Package 'ldmppr'

February 24, 2025

Type Package

Title Estimate and Simulate from Location Dependent Marked Point Processes

Version 1.0.4

Maintainer Lane Drew <lanetdrew@gmail.com>

Description A suite of tools for estimating, assessing model fit, simulating from, and visualizing location dependent marked point processes characterized by regularity in the pattern.
You provide a reference marked point process, a set of raster images containing location specific covariates, and select the estimation algorithm and type of mark model.
'ldmppr' estimates the process and mark models and allows you to check the appropriateness of the model using a variety of diagnostic tools.
Once a satisfactory model fit is obtained, you can simulate from the model and visualize the results.
Documentation for the package 'ldmppr' is available in the form of a vignette.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Imports Rcpp (>= 1.0.12), terra, doParallel, xgboost, ranger, parsnip, dials, bundle, recipes, rsample, tune, workflows, magrittr, hardhat, ggplot2, spatstat.geom, spatstat.explore, nloptr, GET, progress, dplyr, future, furrr, yardstick

LinkingTo Rcpp, RcppArmadillo

URL https://github.com/lanedrew/ldmppr

BugReports https://github.com/lanedrew/ldmppr/issues

RoxygenNote 7.3.2

Suggests knitr, rmarkdown

VignetteBuilder knitr

Depends R (>= 3.5.0)

NeedsCompilation yes

Author Lane Drew [aut, cre, cph] (<https://orcid.org/0009-0006-5427-4092>),
Andee Kaplan [aut] (<https://orcid.org/0000-0002-2940-889X>)

Repository CRAN

Date/Publication 2025-02-24 21:00:02 UTC

Contents

`check_model_fit` *Check the fit of estimated self-correcting model on the reference point pattern dataset*

Description

Allows the user to perform global envelope tests for the nonparametric L, F, G, J, E, and V summary functions from the `spatstat` package. These tests serve as a goodness of fit measure for the estimated model relative to the reference dataset of interest.

Usage

```
check_model_fit(
  reference_data,
  t_min = 0,
  t_max = 1,
  sc_params = NULL,
  anchor_point = NULL,
  raster_list = NULL,
  scaled_rasters = FALSE,
  mark_model = NULL,
  xy_bounds = NULL,
  include_comp_inds = FALSE,
  thinning = TRUE,
  correction = "none",
  competition_radius = 15,
  n_sim = 2500,
  save_sims = TRUE,
  verbose = TRUE,
  seed = 0
)
```

Arguments

`reference_data`	a ppp object for the reference dataset.
`t_min`	minimum value for time.
`t_max`	maximum value for time.
`sc_params`	vector of parameter values corresponding to (alpha_1, beta_1, gamma_1, alpha_2, beta_2, alpha_3, beta_3, gamma_3).
`anchor_point`	vector of (x,y) coordinates of point to condition on.
`raster_list`	a list of raster objects.
`scaled_rasters`	‘TRUE‘ or ‘FALSE‘ indicating whether the rasters have been scaled.
`mark_model`	a model object (typically from `train_mark_model`).
`xy_bounds`	a vector of domain bounds (2 for x, 2 for y).
`include_comp_inds`	‘TRUE‘ or ‘FALSE‘ indicating whether to generate and use competition indices as covariates.
`thinning`	‘TRUE‘ or ‘FALSE‘ indicating whether to use the thinned or unthinned simulated values.
`correction`	type of correction to apply ("none" or "toroidal").
`competition_radius`	distance for competition radius if `include_comp_inds` is ‘TRUE‘.
`n_sim`	number of simulated datasets to generate.
`save_sims`	‘TRUE‘ or ‘FALSE‘ indicating whether to save and return the simulated datasets.
`verbose`	‘TRUE‘ or ‘FALSE‘ indicating whether to show progress of model checking.
`seed`	an integer value to set the seed for reproducibility.

