Package 'ebirdst'

November 12, 2021

Type Package

Title Access and Analyze eBird Status and Trends Data

Version 0.3.3

Description Tools to download, map, plot and analyze eBird Status and
Trends data (<https://ebird.org/science/status-and-trends>). eBird
(<https://ebird.org/home>) is a global database of bird observations
collected by citizen scientists. eBird Status and Trends uses these
data to model continental bird abundances, range boundaries, habitat
associations, and trends.

License GPL-3

URL https://github.com/CornellLabofOrnithology/ebirdst

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

Depends R (>= 3.3.0)

Imports DBI, dplyr (>= 0.7.0), fasterize, gbm, ggplot2, grDevices,
jsonlite, gridExtra, magrittr, methods, mgcv, PresenceAbsence,
rappdirs, raster, rgdal, rlang, RSQLite, sf (>= 1.0-0), stats,
stringr, tidyr (>= 1.0.0), tools, utils, viridisLite, xml2

Suggests covr, fields, knitr, rmarkdown, rnaturalearth, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2021-11-12 18:30:02 UTC

R topics documented:

---

abundance_palette   *eBird Status and Trends color palettes for mapping*

Description

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

Usage

```
abundance_palette(
  n,
 season = c("weekly", "breeding", "nonbreeding", "migration", "prebreeding_migration",
    "postbreeding_migration", "year_round")
)
```

Arguments

n	integer; the number of colors to be in the palette.
season	character; the season to generate colors for or "weekly" to get the color palette used in the weekly abundance animations.

Value

A character vector of hex color codes.

Examples

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

---

bernoulli_dev	*Bernoulli deviance*

Description

Bernoulli deviance

Usage

```
bernoulli_dev(obs, pred)
```

Arguments

obs	numeric; observed values.
pred	numeric; predicted values.

Value

A named numeric vector with three elements: model deviance, mean deviance, and deviance explained.

Examples

```
obs <- c(1, 1, 1, 0, 0, 0)
pred <- c(0.9, 0.8, 0.7, 0.3, 0.1, 0.2)
ebirdst:::bernoulli_dev(obs, pred)
```

binom_test_p	*Binomial test for ensemble support*

Description

Binomial test for ensemble support

Usage

```
binom_test_p(x, pat_cutoff = 1/10)
```

Arguments

x	numeric; named numeric vector with values for "pat" and "pi_es".
pat_cutoff	numeric; percent above threshold cutoff

Value

A numeric p-value.

Examples

```
ebirdst:::binom_test_p(c(pat = 0.1, pi_es = 75))
```

calc_bins	*Calculates relative abundance bins (breaks) based for mapping*

Description

Mapping species abundance across the full-annual cycle presents a challenge, in that patterns of concentration and dispersion in abundance change throughout the year, making it difficult to define color bins that suit all seasons and accurately reflect the detail of abundance predictions. To address this, when mapping the relative abundance data, we recommend using quantile bins based on the underlying count distribution, adjusted according to the relative abundance distribution. To access pre-calculated bins for the full annual cycle use [load_fac_map_parameters()](#).

Usage

```
calc_bins(abundance, count)
```

Arguments

abundance	Raster object; eBird Status and Trends relative abundance data cube, or a subset of the data cube.
count	Raster object; eBird Status and Trends count data cube, or a subset of the data cube.

Value

A numeric vector defining the breaks of the relative abundance bins. In addition, the labels attribute of this vector provides the 5th, 50th, and 95th quantile of abundance, which can be used to label the bottom, middle, and top of a legend.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data")
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# abundance data
abd <- load_raster(path, "abundance")
# count data
cnt <- load_raster(path, "count")

# calculate bins for a single week for this example
bins <- calc_bins(abd, cnt)

## End(Not run)
```

---

calc_full_extent   *Calculate the spatial extent of non-zero data in a raster*

Description

eBird Status and Trends data cubes are defined over broad areas, filling in regions where the species doesn't occur with zeros (predicted absences) or NAs (regions where models weren't fit). When producing maps, it's best to only display the spatial extent where the species occurs. To show determine an ideal extent for mapping, this function trims away 0 and NA values. When called on a RasterStack (e.g., a data cube consisting of all 52 weeks), this function returns the extent of occurrence across all layers.To access a pre-calculated extent for the full annual cycle use [load_fac_map_parameters()](#).

Usage

```
calc_full_extent(x, aggregate = TRUE)
```

Arguments

x	Raster object; either a full 52-week data cube or a subset.
aggregate	logical; whether data should be aggregated by a factor of 3 in each dimension prior to calculating the extent. When working with the high resolution cubes, data should be aggregated otherwise processing times will be extremely long.

Value

The extent of occurrence as a raster [Extent](#) object.

Examples

```
# simple toy example
r <- raster::raster(nrow = 100, ncol = 100)
r[5025:5075] <- 1
raster::extent(r)
calc_full_extent(r)

## Not run:
# download example data
path <- ebirdst_download("example_data")
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# load abundance data
abd <- load_raster(path, "abundance")

# calculate full extent
map_extent <- calc_full_extent(abd)

# plot
raster::plot(abd[[20]], axes = FALSE, ext = map_extent)

## End(Not run)
```

---

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)
```

Arguments

dates   a vector of dates.

