

Package ‘blockCV’

August 20, 2026

Type Package

Title Spatial and Environmental Blocking for Cross-Validation

Version 4.0-0

Date 2026-08-15

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

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

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

Description Creates spatially or environmentally separated, or group-preserving, training and testing folds for k-fold, leave-group-out, and leave-one-out cross-validation. Provides spatial blocking, clustering, buffering, and nearest-neighbour distance-matching methods, together with tools to visualise folds, summarise fold sizes and class balance, and assess train–test separation and environmental novelty. Also estimates spatial autocorrelation ranges in point samples and continuous raster covariates to provide an initial distance scale for designing spatial folds. Methods are described in Valavi, R. et al. (2019) <[doi:10.1111/2041-210X.13107](https://doi.org/10.1111/2041-210X.13107)>.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 3.6.0)

Imports sf (>= 1.0), terra (>= 1.6-41), ggplot2 (>= 3.3.6), cowplot, automap (>= 1.1-20), Rcpp (>= 1.0.2)

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

VignetteBuilder knitr

Config/testthat/edition 3

LinkingTo Rcpp

Config/roxygen2/version 8.0.0

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 2026-08-20 11:12:25 UTC

Contents

blockCV	2
cv_block_size	3
cv_buffer	4
cv_cluster	7
cv_distance	11
cv_group	15
cv_knndm	17
cv_nndm	22
cv_plot	25
cv_similarity	27
cv_spatial	30
cv_spatial_autocor	34
cv_summary	37
Index	40

blockCV	<i>blockCV: Spatial, Environmental, and Grouped Cross-Validation</i>
---------	--

Description

Random assignment of spatially structured or grouped observations to training and testing folds can underestimate prediction error and lead to inappropriate model selection (see Valavi et al., 2019, and references therein). The blockCV package creates training and testing folds for **k-fold**, **leave-group-out**, and **leave-one-out (LOO)** cross-validation. Strategies include spatial blocks, spatial or environmental clustering, existing grouping factors, spatial buffers, and nearest neighbour distance matching. The package also provides tools to visualise and diagnose fold designs, compare train-test separation with the prediction domain, assess environmental novelty, and investigate spatial autocorrelation and candidate block sizes. It supports spatial modelling applications such as remote-sensing classification, soil mapping, and species distribution modelling, including presence-absence and presence-background data.

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

`vignette("tutorial_1", package = "blockCV")` for examples of all fold-construction strategies, and `vignette("tutorial_2", package = "blockCV")` for fold assessment and design.

cv_block_size	<i>Explore spatial block size</i>
---------------	-----------------------------------

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

<code>r</code>	a terra <code>SpatRaster</code> object (optional). If provided, its extent will be used to specify the blocks. It also supports <i>stars</i> , <i>raster</i> , or path to a raster file on disk.
<code>x</code>	a simple features (<code>sf</code>) or <code>SpatialPoints</code> object of spatial sample data. If <code>r</code> is supplied, this is only added to the plot. Otherwise, the extent of <code>x</code> is used for creating the blocks.
<code>column</code>	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.
<code>min_size</code>	numeric; the minimum size of the blocks (in metres) to explore.
<code>max_size</code>	numeric; the maximum size of the blocks (in metres) to explore.

Value

an interactive shiny session

See Also

[cv_spatial](#) and [cv_spatial_autocor](#); [cv_plot](#), [cv_distance](#) and [cv_similarity](#) to visualise and evaluate folds

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 in model residuals, 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,
  num_bins = 4L,
  report = interactive(),
  progress = interactive()
)
```

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 <code>presence_bg = TRUE</code> , 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 <code>cv_spatial_autocor</code> . For residual-based guidance, add model residuals to <code>x</code> and pass that residual column to <code>cv_spatial_autocor</code> .
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 <code>FALSE</code> .
add_bg	logical; add background points to the test set when <code>presence_bg = TRUE</code> . We do not recommend this according to Radosavljevic & Anderson (2014). Keep it <code>FALSE</code> , unless you mean to add the background points to testing points.
num_bins	integer; the number of quantile bins used to stratify a continuous numeric column. The default is 4. Set <code>num_bins = NULL</code> to disable binning and treat every unique value as a separate class (the behaviour prior to version 3.3). If quantile breaks are tied, fewer bins may be used. The raw response values are not modified; bins are only used for fold balancing and record summaries.
report	logical; whether to print summary of records in each fold. Defaults to <code>interactive()</code> .
progress	logical; whether to shows a progress bar. Defaults to <code>interactive()</code> .

Details

In *presence-background* data the response column holds 1 for recorded *presences* and 0 for *background* (or “pseudo-absence”) points – locations sampled across the study area to characterise the available conditions rather than confirmed absences. This framing is common in species distribution modelling, but the option suits any presence-only-style data (e.g. sightings, occurrences or detections without reliable absences). When working with such data (specified by `presence_bg = TRUE`), 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

Valavi, R., Elith, J., Lahoz-Monfort, J. J., & Guillera-Aroita, 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.

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](#); [cv_plot](#) to visualise, and [cv_distance](#) and [cv_similarity](#) to evaluate, the folds

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)

# presence-background data
points_pb <- read.csv(system.file("extdata/", "species_pb.csv", package = "blockCV"))
pb_data <- sf::st_as_sf(points_pb, coords = c("x", "y"), crs = 7845)

bloo_pb <- cv_buffer(x = pb_data,
                    column = "occ",
                    size = 350000,
                    presence_bg = TRUE)
```

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 coordinates of sample data. Otherwise, by providing `r`, environmental clustering is done. Environmental clustering can also be spatially restricted with `spatial_weight`, and fold sizes or classes can be balanced with `balance = TRUE`; see Details.

Usage

```
cv_cluster(
  x,
  column = NULL,
  r = NULL,
  k = 5L,
  scale = TRUE,
  raster_cluster = FALSE,
  num_sample = 10000L,
  spatial_weight = 0,
  balance = FALSE,
  presence_bg = FALSE,
  k_multiplier = 5L,
  iteration = 100L,
  seed = NULL,
  biomod2 = TRUE,
  num_bins = 4L,
  report = interactive(),
  progress = interactive(),
  ...
)
```

Arguments

<code>x</code>	a simple features (sf) or SpatialPoints object of spatial sample data (e.g., species data or ground truth sample for image classification).
<code>column</code>	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. It is used to report whether all the folds contain all the classes and, when <code>balance = TRUE</code> , to balance those classes across the folds. Continuous numeric responses are binned into quantiles using <code>num_bins</code> before records are counted.