Details

This function relies on the `spatstat` package for the calculation of the point pattern metrics and the `GET` package for the global envelope tests. The L, F, G, J, E, and V functions are a collection of non-parametric summary statistics that describe the spatial distribution of points and marks in a point pattern. See the documentation for [spatstat.explore::Lest()], [spatstat.explore::Fest()], [spatstat.explore::Gest()], [spatstat.explore::Jest()], [spatstat.explore::Emark()], and [spatstat.explore::Vmark()] for more information. Also, see the [GET::global_envelope_test()] function for more information on the global envelope tests.

Value

a list containing a combined global envelope test, individual global envelope tests for the L, F, G, J, E, and V functions, and simulated metric values (if specified).

References

Baddeley, A., Rubak, E., & Turner, R. (2015). *Spatial Point Patterns: Methodology and Applications with R*. Chapman and Hall/CRC Press, London. ISBN 9781482210200. Available at: https://www.routledge.com/Spatial-Point-Patterns-Methodology-and-Applications-with-R/Baddeley-Rubak-Turner/p/book/9781482210200.

Myllymäki, M., & Mrkvička, T. (2023). GET: Global envelopes in R. *arXiv:1911.06583 [stat.ME]*. doi:10.48550/arXiv.1911.06583.

Examples

```
# Note: The example below is provided for illustrative purposes and may take some time to run.

# Load the small example data
data(small_example_data)

# Load the example mark model that previously was trained on the small example data
file_path <- system.file("extdata", "example_mark_model.rds", package = "ldmppr")
mark_model <- bundle::unbundle(readRDS(file_path))

# Load the raster files
raster_paths <- list.files(system.file("extdata", package = "ldmppr"),
                           pattern = "\\.tif$", full.names = TRUE)
raster_paths <- raster_paths[!grepl("_med\\.tif$", raster_paths)]
rasters <- lapply(raster_paths, terra::rast)

# Scale the rasters
scaled_raster_list <- scale_rasters(rasters)

# Generate the reference pattern
reference_data <- generate_mpp(
  locations = small_example_data[, c("x", "y")],
  marks = small_example_data$size,
  xy_bounds = c(0, 25, 0, 25)
)

# Define an anchor point
M_n <- as.matrix(small_example_data[1, c("x", "y")])

# Specify the estimated parameters of the self-correcting process
# Note: These would generally be estimated using estimate_parameters_sc
# or estimate_parameters_sc_parallel. These values are estimates from
# the small_example_data generating script.
estimated_parameters <- c(
  1.42936311, 8.59251417, 0.02153924, 1.89763856,
  2.33256061, 1.09522235, 2.66250000, 0.16499789
)

# Check the model fit
example_model_fit <- check_model_fit(
  reference_data = reference_data,
  t_min = 0,
  t_max = 1,
  sc_params = estimated_parameters,
  anchor_point = M_n,
  raster_list = scaled_raster_list,
  scaled_rasters = TRUE,
  mark_model = mark_model,
  xy_bounds = c(0, 25, 0, 25),
  include_comp_inds = TRUE,
  thinning = TRUE,
  correction = "none",
  competition_radius = 10,
  n_sim = 100,
  save_sims = FALSE,
  verbose = TRUE,
  seed = 90210
)

plot(example_model_fit$combined_env)
```

---

estimate_parameters_sc

Estimate parameters of the self-correcting model using log-likelihood optimization

---

Description

Estimate the parameters of the self-correcting model using the [nloptr::nloptr()] function given a formatted dataset.

Usage

```
estimate_parameters_sc(
  data,
  x_grid = NULL,
  y_grid = NULL,
  t_grid = NULL,
  parameter_inits = NULL,
  upper_bounds = NULL,
  opt_algorithm = "NLOPT_LN_SBPLX",
  nloptr_options = list(maxeval = 400, xtol_rel = 1e-05, maxtime = NULL),
  verbose = TRUE
)
```

Arguments

`data`	a matrix or data frame of times and locations in the form (time, x, y).
`x_grid`	a vector of grid values for x.

y_grid	a vector of grid values for y.
t_grid	a vector of grid values for t.
parameter_inits	a vector of parameter initialization values.
upper_bounds	a vector of upper bounds for time, x, and y.
opt_algorithm	the NLopt algorithm to use for optimization.
nloptr_options	a list of named options for [nloptr::nloptr()] including "maxeval", "xtol_rel", and "maxtime".
verbose	‘TRUE‘ or ‘FALSE‘ indicating whether to show progress of optimization.