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	ebirdst: *Tools to Load, Map, Plot, and Analyze eBird Status and Trends Data Products*

Description

Tools to load, map, plot, and analyze eBird Status and Trends data products

---

ebirdst_download	*Download eBird Status and Trends Data*

Description

Download an eBird Status and Trends data package for a single species, or for an example species, to a specified path. Accessing 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. In addition, the example data are accessible without an access key.

Usage

```
ebirdst_download(
  species,
  path = rappdirs::user_data_dir("ebirdst"),
  tifs_only = TRUE,
  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 can be viewed in the ebirdst_runs data frame included in this package. To download the example dataset, use "example_data".

path   character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named according to the unique run ID associated with this species. Defaults to a persistent data directory, which can be found by calling rappdirs::user_data_dir("ebirdst")).

tifs_only	logical; whether to only download the GeoTIFFs for abundance and occurrence (the default), or download the entire data package, including data for predictor importance, partial dependence, and predictive performance metrics.
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 package for the given species.

Examples

```
## Not run:
# download the example data
ebirdst_download("example_data")

# download the data package for wood thrush, geotiffs only
ebirdst_download("woothr")
# download the data package for wood thrush, all data
ebirdst_download("woothr", tifs_only = FALSE)

## End(Not run)
```

| ebirdst_extent | *Construct a spatiotemporal extent object to subset Status and Trends data* |

Description

ebirdst_extent object are used to subset the eBird Status and Trends data spatially and temporally. This function constructs these objects.

Usage

```
ebirdst_extent(x, t, ...)

## S3 method for class 'bbox'
ebirdst_extent(x, t, ...)

## S3 method for class 'numeric'
ebirdst_extent(x, t, crs = 4326, ...)

## S3 method for class 'sfc'
ebirdst_extent(x, t, ...)

## S3 method for class 'sf'
ebirdst_extent(x, t, ...)
```

Arguments

x	the spatial extent; either a rectangular bounding box (defined as a vector of numbers representing the coordinates of the boundaries or an st_bbox() object) or a polygon (an sf object). See Details for further explanation of the format of x.
t	the temporal extent; a 2-element vector of the start and end dates of the temporal extent, provided either as dates (Date objects or strings in ISO format "YYYY-MM-DD") or numbers between 0 and 1 representing the fraction of the year. Note that dates can wrap around the year, e.g. 'c("2018-12-01", "2018-01-31") is acceptable. See Details for further explanation of the format of t. **Leave the argument blank to include the full year of data.**
...	Additional arguments used by methods.
crs	coordinate reference system, provided as a crs object or argument to st_crs(). Defaults to unprojected, lat/long coordinates (crs = 4326). **Only required if x is given as a numeric vector defining the bounding box, ignored in all other cases.**

Details

The spatial extent, x, can be either a rectangular bounding box or a set of spatial polygons. The bounding box can be defined either as an st_bbox() object or by providing the coordinates of the rectangle edges directly as a named vector with elements xmin, xmax, ymin, and ymax (note that latitude and longitude correspond to y and x, respectively). In this latter case, a coordinate reference system must be provided explicitly via the crs argument (crs = 4326 is the default and is a short form for unprojected lat/long coordinates). For a polygon spatial extent, x should be either an sf or sfc object (with feature type POLYGON or MULTIPOLYGON) from the sf package. To import data from a Shapefile or GeoPackage into this format, use read_sf().

The temporal extent defines the start and end dates of the time period. These are most easily provided as Date objects or date strings in ISO format ("YYYY-MM-DD"). If dates are defined as strings, the year can be omitted (i.e. "MM-DD"). Alternatively, dates can be defined in terms of fractions of the year, e.g. t = c(0.25, 0.5) would subset to data within the second quarter of the year. In all cases, dates can wrap around the year, e.g. c("2018-12-01", "2018-01-31") would subset to data in December or January.

Value

An ebirdst_extent object consisting of a list with three elements: the spatial extent extent, the temporal extent t, and type (either "bbox" or "polygon").

Methods (by class)

- bbox: bounding box created with st_bbox()
- numeric: bounding box given as edges
- sfc: polygons as sfc spatial feature column
- sf: polygons as sf object

Examples

```
# bounding box of the north eastern united stats as a numeric vector
bb_vec <- c(xmin = -80, xmax = -70, ymin = 40, ymax = 47)
ebirdst_extent(bb_vec)

