Package 'h3sdm'

June 15, 2026

Type Package

Title Species Distribution Modeling with H3 Grids

Version 0.1.6

Description Provides tools for species distribution modeling using H3 hexagonal grids (Uber Technologies Inc., 2022, <https://h3geo.org>). Facilitates retrieval of species occurrence records, generation of H3 grids, computation of landscape metrics, and preparation of spatial data for modern species distribution models workflows. Designed for biodiversity and landscape ecology research.

URL https://github.com/ManuelSpinola/h3sdm

BugReports https://github.com/ManuelSpinola/h3sdm/issues

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1)

Config/Needs/website tidyverse/tidytemplate

Imports stats, sf, dplyr, purrr, tibble, rlang, terra, spatialsample, recipes, rsample, tune, workflows, yardstick, ecospat, DALEX, stacks, h3jsr, landscapemetrics, rbiodatacr, spocc, vip, exactextractr, tidyr, cli

Suggests ggplot2, paisaje, knitr, rmarkdown, here, themis, DALEXtra, ingredients, tidyterra, tidymodels, workflowsets, ranger, xgboost, ggbrick, parsnip, tidyverse, geodata

VignetteBuilder knitr

LazyData true

Language en-US

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Manuel Spínola [aut, cre]

Maintainer Manuel Spínola <mspinola10@gmail.com>

Repository CRAN

Date/Publication 2026-06-15 07:50:25 UTC

Contents

bioclim_current . . . 2
bioclim_future . . . 3
cr_outline . . . 4
cr_outline_c . . . 5
h3sdm_aoa . . . 6
h3sdm_calculate_it_metrics . . . 7
h3sdm_classify . . . 9
h3sdm_compare_models . . . 10
h3sdm_count_from_records . . . 11
h3sdm_data . . . 13
h3sdm_eval_metrics . . . 14
h3sdm_explain . . . 15
h3sdm_extract_cat . . . 16
h3sdm_extract_num . . . 18
h3sdm_filter_outliers . . . 19
h3sdm_filter_range . . . 21
h3sdm_fit_model . . . 22
h3sdm_fit_models . . . 23
h3sdm_get_grid . . . 25
h3sdm_get_records . . . 26
h3sdm_get_records_by_hexagon . . . 27
h3sdm_pa . . . 29
h3sdm_pa_from_records . . . 31
h3sdm_predict . . . 32
h3sdm_predictors . . . 33
h3sdm_pres . . . 34
h3sdm_pres_from_sf . . . 36
h3sdm_recipe . . . 37
h3sdm_recipe_gam . . . 38
h3sdm_spatial_cv . . . 40
h3sdm_stack_fit . . . 41
h3sdm_workflow . . . 41
h3sdm_workflows . . . 43
h3sdm_workflow_gam . . . 44
records . . . 46

Index **47**

`bioclim_current` *Current bioclimatic raster*

Description

A GeoTIFF with current bioclimatic variables for Costa Rica.

Format

GeoTIFF file, readable with `terra::rast()`.

Details

This file is stored in `inst/extdata/` and can be accessed with: `terra::rast(system.file("extdata", "bioclim_current.tif", package = "h3sdm"))`

Examples

```
library(terra)
bio <- terra::rast(system.file("extdata", "bioclim_current.tif", package = "h3sdm"))
```

---

bioclim_future *Future bioclimatic raster*

Description

A GeoTIFF with projected bioclimatic variables for Costa Rica.

Format

GeoTIFF file, readable with `terra::rast()`.

Details

This dataset corresponds to the climate projection:

- Model: INM-CM4-8
- Scenario: SSP1-2.6
- Period: 2021-2040

The file is stored in `inst/extdata/` and can be accessed with: `terra::rast(system.file("extdata", "bioclim_future.tif", package = "h3sdm"))`

Examples

```
library(terra)
bio <- terra::rast(system.file("extdata", "bioclim_future.tif", package = "h3sdm"))
```

`cr_outline` *Costa Rica Full Outline (Continental + Islands)*

Description

An `sf` multipolygon containing the full outline of Costa Rica, derived from GADM 4.1. Includes the continental landmass, the Isla del Coco (~550 km offshore in the Pacific Ocean), and all other minor oceanic islands.

For the continental outline only (without islands), see `cr_outline_c`.

Usage

```
cr_outline
```

Format

An `sf` object with 1 feature and 1 column:

geometry MULTIPOLYGON in WGS 84 (EPSG:4326) representing the full national territory of Costa Rica including all islands.

Details

When to use this vs cr_outline_c:

Use `cr_outline` when your analysis requires the full national territory. Use `cr_outline_c` for mainland ecological analyses where oceanic islands would distort results (species distribution models, landscape metrics, climate extraction).

Reproducibility:

Generated with `data-raw/cr_outline.R`. To regenerate:

```
source("data-raw/cr_outline.R")
```

Source

Global Administrative Areas (GADM) version 4.1. Downloaded via `geodata::gadm("CRI", level = 0)`. https://gadm.org

See Also

- `cr_outline_c` – continental outline only (no islands).

Examples

```
data(cr_outline)
plot(sf::st_geometry(cr_outline), main = "Costa Rica (full territory)")

## Not run:
# Compare continental vs full
par(mfrow = c(1, 2))
plot(sf::st_geometry(cr_outline_c), main = "Continental")
plot(sf::st_geometry(cr_outline),   main = "Full territory")

## End(Not run)
```

---

cr_outline_c *Costa Rica Continental Outline*

Description

An `sf` polygon containing the continental outline of Costa Rica, derived from GADM 4.1. The Isla del Coco and all other minor oceanic islands have been removed, retaining only the largest polygon (the continental landmass).

For the full outline including all islands, see `cr_outline`.

Usage

```
cr_outline_c
```

Format

An `sf` object with 1 feature and 1 column:

geometry POLYGON in WGS 84 (EPSG:4326) with 30,261 vertices, representing the continental outline of Costa Rica.

Details

Island removal:

Costa Rica includes the Isla del Coco (~550 km offshore in the Pacific), which is excluded here. The continental polygon is obtained by casting the GADM multipolygon to individual polygons and retaining the one with the largest area. For analyses requiring all national territory use `cr_outline`.

Reproducibility:

Generated with `data-raw/cr_outline.R`. To regenerate:

```
source("data-raw/cr_outline.R")
```

Source

Global Administrative Areas (GADM) version 4.1. Downloaded via `geodata::gadm("CRI", level = 0)`. https://gadm.org

See Also

- `cr_outline` – full outline including all islands.

Examples

```
data(cr_outline_c)
plot(sf::st_geometry(cr_outline_c), main = "Costa Rica (continental)")
```

---

h3sdm_aoa *Area of Applicability (AOA) of spatial prediction models*

Description

Estimates the Dissimilarity Index (DI) and the Area of Applicability (AOA) for new data given the training data and a fitted model. This function is designed to be applied directly to the output of `h3sdm_predict()`, so that both the predicted values and the AOA are available in a single `sf` object ready for mapping.

Usage

```
h3sdm_aoa(newdata, train, fit_object, cv = NULL, verbose = TRUE)
```

Arguments

`newdata`	An `sf` object, typically the direct output of `h3sdm_predict()`, containing the predictor variables and a `prediction` column.
`train`	An `sf` object, typically the output of `h3sdm_data()`, containing the training observations with the same predictor variables used in `fit_object`.
`fit_object`	The list returned by `h3sdm_fit_model()`, used to extract the model formula (predictor variable names) and variable importances when available.
`cv`	An `rset` object returned by `h3sdm_spatial_cv()`, used to extract cross-validation fold assignments. If `NULL` (default), Leave-One-Out (LOO) cross-validation is used.
`verbose`	Logical. Should progress messages be printed? Default `TRUE`.

Details

The algorithm follows Meyer & Pebesma (2021). Predictor variables are extracted automatically from the model formula inside `fit_object`. They are standardized using z-score scaling computed from `train`, then optionally weighted by variable importance. The mean nearest-neighbor distance among training points (`trainDist_avrgmean`) is used to normalize DI values. The AOA threshold is the maximum cross-validated training DI after removing outliers with Tukey's rule (Q3 + 1.5 * IQR). Locations with `DI <= threshold` are inside the AOA; locations above the threshold should be interpreted with caution.

Variable importance is extracted automatically for `ranger` and `xgboost` models via `vip::vi()`. For GAM models, or when importance cannot be extracted, all variables receive equal weight.

Value

The input `newdata` `sf` object with two additional columns:

`DI` Numeric. Dissimilarity Index for each observation. Values near 0 indicate high similarity to the training data; larger values indicate increasing dissimilarity.

`AOA` Integer. `1` = inside the AOA; `0` = outside the AOA.