Details

This function estimates the parameters of the self-correcting model presented in Møller et al. (2016) using the full likelihood. Details regarding the self-correcting model and the estimation procedure can be found in the references.

Value

an [nloptr::nloptr()] object with details of the optimization including solution.

References

Møller, J., Ghorbani, M., & Rubak, E. (2016). Mechanistic spatio-temporal point process models for marked point processes, with a view to forest stand data. *Biometrics*, 72(3), 687–696. doi:10.1111/biom.12466.

Examples

```
# Load the small example data
data(small_example_data)
small_example_data <- small_example_data %>%
  dplyr::mutate(time = power_law_mapping(size, .5)) %>%
  dplyr::select(time, x, y)

# Define the grid values
x_grid <- seq(0, 25, length.out = 5)
y_grid <- seq(0, 25, length.out = 5)
t_grid <- seq(0, 1, length.out = 5)

# Define the parameter initialization values
parameter_inits <- c(1.5, 8.5, .015, 1.5, 3.2, .75, 3, .08)

# Define the upper bounds
upper_bounds <- c(1, 25, 25)

# Estimate the parameters
estimate_parameters_sc(
  data = small_example_data,
  x_grid = x_grid,
  y_grid = y_grid,
  t_grid = t_grid,
  parameter_inits = parameter_inits,
  upper_bounds = upper_bounds,
  opt_algorithm = "NLOPT_LN_SBPLX",
  nloptr_options = list(
    maxeval = 25,
    xtol_rel = 1e-2
  ),
  verbose = TRUE
)
```

---

estimate_parameters_sc_parallel

Estimate parameters of the self-correcting model using log-likelihood maximization in parallel

---

Description

Estimate the parameters of the self-correcting model using [nloptr::nloptr()] given a set of delta values. Makes use of parallel computation to allow the user to identify the optimal delta value more quickly given available computational resources.

Usage

```
estimate_parameters_sc_parallel(
  data,
  x_grid = NULL,
  y_grid = NULL,
  t_grid = NULL,
  delta_values = NULL,
  parameter_inits = NULL,
  upper_bounds = NULL,
  opt_algorithm = "NLOPT_LN_SBPLX",
  nloptr_options = list(maxeval = 400, xtol_rel = 1e-05, maxtime = NULL),
  verbose = TRUE,
  num_cores = parallel::detectCores() - 1,
  set_future_plan = FALSE
)
```

Arguments

`data`	a data frame of locations and sizes labelled ("x", "y", "size").
`x_grid`	a vector of grid values for x.
`y_grid`	a vector of grid values for y.
`t_grid`	a vector of grid values for t.

`delta_values`	a vector of delta values.
`parameter_inits`	a vector of parameter initialization values.
`upper_bounds`	a vector of upper bounds for time, x, and y.
`opt_algorithm`	the NLopt algorithm to use for maximization.
`nloptr_options`	a list of named options for nloptr including "maxeval", "xtol_rel", and "maxtime".
`verbose`	‘TRUE‘ or ‘FALSE‘ indicating whether to show progress of optimization.
`num_cores`	number of cores to use in parallel estimation.
`set_future_plan`	‘TRUE‘ or ‘FALSE‘ indicating whether to change the parallelization plan for use in the function.

Details

Details regarding the self-correcting model and the estimation procedure can be found in the references.

Value

a list containing the optimal obtained values and the full [nloptr::nloptr()] results for all provided delta values.

