Package 'TemporalModelR'

June 30, 2026

Type Package

Title Temporally Explicit Species Distribution Modelling

Version 0.2.0

Description Increases the ease of implementing a temporally-explicit modeling methodology when building ecological niche and species distribution models. Provides functions to assist with three major steps of temporally-explicit models: (i) preprocessing species and environmental data and generating suitable background or pseudoabsence data, (ii) building a niche model and generating temporally-explicit predictions from that model, and (iii) model postprocessing to explore spatiotemporal trends in model predictions. Methodological and theoretical foundations are described in Ingenloff and Peterson (2021) <doi:10.1111/2041-210X.13564>, Franklin (2010, ISBN:9780521700023), Peterson et al. (2011, ISBN:9780691136882), Blonder (2018) <doi:10.1111/ecog.03187>, Senay et al. (2013) <doi:10.1371/journal.pone.0071218>, and Li and Zhang (2024) <doi:10.48550/arXiv.2404.05933>.

URL https://github.com/CJHughes926/TemporalModelR, https://cjhughes926.github.io/TemporalModelR/

BugReports https://github.com/CJHughes926/TemporalModelR/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

Imports deldir, graphics, grDevices, sf, stats, terra, tools, utils, exactextractr

Suggests fastcpd, ggplot2, hypervolume, knitr, mgcv, randomForest, RColorBrewer, rmarkdown, scatterpie

VignetteBuilder knitr

Language en-US

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.2

NeedsCompilation no

Author Connor Hughes [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3720-0837>),
Mariana Castaneda-Guzman [aut] (ORCID:
<https://orcid.org/0000-0001-6106-4284>),
Luis E. Escobar [aut] (ORCID: <https://orcid.org/0000-0001-5735-2750>)

Maintainer Connor Hughes <connorhughes926@gmail.com>

Repository CRAN

Date/Publication 2026-06-30 10:40:19 UTC

Contents

`analyze_temporal_patterns`
Analyze Temporal Patterns in Binary Raster Time Series

Description

Post-processing function that applies changepoint detection methods to identify temporal trends in habitat suitability across consecutive predictions. Classifies pixels as stable, increasing in quality, or decreasing in quality, and identifies time periods of significant change.

Usage

```
analyze_temporal_patterns(binary_stack, summary_raster, time_steps,
                          fastcpd_params = list(), output_dir = NULL,
                          n_tiles_x = 1, n_tiles_y = 1, alpha = 0.05,
                          spatial_autocorrelation = TRUE, verbose = TRUE,
                          estimate_time = TRUE, overwrite = FALSE)
```

Arguments

`binary_stack`	RasterStack, RasterBrick, or character. Stack of binary raster layers across time, or path to directory containing binary rasters. Typically from `summarize_raster_outputs`.
`summary_raster`	RasterLayer. Per-pixel proportion of time periods where a pixel is suitable. From `summarize_raster_outputs`.
`time_steps`	Integer vector. Time labels corresponding to raster layers (same length as number of layers).
`fastcpd_params`	List. Named list of parameters passed to fastcpd changepoint detection function. Default is empty list. Supports parameterization from `fastcpd_binomial`.
`output_dir`	Character. Optional. Output directory for pattern rasters. When `NULL` (default), rasters are written to temporary files and not saved persistently. Provide a path to write named output files to disk.
`n_tiles_x`	Integer. Number of tiles in the x direction for tiled processing. Default is `1`. Increase to reduce peak memory use for large rasters and prevent crashes.
`n_tiles_y`	Integer. Number of tiles in the y direction for tiled processing. Default is `1`. Increase to reduce peak memory use for large rasters and prevent crashes.
`alpha`	Numeric. Significance level for changepoint detection. Default is `0.05`.
`spatial_autocorrelation`	Logical. If `TRUE` (default), includes a neighbor variable in the changepoint analysis to account for spatial autocorrelation.
`verbose`	Logical. If `TRUE` (default), prints progress messages during processing.
`estimate_time`	Logical. If `TRUE` (default), estimates runtime from a sample of pixels before full processing begins. If `FALSE`, proceeds directly to processing without a time estimate.
`overwrite`	Logical. If `TRUE`, overwrites existing output files. If `FALSE` (default), existing files are skipped.

Details

Applies changepoint detection using fastcpd to identify significant temporal shifts in spatial suitability. Accounts for spatial and temporal autocorrelation when `spatial_autocorrelation = TRUE`. The `fastcpd_params` list allows customization of the changepoint detection algorithm.

Pattern classifications enable identification of expanding, contracting, or stable g-space distributions over time or site level assessments of directional change in suitability.

Classification assumes consecutive rasters. Time periods shorter than ~15 time steps may be too short to classify increases or decreases.

Value

Invisibly returns a list containing:

- `pattern`: `SpatRaster` classifying pixels as integer values 1-6, corresponding to "Never Suitable", "Always Suitable", "No Pattern", "Increasing Suitability", "Decreasing Suitability", or "Fluctuating".
- `time_decrease`: `SpatRaster` showing the time step of first significant decrease for pixels classified as decreasing.
- `time_increase`: `SpatRaster` showing the time step of first significant increase for pixels classified as increasing.

See Also

Post-processing: `summarize_raster_outputs`

External: `fastcpd`

Examples

```
con_file <- system.file("extdata/binary/consensus_stack.tif",
      package = "TemporalModelR")

frq_file <- system.file("extdata/binary/frequency_raster.tif",
      package = "TemporalModelR")

binary_stack   <- terra::rast(con_file)

summary_raster <- terra::rast(frq_file)

time_steps <- expand.grid(
  year    = 1:15,
  season  = "Spring",
  stringsAsFactors = FALSE
)

analyze_temporal_patterns(
  binary_stack   = binary_stack,
  summary_raster = summary_raster,
  time_steps     = time_steps,
  output_dir     = tempdir(),
  spatial_autocorrelation = FALSE,
  overwrite      = TRUE,
  estimate_time  = FALSE,
  verbose        = FALSE
)
```

`analyze_trends_by_spatial_unit` *Summarize Temporal Patterns and Trends by Spatial Unit*

Description

Aggregates temporal pattern classifications and change metrics across user-defined spatial units (e.g., states, counties, watersheds). Returns summary tables and optionally generates simple visualizations.

Usage

```
analyze_trends_by_spatial_unit(shapefile_path, name_field, binary_stack = NULL,
                               pattern_raster = NULL, time_decrease_raster = NULL,
                                  time_increase_raster = NULL, time_steps = NULL,
                                  output_dir = NULL, overwrite = FALSE,
                                  create_plot = TRUE, pie_scale = 0.15,
                                  verbose = TRUE)
```

Arguments

`shapefile_path`	Character, sf object, or sfc object. Path to a shapefile or directory containing one, an sf object, or an sfc geometry. Shapefiles spatial units will be used as the units for the data summary.
`name_field`	Character. Attribute field to use as spatial unit labels.
`binary_stack`	`SpatRaster` or character. Optional stack of binary prediction rasters across time. Required for per-unit habitat summaries and time series plots.
`pattern_raster`	`SpatRaster` or character. Optional pattern classification raster from `analyze_temporal_patterns`. Required for pattern composition summaries and scatterpie map.
`time_decrease_raster`	`SpatRaster` or character. Optional raster of first decrease time step from `analyze_temporal_patterns`.
`time_increase_raster`	`SpatRaster` or character. Optional raster of first increase time step from `analyze_temporal_patterns`.
`time_steps`	Vector. Time labels for layers in `binary_stack`. Required when `binary_stack` or change rasters are provided.
`output_dir`	Character or `NULL`. Directory for CSV outputs and plots. If `NULL`, results are returned in memory only and no files are written. Default is `NULL`.
`overwrite`	Logical. If `TRUE`, recomputes and overwrites existing CSVs. Default is `FALSE`.
`create_plot`	Logical. If `TRUE` (default), generates and saves plots.
`pie_scale`	Numeric in (0, 1]. Largest pie's radius as a fraction of the smaller map dimension. Default is `0.15`, meaning the biggest pie spans roughly 30% of the smaller map dimension (radius = 15%, so diameter = 30%). Smaller pies scale so their area is proportional to the unit's pixel count. The fraction is converted internally to coordinate units, so this argument works identically across CRSes (degrees, meters, etc.) without manual rescaling.

verbose — Logical. If TRUE (default), prints progress messages during processing. Includes details on raster extraction and per-spatial-unit summaries.

Details

Summarizes results from modeling and post-processing at the scale of specific spatial blocks to allow for a nuanced look at spatiotemporal patterns.

Value

Invisibly returns a list containing:

- overall_summary: Pattern composition per spatial unit (present when pattern_raster is supplied).
- timestep_summary: Suitable pixel counts per unit per time step (present when binary_stack is supplied).
- change_by_timestep: Gain and loss pixel counts per unit per time step (present when both change rasters are supplied).
- plots: Named list of recorded plot objects (present when create_plot = TRUE).

See Also

Post-processing: summarize_raster_outputs, analyze_temporal_patterns

Examples

```
con_file <- system.file("extdata/binary/consensus_stack.tif",
                        package = "TemporalModelR")

binary_stack <- terra::rast(con_file)

study_crs <- sf::st_crs(binary_stack)

zones_sf <- rbind(
  sf::st_sf(ZONE = "West",
            geometry = sf::st_sfc(sf::st_polygon(list(
              matrix(c(0, 0, 1500, 1500, 0,
                       0, 1500, 1500, 0, 0), ncol = 2)
            )), crs = study_crs)),
  sf::st_sf(ZONE = "East",
            geometry = sf::st_sfc(sf::st_polygon(list(
              matrix(c(1500, 1500, 3000, 3000, 1500,
                       0,    1500, 1500, 0,    0),    ncol = 2)
            )), crs = study_crs))
)

time_steps <- expand.grid(
  year             = 1:15,
  season           = "Spring",
  stringsAsFactors = FALSE
)
```

```
analyze_trends_by_spatial_unit(
  shapefile_path = zones_sf,
  name_field     = "ZONE",
  binary_stack   = binary_stack,
  time_steps     = time_steps,
  create_plot    = FALSE,
  verbose        = FALSE
)
```

---

build_temporal_gam *Build Temporal GAM Models Across Cross-Validation Folds*

Description

Modeling function that constructs binomial generalized additive models (GAMs) for each cross-validation fold using presence and pseudoabsence data. Each model reserves one fold as testing data and uses the remaining folds as training data. The user supplies the model formula directly using standard **mgcv** formula syntax, including smooth terms such as `s()`, `te()`, and `ti()`. Supports automatic or manual probability thresholding for converting continuous predictions to binary suitability classifications necessary for downstream analyses. The returned object follows the same structure as `build_temporal_glm`, `build_temporal_hv`, and `build_temporal_rf`, and is accepted directly by `generate_spatiotemporal_predictions`.