<code>r</code>	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. Only numeric (quantitative) covariates are supported; categorical (factor) layers are rejected because k-means relies on Euclidean distance.
<code>k</code>	integer value. The number of desired folds for cross-validation. The default is <code>k = 5</code> .
<code>scale</code>	logical; whether to scale the input rasters (recommended) for clustering.
<code>raster_cluster</code>	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.
<code>num_sample</code>	integer; the number of samples from raster layers to build the clusters (when <code>raster_cluster = FALSE</code>).
<code>spatial_weight</code>	numeric in <code>[0, 1]</code> ; only used (and only validated) for environmental clustering (when <code>r</code> is provided). It adds a soft spatial compactness pressure by blending the geographic coordinates into the environmental clustering. <code>0</code> (the default) does pure environmental clustering; larger values (e.g. <code>0.3–0.5</code>) give environmentally coherent folds that are also geographically separated; <code>1</code> clusters on the coordinates alone. See ‘Details’.
<code>balance</code>	logical. If TRUE, the sample points are first grouped into <code>k * k_multiplier</code> clusters (see <code>k_multiplier</code>) and these small clusters are then assigned to <code>k</code> folds over iteration random attempts to balance the training/testing records (or the classes/bins of column when it is provided). If FALSE (default), the points are clustered directly into <code>k</code> folds, which keeps the folds compact but does not control their size or class balance.
<code>presence_bg</code>	logical; whether to treat column as species presence-background data (0s for background points and 1s for presences; see ‘Details’). When TRUE (and <code>balance = TRUE</code>), the balancing search equalises only the presence (1s) records across folds so the many background points cannot dominate the objective; the background points are still clustered spatially but ignored when scoring the balance. Requires a binary numeric column. The default is FALSE.
<code>k_multiplier</code>	integer. The multiplier controlling how many clusters are created before they are merged into folds (i.e. <code>k * k_multiplier</code>). Only used when <code>balance = TRUE</code> . Larger values give more balanced folds at the cost of less compact folds; smaller values keep the folds compact but less balanced. The default is 5.
<code>iteration</code>	integer value. The number of random attempts to assign the clusters to folds when <code>balance = TRUE</code> .
<code>seed</code>	integer; a random seed for reproducibility of the balancing search.
<code>biomod2</code>	logical. Creates a matrix of folds that can be directly used in the biomod2 package as a <i>CV.user.table</i> for cross-validation.
<code>num_bins</code>	integer; the number of quantile bins used to stratify a continuous numeric column. The default is 4. Set <code>num_bins = NULL</code> to disable binning and treat every unique value as a separate class (the behaviour prior to version 3.3). If quantile breaks are tied, fewer bins may be used. The raw response values are not modified; bins are only used for fold balancing and record summaries.

report	logical; whether to print the report of the records per fold. Defaults to <code>interactive()</code> .
progress	logical; whether to shows a progress bar for random fold selection. Defaults to <code>interactive()</code> .
...	additional arguments for <code>stats::kmeans</code> function, e.g. <code>algorithm = "MacQueen"</code> .

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.

By default (`balance = FALSE`) the points are clustered directly into `k` groups, so the folds are compact but their sizes – and the number of records of each class – can be very uneven. When `balance = TRUE`, the points are instead grouped into a larger number of clusters (`k * k_multiplier`) that are then assigned to `k` folds over iteration random attempts, keeping the split that best balances the training/testing records (or the classes/bins of column when it is provided). This mirrors the balancing used by `cv_spatial` with `selection = "random"`, and `k_multiplier` controls the trade-off between fold balance (higher values) and fold compactness (lower values).

For environmental clustering (when `r` is provided), `spatial_weight` adds a soft spatial compactness pressure by blending the coordinates into the covariates before clustering, so the k-means Euclidean distance becomes $(1 - w)d_{env}^2 + w d_{geo}^2$: `spatial_weight = 0` (default) is pure environmental clustering, larger values pull the environmental clusters to be more geographically compact, and `spatial_weight = 1` clusters on the coordinates alone. This is a soft pressure – it favours, but does not guarantee, spatially contiguous folds. The coordinates are always standardised (so a `spatial_weight = 1` clustering is *not* identical to the coordinate-only clustering obtained with `r = NULL`, which uses the raw coordinates), while the covariates are standardised only when `scale = TRUE`; `spatial_weight` is best calibrated with the default `scale = TRUE`. Coordinates should be projected (a warning is issued for lon/lat data).

For presence-background data (`presence_bg = TRUE`), `column` holds 1 for presences and 0 for *background* points – locations sampled across the study area to represent the available conditions rather than confirmed absences. When `balance = TRUE` the balancing then equalises only the presence records across folds, so the abundant background cannot dominate the split.

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.
- `spatial_weight` - the spatial weight used for environmental clustering (NA for spatial clustering).
- `records` - a table with the number of points in each category of training and testing

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.

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`; `cv_plot` to visualise, and `cv_distance` and `cv_similarity` to evaluate, the folds

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)

# spatially constrained environmental clustering
set.seed(6)
sec <- cv_cluster(r = covars,
                  x = pa_data,
                  column = "occ",
                  k = 5,
                  scale = TRUE,
                  spatial_weight = 0.4) # blend geography into the environmental clusters

# spatial clustering with balanced folds
set.seed(6)
bc <- cv_cluster(x = pa_data,
                 column = "occ",
                 k = 5,
                 balance = TRUE, # balance the records across the folds
                 k_multiplier = 3) # cluster into k * k_multiplier groups, then merge into k folds

# presence-background data: balance folds based on presences
points_pb <- read.csv(system.file("extdata/", "species_pb.csv", package = "blockCV"))
pb_data <- sf::st_as_sf(points_pb, coords = c("x", "y"), crs = 7845)
# pb_data <- pb_data[c(which(pb_data$occ == 1)[1:50], which(pb_data$occ == 0)[1:200]), ]

bc_pb <- cv_cluster(x = pb_data,
                   column = "occ",
                   k = 5,
                   presence_bg = TRUE,
                   balance = TRUE,
                   k_multiplier = 5)

```

cv_distance

Compare a cross-validation design to the prediction domain via nearest-neighbour distances

Description

A diagnostic that works on *any* blockCV cross-validation object (the output of [cv_spatial](#), [cv_cluster](#), [cv_group](#), [cv_buffer](#), [cv_nndm](#) or [cv_knndm](#)). It draws the same nearest-neighbour distance distribution plot produced internally by [cv_nndm](#) and [cv_knndm](#), but for a fold configuration that has *already* been generated. This lets you check whether a given fold-generation strategy actually pushes the test-to-train distances towards the distances the model will face when predicting over the target area. It does not compute pairwise distances for general use; it is a fold-quality diagnostic.

Usage