References

Møller, J., Ghorbani, M., & Rubak, E. (2016). Mechanistic spatio-temporal point process models for marked point processes, with a view to forest stand data. *Biometrics*, 72(3), 687–696. doi:10.1111/biom.12466.

Examples

```
# Note: This function is designed to be run in parallel and may be computationally intensive.


# Load the small example data
data(small_example_data)

# Define the grid values
x_grid <- seq(0, 25, length.out = 5)
y_grid <- seq(0, 25, length.out = 5)
t_grid <- seq(0, 1, length.out = 5)

# Define the delta values
delta_values <- seq(0.25, 1, by = 0.25)

# Define the parameter initialization values
parameter_inits <- c(1.5, 8.5, .015, 1.5, 3.2, .75, 3, .08)

# Define the upper bounds
```

```
upper_bounds <- c(1, 25, 25)

# Estimate the parameters in parallel
estimate_parameters_sc_parallel(
  data = small_example_data,
  x_grid = x_grid,
  y_grid = y_grid,
  t_grid = t_grid,
  delta_values = delta_values,
  parameter_inits = parameter_inits,
  upper_bounds = upper_bounds,
  opt_algorithm = "NLOPT_LN_SBPLX",
  nloptr_options = list(
    maxeval = 50,
    xtol_rel = 1e-2
  ),
  verbose = TRUE,
  set_future_plan = TRUE
)
```

extract_covars   *Extract covariate values from a set of rasters*

Description

Extract covariate values from a set of rasters

Usage

```
extract_covars(locations, raster_list)
```

Arguments

`locations`	a data frame of (x,y) locations with names "x" and "y".
`raster_list`	a list of raster objects.

Value

a matrix of covariates drawn from the provided rasters.

Examples

```
# Load example raster data
raster_paths <- list.files(system.file("extdata", package = "ldmppr"),
  pattern = "\\.tif$", full.names = TRUE
)
rasters <- lapply(raster_paths, terra::rast)
```

```
# Scale the rasters
scaled_raster_list <- scale_rasters(rasters)

# Load example locations
locations <- small_example_data %>%
  dplyr::select(x, y) %>%
  as.matrix()

# Extract covariates
example_covars <- extract_covars(locations, scaled_raster_list)
head(example_covars)
```

`generate_mpp` *Generate a marked process given locations and marks*

Description

Creates an object of class "ppp" that represents a marked point pattern in the two-dimensional plane.

Usage

```
generate_mpp(locations, marks = NULL, xy_bounds = NULL)
```

Arguments

`locations`	a data frame of (x,y) locations with names "x" and "y".
`marks`	a vector of marks.
`xy_bounds`	a vector of domain bounds (2 for x, 2 for y).

Value

a ppp object with marks.

Examples

```
# Load example data
data(small_example_data)

# Generate a marked point process
generate_mpp(
  locations = small_example_data %>% dplyr::select(x, y),
  marks = small_example_data$size,
  xy_bounds = c(0, 25, 0, 25)
)
```

## `medium_example_data` *Medium Example Data*

Description

A medium sized example dataset consisting of 111 observations in a (50m x 50m) square domain.

Usage

```
data("medium_example_data")
```

Format

## ‘medium_example_data‘ A data frame with 111 rows and 3 columns:

x x coordinate

y y coordinate

size Size ...

Details

The dataset was generated using the Snodgrass dataset available at https://data.ess-dive.lbl.gov/view/doi:10.15485/2476543. The full code to generate this dataset is available in the package’s ‘data_raw‘ directory.

Source

Real example dataset. Code to generate it can be found in ‘data_raw/medium_example_data.R‘.

`plot_mpp` *Plot a marked point process*

Description

Plot a marked point process

Usage

```
plot_mpp(mpp_data, pattern_type)
```

Arguments

`mpp_data`	ppp object with marks or data frame with columns (x, y, size).
`pattern_type`	type of pattern to plot ("reference" or "simulated").

Value

a ggplot object of the marked point process.