# bbox object
bb <- sf::st_bbox(bb_vec, crs = 4326)
ebirdst_extent(bb)

# polygon imported from a shapefile
poly <- sf::read_sf(system.file("shape/nc.shp", package="sf"))
ebirdst_extent(poly)

# subset to january
ebirdst_extent(bb, t = c("2018-01-01", "2018-01-31"))

# dates can wrap around, e.g. to use dec-jan
ebirdst_extent(bb, t = c("2018-12-01", "2018-01-31"))

# dates can also be given without an associated year
ebirdst_extent(bb, t = c("12-01", "01-31"))
```

---

ebirdst_habitat   *eBird Status and Trends predictive habitat associations*

Description

Combine the predictor importance (PI) and partial dependence (PD) data to provide an estimate of the importance and directionality of the land cover classes (i.e. habitat) used as covariates in the occurrence probability model. **Note:** This is one of, if not the most, computationally expensive operations in the package.

Usage

```
ebirdst_habitat(path, ext, pis = NULL, pds = NULL, stixels = NULL)

## S3 method for class 'ebirdst_habitat'
plot(x, n_predictors = 15, date_range = c(0, 1), ...)
```

Arguments

path   character; directory that the Status and Trends data for a given species was downloaded to. This path is returned by ebirdst_download() or get_species_path().

ext   [ebirdst_extent](#) object; the spatiotemporal extent over which to calculate the habitat associations. Note that **temporal component of** ext **is ignored is this function**, habitat associations are always calculated for the full year.

pis, pds, stixels

> as an alternative to providing the `path` argument specifying the location of the data package, the data required to calculate habitat associations can be provided explicitly. PI, PD, and stixel data frames can provided, which come from the `load_pis()`, `load_pds()`, and `load_stixels()` functions, respectively. Ignored if `path` is provided. **In most cases, users will want to avoid using these arguments and simply provide** `path` **instead.**

x   `ebirdst_habitat` object; habitat relationships as calculated by `ebirdst_habitat()`.

n_predictors   number of predictors to include in the cake plot. The most important set of predictors will be chosen based on the maximum weekly importance value across the whole year.

date_range   the range of dates for plotting; a 2-element vector of the start and end dates of the date range, provided either as dates (Date objects or strings in ISO format "YYYY-MM-DD") or numbers between 0 and 1 representing the fraction of the year. When providing dates as a string, the year can be omitted (i.e. "MM-DD"). By default the full year of data are plotted.

...   ignored.

Details

The Status and Trends models use both effort (e.g. number of observers, length of checklist) and habitat (e.g. elevation, percent forest cover) covariates; for the full list consult ebirdst_predictors. This function calculates habitat associations only for the following covariates that most closely represent metrics of available habitat. In all cases these are calculated within a 1.5 km radius of each checklist:

- Land cover: percent of each landcover class
- Water cover: percent of each watercover class
- Intertidal: percent cover of intertidal mudflats
- Nighttime lights: total refelctance of nighttime lights
- Roads: road density. There are 5 covariates distinguishing between different road types; however, these are grouped together for the sake of the habitat associations.

The `plot()` method can be used to produce a cake plot, a stacked area chart showing habitat associations in which area indicates the importance of a given land cover class and the position above or below the x-axis indicates the direction of the relationship.

Value

An `ebirdst_habitat` object, consisting of a data frame giving the predictor importance and directionality for each predictor for each week of the year. The columns are:

- `predictor`: the name of the predictor
- `date`: the week centroid expressed as a continuous value between 0-1. See ebirdst_weeks to convert these values to ISO dates.
- `importance`: the relative importance of the predictor, these values are scaled so they sum to 1 within each week.
- `direction`: the direction of the relationship, either 1 for a positive relationship, -1 for a negative relationship, or NA when the direction of the relationship is not significant.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# define a spatial extent to calculate ppms over
bb_vec <- c(xmin = -86, xmax = -83, ymin = 42.5, ymax = 44.5)
e <- ebirdst_extent(bb_vec)

# compute habitat associations
habitat <- ebirdst_habitat(path = path, ext = e)
print(habitat)
# produce a cake plot
plot(habitat)

## End(Not run)
```

---

ebirdst_ppms   *eBird Status and Trends predictive performance metrics (PPMs)*

Description

Calculate a suite of predictive performance metrics (PPMs) for the eBird Status and Trends model
of a given species within a spatiotemporal extent.

Usage

```
ebirdst_ppms(path, ext, es_cutoff)

## S3 method for class 'ebirdst_ppms'
plot(x, ...)
```

Arguments

path	character; directory that the Status and Trends data for a given species was downloaded to. This path is returned by ebirdst_download() or get_species_path().
ext	[ebirdst_extent](#) object (optional); the spatiotemporal extent over which to calculate the PPMs.
es_cutoff	integer between 0-100; the ensemble support cutoff to use in distinguishing zero and non-zero predictions. Optimal ensemble support cutoff values are calculated for each week during the modeling process and stored in the data package for each species. **In general, you should not specify a value for** es_cutoff **and instead allow the function to use the species-specific model-base values.**
x	[ebirdst_ppms](#) object; PPMs as calculated by [ebirdst_ppms()](#).
...	ignored.