References

Meyer, H., Pebesma, E. (2021): Predicting into unknown space? Estimating the area of applicability of spatial prediction models. *Methods in Ecology and Evolution* 12: 1620–1633. doi:10.1111/2041-210X.13650

See Also

`h3sdm_predict()`, `h3sdm_fit_model()`, `h3sdm_spatial_cv()`, `h3sdm_data()`

Examples

```
## Not run:
cv      <- h3sdm_spatial_cv(dat, method = "block")
fit     <- h3sdm_fit_model(workflow, cv)
pred    <- h3sdm_predict(fit, new_data = h7)
result  <- h3sdm_aoa(pred, train = dat, fit_object = fit, cv = cv)

## End(Not run)
```

---

h3sdm_calculate_it_metrics

Calculate Information Theory Landscape Metrics for Hexagonal Grid

---

Description

Calculates 5 Information Theory (IT)-based landscape metrics (`condent`, `ent`, `joinent`, `mutinf`, `relmutinf`) for each hexagon in a given H3 hexagonal grid.

Usage

```
h3sdm_calculate_it_metrics(landscape_raster, sf_grid)
```

Arguments

`landscape_raster`	A categorical SpatRaster containing land-cover data.
`sf_grid`	An `sf` object containing the hexagonal grid with species or land-cover data.

Details

This function computes landscape metrics using the `landscapemetrics::sample_lsm()` workflow. The results are pivoted to a wide format for easy use.

Value

An `sf` object containing the input hex grid with new columns for each calculated metric.

References

Hesselbarth et al., 2019. landscapemetrics: an open-source R tool to calculate landscape metrics. Ecography 42: 1648–1657.

Nowosad & Stepinski, 2019. Information theory as a consistent framework for landscape patterns. doi:10.1007/s1098001900830x

Examples

```
library(sf)
library(terra)

# Create a categorical SpatRaster (land-cover map)
landscape_raster <- terra::rast(
  nrows = 30, ncols = 30,
  xmin = -85.0, xmax = -83.0,
  ymin = 9.0,  ymax = 11.0,
  crs = "EPSG:4326"
)
terra::values(landscape_raster) <- sample(1:4, terra::ncell(landscape_raster),
                                          replace = TRUE)
names(landscape_raster) <- "landcover"

# Create a simple hexagon grid as sf polygons
hex_grid <- sf::st_make_grid(
  sf::st_as_sfc(sf::st_bbox(c(
    xmin = -84.5, xmax = -83.5,
    ymin = 9.5,  ymax = 10.5
  ), crs = sf::st_crs(4326))),
  n = c(3, 3),
  square = FALSE
)
```

```
sf_grid <- sf::st_sf(h3_address = paste0("hex_", seq_along(hex_grid)),
                     geometry = hex_grid)

# Calculate Information Theory (IT) landscape metrics per hexagon
result_sf <- h3sdm_calculate_it_metrics(landscape_raster, sf_grid)
head(result_sf)
```

`h3sdm_classify` *Classify predictions based on an optimal threshold*

Description

Converts continuous probability predictions into binary presence/absence based on a specified threshold.

Usage

```
h3sdm_classify(predictions_sf, threshold)
```

Arguments

`predictions_sf`	An `sf` object containing a numeric column named `prediction`, typically produced by `h3sdm_predict()`.
`threshold`	A numeric value representing the probability threshold (e.g., `0.45`) above which predictions are classified as presence (1).

Details

This function is useful for converting continuous probability outputs into binary presence/absence data for mapping or model evaluation purposes.

Value

An `sf` object with the same geometry and all original columns, plus a new integer column `predicted_presence` with values `0` (absence) or 1 (presence).

Examples

```
## Not run:
library(sf)
library(dplyr)

# Crear un sf de ejemplo
df <- data.frame(
  id = 1:5,
  prediction = c(0.2, 0.6, 0.45, 0.8, 0.3),
  lon = c(-75, -74, -73, -72, -71),
```

```
  lat = c(10, 11, 12, 13, 14)
)

df_sf <- st_as_sf(df, coords = c("lon", "lat"), crs = 4326)

# Clasificar usando un umbral
classified_sf <- h3sdm_classify(df_sf, threshold = 0.5)

# Revisar resultados
print(classified_sf)

## End(Not run)
```

`h3sdm_compare_models` *Compare multiple H3SDM species distribution models*

Description

Computes and combines performance metrics for multiple species distribution models created with `h3sdm_fit_models()` or similar workflows. Metrics include standard yardstick metrics (ROC AUC, TSS, Boyce index, etc.). Returns a tibble summarizing model performance.

Usage

```
h3sdm_compare_models(h3sdm_results)
```

Arguments

`h3sdm_results`	A list or workflow set containing fitted models with a `metrics` tibble. Typically, this object is the output of `h3sdm_fit_models()`.

Value

A tibble with one row per model per metric, containing:

model Model name

.metric Metric name (ROC AUC, TSS, Boyce, etc.)

.estimator Metric type (usually "binary")

mean Metric value

Examples

```
# Minimal reproducible example
example_metrics <- tibble::tibble(
  model = c("model1", "model2"),
  .metric = c("roc_auc", "tss_max"),
  .estimator = c("binary", "binary"),
```

```
  mean = c(0.85, 0.7)
)
example_results <- list(metrics = example_metrics)
h3sdm_compare_models(example_results)
```

`h3sdm_count_from_records`

Generate species richness or abundance count dataset from records

---

Description

Takes a user-provided dataset with presence records (from Excel, fieldwork, acoustic detections, camera traps, or any other source) and generates a hexagonal grid with counts (species richness, total detections, or individuals) ready for analysis with h3sdm. The input can be a `data.frame` with coordinate columns or an `sf` object. Coordinates are assumed to be in WGS84 (EPSG:4326).

Usage

```
h3sdm_count_from_records(
  records,
  aoi_sf,
  res = 7,
  expand_factor = 0.1,
  lon_col = "x",
  lat_col = "y",
  species_col = NULL,
  count_type = c("richness", "detections", "individuals"),
  presence_col = NULL,
  abundance_col = NULL,
  confidence_col = NULL,
  confidence_threshold = NULL,
  date_col = NULL,
  date_min = NULL,
  date_max = NULL
)
```

Arguments

`records`	`data.frame` or `sf` object containing records.
`aoi_sf`	`sf` AOI (area of interest) polygon.
`res`	`integer` H3 resolution for the hexagonal grid. Default `7`.
`expand_factor`	`numeric` Factor to expand AOI before creating hex grid. Default `0.1`.
`lon_col`	`character` Name of the longitude column. Ignored if `records` is already an `sf` object. Default `"x"`.

`lat_col` `character` Name of the latitude column. Ignored if `records` is already an `sf` object. Default `"y"`.

`species_col` `character` Name of the column containing species names. Required when `count_type = "richness"`.

`count_type` `character` One of `"richness"` (number of unique species per hexagon), `"detections"` (total number of records per hexagon), or `"individuals"` (sum of a numeric abundance column per hexagon). Default `"richness"`.

`presence_col` `character` Optional. Name of the column indicating presence (1) or absence (0). If provided, only records with value == 1 are used.

`abundance_col` `character` Required when `count_type = "individuals"`. Name of the column with individual counts to sum per hexagon.

`confidence_col` `character` Optional. Name of the column with detection confidence scores (numeric between 0 and 1). Useful for acoustic detection data (e.g. BirdNET output).

`confidence_threshold` `numeric` Optional. Minimum confidence score to retain a record. Ignored if `confidence_col` is `NULL`.

`date_col` `character` Optional. Name of the date column. The column must be of class `Date`. If your dates are stored as Excel numeric values, convert them first with `as.Date(datos$Fecha, origin = "1899-12-30")`.

`date_min` `character` or `Date` Optional. Minimum date to retain records (inclusive). Format `"YYYY-MM-DD"`.

`date_max` `character` or `Date` Optional. Maximum date to retain records (inclusive). Format `"YYYY-MM-DD"`.

Value

`sf` object with columns:

h3_address H3 index of the hexagon.

count Numeric count per hexagon (richness, detections, or individuals).

geometry MULTIPOLYGON of each hexagon.

Examples

```
data(cr_outline_c, package = "h3sdm")

my_records <- data.frame(
  x         = c(-84.1, -84.2, -83.9, -84.0, -84.1),
  y         = c(9.9, 10.1, 9.8, 9.95, 10.0),
  Especie   = c("Ara macao", "Ara macao", "Pharomachrus mocinno",
                "Tapirus bairdii", "Ara macao"),
  Presencia = c(1, 1, 1, 1, 0)
)

richness_hex <- h3sdm_count_from_records(
  records      = my_records,
```

```
  aoi_sf       = cr_outline_c,
  res          = 7,
  lon_col      = "x",
  lat_col      = "y",
  species_col  = "Especie",
  count_type   = "richness",
  presence_col = "Presencia"
)
```

`h3sdm_data` *Combine species and environmental data for SDMs using H3 grids*

Description

Combines species presence–absence data with environmental predictors. It also calculates centroid coordinates (x and y) for each hexagon grid cell.

