Package ‘mbg’

July 22, 2025

Title Model-Based Geostatistics

Version 1.1.0

Description Modern model-based geostatistics for point-referenced data. This package provides a simple interface to run spatial machine learning models and geostatistical models that estimate a continuous (raster) surface from point-referenced outcomes and, optionally, a set of raster covariates. The package also includes functions to summarize raster outcomes by (polygon) region while preserving uncertainty.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Imports assertthat, caret, data.table, glue, Matrix, matrixStats, purrr, R6, sf, terra, tictoc

Suggests INLA, knitr, rmarkdown, ggplot2, scales

Additional_repositories `https://inla.r-inla-download.org/R/stable/`

Encoding UTF-8

RoxygenNote 7.2.3

VignetteBuilder knitr

URL `https://henryspatialanalysis.github.io/mbg/`

BugReports `https://github.com/henryspatialanalysis/mbg/issues`

NeedsCompilation no

Author Nathaniel Henry [aut, cre] (ORCID: <`https://orcid.org/0000-0001-8150-4988`>), Benjamin Mayala [aut]

Maintainer Nathaniel Henry <`nat@henryspatialanalysis.com`>

Repository CRAN

Date/Publication 2025-04-26 02:20:02 UTC

Contents

`.onAttach` *Behavior when attaching the mbg package*

Description

Behavior when attaching the mbg package

Usage

```
.onAttach(libname, pkgname)
```

Arguments

`libname`	(character(1)) A character string giving the library directory where the package defining the namespace was found.
`pkgname`	(character(1)) A character string giving the name of the package.

Details

Yields a message if the INLA package namespace is not available.

Value

(invisible) A message may be printed to the console.

`aggregate_draws_to_polygons`
Aggregate grid cell draws to polygons

Description

Aggregate grid cell draws to polygons using an aggregation table

Usage

```
aggregate_draws_to_polygons(
  draws_matrix,
  aggregation_table,
  aggregation_cols = "polygon_id",
  method = "mean",
  z_dimension = NULL,
  z_dimension_name = "z",
  weighting_raster = NULL,
  na.rm = TRUE
)
```

Arguments

`draws_matrix` `matrix`, `array`, or `data.frame` corresponding to grid cell draws that will be aggregated to polygons:

- Each row represents a non-NA grid cell in the ID raster. If the matrix contains multiple years of estimates, the matrix is ordered by year, then by masked_pixel_id. For example, if there are 200 non-NA pixels in the ID raster and five years of draws, then the matrix contains 1000 rows: row 200 corresponds to (year 1, masked_pixel_id 200), row 201 corresponds to (year 2, masked_pixel_id 1), and so on.
- Each column represents a draw. There should be no non-draw columns (such as ID fields) in the `draws_matrix`.

`aggregation_table` data.table::data.table Aggregation table linking pixels to polygon identifiers, created using `build_aggregation_table()`

`aggregation_cols` (character vector, default 'polygon_id') Polygon identifiers to use for aggregation. Must be field names within `aggregation_table`.

`method` (character, default 'mean') Aggregation method: one of 'mean', 'sum', 'weighted.mean', or 'weighted.sum'. The latter two methods require a `weighting_raster`.

`z_dimension` (vector, default NULL) If passing a `data_raster` with multiple layers, how should each layer be identified? Should be a vector with length equal to the number of layers in `data_raster`. If left as `NULL`, the default, and `data_raster` has 1 layer, no z dimension will be added. If left as `NULL` and `data_raster` has more than 1 layer, will default to (1, 2, ..., N layers).

`z_dimension_name` (default 'z') The field name for the "z" dimension corresponding to each layer of the `data_raster`. This field is only added if `z_dimension` is passed or if `data_raster` has more than one layer.

`weighting_raster` (terra::SpatRaster, default NULL) The relative weighting of each whole pixel to the overall polygon value, for example, if calculating a population-weighted mean. Required for methods 'weighted.mean' and 'weighted.sum', ignored for the other methods.

`na.rm` (bool, default TRUE) How to handle NA pixels in `data_raster` and `weighting_raster`. If set to TRUE but ALL pixels in a polygon are NA, will still return an NA value for the polygon.

Details

This is a more efficient and feature-rich alternative to terra's zonal statistics functions. Features include:

- Always fractionally aggregate, weighting by area of the pixel in a polygon
- Optionally weight by both area fraction and a weighting raster (e.g. population)
- Means or sums of raster values across polygons
- Optionally aggregate multiple years of raster data at once

Value

data.table::data.table containing polygon identifiers, (optionally) layer identifiers in the `z_dimension_name` column, and data values aggregated by polygon.

See Also

build_aggregation_table

Examples

```
## Not run:
  polygons <- sf::st_read(system.file('extdata/Benin_communes.gpkg', package = 'mbg'))
  id_raster <- build_id_raster(polygons)
  n_data_pixels <- sum(!is.na(terra::values(id_raster)))
  # Example grid-level draws from e.g. mbg::generate_cell_draws_and_summarize()
  draws_matrix <- matrix(rnorm(n_data_pixels * 5), nrow = n_data_pixels)
  # Build aggregation table, which can be used across many aggregations
  aggregation_table <- build_aggregation_table(
    polygons, id_raster, polygon_id_field = 'commune_code'
  )
  # Aggregate the grid-level draws to polygon-level draws
  aggregated <- aggregate_draws_to_polygons(
    draws_matrix = draws_matrix,
    aggregation_table = aggregation_table,
    aggregation_cols = 'commune_code',
    method = 'mean'
  )
  head(aggregated)

## End(Not run)
```

---

aggregate_raster_to_polygons

Aggregate a raster to polygons

Description

Aggregate raster values to polygons using an aggregation table

Usage

```
aggregate_raster_to_polygons(
  data_raster,
  aggregation_table,
  aggregation_cols = "polygon_id",
  method = "mean",
  aggregated_field = "data",
  z_dimension = NULL,
  z_dimension_name = "z",
  weighting_raster = NULL,
  na.rm = TRUE
)
```

Arguments

`data_raster`	terra::SpatRaster containing data to be aggregated to polygons.
`aggregation_table`	data.table::data.table Aggregation table linking pixels to polygon identifiers, created using `build_aggregation_table()`
`aggregation_cols`	(character vector, default 'polygon_id') Polygon identifiers to use for aggregation. Must be field names within `aggregation_table`.
`method`	(character, default 'mean') Aggregation method: one of 'mean', 'sum', 'weighted.mean', or 'weighted.sum'. The latter two methods require a `weighting_raster`.
`aggregated_field`	(character, default 'data') Name of the aggregated raster values in the output table.