Details

During the eBird Status and Trends modeling process, a subset of observations (the "test data") are held out from model fitting to be used for evaluating model performance. Model predictions are made for each of these observations and this function calculates a suite of predictive performance metrics (PPMs) by comparing the predictions with the observed count on the eBird checklist.

Three types of PPMs are calculated: binary or range-based PPMs assess the ability of model to predict range boundaries, occurrence PPMs assess the occurrence probability predictions, and abundance PPMs assess the predicted abundance. Both the occurrence and count PPMs are within-range metrics, meaning the comparison between observations and predictions is only made within the range where the species occurs.

Prior to calculating PPMS, the test dataset is subsampled spatiotemporally using `ebirdst_subset()`. This process is performed for 25 monte carlo iterations resulting in 25 estimates of each PPM.

Value

An `ebirdst_pppms` object containing a list of three data frames: `binary_ppms`, `occ_ppms`, and `abd_ppms`. These data frames have 25 rows corresponding to 25 Monte Carlo iterations each estimating the PPMs using a spatiotemporal subsample of the test data. Columns correspond to the different PPMS. `binary_ppms` contains binary or range-based PPMS, `occ_ppms` contains within-range occurrence probability PPMs, and `abd_ppms` contains within-range abundance PPMs. In some cases, PPMs may be missing, either because there isn't a large enough test set within the spatiotemporal extent or because average occurrence or abundance is too low. In these cases, try increasing the size of the ebirdst_extent object.

`plot()` can be called on the returned `ebirdst_ppms` object to produce a boxplot of PPMs in all three categories: Binary Occurrence, Occurrence Probability, and Abundance.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# define a spatiotemporal extent to calculate ppms over
bb_vec <- c(xmin = -86, xmax = -83, ymin = 42.5, ymax = 44.5)
e <- ebirdst_extent(bb_vec, t = c("05-01", "05-31"))

# compute predictive performance metrics
ppms <- ebirdst_ppms(path = path, ext = e)
plot(ppms)

## End(Not run)
```

---

ebirdst_ppms_ts   *Time series of eBird Status and Trends PPMs summarized temporally*

Description

Calculate a time series of predictive performance metrics (PPMs) for the eBird Status and Trends model. For each week or month of the year, PPMs will be summarized independently to produce a time series. For further details on eBird Status and Trends PPMs consult the help for ebirdst_ppms.

Usage

```
ebirdst_ppms_ts(path, ext, summarize_by = c("weeks", "months"), ...)

## S3 method for class 'ebirdst_ppms_ts'
plot(x, type = c("binary", "occurrence", "abundance"), metric = "kappa", ...)
```

Arguments

path	character; directory that the Status and Trends data for a given species was downloaded to. This path is returned by ebirdst_download() or get_species_path().
ext	ebirdst_extent object (optional); the spatial extent over which to calculate the PPMs. Note that ebirdst_extent objects typically specify both a spatial and temporal extent, however, **within this function only the spatial component of the extent is used.**
summarize_by	character; periods over which to summarize PPMs. PPMs can either be calculated for eBird Status and Trends weeks (as defined in ebirdst_weeks) or for the months of the year.
...	ignored.
x	ebirdst_ppms_ts object; PPMs summarized by weeks or months as calculated by ebirdst_ppms_ts().
type	character; the PPM type to plot, either a binary, occurrence, or abundance PPM can be plotted.
metric	character; the specific metric to plot, the list list of possible metrics varies by PPM type:

- Binary or occurrence: auc, ppc, kappa, bernoulli_dev, sensitivity, specificity
- Abundance: poisson_dev_abd, poisson_dev_occ, spearman_abd, spearman_occ

Value

An ebirdst_pppms_ts object containing a list of three data frames: binary_ppms, occ_ppms, and abd_ppms. Each row of these data frames corresponds to the PPMs from one Monte Carlo iteration for a given time period. Columns correspond to the different PPMS. binary_ppms contains binary or range-based PPMS, occ_ppms contains within-range occurrence probability PPMs, and abd_ppms contains within-range abundance PPMs. In some cases, PPMs may be missing, either

because there isn't a large enough test set within the spatiotemporal extent or because average occurrence or abundance is too low. In these cases, try increasing the size of the ebirdst_extent object. plot() can be called on the returned ebirdst_pppms_ts object to plot a time series of a single PPM.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# define a spatial extent to calculate ppms over
e <- ebirdst_extent(c(xmin = -86, xmax = -83, ymin = 42.5, ymax = 44.5))

# compute predictive performance metrics, summarized by months
ppms <- ebirdst_ppms_ts(path = path, ext = e, summarize_by = "months")

# plot time series
# binary, kappa
plot(ppms, type = "binary", metric = "kappa")
# occurrence, sensitivity
plot(ppms, type = "occurrence", metric = "sensitivity")
#' # abundance, poisson deviance
plot(ppms, type = "abundance", metric = "poisson_dev_abd")

## End(Not run)
```

---

ebirdst_predictors   *eBird Status and Trends predictors*

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 land and water cover variables. These landcover variables are derived from the MODIS MCD12Q1 500 m landcover product, and for each land cover class two FRAGSTATS metrics are calculated within a 1.5 km buffer around each checklist: % landcover (PLAND) and edge density (ED).