Usage

```
h3sdm_data(pa_sf, predictors_sf)
```

Arguments

`pa_sf`	An `sf` object from `h3sdm_pa()` containing species presence–absence data.
`predictors_sf`	An `sf` object from `h3sdm_predictors()` containing environmental predictors.

Value

An `sf` object containing species presence–absence, environmental predictor variables, and centroid coordinates for each hexagon cell.

Examples

```
## Not run:
my_species_pa <- h3sdm_pa("Panthera onca", res = 6)
my_predictors <- h3sdm_predictors(my_species_pa)
combined_data <- h3sdm_data(my_species_pa, my_predictors)

## End(Not run)
```

`h3sdm_eval_metrics` *Evaluate performance metrics for a fitted H3SDM model*

Description

Computes a set of performance metrics for a single fitted species distribution model. Includes standard yardstick metrics such as ROC AUC, accuracy, sensitivity, specificity, F1-score, Kappa, as well as ecological metrics such as the True Skill Statistic (TSS) and Boyce index. This function is designed as a helper for evaluating models produced by `h3sdm_fit_model` or `h3sdm_fit_models`.

Usage

```
h3sdm_eval_metrics(
  fitted_model,
  presence_data = NULL,
  truth_col = "presence",
  pred_col = ".pred_1"
)
```

Arguments

`fitted_model`	A fitted model object, typically the output of `h3sdm_fit_model()`.
`presence_data`	Optional. An `sf` object or tibble containing presence locations used to compute the Boyce index. If not provided, the Boyce index will not be calculated.
`truth_col`	Character. Name of the column containing the true presence/absence values (default `"presence"`).
`pred_col`	Character. Name of the column containing predicted probabilities (default `".pred_1"`).

Details

This function centralizes model evaluation for a single fitted H3SDM model, combining both general classification metrics and ecological indices. It is especially useful for systematically comparing model performance across species or modeling approaches.

Value

A tibble with one row per metric, containing:

.metric Metric name (e.g., "roc_auc", "tss", "boyce").

.estimator Estimator type (usually "binary").

mean Metric value.

std_err Standard error (NA for TSS and Boyce).

conf_low Lower bound of the 95% confidence interval (NA for TSS and Boyce).

conf_high Upper bound of the 95% confidence interval (NA for TSS and Boyce).

Examples

```
## Not run:
# Assuming 'fitted' is the result of h3sdm_fit_model()
metrics <- h3sdm_eval_metrics(
  fitted_model = fitted,
  presence_data = presence_sf,
  truth_col = "presence",
  pred_col = ".pred_1"
)
print(metrics)

## End(Not run)
```

`h3sdm_explain` *Create a DALEX explainer for h3sdm workflows*

Description

Creates a DALEX explainer for a species distribution model fitted with `h3sdm_fit_model()`. Prepares response and predictor variables, ensuring that all columns used during model training (including `h3_address` and coordinates) are included. The explainer can be used for feature importance, model residuals, and other DALEX diagnostics.

Usage

```
h3sdm_explain(model, data, response = "presence", label = "h3sdm workflow")
```

Arguments

`model`	A fitted workflow returned by `h3sdm_fit_model()`.
`data`	A `data.frame` or `sf` object containing the original predictors and response variable. If an `sf` object, geometry is dropped automatically.
`response`	Character string specifying the name of the response column. Must be a binary factor or numeric vector (0/1). Defaults to `"presence"`.
`label`	Character string specifying a label for the explainer. Defaults to `"h3sdm workflow"`.

Value

An object of class `explainer` from the **DALEX** package, ready to be used with `feature_importance()`, `model_performance()`, `predict_parts()`, and other DALEX functions.

Examples

```
library(h3sdm)
library(DALEX)
library(parsnip)

dat <- data.frame(
  x1 = rnorm(20),
  x2 = rnorm(20),
  presence = factor(sample(0:1, 20, replace = TRUE))
)

model <- logistic_reg() |>
  fit(presence ~ x1 + x2, data = dat)

explainer <- h3sdm_explain(model, data = dat, response = "presence")
feature_importance(explainer)
```

---

h3sdm_extract_cat *Calculate Area Proportions for Categorical Raster Classes*

Description

Extracts and calculates the **area proportion** of each land-use/land-cover (LULC) category found within each input polygon of the `sf_hex_grid`. This function is tailored for categorical rasters and ensures accurate, sub-pixel weighted statistics.

Usage

```
h3sdm_extract_cat(spat_raster_cat, sf_hex_grid, proportion = TRUE)
```

Arguments

`spat_raster_cat`	A single-layer `SpatRaster` object containing categorical values (e.g., LULC classes).
`sf_hex_grid`	An `sf` object containing polygonal geometries (e.g., H3 hexagons). Must contain a column named `h3_address` for joining and grouping.
`proportion`	Logical. If `TRUE` (default), the output values are the proportion of the polygon area covered by each category (summing to 1 for covered area). If `FALSE`, the output is the raw sum of the coverage fraction (area).

Details

The function uses a custom function with `exactextractr::exact_extract` to perform three critical steps:

1. **Filtering NA/NaN:** Raster cells with missing values (NA) are explicitly excluded from the calculation, preventing the creation of a _prop_NaN column.
2. **Area Consolidation:** It sums the coverage fractions for all fragments belonging to the same category within the same hexagon, which is essential when polygons have been clipped or fragmented.
3. **Numerical Ordering:** The final columns are explicitly sorted based on the numerical value of the category (e.g., _prop_70 appears before _prop_80) to correct the default alphanumeric sorting behavior of tidyr::pivot_wider.

Value

An sf object identical to sf_hex_grid, but with new columns appended for each categorical value found in the raster. Column names follow the pattern <layer_name>_prop_<category_value>. Columns are **numerically ordered** by the category value.

Examples

```
library(sf)
library(terra)

# Create a simple categorical SpatRaster
lulc <- terra::rast(
  nrows = 20, ncols = 20,
  xmin = -85.0, xmax = -83.0,
  ymin = 9.0,  ymax = 11.0,
  crs = "EPSG:4326"
)
terra::values(lulc) <- sample(1:4, terra::ncell(lulc), replace = TRUE)
names(lulc) <- "landuse"

# Define categorical levels explicitly
levels(lulc) <- data.frame(
  value = 1:4,
  class = c("forest", "grassland", "urban", "water")
)

# Create a simple hexagon grid as sf polygons (smaller than raster extent)
hex_grid <- sf::st_make_grid(
  sf::st_as_sfc(sf::st_bbox(c(
    xmin = -84.5, xmax = -83.5,
    ymin = 9.5,  ymax = 10.5
  ), crs = sf::st_crs(4326))),
  n = c(3, 3),
  square = FALSE
)
h7 <- sf::st_sf(h3_address = paste0("hex_", seq_along(hex_grid)),
                geometry = hex_grid)

# Extract categorical raster values by hexagon
lulc_p <- h3sdm_extract_cat(lulc, h7, proportion = TRUE)
head(lulc_p)
```

h3sdm_extract_num *Extract Area-Weighted Mean from Numeric Raster Stack*

Description

Calculates the **area-weighted mean** value for each layer in a numeric SpatRaster (or single layer) within each polygon feature of an sf object. This function is designed to efficiently summarize continuous environmental variables (such as bioclimatic data) for predefined spatial units (e.g., H3 hexagons). It utilizes exactextractr to ensure highly precise zonal statistics by accounting for sub-pixel coverage fractions.

Usage

```
h3sdm_extract_num(spat_raster_multi, sf_hex_grid)
```

Arguments

spat_raster_multi	A SpatRaster object from the terra package. Must contain numeric layers (can be a single layer or a stack/brick).
sf_hex_grid	An sf object containing polygonal geometries (e.g., H3 hexagons). Must be a valid set of polygons for extraction.

Details

The function relies on exactextractr::exact_extract with fun = "weighted_mean" and weights = "area". This methodology is crucial for maintaining spatial accuracy when polygons are irregular or small relative to the raster resolution. A critical check (nrow match) is performed before binding columns to ensure data integrity and prevent misalignment errors.

Value

An sf object identical to sf_hex_grid, but with new columns appended. The new column names match the original SpatRaster layer names. The values represent the area-weighted mean for that variable within each polygon.