```
cv_distance(
```

```

cv,
x,
r = NULL,
pred_points = NULL,
model_domain = NULL,
transfer = NULL,
space = "geographical",
add_random = TRUE,
num_random = 10L,
num_sample = 10000L,
sampling = "regular",
scale = TRUE,
seed = NULL,
plot = TRUE
)

```

Arguments

cv	a blockCV cross-validation object, i.e. the output of cv_spatial , cv_cluster , cv_group , cv_buffer , cv_nndm or cv_knndm .
x	a simple features (sf) object of the spatial sample points used to create the cv object.
r	a terra SpatRaster object. It defines the area the model predicts over; when neither pred_points nor model_domain is supplied, prediction points are sampled from it. It is also required (for the covariates) when space = "feature". One of r, pred_points, or model_domain is required.
pred_points	a simple features (sf) object of prediction points (optional). If provided, these are used directly as the prediction locations instead of sampling from r or model_domain.
model_domain	an sf polygon of the prediction area (optional). If provided (and pred_points is not), prediction points are sampled from it.
transfer	a terra SpatRaster object (optional) defining a second, transfer prediction domain – for example a future climate scenario or a different region – to compare the fold design against. It must carry the same predictor layers as r (identical names) and is only supported when space = "feature", where its covariates are standardised with the same centring/scaling as the training data. See details.
space	character; "geographical" (default) matches distances in geographical space, or "feature" matches distances in the (scaled) covariate space of r.
add_random	logical; when TRUE (default), overlay the CV curve of a random k-fold split with the same number of folds as cv (ignored for leave-one-out objects, see details).
num_random	integer; the number of random k-fold splits used to estimate the random baseline when add_random = TRUE. The plotted random curve is the mean across these splits, with a 10-90 band.
num_sample	integer; the number of prediction points to sample from r or model_domain when pred_points is not supplied.
sampling	either "regular" (default) or "random" for sampling prediction points.
scale	logical; whether to scale the covariates when space = "feature".

seed	integer; a random seed for reproducibility.
plot	logical; whether to draw the distance-distribution plot (default TRUE). The plot object is always built and returned in \$plot either way (so it can be customised or arranged with e.g. cowplot); plot = FALSE only suppresses drawing it now.

Details

Three to five empirical cumulative distribution functions of nearest-neighbour distances are compared over a distance r :

- **Prediction** ($\hat{G}_{ij}$) - from each prediction point to its nearest sample point. This is the distance regime the model meets at prediction time and is the target to match.
- **LOO** ($\hat{G}_j$) - from each sample point to its nearest *other* sample point. This is the nearest-neighbour (leave-one-out) limit, i.e. the most optimistic, random-like case.
- **CV** ($\hat{G}_j^*$) - from each test point to its nearest *training* point *within the same fold* of the supplied cv object. The closer this curve sits to the Prediction curve, the more the fold design mimics the prediction task.
- **Random** - the CV curve of a random k-fold split with the same number of folds as cv, drawn only when add_random = TRUE. It shows where a naive random assignment lands, so the improvement of the supplied design is visible. For leave-one-out objects (cv_buffer, cv_nndm) a random split coincides with the LOO curve, so it is omitted.
- **Transfer** - from each point of a transfer domain to its nearest sample point, drawn only when transfer is supplied (see below). This is the distance regime the model meets when transferred to that domain.

The gap between each curve and the Prediction curve is summarised by the Wasserstein-1 distance (the area between the two empirical distribution functions; lower means a closer match). These values are shown in the subtitle and attached to the returned plot as attr(p, "W").

When space = "feature" the distances are computed in the (optionally scaled) covariate space of r instead of the geographical space, mirroring [cv_knndm](#).

Transfer domain. When a transfer raster is supplied, the diagnostic adds the **Transfer** curve above and, in \$W_transfer, the Wasserstein-1 distance of the CV, LOO and Prediction curves to it. W_transfer["CV"] is the headline transferability read: how faithfully the fold design rehearses the transfer task (lower means the cross-validation is a fairer rehearsal of the transfer). A Transfer curve that sits to the right of the LOO curve flags a domain that no fold design built from these samples can emulate, i.e. extrapolation beyond the support of the training data, which cross-validation cannot validate. Because the transfer domain is compared in the (shared, training-scaled) covariate space, transfer is only supported when space = "feature"; it must carry the same predictor layers as r (matched by name and standardised the same way), and it may cover a different time (e.g. a future climate scenario) or a different region entirely.

When the supplied cv object was built with presence_bg = TRUE (in [cv_spatial](#), [cv_cluster](#), [cv_knndm](#), [cv_buffer](#) or [cv_nndm](#)), all three distance distributions are computed on the *presences* only: the prediction, LOO and CV curves are expressed relative to the presence-only training data rather than the (often random) *background* points (locations sampled across the study area to represent the available conditions rather than confirmed absences), matching the nearest neighbour distance matching framing. This is read automatically from cv.

Value

an object of class `cv_distance`: a list with

- `distances` - a per-fold data.frame of the test-to-nearest-train distances (the leakage signal): the number of test points (`n_test`), the min, quartiles (`q1`, `median`, `q3`) and max of those distances, and `pct_below_pred`, the percentage of test points nearer to training than the median prediction distance (higher means a more optimistic, leakier fold). For leave-one-out objects each fold holds a single test point, so there is one row per held-out point.
- `W` - a named numeric vector of the Wasserstein-1 distance of each curve to the Prediction curve (lower means a closer match to the prediction domain).
- `W_transfer` - only present when `transfer` is supplied: a named numeric vector of the Wasserstein-1 distance of the CV, L00 and Prediction curves to the Transfer curve (lower means a closer match to the transfer domain). `W_transfer["CV"]` is the transferability read.
- `plot` - the ggplot of the nearest-neighbour distance distributions (always built, whether or not it is drawn).

The distributions are drawn when `plot = TRUE` (default). Printing the object shows a compact text summary rather than redrawing the plot; call `plot()` on the returned object to redraw it.

References

Milà, C., Mateu, J., Pebesma, E., & Meyer, H. (2022). Nearest neighbour distance matching leave-one-out cross-validation for map validation. *Methods in Ecology and Evolution*, 13(6), 1304-1316.

Linnenbrink, J., Milà, C., Ludwig, M., & Meyer, H. (2024). kNNDM CV: k-fold nearest neighbour distance matching cross-validation for map accuracy estimation. *Geoscientific Model Development*, 17(15), 5897-5912.

See Also

[cv_similarity](#), [cv_knndm](#), [cv_nndm](#), [cv_spatial](#), [cv_cluster](#), [cv_group](#), [cv_buffer](#), and [cv_plot](#) to visualise the folds

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)

# generate spatial folds
sb <- cv_spatial(x = pa_data, column = "occ", size = 450000, k = 5, iteration = 1)

# assess how close the folds are to the prediction domain
```

```
cv_distance(cv = sb, x = pa_data, r = covar, num_sample = 5000)
```

