folds) is high. In this case, the number of folds is less than specified `k`. If `raster_cluster = FALSE`, the clustering will be done in species points and the number of the folds will be the same as `k`.

Note that the input raster layer should cover all the species points, otherwise an error will rise. The records with no raster value should be deleted prior to the analysis or another raster layer must be provided.

Value

An object of class S3. A list of objects including:

- folds_list - a list containing the folds. Each fold has two vectors with the training (first) and testing (second) indices
- folds_ids - a vector of values indicating the number of the fold for each observation (each number corresponds to the same point in x)
- biomod_table - a matrix with the folds to be used in **biomod2** package
- k - number of the folds
- column - the name of the column if provided
- type - indicates whether spatial or environmental clustering was done.
- records - a table with the number of points in each category of training and testing

References

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed., Vol. 1).

See Also

cv_buffer and cv_spatial

Examples

```
library(blockCV)

# import presence-absence species data
points <- read.csv(system.file("extdata/", "species.csv", package = "blockCV"))
# make an sf object from data.frame
pa_data <- sf::st_as_sf(points, coords = c("x", "y"), crs = 7845)

# load raster data
path <- system.file("extdata/au/", package = "blockCV")
files <- list.files(path, full.names = TRUE)
covars <- terra::rast(files)

# spatial clustering
set.seed(6)
sc <- cv_cluster(x = pa_data,
                 column = "occ", # optional; name of the column with response
                 k = 5)

# environmental clustering
set.seed(6)
ec <- cv_cluster(r = covars, # if provided will be used for environmental clustering
                 x = pa_data,
                 column = "occ", # optional; name of the column with response
                 k = 5,
                 scale = TRUE)
```

---

cv_nndm *Use the Nearest Neighbour Distance Matching (NNDM) to separate train and test folds*

Description

A fast implementation of the Nearest Neighbour Distance Matching (NNDM) algorithm (Milà et al., 2022) in C++. Similar to cv_buffer, this is a variation of leave-one-out (LOO) cross-validation. It tries to match the nearest neighbour distance distribution function between the test and training data to the nearest neighbour distance distribution function between the target prediction and training points (Milà et al., 2022).

Usage

```
cv_nndm(
  x,
  column = NULL,
  r,
  size,
  num_sample = 10000,
  sampling = "random",
  min_train = 0.05,
  presence_bg = FALSE,
  add_bg = FALSE,
  plot = TRUE,
  report = TRUE
)
```

Arguments

x	a simple features (sf) or SpatialPoints object of spatial sample data (e.g., species data or ground truth sample for image classification).
column	character; indicating the name of the column in which response variable (e.g. species data as a binary response i.e. 0s and 1s) is stored. This is required when presence_bg = TRUE, otherwise optional.
r	a terra SpatRaster object of a predictor variable. This defines the area that model is going to predict.
size	numeric value of the range of spatial autocorrelation (the phi parameter). This distance should be in **metres**. The range could be explored by cv_spatial_autocor.
num_sample	integer; the number of sample points from predictor (r) to be used for calculating the G function of prediction points.
sampling	either "random" or "regular" for sampling prediction points. When sampling = "regular", the actual number of samples might be less than num_sample for non-rectangular rasters (points falling on no-value areas are removed).
min_train	numeric; between 0 and 1. A constraint on the minimum proportion of train points in each fold.
presence_bg	logical; whether to treat data as species presence-background data. For all other data types (presence-absence, continuous, count or multi-class responses), this option should be FALSE.
add_bg	logical; add background points to the test set when presence_bg = TRUE. We do not recommend this according to Radosavljevic & Anderson (2014). Keep it FALSE, unless you mean to add the background pints to testing points.
plot	logical; whether to plot the G functions.
report	logical; whether to generate print summary of records in each fold; for very big datasets, set to FALSE for slightly faster calculation.