Usage

```
build_temporal_gam(partition_result, pseudoabsence_result, model_formula,
                   link = "logit", gam_params = list(method = "REML"),
                   threshold_method  = "tss",
                   output_dir = file.path(tempdir(), "GAM_Models"),
                   create_plot = TRUE, plot_palette = "Dark 2",
                   overwrite = FALSE, time_cols = NULL, verbose = TRUE)
```

Arguments

`partition_result`
: List or character. Output from `spatiotemporal_partition` or path to an `.rds` file containing that output.

`pseudoabsence_result`
: List or character. Output from `generate_absences` or path to an `.rds` file containing that output.

`model_formula`
: Formula or character. The right-hand side of the model formula supplied as either a formula object or a character string. The response variable (`presence`) is always added automatically on the left-hand side, so only the right-hand side needs to be provided. Both of the following are accepted and equivalent:

- `~ s(Var1) + s(Var2) + Var3`
- `"~ s(Var1) + s(Var2) + Var3"`

Standard **mgcv** formula syntax applies. Smooth terms are specified with `s()` for univariate smooths, `te()` for tensor product smooths of two or more variables, and `ti()` for tensor product interaction terms. Parametric terms can be included alongside smooth terms using `+`. The basis type and dimension can be controlled via arguments to `s()`, e.g. `s(Var1, k = 5, bs = "tp")`. All predictor names referenced in the formula must be present as columns in both the presence and pseudoabsence data.

`link` Character. The link function for the binomial GAM. One of `"logit"` (default), `"probit"`, `"cloglog"`, or `"cauchit"`. See `binomial` for details on each link function.

`gam_params` Named list. Additional arguments passed to `gam`, such as `method` for the smoothing parameter estimation method (e.g. `"REML"`) or `select` for additional shrinkage. Default is `list(method = "REML")`.

`threshold_method`
Character or numeric. Method used to convert continuous predicted probabilities to binary suitability. Accepted values:

- `"prevalence"`: Sets threshold equal to the prevalence (proportion of presences) in the training data for that fold.
- `"tss"`: Selects the threshold that maximizes the True Skill Statistic (sensitivity + specificity - 1) on the training data. Default.
- A numeric value between 0 and 1 (e.g. `0.4`): Uses that value as a fixed threshold for all folds directly.

`output_dir` Character. Directory to write output files including saved model objects and plots. Default is `file.path(tempdir(), "GAM_Models")`.

`create_plot` Logical. If `TRUE`, generates per-fold response curve plots and a combined ROC curve summary. Default is `TRUE`.

`plot_palette` Character. Name of an HCL or RColorBrewer palette used to color folds in diagnostic plots. Accepts any HCL palette name (see `hcl.pals`) or, if **RColorBrewer** is installed, any Brewer palette name. Default is `"Dark 2"`.

`overwrite` Logical. If `TRUE`, overwrites existing saved model files. If `FALSE`, loads existing files when available. Default is `FALSE`.

`time_cols` Character. Name of the column(s) containing year or time step values in the occurrence data. Must match `time_cols` used in `spatiotemporal_partition`. Default is `NULL`.

`verbose` Logical. If `TRUE` (default), prints progress messages during processing. Includes per-fold training summaries and file-saved messages. The completion summary and metrics table are always printed regardless of this setting.

Details

GAMs are fit using `gam` from the **mgcv** package with `family = binomial(link = link)`. Smooth terms default to thin plate regression splines (`bs = "tp"`) with the basis dimension `k` chosen automatically by **mgcv** unless specified in the formula. Smoothing parameters are estimated by REML by default.

The returned object is recognized by `generate_spatiotemporal_predictions`, which uses the `model_type` field to use the correct prediction and evaluation logic.

Value

A list with class "TemporalGAM" containing:

- models: Named list of fitted gam objects, one per fold.
- thresholds: Named numeric vector of probability thresholds used for binary classification, one per fold.
- threshold_method: Character string recording the thresholding method used.
- model_formula: The formula object as passed to the fitting function.
- link: Character string recording the link function used.
- model_vars: Character vector of predictor names extracted from the formula right-hand side.
- fold_training_data: Named list of training data frames used to fit each fold model, retained for downstream prediction.
- fold_test_metrics: Data frame of held-out test fold metrics per fold: Threshold, AUC, TSS, Kappa, Sensitivity, and Specificity. Also written to Fold_Test_Metrics.csv in output_dir.
- output_dir: Path to the output directory.
- model_type: Character string "gam", used by generate_spatiotemporal_predictions.
- plots: Named list of recorded plot objects when create_plot = TRUE. Plots can be replayed with grDevices::replayPlot().

See Also

Preprocessing: spatiotemporal_partition, generate_absences

Modeling: build_temporal_glm, build_temporal_rf, build_temporal_hv, generate_spatiotemporal_predictions

External: gam, s

Examples

```
data(tmr_partition, package = "TemporalModelR")

data(tmr_absences,  package = "TemporalModelR")

build_temporal_gam(
  partition_result     = tmr_partition,
  pseudoabsence_result = tmr_absences,
  model_formula        = ~ s(elevation) + s(forest_cover) + s(prseas),
  threshold_method     = "tss",
  output_dir           = tempdir(),
  create_plot          = FALSE,
  time_cols            = c("year", "season"),
  verbose              = FALSE
)
```

build_temporal_glm *Build Temporal GLM Models Across Cross-Validation Folds*

Description

Modeling function that constructs binomial generalized linear models (GLMs) for each cross-validation fold using presence and pseudoabsence data. Each model reserves one fold as testing data and uses the remaining folds as training data. The user supplies the model formula directly, giving full control over predictor terms, polynomials, and interactions. The link function can be set to logit, probit, complementary log-log, or cauchit. Supports automatic or manual probability thresholding for converting continuous predictions to binary suitability classifications necessary for downstream analyses. The returned object follows the same structure as build_temporal_hv, build_temporal_gam, and build_temporal_rf, and is accepted directly by generate_spatiotemporal_predictions.

Usage

```
build_temporal_glm(partition_result, pseudoabsence_result, model_formula,
          link = "logit", threshold_method = "tss",
          output_dir = file.path(tempdir(), "GLM_Models"),
          create_plot = TRUE, plot_palette = "Dark 2", overwrite = FALSE,
          time_cols = NULL, verbose = TRUE)
```

Arguments

partition_result
: List or character. Output from spatiotemporal_partition or path to an .rds file containing that output.

pseudoabsence_result
: List or character. Output from generate_absences or path to an .rds file containing that output.

model_formula
: Formula or character. The right-hand side of the model formula supplied as either a formula object or a character string. The response variable (presence) is always added automatically on the left-hand side, so only the right-hand side needs to be provided. Both of the following are accepted and equivalent:

  - ~ Var1 + Var2 + I(Var1^2)
  - "~ Var1 + Var2 + I(Var1^2)"

  Standard R formula syntax applies: + for additive terms, * for main effects plus interaction, : for interaction only, I() for arithmetic transformations, poly() for orthogonal polynomials, log(), sqrt(), and any other base R function that can appear in a formula. All predictor names referenced in the formula must be present as columns in both the presence and pseudoabsence data.

link
: Character. The link function for the binomial GLM. One of "logit" (default), "probit", "cloglog", or "cauchit". See binomial for details on each link function.

threshold_method
: Character or numeric. Method used to convert continuous predicted probabilities to binary suitability. Accepted values:
  - "prevalence": Sets threshold equal to the prevalence (proportion of presences) in the training data for that fold.
  - "tss": Selects the threshold that maximizes the True Skill Statistic (sensitivity + specificity - 1) on the training data. Default.
  - A numeric value between 0 and 1 (e.g. 0.4): Uses that value as a fixed threshold for all folds directly.

output_dir
: Character. Directory to write output files including saved model objects and plots. Default is file.path(tempdir(), "GLM_Models").

create_plot
: Logical. If TRUE, generates per-fold response curve plots and a combined ROC curve summary. Default is TRUE.

plot_palette
: Character. Name of an HCL or RColorBrewer palette used to color folds in diagnostic plots. Accepts any HCL palette name (see hcl.pals) or, if **RColorBrewer** is installed, any Brewer palette name. Default is "Dark 2".

overwrite
: Logical. If TRUE, overwrites existing saved model files. If FALSE, loads existing files when available. Default is FALSE.

time_cols
: Character. Name of the column(s) containing year or time step values in the occurrence data. Must match time_cols used in spatiotemporal_partition. Default is NULL.

verbose
: Logical. If TRUE (default), prints progress messages during processing. Includes per-fold training summaries and file-saved messages. The completion summary and metrics table are always printed regardless of this setting.

Details

The model_formula argument accepts any standard R formula right-hand side. The response (presence) is prepended automatically. All R formula operators are valid, including I(), poly(), log(), sqrt(), :, and *. Variable names must match column names in the data exactly. Predictor names for response curve plots are extracted via all.vars(), which correctly unwraps terms such as I(Var1^2) to the base variable Var1.

All models are fit as stats::glm(..., family = binomial(link = link)). Predicted values are always probabilities on the 0-1 scale.

The returned object is recognized by generate_spatiotemporal_predictions, which uses the model_type field to use the correct prediction and evaluation logic.

Value

A list with class "TemporalGLM" containing:

- models: Named list of fitted glm objects, one per fold.
- thresholds: Named numeric vector of probability thresholds used for binary classification, one per fold.
- threshold_method: Character string recording the thresholding method used.
- model_formula: The formula object as passed to the fitting function.