cv_group

Leave-group-out cross-validation using an existing grouping factor

Description

This function creates cross-validation folds from a grouping column that is already present in the data – for example a *site*, *plot*, *campaign*, or *individual* identifier. All records that share a group are always kept together, so a group is never split across the training and testing sets. This is the hierarchical / grouped ("leave-group-out") blocking that is not covered by the clustering functions: unlike `cv_cluster` (which *builds* groups with k-means) or `cv_knndm` (whose "hierarchical" option is agglomerative clustering), `cv_group` takes the groups as given.

Usage

```
cv_group(
  x,
  group_col,
  k = NULL,
  column = NULL,
  balance = FALSE,
  iteration = 100L,
  seed = NULL,
  biomod2 = TRUE,
  num_bins = 4L,
  report = interactive(),
  progress = interactive()
)
```

Arguments

x	a simple features (sf) or SpatialPoints object of spatial sample data (e.g., species data or ground truth sample for image classification).
group_col	character. The name of the column in x that holds the grouping factor (e.g. site, plot, campaign, or individual ID). Records sharing a value are kept together in the same fold. Missing values are not allowed.
k	integer (optional). The number of desired folds. When NULL (default) or greater than or equal to the number of groups, leave-group-out is used (one fold per group). When smaller than the number of groups, the groups are merged into k folds. See ‘Details’.
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. It is used to report whether all the folds contain all the classes and, when balance = TRUE, to balance those classes across the folds. Continuous numeric responses are binned into quantiles using num_bins before records are counted.

balance	logical. Only used when k is smaller than the number of groups. If TRUE, the groups are assigned to the k folds over iteration random attempts to balance the training/testing records (or the classes/bins of column when it is provided). If FALSE (default), the groups are distributed across the folds deterministically. Ignored for leave-group-out.
iteration	integer value. The number of random attempts to assign the groups to folds when balance = TRUE.
seed	integer; a random seed for reproducibility of the balancing search.
biomod2	logical. Creates a matrix of folds that can be directly used in the biomod2 package as a <i>CV.user.table</i> for cross-validation.
num_bins	integer; the number of quantile bins used to stratify a continuous numeric column. The default is 4. Set num_bins = NULL to disable binning and treat every unique value as a separate class (the behaviour prior to version 3.3). If quantile breaks are tied, fewer bins may be used. The raw response values are not modified; bins are only used for fold balancing and record summaries.
report	logical; whether to print the report of the records per fold. Defaults to <code>interactive()</code> .
progress	logical; whether to shows a progress bar for random fold selection. Defaults to <code>interactive()</code> .

Details

The number of folds is controlled by k:

- k = NULL (default) or k greater than or equal to the number of groups – **leave-group-out**: every group forms its own fold and is left out once. In this mode balance has no effect because the folds are fixed by the groups.
- k less than the number of groups – the groups are **merged into k folds**. With balance = FALSE (default) the groups are distributed across the folds deterministically (keeping a similar number of groups per fold). With balance = TRUE the groups are assigned to the k folds over iteration random attempts, keeping the split that best balances the training/testing records (or the classes/bins of column when it is provided), mirroring the balancing used by `cv_cluster`.

In every case whole groups move together, so $k < n_{\text{groups}}$ still yields folds that respect the grouping structure while giving fewer, larger folds than leave-group-out.

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

- group_col - the name of the grouping column
- type - indicates whether leave-group-out or merged grouping was used
- records - a table with the number of points in each category of training and testing

See Also

[cv_cluster](#) and [cv_spatial](#); [cv_plot](#) to visualise, and [cv_distance](#) and [cv_similarity](#) to evaluate, the folds

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)

# add an example grouping column (e.g. survey site)
pa_data$site <- sample(paste0("site_", 1:8), nrow(pa_data), replace = TRUE)

# leave-group-out: one fold per site
lgo <- cv_group(x = pa_data,
                group_col = "site",
                column = "occ") # optional; name of the column with response

# merge the sites into 4 balanced folds
set.seed(6)
gm <- cv_group(x = pa_data,
               group_col = "site",
               column = "occ",
               k = 4,
               balance = TRUE)
```

cv_knndm

Use the k-fold Nearest Neighbour Distance Matching (kNNDM) to separate train and test folds

Description

A k-fold version of the Nearest Neighbour Distance Matching algorithm (Linnenbrink et al., 2024). Unlike [cv_nndm](#), which is a leave-one-out (LOO) strategy producing as many folds as there are points, kNNDM produces a small number of k folds (cheap to fit models on) while still matching the nearest neighbour distance distribution function between the test and training data to the one between the target prediction points and the training data.

Usage