Details

When working with presence-background (presence and pseudo-absence) species distribution data (should be specified by `presence_bg = TRUE` argument), only presence records are used for specifying the folds (recommended). The testing fold comprises only the target *presence* point (optionally, all background points within the distance are also included when `add_bg = TRUE`; this is the distance that matches the nearest neighbour distance distribution function of training-testing presences and training-presences and prediction points; often lower than `size`). Any non-target presence points inside the distance are excluded. All points (presence and background) outside of distance are used for the training set. The methods cycles through all the presence data, so the number of folds is equal to the number of presence points in the dataset.

For all other types of data (including presence-absence, count, continuous, and multi-class) set `presence_bg = FALE`, and the function behaves similar to the methods explained by Milà and colleagues (2022).

Value

An object of class S3. A list of objects including:

- folds_list - a list containing the folds. Each fold has two vectors with the training (first) and testing (second) indices
- k - number of the folds
- size - the distance band to separated trainig and testing folds)
- column - the name of the column if provided
- presence_bg - whether this was treated as presence-background data
- records - a table with the number of points in each category of training and testing

References

C. Milà, J. Mateu, E. Pebesma, and H. Meyer, Nearest Neighbour Distance Matching Leave-One-Out Cross-Validation for map validation, Methods in Ecology and Evolution (2022).

See Also

`cv_buffer` and `cv_spatial_autocor`

Examples

```
library(blockCV)

# import presence-absence species data
points <- read.csv(system.file("extdata/", "species.csv", package = "blockCV"))
# make an sf object from data.frame
pa_data <- sf::st_as_sf(points, coords = c("x", "y"), crs = 7845)

# load raster data
path <- system.file("extdata/au/bio_5.tif", package = "blockCV")
covar <- terra::rast(path)
```

```
nndm <- cv_nndm(x = pa_data,
                column = "occ", # optional
                r = covar,
                size = 350000, # size in metres no matter the CRS
                num_sample = 10000,
                sampling = "regular",
                min_train = 0.1)
```

---

cv_plot *Visualising folds created by blockCV in ggplot*

Description

This function visualises the folds create by blockCV. It also accepts a raster layer to be used as background in the output plot.

Usage

```
cv_plot(
  cv,
  x,
  r = NULL,
  nrow = NULL,
  ncol = NULL,
  num_plots = 1:10,
  max_pixels = 3e+05,
  remove_na = TRUE,
  raster_colors = gray.colors(10, alpha = 1),
  points_colors = c("#E69F00", "#56B4E9"),
  points_alpha = 0.7,
  label_size = 4
)
```

Arguments

cv	a blockCV cv_* object; a `cv_spatial`, `cv_cluster`, `cv_buffer` or `cv_nndm`
x	a simple features (sf) or SpatialPoints object of the spatial sample data used for creating the `cv` object. This could be empty when `cv` is a `cv_spatial` object.
r	a terra SpatRaster object (optional). If provided, it will be used as background of the plots. It also supports *stars*, *raster*, or path to a raster file on disk.
nrow	integer; number of rows for facet plot
ncol	integer; number of columns for facet plot
num_plots	a vector of indices of folds; by default the first 10 are shown (if available). You can choose any of the folds to be shown e.g. `1:3` or `c(2, 7, 16, 22)`

max_pixels	integer; maximum number of pixels used for plotting r
remove_na	logical; whether to remove excluded points in cv_buffer from the plot
raster_colors	character; a character vector of colours for raster background e.g. terrain.colors(20)
points_colors	character; two colours to be used for train and test points
points_alpha	numeric; the opacity of points
label_size	integer; size of fold labels when a cv_spatial object is used.

Value

a ggplot object

Examples

```
library(blockCV)

# import presence-absence species data
points <- read.csv(system.file("extdata/", "species.csv", package = "blockCV"))
pa_data <- sf::st_as_sf(points, coords = c("x", "y"), crs = 7845)

# spatial clustering
sc <- cv_cluster(x = pa_data, k = 5)

# now plot the create folds
cv_plot(cv = sc,
        x = pa_data, # sample points
        nrow = 2,
        points_alpha = 0.5)
```

---

cv_similarity *Compute similarity measures to evaluate possible extrapolation in testing folds*

Description

This function evaluates environmental similarity between training and testing folds, helping to detect potential extrapolation in the testing data. It supports three similarity measures: Multivariate Environmental Similarity Surface (MESS), Manhattan distance (L1), and Euclidean distance (L2).