Examples

```
# Load example data
data(small_example_data)
mpp_data <- generate_mpp(
  locations = small_example_data %>% dplyr::select(x, y),
  marks = small_example_data$size,
  xy_bounds = c(0, 25, 0, 25)
)

# Plot the marked point process
plot_mpp(mpp_data, pattern_type = "reference")
```

---

power_law_mapping *Gentle decay (power-law) mapping function from sizes to arrival times*

Description

Gentle decay (power-law) mapping function from sizes to arrival times

Usage

```
power_law_mapping(sizes, delta)
```

Arguments

sizes	vector of sizes to be mapped to arrival times.
delta	numeric value (greater than 0) for the exponent in the mapping function.

Value

vector of arrival times.

Examples

```
# Generate a vector of sizes
sizes <- runif(100, 0, 100)

# Map the sizes to arrival times using a power-law mapping with delta = .5
power_law_mapping(sizes, .5)
```

predict_marks — *Predict values from the mark distribution*

Description

Predict values from the mark distribution

Usage

```
predict_marks(
  sim_realization,
  raster_list = NULL,
  scaled_rasters = FALSE,
  mark_model = NULL,
  xy_bounds = NULL,
  include_comp_inds = FALSE,
  competition_radius = 15,
  correction = "none"
)
```

Arguments

`sim_realization`	a data frame containing a thinned or unthinned realization from `simulate_sc`.
`raster_list`	a list of raster objects.
`scaled_rasters`	‘TRUE‘ or ‘FALSE‘ indicating whether the rasters have been scaled.
`mark_model`	a model object (typically from `train_mark_model`).
`xy_bounds`	a vector of domain bounds (2 for x, 2 for y).
`include_comp_inds`	‘TRUE‘ or ‘FALSE‘ indicating whether to generate and use competition indices as covariates.
`competition_radius`	distance for competition radius if `include_comp_inds` is ‘TRUE‘.
`correction`	type of correction to apply ("none" or "toroidal").

Value

a vector of predicted mark values.

Examples

```
# Simulate a realization
generating_parameters <- c(2, 8, .02, 2.5, 3, 1, 2.5, .2)
M_n <- matrix(c(10, 14), ncol = 1)
generated_locs <- simulate_sc(
  t_min = 0,
  t_max = 1,
  sc_params = generating_parameters,
  anchor_point = M_n,
  xy_bounds = c(0, 25, 0, 25)
)

# Load the raster files
raster_paths <- list.files(system.file("extdata", package = "ldmppr"),
  pattern = "\\.tif$", full.names = TRUE
)
rasters <- lapply(raster_paths, terra::rast)

# Scale the rasters
scaled_raster_list <- scale_rasters(rasters)

# Load the example mark model
file_path <- system.file("extdata", "example_mark_model.rds", package = "ldmppr")
example_mark_model <- readRDS(file_path)

# Unbundle the model
mark_model <- bundle::unbundle(example_mark_model)

# Predict the mark values
predict_marks(
  sim_realization = generated_locs$thinned,
  raster_list = scaled_raster_list,
  scaled_rasters = TRUE,
  mark_model = mark_model,
  xy_bounds = c(0, 25, 0, 25),
  include_comp_inds = TRUE,
  competition_radius = 10,
  correction = "none"
)
```

scale_rasters *Scale a set of rasters*

Description

Scale a set of rasters

Usage

```
scale_rasters(raster_list, reference_resolution = NULL)
```

Arguments

`raster_list`	a list of raster objects.
`reference_resolution`	the resolution to resample the rasters to.

Value

a list of scaled raster objects.