```

cv_knndm(
  x,
  column = NULL,
  r = NULL,
  pred_points = NULL,
  model_domain = NULL,
  k = 5L,
  maxp = 0.5,
  clustering = "blocks",
  space = "geographical",
  hexagon = TRUE,
  keep_blocks = TRUE,
  num_sample = 10000L,
  sampling = "regular",
  nk_len = 100L,
  linkage = "ward.D2",
  scale = TRUE,
  balance = TRUE,
  presence_bg = FALSE,
  biomod2 = TRUE,
  deg_to_metre = 111325,
  seed = NULL,
  num_bins = 4L,
  plot = interactive(),
  report = interactive()
)

```

Arguments

<code>x</code>	a simple features (sf) or SpatialPoints object of spatial sample data (e.g., species data or ground truth sample for image classification).
<code>column</code>	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 used to report whether all folds contain all classes and, when <code>balance = TRUE</code> , to prefer class-complete candidate groupings.
<code>r</code>	a terra SpatRaster object. This defines the area that the model is going to predict; when neither <code>pred_points</code> nor <code>model_domain</code> is supplied, prediction points are sampled from it. It is also required (for the covariates) when <code>space = "feature"</code> .
<code>pred_points</code>	a simple features (sf) object of prediction points (optional). If provided, these are used directly as the prediction locations instead of sampling from <code>r</code> or <code>model_domain</code> .
<code>model_domain</code>	an sf polygon of the prediction area (optional). If provided (and <code>pred_points</code> is not), prediction points are sampled from it.
<code>k</code>	integer value. The number of desired folds for cross-validation. The default is <code>k = 5</code> .

maxp	numeric; strictly between 1/k and 1. The maximum proportion of the points allowed in any one fold; groupings that exceed it are discarded.
clustering	character; the method used to group the points before allocating them to folds. One of "blocks" (default; a blockCV spatial-blocks variant), "hierarchical" (hclust), or "kmeans" (kmeans).
space	character; "geographical" (default) matches distances in geographical space, or "feature" matches distances in the (scaled) covariate space of <i>r</i> .
hexagon	logical. Creates hexagonal (default) spatial blocks when clustering = "blocks". If FALSE, square blocks are used.
keep_blocks	logical; if TRUE (default) and the folds were produced by clustering = "blocks", the spatial blocks of the selected grouping are returned in the output (element blocks) so they can be drawn as a background layer by cv_plot . It has no effect for the "hierarchical" and "kmeans" methods, in feature space, or when a random cross-validation is returned; in those cases blocks is NULL.
num_sample	integer; the number of prediction points to sample from <i>r</i> or model_domain when pred_points is not supplied.
sampling	either "regular" (default) or "random" for sampling prediction points.
nk_len	integer; the number of candidate groupings (numbers of clusters / block sizes) to explore.
linkage	character; the agglomeration method passed to hclust when clustering = "hierarchical".
scale	logical; whether to scale the covariates when space = "feature".
balance	logical. When TRUE (default) and column is supplied, class completeness is used as a validity gate to prefer groupings in which every class is present in every test fold (see details). When FALSE, the grouping with the smallest <i>W</i> is returned regardless of class completeness and column only feeds the report.
presence_bg	logical; whether to treat column as species presence-background data (0s for background points and 1s for presences; see 'Details'). When TRUE, the whole distance matching (the sample-to-sample, prediction-to-sample and test-to-train nearest-neighbour distances, and the grouping search) is computed on the <i>presences</i> only, so the prediction domain is expressed relative to the presence-only training data rather than a random background. Each background point then inherits the fold of its nearest presence. The class-completeness gate of balance is not applied in this mode. Requires a binary numeric column. The default is FALSE.
biomod2	logical. Creates a matrix of folds that can be directly used in the biomod2 package as a <i>CV.user.table</i> for cross-validation.
deg_to_metre	integer. The conversion rate of metres to degree. See the details section for more information.
seed	integer; a random seed for reproducibility.
num_bins	integer; the number of quantile bins used to stratify a continuous numeric column. The default is 4. Set num_bins = NULL to disable binning and treat every unique value as a separate class (the behaviour prior to version 3.3). If quantile breaks are tied, fewer bins may be used. The raw response values are not modified; bins are only used for fold balancing and record summaries.

plot	logical; whether to plot the distance distribution functions. Defaults to <code>interactive()</code> .
report	logical; whether to print summary of records in each fold. Defaults to <code>interactive()</code> .

Details

The method groups the sample points and assigns whole groups to folds, tuning the degree of aggregation so that the resulting test-to-train nearest neighbour distance distribution ($\hat{G}_j^*$) best matches the prediction-to-sample distribution ($\hat{G}_{ij}$). The match is scored with the Wasserstein-1 distance (W in the output; the area between the two empirical distribution functions), and the grouping with the smallest value is returned. Following Linnenbrink et al. (2024), a one-sided Kolmogorov-Smirnov test is first used to check whether the points are clustered relative to the prediction area; if they are not, a random cross-validation is returned. When a clustering is needed, the intermediate groups are ordered along the first principal component of the coordinates and distributed across the k folds, and only groupings in which no fold holds more than `maxp` of the data are considered.

Three ways of generating the groups are available via clustering: "hierarchical" and "kmeans" follow Linnenbrink et al. (2024), while "blocks" (the default) is a blockCV-specific variant that groups the points with hexagonal (or square) spatial blocks, in the spirit of `cv_spatial`. The "blocks" option is a blockCV extension and is not part of the original kNNDM algorithm. When `space = "feature"` the distances are computed in the (scaled) covariate space of `r` instead of the geographical space, and "blocks" is not available.

When `column` is supplied and `balance = TRUE`, class balance is used as a validity gate during the candidate scan: among groupings with every class present in every test fold, the one with the smallest W is selected. If no class-complete grouping is available, the overall smallest- W grouping is returned and the final records table reports the missing class(es). Note that this gate is weaker than the balancing in `cv_spatial`: it only ensures every class is represented in every test fold (no empty class), it does not equalise the class counts across folds. With `balance = FALSE` the grouping with the overall smallest W is always returned and `column` is used only for the report.

For presence-background data (`presence_bg = TRUE`), `column` holds 1 for presences and 0 for *background* points – locations sampled across the study area to represent the available conditions rather than confirmed absences. The distance matching is then computed on the presences only and each background point inherits the fold of its nearest presence, so the prediction geometry reflects the presence data rather than a random background.

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` - the clustering method used (or "random" if a random CV was returned)
- `space` - whether distances were matched in geographical or feature space

- q - the selected number of intermediate groups
- W - the Wasserstein statistic of the selected folds (lower means a better match)
- Gij - the prediction-to-sample nearest neighbour distances
- Gj - the sample-to-sample (leave-one-out) nearest neighbour distances
- Gjstar - the test-to-train nearest neighbour distances of the selected folds
- blocks - the spatial blocks of the selected grouping, each tagged with its fold; only when clustering = "blocks" and keep_blocks = TRUE, otherwise NULL
- records - a table with the number of points in each category of training and testing

References

Linnenbrink, J., Milà, C., Ludwig, M., & Meyer, H. (2024). kNNDM CV: k-fold nearest neighbour distance matching cross-validation for map accuracy estimation. *Geoscientific Model Development*, 17(15), 5897-5912. <https://doi.org/10.5194/gmd-17-5897-2024>

Milà, C., Mateu, J., Pebesma, E., & Meyer, H. (2022). Nearest neighbour distance matching leave-one-out cross-validation for map validation. *Methods in Ecology and Evolution*, 13(6), 1304-1316.

See Also

[cv_nndm](#), [cv_spatial](#), [cv_cluster](#), [cv_spatial_autocor](#), and `CAST::knndm()` for the authors' CAST implementation of kNNDM. Use [cv_plot](#) to visualise, and [cv_distance](#) and [cv_similarity](#) to evaluate, the folds.

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)

knndm <- cv_knndm(x = pa_data,
                  column = "occ", # optional
                  r = covar,
                  k = 5,
                  num_sample = 5000)
```

cv_nndm	<i>Use the Nearest Neighbour Distance Matching (NNDM) to separate train and test folds</i>
---------	--