Usage

```
cv_similarity(
  cv,
  x,
  r,
  num_plot = seq_along(cv$folds_list),
```

```
  method = "MESS",
  num_sample = 10000L,
  jitter_width = 0.1,
  points_size = 2,
  points_alpha = 0.7,
  points_colors = NULL,
  progress = TRUE
)
```

Arguments

cv	a blockCV cv_* object; a cv_spatial, cv_cluster, cv_buffer or cv_nndm
x	a simple features (sf) or SpatialPoints object of the spatial sample data used for creating the cv object.
r	a terra SpatRaster object of environmental predictor that are going to be used for modelling. This is used to calculate similarity between the training and testing points.
num_plot	a vector of indices of folds.
method	the similarity method including: MESS, L1 and L2. Read the details section.
num_sample	number of random samples from raster to calculate similarity distances (only for L1 and L2).
jitter_width	numeric; the width of jitter points.
points_size	numeric; the size of points.
points_alpha	numeric; the opacity of points
points_colors	character; a character vector of colours for points
progress	logical; whether to shows a progress bar for random fold selection.

Details

The MESS is calculated as described in Elith et al. (2010). MESS represents how similar a point in a testing fold is to a training fold (as a reference set of points), with respect to a set of predictor variables in r. The negative values are the sites where at least one variable has a value that is outside the range of environments over the reference set, so these are novel environments.

When using the L1 (Manhattan) or L2 (Euclidean) distance options (experimental), the function performs the following steps for each test sample:

- 1. Calculates the minimum distance between each test sample and all training samples in the same fold using the selected metric (L1 or L2).
- 2. Calculates a baseline distance: the average of the minimum distances between a set of random background samples (defined by num_sample) from the raster and all training/test samples combined.
- 3. Computes a similarity score by subtracting the test sample's minimum distance from the baseline average. A higher score indicates the test sample is more similar to the training data, while lower or negative scores indicate novelty.

This provides a simple, distance-based novelty metric, useful for assessing extrapolation or dissimilarity in prediction scenarios. Note that this approach is experimental.

Value

a ggplot object

References

Elith, J., Kearney, M., & Phillips, S. (2010). The art of modelling range-shifting species: The art of modelling range-shifting species. Methods in Ecology and Evolution, 1(4), 330–342.

See Also

cv_spatial, cv_cluster, cv_buffer, and cv_nndm

Examples

```
library(blockCV)

# import presence-absence species data
points <- read.csv(system.file("extdata/", "species.csv", package = "blockCV"))
# make an sf object from data.frame
pa_data <- sf::st_as_sf(points, coords = c("x", "y"), crs = 7845)

# load raster data
path <- system.file("extdata/au/", package = "blockCV")
files <- list.files(path, full.names = TRUE)
covars <- terra::rast(files)

# hexagonal spatial blocking by specified size and random assignment
sb <- cv_spatial(x = pa_data,
                 column = "occ",
                 size = 450000,
                 k = 5,
                 iteration = 1)

# compute extrapolation
cv_similarity(cv = sb, r = covars, x = pa_data)
```

---

cv_spatial *Use spatial blocks to separate train and test folds*

Description

This function creates spatially separated folds based on a distance to number of row and/or column. It assigns blocks to the training and testing folds **randomly**, **systematically** or in a **checkerboard pattern**. The distance (size) should be in **metres**, regardless of the unit of the reference system of the input data (for more information see the details section). By default, the function creates blocks according to the extent and shape of the spatial sample data (x e.g. the species occurrence), Alternatively, blocks can be created based on r assuming that the user has considered the landscape

for the given species and case study. Blocks can also be offset so the origin is not at the outer corner of the rasters. Instead of providing a distance, the blocks can also be created by specifying a number of rows and/or columns and divide the study area into vertical or horizontal bins, as presented in Wenger & Olden (2012) and Bahn & McGill (2012). Finally, the blocks can be specified by a user-defined spatial polygon layer.