- `link`: Character string recording the link function used.
- `model_vars`: Character vector of predictor names extracted from the formula right-hand side.
- `fold_training_data`: Named list of training data frames used to fit each fold model, retained for downstream prediction.
- `fold_test_metrics`: Data frame of held-out test fold metrics per fold: `Threshold`, `AUC`, `TSS`, `Kappa`, `Sensitivity`, and `Specificity`. Also written to `Fold_Test_Metrics.csv` in `output_dir`.
- `output_dir`: Path to the output directory.
- `model_type`: Character string `"glm"`, used by `generate_spatiotemporal_predictions`.
- `plots`: Named list of recorded plot objects when `create_plot = TRUE`. Plots can be replayed with `grDevices::replayPlot()`.

See Also

Preprocessing: `spatiotemporal_partition`, `generate_absences`

Modeling: `build_temporal_gam`, `build_temporal_rf`, `build_temporal_hv`, `generate_spatiotemporal_predictions`

Examples

```
data(tmr_partition, package = "TemporalModelR")

data(tmr_absences,  package = "TemporalModelR")

build_temporal_glm(
  partition_result     = tmr_partition,
  pseudoabsence_result = tmr_absences,
  model_formula        = ~ elevation + forest_cover + prseas,
  threshold_method     = "tss",
  output_dir           = tempdir(),
  create_plot          = FALSE,
  time_cols            = c("year", "season"),
  verbose              = FALSE
)
```

`build_temporal_hv` *Build Temporal Hypervolume Models Across Cross-Validation Folds*

Description

Modeling function that constructs hypervolume models for each cross-validation fold using either Gaussian kernel density estimation or one-class SVM. Each hypervolume reserves one fold as testing data and uses the remaining folds as training data. The returned object follows the same structure as `build_temporal_glm`, `build_temporal_gam`, and `build_temporal_rf`, and is accepted directly by `generate_spatiotemporal_predictions`.

Usage

```
build_temporal_hv(partition_result, model_vars, method,
                  hypervolume_params = list(),
                  output_dir = file.path(tempdir(), "Hypervolume_Models"),
                  create_plot = TRUE, overwrite = FALSE,
                  plot_palette = "Dark 2", verbose = TRUE)
```

Arguments

partition_result	List or character. Output from spatiotemporal_partition or path to an .rds file containing that output.
model_vars	Character vector. Names of predictor columns to use in hypervolume construction. All variables must be present as columns in the occurrence data produced by temporally_explicit_extraction.
method	Character. Hypervolume method. One of "gaussian" for Gaussian kernel density estimation or "svm" for one-class support vector machine. Required.
hypervolume_params	Named list. Additional parameters passed to hypervolume_gaussian or hypervolume_svm. For Gaussian models, valid keys are kde.bandwidth, quantile.requested, quantile.requested.type, chunk.size, verbose, and samples.per.point. For SVM models, valid keys are svm.nu, svm.gamma, chunk.size, verbose, and samples.per.point. Default is an empty list, which uses the built-in defaults.
output_dir	Character. Directory to write output files including saved hypervolume objects and plots. Default is file.path(tempdir(), "Hypervolume_Models").
create_plot	Logical. If TRUE, generates pairplot visualisations of each fold's hypervolume and a combined comparison plot. Default is TRUE.
overwrite	Logical. If TRUE, overwrites existing saved hypervolume files. If FALSE, loads existing files when available. Default is FALSE.
plot_palette	Character. Name of an HCL or RColorBrewer palette used to color folds in diagnostic plots. Accepts any HCL palette name (see hcl.pals) or, if **RColorBrewer** is installed, any Brewer palette name. Default is "Dark 2".
verbose	Logical. If TRUE (default), prints progress messages during processing. Includes per-fold hypervolume construction progress, volume summaries, and overlap statistics.

Details

For N folds, constructs N hypervolumes where each fold reserves one group of points for testing and uses the remaining N-1 groups for training. Pairwise overlap statistics quantify similarity between hypervolumes across folds, with low overlap indicating that the environmental space sampled differs substantially across folds.

Hypervolume objects are saved as a combined .rds file in output_dir. If the file already exists and overwrite = FALSE, the saved file is loaded rather than re-fitting.

The returned object is accepted directly by generate_spatiotemporal_predictions, which uses the model_type field to dispatch hypervolume-specific projection logic via hypervolume_project.

Value

A list with class "TemporalHypervolume" containing:

- models: Named list of fitted Hypervolume objects, one per fold, named fold1, fold2, etc. This naming convention matches the other build_temporal_*() functions and is required by generate_spatiotemporal_predictions.
- volumes: Named numeric vector of hypervolume sizes (units depend on the number of dimensions and bandwidth).
- overlaps: Named list of pairwise percent volume overlaps between all fold combinations.
- method: Character string recording the method used.
- model_vars: Character vector of predictor names used.
- fold_training_data: Named list of training data frames used to fit each fold model, retained for consistency with the other model types and for downstream use.
- fold_test_metrics: Data frame of E-space inclusion metrics computed on the held-out test points for each fold. Columns: fold, n_test, volume, tp, fn, sensitivity, Sensitivity. Printed as a summary table at the end of model building.
- output_dir: Path to the output directory.
- model_type: Character string "hypervolume", used by generate_spatiotemporal_predictions.
- plots: List of recorded plot objects (if create_plot = TRUE). Plots can be replayed with grDevices::replayPlot().

See Also

Preprocessing: spatiotemporal_partition

Modeling: build_temporal_glm, build_temporal_gam, build_temporal_rf, generate_spatiotemporal_predictions

External: hypervolume_gaussian, hypervolume_svm

Examples

```
data(tmr_partition_small, package = "TemporalModelR")

build_temporal_hv(
  partition_result = tmr_partition_small,
  model_vars       = c("elevation", "forest_cover"),
  method           = "svm",
  output_dir       = tempdir(),
  create_plot      = FALSE,
  verbose          = FALSE
)
```

build_temporal_rf *Build Temporal Random Forest Models Across Cross-Validation Folds*

Description

Modeling function that constructs Random Forest classification models for each cross-validation fold using presence and pseudoabsence data. Each model reserves one fold as testing data and uses the remaining folds as training data. The user specifies predictors as a character vector. Predicted probabilities of presence are extracted from the out-of-bag or in-bag vote fractions and thresholded to produce binary suitability classifications. Variable importance is recorded for each fold. The returned object follows the same structure as `build_temporal_glm`, `build_temporal_gam`, and `build_temporal_hv`, and is accepted directly by `generate_spatiotemporal_predictions`.

Usage

```
build_temporal_rf(partition_result, pseudoabsence_result, model_vars,
                  rf_params = list(), threshold_method = "tss",
                  output_dir = file.path(tempdir(), "RF_Models"),
                  create_plot = TRUE, plot_palette = "Dark 2",
                  overwrite = FALSE, time_cols = NULL, verbose = TRUE)
```

Arguments

`partition_result`
: List or character. Output from `spatiotemporal_partition` or path to an `.rds` file containing that output.

`pseudoabsence_result`
: List or character. Output from `generate_absences` or path to an `.rds` file containing that output.

`model_vars`
: Character vector. Names of predictor columns to include in the Random Forest. All variables must be present as columns in both the presence and pseudoabsence data.

`rf_params`
: Named list. Additional arguments passed to `randomForest`, such as `ntree` (number of trees, default 500), `mtry` (number of variables tried at each split, default `floor(sqrt(length(model_vars)))`), and `nodesize` (minimum node size, default 1 for classification). Default is an empty list, which uses **randomForest** defaults.

`threshold_method`
: Character or numeric. Method used to convert continuous predicted probabilities to binary suitability. Accepted values:

- `"prevalence"`: Sets threshold equal to the prevalence (proportion of presences) in the training data for that fold.
- `"tss"`: Selects the threshold that maximizes the True Skill Statistic (sensitivity + specificity - 1) on the training data. Default.
- A numeric value between 0 and 1 (e.g. `0.4`): Uses that value as a fixed threshold for all folds directly.

output_dir	Character. Directory to write output files including saved model objects and plots. Default is file.path(tempdir(), "RF_Models").
create_plot	Logical. If TRUE, generates a per-fold variable importance plot, partial dependence curves for each predictor, and a combined ROC curve summary. Default is TRUE.
plot_palette	Character. Name of an HCL or RColorBrewer palette used to color folds in diagnostic plots. Accepts any HCL palette name (see hcl.pals) or, if **RColorBrewer** is installed, any Brewer palette name. Default is "Dark 2".
overwrite	Logical. If TRUE, overwrites existing saved model files. If FALSE, loads existing files when available. Default is FALSE.
time_cols	Character. Name of the column(s) containing year or time step values in the occurrence data. Must match time_cols used in spatiotemporal_partition. Default is NULL.
verbose	Logical. If TRUE (default), prints progress messages during processing. Includes per-fold training summaries and file-saved messages.

Details

Random Forests are fit using randomForest from the **randomForest** package. The response is treated as a factor (0/1) so the model runs in classification mode, which produces class vote fractions used as predicted probabilities. Importance is computed with importance = TRUE and type = 1 (mean decrease in accuracy).

Predicted probabilities are the vote fraction for class 1 from predict(..., type = "prob")[, "1"]. These are used for threshold selection and ROC curve construction.

Diagnostic plots include: a variable importance bar chart (mean decrease in accuracy across folds), partial dependence curves for each predictor showing the marginal effect of each variable while averaging over all others (with rug marks for presences and pseudoabsences), and a combined ROC curve panel.

The returned object is recognized by generate_spatiotemporal_predictions, which uses the model_type field to use the correct prediction and evaluation logic.

Value

A list with class "TemporalRF" containing:

- models: Named list of fitted randomForest objects, one per fold.
- thresholds: Named numeric vector of probability thresholds used for binary classification, one per fold.
- threshold_method: Character string recording the thresholding method used.
- model_vars: Character vector of predictor names used.
- variable_importance: Named list of importance data frames, one per fold, with mean decrease in accuracy for each predictor.
- fold_training_data: Named list of training data frames used to fit each fold model, retained for downstream prediction.