`z_dimension`	(vector, default NULL) If passing a `data_raster` with multiple layers, how should each layer be identified? Should be a vector with length equal to the number of layers in `data_raster`. If left as `NULL`, the default, and `data_raster` has 1 layer, no z dimension will be added. If left as `NULL` and `data_raster` has more than 1 layer, will default to (1, 2, ..., N layers).
`z_dimension_name`	(default 'z') The field name for the "z" dimension corresponding to each layer of the `data_raster`. This field is only added if `z_dimension` is passed or if `data_raster` has more than one layer.
`weighting_raster`	(terra::SpatRaster, default NULL) The relative weighting of each whole pixel to the overall polygon value, for example, if calculating a population-weighted mean. Required for methods 'weighted.mean' and 'weighted.sum', ignored for the other methods.
`na.rm`	(bool, default TRUE) How to handle NA pixels in `data_raster` and `weighting_raster`. If set to TRUE but ALL pixels in a polygon are NA, will still return an NA value for the polygon.

Details

This is a more efficient and feature-rich alternative to terra's zonal statistics functions. Features include:

- Always fractionally aggregate, weighting by area of the pixel in a polygon
- Optionally weight by both area fraction and a weighting raster (e.g. population)
- Means or sums of raster values across polygons
- Optionally aggregate multiple years of raster data at once

Value

data.table containing polygon identifiers, (optionally) layer identifiers in the `z_dimension_name` column, and data values aggregated by polygon.

See Also

build_aggregation_table

Examples

```
## Not run:
  polygons <- sf::st_read(system.file('extdata/Benin_communes.gpkg', package = 'mbg'))
  id_raster <- build_id_raster(polygons)
  n_data_pixels <- sum(!is.na(terra::values(id_raster)))
  # Example ID raster filled with data
  # This is an example of pixel-level covariate data or model estimates
  data_raster <- mbg::values_to_raster(stats::rnorm(n_data_pixels), id_raster)
  # Build aggregation table, which can be used across many aggregations
  aggregation_table <- build_aggregation_table(
    polygons, id_raster, polygon_id_field = 'commune_code'
  )
  # Aggregate the raster to the polygons
   aggregated <- aggregate_raster_to_polygons(
     data_raster = data_raster,
     aggregation_table = aggregation_table,
     aggregation_cols = 'commune_code',
     method = 'mean'
   )
   head(aggregated)

## End(Not run)
```

`build_aggregation_table` *Build aggregation table*

Description

Build a table to quickly aggregate from pixels to polygons

Usage

```
build_aggregation_table(polygons, id_raster, polygon_id_field, verbose = FALSE)
```

Arguments

`polygons`	terra::SpatVector object. Should contain a unique ID field.
`id_raster`	terra::SpatRaster object. ID raster created by `build_id_raster()` for the polygons object. Should have the same CRS as polygons and completely cover it.
`polygon_id_field`	(`character(1)`) Unique identifier field in polygons.
`verbose`	(`logical(1)`, default FALSE) Show progress for building aggregation rows for each polygon?

Value

data.table with fields:

- polygon_id: Unique polygon identifier
- pixel_id: unique pixel ID from the ID raster
- masked_pixel_id: Index counting only non-NA pixels from the ID raster
- area_fraction: fraction of the pixel area falling within this polygon
- Merged fields from the table of polygons

See Also

calculate_pixel_fractions_single_polygon()

Examples

```
## Not run:
  polygons <- sf::st_read(system.file('extdata/Benin_communes.gpkg', package = 'mbg'))
  id_raster <- build_id_raster(polygons)
  aggregation_table <- build_aggregation_table(
    polygons, id_raster, polygon_id_field = 'commune_code'
  )

## End(Not run)
```

`build_id_raster` *Build ID raster*

Description

Build an ID raster matching the extent of a vector dataset

Usage

```
build_id_raster(polygons, template_raster = NULL)
```

Arguments

`polygons`	terra::SpatVector object. The polygons to be aggregated to
`template_raster`	(optional) terra::SpatRaster object. The template raster should contain and have the same CRS as the polygons. If template raster is `NULL`, the default, uses the default world template raster from `make_world_template_raster()`.

Details

The ID raster will be used to build the aggregation table. Each pixel has a unique integer value from 1 to the number of pixels in the ID raster.

Value

ID raster. A terra::SpatRaster object that minimally encloses the polygons

Examples

```
## Not run:
  polygons <- sf::st_read(system.file('extdata/Benin_communes.gpkg', package = 'mbg'))
  build_id_raster(polygons)

## End(Not run)
```

`calculate_pixel_fractions_single_polygon`
Calculate fractional pixels in a polygon

Description

Calculate the fraction of each pixel's area that falls within a single polygon

Usage

```
calculate_pixel_fractions_single_polygon(polygon, id_raster, polygon_id = NULL)
```

Arguments

`polygon`	terra::SpatVector object of length 1. The polygon to calculate fractional areas across.
`id_raster`	terra::SpatRaster object. ID raster created for the set of all polygons to be considered, created by `build_id_raster()`.
`polygon_id`	(optional). ID for this polygon. Must have length 1.

Details

This is a helper function called by `build_aggregation_table()`.

Value

data.table containing two or three columns:

- pixel_id: unique pixel ID from the ID raster
- area_fraction: fraction of the pixel area falling within this polygon
- polygon_id (optional): If `polygon_id` was defined, it is added to the table

See Also

build_aggregation_table

Examples

```
## Not run:
  polygons <- sf::st_read(system.file('extdata/Benin_communes.gpkg', package = 'mbg'))
  id_raster <- build_id_raster(polygons)
  pixel_fractions <- calculate_pixel_fractions_single_polygon(
    polygon = polygons[1, ], id_raster
  )
  head(pixel_fractions)

## End(Not run)
```

`dissolve_sf_by_attribute` *Dissolve sf object by attribute*

Description

Dissolve an SF object by attribute

Usage

```
dissolve_sf_by_attribute(x, by = character(0))
```

Arguments

`x`	(sf::sf object) SF object to dissolve
`by`	(`character(N)`, default character(0)) Attributes to dissolve by