Examples

```
library(sf)
library(terra)

# Create a SpatRaster stack with two numeric layers (e.g., bioclimatic variables)
bio1 <- terra::rast(
  nrows = 10, ncols = 10,
  xmin = -84.5, xmax = -83.5,
  ymin = 9.5,  ymax = 10.5,
```

```
  crs = "EPSG:4326"
)
bio2 <- bio1
terra::values(bio1) <- runif(terra::ncell(bio1), 15, 30)
terra::values(bio2) <- runif(terra::ncell(bio2), 500, 3000)
names(bio1) <- "bio1_temp"
names(bio2) <- "bio12_precip"
bio <- c(bio1, bio2)

# Create a simple hexagon grid as sf polygons
hex_grid <- sf::st_make_grid(
  sf::st_as_sfc(sf::st_bbox(c(
    xmin = -84.5, xmax = -83.5,
    ymin = 9.5,  ymax = 10.5
  ), crs = sf::st_crs(4326))),
  n = c(3, 3),
  square = FALSE
)
h7 <- sf::st_sf(h3_address = paste0("hex_", seq_along(hex_grid)),
                geometry = hex_grid)

# Extract numeric raster values by hexagon (mean per cell)
bio_p <- h3sdm_extract_num(bio, h7)
head(bio_p)
```

`h3sdm_filter_outliers` *Filter environmental outliers from presence records using Mahalanobis distance*

Description

Identifies and removes presence records that are environmental outliers based on their Mahalanobis distance (D^2) to the centroid of the presence cloud in environmental space. Only presence records (`presence == "1"`) are evaluated; pseudo-absences are always retained unchanged.

Usage

```
h3sdm_filter_outliers(pa, vars_cov = NULL, threshold = 0.975)
```

Arguments

`pa`	An `sf` object produced by `h3sdm_pa_from_records()` or `h3sdm_pa()`, containing a `presence` column with values `"1"` (presence) and `"0"` (pseudo-absence), plus numeric covariate columns.
`vars_cov`	Character vector of covariate column names to use for the Mahalanobis distance calculation. Typically the bioclimatic variables (e.g. CHELSA). If `NULL` (default), all numeric columns except `h3_address`, `presence`, `x`, and `y` are used automatically.

`threshold` Numeric in (0, 1). Percentile of the chi-squared distribution used as the outlier threshold. Default `0.975`.

Details

The threshold is derived from the chi-squared distribution: `qchisq(threshold, df = length(vars_cov))`. A record is flagged as an outlier when its D^2 exceeds this value. The default `threshold = 0.975` corresponds to the standard 97.5th percentile used in SDM practice.

Why only presences? Pseudo-absences are algorithmically generated by `h3sdm` and carry no georeferencing or identification error. Filtering them would remove points at the environmental periphery – exactly those most informative for defining the species niche boundary.

Singular covariance matrix: When presences are too few or covariates are perfectly collinear the covariance matrix becomes singular and cannot be inverted. In that case the function returns the original PA unchanged with a warning.

Relationship with AOA: The AOA (see `h3sdm_aoa()`) evaluates prediction reliability *after* training. This function improves *input data quality before* training. Both are complementary: an unfiltered outlier can artificially expand the AOA into ecologically implausible areas.

Value

A list with four elements:

`pa_clean` `sf` object – the input PA dataset with outlier presences removed. Pseudo-absences are untouched.

`outliers_df` `data.frame` with the removed records and an additional column `mahal_d2` with their D^2 values. Empty `data.frame` if no outliers were detected.

`n_removed` Integer. Number of presence records removed.

`threshold_d2` Numeric. The D^2 threshold value used (`qchisq(threshold, df = k)`).

See Also

`h3sdm_pa_from_records()`, `h3sdm_aoa()`

Examples

```
## Not run:
# After generating the PA dataset:
pa <- h3sdm_pa_from_records(records, aoi_sf, res = 7L,
                            predictors_sf = cov_actual)

# Auto-detect covariates and filter with default threshold (97.5%)
result <- h3sdm_filter_outliers(pa)

# Inspect removed records
result$outliers_df

# Use the clean PA for modeling
dat <- h3sdm_data(result$pa_clean, pred_sf)
```

```
# Custom covariate selection and stricter threshold
result2 <- h3sdm_filter_outliers(pa,
                                 vars_cov  = c("bio1", "bio12", "bio15"),
                                 threshold = 0.95)

## End(Not run)
```

`h3sdm_filter_range` *Filter predictions outside the univariate range of training data*

Description

Adds a `range_filter` column to `newdata` indicating whether each observation falls within the univariate range of the training data for all specified variables. This function is complementary to `h3sdm_aoa()` and Mahalanobis distance filtering, as it detects extrapolation at the margins of individual variables that multivariate methods may not capture.

Usage

```
h3sdm_filter_range(newdata, train, variables)
```

Arguments

`newdata`	An `sf` object with predictions (output of `h3sdm_predict()` or `h3sdm_aoa()`).
`train`	An `sf` object with the training data used to fit the model.
`variables`	A character vector of variable names to check. Should match the covariates used in the model.

Details

This function implements univariate range filtering as a quality control step for spatial predictions. It complements multivariate methods such as the Area of Applicability (AOA) and Mahalanobis distance, which may not detect extrapolation when a prediction point lies at the margin of a single variable but within the multivariate space of the training data.

The three methods detect different types of extrapolation:

- **Range filter**: extrapolation in individual variables.
- **Mahalanobis distance**: points far from the multivariate centroid.
- **AOA**: combinations without analogues in training data.

Value

The `newdata` object with an additional integer column `range_filter`:

1 Observation is within the training range for all variables.

0 Observation is outside the training range for at least one variable.

Examples

```
## Not run:
aoa_result <- h3sdm_aoa(
  newdata    = prediccion_especies,
  train      = dat_modelo,
  fit_object = mod_gam,
  cv         = scv
)

aoa_result <- h3sdm_filter_range(
  newdata   = aoa_result,
  train     = dat_modelo,
  variables = c("bio12", "bio17", "bio8", "class_prop_1", "class_prop_11")
)

# Inspect filtered hexagons
table(aoa_result$range_filter)

# Mask predictions outside range
aoa_result <- aoa_result |>
  dplyr::mutate(prediction = ifelse(range_filter == 0, NA, prediction))

## End(Not run)
```

`h3sdm_fit_model` *Fits an SDM workflow to data using resampling and prepares it for stacking.*

Description

Fits a Species Distribution Model (SDM) workflow to resampling data (cross-validation). This function is the main training step and optionally configures the results to be used with the 'stacks' package. Supports both classification (presence/absence) and regression (count-based) models, detected automatically from the workflow mode.

Usage

```
h3sdm_fit_model(
  workflow,
  data_split,
  presence_data = NULL,
  truth_col = NULL,
  pred_col = NULL,
  for_stacking = FALSE,
  ...
)
```

Arguments

workflow	A 'workflow' object from tidymodels (e.g., GAM or Random Forest).
data_split	An 'rsplit' or 'rset' object (e.g., result of vfold_cv or spatial_block_cv).
presence_data	(Optional) Original presence data (used for extended metrics).
truth_col	Column name of the response variable. Defaults to "presence" for classification models and "count" for regression models.
pred_col	Column name for the prediction of the class of interest. Defaults to ".pred_1" for classification models and ".pred" for regression models.
for_stacking	Logical. If TRUE, uses control_stack_resamples() to save all workflow information required for the 'stacks' package. If FALSE, uses the standard control with save_pred = TRUE.
...	Arguments passed on to other functions (e.g., to tune::fit_resamples if needed).

Value

A list with three elements:

- cv_model: The result of fit_resamples().
- final_model: The model fitted to the entire training set (first split).
- metrics: Extended evaluation metrics (if presence_data is provided).

---

h3sdm_fit_models *Fit and evaluate multiple H3SDM species distribution models*

Description

Fits one or more species distribution models using tidymodels workflows and a specified resampling scheme, then computes standard metrics (ROC AUC, accuracy, sensitivity, specificity, F1-score, Kappa) along with TSS (True Skill Statistic) and the Boyce index for model evaluation. Returns both the fitted models and a comparative metrics table.

Usage

```
h3sdm_fit_models(
  workflows,
  data_split,
  presence_data = NULL,
  truth_col = "presence",
  pred_col = ".pred_1"
)
```

Arguments

workflows	A named list of tidymodels workflows created with h3sdm_workflow() or manually.
data_split	A resampling object (e.g., from vfold_cv() or h3sdm_spatial_cv()) for cross-validation.
presence_data	An sf object or tibble with presence locations to compute the Boyce index (optional).
truth_col	Character. Name of the column containing true presence/absence values (default "presence").
pred_col	Character. Name of the column containing predicted probabilities (default ".pred_1").

Value

A list with two elements:

models A list of fitted models returned by h3sdm_fit_model().

metrics A tibble with one row per model per metric, including standard yardstick metrics, TSS, and Boyce index.