Usage

```
cv_spatial(
  x,
  column = NULL,
  r = NULL,
  k = 5L,
  hexagon = TRUE,
  flat_top = FALSE,
  size = NULL,
  rows_cols = c(10, 10),
  selection = "random",
  iteration = 100L,
  user_blocks = NULL,
  folds_column = NULL,
  deg_to_metre = 111325,
  biomod2 = TRUE,
  offset = c(0, 0),
  extend = 0,
  seed = NULL,
  progress = TRUE,
  report = TRUE,
  plot = TRUE,
  ...
)
```

Arguments

`x`	a simple features (sf) or SpatialPoints object of spatial sample data (e.g., species data or ground truth sample for image classification).
`column`	character (optional). Indicating the name of the column in which response variable (e.g. species data as a binary response i.e. 0s and 1s) is stored to find balanced records in cross-validation folds. If `column = NULL` the response variable classes will be treated the same and only training and testing records will be counted. This is used for binary (e.g. presence-absence/background) or multi-class responses (e.g. land cover classes for remote sensing image classification), and *you can ignore it when the response variable is continuous or count data*.
`r`	a terra SpatRaster object (optional). If provided, its extent will be used to specify the blocks. It also supports *stars*, *raster*, or path to a raster file on disk.
`k`	integer value. The number of desired folds for cross-validation. The default is `k = 5`.

hexagon	logical. Creates hexagonal (default) spatial blocks. If FALSE, square blocks is created.
flat_top	logical. Creating hexagonal blocks with topped flat.
size	numeric value of the specified range by which blocks are created and training/testing data are separated. This distance should be in **metres**. The range could be explored by cv_spatial_autocor and cv_block_size functions.
rows_cols	integer vector. Two integers to define the blocks based on row and column e.g. c(10, 10) or c(5, 1). Hexagonal blocks uses only the first one. This option is ignored when size is provided.
selection	type of assignment of blocks into folds. Can be **random** (default), **systematic**, **checkerboard**, or **predefined**. The checkerboard does not work with hexagonal and user-defined spatial blocks. If the selection = 'predefined', user-defined blocks and folds_column must be supplied.
iteration	integer value. The number of attempts to create folds with balanced records. Only works when selection = "random".
user_blocks	an sf or SpatialPolygons object to be used as the blocks (optional). This can be a user defined polygon and it must cover all the species (response) points. If selection = 'predefined', this argument and **folds_column** must be supplied.
folds_column	character. Indicating the name of the column (in user_blocks) in which the associated folds are stored. This argument is necessary if you choose the 'predefined' selection.
deg_to_metre	integer. The conversion rate of metres to degree. See the details section for more information.
biomod2	logical. Creates a matrix of folds that can be directly used in the **biomod2** package as a *CV.user.table* for cross-validation.
offset	two number between 0 and 1 to shift blocks by that proportion of block size. This option only works when size is provided.
extend	numeric; This parameter specifies the percentage by which the map's extent is expanded to increase the size of the square spatial blocks, ensuring that all points fall within a block. The value should be a numeric between 0 and 5.
seed	integer; a random seed for reproducibility (although an external seed should also work).
progress	logical; whether to shows a progress bar for random fold selection.
report	logical; whether to print the report of the records per fold.
plot	logical; whether to plot the final blocks with fold numbers in ggplot. You can re-create this with cv_plot.
...	additional option for cv_plot.

Details

To maintain consistency, all functions in this package use **meters** as their unit of measurement. However, when the input map has a geographic coordinate system (in decimal degrees), the block

size is calculated by dividing the `size` parameter by `deg_to_metre` (which defaults to 111325 meters, the standard distance of one degree of latitude on the Equator). In reality, this value varies by a factor of the cosine of the latitude. So, an alternative sensible value could be `cos(mean(sf::st_bbox(x)[c(2,4)]) * pi/180) * 111325`.

