Package 'ebirdst'

May 8, 2026

Type Package

Title Access and Analyze eBird Status and Trends Data Products

Version 3.2023.1

Description Tools for accessing and analyzing eBird Status and Trends Data Products (<https://science.ebird.org/en/status-and-trends>). eBird (<https://ebird.org/home>) is a global database of bird observations collected by member of the public. eBird Status and Trends uses these data to model global bird distributions, abundances, and population trends at a high spatial and temporal resolution.

License GPL-3

URL https://ebird.github.io/ebirdst/, https://github.com/ebird/ebirdst

BugReports https://github.com/ebird/ebirdst/issues

Depends R (>= 4.0.0)

Imports arrow, dplyr (>= 1.0.0), grDevices, jsonlite, magrittr, RColorBrewer, rlang, scales, sf (>= 1.0-0), stats, stringr, terra (>= 1.6-3), tools, utils, viridisLite

Suggests fields, ggplot2, knitr, lubridate, PresenceAbsence, rmarkdown, rnaturalearth, scico, testthat, tidyr, withr

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Matthew Strimas-Mackey [aut, cre] (ORCID: <https://orcid.org/0000-0001-8929-7776>), Shawn Ligocki [aut], Tom Auer [aut] (ORCID: <https://orcid.org/0000-0001-8619-7147>), Daniel Fink [aut] (ORCID: <https://orcid.org/0000-0002-8368-1248>), Cornell Lab of Ornithology [cph]

Maintainer Matthew Strimas-Mackey <mes335@cornell.edu>

Repository CRAN

Date/Publication 2025-10-19 09:40:02 UTC

Contents

`assign_to_grid` *Assign points to a spacetime grid*

Description

Given a set of points in space and (optionally) time, define a regular grid with given dimensions, and return the grid cell index for each point.

Usage

```
assign_to_grid(
  points,
  coords = NULL,
  is_lonlat = FALSE,
  res,
  jitter_grid = TRUE,
  grid_definition = NULL
)
```

Arguments

points	data frame; points with spatial coordinates x and y, and an optional time coordinate t.
coords	character; names of the spatial and temporal coordinates in the input dataframe. Only provide these names if you want to overwrite the default coordinate names: c("x", "y", "t") or c("longitude", "latitude", "t") if is_lonlat = TRUE.
is_lonlat	logical; if the points are in unprojected, lon-lat coordinates. In this case, the input data frame should have columns "longitude" and "latitude" and the points will be projected to an equal area Eckert IV CRS prior to grid assignment.
res	numeric; resolution of the grid in the x, y, and t dimensions, respectively. If only 2 dimensions are provided, a space only grid will be generated. The units of res are the same as the coordinates in the input data unless is_lonlat is true in which case the x and y resolution should be provided in meters.
jitter_grid	logical; whether to jitter the location of the origin of the grid to introduce some randomness.
grid_definition	list; object defining the grid via the origin and resolution components. To assign multiple sets of points to exactly the same grid, assign_to_grid() returns a data frame with a grid_definition attribute that can be passed to subsequent calls to assign_to_grid(). res and jitter are ignored if grid_definition is provided.

Value

Data frame with the indices of the space-only and spacetime grid cells. This data frame will have a grid_definition attribute that can be used to reconstruct the grid.

Examples

```
set.seed(1)

# generate some example points
points_xyt <- data.frame(x = runif(100), y = runif(100), t = rnorm(100))
# assign to grid
cells <- assign_to_grid(points_xyt, res = c(0.1, 0.1, 0.5))

# assign a second set of points to the same grid
```

```
assign_to_grid(points_xyt, grid_definition = attr(cells, "grid_definition"))

# assign lon-lat points to a 10km space-only grid
points_ll <- data.frame(longitude = runif(100, min = -180, max = 180),
                        latitude = runif(100, min = -90, max = 90))
assign_to_grid(points_ll, res = c(10000, 10000), is_lonlat = TRUE)

# overwrite default coordinate names, 5km by 1 week grid
points_names <- data.frame(lon = runif(100, min = -180, max = 180),
                           lat = runif(100, min = -90, max = 90),
                           day = sample.int(365, size = 100))
assign_to_grid(points_names,
               res = c(5000, 5000, 7),
               coords = c("lon", "lat", "day"),
               is_lonlat = TRUE)
```

`calculate_mcc_f1` *Calculate MCC and F1 score*

Description

Given binary observed and predicted response, estimate Matthews correlation coefficient (MCC) and the F1 score.

Usage

```
calculate_mcc_f1(observed, predicted)
```

Arguments

`observed`	logical or 0/1; the observed binary response.
`predicted`	logical or 0/1; the predicted binary response. This will typically need to be generated by applying a threshold to the continuous predicted response.

Value

A list with two elements: `mcc` and `f1`.

Examples

```
obs <- c(rep(1L, 1000L), rep(0L, 10000L))
pred <- c(rbeta(300L, 12, 2), rbeta(700L, 3, 4), rbeta(10000L, 2, 3))
calculate_mcc_f1(obs > 0, pred > 0.5)
```

convert_ppy_to_cumulative *Convert percent per year trend to cumulative trend*

Description

Convert percent per year trend to cumulative trend

Usage

```
convert_ppy_to_cumulative(x, n_years)
```

Arguments

x	numeric; percent per year trend on the 0-100 scale rather than the 0-1 scale.
n_years	integer; number of years.

Value

A numeric vector of the same length as x that contains the cumulative trend resulting from n_years years of compounding annual trend.

Examples

```
ppy_trend <- runif(100, min = -100, 100)
cumulative_trend <- convert_ppy_to_cumulative(ppy_trend, n_years = 5)
cbind(ppy_trend, cumulative_trend)
```

date_to_st_week *Get the Status and Trends week that a date falls into*

Description

Get the Status and Trends week that a date falls into

Usage

```
date_to_st_week(dates, version = 2022)
```

Arguments

dates	a vector of dates.
version	One of 2021 for the date scheme used for the 2021 and prior data releases or 2022 for the date scheme used in the 2022 and subsequent releases. Default is 2022.

Value

An integer vector of weeks numbers from 1-52.

Examples

```
d <- as.Date(c("2016-04-08", "2018-12-31", "2014-01-01", "2018-09-04"))
date_to_st_week(d)
```

---

ebirdst_data_dir *Path to eBird Status and Trends data download directory*

Description

Identify and return the path to the default download directory for eBird Status and Trends data products. This directory can be defined by setting the environment variable `EBIRDST_DATA_DIR`, otherwise the directory returned by `tools::R_user_dir("ebirdst", which = "data")` will be used.