Details

Inspired by spatialEco::sf_dissolve

Value

Dissolved sf::sf object

Examples

```
## Not run:
  communes_sf <- sf::st_read(system.file("extdata/Benin_communes.gpkg", package = "mbg"))
  departments_sf <- mbg::dissolve_sf_by_attribute(
    x = communes_sf,
    by = c('department', 'department_code')
  )

## End(Not run)
```

`fit_inla_model` *Fit INLA model*

Description

Fit an INLA model based on a constructed data stack and formula

Usage

```
fit_inla_model(
  formula,
  data_stack,
  spde,
  samplesize_vec = 1,
  precision_vec = 1,
  family = "binomial",
  link = "logit",
  fixed_effects_pc_prior = list(threshold = 3, prob_above = 0.05),
  verbose = TRUE
)
```

Arguments

`formula`	(character) INLA formula to fit. Generated in `prepare_inla_data_stack()`, then interpreted using `stats::as.formula()` within the call to `INLA::inla()`.
`data_stack`	Stacked data, covariates, and spatial index. Generated in `prepare_inla_data_stack()`.
`spde`	SPDE object generated by `prepare_inla_data_stack()`.
`samplesize_vec`	(`integer(N)`, default 1) Sample sizes for each data observation. Only used for binomial data models.
`precision_vec`	(`numeric(N)`, default 1) Precision for each data observation. Only used for gaussian data models.
`family`	(character, default 'binomial') GLM family to use. For more information, see `stats::family()`.
`link`	(character, default 'logit') Link function to use, typically related to the GLM `family`.
`fixed_effects_pc_prior`	A named list specifying the penalized complexity prior for all fixed effects except for the intercept. The two named items are "threshold", the test threshold for the size of each fixed effect, and "prob_above", the prior probability that the beta for each covariate will EXCEED that threshold.
`verbose`	(`logical(1)`, default TRUE) Log progress for INLA model fitting?

Details

Using `INLA::inla()` with reasonable defaults and settings tuned to predict across a grid.

Value

A fitted INLA model object created by `INLA::inla()`

See Also

MbgModelRunner

`generate_cell_draws_and_summarize`
Generate cell draws and summary rasters from INLA model

Description

Use INLA posteriors to predict out across a grid

Usage

```
generate_cell_draws_and_summarize(
  inla_model,
  inla_mesh,
  n_samples,
  id_raster,
  covariates,
  inverse_link_function,
  nugget_in_predict = TRUE,
  admin_boundaries = NULL,
  ui_width = 0.95,
  verbose = TRUE
)
```

Arguments

`inla_model`	Output from `fit_inla_model()`
`inla_mesh`	An SPDE mesh used to define the spatial integration points of the INLA geostatistical model. Typically created using `INLA::inla.mesh.2d()` or a similar function.
`n_samples`	(numeric) Number of posterior predictive samples to draw.
`id_raster`	(terra::SpatRaster) raster showing all cell locations where predictions should be taken.
`covariates`	(list) Named list of all covariate effects included in the model, typically generated by `load_covariates()`.
`inverse_link_function`	(character) If a link function was used in the INLA model, name of the R function to transform the predictive draws from link space to natural space. For example, in a logit-linked binomial model, pass 'plogis' (as a string is fine) to invert-logit the predictive draws.
`nugget_in_predict`	(`logical(1)`, default TRUE) Should the nugget term be used as an IID noise term applied to each pixel-draw?
`admin_boundaries`	(sf object, default NULL) The same admin boundaries used to create the admin-level effect, if one was defined in the model. Only used if an admin-level effect was defined in the model.

`ui_width`	(numeric, default 0.95) Size of the uncertainty interval width when calculating the upper and lower summary rasters
`verbose`	(`logical(1)`, default TRUE) Log progress for draw generation?

Details

Based on a fitted INLA model, the survey area defined in an ID raster, and a set of covariates, generate predictive grid cell draws and summary rasters across a study area.

Value

Named list containing at least the following items:

- "parameter_draws": posterior samples generated from `INLA::inla.posterior.sample()`
- "cell_draws": A matrix of grid cell draws. Each row represents a non-NA pixel in the `id_raster`, in the same order that would be pulled by `terra::values()`, and each column represents a different posterior draw.
- "cell_pred_mean": Mean predictive estimate by grid cell, formatted as a terra SpatRaster
- "cell_pred_lower": Lower bound of (X%) uncertainty interval, formatted as a terra SpatRaster
- "cell_pred_upper": Upper bound of (X%) uncertainty interval, formatted as a terra SpatRaster

---

`load_covariates` *Load covariates*

Description

Load covariates

Usage

```
load_covariates(
  directory,
  covariates_table,
  id_raster,
  year = NULL,
  file_format = "tif",
  add_intercept = FALSE,
  check_previous_years = 10
)
```

Arguments

`directory` Directory containing all covariate sub-directories

`covariates_table`
`data.frame` containing at least the following fields:

- ’covariate’: (character): Name of the covariate

- 'annual': (logical) Does the covariate vary by year? If so, look for the `year` in the name of the file.
- 'transform': (character) Name of a function to use to transform the covariate. Common options include 'identity' (no transformation), 'sqrt', 'abs', and 'log1p'
- 'normalize': (logical) Should the covariate be rescaled to have mean 0 and standard deviation 1 across all pixels in the study area? Generally should be set to TRUE for predictive covariates.

`id_raster`	terra::SpatRaster with non-NA pixels delineating the extent of the study area
`year`	(`numeric`, default NULL) Year of data to for time-varying covariates. If NULL, the default, uses the current year.
`file_format`	(`character`, default 'tif') File format for the raster covariate data. Used to search for the input file within the proper containing folder.
`add_intercept`	(`logical`, default FALSE) Should a covariate called "intercept", a raster object with 1s in all required cells, be placed at the start of the returned covariates list?
`check_previous_years`	(`integer` > 0, default 10) If annual data is not found in this year, how many previous years should be checked? If 0, will not check any previous years.

Details

Load a list of covariates from a specified directory structure

Value

A named list of formatted covariates. Each list item is a terra::SpatRaster with one layer and the same dimensions as the `id_raster`

`logging_get_timer_log` *Get timer log*

Description

Return a log of all timed events as a data.table

Usage

```
logging_get_timer_log(clear_log = FALSE, deindent = TRUE)
```

Arguments

`clear_log`	(`logical(1)`, default FALSE) Should the log be cleared afterwards?
`deindent`	(`logical(1)`, default TRUE) Should leading whitespace be removed from timer messages?