Examples

```
# Create two example rasters
rast_a <- terra::rast(
  ncol = 10, nrow = 10,
  xmin = 0, xmax = 10,
  ymin = 0, ymax = 10,
  vals = runif(100)
)

rast_b <- terra::rast(
  ncol = 10, nrow = 10,
  xmin = 0, xmax = 10,
  ymin = 0, ymax = 10,
  vals = runif(100)
)

# Scale example rasters in a list
rast_list <- list(rast_a, rast_b)
scale_rasters(rast_list)
```

simulate_mpp *Simulate a realization of a location dependent marked point process*

Description

Simulate a realization of a location dependent marked point process

Usage

```
simulate_mpp(
  sc_params = NULL,
  t_min = 0,
  t_max = 1,
  anchor_point = NULL,
  raster_list = NULL,
  scaled_rasters = FALSE,
  mark_model = NULL,
  xy_bounds = NULL,
  include_comp_inds = FALSE,
  competition_radius = 15,
  correction = "none",
  thinning = TRUE
)
```

Arguments

`sc_params`	vector of parameter values corresponding to (alpha_1, beta_1, gamma_1, alpha_2, beta_2, alpha_3, beta_3, gamma_3).
`t_min`	minimum value for time.
`t_max`	maximum value for time.
`anchor_point`	vector of (x,y) coordinates of point to condition on.
`raster_list`	list of raster objects.
`scaled_rasters`	‘TRUE‘ or ‘FALSE‘ indicating whether the rasters have been scaled.
`mark_model`	a model object (typically from `train_mark_model`).
`xy_bounds`	a vector of domain bounds (2 for x, 2 for y).
`include_comp_inds`	‘TRUE‘ or ‘FALSE‘ indicating whether to generate and use competition indices as covariates.
`competition_radius`	distance for competition radius if `include_comp_inds` is ‘TRUE‘.
`correction`	type of correction to apply ("none" or "toroidal").
`thinning`	‘TRUE‘ or ‘FALSE‘ indicating whether to thin the realization.

Value

a list containing the marked point process realization and the data frame of the realization.

Examples

```
# Specify the generating parameters of the self-correcting process
generating_parameters <- c(2, 8, .02, 2.5, 3, 1, 2.5, .2)

# Specify an anchor point
M_n <- matrix(c(10, 14), ncol = 1)

# Load the raster files
raster_paths <- list.files(system.file("extdata", package = "ldmppr"),
  pattern = "\\.tif$", full.names = TRUE
)
raster_paths <- raster_paths[!grepl("_med\\.tif$", raster_paths)]
rasters <- lapply(raster_paths, terra::rast)

# Scale the rasters
scaled_raster_list <- scale_rasters(rasters)

# Load the example mark model
file_path <- system.file("extdata", "example_mark_model.rds", package = "ldmppr")
mark_model <- bundle::unbundle(readRDS(file_path))

# Simulate a realization
example_mpp <- simulate_mpp(
  sc_params = generating_parameters,
  t_min = 0,
  t_max = 1,
  anchor_point = M_n,
  raster_list = scaled_raster_list,
  scaled_rasters = TRUE,
  mark_model = mark_model,
  xy_bounds = c(0, 25, 0, 25),
  include_comp_inds = TRUE,
  competition_radius = 10,
  correction = "none",
  thinning = TRUE
)

# Plot the realization
plot_mpp(example_mpp$mpp, pattern_type = "simulated")
```

`simulate_sc` *Simulate from the self-correcting model*

Description

Allows the user to simulate a realization from the self-correcting model given a set of parameters and a point to condition on.

Usage

```
simulate_sc(
  t_min = 0,
  t_max = 1,
  sc_params = NULL,
  anchor_point = NULL,
  xy_bounds = NULL
)
```

Arguments

`t_min`	minimum value for time.
`t_max`	maximum value for time.
`sc_params`	vector of parameter values corresponding to (alpha_1, beta_1, gamma_1, alpha_2, beta_2, alpha_3, beta_3, gamma_3).
`anchor_point`	vector of (x,y) coordinates of point to condition on.
`xy_bounds`	a vector of domain bounds (2 for x, 2 for y).

Value

a list containing the thinned and unthinned simulation realizations.