Examples

```
## Not run:
# Example requires prepared recipes and resampling objects
mod_log <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

mod_rf <- rand_forest() %>%
  set_engine("ranger") %>%
  set_mode("classification")

workflows_list <- list(
  logistic = h3sdm_workflow(mod_log, my_recipe),
  rf       = h3sdm_workflow(mod_rf, my_recipe)
)

results <- h3sdm_fit_models(
  workflows     = workflows_list,
  data_split    = my_cv_folds,
  presence_data = presence_sf
)
metrics_table <- results$metrics

## End(Not run)
```

h3sdm_get_grid *Generar cuadrícula H3 para un área de interés*

Description

Crea una cuadrícula de hexágonos H3 que cubre un área de interés (sf_object), asegurando que las celdas se ajusten a la extensión del área y se recorten opcionalmente al contorno del AOI.

Esta función es equivalente a la usada en los módulos de paisaje de h3sdm, pero con el nombre estandarizado para mantener consistencia en el paquete.

Usage

```
h3sdm_get_grid(sf_object, res = 6, expand_factor = 0.1, clip_to_aoi = TRUE)
```

Arguments

sf_object	Objeto sf que define el área de interés (AOI).
res	Entero entre 1 y 16. Define la resolución del índice H3. Valores mayores producen hexágonos más pequeños.
expand_factor	Valor numérico que amplía ligeramente el bounding box del AOI antes de generar los hexágonos. Por defecto 0.1.
clip_to_aoi	Lógico (TRUE o FALSE), indica si los hexágonos deben recortarse exactamente al contorno del AOI. Por defecto TRUE.

Value

Un objeto sf con los hexágonos H3 correspondientes al área de interés, con geometrías válidas (MULTIPOLYGON).

Examples

```
## Not run:
library(sf)

# Crear un polígono de ejemplo
cr <- st_as_sf(data.frame(
  lon = c(-85, -85, -83, -83, -85),
  lat = c(9, 11, 11, 9, 9)
), coords = c("lon", "lat"), crs = 4326) |>
  summarise(geometry = st_combine(geometry)) |>
  st_cast("POLYGON")

# Generar cuadrícula H3
h5 <- h3sdm_get_grid(cr, res = 5)
plot(st_geometry(h5))

## End(Not run)
```

h3sdm_get_records *Query Species Occurrence Records within an H3 Area of Interest (AOI)*

Description

Downloads species occurrence records from providers (e.g., GBIF, iNaturalist, BiodataCR) and filters them by the exact polygonal boundary of the Area of Interest (AOI). Providers supported by spocc (e.g., "gbif", "inat") are queried via spocc::occ(). "biodatacr" is queried via rbiodatacr::bdcr_occurrences() and its output is standardized to the same sf format.

Usage

```
h3sdm_get_records(
  species,
  aoi_sf,
  providers = NULL,
  limit = 500,
  remove_duplicates = FALSE,
  date = NULL
)
```

Arguments

species	Character string specifying the species name to query (e.g., "Puma concolor").
aoi_sf	An sf object defining the Area of Interest (AOI). Its CRS will be transformed to WGS84 (EPSG:4326) before query.
providers	Character vector of data providers to query. Accepted values: any provider supported by spocc (e.g., "gbif", "inat") plus "biodatacr" for BiodataCR (Costa Rica). If NULL (default), all spocc providers are used.
limit	Numeric. Maximum number of records to retrieve per provider. Default is 500.
remove_duplicates	Logical. If TRUE, records with identical coordinates are removed. Default is FALSE.
date	Character vector specifying a date range (e.g., c('2000-01-01', '2020-12-31')). Applied to spocc providers only.

Details

When "biodatacr" is included in providers, the function calls rbiodatacr::bdcr_occurrences() and standardizes its output (decimalLatitude/decimalLongitude) to the same sf geometry format used by the spocc providers. Records from all providers are then combined and clipped to the AOI.

Value

An `sf` object of points with the filtered occurrence records whose geometry falls strictly within the `aoi_sf` boundary. If no records are found, an empty `sf` object with the expected structure is returned.

Examples

```
library(sf)

aoi_sf <- sf::st_as_sf(
  data.frame(
    lon = c(-84.5, -83.5, -83.5, -84.5, -84.5),
    lat = c(9.5, 9.5, 10.5, 10.5, 9.5)
  ) |>
    {\(d) sf::st_sfc(sf::st_polygon(list(as.matrix(d))), crs = 4326)}(),
  data.frame(id = 1)
)

# GBIF only
records <- h3sdm_get_records(
  species   = "Puma concolor",
  aoi_sf    = aoi_sf,
  providers = "gbif",
  limit     = 100
)

# GBIF + BiodataCR (Costa Rica)
records_cr <- h3sdm_get_records(
  species   = "Agalychnis callidryas",
  aoi_sf    = aoi_sf,
  providers = c("gbif", "biodatacr"),
  limit     = 200
)
```

`h3sdm_get_records_by_hexagon`

Download Species Records and Count Occurrences per H3 Hexagon

---

Description

This function downloads occurrence records for one or more species and counts the number of records falling inside each H3 hexagon covering the specified Area of Interest (AOI).

Usage

```
h3sdm_get_records_by_hexagon(
  species,
  aoi_sf,
```

```
  res = 6,
  providers = NULL,
  remove_duplicates = FALSE,
  date = NULL,
  expand_factor = 0.1,
  limit = 500
)
```

Arguments

species	Character vector of species names to query (e.g., c("Puma concolor", "Panthera onca")).
aoi_sf	An sf polygon defining the Area of Interest (AOI).
res	Numeric. H3 resolution level (default 6), determining hexagon size.
providers	Character vector of data providers (e.g., "gbif"). If NULL, all providers are used.
remove_duplicates	Logical. If TRUE, duplicate coordinates are removed before counting. Default is FALSE.
date	Character vector specifying a date range (e.g., c('2000-01-01','2020-12-31')).
expand_factor	Numeric. Factor to expand the AOI bounding box before generating the H3 grid. Default is 0.1.
limit	Numeric. Maximum number of records to retrieve per species per provider. Default is 500.

Details

Download Species Records and Count Occurrences per H3 Hexagon

For each species:

1. An H3 grid is generated across the AOI using h3sdm_get_grid().
2. Occurrence records are downloaded using h3sdm_get_records().
3. Points are joined to the hexagonal grid with sf::st_join().
4. Counts of points per hexagon are calculated.
5. Counts are merged into the main hex grid.

The function ensures column names derived from species names are safe in R by replacing spaces with underscores and handles API failures gracefully.

Value

An sf object containing the H3 hexagonal grid (MULTIPOLYGON) with additional integer columns for each species (spaces replaced by underscores) showing the count of occurrence records in each hexagon. Hexagons with no records have 0.

See Also

h3sdm_get_grid, h3sdm_get_records

Examples

```
library(sf)

# Create a simple AOI polygon in Costa Rica
aoi_sf <- sf::st_as_sf(
  data.frame(id = 1),
  geometry = sf::st_sfc(
    sf::st_polygon(list(matrix(
      c(-84.5, 9.5,
        -83.5, 9.5,
        -83.5, 10.5,
        -84.5, 10.5,
        -84.5, 9.5),
      ncol = 2, byrow = TRUE
    ))),
    crs = 4326
  )
)

hex_counts <- h3sdm_get_records_by_hexagon(
  species = c("Agalychnis callidryas", "Smilisca baudinii"),
  aoi_sf  = aoi_sf,
  res     = 7,
  providers = "gbif",
  limit   = 100
)

print(hex_counts)
```

`h3sdm_pa` *Generate presence/pseudo-absence dataset stratified in environmental space*

Description

Combines presence hexagons with pseudo-absences sampled in environmental space. Pseudo-absences are selected by clustering the environmental conditions of hexagons without presence records using k-means, then choosing the hexagon closest to each cluster centroid. This ensures pseudo-absences cover the full range of environmental conditions available in the AOI, reducing bias from spatially clustered occurrence records.

Usage

```
h3sdm_pa(pres_sf, predictors_sf, n_pseudoabs = 500, buffer_k = 1L)
```

Arguments

pres_sf	sf Presence hexagons returned by h3sdm_pres().
predictors_sf	sf Full hexagonal grid with extracted environmental variables, returned by h3sdm_predictors().
n_pseudoabs	integer Number of pseudo-absence hexagons to sample. If larger than the number of available hexagons without presence, all available hexagons are used. Default is 500.
buffer_k	integer Number of H3 grid rings to exclude around each presence hexagon when building the pseudo-absence candidate pool. Hexagons within buffer_k rings of any presence are removed before sampling, preventing pseudo-absences from being placed in areas likely occupied but not yet recorded. Default is 1. Set to 0 to disable.

Details

The function scales all numeric predictor columns before clustering. Non-numeric columns and columns with zero variance are excluded from clustering. Pseudo-absences are selected as the hexagon nearest to each k-means centroid in scaled environmental space (Euclidean distance).