Usage

```
ebirdst_data_dir()
```

Value

The path to the data download directory.

Examples

```
ebirdst_data_dir()
```

---

ebirdst_download_data_coverage *Download eBird Status and Trends Data Coverage Products*

Description

In addition to the species-specific data products, the eBird Status data products include two products providing estimates of weekly data coverage at 3 km spatial resolution: site selection probability and spatial coverage. This function downloads these data products in raster GeoTIFF format.

Usage

```
ebirdst_download_data_coverage(
  path = ebirdst_data_dir(),
  pattern = NULL,
  dry_run = FALSE,
  force = FALSE,
  show_progress = TRUE
)
```

Arguments

path	character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling `ebirdst_data_dir()`.
pattern	character; regular expression pattern to supply to str_detect() to filter files to download. This filter will be applied in addition to any of the `download_` arguments. Note that some files are mandatory and will always be downloaded.
dry_run	logical; whether to do a dry run, just listing files that will be downloaded. This can be useful when testing the use of `pattern` to filter the files to download.
force	logical; if the data have already been downloaded, should a fresh copy be downloaded anyway.
show_progress	logical; whether to print download progress information.

Value

Path to the folder containing the downloaded data coverage products.

Examples

```
## Not run:
# download all data coverage products
ebirdst_download_data_coverage()

# download just the spatial coverage products
ebirdst_download_data_coverage(pattern = "spatial-coverage")

# download a single week of data coverage products
ebirdst_download_data_coverage(pattern = "01-04")

# download all weeks in april
ebirdst_download_data_coverage(pattern = "04-")

## End(Not run)
```

`ebirdst_download_status` *Download eBird Status Data Products*

Description

Download eBird Status Data Products for a single species, or for an example species. Downloading Status and Trends data requires an access key, consult `set_ebirdst_access_key()` for instructions on how to obtain and store this key. The example data consist of the results for Yellow-bellied Sapsucker subset to Michigan and are much smaller than the full dataset, making these data quicker to download and process. Only the low resolution (27 km) data are available for the example data. In addition, the example data are accessible without an access key.

Usage

```
ebirdst_download_status(
  species,
  path = ebirdst_data_dir(),
  download_abundance = TRUE,
  download_occurrence = FALSE,
  download_count = FALSE,
  download_ranges = FALSE,
  download_regional = FALSE,
  download_pis = FALSE,
  download_ppms = FALSE,
  download_all = FALSE,
  pattern = NULL,
  dry_run = FALSE,
  force = FALSE,
  show_progress = TRUE
)
```

Arguments

`species`	character; a single species given as a scientific name, common name or six-letter species code (e.g. "woothr"). The full list of valid species is in the ebirdst_runs data frame included in this package. To download the example dataset, use `"yebsap-example"`.
`path`	character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling `ebirdst_data_dir()`.
`download_abundance`	whether to download estimates of abundance and proportion of population.
`download_occurrence`	logical; whether to download estimates of occurrence.

download_count logical; whether to download estimates of count.

download_ranges
: logical; whether to download the range polygons.

download_regional
: logical; whether to download the regional summary stats, e.g. percent of population in regions.

download_pis logical; whether to download spatial estimates of predictor importance.

download_ppms logical; whether to download spatial predictive performance metrics.

download_all logical; download all files in the data package. Equivalent to setting all the download_ arguments to TRUE.

pattern character; regular expression pattern to supply to str_detect() to filter files to download. This filter will be applied in addition to any of the download_ arguments. Note that some files are mandatory and will always be downloaded.

dry_run logical; whether to do a dry run, just listing files that will be downloaded. This can be useful when testing the use of pattern to filter the files to download.

force logical; if the data have already been downloaded, should a fresh copy be downloaded anyway.

show_progress logical; whether to print download progress information.

Details

The complete data package for each species contains a large number of files, all of which are cataloged in the vignettes. Most users will only require a small subset of these files, so by default this function only downloads the most commonly used files: GeoTIFFs providing estimate of relative abundance and proportion of population. For those interested in additional data products, the arguments starting with download_ control the download of these other products. The pattern argument provides even finer grained control over what gets downloaded.

Value

Path to the folder containing the downloaded data package for the given species. If dry_run = TRUE a list of files to download will be returned.

Examples

```
## Not run:
# download the example data
ebirdst_download_status("yebsap-example")

# download the data package for wood thrush
ebirdst_download_status("woothr")

# use pattern to only download low resolution (27 km) geotiff data
# dry_run can be used to see what files will be downloaded
ebirdst_download_status("lobcur", pattern = "_27km_", dry_run = TRUE)
# use pattern to only download high resolution (3 km) weekly abundance data
ebirdst_download_status("lobcur", pattern = "abundance_median_3km",
                        dry_run = TRUE)
```

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

`ebirdst_download_trends` *Download eBird Trends Data Products*

Description

Download eBird Trends Data Products for set of species, or for an example species. Downloading Status and Trends data requires an access key, consult `set_ebirdst_access_key()` for instructions on how to obtain and store this key. The example data consist of the results for Yellow-bellied Sapsucker subset to Michigan and are much smaller than the full dataset, making these data quicker to download and process. The example data are accessible without an access key.

Usage

```
ebirdst_download_trends(
  species,
  path = ebirdst_data_dir(),
  force = FALSE,
  show_progress = TRUE
)
```

Arguments

`species`	character; one or more species given as scientific names, common names or six-letter species codes (e.g. "woothr"). The full list of valid species can be viewed in the ebirdst_runs data frame included in this package; species with trends estimates are indicated by the `has_trends` column. To access the example dataset, use "yebsap-example".
`path`	character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling `ebirdst_data_dir()`.
`force`	logical; if the data have already been downloaded, should a fresh copy be downloaded anyway.
`show_progress`	logical; whether to print download progress information.

Value

Character vector of paths to the folders containing the downloaded data packages for the given species. The trends data will be in the `trends/` subdirectory.

Examples

```
## Not run:
# download the example data
ebirdst_download_trends("yebsap-example")

# download the data package for wood thrush
ebirdst_download_trends("woothr")

# multiple species can be downloaded at once
ebirdst_download_trends(c("Sage Thrasher", "Abert's Towhee"))

## End(Not run)
```

`ebirdst_palettes` *eBird Status and Trends color palettes for mapping*

Description

Generate the color palettes used for the eBird Status and Trends relative abundance and trends maps.