Examples

```
# Specify the generating parameters of the self-correcting process
generating_parameters <- c(2, 8, .02, 2.5, 3, 1, 2.5, .2)

# Specify an anchor point
M_n <- matrix(c(10, 14), ncol = 1)

# Simulate the self-correcting process
generated_locs <- simulate_sc(
  t_min = 0,
  t_max = 1,
  sc_params = generating_parameters,
  anchor_point = M_n,
  xy_bounds = c(0, 25, 0, 25)
)
```

## `small_example_data` *Small Example Data*

Description

A small example dataset for testing and examples consisting of 121 observations in a (25m x 25m) square domain.

Usage

```
data("small_example_data")
```

Format

## ‘small_example_data‘ A data frame with 121 rows and 3 columns:

x x coordinate

y y coordinate

size Size ...

Details

The dataset was generated using the example raster data and an exponential decay size function.

The full code to generate this dataset is available in the package’s ‘data_raw‘ directory.

Source

Simulated dataset. Code to generate it can be found in ‘data_raw/small_example_data.R‘.

`train_mark_model` *Train a flexible model for the mark distribution*

Description

Trains a predictive model for the mark distribution of a spatio-temporal process. Allows the user to incorporate location specific information and competition indices as covariates in the mark model.

Usage

```
train_mark_model(
  data,
  raster_list = NULL,
  scaled_rasters = FALSE,
  model_type = "xgboost",
  xy_bounds = NULL,
  save_model = FALSE,
  save_path = NULL,
  parallel = TRUE,
  include_comp_inds = FALSE,
  competition_radius = 15,
  correction = "none",
  selection_metric = "rmse",
  cv_folds = 5,
  tuning_grid_size = 200,
  verbose = TRUE
)
```

Arguments

`data`	a data frame containing named vectors x, y, size, and time.
`raster_list`	a list of raster objects.
`scaled_rasters`	‘TRUE‘ or ‘FALSE‘ indicating whether the rasters have been scaled.
`model_type`	the machine learning model type ("xgboost" or "random_forest").
`xy_bounds`	a vector of domain bounds (2 for x, 2 for y).
`save_model`	‘TRUE‘ or ‘FALSE‘ indicating whether to save the generated model.
`save_path`	path for saving the generated model.
`parallel`	‘TRUE‘ or ‘FALSE‘ indicating whether to use parallelization in model training.
`include_comp_inds`	‘TRUE‘ or ‘FALSE‘ indicating whether to generate and use competition indices as covariates.
`competition_radius`	distance for competition radius if `include_comp_inds` is ‘TRUE‘.
`correction`	type of correction to apply ("none", "toroidal", or "truncation").

`selection_metric`	metric to use for identifying the optimal model ("rmse" or "mae").
`cv_folds`	number of cross-validation folds to use in model training.
`tuning_grid_size`	size of the tuning grid for hyperparameter tuning.
`verbose`	‘TRUE‘ or ‘FALSE‘ indicating whether to show progress of model training.

Value

a list containing the raw trained model and a bundled model object.

Examples

```
# Load example raster data
raster_paths <- list.files(system.file("extdata", package = "ldmppr"),
  pattern = "\\.tif$", full.names = TRUE
)
raster_paths <- raster_paths[!grepl("_med\\.tif$", raster_paths)]
rasters <- lapply(raster_paths, terra::rast)

# Scale the rasters
scaled_raster_list <- scale_rasters(rasters)

# Load example locations
locations <- small_example_data %>%
  dplyr::mutate(time = power_law_mapping(size, .5))

# Train the model
train_mark_model(
  data = locations,
  raster_list = scaled_raster_list,
  scaled_rasters = TRUE,
  model_type = "xgboost",
  xy_bounds = c(0, 25, 0, 25),
  parallel = FALSE,
  include_comp_inds = FALSE,
  competition_radius = 10,
  correction = "none",
  selection_metric = "rmse",
  cv_folds = 3,
  tuning_grid_size = 2,
  verbose = TRUE
)
```

Index