This function is designed to be used after h3sdm_pres() and h3sdm_predictors() in the following workflow:

```
pres        <- h3sdm_pres("Species name", aoi_sf, res = 7)
num_vars    <- h3sdm_extract_num(raster_stack, grid)
predictors  <- h3sdm_predictors(num_vars)
pa          <- h3sdm_pa(pres, predictors, n_pseudoabs = 500)
```

Value

sf object with columns:

- h3_address: H3 index of the hexagon.
- presence: factor with levels "0" (pseudo-absence) and "1" (presence).
- geometry: MULTIPOLYGON of each hexagon.

Examples

```
## Not run:
data(cr_outline_c, package = "h3sdm")
pres       <- h3sdm_pres("Agalychnis callidryas", cr_outline_c, res = 7)
grid       <- h3sdm_get_grid(cr_outline_c, res = 7)
num_vars   <- h3sdm_extract_num(bio, grid)
predictors <- h3sdm_predictors(num_vars)
pa         <- h3sdm_pa(pres, predictors, n_pseudoabs = 300)

## End(Not run)
```

h3sdm_pa_from_records *Generate presence/pseudo-absence dataset from user-provided records*

Description

Adapts a user-provided dataset with presence records (from personal fieldwork, BiodataCR, or any other source) into a hexagonal presence/pseudo-absence dataset ready for analysis with h3sdm. The input can be a `data.frame` with coordinate columns or an `sf` object. Coordinates are assumed to be in WGS84 (EPSG:4326).

Usage

```
h3sdm_pa_from_records(
  records,
  aoi_sf,
  res = 6,
  n_pseudoabs = 500,
  expand_factor = 0.1,
  lon_col = "lon",
  lat_col = "lat",
  species_col = NULL,
  predictors_sf = NULL,
  geospatial_filter = TRUE,
  buffer_k = 1L
)
```

Arguments

`records`	`data.frame` or `sf` object containing presence records.
`aoi_sf`	`sf` AOI (area of interest) polygon.
`res`	`integer` H3 resolution for the hexagonal grid.
`n_pseudoabs`	`integer` Number of pseudo-absence hexagons to sample.
`expand_factor`	`numeric` Factor to expand AOI before creating hex grid.
`lon_col`	`character` Name of the longitude column. Ignored if `records` is already an `sf` object.
`lat_col`	`character` Name of the latitude column. Ignored if `records` is already an `sf` object.
`species_col`	`character` Optional. Name of the column containing the species name. If provided, the column is retained in the output as metadata.
`predictors_sf`	`sf` Optional. Full hexagonal grid with extracted environmental variables, returned by `h3sdm_predictors()`. If provided, pseudo-absences are selected by stratified sampling in environmental space using k-means clustering, ensuring coverage of the full range of environmental conditions in the AOI. If `NULL` (default), pseudo-absences are sampled randomly in geographic space.

geospatial_filter

: logical If TRUE (default) and the input contains a geospatialKosher column, records with geospatialKosher == FALSE are removed before processing. Ignored if the column is absent.

buffer_k

: integer Number of H3 grid rings to exclude around each presence hexagon when building the pseudo-absence candidate pool. Hexagons within buffer_k rings of any presence are removed before sampling, preventing pseudo-absences from being placed in areas likely occupied but not yet recorded. Default is 1. Set to 0 to disable.

Value

sf object with columns:

h3_address H3 index of the hexagon.

presence Factor with levels "0" (pseudo-absence) and "1" (presence).

species Species name (only if species_col is provided).

geometry MULTIPOLYGON of each hexagon.

Examples

```
data(cr_outline_c, package = "h3sdm")

my_records <- data.frame(
  lon = c(-84.1, -84.2, -83.9),
  lat = c(9.9, 10.1, 9.8),
  species = "Agalychnis callidryas"
)

dataset <- h3sdm_pa_from_records(
  records     = my_records,
  aoi_sf      = cr_outline_c,
  res         = 7,
  n_pseudoabs = 100,
  lon_col     = "lon",
  lat_col     = "lat",
  species_col = "species"
)
```

---

h3sdm_predict *Predict species presence probability or counts using H3 hexagons*

Description

Uses a fitted tidymodels workflow (from h3sdm_fit_model or a standalone workflow) to predict species presence probabilities or counts on a new spatial H3 grid. Automatically generates centroid coordinates (x and y) if missing. The new_data must contain the same predictor variables as used in model training. Model mode (classification or regression) is detected automatically.

Usage

```
h3sdm_predict(fit_object, new_data)
```

Arguments

fit_object	A fitted tidymodels workflow or the output list from h3sdm_fit_model.
new_data	An sf object containing the spatial grid and the same predictor variables used for model training.

Value

An sf object with the original geometry and a new column prediction containing the predicted probability of presence (classification) or predicted count (regression) for each hexagon.

See Also

h3sdm_fit_model(), h3sdm_aoa()

Examples

```
## Not run:
# Predict presence probabilities on a new hex grid
predictions_sf <- h3sdm_predict(
  fit_object = fitted_model,
  new_data   = grid_sf
)

## End(Not run)
```

---

h3sdm_predictors *Combine Predictor Data from Multiple sf Objects*

Description

This function merges predictor variables from multiple sf objects into a single sf object. It preserves the geometry from the first input and joins columns from the other sf objects using a common key (h3_address or ID).

Usage

```
h3sdm_predictors(...)
```

Arguments

...	Two or more sf objects containing predictor variables. The first object must contain the geometry to preserve. All objects must share a common key column (h3_address or ID).

Details

The function uses a left join based on the `h3_address` column if present, otherwise it falls back to `ID`. Geometries from the right-hand side `sf` objects are dropped to avoid conflicts, and the final geometry is cast to `MULTIPOLYGON`.

Value

An `sf` object containing the geometry of the first input and all predictor columns from all provided `sf` objects.

Examples

```
## Not run:
# Combine sf objects with different predictor types into one
combined <- h3sdm_predictors(num_sf, cat_sf, it_sf)
head(combined)

## End(Not run)
```

`h3sdm_pres` *Assign species presence records to H3 hexagons*

Description

Generates a hexagonal grid over the AOI, downloads species occurrence records, and assigns them to hexagons. Returns only hexagons with at least one presence record. This is the first step of a two-stage workflow where pseudo-absences are generated later using `h3sdm_pa()` after environmental variables have been extracted with `h3sdm_extract_num()` and related functions.

Usage

```
h3sdm_pres(
  species,
  aoi_sf,
  res = 6,
  providers = NULL,
  remove_duplicates = FALSE,
  date = NULL,
  limit = 500,
  expand_factor = 0.1
)
```

Arguments

species	character Species name (single string) for which records are requested.
aoi_sf	sf AOI (area of interest) polygon.
res	integer H3 resolution for the hexagonal grid.
providers	character Optional vector of data providers. Accepted values: any provider supported by spocc (e.g., "gbif", "inat") plus "biodatacr" for BiodataCR (Costa Rica), queried via the rbiodatacr package. If NULL (default), all spocc providers are used.
remove_duplicates	logical Remove duplicate records at the same coordinates. Default is FALSE.
date	character Optional date filter for records.
limit	integer Maximum number of records to download. Default is 500.
expand_factor	numeric Factor to expand AOI before creating the hexagonal grid. Default is 0.1.

Details

Unlike h3sdm_pa(), this function does not sample pseudo-absences. It is intended to be used as the first step of a workflow where environmental variables are extracted for the full hexagonal grid before pseudo-absences are selected in environmental space using h3sdm_pa().

Value

sf object with columns:

- h3_address: H3 index of the hexagon.
- presence: integer column with value 1 for all returned hexagons.
- geometry: MULTIPOLYGON of each hexagon.

Examples

```
## Not run:
data(cr_outline_c, package = "h3sdm")
pres <- h3sdm_pres("Agalychnis callidryas", cr_outline_c, res = 7)

## End(Not run)
```

h3sdm_pres_from_sf *Assign pre-downloaded species presence records to H3 hexagons*

Description

Takes an sf object of species occurrence records already downloaded (e.g. from h3sdm_get_records()) and assigns them to H3 hexagons, returning only hexagons with at least one presence record.

This function extracts the hexagon-assignment logic from h3sdm_get_records_by_hexagon() without downloading records internally, making it suitable for workflows where records have already been retrieved.

Usage

```
h3sdm_pres_from_sf(records_sf, aoi_sf, res = 6, expand_factor = 0.1)
```

Arguments

records_sf	An sf object with presence records in any CRS. Typically the output of h3sdm_get_records().
aoi_sf	An sf object defining the area of interest.
res	Integer. H3 resolution (0–15). Default is 6.
expand_factor	Numeric. Expansion factor for the H3 grid beyond the AOI bounding box. Default is 0.1.