Usage

```
ebirdst_palettes(
  n,
  type = c("weekly", "breeding", "nonbreeding", "migration", "prebreeding_migration",
    "postbreeding_migration", "year_round", "trends")
)
```

Arguments

`n`	integer; the number of colors to be in the palette.
`type`	character; the type of color palette: "weekly" for the weekly relative abundance, "trends" for trends color palette, and a season name for the seasonal relative abundance. Note that for trends a diverging palette is returned, while all other palettes are sequential.

Value

A character vector of hex color codes.

Examples

```
# breeding season color palette
ebirdst_palettes(10, type = "breeding")
```

ebirdst_predictors *eBird Status and Trends predictor variables*

Description

A data frame of the predictors used in the eBird Status and Trends models. These include effort variables (e.g. distance traveled, number of observers, etc.) in addition to variables describing the environment (e.g. elevation, land cover, water cover, etc.). The environmental variables are derived by summarizing remotely sensed datasets (described in ebirdst_predictor_descriptions) over a 3 km diameter neighborhood around each checklist. For categorical datasets, two variables are generated for each class describing the percent cover (`pland`) and edge density (`ed`).

Usage

```
ebirdst_predictors
```

Format

A data frame with 150 rows and 4 columns:

- `predictor`: predictor name.
- `dataset`: dataset name, which can be cross referenced in ebirdst_predictor_descriptions for further details.
- `class`: class number or name for categorical variables.
- `label`: descriptive labels for each predictor variable.

ebirdst_predictor_descriptions *eBird Status and Trends predictors descriptions*

Description

Details on the eBird Status and Trends predictor variables or, for variables all derived from the same dataset, details on the dataset.

Usage

```
ebirdst_predictor_descriptions
```

Format

A data frame with 37 rows and 4 columns

- `dataset`: dataset name.
- `predictor`: predictor name or, if multiple variables are derived from this dataset, the pattern used to generate the names.
- `description`: detailed description of the dataset or variable.
- `reference`: a reference to consult for further information on the dataset.

`ebirdst_runs` *Data frame of species with eBird Status and Trends Data Products*

Description

A dataset listing the species for which eBird Status and Trends Data Products are available, with additional information relevant to both the Status and Trends results for each species.

Usage

```
ebirdst_runs
```

Format

A data frame with 29 variables:

- `species_code`: alphanumeric eBird species code uniquely identifying the species
- `scientific_name`: scientific name.
- `common_name`: English common name.
- `is_resident`: classifies this species a resident or a migrant.
- `breeding_quality`: breeding season quality.
- `breeding_start`: breeding season start date.
- `breeding_end`: breeding season start date.
- `nonbreeding_quality`: non-breeding season quality.
- `nonbreeding_start`: non-breeding season start date.
- `nonbreeding_end`: non-breeding season start date.
- `postbreeding_migration_quality`: post-breeding season quality.
- `postbreeding_migration_start`: post-breeding season start date.
- `postbreeding_migration_end`: post-breeding season start date.
- `prebreeding_migration_quality`: pre-breeding season quality.
- `prebreeding_migration_start`: pre-breeding season start date.
- `prebreeding_migration_end`: pre-breeding season start date.

- `resident_quality`: resident quality.
- `resident_start`: for resident species, the year-round start date.
- `resident_end`: for resident species, the year-round end date.
- `status_version_year`: the release version of the Status data products.
- `has_trends`: whether or not this species has trends estimates.
- `trends_season`: season that the trend was estimated for: breeding, nonbreeding, or resident.
- `trends_region`: the geographic region that the trend model was run for. Note that broadly distributed species (e.g. Barn Swallow) will only have trend estimates for a regional subset of their full range.
- `trends_start_year`: start year of the trend time period.
- `trends_end_year`: end year of the trend time period.
- `trends_start_date`: start date (`MM-DD` format) of the season for which the trend was estimated.
- `trends_end_date`: end date (`MM-DD` format) of the season for which the trend was estimated.
- `rsquared`: R-squared value comparing the actual and estimated trends from the simulations.
- `beta0`: the intercept of a linear model fitting actual vs. estimated trends (`actual ~ estimated`) for the simulations. Positive values of `beta0` indicate that the models are systematically *underestimating* the simulated trend for this species.
- `trends_version_year`: the release version of the Trends data products.

Details

For the Status Data Products, the dates defining the boundaries of the seasons are provided in additional to a quality rating from 0-3 for each season. These dates and quality ratings are assigned through a process of expert review. expert review. Note that missing dates imply that a season failed expert review for that species within that season.

Trends Data Products are only available for a subset of species, indicated by the `has_trends` variable, and for each species the trends is estimated for a single season. The two predictive performance metrics (`rsquared` and `beta0`) are based on a comparison of actual and estimated percent per year trends for a suite of simulations (see Fink et al. 2023 for further details). The trends regions are defined as follows:

- `aus_nz`: Australia and New Zealand
- `iberia`: Spain and Portugal
- `india_se_asia`: India, Nepal, Bhutan, Sri Lanka, Thailand, Cambodia, Malaysia, Brunei, Singapore, and Philippines
- `japan`: Japan
- `north_america`: North America including Mexico, Central America, and the Caribbean, but excluding Nunavut, North West Territories, and Hawaii
- `south_africa`: South Africa, Lesotho, and Eswatini
- `south_america`: Colombia, Ecuador, Peru, Chile, Argentina, and Uruguay
- `taiwan`: Taiwan
- `turkey_plus`: Turkey, Cyprus, Israel, Palestine, Greece, Armenia, and Georgia

ebirdst_version *eBird Status and Trends Data Products version*

Description

Identify the version of the eBird Status and Trends Data Products that this version of the R package works with. Versions are defined by the year that all model estimates are made for.

Usage

```
ebirdst_version()
```

Value

A list with three components: `status_version_year` is the version year for the eBird Status Data Products, `trends_version_year` is the version year for the eBird Trends Data Products, `release_year` is the year this version of the data were released.

Examples

```
ebirdst_version()
```

get_species *Get eBird species code for a set of species*

Description

Give a vector of species codes, common names, and/or scientific names, return a vector of 6-letter eBird species codes. This function will only look up codes for species for which eBird Status and Trends results exist.

Usage

```
get_species(x)
```

Arguments

`x`	character; vector of species codes, common names, and/or scientific names.

Value

A character vector of eBird species codes.

Examples

```
get_species(c("Black-capped Chickadee", "Poecile gambeli", "carchi"))
```

`get_species_path` *Get the path to the data package for a given species*

Description

This helper function can be used to get the path to a data package for a given species.