Examples

```
mbg::logging_start_timer(msg = 'Test logging')
Sys.sleep(0.1)
mbg::logging_stop_timer()
log_results <- mbg::logging_get_timer_log()
print(log_results)
```

`logging_start_timer` *Start logging timer*

Description

Start a nested timer with an optional message

Usage

```
logging_start_timer(msg, echo = TRUE, indentation_text = "  ")
```

Arguments

`msg`	(`character(1)`) Logging message
`echo`	(`logical(1)`, default TRUE) Should the message be written to screen?
`indentation_text`	(`character(1)`, default " ") Text that will be repeated at the beginning of the message for each layer of indentation

Examples

```
mbg::logging_start_timer(msg = 'Test logging')
Sys.sleep(0.1)
mbg::logging_stop_timer()
log_results <- mbg::logging_get_timer_log()
print(log_results)
```

`logging_stop_timer` *End logging timer*

Description

End a nested timer

Usage

```
logging_stop_timer(echo = TRUE)
```

Arguments

`echo`	(`logical(1)`, default = TRUE) Should the message be written to screen?

Examples

```
mbg::logging_start_timer(msg = 'Test logging')
Sys.sleep(0.1)
mbg::logging_stop_timer()
log_results <- mbg::logging_get_timer_log()
print(log_results)
```

`log_posterior_density` *Generate log posterior predictive density from a geostatistical surface onto point data*

Description

Generate log posterior predictive density from a geostatistical surface onto point data

Usage

```
log_posterior_density(draws, validation_data, id_raster, na.rm = FALSE)
```

Arguments

`draws` (`matrix`) A predictive draw matrix, where each row corresponds to a pixel in the `id_raster` and each column corresponds to one sampled estimate of the outcome.

`validation_data` (`data.frame`) Table containing at least the following fields:

- x (`numeric`) location x position, in the same projection as `id_raster`
- y (`numeric`) location y position, in the same projection as `id_raster`
- indicator (`integer`) The number of events in the population
- samplesize (`integer`) The total population, denominator for `indicator`

`id_raster` (terra::SpatRaster) Raster showing the sample study area, created using build_id_raster.

`na.rm` (`logical(1)`, default FALSE) Should NA values be omitted from the LPD calculation?

Details

Calculated across draws. Requires an ID raster to match each point observation to a set of draws. Assumes binomial data.

For examples, see `vignette('model-comparison', package = 'mbg')`

Value

(`numeric(1)`) Log predictive density of the validation data given the draw estimates.

`make_time_stamp` *Make time stamp*

Description

Create a string time stamp based on current detailed date/time

Usage

```
make_time_stamp(suffix = NULL, milliseconds = TRUE)
```

Arguments

`suffix`	(`character(1)`, default NULL) suffix to append to the time stamp. Useful when running batches of related models
`milliseconds`	(`logical(1)`, default TRUE) Should milliseconds be appended to the timestamp? Useful when launching many models in quick succession.

Value

A string formatted as `'YYYYMMDD_HH_MM_SS(_optional MS)(_optional suffix)'`

`make_world_template_raster` *Make world template raster*

Description

Create a template raster for the world with approximately 5x5km resolution at the equator, matching many common raster covariates for health.

Usage

```
make_world_template_raster()
```

Details

The raster has the following specifications:

- 4320 rows, 8640 columns
- Resolution: 0.04166667 decimal degrees, approx. 5km at the equator
- CRS: WGS 1984 unprojected latitude/longitude, `terra::crs('EPSG:4326')`
- Values: All NA. Used exclusively for creating a shapefile-specific ID raster

Value

terra::SpatRaster object matching the specifications above

`MbgModelRunner` *MBG model runner class*

Description

R6 class to run a full MBG model and make predictions.

Details

To see examples of this object, run `vignette('mbg')`

Public fields

`input_data` (data.table::data.table)
Table containing at least the following fields:

- x (`numeric`) location longitude in decimal degrees
- y (`numeric`) location latitude in decimal degrees
- indicator (`integer`) The number of events in the population
- samplesize (`integer`) The total population, denominator for `indicator`

`id_raster` (terra::SpatRaster)
raster showing the total area that will be predicted using this model.

`covariate_rasters (list())`
A list containing all predictor covariates. Each covariate is a terra::SpatRaster object with the same extent and dimensions as `id_raster`.

`aggregation_table` (data.table::data.table)
A table created by build_aggregation_table, used to link each grid cell to higher-level administrative units.

`aggregation_levels` (`list()`)
: A named list: for each named item, the name is the label for that aggregation level, and the value is a character vector of all fields in the original polygons to be used for aggregation at that level.

`population_raster` (terra::SpatRaster)
: A raster giving population for each grid cell, to be used for population-weighted aggregation from grid cells to polygon boundaries. Should have the same dimensions as `id_raster`. If no population raster is passed and the results are aggregated, aggregation will be by simple mean rather than population-weighted mean

`admin_bounds` (sf::sf)
: Polygons showing the boundaries of administrative divisions within the study region. Only required if `use_admin_effect` OR `stacking_use_admin_bounds` is `TRUE`.

`admin_bounds_id` (`character`)
: Field containing unique identifiers for `admin_bounds`, if passed.

`use_covariates` (`logical(1)`)
: Should covariate effects be included in the predictive model?

`use_gp` (`logical(1)`)
: Should a smoothed spatial surface be included in the predictive model?

`use_admin_effect` (`logical(1)`)
: Should IID administrative-level effects be included in the predictive model?

`use_nugget` (`logical(1)`)
: Should an IID effect by pixel be included in the predictive model?

`use_stacking` (`logical(1)`)
: Should machine learning submodels be trained to relate the covariate rasters with the outcome data? Only run if `use_covariates` is `TRUE`.

`stacking_model_settings` (`list()`)
: A named list of submodels to be run. For more information about this term, see run_regression_submodels. Only considered if `use_stacking` is TRUE.

`stacking_cv_settings` (`list()`)
: How should the stacking submodels be cross-validated? For more information about this term, see run_regression_submodels. Only considered if `use_stacking` is `TRUE`.

`stacking_use_admin_bounds` (`logical(1)`)
: Should admin boundaries be included as features in the stacking submodels? For more information about this term, see run_regression_submodels. Only considered if `use_stacking` is TRUE.