Details

This function is designed to be used in combination with h3sdm_filter_outliers() and h3sdm_pa() for a balanced presence/pseudo-absence workflow:

```
# 1. Download records
records_sf <- h3sdm_get_records("Species name", aoi_sf, providers = c("gbif", "biodatacr"))

# 2. Assign to hexagons
pres_sf <- h3sdm_pres_from_sf(records_sf, aoi_sf, res = 7)

# 3. Filter environmental outliers (only presences)
filtro <- h3sdm_filter_outliers(pres_sf_env, vars_cov)
pres_clean <- filtro$pa_clean

# 4. Generate balanced pseudo-absences (1:1)
pa <- h3sdm_pa(pres_clean, predictors_sf, n_pseudoabs = nrow(pres_clean))
```

Value

An sf object with one row per presence hexagon, containing:

h3_address H3 index of the hexagon.

n Number of records assigned to the hexagon.

geometry MULTIPOLYGON geometry of the hexagon.

See Also

h3sdm_get_records(), h3sdm_pa(), h3sdm_filter_outliers()

Examples

```
## Not run:
data(cr_outline_c, package = "h3sdm")

records_sf <- h3sdm_get_records(
  species   = "Panthera onca",
  aoi_sf    = cr_outline_c,
  providers = c("gbif", "biodatacr"),
  limit     = 500
)

pres_sf <- h3sdm_pres_from_sf(records_sf, cr_outline_c, res = 7)
nrow(pres_sf)

## End(Not run)
```

---

h3sdm_recipe *Create a tidymodels recipe for H3-based SDMs*

Description

Prepares an sf object with H3 hexagonal data for modeling with the tidymodels ecosystem. Extracts centroid coordinates, assigns appropriate roles to the variables automatically, and returns a ready-to-use recipe for modeling species distributions.

Usage

```
h3sdm_recipe(data, response_col = "presence")
```

Arguments

data	An sf object, typically the output of h3sdm_data(), including species presence-absence or count data, H3 addresses, and environmental predictors. The geometry must be of type MULTIPOLYGON.
response_col	character Name of the column to use as the outcome (response variable). Default "presence" for presence/absence models. Use "count" when working with count data generated by h3sdm_count_from_records().

Details

This function prepares spatial H3 grid data for species distribution modeling:

- Extracts centroid coordinates (`x` and `y`) from MULTIPOLYGON geometries.
- Removes the geometry column to create a purely tabular dataset for tidymodels.
- Assigns roles to columns:
  - `response_col` → `"outcome"` (target variable)
  - `h3_address` → `"id"` (used for joining predictions later)
  - `x` and `y` → `"spatial_predictor"`
- All other columns are assigned `"predictor"` role.

Value

A `tidymodels` recipe object (class `"h3sdm_recipe"`) ready for modeling.

Examples

```
## Not run:
# Presence/absence model (default)
sdm_recipe <- h3sdm_recipe(combined_data)

# Count-based model
sdm_recipe <- h3sdm_recipe(combined_data, response_col = "count")

## End(Not run)
```

---

h3sdm_recipe_gam *Creates a 'recipe' object for Generalized Additive Models (GAM) in SDM*

Description

This function prepares an `sf` object for use in a Species Distribution Model (SDM) workflow with the 'mgcv' GAM engine within the 'tidymodels' ecosystem. Extracts centroid coordinates and assigns appropriate roles to all variables, including the response variable and spatial coordinates.

Usage

```
h3sdm_recipe_gam(data, response_col = "presence")
```

Arguments

`data`	An `sf` object containing the response variable, environmental predictors, and geometry (e.g., H3 hexagon polygons).
`response_col`	`character` Name of the column to use as the outcome (response variable). Default `"presence"` for presence/absence models. Use `"count"` when working with count data generated by `h3sdm_count_from_records()`.

Details

Assigned Roles:

- `outcome`: the column specified in `response_col`.
- `id`: `"h3_address"` (cell identifier, not used for modeling).
- `predictor`: all other variables, including `x` and `y` for the GAM spatial smooth term (`s(x, y, bs = "tp")`).

Value

A `recipe` object of class `h3sdm_recipe_gam`, ready to be chained with additional preprocessing steps.

See Also

Other h3sdm_tools: `h3sdm_stack_fit()`, `h3sdm_workflow_gam()`

Examples

```
library(sf)
library(recipes)

set.seed(42)
n <- 20

pts <- sf::st_as_sf(
  data.frame(
    h3_address   = paste0("hex_", seq_len(n)),
    presence     = sample(0:1, n, replace = TRUE),
    count        = sample(0:9, n, replace = TRUE),
    bio1_temp    = runif(n, 15, 30),
    bio12_precip = runif(n, 500, 3000)
  ),
  geometry = sf::st_sfc(
    lapply(seq_len(n), function(i) {
      sf::st_point(c(runif(1, -84.5, -83.5), runif(1, 9.5, 10.5)))
    }),
    crs = 4326
  )
)

# Presence/absence model (default)
gam_rec <- h3sdm_recipe_gam(pts)

# Count-based model
gam_rec <- h3sdm_recipe_gam(pts, response_col = "count")
```

h3sdm_spatial_cv *Create a spatial-aware cross-validation split for H3 data*

Description

Generates a spatially aware cross-validation split for species distribution modeling using H3 hexagonal grids. This helps avoid inflated model performance estimates caused by spatial autocorrelation, producing more robust model evaluation.

Usage

```
h3sdm_spatial_cv(data, method = "block", v = 5, ...)
```

Arguments

data	An sf object, typically the output of h3sdm_data().
method	Character. The spatial resampling method to use:
	"block" Use spatialsample::spatial_block_cv() for block-based spatial CV.
	"cluster" Use spatialsample::spatial_clustering_cv() for cluster-based spatial CV.
v	Integer. Number of folds (default = 5).
...	Additional arguments passed to the underlying spatialsample function.

Details

Spatial cross-validation avoids overly optimistic performance estimates by ensuring that training and testing data are spatially separated.

- "block": Divides the spatial domain into contiguous blocks.
- "cluster": Groups locations into spatial clusters before splitting.

Value

An rsplit object (from rsample) representing the spatial CV folds.

Examples

```
## Not run:
# Example: Create spatial cross-validation splits for H3 data

# Block spatial CV with default folds
spatial_cv_block <- h3sdm_spatial_cv(combined_data, method = "block")

# Cluster spatial CV with 10 folds
spatial_cv_cluster <- h3sdm_spatial_cv(combined_data, method = "cluster", v = 10)
```

```
## End(Not run)
```

`h3sdm_stack_fit` *Creates and fully fits an ensemble model (Stack).*

Description

This function combines the process of creating the model stack, optimizing the weights (`blend_predictions`), and fitting the base models to the complete training set (`fit_members()`) into a single step.

Warning: It does not follow the canonical tidymodels flow but is convenient. It requires that the fitting results were generated using `h3sdm_fit_model(..., for_stacking = TRUE)`.

Usage

```
h3sdm_stack_fit(..., non_negative = TRUE, metric = NULL)
```

Arguments

`...`	List objects that are the result of `h3sdm_fit_model()`. Each object must contain the `cv_model` element (result of fit_resamples).
`non_negative`	Logical. If `TRUE` (default), forces the candidate model weights to be non-negative.
`metric`	The metric used to optimize the combination of weights.

Value

A list containing two elements: `blended_model` (the stack after blending) and `final_model` (a fully `fitted model_stack` object). The `final_model` is ready for direct prediction with `predict()`.

See Also

Other h3sdm_tools: `h3sdm_recipe_gam()`, `h3sdm_workflow_gam()`

`h3sdm_workflow` *Create a tidymodels workflow for H3-based SDMs*

Description

Combines a model specification and a prepared recipe into a single `tidymodels` workflow. This workflow is suitable for species distribution modeling using H3 hexagonal grids and can be directly fitted or cross-validated.

Usage

```
h3sdm_workflow(model_spec, recipe)
```

Arguments

model_spec	A tidymodels model specification (e.g., logistic_reg(), rand_forest(), or boost_tree()), describing the model type and engine to use for fitting. Use set_mode("classification") for presence/absence models and set_mode("regression") for count-based models (species richness, detections, or individuals).
recipe	A tidymodels recipe object, typically created with h3sdm_recipe(), which preprocesses the data and defines predictor/response roles. Use response_col = "count" in h3sdm_recipe() when working with count data.

Details

The function creates a workflow that combines preprocessing and modeling steps. This encapsulation allows consistent model training and evaluation with tidymodels functions like fit() or fit_resamples(), and is particularly useful when applying multiple models in parallel.

Choosing the model mode:

- For **presence/absence** data: use set_mode("classification").
- For **count** data (species richness, detections, individuals): use set_mode("regression").