The `offset` can be used to change the spatial position of the blocks. It can also be used to assess the sensitivity of analysis results to shifting in the blocking arrangements. These options are available when `size` is defined. By default the region is located in the middle of the blocks and by setting the offsets, the blocks will shift.

Roberts et. al. (2017) suggest that blocks should be substantially bigger than the range of spatial autocorrelation (in model residual) to obtain realistic error estimates, while a buffer with the size of the spatial autocorrelation range would result in a good estimation of error. This is because of the so-called edge effect (O'Sullivan & Unwin, 2014), whereby points located on the edges of the blocks of opposite sets are not separated spatially. Blocking with a buffering strategy overcomes this issue (see `cv_buffer`).

Value

An object of class S3. A list of objects including:

- folds_list - a list containing the folds. Each fold has two vectors with the training (first) and testing (second) indices
- folds_ids - a vector of values indicating the number of the fold for each observation (each number corresponds to the same point in species data)
- biomod_table - a matrix with the folds to be used in **biomod2** package
- k - number of the folds
- size - input size, if not null
- column - the name of the column if provided
- blocks - spatial polygon of the blocks
- records - a table with the number of points in each category of training and testing

References

Bahn, V., & McGill, B. J. (2012). Testing the predictive performance of distribution models. Oikos, 122(3), 321-331.

O'Sullivan, D., Unwin, D.J., (2010). Geographic Information Analysis, 2nd ed. John Wiley & Sons.

Roberts et al., (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography. 40: 913-929.

Wenger, S.J., Olden, J.D., (2012). Assessing transferability of ecological models: an underappreciated aspect of statistical validation. Methods Ecol. Evol. 3, 260-267.

See Also

`cv_buffer` and `cv_cluster`; `cv_spatial_autocor` and `cv_block_size` for selecting block size

For *CV.user.table* see `BIOMOD_Modeling` in **biomod2** package

Examples

```
library(blockCV)

# import presence-absence species data
points <- read.csv(system.file("extdata/", "species.csv", package = "blockCV"))
# make an sf object from data.frame
pa_data <- sf::st_as_sf(points, coords = c("x", "y"), crs = 7845)

# hexagonal spatial blocking by specified size and random assignment
sb1 <- cv_spatial(x = pa_data,
                  column = "occ",
                  size = 450000,
                  k = 5,
                  selection = "random",
                  iteration = 50)

# spatial blocking by row/column and systematic fold assignment
sb2 <- cv_spatial(x = pa_data,
                  column = "occ",
                  rows_cols = c(8, 10),
                  k = 5,
                  hexagon = FALSE,
                  selection = "systematic")
```

`cv_spatial_autocor` *Measure spatial autocorrelation in spatial response data or predictor raster files*

Description

This function provides a quantitative basis for choosing block size. The spatial autocorrelation in either the spatial sample points or all continuous predictor variables available as raster layers is assessed and reported. The response (as defined be `column`) in spatial sample points can be binary such as species distribution data, or continuous response like soil organic carbon. The function estimates spatial autocorrelation *ranges* of all input raster layers or the response data. This is the range over which observations are independent and is determined by constructing the empirical variogram, a fundamental geostatistical tool for measuring spatial autocorrelation. The empirical variogram models the structure of spatial autocorrelation by measuring variability between all possible pairs of points (O'Sullivan and Unwin, 2010). Results are plotted. See the details section for further information.