- `fold_test_metrics`: Data frame of held-out test fold metrics per fold: `Threshold`, `AUC`, `TSS`, `Kappa`, `Sensitivity`, and `Specificity`. Also written to `Fold_Test_Metrics.csv` in `output_dir`.
- `output_dir`: Path to the output directory.
- `model_type`: Character string `"rf"`, used by `generate_spatiotemporal_predictions`.
- `plots`: Named list of recorded plot objects when `create_plot = TRUE`. Plots can be replayed with `grDevices::replayPlot()`.

See Also

Preprocessing: `spatiotemporal_partition`, `generate_absences`

Modeling: `build_temporal_glm`, `build_temporal_gam`, `build_temporal_hv`, `generate_spatiotemporal_predictions`

External: `randomForest`

Examples

```
data(tmr_partition, package = "TemporalModelR")

data(tmr_absences,  package = "TemporalModelR")

build_temporal_rf(
  partition_result     = tmr_partition,
  pseudoabsence_result = tmr_absences,
  model_vars           = c("elevation", "forest_cover", "prseas"),
  rf_params            = list(ntree = 100),
  threshold_method     = "tss",
  output_dir           = tempdir(),
  create_plot          = FALSE,
  time_cols            = c("year", "season"),
  verbose              = FALSE
)
```

`extdata` *Bundled rasters, point files, and prediction outputs*

Description

Several non-R objects ship in `inst/extdata/` for use in function examples and the package vignette. They cannot be portably serialized as `.rda` files, so they are stored as GeoTIFF and CSV files and loaded via `system.file`.

Details

`extdata/rasters_raw/` Raw synthetic environmental rasters before alignment. Includes `elevation.tif`, `forest_cover_1.tif` through `forest_cover_15.tif`, `pr_ann_1.tif` through `pr_ann_15.tif`, and `prseas_<year>_<season>.tif` for each of 15 years and 4 seasons (60 files).

`extdata/rasters_aligned/` Same rasters after `raster_align` has reprojected and masked them to the reference grid.

`extdata/rasters_scaled/` Aligned rasters z-scored using the scaling parameters from the seasonal extraction. Used by examples and modeling functions that work with the seasonal predictor set.

`extdata/rasters_scaled_annual/` Aligned rasters z-scored using the scaling parameters from the annual extraction. Used by examples and modeling functions that work with the annual predictor set (`pr_ann` instead of `prseas`).

`extdata/points/synthetic_occurrence_points.csv` The raw synthetic presence dataset: 150 points with `x`, `y`, `year`, `season`, and `pres = 1`.

`extdata/points/synthetic_occurrence_points.shp` Same points as a shapefile.

`extdata/points/extracted_seasonal_*.csv` Outputs from `temporally_explicit_extraction` using the seasonal predictor set: `_Raw_Values.csv`, `_Scaled_Values.csv`, and `_Scaling_Parameters.csv`.

`extdata/points/extracted_annual_*.csv` Same outputs but using the annual predictor set.

`extdata/predictions/` Per-timestep fold-vote rasters from the seasonal workflow's `generate_spatiotemporal_predict` call. Fifteen files (`Prediction_<year>_Spring.tif`) suitable for direct input to `summarize_raster_outputs`.

`extdata/binary/consensus_stack.tif` Multi-layer GeoTIFF of binary suitable / unsuitable rasters produced by `summarize_raster_outputs` with `consensus = 3`. Fifteen layers, one per year, ordered 1 through 15.

`extdata/binary/frequency_raster.tif` Companion single-layer raster giving the proportion of years each pixel was classified as suitable.

`extdata/precomputed/` Precomputed prediction outputs read directly by the modeling vignettes (V3a-V3d) so that `generate_spatiotemporal_predictions` does not need to be rerun at vignette build time. One subdirectory per model type (`glm/`, `gam/`, `rf/`, `hv/`), each containing `preds.rds` and a `pred_tifs/` folder. `preds.rds` is the list returned by `generate_spatiotemporal_predictions` for that model (`$timestep_metrics`, `$overall_summary`, `$prediction_files`, and `$model_type`), with `$prediction_files` reduced to bare file names rather than absolute build-time paths. `pred_tifs/` holds the 60 per-timestep fold-vote rasters (`Prediction_<year>_<season>.tif`, 15 years x 4 seasons) projected from that model. The vignettes load `preds.rds` and rebuild `$prediction_files` from `pred_tifs/` via `system.file`.

Example load patterns:

```
### Aligned raster directory
aln_dir <- system.file("extdata/rasters_aligned",
                       package = "TemporalModelR")

### Consensus stack (multi-layer)
binary_stack <- terra::rast(system.file(
  "extdata/binary/consensus_stack.tif", package = "TemporalModelR"
))

### Frequency raster
frequency_rast <- terra::rast(system.file(
  "extdata/binary/frequency_raster.tif", package = "TemporalModelR"
))
```

```
### Per-timestep prediction directory
pred_dir <- system.file("extdata/predictions",
                        package = "TemporalModelR")

### Precomputed GLM prediction object and its rasters
glm_preds <- readRDS(system.file(
  "extdata/precomputed/glm/preds.rds", package = "TemporalModelR"
))
glm_pred_files <- list.files(
  system.file("extdata/precomputed/glm/pred_tifs",
              package = "TemporalModelR"),
  pattern = "\.tif$", full.names = TRUE
)
```

`generate_absences` *Generate Temporally Explicit Pseudoabsence Points*

Description

Generates pseudoabsence or background points for each fold produced by `spatiotemporal_partition`, distributed across time steps proportionally to the number of presence points in each time step within each fold. Three generation methods are supported: random sampling within the study area, buffer-constrained sampling around presence points, and environmentally biased sampling that targets areas outside the known environmental tolerance of the species.

Usage

```
generate_absences(partition_result, reference_shapefile_path, raster_dir,
                  variable_patterns, method = "random", ratio = 1,
                  buffer_distance = NULL, env_percentile = 0.05,
                  time_cols = NULL, pseudoabsence_times = NULL,
                  min_points_per_timestep = 1, create_plot = TRUE,
                  plot_by_fold = FALSE, plot_palette = "Dark 2",
                  output_file = NULL, verbose = TRUE)
```

Arguments

`partition_result`
: List or character. Output from `spatiotemporal_partition` or path to an `.rds` file containing that output.

`reference_shapefile_path`
: Character or sf object. Path to a polygon file or an `sf` polygon object defining the study area.

`raster_dir`
: Character. Directory containing environmental raster files (`.tif`), typically the output of `raster_align` or `scale_rasters`. File names must follow the patterns supplied in `variable_patterns`, with any time placeholder substituted for the corresponding value from `time_cols`. Required for all methods.

variable_patterns
: Named character vector mapping clean variable names to raster filename patterns. For time-varying variables include the time placeholder in the pattern (e.g. "forest_cover" = "forest_cover_YEAR"); for static variables omit it (e.g. "elevation" = "elevation"). Time placeholders must match entries in time_cols.

method
: Character. Pseudoabsence generation method. One of "random", "buffer", or "environmental". Default is "random".

ratio
: Numeric. Number of pseudoabsence points to generate per presence point. Default is 1. Values of 2, 10, 50, etc. are accepted. Points are always distributed proportionally across time steps within each fold. Set to 0 to disable proportional allocation and use a fixed number of points per time step instead, in which case min_points_per_timestep must be greater than 0. ratio and min_points_per_timestep cannot both be 0.

buffer_distance
: Numeric. Distance in the units of the CRS (typically meters for projected CRS) within which pseudoabsence points are sampled. Required when method = "buffer". When method = "environmental", supplying a value automatically applies a spatial buffer constraint before environmental profiling, following the three-step approach of Senay et al. (2013). If NULL for the environmental method, no spatial constraint is applied. Default is NULL.

env_percentile
: Numeric between 0 and 1. Quantile threshold used to define the boundary of the known environmental tolerance when method = "environmental". Environmental cells within this quantile range across all variables are excluded from pseudoabsence sampling. Default is 0.05 (5th to 95th percentile envelope).

time_cols
: Character or character vector. Name of the column(s) containing the time step values. Must match time_cols used in spatiotemporal_partition and the time placeholders used in variable_patterns. Default is NULL.

pseudoabsence_times
: Vector. Optional vector of specific time step values (for the first time column) at which to generate pseudoabsences. When NULL (default), all time steps present in the occurrence data are used.

min_points_per_timestep
: Integer. Minimum number of pseudoabsence points to generate per time step per fold. Default is 1. When ratio = 0, this value sets the exact (fixed) number of points generated per time step per fold, independent of the number of presence points. ratio and min_points_per_timestep cannot both be 0.

create_plot
: Logical. If TRUE (default), generates diagnostic plots showing the spatial and temporal distribution of generated pseudoabsence points alongside presence points.

plot_by_fold
: Logical. If TRUE, generates one map per fold. If FALSE (default), generates a single combined map.

plot_palette
: Character. Name of an HCL or RColorBrewer palette used to color folds in diagnostic plots. Accepts any HCL palette name (see hcl.pals) or, if **RColorBrewer** is installed, any Brewer palette name. Default is "Dark 2".

output_file
: Character. Optional path to save the result as an .rds file. The parent directory will be created if it does not exist. Default is NULL.

verbose — Logical. If TRUE (default), prints progress messages during processing. Includes per-fold and per-time-step pseudoabsence counts.

Details

Generates sets of background data based on user-specified methodology that can be used as pseudoabsence data for the purposes of training presence/absence models.

The three generation methods differ in how the sampling region is defined:

- **Random**: Points are sampled uniformly at random from the full study area, excluding a negligible buffer around presence locations to prevent exact overlap.
- **Buffer**: Points are sampled within buffers of radius `buffer_distance` drawn around all fold presences, clipped to the reference shapefile boundary.
- **Environmental**: Raster cells whose values fall outside the species tolerance envelope in at least one variable are identified as candidates. K-means clustering then selects a spatially representative subset. If `buffer_distance` is supplied the environmental filtering is applied only within that buffered region, implementing the full three-step approach of Senay et al. (2013).