`stacking_prediction_range` (`logical(1)`)
: Range of possible predictions for the stacking submodels. For more information about this term, see run_regression_submodels. Only considered if `use_stacking` is TRUE.

`mesh_max_edge` (`numeric(2)` or NULL)
: Maximum size of the INLA SPDE mesh inside (1) and outside (2) of the modeled region. Only considered if `use_gp` is TRUE.

`mesh_cutoff` (`numeric(1)`)
: Minimum size of the INLA mesh, usually reached in data-dense areas. Only considered if `use_gp` is TRUE.

`spde_integrate_to_zero` (boolean(1))
: Should the 'volume' under the SPDE mesh integrate to zero? Only considered if `use_gp` is TRUE.

`prior_spde_range` (list())
: A named list specifying the penalized complexity prior for the SPDE range. The two named items are "threshold", the test threshold (set as a proportion of the overall mesh extent), and "prob_below", the prior probability that the value is BELOW that range threshold. The function automatically converts "threshold" from a proportion of the overall mesh extent into a distance. Only considered if `use_gp` is TRUE.

`prior_spde_sigma` (list())
: A named list specifying the penalized complexity prior for sigma (standard deviation) of the SPDE object. The two named items are "threshold", the test threshold for the standard deviation, and "prob_above", the prior probability that sigma will EXCEED that threshold. Only considered if `use_gp` is TRUE

`prior_nugget` (list())
: A named list specifying the penalized complexity prior for the nugget term. The two named items are "threshold", the test threshold for the nugget standard deviation, and "prob_above", the prior probability that the standard deviation will EXCEED that threshold. Only considered if `use_nugget` is TRUE.

`prior_admin_effect` (list())
: A named list specifying the penalized complexity prior for the admin-level IID term. The two named items are "threshold", the test threshold for the standard deviation of admin-level effects, and "prob_above", the prior probability that the standard deviation will EXCEED that threshold. Only considered if `use_admin_effect` is TRUE.

`prior_covariate_effect` (list())
: A named list specifying the penalized complexity prior for all covariate effects except for the intercept, if an intercept is included. The two named items are "threshold", the test threshold for the size of each fixed effect, and "prob_above", the prior probability that the beta for each covariate will EXCEED that threshold. Only considered if `use_covariates` is TRUE and `use_stacking` is FALSE.

`inla_link` (character(1))
: Link function for fitting the INLA model, typically related to the GLM `family`.

`inverse_link` (character(1))
: Inverse function of `inla_link`.

`inla_family` (character)
: GLM family to use. For more information, see stats::family.

`nugget_in_predict` (logical(1))
: If the nugget is used in model fitting, should it also be included as an IID effect by pixel in the model prediction step?

`verbose` Should model progress be timed?

`model_covariates` (list())
: A list of covariates to be included in the INLA model. Either equal to `covariate_rasters`, or ML model predictions for stacked generalization.

`inla_inputs_list` (list())
: List of model inputs yielded by prepare_inla_data_stack

`inla_fitted_model (list())`
: List of model outputs yielded by fit_inla_model

`grid_cell_predictions` List of predictive surfaces yielded by generate_cell_draws_and_summarize

`aggregated_predictions` List of predictions by administrative unit. Only created if `aggregation_table` and `aggregation_levels` are both defined.

Methods

Public methods:

- `MbgModelRunner$new()`
- `MbgModelRunner$prepare_covariates()`
- `MbgModelRunner$fit_mbg_model()`
- `MbgModelRunner$generate_predictions()`
- `MbgModelRunner$aggregate_predictions()`
- `MbgModelRunner$run_mbg_pipeline()`
- `MbgModelRunner$get_predictive_validity()`
- `MbgModelRunner$clone()`

Method `new()`**:** Create a new MbgModelRunner object

Usage:

```
MbgModelRunner$new(
  input_data,
  id_raster,
  covariate_rasters = NULL,
  aggregation_table = NULL,
  aggregation_levels = NULL,
  population_raster = NULL,
  admin_bounds = NULL,
  admin_bounds_id = NULL,
  use_covariates = TRUE,
  use_gp = TRUE,
  use_admin_effect = FALSE,
  use_nugget = TRUE,
  use_stacking = FALSE,
  stacking_cv_settings = list(method = "repeatedcv", number = 5, repeats = 5),
  stacking_model_settings = list(gbm = NULL, treebag = NULL, rf = NULL),
  stacking_use_admin_bounds = FALSE,
  stacking_prediction_range = NULL,
  mesh_max_edge = c(0.2, 5),
  mesh_cutoff = c(0.04),
  spde_integrate_to_zero = FALSE,
  prior_spde_range = list(threshold = 0.1, prob_below = 0.05),
  prior_spde_sigma = list(threshold = 3, prob_above = 0.05),
  prior_nugget = list(threshold = 3, prob_above = 0.05),
  prior_admin_effect = list(threshold = 3, prob_above = 0.05),
  prior_covariate_effect = list(threshold = 3, prob_above = 0.05),
  inla_link = "logit",
  inverse_link = "plogis",
  inla_family = "binomial",
  nugget_in_predict = TRUE,
  verbose = TRUE
)
```

Arguments:

`input_data` (data.table::data.table) Table containing at least the following fields:

- x (`numeric`) location x position, in the same projection as the `id_raster`
- y (`numeric`) location y position, in the same projection as the `id_raster`
- indicator (`integer`) The number of events in the population
- samplesize (`integer`) The total population, denominator for `indicator`

`id_raster` (terra::SpatRaster) raster showing the total area that will be predicted using this model

`covariate_rasters` (`list()`, default NULL) A list containing all predictor covariates. Each covariate is a terra::SpatRaster object with the same extent and dimensions as `id_raster`.

`aggregation_table` (data.table::data.table) A table created by build_aggregation_table, linking each grid cell to one or more polygons

`aggregation_levels` (`list()`) A named list: for each named item, the name is the label for that aggregation level, and the value is a character vector of all fields in the original polygons to be used for aggregation at that level.

`population_raster` (terra::SpatRaster) A raster giving population for each grid cell, to be used for population-weighted aggregation from grid cells to polygon boundaries. Should have the same dimensions as `id_raster`. If no population raster is passed and the results are aggregated, aggregation will be by simple mean rather than population-weighted mean

`admin_bounds` (sf::sf, default `NULL`) Polygons showing the boundaries of administrative divisions within the study region. Only required if `use_admin_effect` OR `stacking_use_admin_bounds` is `TRUE`.