Usage

```
get_species_path(
  species,
  path = ebirdst_data_dir(),
  dataset = c("status", "trends"),
  check_downloaded = TRUE
)
```

Arguments

`species`	character; a single species given as a scientific name, common name or six-letter species code (e.g. "woothr"). The full list of valid species is in the ebirdst_runs data frame included in this package. To download the example dataset, use `"yebsap-example"`.
`path`	character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling `ebirdst_data_dir()`.
`dataset`	character; whether the path to the Status or Trends data products should be returned.
`check_downloaded`	logical; raise an error if no data have been downloaded for this species.

Value

The path to the data package directory.

Examples

```
## Not run:
# get the path
path <- get_species_path("yebsap-example")

# get the path to the full data package for yellow-bellied sapsucker
# common name, scientific name, or species code can be used
path <- get_species_path("Yellow-bellied Sapsucker")
path <- get_species_path("Sphyrapicus varius")
path <- get_species_path("yebsap")
```

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

`grid_sample` *Spatiotemporal grid sampling of observation data*

Description

Sample observation data on a spacetime grid to reduce spatiotemporal bias.

Usage

```
grid_sample(
  x,
  coords = c("longitude", "latitude", "day_of_year"),
  is_lonlat = TRUE,
  res = c(3000, 3000, 7),
  jitter_grid = TRUE,
  sample_size_per_cell = 1,
  cell_sample_prop = 0.75,
  keep_cell_id = FALSE,
  grid_definition = NULL
)

grid_sample_stratified(
  x,
  coords = c("longitude", "latitude", "day_of_year"),
  is_lonlat = TRUE,
  unified_grid = FALSE,
  keep_cell_id = FALSE,
  by_year = TRUE,
  case_control = TRUE,
  obs_column = "obs",
  sample_by = NULL,
  min_detection_probability = 0,
  maximum_ss = NULL,
  jitter_columns = NULL,
  jitter_sd = 0.1,
  ...
)
```

Arguments

x	data frame; observations to sample, including at least the columns defining the location in space and time. Additional columns can be included such as features that will later be used in model training.

coords — character; names of the spatial and temporal coordinates. By default the spatial spatial coordinates should be `longitude` and `latitude`, and temporal coordinate should be `day_of_year`.

is_lonlat — logical; if the points are in unprojected, lon-lat coordinates. In this case, the points will be projected to an equal area Eckert IV CRS prior to grid assignment.

res — numeric; resolution of the spatiotemporal grid in the x, y, and time dimensions. Unprojected locations are projected to an equal area coordinate system prior to sampling, and resolution should therefore be provided in units of meters. The temporal resolution should be in the native units of the time coordinate in the input data frame, typically it will be a number of days.

jitter_grid — logical; whether to jitter the location of the origin of the grid to introduce some randomness.

sample_size_per_cell — integer; number of observations to sample from each grid cell.

cell_sample_prop — proportion `(0-1]`; if less than 1, only this proportion of cells will be randomly selected for sampling.

keep_cell_id — logical; whether to retain a unique cell identifier, stored in column named `.cell_id`.

grid_definition — list defining the spatiotemporal sampling grid as returned by `assign_to_grid()` in the form of an attribute of the returned data frame.

unified_grid — logical; whether a single, unified spatiotemporal sampling grid should be defined and used for all observations in `x` or a different grid should be used for each stratum.

by_year — logical; whether the sampling should be done stratified by year (TRUE) or ignoring year (FALSE). If sampling by year is turned on, N observations will be sampled from each grid cell for each year, while if it is turned off, N observations will be sampled per grid cell across all years. If using sampling by year, the input data frame `x` must have a `year` column.

case_control — logical; whether to apply case control sampling whereby presence and absence are sampled independently.

obs_column — character; if `case_control = TRUE`, this is the name of the column in `x` that defines detection (`obs_column > 0`) and non-detection (`obs_column == 0`).

sample_by — character; additional columns in `x` to stratify sampling by. For example, if a landscape has many small islands (defined by an `island` variable) and we wish to sample from each independently, use `sample_by = "island"`.

min_detection_probability — proportion `[0-1)`; the minimum detection probability in the final dataset. If `case_control = TRUE`, and the proportion of detections in the grid sampled dataset is below this level, then additional detections will be added via grid sampling the detections from the input dataset until at least this proportion of detections appears in the final dataset. This will typically result in duplication of some observations in the final dataset. To turn this off this feature use `min_detection_probability = 0`.

maximum_ss	integer; the maximum sample size in the final dataset. If the grid sampling yields more than this number of observations, maximum_ss observations will be selected randomly from the full set. Note that this subsampling will be performed in such a way that all levels of each strata will have at least one observation within the final dataset, and therefore it is not truly randomly sampling.
jitter_columns	character; if detections are oversampled to achieve the minimum detection probability, some observations will be duplicated, and it can be desirable to slightly "jitter" the values of model training features for these duplicated observations. This argument defines the column names in x that will be jittered.
jitter_sd	numeric; strength of the jittering in units of standard deviations, see jitter_columns.
...	additional arguments defining the spatiotemporal grid; passed to grid_sample().

Details

grid_sample_stratified() performs stratified case control sampling, independently sampling from strata defined by, for example, year and detection/non-detection. Within each stratum, grid_sample() is used to sample the observations on a spatiotemporal grid. In addition, if case control sampling is turned on, the detections are oversampled to increase the frequency of detections in the dataset.

The sampling grid is defined, and assignment of locations to cells occurs, in assign_to_grid(). Consult the help for that function for further details on how the grid is generated and locations are assigned. Note that by providing 2-element vectors to both coords and res the time component of the grid can be ignored and spatial-only subsampling is performed.

Value

A data frame of the spatiotemporally sampled data.