Usage

```
ebirdst_predictors
```

Format

A data frame with 74 rows and 5 columns:

predictor Predictor variable name.

predictor_tidy Predictor variable name, tidied to only contain lowercase letters and underscores.

predictor_label Descriptive labels for predictors for plotting and translating the cryptic variables names (e.g. umd_fs_c1 is Evergreen Needleleaf Forest.

lc_class For the land and water cover FRAGSTATS variables, this gives the associated landcover class. It can be used for grouping and summarizing the four FRAGSTATS metrics to the level of the landcover class.

lc_class_label Similar to predictor_label; however, this variable gives the FRAGSTATS metrics a single name for the landcover class.

---

ebirdst_runs   *Data frame of available eBird Status and Trends species*

Description

A dataset containing the species for which eBird Status and Trends data are available In addition, the dates defining the boundaries of the seasons are provided. These seasons are defined on a species-specific basis through expert review. For information on the details of defining seasons, please see the seasons section of the FAQ. Note that missing dates imply that a season failed expert review for that species within that season.

Usage

```
ebirdst_runs
```

Format

A data frame with 15 variables:

run_name Unique analysis identifier and the top level folder name for all results

species_code Six letter eBird code in eBird Taxonomy v2018

scientific_name Scientific name from eBird Taxonomy v2018

common_name English common name from eBird Taxonomy v2018

resident Classifies this species a resident or a migrant

breeding_quality Breeding season quality

breeding_range_modeled Is the full range modeled?

breeding_start Breeding season start date

breeding_end Breeding season start date

nonbreeding_quality Non-breeding season quality

nonbreeding_range_modeled Is the full range modeled?

nonbreeding_start Non-breeding season start date

nonbreeding_end Non-breeding season start date

postbreeding_migration_quality Post-breeding season quality

postbreeding_migration_range_modeled Is the full range modeled?

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_range_modeled Is the full range modeled?

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

---

ebirdst_subset   *Subset eBird Status and Trends data spatiotemporally*

Description

Spatiotemporally subset the raster or tabular eBird Status and Trends data. The spatiotemporal extent should be defined using `ebirdst_extent()`.

Usage

```
ebirdst_subset(x, ext)

## S3 method for class 'data.frame'
ebirdst_subset(x, ext)

## S3 method for class 'sf'
ebirdst_subset(x, ext)

## S3 method for class 'Raster'
ebirdst_subset(x, ext)
```

Arguments

x	eBird Status and Trends data to subset; either a `RasterStack` object with 52 layers (one for each week) or a data frame with PI or PD data.
ext	ebirdst_extent object; the spatiotemporal extent to filter the data to.

Value

eBird Status and Trends data in the same format as the input data, but subset in space and time.

Methods (by class)

- `data.frame`: PI or PD data
- `sf`: PI or PD data as an `sf` object
- `Raster`: Status and Trends rasters

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data")
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# bbox for southern michigan in may
bb_vec <- c(xmin = -86, xmax = -83, ymin = 41.5, ymax = 43.5)
e <- ebirdst_extent(bb_vec, t = c("05-01", "05-31"))

# load and subset raster data
abd <- load_raster(path, product = "abundance")
abd_ss <- ebirdst_subset(abd, ext = e)

## End(Not run)
```

---

ebirdst_weeks   *eBird Status and Trends weeks*

Description

eBird Status and Trends predictions are made for each of 52 weeks of the year. This data frame provides the boundaries of the weeks.

Usage

```
ebirdst_weeks
```

Format

A data frame with 52 rows and 5 columns:

week_number Integer week number from 1-52.

date Date of the midpoint of the week.

week_midpoint Date of the midpoint of the week expressed as a fraction of the year, i.e. a number from 0-1.

week_start Date of the start of the week expressed as a fraction of the year, i.e. a number from 0-1.

week_end Date of the end of the week expressed as a fraction of the year, i.e. a number from 0-1.

---

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 data package path for a given species*

Description

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

Usage

```
get_species_path(species, path = rappdirs::user_data_dir("ebirdst"))
```

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 can be viewed in the ebirdst_runs data frame included in this package. To download the example dataset, use "example_data".

path   character; directory to download the data to. All downloaded files will be placed in a sub-directory of this directory named according to the unique run ID associated with this species. Defaults to a persistent data directory, which can be found by calling rappdirs::user_data_dir("ebirdst")).

Value

The path to the data package directory.

Examples