Value

Invisibly returns a list containing:

- `pseudoabsences`: An sf object of all generated pseudoabsence points with columns `fold`, `temporal_block`, `presence` (always 0), the time column(s) if provided, and extracted environmental variable values matched to each point's time step.
- `plots`: A named list of recorded plot objects when `create_plot = TRUE`. Contains `temporal_distribution` and either `spatial_combined` or one `spatial_fold_N` entry per fold. Plots can be replayed with `grDevices::replayPlot()`.
- `summary`: A data frame summarising points generated per fold with columns `fold`, `n_presences`, `n_pseudoabsences`, and `ratio_achieved`.

References

Senay SD, Worner SP, Ikeda T (2013) Novel Three-Step Pseudo-Absence Selection Technique for Improved Species Distribution Modeling. PLoS ONE 8(8): e71218.

See Also

Preprocessing: `spatiotemporal_partition`, `temporally_explicit_extraction`

Modeling: `build_temporal_hv`, `build_temporal_glm`, `build_temporal_gam`, `build_temporal_rf`

Examples

```
data(tmr_partition_small, package = "TemporalModelR")

scl_dir   <- system.file("extdata/rasters_scaled",
                         package = "TemporalModelR")
```

```
ref_file  <- system.file("extdata/rasters_raw/elevation.tif",
                         package = "TemporalModelR")

study_crs <- sf::st_crs(terra::rast(ref_file))

study_area_sf <- sf::st_as_sf(sf::st_as_sfc(
  sf::st_bbox(c(xmin = 0, xmax = 3000, ymin = 0, ymax = 1500),
              crs = study_crs)
))

generate_absences(
  partition_result         = tmr_partition_small,
  reference_shapefile_path = study_area_sf,
  raster_dir               = scl_dir,
  variable_patterns        = c(
    "elevation"    = "elevation",
    "forest_cover" = "forest_cover_YEAR"
  ),
  method                   = "random",
  ratio                    = 1,
  time_cols                = c("year"),
  create_plot              = FALSE,
  verbose                  = FALSE
)
```

`generate_spatiotemporal_predictions`
Generate Spatiotemporal Predictions from Temporal Models

Description

Generates temporally explicit habitat suitability predictions by projecting fitted models onto environmental raster stacks matching each time period. Accepts output from any of the four TemporalModelR modeling functions. Produces per-time-step assessment metrics showing how predictions and test point coverage vary over time.

Usage

```
generate_spatiotemporal_predictions(partition_result, model_result,
                                    pseudoabsence_result = NULL, raster_dir,
                                    variable_patterns, time_cols, time_steps,
                                  output_dir = file.path(tempdir(), "Predictions"),
                                    overwrite = FALSE,
                                    verbose = TRUE)
```

Arguments

`partition_result`
: List or character. Output from `spatiotemporal_partition` or path to an `.rds` file containing that output.

`model_result` List or character. Output from any of `build_temporal_hv`, `build_temporal_glm`, `build_temporal_gam`, or `build_temporal_rf`, or a path to an `.rds` file. Model type is detected automatically from the `model_type` field.

`pseudoabsence_result` List, character, or `NULL`. Optional. Output from `generate_absences` or path to an `.rds` file. When supplied for presence/absence models (GLM, GAM, RF), the held-out pseudoabsence test points for each fold are filtered to the current time step and used alongside presence test points to compute per-timestep TN, FP, Specificity, and TSS. These columns are added to the timestep metrics table when pseudoabsences are available. Ignored for hypervolume models. Default is `NULL`.

`raster_dir` Character. Directory containing environmental raster files (`.tif`), typically the output of `raster_align` or `scale_rasters`. File names must follow the patterns supplied in `variable_patterns`, with any time placeholder substituted for the corresponding value from `time_cols`.

`variable_patterns` Named character vector mapping clean variable names to raster filename patterns. For time-varying variables include the time placeholder in the pattern (e.g. `"forest_cover" = "forest_cover_YEAR"`); for static variables omit it (e.g. `"elevation" = "elevation"`). Time placeholders must match entries in `time_cols`.

`time_cols` Character. Name of the column(s) containing year or time step values in the occurrence data. Must match `time_cols` used in `spatiotemporal_partition` and the time placeholders used in `variable_patterns`.

`time_steps` Vector, data frame, or matrix of time periods for which to generate predictions.

`output_dir` Character. Directory to write prediction rasters and the assessment metrics CSV. Default is `file.path(tempdir(), "Predictions")`.

`overwrite` Logical. If `TRUE`, overwrites existing output files. If `FALSE` (default), existing files are skipped.

`verbose` Logical. If `TRUE` (default), prints progress messages during processing. Includes per-time-step prediction progress.

Details

G-space predictions are produced as rasters for each time-step and fold of an input model.

Per-timestep metrics are computed per fold per time step by extracting raster predictions at the held-out presence points that fall within that time step. `Pct_Suitable` records the proportion of the study area predicted suitable. `Sensitivity` shows how consistently test points are captured. `CBP` (cumulative binomial probability) tests whether the observed number of correctly predicted test points is better than expected by random placement: under the null, each point has `Pct_Suitable` probability of falling in suitable area, so `dbinom(TP, N_Test_Pts, Pct_Suitable)` gives the probability of observing exactly `TP` correct by chance. Small values indicate predictions are better than random.

Value

Invisibly returns a list containing:

- `timestep_metrics`: data frame of per-fold per-time-step metrics. Columns always present: `Fold`, the time column(s), `Pct_Suitable`, `N_Pres`, `TP`, `FN`, `Sensitivity`, `CBP`. When `pseudoabsence_result` is supplied for parametric models, additionally: `N_Abs`, `TN`, `FP`, `Specificity`, `TSS`. Saved as `Timestep_Assessment_Metrics.csv`.
- `overall_summary`: data frame of pooled metrics per fold. Columns: `Fold`, `N_Timesteps`, `Mean_Pct_Suitable`, `Total_TP`, `Total_FN`, `Overall_Sensitivity`, `Overall_CBP`. When pseudoabsences available: additionally `Total_TN`, `Total_FP`, `Overall_Specificity`, `Overall_TSS`.
- `prediction_files`: character vector of paths to saved prediction rasters.
- `model_type`: character string recording the model type used.

See Also

Preprocessing: `scale_rasters`, `spatiotemporal_partition`

Modeling: `build_temporal_hv`, `build_temporal_glm`, `build_temporal_gam`, `build_temporal_rf`

Post-processing: `summarize_raster_outputs`, `plot_model_assessment`

Examples

```
data(tmr_partition, package = "TemporalModelR")

data(tmr_glm,       package = "TemporalModelR")

data(tmr_absences,  package = "TemporalModelR")

scl_dir    <- system.file("extdata/rasters_scaled",
                          package = "TemporalModelR")

time_steps <- expand.grid(
  year             = 1:15,
  season           = "Spring",
  stringsAsFactors = FALSE
)

generate_spatiotemporal_predictions(
  partition_result     = tmr_partition,
  model_result         = tmr_glm,
  pseudoabsence_result = tmr_absences,
  raster_dir           = scl_dir,
  variable_patterns    = c(
    "elevation"    = "elevation",
    "forest_cover" = "forest_cover_YEAR",
    "prseas"       = "prseas_YEAR_SEASON"
  ),
  time_cols            = c("year", "season"),
  time_steps           = time_steps,
  output_dir           = tempdir(),
  overwrite            = TRUE,
  verbose              = FALSE
)
```

plot_model_assessment *Visualize Model Assessment Metrics Across Time*

Description

Generates diagnostic plots from the per-timestep assessment table produced by generate_spatiotemporal_predictions, optionally overlaying overall reference values from the model result object.

Usage

```
plot_model_assessment(predictions, time_column,
                      secondary_time_mode = "combine", model_result = NULL,
                      cbp_threshold = 0.05, plot_palette = "Dark 2",
                      verbose = TRUE)
```

Arguments

predictions	List returned by generate_spatiotemporal_predictions, or a named list with at least a timestep_metrics element. The timestep_metrics element may also be a path to a Timestep_Assessment_Metrics.csv file produced by generate_spatiotemporal_predictions.
time_column	Character. Name of the primary time column in timestep_metrics to use as the x axis (e.g. "year"). When predictions span multiple time columns (e.g. "year" and "season"), provide all relevant column names as a character vector and control how secondary columns are handled via secondary_time_mode.
secondary_time_mode	Character. How to handle secondary time columns when time_column has length > 1. One of: • "combine" (default): secondary time values are appended to the primary value to form a single ordered x-axis label (e.g. 1_Spring, 1_Summer, 2_Spring, ...). • "facet": a separate plot is produced for each unique combination of secondary time values, with the primary time column as the x axis on every panel.
model_result	List or character. Optional. Output from a build_temporal_*() function or path to its .rds file. When supplied, overall sensitivity and specificity from model_result$fold_test_metrics are added as per-fold reference lines. Default is NULL.
cbp_threshold	Numeric. Significance threshold for CBP. Default is 0.05.
plot_palette	Character. Name of an HCL or RColorBrewer palette used to color folds in diagnostic plots. Accepts any HCL palette name (see hcl.pals) or, if **RColorBrewer** is installed, any Brewer palette name. Default is "Dark 2".
verbose	Logical. If TRUE (default), prints progress messages during processing.

Details

Plots per-fold and per-timestep diagnostic plots for data produced by generate_spatiotemporal_predictions. These quick visuals can be used by users to assess model performance and significance and decide if the model's performance warrants further interpretation of the results through post-processing analyses.

Value

Invisibly returns a named list containing:

- pct_suitable: Recorded plot of proportion of study area predicted suitable per time step.
- sensitivity: Recorded plot of per-timestep sensitivity.
- specificity: Recorded plot of per-timestep specificity (only present when pseudoabsence data were used).
- cbp: Recorded plot of cumulative binomial probability per time step on a log scale.
- tp_fn: Recorded plot of true positives and false negatives per time step.
- tn_fp: Recorded plot of true negatives and false positives per time step (only present when pseudoabsence data were used).
- timestep_summary: Data frame of per-time-step cross-fold mean and SD for each metric.
- overall_summary: Data frame from predictions$overall_summary, when present.