Examples

```
set.seed(1)

# generate some example observations
n_obs <- 10000
checklists <- data.frame(longitude = rnorm(n_obs, sd = 0.1),
                         latitude = rnorm(n_obs, sd = 0.1),
                         day_of_year = sample.int(28, n_obs, replace = TRUE),
                         year = NA_integer_,
                         obs = rpois(n_obs, lambda = 0.05),
                         forest_cover = runif(n_obs),
                         island = as.integer(runif(n_obs) > 0.95))
# add a year column, giving more data to recent years
checklists$year <- sample(seq(2016, 2020), size = n_obs, replace = TRUE,
                          prob = seq(0.3, 0.7, length.out = 5))
# create several rare islands
checklists$island[sample.int(nrow(checklists), 9)] <- 2:10

# basic spatiotemporal grid sampling
sampled <- grid_sample(checklists)

# plot original data and grid sampled data
```

```
par(mar = c(0, 0, 0, 0))
plot(checklists[, c("longitude", "latitude")],
     pch = 19, cex = 0.3, col = "#00000033",
     axes = FALSE)
points(sampled[, c("longitude", "latitude")],
       pch = 19, cex = 0.3, col = "red")

# case control sampling stratified by year and island
# return a maximum of 1000 checklists
sampled_cc <- grid_sample_stratified(checklists, sample_by = "island",
                                     maximum_ss = 1000)

# case control sampling increases the prevalence of detections
mean(checklists$obs > 0)
mean(sampled$obs > 0)
mean(sampled_cc$obs > 0)

# stratifying by island ensures all levels are retained, even rare ones
table(checklists$island)
# normal grid sampling loses rare island levels
table(sampled$island)
# stratified grid sampling retain at least one observation from each level
table(sampled_cc$island)
```

`load_config` *Load eBird Status Data Products configuration file*

Description

Load the configuration file for an eBird Status run. This configuration file is mostly for internal use and contains a variety of parameters used in the modeling process.

Usage

```
load_config(species, path = ebirdst_data_dir())
```

Arguments

`species`	character; the species to load data for, given as a scientific name, common name or six-letter species code (e.g. "woothr"). The full list of valid species is in the ebirdst_runs data frame included in this package. To download the example dataset, use `"yebsap-example"`.
`path`	character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling `ebirdst_data_dir()`.

Value

A list with the run configuration parameters.

Examples

```
## Not run:
# download example data if hasn't already been downloaded
ebirdst_download_status("yebsap-example")

# load configuration parameters
p <- load_config("yebsap-example")

## End(Not run)
```

`load_data_coverage` *Load eBird Status and Trends Data Coverage Products*

Description

The data coverage products are packaged as individual GeoTIFF files for each product for each week of the year. This function loads one of the available data products for one or more weeks into R as a SpatRaster object. Note that data must be downloaded using `ebirdst_download_data_coverage()` prior to loading it using this function.

Usage

```
load_data_coverage(
  product = c("spatial-coverage", "selection-probability"),
  weeks = NULL,
  path = ebirdst_data_dir()
)
```

Arguments

`product`	character; data coverage raster product to load: spatial coverage or site selection probability.
`weeks`	character; one or more weeks (expressed in `"MM-DD"` format) to load the raster layers for. If this argument is not specified, all downloaded weeks will be loaded. **Note that these rasters are quite large so it's recommended to only load a small number of weeks of data at the same time.**
`path`	character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling `ebirdst_data_dir()`.

Details

In addition to the species-specific data products, the eBird Status data products include two products providing estimates of weekly data coverage at 3 km spatial resolution:

- `spatial-coverage`: a spatially smoothed estimate of the proportion of the area that was covered by eBird checklists for the given week.
- `selection-probability`: a modeled estimate of the probability that the given location and habitat was sampled by eBird data in the given week.

Value

A SpatRaster with between 1 and 52 layers for the given product for the given weeks, where the layer names are the dates (`YYYY-MM-DD` format) of the midpoint of each week.

Examples

```
## Not run:
# download example data if hasn't already been downloaded
ebirdst_download_data_coverage()

# load a single week of site selection probability data
load_data_coverage("selection-probability", weeks = "01-04")

# load all weeks of spatial coverage data
load_data_coverage("spatial-coverage", weeks = c("01-04", "01-11"))

## End(Not run)
```

`load_fac_map_parameters`

Load full annual cycle map parameters

Description

Get the map parameters used on the eBird Status and Trends website to optimally display the full annual cycle data. This includes bins for the abundance data, a projection, and an extent to map. The extent is the spatial extent of non-zero data across the full annual cycle and the projection is optimized for this extent.

Usage

```
load_fac_map_parameters(species, path = ebirdst_data_dir())
```

Arguments

species	character; the species to load data for, given as a scientific name, common name or six-letter species code (e.g. "woothr"). The full list of valid species is in the ebirdst_runs data frame included in this package. To download the example dataset, use "yebsap-example".
path	character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling ebirdst_data_dir().

Value

A list containing elements:

- custom_projection: a custom projection optimized for the given species' full annual cycle
- fa_extent: a SpatExtent object storing the spatial extent of non-zero data for the given species in the custom projection
- res: a numeric vector with 2 elements giving the target resolution of raster in the custom projection
- fa_extent_projected: the extent in projected (Equal Earth) coordinates
- weekly_bins/weekly_labels: weekly abundance bins and labels for the full annual cycle
- seasonal_bins/'seasonal_labels: seasonal abundance bins and labels for the full annual cycle

Examples

```
## Not run:
# download example data if hasn't already been downloaded
ebirdst_download_status("yebsap-example")

# load configuration parameters
load_fac_map_parameters(path)

## End(Not run)
```

---

load_pi *Load predictor importance (PI) rasters*

Description

The eBird Status models estimate the relative importance of each of the core environmental predictor used in the model (i.e. the % land and water cover variables). These predictor importance (PI) data are converted to ranks (with a rank of 1 being the most important) relative to the full suite of environmental predictors. The ranks are summarized to a 27 km resolution raster grid for each predictor, where the cell values are the average across all models in the ensemble contributing to that cell. These data are available in raster format provided download_pis = TRUE was

used when calling ebirdst_download_status(). PI estimates are available separately for both the occurrence and count sub-model and only the 30 most important predictors are distributed. Use list_available_pis() to see which predictors have PI data.

Usage

```
load_pi(
  species,
  predictor,
  response = c("occurrence", "count"),
  path = ebirdst_data_dir()
)

list_available_pis(species, path = ebirdst_data_dir())
```

Arguments

species	character; the species to load data for, given as a scientific name, common name or six-letter species code (e.g. "woothr"). The full list of valid species is in the ebirdst_runs data frame included in this package. To download the example dataset, use "yebsap-example".
predictor	character; the predictor that the PI data should be loaded for. The list of predictors that PI data are available for varies by species, use list_available_pis() to get the list for a given species.
response	character; the model (occurrence or count) that the PI data should be loaded for.
path	character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling ebirdst_data_dir().

Value

A SpatRaster object with the PI ranks for the given predictor. For migrants, the estimates are weekly and the raster will have 52 layers, where the layer names are the dates (MM-DD format) of the midpoint of each week. For residents, a single year round layer is returned.

list_available_pis() returns a data frame listing the top 30 predictors for which PI rasters can be loaded. In addition to the predictor names, the mean range-wide rank (rank_mean) is given as well as the integer rank (rank) relative to the full suite of predictors (environmental and effort).