```
## Not run:
# download the example data
ebirdst_download("example_data")

# get the path
path <- get_species_path("example_data")

# use it to load data
abd <- load_raster(path, "abundance")

# 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)
```

---

label_raster_stack   *Label data cubes with the week date for each band*

Description

The data cubes are saved as GeoTIFFs, which don't allow for band labels. For convenience, this function labels the layers of a data cube once it has been loaded with the week dates for each band.

Usage

```
label_raster_stack(x)
```

Arguments

x   RasterStack or RasterBrick; original eBird Status and Trends data cube with
                     52 bands, one for each week.

Value

A RasterStack or RasterBrick with names assigned for the dates in the format of "wYYYY.MM.DD"
per raster package constraints. The Raster* objects do not allow the names to start with a number,
nor are they allowed to contain "-", so it is not possible to store the date in an ISO compliant format.
Use parse_raster_dates() to convert the layer names to dates.

Examples

```
## Not run:
# download and load example abundance data
sp_path <- ebirdst_download("example_data")
abd <- load_raster(sp_path, "abundance")

# label
abd <- label_raster_stack(abd)
names(abd)

## End(Not run)
```

---

load_config *Load eBird Status and Trends configuration file*

Description

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

Usage

```
load_config(path)
```

Arguments

path   character; directory that the Status and Trends data for a given species was
                downloaded to. This path is returned by ebirdst_download() or get_species_path().

Value

A list with the run configuration parameters.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data")
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# load configuration file
cfg <- load_config(path)

## 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(path)
```

Arguments

path	character; directory that the Status and Trends data for a given species was downloaded to. This path is returned by `ebirdst_download()` or `get_species_path()`.

Value

A list containing elements:

- `custom_projection`: a custom projection optimized for the given species' full annual cycle
- `fa_extent`: an Extent 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_sinu`: the extent in sinusoidal projection
- `abundance_bins`: abundance bins for the full annual cycle

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# get map parameters
load_fac_map_parameters(path)

## End(Not run)
```

Description

Partial dependence (PD) plots depict the relationship between the modeled occurrence probability and each of the predictor variables used in the model. Status and Trends provides the data to generate these plots for every stixel.

Usage

```
load_pds(path, ext, return_sf = FALSE)
```

Arguments

path	character; directory that the Status and Trends data for a given species was downloaded to. This path is returned by ebirdst_download() or get_species_path().
ext	ebirdst_extent object; the spatiotemporal extent to filter the data to. The spatial component of the extent object must be provided in unprojected, latitude-longitude coordinates.
return_sf	logical; whether to return an sf object of spatial points rather then the default data frame.

Value

Data frame, or sf object if return_sf = TRUE, containing PD estimates for each stixel for either the occurrence and relative model. The data frame will have the following columns:

- stixel_id: unique stixel identifier
- lat and lon: stixel centroid
- date: day of year, expressed as a value from 0-1, of the stixel center
- predictor: name of the predictor that the PD data correspond to, for a full list of predictors consult the ebirdst_predictors data frame
- predictor_value: value of the predictor variable at which PD is evaluated
- response: predicted response, occurrence or relative abundance, at the given value of the predictor averaged across all the values of the other predictors

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# load partial dependence data
```

```
pds <- load_pds(path)

# plot the top 15 predictor importances
# define a spatiotemporal extent to plot data from
bb_vec <- c(xmin = -86.6, xmax = -82.2, ymin = 41.5, ymax = 43.5)
e <- ebirdst_extent(bb_vec, t = c("05-01", "05-31"))
plot_pds(pds, "solar_noon_diff", ext = e, n_bs = 5)

## End(Not run)
```

---

load_pis   *Load eBird Status and Trends predictor importance data*

Description

Loads the predictor importance (PI) data from the pi-pd.db sqlite database. PI estimates are provided for each stixel over which a model was run and are identified by a unique stixel ID in addition to the coordinates of the stixel centroid. PI estimates are for the occurrence model only.

Usage

```
load_pis(path, ext, return_sf = FALSE)
```

Arguments

path	character; directory that the Status and Trends data for a given species was downloaded to. This path is returned by ebirdst_download() or get_species_path().
ext	[ebirdst_extent](#) object; the spatiotemporal extent to filter the data to. The spatial component of the extent object must be provided in unprojected, latitude-longitude coordinates.
return_sf	logical; whether to return an [sf](#) object of spatial points rather then the default data frame.