`admin_bounds_id` (`character`, default `NULL`) Field containing unique identifiers for `admin_bounds`, if passed.

`use_covariates` (`logical(1)`, default TRUE) Should covariate effects be included in the predictive model?

`use_gp` (`logical(1)`, default TRUE) Should a smoothed spatial surface be included in the predictive model?

`use_admin_effect` (`logical(1)` default FALSE) Should IID administrative-level effects be included in the predictive model?

`use_nugget` (`logical(1)`, default TRUE) Should an IID effect by pixel be included in the predictive model?

`use_stacking` (`logical(1)`, default FALSE) Should machine learning submodels be trained to relate the covariate rasters with the outcome data? Only run if `use_covariates` is `TRUE`.

`stacking_cv_settings` (`list()`) How should the stacking submodels be cross-validated? For more information about this term, see run_regression_submodels. Only considered if `use_stacking` is `TRUE`.

`stacking_model_settings` (`list()`) A named list of submodels to be run. For more information about this term, see run_regression_submodels. Only considered if `use_stacking` is TRUE.

`stacking_use_admin_bounds` (`logical(1)`, default FALSE) Should admin boundaries be included as features in the stacking submodels? For more information about this term, see run_regression_submodels. Only considered if `use_stacking` is TRUE.

`stacking_prediction_range` (`numeric(2)`, default NULL) Range of possible predictions for the stacking submodels. For more information about this term, see run_regression_submodels. Only considered if `use_stacking` is TRUE.

`mesh_max_edge` (`numeric(2)`, default c(0.2, 5)) Maximum size of the INLA SPDE mesh inside (1) and outside (2) of the modeled region. Only considered if `use_gp` is TRUE.

`mesh_cutoff` (`numeric(1)`, default 0.04) Minimum size of the INLA mesh, usually reached in data-dense areas. Only considered if `use_gp` is TRUE.

`spde_integrate_to_zero` (`boolean(1)`, default FALSE) Should the 'volume' under the SPDE mesh integrate to zero? Only considered if `use_gp` is TRUE.

`prior_spde_range` (`list()`) A named list specifying the penalized complexity prior for the SPDE range. The two named items are "threshold", the test threshold (set as a proportion of the overall mesh extent), and "prob_below", the prior probability that the value is BELOW that range threshold. The function automatically converts "threshold" from a proportion of the overall mesh extent into a distance. Only considered if `use_gp` is TRUE.

`prior_spde_sigma` (`list()`) A named list specifying the penalized complexity prior for sigma (standard deviation) of the SPDE object. The two named items are "threshold", the test threshold for the standard deviation, and "prob_above", the prior probability that sigma will EXCEED that threshold. Only considered if `use_gp` is TRUE

`prior_nugget` (`list()`) A named list specifying the penalized complexity prior for the nugget term. The two named items are "threshold", the test threshold for the nugget standard deviation, and "prob_above", the prior probability that the standard deviation will EXCEED that threshold. Only considered if `use_nugget` is TRUE.

`prior_admin_effect` (`list()`) A named list specifying the penalized complexity prior for the admin-level IID term. The two named items are "threshold", the test threshold for the standard deviation of admin-level effects, and "prob_above", the prior probability that the standard deviation will EXCEED that threshold. Only considered if `use_admin_effect` is TRUE.

`prior_covariate_effect` (`list()`) A named list specifying the penalized complexity prior for all covariate effects except for the intercept, if an intercept is included. The two named items are "threshold", the test threshold for the size of each fixed effect, and "prob_above", the prior probability that the beta for each covariate will exceed that threshold. Only considered if `use_covariates` is TRUE and `use_stacking` is FALSE.

`inla_link` (`character(1)`, default 'logit') Link function for fitting the INLA model, typically related to the GLM `family`.

`inverse_link` (`character(1)`, default 'plogis') Inverse function of `inla_link`.

`inla_family` (`character(1)`, default 'binomial') GLM family to use. For more information, see `stats::family()`.

`nugget_in_predict` (`logical(1)`, default TRUE) If the nugget is used in model fitting, should it also be included as an IID effect by pixel in the model prediction step?

`verbose` (`logical(1)`, default TRUE) Should model progress be timed?

Method `prepare_covariates()`**:** Prepare covariates for MBG model fitting

Usage:

```
MbgModelRunner$prepare_covariates()
```

Method `fit_mbg_model()`**:** Fit MBG model

Usage:

```
MbgModelRunner$fit_mbg_model()
```

Method `generate_predictions()`**:** Generate predictions by grid cell

Usage:

```
MbgModelRunner$generate_predictions(n_samples = 1000, ui_width = 0.95)
```

Arguments:

`n_samples` (`integer(1)`, default 1000) Number of posterior predictive samples to generate from the fitted model object.

`ui_width` (`numeric(1)`, default 0.95) Uncertainty interval width. This method will create summary rasters for quantiles ((1 - ui_width)/2) and (1 - (1 - ui_width)/2).

Method `aggregate_predictions()`**:** Aggregate grid cell predictions

Usage:

```
MbgModelRunner$aggregate_predictions(ui_width = 0.95)
```

Arguments:

`ui_width` (`numeric(1)`, default 0.95) Uncertainty interval width. This method will create summary "upper" and "lower" fields rasters for quantiles ((1 - ui_width)/2) and (1 - (1 - ui_width)/2).

Returns: List with the same names as `self$aggregation_levels`, aggregating by the columns specified in `self$aggregation_levels`

Method `run_mbg_pipeline()`**:** Run a full MBG pipeline, including stacking, MBG model fitting, and prediction

Usage:

```
MbgModelRunner$run_mbg_pipeline(n_samples = 1000, ui_width = 0.95)
```

Arguments:

`n_samples` (`integer(1)`, default 1000) Number of posterior predictive samples to generate from the fitted model object.

`ui_width` (`numeric(1)`, default 0.95)

Method `get_predictive_validity()`**:** Get predictive validity metrics for the fitted model

Usage:

```
MbgModelRunner$get_predictive_validity(
  in_sample = TRUE,
  validation_data = NULL,
  na.rm = FALSE
)
```

Arguments:

`in_sample` (`logical(1)`, default TRUE) Compare model predictions to the data used to generate the model? If FALSE, does not return the WAIC, which is only useful for in-sample predictive validity.