Functions

- list_available_pis(): list the predictors that have PI information for this species.

Examples

```
## Not run:
# download example data if hasn't already been downloaded
ebirdst_download_status("yebsap-example", download_pis = TRUE)
```

```
# identify the top predictor
top_preds <- list_available_pis("yebsap-example")
print(top_preds[1, ])

# load predictor importance raster of top predictor for occurrence
load_pi("yebsap-example", top_preds$predictor[1])

## End(Not run)
```

---

load_ppm *Load predictive performance metric (PPM) rasters*

Description

eBird Status models are evaluated against a test set of eBird data not used during model training and a suite of predictive performance metrics (PPMs) are calculated. The PPMs for each base model are summarized to a 27 km resolution raster grid, where the cell values are the average across all models in the ensemble contributing to that cell. These data are available in raster format provided `download_ppms = TRUE` was used when calling `ebirdst_download_status()`.

Usage

```
load_ppm(
  species,
  ppm = c("binary_f1", "binary_mcc", "binary_prevalence", "occ_bernoulli_dev",
    "occ_bin_spearman", "occ_brier", "occ_pr_auc", "occ_pr_auc_gt_prev",
   "occ_pr_auc_normalized", "count_log_pearson", "count_mae", "count_poisson_dev",
   "count_rmse", "count_spearman", "abd_log_pearson", "abd_mae", "abd_poisson_dev",
    "abd_rmse", "abd_spearman"),
  path = ebirdst_data_dir()
)
```

Arguments

species	character; the species to load data for, given as a scientific name, common name or six-letter species code (e.g. "woothr"). The full list of valid species is in the ebirdst_runs data frame included in this package. To download the example dataset, use `"yebsap-example"`.
ppm	character; the name of a single metric to load data for. See Details for definitions of each metric.
path	character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling `ebirdst_data_dir()`.

Details

Nineteen predictive performance metrics are provided:

- `binary_f1`: F1-score comparing the model predictions converted to binary with the observed detection/non-detection for the test checklists.
- `binary_mcc`: Matthews Correlation Coefficient (MCC) comparing the model predictions converted to binary with the observed detection/non-detection for the test checklists.
- `binary_prevalence`: the observed detection probability after spatiotemporal subsampling.
- `occ_bernoulli_dev`: proportion of Bernoulli deviance explained comparing the predicted occurrence with the observed detection/non-detection for the test checklists.
- `occ_bin_spearman`: test observations are binned by predicted encounter rate with bin widths of 0.05, then the mean observed prevalence and predicted encounter rate are calculated within bins. This metric is the Spearman's rank correlation coefficient comparing the observed and predicted binned mean values.
- `occ_brier`: the Brier score is the mean squared difference between predicted encounter rate and observed detection/non-detection.
- `occ_pr_auc`: the area on the precision-recall curve (PR AUC) generated by comparing the predicted encounter rate with the observed detection/non-detection for the test checklists.
- `occ_pr_auc_gt_prev`: the proportion of the ensemble for which the PR AUC is greater than observed prevalence, which indicates that the model is performing better than random guessing.
- `occ_pr_auc_normalized`: the PR AUC normalized to account for class imbalance so that a value of 0 represents performance equal to random guessing and a value of 1 represents perfect classification.
- `count_log_pearson`: Pearson correlation coefficient comparing the logarithm of the predicted count with the logarithm of the observed count for the subset of test checklists on which the species was detected.
- `count_mae`: the mean absolute error (MAE) comparing the observed and predicted counts for the subset of test checklists on which the species was detected.
- `count_poisson_dev`: proportion of Poisson deviance explained, comparing the observed and predicted counts for the subset of test checklists on which the species was detected.
- `count_rmse`: route mean squared error (RMSE) comparing the observed and predicted counts for the subset of test checklists on which the species was detected.
- `count_spearman`: Spearman's rank correlation coefficient comparing the observed and predicted counts for the subset of test checklists on which the species was detected.
- `abd_log_pearson`: Pearson correlation coefficient comparing the logarithm of the predicted relative abundance with the logarithm of the observed count for the full set of test checklists.
- `abd_mae`: the mean absolute error (MAE) comparing the observed counts and predicted relative abundance for the full set of test checklists.
- `abd_poisson_dev`: proportion of Poisson deviance explained, comparing the predicted relative abundance with the observed count for the full set of test checklists.
- `abd_rmse`: root mean squared error comparing the predicted relative abundance with the observed count for the full set of test checklists.
- `abd_spearman`: Spearman's rank correlation coefficient comparing the predicted relative abundance with the observed count for the full set of test checklists.

Value

A SpatRaster object with the PPM data. For migrants, rasters are weekly with 52 layers, where the layer names are the dates (MM-DD format) of the midpoint of each week. For residents, a single year round layer is returned.

Examples

```
## Not run:
# download example data if hasn't already been downloaded
ebirdst_download_status("yebsap-example", download_ppms = TRUE)

# load area under the precision-recall curve PPM raster
load_ppm("yebsap-example", ppm = "binary_pr_auc")

## End(Not run)
```

---

load_ranges *Load seasonal eBird Status and Trends range polygons*

Description

Range polygons are defined as the boundaries of non-zero seasonal relative abundance estimates, which are then (optionally) smoothed to produce more aesthetically pleasing polygons using the smoothr package.

Usage

```
load_ranges(
  species,
  resolution = c("9km", "27km"),
  smoothed = TRUE,
  path = ebirdst_data_dir()
)
```

Arguments

species	character; the species to load data for, given as a scientific name, common name or six-letter species code (e.g. "woothr"). The full list of valid species is in the ebirdst_runs data frame included in this package. To download the example dataset, use "yebsap-example".
resolution	character; the raster resolution from which the range polygons were derived.
smoothed	logical; whether smoothed or unsmoothed ranges should be loaded.
path	character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling ebirdst_data_dir().

Value

An sf update containing the seasonal range boundaries, with each season provided as a different feature.

Examples

```
## Not run:
# download example data if hasn't already been downloaded
ebirdst_download_status("yebsap-example")

# load smoothed ranges
# note that only 27 km data are provided for the example data
ranges <- load_ranges("yebsap-example", resolution = "27km")

## End(Not run)
```

---

load_raster *Load eBird Status Data Products raster data*

Description

Each of the eBird Status raster products is packaged as a GeoTIFF file representing predictions on a regular grid. The core products are occurrence, count, relative abundance, and proportion of population. This function loads one of the available data products into R as a SpatRaster object. Note that data must be downloaded using ebirdst_download_status() prior to loading it using this function.