See Also

Preprocessing: spatiotemporal_partition, generate_absences

Modeling: build_temporal_glm, build_temporal_gam, build_temporal_rf, build_temporal_hv,

Post-processing: generate_spatiotemporal_predictions, summarize_raster_outputs

Examples

```
data(tmr_predictions, package = "TemporalModelR")

plot_model_assessment(
  predictions         = tmr_predictions,
  time_column         = c("year", "season"),
  secondary_time_mode = "combine",
  verbose             = FALSE
)
```

raster_align *Align and Standardize Raster Files to a Reference Raster*

Description

Preprocessing function that aligns a batch of raster files to a specified reference raster by performing reprojection, resampling, and masking operations. Ensures all rasters share identical CRS, resolution, and spatial extent for later analyses.

Usage

```
raster_align(input_dir, output_dir, reference_raster,
             output_suffix = "_Masked_Updated", pattern = ".*\\.tif$",
             resample_method = "bilinear",
             overwrite = FALSE, verbose = TRUE)
```

Arguments

input_dir	Character. Directory containing the input raster files.
output_dir	Character. Directory where processed rasters will be saved.
reference_raster	Character, SpatRaster, or RasterLayer. File path or raster object used as the alignment reference.
output_suffix	Character. Suffix appended to output filenames. Default is "_Masked_Updated". Output files are named <original_name><output_suffix>.tif.
pattern	Character. Regular expression used to match raster files within input_dir. Default is ".*\.tif$".
resample_method	Character. Resampling method passed to resample. Default is "bilinear", which is appropriate for continuous variables. Use "near" for categorical rasters (e.g. land cover classes) to avoid interpolating between class codes. Other accepted values include "cubic" and "lanczos".
overwrite	Logical. If TRUE, overwrites existing output files. If FALSE (default), existing files are skipped.
verbose	Logical. If TRUE (default), prints progress messages during processing. Includes file counts, overwrite mode, per-file progress, and a completion summary.

Details

For each raster in input_dir matching pattern, the function:

1. Reprojects to the CRS of the reference raster
2. Resamples to match the reference resolution using resample_method
3. Masks values outside the reference raster's non-NA extent
4. Saves the result as a GeoTIFF to output_dir

This preprocessing step ensures spatial alignment before applying other downstream analyses such as temporally_explicit_extraction or scale_rasters.

Value

Invisibly returns a list containing:

- output_files: Character vector of file paths written to output_dir.
- n_processed: Integer. Number of rasters successfully processed in this call (excludes skipped files when overwrite = FALSE).

See Also

Preprocessing: scale_rasters, temporally_explicit_extraction

Examples

```
raw_dir <- system.file("extdata/rasters_raw", package = "TemporalModelR")

ref     <- file.path(raw_dir, "elevation.tif")

out_dir <- file.path(tempdir(), "aligned")

raster_align(
  input_dir        = raw_dir,
  output_dir       = out_dir,
  reference_raster = ref,
  overwrite        = TRUE,
  verbose          = FALSE
)
```

---

scale_rasters *Scale Environmental Rasters Using Species Occurrence Data*

Description

Preprocessing function that standardizes raster files using precomputed mean and standard deviation values from species occurrence data. Supports both temporally static and dynamic rasters.

Usage

```
scale_rasters(input_dir, output_dir, scaling_params_file,
              variable_patterns, time_cols = NULL, output_suffix = "_Scaled",
              overwrite = FALSE, verbose = TRUE)
```

Arguments

input_dir	Character. Directory containing aligned input .tif raster files, typically the output of raster_align.
output_dir	Character. Directory where scaled rasters will be saved.
scaling_params_file	Character. Path to CSV file with columns: variable, mean, sd. Typically generated by temporally_explicit_extraction.
variable_patterns	Named character vector mapping clean variable names to raster filename patterns. For time-varying variables include the time placeholder in the pattern (e.g. "forest_cover" = "forest_cover_YEAR"); for static variables omit it (e.g. "elevation" = "elevation"). Time placeholders must match entries in time_cols.
time_cols	Character vector of time placeholders for dynamic variables. Default is NULL.
output_suffix	Character. Suffix to append to output raster filenames. Default is "_Scaled".
overwrite	Logical. If TRUE, overwrites existing output files. If FALSE (default), existing files are skipped.
verbose	Logical. If TRUE (default), prints progress messages during processing. Includes per-variable scaling progress.

Details

Applies scaling to rasters via z-score transformation: (value - mean) / sd as is calculated during temporally_explicit_extraction. Scaled rasters are written to output_dir.

Scaled rasters should use the same scaling parameters derived from species occurrence data to ensure consistent standardization between training data and prediction layers.

Value

Invisibly returns a list containing:

- n_scaled: Integer. Total number of variables successfully scaled (static plus dynamic).
- output_dir: Character. Path to the directory containing scaled rasters.

See Also

Preprocessing: temporally_explicit_extraction, raster_align

Examples

```
aln_dir     <- system.file("extdata/rasters_aligned",
                           package = "TemporalModelR")

params_file <- system.file(
  "extdata/points/extracted_seasonal_Scaling_Parameters.csv",
  package = "TemporalModelR"
)
```

```
out_dir <- file.path(tempdir(), "scaled")

scale_rasters(
  input_dir           = aln_dir,
  output_dir          = out_dir,
  scaling_params_file = params_file,
  variable_patterns   = c(
    "elevation"    = "elevation",
    "forest_cover" = "forest_cover_YEAR",
    "prseas"       = "prseas_YEAR_SEASON"
  ),
  time_cols           = c("year", "season"),
  overwrite           = TRUE,
  verbose             = FALSE
)
```

`spatiotemporal_partition` *Spatiotemporal Cross-Validation Partitioning*

Description

Preprocesses species occurrence data by partitioning it into spatially and temporally structured folds for cross-validation. Supports creation of spatial-only folds, temporal-only folds, and random folds.

Usage

```
spatiotemporal_partition(reference_shapefile_path, points_file_path,
                         time_cols = NULL, xcol = NULL, ycol = NULL,
                         points_crs = NULL, n_spatial_folds = 0,
                         n_temporal_folds = 0, n_balanced_folds = 0,
                         n_random_folds = 0, single_fold= FALSE,
                         max_imbalance = 0.05, max_attempts = 10,
                         create_plot = TRUE, plot_palette = "Dark 2",
                         output_file= NULL, verbose    = TRUE)
```

Arguments

`reference_shapefile_path`
: Character or sf object. Path to a polygon file or an `sf` polygon object defining the study area.

`points_file_path`
: Character, sf object, sfc object, Spatial object, or data frame. Path to occurrence data (`.csv`, `.shp`, `.geojson`, `.gpkg`) or a spatial object.

`time_cols`
: Character. Name of a single column containing temporal values (e.g. year). Used to define temporal blocks. Required when using temporal folds. Must be a single column name; does not support more than one time column unlike other

functions in this package. Compound time representations (e.g. year + season) should be encoded into a single ordered numeric column before partitioning, or only one (e.g. year) should be used.

`xcol` Character. Name of the x-coordinate column. Required when `points_file_path` is a CSV file or data frame.

`ycol` Character. Name of the y-coordinate column. Required when `points_file_path` is a CSV file or data frame.

`points_crs` Character or CRS object. CRS of the input points. Required when `points_file_path` is a CSV file or data frame.

`n_spatial_folds`
Integer. Number of spatially explicit folds. Ignored when using random folds. Default is `0`.

`n_temporal_folds`
Integer. Number of temporally explicit folds. When used alone (with `n_spatial_folds = 0`), creates temporal-only folds where each fold spans the full study area but covers a distinct slice of the time series. When combined with `n_spatial_folds`, creates a spatiotemporal design. Ignored when using random folds. Default is `0`.

`n_balanced_folds`
Integer. Reserved for future use. Default is `0` (disabled).

`n_random_folds` Integer. Number of random folds with no spatial or temporal structure. Overrides all other fold parameters. Default is `0`.

`single_fold` Logical. If `TRUE`, bypasses all partitioning and assigns all points to a single fold (fold 1). In this mode all points are used for both training and testing, producing a single model trained on the full dataset. All downstream functions accept the result identically to a standard multi-fold partition. Overrides all fold count parameters. Default is `FALSE`.

`max_imbalance` Numeric. Maximum allowed fold size imbalance as a proportion between 0 and 1. Default is `0.05`.

`max_attempts` Integer. Maximum number of partitioning attempts for spatiotemporal and balanced modes. Each attempt re-runs the spatial block construction; the attempt with the lowest imbalance is returned. Ignored for random and spatial-only modes. Default is `10`.

`create_plot` Logical. If `TRUE` (default), generates diagnostic plots showing fold distributions.

`plot_palette` Character. Name of an HCL or RColorBrewer palette used to color folds in diagnostic plots. Accepts any HCL palette name (see `hcl.pals`) or, if **RColorBrewer** is installed, any Brewer palette name. Default is `"Dark 2"`.

`output_file` Character. Optional path to save the result as an `.rds` file. The parent directory will be created if it does not exist. Default is `NULL`.

`verbose` Logical. If `TRUE` (default), prints progress messages during processing. Includes the partition mode, fold structure, per-fold point counts, and file-save confirmation.

Details

Works better with smaller numbers of folds and may have difficulties creating even folds for large numbers of groups or where sample sizes are very small.