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 = NULL,
  size,
  pred_points = NULL,
  model_domain = NULL,
  num_sample = 10000L,
  sampling = "random",
  min_train = 0.05,
  presence_bg = FALSE,
  add_bg = FALSE,
  num_bins = 4L,
  plot = interactive(),
  report = interactive()
)
```

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 (optional). This defines the area that the model is going to predict; prediction points are sampled from it when neither pred_points nor model_domain is supplied. One of r, pred_points, or model_domain is required.
size	numeric value of the range of spatial autocorrelation (the phi parameter). This distance should be in metres (or in the coordinate units of x when its CRS is undefined). The range could be explored by cv_spatial_autocor . For residual-based guidance, add model residuals to x and pass that residual column to cv_spatial_autocor .

pred_points	a simple features (sf) object of prediction points (optional). If provided, these are used directly as the prediction locations instead of sampling from <code>r</code> or <code>model_domain</code> .
model_domain	an sf polygon of the prediction area (optional). If provided (and <code>pred_points</code> is not), prediction points are sampled from it.
num_sample	integer; the number of sample points from predictor (<code>r</code>) to be used for calculating the G function of prediction points.
sampling	either "random" or "regular" for sampling prediction points. When <code>sampling = "regular"</code> , the actual number of samples might be less than <code>num_sample</code> 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 <code>presence_bg = TRUE</code> . We do not recommend this according to Radosavljevic & Anderson (2014). Keep it FALSE, unless you mean to add the background points to testing points.
num_bins	integer; the number of quantile bins used to stratify a continuous numeric column. The default is 4. Set <code>num_bins = NULL</code> to disable binning and treat every unique value as a separate class (the behaviour prior to version 3.3). If quantile breaks are tied, fewer bins may be used. The raw response values are not modified; bins are only used for fold balancing and record summaries.
plot	logical; whether to plot the G functions. Defaults to <code>interactive()</code> .
report	logical; whether to print summary of records in each fold. Defaults to <code>interactive()</code> .

Details

When working with presence-background data (presences and *background* points sampled across the study area to represent the available conditions rather than confirmed absences, specified by `presence_bg = TRUE`), 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 = FALSE`, 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 separate training and testing folds)
- `exclusion` - a data.frame with one row per fold: the fold number, the row index of its test point in `x` (`test_id`), and the `exclusion_distance` matched to that point. Unlike `cv_buffer`, where the exclusion distance is the constant size, NNDM matches a distinct radius to each test point: every point within `exclusion_distance` of the test point is held out, and the points beyond it form the training set. Note this is *not* capped by `size`: the matching only thins neighbours closer than `size`, so a point whose nearest neighbour already lies beyond `size` is left untouched and reports its (larger) natural nearest-neighbour distance
- `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`, `cv_knndm` and `cv_spatial_autocor`; `cv_plot` to visualise, and `cv_distance` and `cv_similarity` to evaluate, the folds

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. The output of `cv_plot()` is a ggplot object, so it can be customised with standard ggplot2 layers. For example, you can add a title, change the theme, or modify colours and scales.

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,
  bg_alpha = 0.1,
  label_size = 4,
  combine_folds = FALSE,
  fold_colors = NULL
)
```

Arguments

cv	a blockCV <code>cv_*</code> object; a <code>cv_spatial</code> , <code>cv_cluster</code> , <code>cv_group</code> , <code>cv_buffer</code> , <code>cv_nndm</code> , or <code>cv_knndm</code>
x	a simple features (sf) or SpatialPoints object of the spatial sample data used for creating the cv object. This is required for point-based objects such as <code>cv_cluster</code> , <code>cv_group</code> , <code>cv_buffer</code> , <code>cv_nndm</code> , and <code>cv_knndm</code> ; it can be omitted for <code>cv_spatial</code> objects.
r	a terra SpatRaster object (optional). If provided, it will be used as background of the plots. It also supports <i>stars</i> , <i>raster</i> , 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. <code>1:3</code> or <code>c(2, 7, 16, 22)</code>
max_pixels	integer; maximum number of pixels used for plotting r

<code>remove_na</code>	logical; whether to remove excluded points in <code>cv_buffer</code> from the plot
<code>raster_colors</code>	character; a character vector of colours for raster background e.g. <code>terrain.colors(20)</code>
<code>points_colors</code>	character; two colours to be used for train and test points
<code>points_alpha</code>	numeric; the opacity of points
<code>bg_alpha</code>	numeric; opacity of the <i>background</i> points (response 0) when the <code>cv</code> object was built with <code>presence_bg = TRUE</code> (see ‘Details’). Lower values fade them so the presences (response 1) stand out; set <code>bg_alpha = points_alpha</code> to disable the fading. Values above <code>points_alpha</code> are capped at <code>points_alpha</code> (with a warning) so the background is never drawn more prominent than the presences. Has no effect for objects that are not presence-background. The default is 0.1.
<code>label_size</code>	integer; size of fold labels when a <code>cv_spatial</code> object is used.
<code>combine_folds</code>	logical; if TRUE, all folds are shown in a single map with points coloured by their fold ID instead of separate train/test facets. Only available for <code>cv_spatial</code> , <code>cv_cluster</code> , <code>cv_group</code> and <code>cv_knndm</code> objects.
<code>fold_colors</code>	character; a vector of colours for the folds when <code>combine_folds = TRUE</code> ; by default a qualitative palette is generated for the number of folds.

Details

The `bg_alpha` argument only applies to objects created with `presence_bg = TRUE`. There the `cv` response holds 1 for presences and 0 for *background* points – locations sampled across the study area to represent the available conditions rather than confirmed absences – and the background points are drawn faded so the presences stand out. This point-level “background” is unrelated to the raster `r` used as a map backdrop.

Value

a `ggplot` object

See Also

[cv_distance](#) and [cv_similarity](#) to evaluate the folds; [cv_spatial](#), [cv_cluster](#), [cv_buffer](#), [cv_nndm](#) and [cv_knndm](#) to create them

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	<i>Compute similarity measures to evaluate possible extrapolation in testing folds</i>
---------------	--

Description

This function evaluates environmental similarity between training and testing folds, helping to detect potential extrapolation in the testing data. It supports three similarity outputs: MESS, L1, and L2. The L1 and L2 options are distance-based similarity scores.