Usage

```
load_raster(
  species,
  product = c("abundance", "count", "occurrence", "proportion-population"),
  period = c("weekly", "seasonal", "full-year"),
  metric = NULL,
  resolution = c("3km", "9km", "27km"),
  path = ebirdst_data_dir()
)
```

Arguments

species	character; the species to load data for, given as a scientific name, common name or six-letter species code (e.g. "woothr"). The full list of valid species is in the ebirdst_runs data frame included in this package. To download the example dataset, use "yebsap-example".
product	character; eBird Status raster product to load: occurrence, count, relative abundance, or proportion of population. See Details for a detailed explanation of each of these products.

`period` character; temporal period of the estimation. The eBird Status models make predictions for each week of the year; however, as a convenience, data are also provided summarized at the seasonal or annual ("full-year") level.

`metric` character; by default, the weekly products provide estimates of the median value (`metric = "median"`) and the summarized products are the cell-wise mean across the weeks within the season (`metric = "mean"`). However, additional variants exist for some of the products. For the weekly relative abundance, confidence intervals are provided: specify `metric = "lower"` to get the 10th quantile or `metric = "upper"` to get the 90th quantile. For the seasonal and annual products, the cell-wise maximum values across weeks can be obtained with `metric = "max"`.

`resolution` character; the resolution of the raster data to load. The default is to load the native 3 km resolution data; however, for some applications 9 km or 27 km data may be suitable.

`path` character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling `ebirdst_data_dir()`.

Details

The core eBird Status data products provide weekly estimates across a regular spatial grid. They are packaged as rasters with 52 layers, each corresponding to estimates for a week of the year, and we refer to them as "cubes" (e.g. the "relative abundance cube"). All estimates are the median expected value for a standard 2 km, 1 hour eBird Traveling Count by an expert eBird observer at the optimal time of day and for optimal weather conditions to observe the given species. These products are:

- `occurrence`: the expected probability (0-1) of occurrence of a species.
- `count`: the expected count of a species, conditional on its occurrence at the given location.
- `abundance`: the expected relative abundance of a species, computed as the product of the probability of occurrence and the count conditional on occurrence.
- `proportion-population`: the proportion of the total relative abundance within each cell. This is a derived product calculated by dividing each cell value in the relative abundance raster by the total abundance summed across all cells.

In addition to these weekly data cubes, this function provides access to data summarized over different periods. Seasonal cubes are produced by taking the cell-wise mean or max across the weeks within each season. The boundary dates for each season are species specific and are available in `ebirdst_runs`, and if a season failed review no associated layer will be included in the cube. In addition, full-year summaries provide the mean or max across all weeks of the year that fall within a season that passed review. Note that this is not necessarily all 52 weeks of the year. For example, if the estimates for the non-breeding season failed expert review for a given species, the full-year summary for that species will not include the weeks that would fall within the non-breeding season.

Value

For the weekly cubes, a SpatRaster with 52 layers for the given product, where the layer names are the dates (YYYY-MM-DD format) of the midpoint of each week. Seasonal cubes will have up to four layers named with the corresponding season. The full-year products will have a single layer.

Examples

```
## Not run:
# download example data if hasn't already been downloaded
ebirdst_download_status("yebsap-example")

# weekly relative abundance
# note that only 27 km data are available for the example data
abd_weekly <- load_raster("yebsap-example", "abundance", resolution = "27km")

# the weeks for each layer are stored in the layer names
names(abd_weekly)
# they can be converted to date objects with as.Date
as.Date(names(abd_weekly))

# max seasonal abundance
abd_seasonal <- load_raster("yebsap-example", "abundance",
                            period = "seasonal", metric = "max",
                            resolution = "27km")
# available seasons in stack
names(abd_seasonal)
# subset to just breeding season abundance
abd_seasonal[["breeding"]]

## End(Not run)
```

---

load_regional_stats *Load regional summary statistics*

Description

Load seasonal summary statistics for regions consisting of countries and states/provinces.

Usage

```
load_regional_stats(species, path = ebirdst_data_dir())
```

Arguments

species	character; the species to load data for, given as a scientific name, common name or six-letter species code (e.g. "woothr"). The full list of valid species is in the ebirdst_runs data frame included in this package. To download the example dataset, use "yebsap-example".

path — character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling `ebirdst_data_dir()`.

Value

A data frame containing regional summary statistics with columns:

- `species_code`: alphanumeric eBird species code.
- `region_type`: `country` for countries or `state` for states, provinces, or other sub-national regions.
- `region_code`: alphanumeric code for the region.
- `region_name`: English name of the region.
- `continent_code`: alphanumeric code for continent that this region belongs to.
- `continent_name`: name of the continent that this region belongs to.
- `season`: name of the season that the summary statistics were calculated for.
- `abundance_mean`: mean relative abundance in the region.
- `total_pop_percent`: proportion of the seasonal modeled population falling within the region.
- `range_percent_occupied`: the proportion of the region occupied by the species during the given season.
- `range_total_percent`: the proportion of the species seasonal range falling within the region.
- `range_days_occupation`: number of days of the season that the region was occupied by this species.

Examples

```
## Not run:
# download example data if hasn't already been downloaded
ebirdst_download_status("yebsap-example")

# load configuration parameters
regional <- load_regional_stats("yebsap-example")

## End(Not run)
```

`load_trends` *Load eBird Trends estimates for a set of species*

Description

Load the relative abundance trend estimates for a single species or a set of species. Trends are estimated on a 27 km by 27 km grid for a single season per species (breeding, non-breeding, or resident). Note that data must be downloaded using `ebirdst_download_trends()` prior to loading it using this function.

Usage

```
load_trends(species, fold_estimates = FALSE, path = ebirdst_data_dir())
```

Arguments

`species`	character; one or more species given as scientific names, common names or six-letter species codes (e.g. "woothr"). The full list of valid species can be viewed in the ebirdst_runs data frame included in this package; species with trends estimates are indicated by the `has_trends` column. To access the example dataset, use "yebsap-example".
`fold_estimates`	logical; by default, the trends summarized across the 100-fold ensemble are returned; however, by setting `fold_estimates = TRUE` the individual fold-level estimates are returned.
`path`	character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named for the data version year, e.g. "2020" for the 2020 Status Data Products. Each species' data package will then appear in a directory named with the eBird species code. Defaults to a persistent data directory, which can be found by calling `ebirdst_data_dir()`.