The function partitions data into folds using one of five modes:

- **Single fold**: All points are assigned to fold 1 and used for both training and testing. This produces a single model trained on the full dataset with no held-out validation. Useful when sample sizes are too small for cross-validation, or as a final production model step after cross-validation has already established model quality. Set `single_fold = TRUE`. All downstream functions accept the result identically to standard multi-fold output.
- **Random**: Points are assigned to folds by random shuffling with no spatial or temporal structure. Each fold is a simple random sample of the full dataset, intended as a naive baseline that makes no attempt to reduce spatial or temporal autocorrelation between training and test sets. Use `n_random_folds`.
- **Spatial-only**: The study area is divided into k contiguous spatial regions using a recursive k-d tree bisection algorithm. At each step the point set is split along its longest spatial axis, recursively halving until the target number of folds is reached. A centroid reassignment pass then refines boundaries to improve balance. Each region becomes one fold, so training always occurs on data from geographically distinct areas relative to the test fold. No temporal separation is imposed, meaning that points from any time period may appear in any fold. Use `n_spatial_folds` alone.
- **Temporal-only**: Each fold covers the full spatial extent of the study area but is restricted to a distinct, non-overlapping slice of the time series. The global time series is divided into `n_temporal_folds` equal intervals using quantile-based breaks, and all points within each interval form one fold. This design tests model transferability across time while retaining full spatial coverage in every fold. Use `n_temporal_folds` alone (with `n_spatial_folds = 0`). Requires `time_cols`.
- **Spatiotemporal**: Folds are assigned using the same recursive k-d tree bisection as spatial-only mode, operating on the full point set to produce spatially contiguous groups. The resulting groups are then split into a spatial pool (`n_spatial_folds` folds drawn from geographically distinct regions) and a temporal pool (`n_temporal_folds` folds each restricted to a distinct slice of the time series but spanning the full study area). Together the two pools assess both geographic and temporal transferability in a single cross-validation design. Use `n_spatial_folds` and `n_temporal_folds` together. Requires `time_cols`.

Fold assignment uses a recursive k-d tree bisection algorithm that splits points along their longest spatial axis at each step, followed by a centroid reassignment pass to improve boundary regularity and point-count balance. Voronoi tessellation on fold centroids is used only for visualisation of the resulting spatial boundaries. For temporal mode, temporal blocks are defined by dividing the global time series into equal intervals using quantile-based breaks. For spatiotemporal mode, the typical spatial assignment is done, but with one larger spatial block made with enough points to represent all of the temporal folds, then the temporal blocking is applied to those points.

Partitioned datasets are suitable for cross-validation in modeling workflows, ensuring spatial and/or temporal independence between folds.

Value

Invisibly returns a list containing:

- `folds`: Data frame of fold assignments with a `fold` column identifying each point's cross-validation fold.
- `points_sf`: sf object of occurrence points with assigned folds.
- `voronoi_folds`: sf object of Voronoi polygons representing the spatial fold boundaries. `NULL` for random folds, temporal-only folds, and single-fold mode.
- `summary`: Data frame of partitioning summary statistics.
- `plots`: Named list of recorded plot objects when `create_plot = TRUE`. Empty list in single-fold mode.

See Also

Preprocessing: spatiotemporal_rarefaction, temporally_explicit_extraction, generate_absences

Modeling: build_temporal_hv, build_temporal_glm, build_temporal_gam, build_temporal_rf

Examples

```
pts_file <- system.file(
  "extdata/points/extracted_seasonal_Scaled_Values.csv",
  package = "TemporalModelR"
)

ref_file <- system.file("extdata/rasters_raw/elevation.tif",
                        package = "TemporalModelR")

study_crs <- sf::st_crs(terra::rast(ref_file))

study_area_sf <- sf::st_as_sf(sf::st_as_sfc(
  sf::st_bbox(c(xmin = 0, xmax = 3000, ymin = 0, ymax = 1500),
              crs = study_crs)
))

spatiotemporal_partition(
  reference_shapefile_path = study_area_sf,
  points_file_path         = pts_file,
  xcol                     = "x",
  ycol                     = "y",
  points_crs               = study_crs,
  time_cols                = "year",
  n_spatial_folds          = 2,
  n_temporal_folds         = 2,
  create_plot              = FALSE,
  verbose                  = FALSE
)
```

`spatiotemporal_rarefaction` *Spatiotemporal Rarefaction of Species Occurrence Data*

Description

Preprocessing function that rarefies species occurrence data to one point per raster pixel, optionally accounting for temporal components. Reduces sampling bias and spatial autocorrelation in occurrence datasets.

Usage

```
spatiotemporal_rarefaction(points_sp, output_dir, reference_raster,
                           time_cols = NULL, xcol = NULL, ycol = NULL,
                           points_crs = NULL, output_prefix = "Pts_Database",
                           verbose = TRUE)
```

Arguments

`points_sp`	Input point data. Accepts an sf object, data frame, SpatialPointsDataFrame, or file path to a `.csv`, `.shp`, `.geojson`, or `.gpkg` file.
`output_dir`	Character. Directory where output CSV files will be saved.
`reference_raster`	Character, SpatRaster, or RasterLayer. Raster used to define pixel boundaries for rarefaction. Accepts a file path, RasterLayer, or SpatRaster.
`time_cols`	Character vector. Column names in `points_sp` defining temporal grouping for spatiotemporal rarefaction. When `NULL` (default), only spatial rarefaction is performed.
`xcol`	Character. Name of the x-coordinate column. Required when `points_sp` is a CSV file or data frame.
`ycol`	Character. Name of the y-coordinate column. Required when `points_sp` is a CSV file or data frame.
`points_crs`	Character or CRS object. CRS of the input points. Required when `points_sp` is a CSV file or data frame.
`output_prefix`	Character. Prefix for output file names. Default is `"Pts_Database"`. Output files are named `<output_prefix>_OnePerPix.csv` and, when `time_cols` are provided, `<output_prefix>_OnePerPixPerTimeStep.csv`.
`verbose`	Logical. If `TRUE` (default), prints progress messages during processing. Includes file loading, missing-data removal counts, rarefaction summaries, and a final comparison of spatial vs spatiotemporal point counts.

Details

The function assigns each point to a raster pixel using the resolution and extent of `reference_raster`, then performs:

- Spatial rarefaction: retains one point per pixel, written to `<output_prefix>_OnePerPix.csv`.
- Spatiotemporal rarefaction (when `time_cols` are provided): retains one point per pixel per unique combination of time column values, written to `<output_prefix>_OnePerPixPerTimeStep.csv`.

Output CSV files are suitable as direct input to `temporally_explicit_extraction`.

Value

Invisibly returns a list containing:

- `input_points`: Integer. Number of input points after CRS alignment and removal of rows with missing time column values.
- `spatial_points`: Integer. Number of points retained after spatial-only rarefaction (one per pixel).
- `spatiotemporal_points`: Integer. Number of points retained after spatiotemporal rarefaction (one per pixel per time combination). `NA` when `time_cols` is not provided.
- `time_cols_used`: Character vector of time columns used. `NULL` when `time_cols` is not provided.
- `spatial_table`: Data frame of spatially rarefied points with columns `pixel_id`, `X`, `Y`, and any `time_cols`.
- `spatiotemporal_table`: Data frame of spatiotemporally rarefied points with columns `pixel_id`, `X`, `Y`, and `time_cols`. `NULL` when `time_cols` is not provided.
- `files_created`: Named list of file paths written. Always contains `$spatial`; additionally contains `$spatiotemporal` when `time_cols` are provided.

See Also

Preprocessing: `temporally_explicit_extraction`, `spatiotemporal_partition`

Examples

```
pts_file <- system.file(
  "extdata/points/synthetic_occurrence_points.csv",
  package = "TemporalModelR"
)

ref_file <- system.file("extdata/rasters_raw/elevation.tif",
                        package = "TemporalModelR")

out_dir  <- file.path(tempdir(), "rarefied")

spatiotemporal_rarefaction(
  points_sp        = pts_file,
  output_dir       = out_dir,
  reference_raster = ref_file,
```

```
  time_cols        = c("year", "season"),
  xcol             = "x",
  ycol             = "y",
  points_crs       = terra::crs(terra::rast(ref_file)),
  verbose          = FALSE
)
```

---

summarize_raster_outputs *Summarize Prediction Rasters into Consensus Outputs*

Description

Post-processing function that synthesizes per-time-step fold-vote rasters (output from generate_spatiotemporal_predicti into binary consensus predictions and a temporal frequency summary. Each input raster contains integer vote counts per pixel the number of cross-validation folds that classified that pixel as suitable. The consensus threshold controls how many folds must agree for a pixel to be classified as suitable in the output binary rasters.

Usage

```
summarize_raster_outputs(predictions_dir, output_dir = NULL,
                         consensus = 1, file_pattern = "Prediction_.*\\.tif$",
                         overwrite = FALSE, verbose = TRUE)
```

Arguments

predictions_dir	Character. Directory containing prediction raster files, typically the output_dir used in generate_spatiotemporal_predictions.
output_dir	Character. Directory for output files. Defaults to predictions_dir if NULL.
consensus	Integer. Minimum number of folds that must agree on suitability for a pixel to be classified as suitable in the binary output. For example, consensus = 1 marks any pixel suitable if at least one fold predicts it suitable; consensus = 4 requires at least four folds to agree. Must be between 1 and the maximum vote count in the rasters (number of folds). Default is 1.
file_pattern	Character. Regular expression to match prediction raster files. Default is "Prediction_.*\.tif$".
overwrite	Logical. If TRUE, overwrites existing output files. If FALSE (default), existing files are skipped.
verbose	Logical. If TRUE (default), prints progress messages during processing.

Details

Input rasters are fold-vote-count rasters produced by `generate_spatiotemporal_predictions`, where each pixel value is the number of cross-validation fold models that predicted suitability at that location for that time step. The `consensus` threshold is applied as: `binary = as.integer(vote_count >= consensus)`.

The frequency raster (proportion of time steps suitable under the chosen consensus) serves as input to `analyze_temporal_patterns` for identifying long-term trends in suitability.

Value

Invisibly returns a list containing:

- `consensus_stack`: `SpatRaster` stack of per-time-step binary consensus rasters (one layer per input file).
- `frequency_raster`: `SpatRaster` showing the proportion of time steps during which each pixel met the consensus threshold. Values range from 0 (never suitable) to 1 (always suitable).
- `consensus`: the consensus threshold used.
- `n_timesteps`: number of time steps processed.
- `consensus_dir`: path to the directory containing per-time-step binary consensus rasters.
- `frequency_file`: path to the written frequency raster file.