`validation_data` (data.table::data.table, default NULL) Observed data to compare against. Expected for out-of-sample model validation. Table containing at least the following fields:

- x (`numeric`) location x position, in the same projection as the `id_raster`
- y (`numeric`) location y position, in the same projection as the `id_raster`
- indicator (`integer`) The number of events in the population
- samplesize (`integer`) The total population, denominator for `indicator`

`na.rm` (`logical(1)`, default FALSE) Should NA values be dropped from the RMSE and log predictive density calculations?

Details: Returns the point RMSE (compared against the mean estimates by pixel), log-posterior density (compared against the predictive draws), and the Watanabe-Aikake Information Criterion (WAIC, only returned for in-sample predictive validity).

Returns: data.table::data.table Containing the following fields:

- 'rmse': Root mean squared error when compared against the mean estimates by pixel. Lower RMSE is better.
- 'lpd': Log posterior predictive density when compared against pixel-level samples from the model. Higher LPD is better.
- 'waic' (in-sample only): Watanable-Aikake information criterion estimated by INLA. Lower WAIC is better.
  For clarity, these fields will have the suffix "_is" for in-sample models, and "_oos" for out-of-sample models.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
MbgModelRunner$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

prepare_inla_data_stack

Prepare data stack for INLA

Description

Prepare data stack for INLA

Usage

```
prepare_inla_data_stack(
  input_data,
  id_raster,
  covariates,
  use_covariates = TRUE,
  covariates_sum_to_one = FALSE,
  family = "binomial",
  use_spde = TRUE,
  spde_range_pc_prior = list(threshold = 0.1, prob_below = 0.05),
  spde_sigma_pc_prior = list(threshold = 3, prob_above = 0.05),
  spde_integrate_to_zero = TRUE,
  mesh_max_edge = c(0.2, 5),
  mesh_cutoff = 0.04,
  use_nugget = TRUE,
  nugget_pc_prior = list(threshold = 3, prob_above = 0.05),
  use_admin_effect = FALSE,
  admin_boundaries = NULL,
  admin_pc_prior = list(threshold = 3, prob_above = 0.05)
)
```

Arguments

`input_data`	A data.frame with at least the following columns: • 'indicator': number of "hits' per site, e.g. tested positive for malaria • 'samplesize': total population sampled at the site • 'x': x position, often longitude • 'y': y position, often latitude
`id_raster`	terra::SpatRaster with non-NA pixels delineating the extent of the study area
`covariates`	(list) Named list of all covariate effects included in the model, typically generated by `load_covariates()`. Only used if `use_covariates` is TRUE.
`use_covariates`	(boolean(1), default TRUE) Should covariate fixed effects be included in the model? If TRUE, includes fixed effects for all covariates in the `covariates` argument. If FALSE, includes only an intercept.

`covariates_sum_to_one`	(logical, default FALSE) Should the input covariates be constrained to sum to one? Usually FALSE when raw covariates are passed to the model, and TRUE if running an ensemble (stacking) model.
`family`	(`character(1)`, default 'binomial') Statistical family; used to link stacked covariates with outcomes
`use_spde`	(`boolean(1)`, default TRUE) Should an SPDE approximation of a Gaussian process be included in the model?
`spde_range_pc_prior`	(list) A named list specifying the penalized complexity prior for the SPDE range. The two named items are "threshold", the test threshold (set as a proportion of the overall mesh extent), and "prob_below", the prior probability that the value is BELOW that range threshold. The function automatically converts "threshold" from a proportion of the overall mesh extent into a distance.
`spde_sigma_pc_prior`	(list) A named list specifying the penalized complexity prior for sigma (standard deviation) of the SPDE object. The two named items are "threshold", the test threshold for the standard deviation, and "prob_above", the prior probability that sigma will EXCEED that threshold.
`spde_integrate_to_zero`	(`boolean(1)`, default TRUE) Should the 'volume' under the SPDE mesh integrate to zero?
`mesh_max_edge`	(`numeric(2)`, default c(0.2, 5)) Maximum size of the INLA SPDE mesh inside (1) and outside (2) of the modeled region.
`mesh_cutoff`	(`numeric(1)`, default 0.04) Minimum size of the INLA mesh, usually reached in data-dense areas.
`use_nugget`	(`boolean(1)`, default TRUE) Should a nugget (IID observation-level error or noise) term be included?
`nugget_pc_prior`	A named list specifying the penalized complexity prior for the nugget term. The two named items are "threshold", the test threshold for the nugget standard deviation, and "prob_above", the prior probability that the standard deviation will EXCEED that threshold. Only used if `use_nugget` is TRUE
`use_admin_effect`	(`boolean(1)`, default FALSE) Should an IID random effect by administrative unit be included?
`admin_boundaries`	(sf object, default NULL) admin boundaries spatial object. Only used if `use_admin_effect` is TRUE
`admin_pc_prior`	(list) A named list specifying the penalized complexity prior for the admin-level IID term. The two named items are "threshold", the test threshold for the standard deviation of admin-level effects, and "prob_above", the prior probability that the standard deviation will EXCEED that threshold. Only used if `use_admin_effect` is TRUE.

Details

Creates the formatted input data to be used by the INLA model. For more information about penalized complexity priors, see Daniel Simpson's paper on the subject: doi:10.1214/16STS576

Value

List containing the following items:

- "mesh": The mesh used to approximate the latent Gaussian process
- "spde": The SPDE object that will be used to fit the INLA model
- "inla_data_stack": The data stack to be passed to `INLA::inla()`
- "formula_string": The formula specification to be passed to `INLA::inla()`

See Also

`fit_inla_model()` MbgModelRunner

`rmse_raster_to_point` *Generate RMSE from an estimated raster surface and point data*

Description

Generate RMSE from an estimated raster surface and point data

Usage

```
rmse_raster_to_point(estimates, validation_data, outcome_field, na.rm = FALSE)
```

Arguments

`estimates`	(terra::SpatRaster) Raster surface containing point estimates. This could also be the mean surface of a Bayesian geostatistical model
`validation_data`	(`data.frame`) Table containing at least the following fields: • x (`numeric`) location x position, in the same projection as `estimates` • y (`numeric`) location y position, in the same projection as `estimates` • (Outcome field) See below
`outcome_field`	(`character(1)`) Column in `validation_data` containing the values that should be compared against the `estimates` raster surface.
`na.rm`	(`logical(1)`, default FALSE) Should NA values be dropped from the RMSE calculation?