Value

Data frame, or [sf](#) object if return_sf = TRUE, containing PI estimates for each stixel for both the occurrence and relative abundance models. The data are provided in a 'wide' format, with each row corresponding to the PI estimates for a give stixel for the occurrence count model, and the relative importance of each predictor in columns. Stixels are identified by a unique stixel_id, the centroid of the stixel in space and time is specified by the lat, lon, and date column, which expresses the day of year as a value from 0-1.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
```

```
path <- get_species_path("example_data")

# load predictor importance
pis <- load_pis(path)

# plot the top 15 predictor importances
# define a spatiotemporal extent to plot data from
bb_vec <- c(xmin = -86.6, xmax = -82.2, ymin = 41.5, ymax = 43.5)
e <- ebirdst_extent(bb_vec, t = c("05-01", "05-31"))
plot_pis(pis, ext = e, n_top_pred = 15, by_cover_class = TRUE)

## End(Not run)
```

---

load_predictions   *Load eBird Status and Trends test data predictions*

Description

During eBird Status and Trends modeling, predictions are made for checklists in a test dataset that is not included in the model fitting process. This function loads these predictions in addition to the actual observed count on the associated checklist. These data are used by ebirdst_ppms() to get for calculating predictive performance metrics.

Usage

```
load_predictions(path, return_sf = FALSE)
```

Arguments

path	character; directory that the Status and Trends data for a given species was downloaded to. This path is returned by ebirdst_download() or get_species_path().
return_sf	logical; whether to return an sf object of spatial points rather then the default data frame.

Value

Data frame, or sf object if return_sf = TRUE, containing observed counts and model predictiosn for the test data.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# test data
test_predictions <- load_predictions(path)
```

```
dplyr::glimpse(test_predictions)

## End(Not run)
```

---

load_raster   *Load eBird Status and Trends raster data cubes*

Description

Each of the eBird Status and Trends products is packaged as a GeoTIFF file (referred to as a "cube) with 52 bands, one for each week of the year. This function loads the cube for a given product and species as a RasterStack object.

Usage

```
load_raster(
  path,
  product = c("abundance", "abundance_seasonal", "count", "occurrence",
    "abundance_lower", "abundance_upper", "template"),
  resolution = c("hr", "mr", "lr")
)
```

Arguments

path	character; directory that the Status and Trends data for a given species was downloaded to. This path is returned by ebirdst_download() or get_species_path().
product	character; Status and Trends product to load, see Details for available products. It is also possible to return a template raster with no data.
resolution	character; the resolution of the raster data to load. The default is to load the native ~3 km resolution ("hr"); however, for some applications 9 km ("mr") or 27 km ("lr") data may be suitable.

Details

The available Status and Trends data cubes are as follows:

- occurrence: the expected probability of occurrence of the species, ranging from 0 to 1, on an eBird Traveling Count by a skilled eBirder starting at the optimal time of day with the optimal search duration and distance that maximizes detection of that species in a region.

- count: the expected count of a species, conditional on its occurrence at the given location, on an eBird Traveling Count by a skilled eBirder starting at the optimal time of day with the optimal search duration and distance that maximizes detection of that species in a region.

- abundance: the expected relative abundance, computed as the product of the probability of occurrence and the count conditional on occurrence, of the species on an eBird Traveling Count by a skilled eBirder starting at the optimal time of day with the optimal search duration and distance that maximizes detection of that species in a region.

- abundance_lower: the lower 10th quantile of the expected relative abundance of the species on an eBird Traveling Count by a skilled eBirder starting at the optimal time of day with the optimal search duration and distance that maximizes detection of that species in a region.

- abundance_upper: the upper 90th quantile of the expected relative abundance of the species on an eBird Traveling Count by a skilled eBirder starting at the optimal time of day with the optimal search duration and distance that maximizes detection of that species in a region.

In addition to these cubes with 52 layers (one for each week), it is possible to load:

- abundance_seasonal: the expected relative abundance averaged across the weeks within each season. The date boundaries used for the seasonal definitions appear in ebirdst_runs and if a season failed review no associated layer will be included.

- template: a template raster covering the whole Earth and without any data.

Value

A RasterStack with 52 layers for the given product, labeled by week. Seasonal abundance will have up to four layers labeled according to the seasons. The template raster will be returned as a RasterLayer.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data")
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# load data
load_raster(path, "abundance")

## End(Not run)
```

---

load_stixels   *Load summary data for eBird Status and Trends stixels*

Description

eBird Status and Trends divides space and time into variably sized "stixels" within which individual base models are fit. The process of stixelization is performed many times and the prediction at any given point is the median of the predictions from all the stixels that that point falls in. This function loads summary statistics for each stixel, for example, the size of the stixels and the number of observations within each stixel.

Usage

```
load_stixels(path, ext, return_sf = FALSE)
```

Arguments

`path`	character; directory that the Status and Trends data for a given species was downloaded to. This path is returned by `ebirdst_download()` or `get_species_path()`.
`ext`	[ebirdst_extent](#) object; the spatiotemporal extent to filter the data to. The spatial component of the extent object must be provided in unprojected, latitude-longitude coordinates.
`return_sf`	logical; whether to return an [sf](#) object of spatial points rather then the default data frame.

Value

Data frame, or [sf](#) object if `return_sf = TRUE`, containing stixel summary data. Data are organized with one stixel per row and each stixel identified by a unique `stixel_id`, the centroid of each stixel in space and time is specified by `lat`, `lon`, and `date`.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# load stixel summary information
stixels <- load_stixels(path)
dplyr::glimpse(stixels)

## End(Not run)
```

---

parse_raster_dates   *Parse data cube layer names into dates*

Description

[label_raster_stack()](#) labels the layers of a data cube with the associated week dates in the format of "wYYYY.MM.DD", because of constraints in the `raster` package. This function converts that character vector into an ISO compliant Date vector.