Usage

```
cv_similarity(
  cv,
  x,
  r,
  num_plots = seq_along(cv$folds_list),
  method = "MESS",
  type = "distribution",
  num_sample = 10000L,
  seed = NULL,
  jitter_width = 0.1,
  points_size = 2,
  points_alpha = 0.7,
  points_colors = NULL,
  progress = interactive(),
  plot = TRUE
)
```

Arguments

cv	a blockCV cv_* object; a cv_spatial, cv_cluster, cv_group, cv_buffer, cv_nndm, or cv_knndm
x	a simple features (sf) object of the spatial sample points 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_plots	a vector of indices of folds for plotting (default uses all).
method	the similarity method: MESS, L1, or L2. Read the details section.

type	character; "distribution" (default) draws the per-fold similarity distributions, while "map" plots the sample points in geographical space and colours each test point by its similarity value, showing <i>where</i> extrapolation occurs.
num_sample	number of random raster samples used for the L1/L2 baseline.
seed	integer; an optional random seed. The L1/L2 baseline is built from a random raster sample (num_sample); set seed to make the result reproducible.
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 the diverging value scale. Defaults to a red-grey-blue ramp (red = novel/negative, blue = similar/positive) centred at zero.
progress	logical; whether to show a progress bar while computing fold similarities. Defaults to <code>interactive()</code> .
plot	logical; whether to draw the plot (default TRUE). The plot object is always built and returned in <code>\$plot</code> either way (so it can be customised or arranged); <code>plot = FALSE</code> only suppresses drawing it now.

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) score 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 similarity score, not a raw distance: values below zero indicate test samples that are farther from their training data than the random-background baseline. Note that this approach is experimental.

When the supplied *cv* object was built with `presence_bg = TRUE`, the similarity is computed on the *presences* only: the background points (locations sampled to represent the available conditions rather than confirmed absences) are excluded from both the training and testing sets of every fold, matching the framing used in `cv_distance`. This is read automatically from *cv*.

Value

an object of class `cv_similarity`: a list with

- `extrapolation` - a per-fold `data.frame` with the number of test points, the percentage flagged as novel (similarity value below zero), the minimum and median similarity, and – for method = "MESS" – the most limiting variable (the covariate most often driving the novelty).
- `overall` - the overall novelty rate (percentage of test points below zero) across the plotted folds.
- `plot` - the `ggplot` (a per-fold distribution, or a spatial map for `type = "map"`), always built whether or not it is drawn.

The plot is drawn when `plot = TRUE` (default). Printing the object shows a compact text summary rather than redrawing the plot; call `plot()` on the returned object to redraw it.

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](#); [cv_plot](#) to visualise, and [cv_distance](#) to evaluate, the folds

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, method = "MESS")
```

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,
  balance = TRUE,
  presence_bg = FALSE,
  user_blocks = NULL,
  folds_column = NULL,
  deg_to_metre = 111325,
  biomod2 = TRUE,
  offset = c(0, 0),
  extend = 0,
  seed = NULL,
  num_bins = 4L,
  plot = interactive(),
  report = interactive(),
  progress = interactive(),
  ...
)
```

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. It is used to report the number of records of each class/bin in every fold and, when balance = TRUE, to balance those classes across the folds. If column = NULL the response variable classes will be treated the same and only training and testing records will be counted (and balanced). This is used for binary (e.g. presence-absence/background) or multi-class responses (e.g. land cover classes for remote sensing image classification). Continuous numeric responses are binned into quantiles using num_bins before balancing.
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. For residual-based block-size guidance, add model residuals to x and pass that residual column to cv_spatial_autocor .
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".
balance	logical. When TRUE (default) and selection = "random", the folds are chosen from iteration random block assignments to balance the training/testing records (or the classes/bins of column when it is provided). If FALSE, a single random assignment is returned without balancing (equivalent to iteration = 1). This argument has no effect on the other selection methods.
presence_bg	logical; whether to treat column as species presence-background data (0s for background points and 1s for presences; see 'Details'). When TRUE (with selection = "random" and balance = TRUE), the balancing search equalises only the presence (1s) records across folds so the many background points cannot dominate the objective; the blocks still contain all points but the background is ignored when scoring the balance. Requires a binary numeric column. The default is FALSE.

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 'pre-defined' 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 <i>CV.user.table</i> 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).
num_bins	integer; the number of quantile bins used to stratify a continuous numeric column. The default is 4. Set num_bins = NULL to disable binning and treat every unique value as a separate class (the behaviour prior to version 3.3). If quantile breaks are tied, fewer bins may be used. The raw response values are not modified; bins are only used for fold balancing and record summaries.
plot	logical; whether to plot the final blocks with fold numbers in ggplot. Defaults to interactive(). You can re-create this with cv_plot .
report	logical; whether to print the report of the records per fold. Defaults to interactive().
progress	logical; whether to shows a progress bar for random fold selection. Defaults to interactive().
...	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 $\cos(\text{mean}(\text{sf}::\text{st_bbox}(x)[c(2,4)]) * \pi/180) * 111325$.

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](#)). When using [cv_spatial_autocor](#) to inform size, note that response or covariate variograms are exploratory proxies for residual autocorrelation unless model residuals have been added to `x` and supplied through the `column` argument.

For presence-background data (`presence_bg = TRUE`), `column` holds 1 for presences and 0 for *background* points – locations sampled across the study area to represent the available conditions rather than confirmed absences. The fold balancing then targets only the presence records (the blocks still contain all points), so the abundant background cannot dominate the split.

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
- `block_shape` - the block geometry used: "hexagon", "square", or "user-defined"
- `selection` - how blocks were assigned to folds: "random", "systematic", "checkerboard", or "predefined"
- `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

- 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.
- 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; [cv_plot](#) to visualise, and [cv_distance](#) and [cv_similarity](#) to evaluate, the folds

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	<i>Measure spatial autocorrelation in spatial response data or predictor raster files</i>
--------------------	---

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 variable defined by `column` in spatial sample points can be a binary response such as species distribution data, a continuous response such as soil organic carbon, or model residuals that have been added to `x`. The function estimates spatial autocorrelation *ranges* of all input raster layers or the supplied point-data column. 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

<code>r</code>	a terra SpatRaster object. If provided (and <code>x</code> is missing), it will be used for to calculate range.
<code>x</code>	a simple features (sf) or SpatialPoints object of spatial sample data (e.g., species binary or continuous data).
<code>column</code>	character; indicating the name of the column in which the response variable (e.g. species data as a binary response i.e. 0s and 1s) or model residuals are stored for calculating spatial autocorrelation range. This supports multiple column names.
<code>num_sample</code>	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).
<code>deg_to_metre</code>	integer. The conversion rate of degrees to metres.
<code>plot</code>	logical; whether to plot the results.
<code>progress</code>	logical; whether to shows a progress bar.
<code>...</code>	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.