Usage

```
cv_spatial_autocor(
  r,
  x,
```

```
  column = NULL,
  num_sample = 5000L,
  deg_to_metre = 111325,
  plot = TRUE,
  progress = TRUE,
  ...
)
```

Arguments

r	a terra SpatRaster object. If provided (and x is missing), it will be used for to calculate range.
x	a simple features (sf) or SpatialPoints object of spatial sample data (e.g., species binary or continuous date).
column	character; indicating the name of the column in which response variable (e.g. species data as a binary response i.e. 0s and 1s) is stored for calculating spatial autocorrelation range. This supports multiple column names.
num_sample	integer; the number of sample points of each raster layer to fit variogram models. It is 5000 by default, however it can be increased by user to represent their region well (relevant to the extent and resolution of rasters).
deg_to_metre	integer. The conversion rate of degrees to metres.
plot	logical; whether to plot the results.
progress	logical; whether to shows a progress bar.
...	additional option for cv_plot

Details

The input raster layers should be continuous for computing the variograms and estimating the range of spatial autocorrelation. The input rasters should also have a specified coordinate reference system. However, if the reference system is not specified, the function attempts to guess it based on the extent of the map. It assumes an un-projected reference system for layers with extent lying between -180 and 180.

Variograms are calculated based on the distances between pairs of points, so un-projected rasters (in degrees) will not give an accurate result (especially over large latitudinal extents). For un-projected rasters, *the great circle distance* (rather than Euclidean distance) is used to calculate the spatial distances between pairs of points. To enable more accurate estimate, it is recommended to transform un-projected maps (geographic coordinate system / latitude-longitude) to a projected metric reference system (e.g. UTM or Lambert) where it is possible. See autofitVariogram from **automap** and variogram from **gstat** packages for further information.

Value

An object of class S3. A list object including:

- range - the suggested range (i.e. size), which is the median of all calculated ranges in case of 'r'.
- range_table - a table of input covariates names and their autocorrelation range

- plots - the output plot (the plot is shown by default)
- num_sample - number sample of 'r' used for analysis
- variograms - fitted variograms for all layers

References

O'Sullivan, D., Unwin, D.J., (2010). Geographic Information Analysis, 2nd ed. John Wiley & Sons.

Roberts et al., (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography. 40: 913-929.

See Also

cv_block_size

Examples

```
library(blockCV)

# import presence-absence species data
points <- read.csv(system.file("extdata/", "species.csv", package = "blockCV"))
# make an sf object from data.frame
pa_data <- sf::st_as_sf(points, coords = c("x", "y"), crs = 7845)

# load raster data
path <- system.file("extdata/au/", package = "blockCV")
files <- list.files(path, full.names = TRUE)
covars <- terra::rast(files)

# spatial autocorrelation of a binary/continuous response
sac1 <- cv_spatial_autocor(x = pa_data,
                           column = "occ", # binary or continuous data
                           plot = TRUE)


# spatial autocorrelation of continuous raster files
sac2 <- cv_spatial_autocor(r = covars,
                           num_sample = 5000,
                           plot = TRUE)

# show the result
summary(sac2)
```

envBlock *Use environmental clustering to separate train and test folds*

Description

This function is deprecated and will be removed in future updates! Please use `cv_cluster` instead!

Usage

```
envBlock(
  rasterLayer,
  speciesData,
  species = NULL,
  k = 5,
  standardization = "normal",
  rasterBlock = TRUE,
  sampleNumber = 10000,
  biomod2Format = TRUE,
  numLimit = 0,
  verbose = TRUE
)
```

Arguments

`rasterLayer`	A raster object of covariates to identify environmental groups.
`speciesData`	A simple features (sf) or SpatialPoints object containing species data (response variable).
`species`	Character. Indicating the name of the field in which species data (binary response i.e. 0 and 1) is stored. If `speceis = NULL` the presence and absence data (response variable) will be treated the same and only training and testing records will be counted. This can be used for multi-class responses such as land cover classes for remote sensing image classification, but it is not necessary. *Do not use this argument when the response variable is continuous or count data*.
`k`	Integer value. The number of desired folds for cross-validation. The default is `k = 5`.
`standardization`	Standardize input raster layers. Three possible inputs are "normal" (the default), "standard" and "none". See details for more information.
`rasterBlock`	Logical. If TRUE, the clustering is done in the raster layer rather than species data. See details for more information.
`sampleNumber`	Integer. The number of samples from raster layers to build the clusters.
`biomod2Format`	Logical. Creates a matrix of folds that can be directly used in the **biomod2** package as a *DataSplitTable* for cross-validation.
`numLimit`	Integer value. The minimum number of points in each category of data (*train_0*, *train_1*, *test_0* and *test_1*). Shows a message if the number of points in any of the folds happens to be less than this number.

verbose	Logical. To print the report of the recods per fold.