Value

A workflow object ready to be used for model fitting with fit() or cross-validation.

Examples

```
## Not run:
library(parsnip)

# --- Presence/absence model ---
rf_spec_pa <- rand_forest() %>%
  set_engine("ranger") %>%
  set_mode("classification")

rec_pa <- h3sdm_recipe(combined_data)

wf_pa <- h3sdm_workflow(model_spec = rf_spec_pa, recipe = rec_pa)

# --- Count-based model ---
rf_spec_count <- rand_forest() %>%
  set_engine("ranger") %>%
  set_mode("regression")

rec_count <- h3sdm_recipe(combined_data, response_col = "count")

wf_count <- h3sdm_workflow(model_spec = rf_spec_count, recipe = rec_count)

## End(Not run)
```

h3sdm_workflows *Create multiple tidymodels workflows for H3-based SDMs*

Description

Creates a list of tidymodels workflows from multiple model specifications and a prepared recipe. This is useful for comparing different modeling approaches in species distribution modeling using H3 hexagonal grids.

Usage

```
h3sdm_workflows(model_specs, recipe)
```

Arguments

model_specs	A named list of tidymodels model specifications (e.g., logistic_reg(), rand_forest(), boost_tree()), where each element specifies a different modeling approach. All specifications must use the same mode: set_mode("classification") for presence/absence models or set_mode("regression") for count-based models.
recipe	A tidymodels recipe object, typically created with h3sdm_recipe(), which prepares and preprocesses the data for modeling. Use response_col = "count" in h3sdm_recipe() when working with count data.

Details

This function automates the creation of workflows for multiple model specifications. Each workflow combines the same preprocessing steps (recipe) with a different modeling method, facilitating systematic comparison of models.

Choosing the model mode:

- For **presence/absence** data: use set_mode("classification") for all model specifications.
- For **count** data (species richness, detections, individuals): use set_mode("regression") for all model specifications.

Value

A named list of workflow objects, one per model specification.

Examples

```
## Not run:
library(parsnip)

# --- Presence/absence models ---
specs_pa <- list(
  rf  = rand_forest() %>% set_engine("ranger") %>% set_mode("classification"),
```

```
  glm = logistic_reg() %>% set_engine("glm") %>% set_mode("classification")
)

rec_pa <- h3sdm_recipe(combined_data)

wfs_pa <- h3sdm_workflows(model_specs = specs_pa, recipe = rec_pa)

# --- Count-based models ---
specs_count <- list(
  rf  = rand_forest() %>% set_engine("ranger") %>% set_mode("regression"),
  xgb = boost_tree() %>% set_engine("xgboost") %>% set_mode("regression")
)

rec_count <- h3sdm_recipe(combined_data, response_col = "count")

wfs_count <- h3sdm_workflows(model_specs = specs_count, recipe = rec_count)

## End(Not run)
```

`h3sdm_workflow_gam` *Creates a tidymodels workflow for Generalized Additive Models (GAM).*

Description

This function constructs a `workflow` object by combining a GAM model specification (`gen_additive_mod` with the `mgcv` engine) with either a `recipe` object or an explicit model formula.

It is optimized for Species Distribution Models (SDM) that use smooth splines, ensuring that the specialized GAM formula (containing `s()` terms) is correctly passed to the model, even when a recipe is provided for general data preprocessing.

Usage

```
h3sdm_workflow_gam(gam_spec, recipe = NULL, formula = NULL)
```

Arguments

`gam_spec`	A `parsnip` model specification of type `gen_additive_mod()`, configured with `set_engine("mgcv")`. Use `set_mode("classification")` for presence/absence models and `set_mode("regression")` for count-based models.
`recipe`	(Optional) A `recipes` package `recipe` object (e.g., the output of `h3sdm_recipe_gam`). Used for general data preprocessing like normalization or dummy variable creation.
`formula`	(Optional) A `formula` object that defines the structure of the GAM, including smooth terms (e.g., `y ~ s(x1) + s(x, y)`). If provided alongside `recipe`, this formula overrides the recipe's implicit formula for the final model fit.

Details

Formula Priority:

- If **only** `recipe` is provided, the workflow uses the recipe's implicit formula (e.g., `outcome ~ .`).
- If `recipe` **and** `formula` are provided, the workflow uses the `recipe` for data preprocessing but explicitly passes the `formula` to the `mgcv` engine for fitting, enabling the use of specialized terms like `s(x, y)`.

Choosing the model mode and family:

- For **presence/absence** data: use `set_mode("classification")`. The `mgcv` engine uses `binomial()` family by default.
- For **count** data (species richness, detections, individuals): use `set_mode("regression")` and specify `set_engine("mgcv", family = poisson())`.

Value

A `workflow` object, ready for fitting with `fit()` or resampling with `fit_resamples()` or `tune_grid()`.

See Also

Other h3sdm_tools: `h3sdm_recipe_gam()`, `h3sdm_stack_fit()`

Examples

```
## Not run:
library(parsnip)

# --- Presence/absence model (binomial) ---
gam_spec_pa <- gen_additive_mod() %>%
  set_engine("mgcv") %>%
  set_mode("classification")

gam_formula_pa <- presence ~ s(bio1) + s(bio12) + s(x, y, bs = "tp")

base_rec_pa <- h3sdm_recipe_gam(data)

h3sdm_wf_pa <- h3sdm_workflow_gam(
  gam_spec = gam_spec_pa,
  recipe   = base_rec_pa,
  formula  = gam_formula_pa
)

# --- Count-based model (Poisson) ---
gam_spec_count <- gen_additive_mod() %>%
  set_engine("mgcv", family = poisson()) %>%
  set_mode("regression")

gam_formula_count <- count ~ s(bio1) + s(bio12) + s(x, y, bs = "tp")
```

```
base_rec_count <- h3sdm_recipe_gam(data, response_col = "count")

h3sdm_wf_count <- h3sdm_workflow_gam(
  gam_spec = gam_spec_count,
  recipe   = base_rec_count,
  formula  = gam_formula_count
)

## End(Not run)
```

`records` *Presence/pseudo-absence records for Silverstoneia flotator*

Description

A dataset containing presence and pseudo-absence records for the species *Silverstoneia flotator* in Costa Rica, generated using H3 hexagonal grids at resolution 7.

Usage

```
records
```

Format

An `sf` object with columns:

h3_address H3 index of the hexagon

presence factor with levels "0" (pseudo-absence) and "1" (presence)

geometry MULTIPOLYGON of each hexagon

Source

Generated using `h3sdm_pa()` with occurrence data from GBIF (https://www.gbif.org).

Examples

```
data(records)
head(records)
table(records$presence)
```

Index

∗ **datasets**
  cr_outline, 4
  cr_outline_c, 5
  records, 46
∗ **h3sdm_tools**
  h3sdm_recipe_gam, 38
  h3sdm_stack_fit, 41
  h3sdm_workflow_gam, 44

bioclim_current, 2
bioclim_future, 3

cr_outline, 4, *5, 6*
cr_outline_c, *4*, 5

h3sdm_aoa, 6
h3sdm_aoa(), *20, 21*, *33*
h3sdm_calculate_it_metrics, 7
h3sdm_classify, 9
h3sdm_compare_models, 10
h3sdm_count_from_records, 11
h3sdm_data, 13
h3sdm_data(), *6, 7*
h3sdm_eval_metrics, 14
h3sdm_explain, 15
h3sdm_extract_cat, 16
h3sdm_extract_num, 18
h3sdm_filter_outliers, 19
h3sdm_filter_outliers(), *36, 37*
h3sdm_filter_range, 21
h3sdm_fit_model, 22
h3sdm_fit_model(), *6, 7*, *33*
h3sdm_fit_models, 23
h3sdm_get_grid, 25, *28*
h3sdm_get_records, 26, *28*
h3sdm_get_records(), *36, 37*
h3sdm_get_records_by_hexagon, 27
h3sdm_get_records_by_hexagon(), *36*
h3sdm_pa, 29
h3sdm_pa(), *19*, *36, 37*
h3sdm_pa_from_records, 31
h3sdm_pa_from_records(), *19, 20*
h3sdm_predict, 32
h3sdm_predict(), *6, 7*, *9*, *21*
h3sdm_predictors, 33
h3sdm_pres, 34
h3sdm_pres_from_sf, 36
h3sdm_recipe, 37
h3sdm_recipe_gam, 38, *44*
h3sdm_recipe_gam(), *41*, *45*
h3sdm_spatial_cv, 40
h3sdm_spatial_cv(), *6, 7*
h3sdm_stack_fit, 41
h3sdm_stack_fit(), *39*, *45*
h3sdm_workflow, 41
h3sdm_workflow_gam, 44
h3sdm_workflow_gam(), *39*, *41*
h3sdm_workflows, 43

records, 46