Details

The trends in relative abundance are estimated using a double machine learning model. To quantify uncertainty, an ensemble of 100 estimates is made at each location, each based on a random subsample of eBird data. The estimated trend is the median across the ensemble, and the 80% confidence intervals are the lower 10th and upper 90th percentiles across the ensemble. To access estimates from the individual folds making up the ensemble use `fold_estimates = TRUE`. These fold-level estimates can be used to quantify uncertainty, for example, when calculating the trend for a given region. For further details on the methodology used to estimate trends consult Fink et al. 2023.

Value

A data frame containing the trends estimates for a set of species. The following columns are included:

- `species_code`: the alphanumeric eBird species code uniquely identifying the species.
- `season`: season that the trend was estimated for: breeding, non-breeding, or resident.

- start_year/end_year: the start and end years of the trend time period.
- start_date/end_date: the start and end dates (MM-DD format) of the season for which the trend was estimated.
- srd_id: unique integer identifier for the grid cell.
- longitude/latitude: longitude and latitude of the grid cell center.
- abd: relative abundance estimate for the middle of the trend time period (e.g. 2014 for a 2007-2021 trend).
- abd_ppy: the median estimated percent per year change in relative abundance.
- abd_ppy_lower/abd_ppy_upper: the 80% confidence interval for the estimated percent per year change in relative abundance.
- abd_ppy_nonzero: a logical (TRUE/FALSE) value indicating if the 80% confidence limits overlap zero (FALSE) or don't overlap zero (TRUE)
- abd_trend: the median estimated cumulative change in relative abundance over the trend time period.
- abd_trend_lower/abd_trend_upper: the 80% confidence interval for the estimated cumulative change in relative abundance over the trend time period.

If fold_estimates = TRUE, a data frame of fold-level trend estimates is returned with the following columns:

- species_code: the alphanumeric eBird species code uniquely identifying the species.
- season: season that the trend was estimated for: breeding, non-breeding, or resident.
- srd_id: unique integer identifier for the grid cell.
- abd: relative abundance estimate for the middle of the trend time period (e.g. 2014 for a 2007-2021 trend).
- abd_ppy: the estimated percent per year change in relative abundance.

References

Fink, D., Johnston, A., Strimas-Mackey, M., Auer, T., Hochachka, W. M., Ligocki, S., Oldham Jaromczyk, L., Robinson, O., Wood, C., Kelling, S., & Rodewald, A. D. (2023). A Double machine learning trend model for citizen science data. Methods in Ecology and Evolution, 00, 1–14. https://doi.org/10.1111/2041-210X.14186

Examples

```
## Not run:
# download example trends data if it hasn't already been downloaded
ebirdst_download_trends("yebsap-example")

# load trends
trends <- load_trends("yebsap-example")

# load fold-level estimates
trends_folds <- load_trends("yebsap-example", fold_estimates = TRUE)

## End(Not run)
```

`rasterize_trends` *Convert eBird Trends Data Products to raster format*

Description

The eBird trends data are stored in a tabular format, where each row gives the trend estimate for a single cell in a 27 km x 27 km equal area grid. For many applications, an explicitly spatial format is more useful. This function uses the cell center coordinates to convert the tabular trend estimates to raster format in `terra` SpatRaster format.

Usage

```
rasterize_trends(
  trends,
  layers = c("abd_ppy", "abd_ppy_lower", "abd_ppy_upper"),
  trim = TRUE
)
```

Arguments

`trends`	data frame; trends data for a single species as returned by `load_trends()`.
`layers`	character; column names in the trends data frame to rasterize. These columns will become layers in the raster that is created.
`trim`	logical; flag indicating if the returned raster should be trimmed to remove outer rows and columns that are NA. If `trim = FALSE` the returned raster will have a global extent, which can be useful if rasters will be combined across species with different ranges.

Value

A SpatRaster object.

Examples

```
## Not run:
# download example trends data if it hasn't already been downloaded
ebirdst_download_trends("yebsap-example")

# load trends
trends <- load_trends("yebsap-example")

# rasterize percent per year trend
rasterize_trends(trends, "abd_ppy")

## End(Not run)
```

set_ebirdst_access_key *Store the eBird Status and Trends access key*

Description

Accessing eBird Status and Trends data requires an access key, which can be obtained by visiting https://ebird.org/st/request. This key must be stored as the environment variable `EBIRDST_KEY` in order for `ebirdst_download_status()` and `ebirdst_download_trends()` to use it. The easiest approach is to store the key in your `.Renviron` file so it can always be accessed in your R sessions. Use this function to set `EBIRDST_KEY` in your `.Renviron` file provided that it is located in the standard location in your home directory. It is also possible to manually edit the `.Renviron` file. **The access key is specific to you and should never be shared or made publicly accessible.**

Usage

```
set_ebirdst_access_key(key, overwrite = FALSE)
```

Arguments

`key`	character; API key obtained by filling out the form at https://ebird.org/st/request.
`overwrite`	logical; should the existing `EBIRDST_KEY` be overwritten if it has already been set in .Renviron.

Value

Edits .Renviron, then returns the path to this file invisibly.

Examples

```
## Not run:
# save the api key, replace XXXXXX with your actual key
set_ebirdst_access_key("XXXXXX")

## End(Not run)
```

vectorize_trends *Convert Trends Data Products to points or circles*

Description

The eBird trends data are stored in a tabular format, where each row gives the trend estimate for a single cell in a 27 km x 27 km equal area grid. For many applications, an explicitly spatial format is more useful. This function uses the cell center coordinates to convert the tabular trend estimates to points or circles in sf format. Trends can be converted to points or to circles with areas roughly proportional to the relative abundance within that 27 km grid cell. These abundance-scaled circles are what is used to produce the trends maps on the eBird Status and Trends website.

Usage

```
vectorize_trends(trends, output = c("circles", "points"), crs = 4326)
```

Arguments

trends	data frame; trends data for a single species as returned by load_trends().
output	character; "points" outputs spatial points while "circles" outputs circles with areas roughly proportional to the relative abundance within that 27 km grid cell.
crs	character or sf crs object; coordinate reference system to output the results in. For points, unprojected latitude-longitude coordinates (the default) are most typical, while for circles use whatever equal area CRS you intend to use when mapping the data otherwise the "circles" will appear skewed.

Value

Vetorized trends data as an sf object.

Examples

```
## Not run:
# download example trends data if it hasn't already been downloaded
ebirdst_download_trends("yebsap-example")

# load trends
trends <- load_trends("yebsap-example")

# vectorize as points
vectorize_trends(trends, "points")
# vectorize as circles
vectorize_trends(trends, "circles", crs = "+proj=eqearth")

## End(Not run)
```

Index