Details

For examples, see `vignette('model-comparison', package = 'mbg')`

Value

A single number giving RMSE between the point data and estimates raster.

`run_regression_submodels` *Run regression sub-models*

Description

Wrapper to run many regression sub-models using the caret package

Usage

```
run_regression_submodels(
  input_data,
  id_raster,
  covariates,
  cv_settings,
  model_settings,
  family = "binomial",
  clamping = TRUE,
  use_admin_bounds = FALSE,
  admin_bounds = NULL,
  admin_bounds_id = "polygon_id",
  prediction_range = c(-Inf, Inf),
  verbose = TRUE
)
```

Arguments

`input_data`	A data.frame with at least the following columns: • 'indicator': number of "hits' per site, e.g. tested positive for malaria • 'samplesize': total population sampled at the site • 'x': x position, often longitude • 'y': y position, often latitude
`id_raster`	terra::SpatRaster with non-NA pixels delineating the extent of the study area
`covariates`	(list) Named list of all covariate effects included in the model, typically generated by `load_covariates()`.
`cv_settings`	Named list of cross-validation settings, passed to caret::trainControl.
`model_settings`	Named list where the name of each header corresponds to a model run in caret::train, and the arguments correspond to the model-specific settings for that model type.

`family`	(`character(1)`, default 'binomial') Statistical model family being evaluated. For Gaussian models, this function trains against the 'mean' field; for all other families, this function trains against the ratio of 'indicator':'samplesize'.
`clamping`	(`logical(1)`, default TRUE) Should the predictions of individual ML models be limited to the range observed in the data?
`use_admin_bounds`	(`logical(1)`, default FALSE) Use one-hot encoding of administrative boundaries as a candidate feature?
`admin_bounds`	(sf, default NULL) Administrative boundaries to use. Only considered if `use_admin_bounds` is TRUE.
`admin_bounds_id`	(`character`, default 'polygon_id') Field to use for administrative boundary one-hot encoding. Only considered if `use_admin_bounds` is TRUE.
`prediction_range`	(`numeric(2)`, default c(-Inf, Inf)) Prediction limits for the outcome range. Used when the predictions are in a limited range, for example, 0 to 1 or -1 to 1.
`verbose`	(`logical(1)`, default TRUE) Log progress for ML model fitting?

Value

List with two items:

- "models": A list containing summary objects for each regression model
- "predictions": Model predictions covering the entire id_raster

---

`summarize_draws` *Summarize draws*

Description

Helper function to summarize a matrix or data.frame of predictive draws

Usage

```
summarize_draws(
  draws,
  id_fields = NULL,
  draw_fields = NULL,
  ui_width = 0.95,
  na.rm = TRUE
)
```

Arguments

`draws`	A `matrix`, `data.frame`, or data.table::data.table of predictive draws.
`id_fields`	(default NULL) Only considered for data.frame-like `draws`. What identifier fields in the data should be kept in the summary table and not included among the draw fields?
`draw_fields`	(default NULL) Only considered for data.frame-like `draws`. What fields represent actual draws, as opposed to identifier fields or other metadata like population? If `NULL`, the default, automatically determines the draw fields as all columns not included in the `id_fields`.
`ui_width`	(`numeric`, default 0.95) Size of the uncertainty interval width when calculating the upper and lower summary rasters
`na.rm`	(`logical`, default TRUE) Should NA values be removed when calculating summaries across draws?

Value

A data.table::data.table containing at least the following fields:

- The `id_fields`, if passed
- "mean": Mean across predictive draws
- "lower": Lower bound of the (X%) uncertainty interval
- "upper": Upper bound of the (X%) uncertainty interval
- "ui_width": "upper" - "lower"

Examples

```
# Summarize a draws matrix
draws_matrix <- matrix(rnorm(200), nrow = 10)
summary_table_a <- summarize_draws(draws_matrix)
head(summary_table_a)

# Summarize a draws data.table with location IDs
draws_table <- matrix(c(1:10, rnorm(200)), nrow = 10) |>
  data.table::as.data.table() |>
  data.table::setnames(c('location_id', paste0('draw_', 1:20)))
summary_table_b <- summarize_draws(draws_table, id_fields = 'location_id')
head(summary_table_b)
```

`values_to_raster` *Insert values into a raster*

Description

Insert a vector or matrix of values into an ID spatRaster

Usage

```
values_to_raster(x, id_raster)
```

Arguments

`x`	Vector, matrix, data.frame, or data.table of values that will be inserted into the ID raster. The length of x must be exactly divisible by `sum(!is.na(terra::values(id_raster)))`. Data.frames are converted to matrices, and then matrices are converted to vectors using `as.matrix()` and `as.vector()` respectively before processing. For that reason, data.frames should only contain fields with values to be inserted (such as a data.frame of draws).
`id_raster`	ID raster showing the outline of the study area, created using `build_id_raster()`. Should have 1 layer.

Details

The length of the vector or matrix must be a multiple of the number of non-NA pixels in the ID raster. Values from the vector/matrix are then inserted into the non-NA pixels of the spatRaster.

Value

SpatRaster with the same outline as the ID raster and (# values / # non-NA pixels in the ID raster) layers.

See Also

`build_id_raster()`

Examples

```
# Example ID raster with 10 rows and 10 columns, and 99 valid pixels
example_id_raster <- terra::rast(matrix(c(seq_len(99), NA), nrow = 10))
# Inserting 99 values yields a spatRaster with 1 layer
mbg::values_to_raster(stats::rnorm(99), example_id_raster)
# Inserting 99 * 3 values yields a spatRaster with 3 layers
mbg::values_to_raster(seq_len(99 * 3), example_id_raster)
# Trying to insert values with length not divisible by 99 yields an error
try(mbg::values_to_raster(seq_len(100), example_id_raster))
```

`vif_covariate_select` *Run variance inflation factor (VIF) selection on input covariates*

Description

Run variance inflation factor (VIF) selection on input covariates

Usage

```
vif_covariate_select(dataset, vif_cutoff = 5)
```

Arguments

`dataset`	data.frame-like object with named columns containing all covariates to consider in the VIF analysis.
`vif_cutoff`	(`numeric(1)`) Cutoff for maximum variance inflation factor in `dataset`

Value

data.table listing each variable, VIF in most recent round, and whether the indicator should be included or not

Index