See Also

cv_cluster

---

foldExplorer *Explore the generated folds*

Description

This function is deprecated! Please use cv_plot function for plotting the folds.

Usage

```
foldExplorer(blocks, rasterLayer, speciesData)
```

Arguments

blocks	deprecated!
rasterLayer	deprecated!
speciesData	deprecated!

---

rangeExplorer *Explore spatial block size*

Description

This function is deprecated and will be removed in future updates! Please use cv_block_size instead!

Usage

```
rangeExplorer(
  rasterLayer,
  speciesData = NULL,
  species = NULL,
  rangeTable = NULL,
  minRange = NULL,
  maxRange = NULL
)
```

Arguments

rasterLayer	raster layer for make plot
speciesData	a simple features (sf) or SpatialPoints object containing species data (response variable). If provided, the species data will be shown on the map.
species	character value indicating the name of the field in which the species data (response variable e.g. 0s and 1s) are stored. If provided, species presence and absence data will be shown in different colours.
rangeTable	deprecated option!
minRange	a numeric value to set the minimum possible range for creating spatial blocks. It is used to limit the searching domain of spatial block size.
maxRange	a numeric value to set the maximum possible range for creating spatial blocks. It is used to limit the searching domain of spatial block size.

See Also

cv_block_size

---

spatialAutoRange *Measure spatial autocorrelation in the predictor raster files*

Description

This function is deprecated and will be removed in future updates! Please use cv_spatial_autocor instead!

Usage

```
spatialAutoRange(
  rasterLayer,
  sampleNumber = 5000L,
  border = NULL,
  speciesData = NULL,
  doParallel = NULL,
  nCores = NULL,
  showPlots = TRUE,
  degMetre = 111325,
  maxpixels = 1e+05,
  plotVariograms = FALSE,
  progress = TRUE
)
```

Arguments

rasterLayer	A raster object of covariates to find spatial autocorrelation range.
sampleNumber	Integer. The number of sample points of each raster layer to fit variogram models. It is 5000 by default, however it can be increased by user to represent their region well (relevant to the extent and resolution of rasters).
border	deprecated option!
speciesData	A spatial or sf object (optional). If provided, the sampleNumber is ignored and variograms are created based on species locations. This option is not recommended if the species data is not evenly distributed across the whole study area and/or the number of records is low.
doParallel	deprecated option!
nCores	deprecated option!
showPlots	Logical. Show final plot of spatial blocks and autocorrelation ranges.
degMetre	Numeric. The conversion rate of metres to degree. This is for constructing spatial blocks for visualisation. When the input map is in geographic coordinate system (decimal degrees), the block size is calculated based on deviding the calculated *range* by this value to convert to the input map's unit (by default 111325; the standard distance of a degree in metres, on the Equator).
maxpixels	Number of random pixels to select the blocks over the study area.
plotVariograms	deprecated option!
progress	Logical. Shows progress bar. It works only when doParallel = FALSE.

See Also

cv_spatial_autocor

---

spatialBlock *Use spatial blocks to separate train and test folds*

Description

This function is deprecated and will be removed in future updates! Please use cv_spatial instead!

Usage

```
spatialBlock(
  speciesData,
  species = NULL,
  rasterLayer = NULL,
  theRange = NULL,
  rows = NULL,
  cols = NULL,
  k = 5L,
```

```
  selection = "random",
  iteration = 100L,
  blocks = NULL,
  foldsCol = NULL,
  numLimit = 0L,
  maskBySpecies = TRUE,
  degMetre = 111325,
  border = NULL,
  showBlocks = TRUE,
  biomod2Format = TRUE,
  xOffset = 0,
  yOffset = 0,
  extend = 0,
  seed = 42,
  progress = TRUE,
  verbose = TRUE
)
```

Arguments

`speciesData` A simple features (sf) or SpatialPoints object containing species data (response variable).