See Also

Upstream: `generate_spatiotemporal_predictions`

Downstream: `analyze_temporal_patterns`

Examples

```
pred_dir <- system.file("extdata/predictions",
                        package = "TemporalModelR")

summarize_raster_outputs(
  predictions_dir = pred_dir,
  output_dir      = tempdir(),
  consensus       = 3,
  overwrite       = TRUE,
  verbose         = FALSE
)
```

`temporally_explicit_extraction` *Extract Time-Aligned Environmental Values at Species Occurrences*

Description

Preprocessing function that extracts raster values to species occurrence records based on temporal components. Matches environmental layers to occurrence timestamps and optionally computes scaling parameters for standardization.

Usage

```
temporally_explicit_extraction(points_sp, raster_dir, variable_patterns,
                               time_cols, xcol = NULL, ycol = NULL,
                               points_crs = NULL, output_dir,
                               output_prefix = "temp_explicit_df",
                               save_raw = TRUE, save_scaled = TRUE,
                               save_scaling_params = TRUE,
                               verbose = TRUE)
```

Arguments

`points_sp`	sf object, SpatialPointsDataFrame, file path to .csv/.shp/.geojson/.gpkg, or data frame with coordinate columns.
`raster_dir`	Character. Directory containing environmental raster files (`.tif`), typically the output of `raster_align`. File names must follow the patterns supplied in `variable_patterns`, with any time placeholder substituted for the corresponding value from `time_cols`.
`variable_patterns`	Named character vector mapping clean variable names to raster filename patterns. For time-varying variables include the time placeholder in the pattern (e.g. `"forest_cover" = "forest_cover_YEAR"`); for static variables omit it (e.g. `"elevation" = "elevation"`). Time placeholders must match entries in `time_cols`.
`time_cols`	Character vector of time column names present in the point data (e.g., c("YEAR"), c("YEAR", "MONTH")).
`xcol`	Character. Name of the x-coordinate column. Required when `points_sp` is a CSV file or data frame.
`ycol`	Character. Name of the y-coordinate column. Required when `points_sp` is a CSV file or data frame.
`points_crs`	Character or CRS object. CRS of the input points. Required when `points_sp` is a CSV file or data frame.
`output_dir`	Character. Directory to write output files.
`output_prefix`	Character. Prefix for output filenames. Default is "temp_explicit_df".
`save_raw`	Logical. If `TRUE` (default), writes raw extracted values CSV. If `FALSE`, skips raw values output.

`save_scaled` Logical. If `TRUE` (default), writes z-scaled values CSV. If `FALSE`, skips scaled values output.

`save_scaling_params` Logical. If `TRUE` (default), writes CSV of per-variable means and standard deviations. If `FALSE`, skips scaling parameters output.

`verbose` Logical. If `TRUE` (default), prints progress messages during processing. Includes file loading, extraction progress, and file-save confirmation.

Details

Extracts raster values to species occurrence records based on matched temporal components. Matches environmental layers to occurrence timestamps and optionally computes scaling parameters for standardization.

Output CSV files are written to output_dir containing raw values, scaled values, and scaling parameters.

Scaling parameters (mean and standard deviation) are optionally computed across all occurrence records for each variable. These parameters should be used with `scale_rasters` to standardize prediction layers.

Value

Invisibly returns a list containing:

- `raw_values`: Data frame of raw extracted values at each occurrence record (when `save_raw = TRUE`; `NULL` otherwise).
- `scaled_values`: Data frame of z-scaled extracted values (when `save_scaled = TRUE`; `NULL` otherwise).
- `scaling_params`: Data frame of per-variable means and standard deviations used for scaling (when `save_scaling_params = TRUE`; `NULL` otherwise). Pass this to `scale_rasters`.
- `files_created`: Named list of file paths written, with elements `raw`, `scaled`, and `scaling_params` (each `NULL` when the corresponding save flag is `FALSE`).

See Also

Preprocessing: `spatiotemporal_rarefaction`, `scale_rasters`, `spatiotemporal_partition`

Examples

```
pts_file <- system.file(
  "extdata/points/synthetic_occurrence_points.csv",
  package = "TemporalModelR"
)

aln_dir  <- system.file("extdata/rasters_aligned",
                        package = "TemporalModelR")

ref_file <- system.file("extdata/rasters_raw/elevation.tif",
                        package = "TemporalModelR")
```

```
out_dir  <- file.path(tempdir(), "extracted")

temporally_explicit_extraction(
  points_sp           = pts_file,
  raster_dir          = aln_dir,
  variable_patterns   = c(
    "elevation"    = "elevation",
    "forest_cover" = "forest_cover_YEAR",
    "prseas"       = "prseas_YEAR_SEASON"
  ),
  time_cols           = c("year", "season"),
  xcol                = "x",
  ycol                = "y",
  points_crs          = terra::crs(terra::rast(ref_file)),
  output_dir          = out_dir,
  save_raw            = TRUE,
  save_scaled         = FALSE,
  save_scaling_params = TRUE,
  verbose             = FALSE
)
```

`tmr_absences` *Pre-built pseudoabsence result (seasonal workflow)*

Description

Output from `generate_absences` using the buffer method with a 300 m buffer (3 pixels at the synthetic landscape's 100 m resolution) and a 2:1 pseudoabsence-to-presence ratio. Pseudoabsences are stratified by fold from `tmr_partition` and have `forest_cover`, `prseas`, and `elevation` extracted at each location's year-season combination. Loaded with `data(tmr_absences)`.

Usage

```
tmr_absences
```

Format

A list as returned by `generate_absences`, containing `$pseudoabsences` (an sf object with attached predictor columns), `$plots`, and `$summary`.

tmr_absences_annual *Pre-built pseudoabsence result (annual workflow)*

Description

Annual variant of `tmr_absences`. Buffer method with 300 m buffer and 2:1 ratio, generated against `tmr_partition_annual` with the annual predictor set (`forest_cover`, `pr_ann`, `elevation`). Loaded with `data(tmr_absences_annual)`.

Usage

```
tmr_absences_annual
```

Format

A list as returned by `generate_absences`.

tmr_glm *Pre-built GLM result (seasonal workflow)*

Description

Output from `build_temporal_glm` fit to `tmr_partition` and `tmr_absences` with the formula `~ forest_cover + prseas + elevation`, a logit link, and TSS-based threshold selection. One model per fold. Loaded with `data(tmr_glm)`.

Usage

```
tmr_glm
```

Format

A list of class `"TemporalGLM"` as returned by `build_temporal_glm`, containing `$models`, `$thresholds`, `$threshold_method`, `$model_formula`, `$link`, `$model_vars`, `$fold_training_data`, `$fold_test_metrics`, `$output_dir`, `$model_type`, and `$plots`.

tmr_glm_annual *Pre-built GLM result (annual workflow)*

Description

Annual variant of `tmr_glm`. Fit with the formula `~ forest_cover + pr_ann + elevation`, a logit link, and TSS-based threshold selection. One model per fold. Loaded with `data(tmr_glm_annual)`.

Usage

```
tmr_glm_annual
```

Format

A list of class `"TemporalGLM"`.

tmr_partition *Pre-built spatiotemporal partition (seasonal workflow)*

Description

Output from `spatiotemporal_partition` run on the synthetic occurrence dataset bundled in `inst/extdata/`. Built with `n_spatial_folds = 2` and `n_temporal_folds = 2`, producing four cross-validation folds (2 spatial + 2 temporal). Points span 15 years and the seasons Spring, Summer, and Autumn on a 15 x 30 cell study area (3000 m x 1500 m, 100 m pixels) in a custom synthetic local CRS. Predictors attached to each point are `forest_cover`, `prseas` (seasonal precipitation), and `elevation`, all z-scored. Loaded with `data(tmr_partition)`.

Usage

```
tmr_partition
```

Format

A list as returned by `spatiotemporal_partition`, containing `$folds`, `$points_sf`, `$voronoi_blocks`, `$voronoi_folds`, `$summary`, and `$plots`.

tmr_partition_annual *Pre-built spatiotemporal partition (annual workflow)*

Description

Annual variant of `tmr_partition`. Built from points rarefied with `time_cols = "year"` only (one point per pixel per year) and extracted against annual predictors. Predictors attached to each point are `forest_cover`, `pr_ann` (annual precipitation), and `elevation`, all z-scored. Uses `n_spatial_folds = 2` and `n_temporal_folds = 2` for four folds. Loaded with `data(tmr_partition_annual)`.

Usage

```
tmr_partition_annual
```

Format

A list as returned by `spatiotemporal_partition`.

tmr_partition_small *Pre-built spatiotemporal partition, small version (seasonal workflow)*

Description

A subsampled version of `tmr_partition` retaining approximately half the points per fold, for use in examples and tests where runtime is a concern. Built by sampling `ceiling(n / 2)` rows from each fold of `tmr_partition$points_sf`. Loaded with `data(tmr_partition_small)`.

Usage

```
tmr_partition_small
```

Format

A list with the same structure as `tmr_partition`, containing `$folds`, `$points_sf`, `$voronoi_blocks`, `$voronoi_folds`, `$summary`, and `$plots`.

See Also

`tmr_partition`

tmr_predictions *Pre-built spatiotemporal predictions (seasonal workflow)*

Description

Output from generate_spatiotemporal_predictions projecting tmr_glm across 15 years for the Spring season only (one prediction layer per year-season combination, 15 total). Loaded with data(tmr_predictions).

Usage

```
tmr_predictions
```

Format

A list as returned by generate_spatiotemporal_predictions, containing $timestep_metrics, $overall_summary, $prediction_files, and $model_type.

Details

Note that $prediction_files stores the absolute paths used at build time. To work with the per-timestep rasters on a user machine, regenerate them via generate_spatiotemporal_predictions or use the bundled prediction set in inst/extdata/predictions/.

tmr_predictions_annual *Pre-built spatiotemporal predictions (annual workflow)*

Description

Annual variant of tmr_predictions. Projects tmr_glm_annual across 15 years on a single (year-only) time axis. Loaded with data(tmr_predictions_annual).

Usage

```
tmr_predictions_annual
```

Format

A list as returned by generate_spatiotemporal_predictions.

Index