Roberts et al. (2017) recommend choosing spatial blocks with reference to the autocorrelation range of model residuals. To follow that guidance, fit the model first, add its residuals to `x`, and pass that residual column name to `column`. Ranges estimated from the raw response or raster covariates are useful exploratory guides before model fitting, but they are not residual autocorrelation ranges and may under- or over-estimate the block size needed for residual-based guidance.


```
summary(sac2)
```

cv_summary

Summarise the quality of a set of cross-validation folds

Description

A one-call diagnostic that gathers the fold-quality signals scattered across blockCV into a single object: fold sizes and class prevalence (always), the test-to-nearest-train distance and environmental-novelty diagnostics (when a raster or prediction domain is supplied), and a list of automatically detected warnings about degenerate folds.

Usage

```
cv_summary(
  cv,
  x = NULL,
  r = NULL,
  pred_points = NULL,
  model_domain = NULL,
  method = "MESS",
  num_sample = 10000L,
  min_test = 5L,
  seed = NULL,
  progress = FALSE
)
```

Arguments

cv	a blockCV cross-validation object, i.e. the output of cv_spatial , cv_cluster , cv_group , cv_buffer , cv_nndm or cv_knndm .
x	a simple features (sf) object of the sample points used to create cv. Required only for the distance/novelty diagnostics (when r, pred_points or model_domain is supplied).
r	a terra SpatRaster of the environmental covariates. When supplied, both the distance and the environmental-novelty diagnostics are computed. Optional.
pred_points, model_domain	optional prediction points or a model-domain polygon used by the distance diagnostic; see cv_distance . Supplying either (without r) adds the distance diagnostic but not the novelty diagnostic.
method	the similarity method passed to cv_similarity : "MESS", "L1" or "L2".
num_sample	number of random raster samples used by the distance and novelty diagnostics.
min_test	integer; folds with fewer than this many test points are flagged as tiny_test.
seed	integer; an optional random seed for the raster-sampling baselines.
progress	logical; whether to show a progress bar for the novelty diagnostic.

Details

The structural part (sizes, class prevalence and the structural warnings) is read from the `cv` object itself and needs no raster. When `r` (or `pred_points/model_domain`) is supplied, the per-fold distance diagnostic of `cv_distance` is added; when the covariate raster `r` is supplied, the per-fold environmental-novelty diagnostic of `cv_similarity` is added as well. These reuse the same computations as those functions.

The warnings are returned as data (a `data.frame`), not raised as `warning()` conditions, so they can be inspected programmatically. The structural warnings are a k-fold concept and are skipped for leave-one-out objects (`cv_buffer`, `cv_nndm`), whose folds hold a single test point by design. The flagged issues are:

- `empty_test` - a fold with no test points.
- `single_class_test` - a fold whose test set contains a single class (breaks AUC and other class-wise metrics).
- `class_missing_train` - a class entirely absent from a fold's training set.
- `tiny_test` - a fold with fewer than `min_test` test points (unstable per-fold estimate).
- `imbalance` - a class severely under-represented in a fold's test set relative to an even split.
- `high_leakage` - a fold where at least 90% of test points sit nearer to training than the median prediction distance (only available when the distance diagnostic is computed).

Value

an object of class `cv_summary`: a list with

- `n_folds` - the number of folds.
- `is_loo` - whether `cv` is a leave-one-out object.
- `records` - the per-fold train/test counts per class (fold sizes and class prevalence).
- `distances` - the per-fold test-to-nearest-train distance summary, or `NULL`.
- `novelty` - the per-fold environmental-novelty (extrapolation) summary, or `NULL`.
- `warnings` - a `data.frame` of flagged degenerate folds (fold, type, message); zero rows when nothing is flagged.

See Also

`cv_distance` and `cv_similarity` for the individual diagnostics, and `cv_plot` to visualise the folds

Examples

```
library(blockCV)

points <- read.csv(system.file("extdata/", "species.csv", package = "blockCV"))
pa_data <- sf::st_as_sf(points, coords = c("x", "y"), crs = 7845)
covar <- terra::rast(system.file("extdata/au/bio_5.tif", package = "blockCV"))

sb <- cv_spatial(x = pa_data, column = "occ", size = 450000, k = 5, iteration = 1)

# structural summary + warnings only (no raster)
```

```
cv_summary(sb)

# add the distance and novelty diagnostics
cv_summary(sb, x = pa_data, r = covar, num_sample = 5000)
```

Index

autofitVariogram, [35](#)

BIOMOD_Modeling, [33](#)

blockCV, [2](#)

blockCV-package (blockCV), [2](#)

cv_block_size, [3](#), [31](#), [33](#), [36](#)

cv_buffer, [4](#), [10–14](#), [22](#), [24](#), [26](#), [29](#), [33](#), [37](#), [38](#)

cv_cluster, [7](#), [11–17](#), [21](#), [26](#), [29](#), [33](#), [37](#)

cv_distance, [3](#), [6](#), [10](#), [11](#), [17](#), [21](#), [24](#), [26](#), [28](#),
[29](#), [33](#), [36–38](#)

cv_group, [11](#), [12](#), [14](#), [15](#), [37](#)

cv_knndm, [11–15](#), [17](#), [24](#), [26](#), [37](#)

cv_nndm, [6](#), [11–14](#), [17](#), [21](#), [22](#), [26](#), [29](#), [37](#), [38](#)

cv_plot, [3](#), [6](#), [10](#), [14](#), [17](#), [19](#), [21](#), [24](#), [25](#), [29](#),
[32](#), [33](#), [35](#), [36](#), [38](#)

cv_similarity, [3](#), [6](#), [10](#), [14](#), [17](#), [21](#), [24](#), [26](#),
[27](#), [33](#), [36–38](#)

cv_spatial, [3](#), [6](#), [9–14](#), [17](#), [20](#), [21](#), [26](#), [29](#), [30](#),
[37](#)

cv_spatial_autocor, [3–6](#), [21](#), [22](#), [24](#), [31](#), [33](#),
[34](#)

cv_summary, [37](#)

hclust, [19](#)

kmeans, [7](#), [19](#)

variogram, [35](#)