`species` Character (optional). Indicating the name of the column in which species data (response variable e.g. 0s and 1s) is stored. This argument is used *to make folds with evenly distributed records*. **This option only works by random fold selection and with binary or multi-class responses** e.g. species presence-absence/background or land cover classes for remote sensing image classification. If `speceis = NULL` the response classes will be treated the same and only training and testing records will be counted and balanced.

`rasterLayer` A raster object for visualisation (optional). If provided, this will be used to specify the blocks covering the area.

`theRange` Numeric value of the specified range by which blocks are created and training/testing data are separated. This distance should be in **metres**. The range could be explored by `spatialAutoRange()` and `rangeExplorer()` functions.

`rows` Integer value by which the area is divided into latitudinal bins.

`cols` Integer value by which the area is divided into longitudinal bins.

`k` Integer value. The number of desired folds for cross-validation. The default is `k = 5`.

`selection` Type of assignment of blocks into folds. Can be **random** (default), **systematic**, **checkerboard**, or **predefined**. The checkerboard does not work with user-defined spatial blocks. If the selection = 'predefined', user-defined blocks and foldsCol must be supplied.

`iteration` Integer value. The number of attempts to create folds that fulfil the set requirement for minimum number of points in each training and testing fold (for each response class e.g. *train_0*, *train_1*, *test_0* and *test_1*), as specified by `species` and `numLimit` arguments.

`blocks`	A sf or SpatialPolygons object to be used as the blocks (optional). This can be a user defined polygon and it must cover all the species (response) points. If the selection = 'predefined', this argument (and foldsCol) must be supplied.
`foldsCol`	Character. Indicating the name of the column (in user-defined blocks) in which the associated folds are stored. This argument is necessary if you choose the 'predefined' selection.
`numLimit`	deprecated option!
`maskBySpecies`	Since version 1.1, this option is always set to `TRUE`.
`degMetre`	Integer. The conversion rate of metres to degree. See the details section for more information.
`border`	deprecated option!
`showBlocks`	Logical. If TRUE the final blocks with fold numbers will be created with ggplot and plotted. A raster layer could be specified in `rasterlayer` argument to be as background.
`biomod2Format`	Logical. Creates a matrix of folds that can be directly used in the **biomod2** package as a *DataSplitTable* for cross-validation.
`xOffset`	Numeric value between **0** and **1** for shifting the blocks horizontally. The value is the proportion of block size.
`yOffset`	Numeric value between **0** and **1** for shifting the blocks vertically. The value is the proportion of block size.
`extend`	numeric; This parameter specifies the percentage by which the map's extent is expanded to increase the size of the square spatial blocks, ensuring that all points fall within a block. The value should be a numeric between 0 and 5.
`seed`	Integer. A random seed generator for reproducibility.
`progress`	Logical. If TRUE shows a progress bar when `numLimit = NULL` in random fold selection.
`verbose`	Logical. To print the report of the recods per fold.

See Also

`cv_spatial`

Index

autofitVariogram, *20*

BIOMOD_Modeling, *18*
blockCV, 2
blockCV-package (blockCV), 2
buffering, 3

cv_block_size, 4, *17, 18*, *21*, *23, 24*
cv_buffer, *3, 4*, 5, *9*, *11*, *15*, *18*
cv_cluster, *3*, 7, *15*, *18*, *22, 23*
cv_nndm, *3*, *6*, 9, *15*
cv_plot, 12, *17*, *20*, *23*
cv_similarity, 13
cv_spatial, *3*, *6*, *9*, *15*, 15, *25*, *27*
cv_spatial_autocor, *5, 6*, *10, 11*, *17, 18*, 19, *24, 25*

envBlock, 22

foldExplorer, 23

kmeans, *7*

rangeExplorer, 23

spatialAutoRange, *3*, 24
spatialBlock, 25

variogram, *20*