Usage

```
parse_raster_dates(x)
```

Arguments

`x`	Raster object; labeled Status and Trends data cube.

Value

Date vector.

Examples

```
## Not run:
# download and load example abundance data
sp_path <- ebirdst_download("example_data")
abd <- load_raster(sp_path, "abundance")

# parse dates
parse_raster_dates(abd)

## End(Not run)
```

---

plot_pds   *Plot partial dependency (PD) line plots*

Description

For a given eBird Status and Trends species, produce a line plot showing the partial dependence
(PD) relationship for a given predictor. Two options for smoothing are provided.

Usage

```
plot_pds(
  pds,
  predictor,
  ext,
  bootstrap_smooth = TRUE,
  show_stixel_pds = FALSE,
  show_quantiles = FALSE,
  n_bs = 100,
  ss_equivalent = 10,
  k = 25,
  ci_alpha = 0.05,
  gbm_n_trees = 500,
  ylim = NULL,
  plot = TRUE
)
```

Arguments

pds	data frame; partial dependence data from `load_pds()`.
predictor	character; single predictor name to plot PD for. For a full list of predictors, and their associated definitions, see ebirdst_predictors.

ext	[ebirdst_extent](#) object; the spatiotemporal extent over which to calculate PDs. This is required, since results are less meaningful over large spatiotemporal extents.

bootstrap_smooth

   logical; the ideal visualization of the PD data is a pointwise GAM smoothing of the individual stixel PD values. This argument specifies whether this should be done directly on the full PD dataset (`bootstrap_smooth = FALSE`) or by subsampling and bootstrapping. The latter approach deals with the randomness in the data and can be more efficient for large datasets.

show_stixel_pds

   logical; whether to plot the individual stixel PD values as semi-transparent lines.

show_quantiles	logical; adds a band for the upper (90th) and lower (10th) quantiles of the individual stixel PD values. These are calculated using quantile regression.
n_bs	int; number of GAM bootstrap iterations when estimating PD confidence intervals. Ignored if `bootstrap_smooth = FALSE`.
ss_equivalent	int; when bootstrapping to estimate PD confidence intervals, this argument specifies the size of the subsample of the original data. In particular, `ss_equivalent` should be an integer representing the equivalent sampling size when averaging this number of PD estimates.
k	integer; number of knots to use in the GAM when smooth the PD relationship.
ci_alpha	numeric; alpha level of confidence intervals. Default is 0.05.
gbm_n_trees	integer; number of trees to fit in the GBM when estimating quantiles. Ignored if `show_quantiles = FALSE`. Default is 500.
ylim	numeric; 2-element vector to pre-define the y-limits of plotting. In the format `c(ymin,ymax)`.
plot	logical; whether to plot the PD relationships or just return data.

Value

Plots the smoothed partial dependence relationship for the specified predictor and returns a data frame of the smoothed curve with confidence intervals.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# load predictor dependence data
pds <- load_pds(path)

# define a spatiotemporal extent to plot data from
bb_vec <- c(xmin = -86, xmax = -83, ymin = 41.5, ymax = 43.5)
e <- ebirdst_extent(bb_vec, t = c("05-01", "05-31"))
```

```
# for testing, run with 5 bootstrap iterations for speed
# in practice, best to run with the default number of iterations (100)
pd_smooth <- plot_pds(pds, "solar_noon_diff", ext = e, n_bs = 5)
dplyr::glimpse(pd_smooth)

## End(Not run)
```

---

plot_pis   *Plot predictor importance (PI) box plots*

Description

For a given eBird Status and Trends species, produce a box plot showing the predictor importance (PI) for each of the predictors used in the occurrence model. Predictors are plotted in order from highest to lowest importance. Many function parameters allow for customized plots.

Usage

```
plot_pis(
  pis,
  ext,
  by_cover_class = TRUE,
  n_top_pred = 20,
  pretty_names = TRUE,
  plot = TRUE
)
```

Arguments

pis	data frame; predictor importance data from [load_pis()](#).
ext	[ebirdst_extent](#) object; the spatiotemporal extent over which to calculate PIs. This is required, since results are less meaningful over large spatiotemporal extents.
by_cover_class	logical; whether to aggregate the FRAGSTATS metrics (PLAND and ED) for the land cover classes into single values for the land cover classes.
n_top_pred	integer; how many predictors to show.
pretty_names	logical; whether to convert cryptic land cover codes to readable land cover class names.
plot	logical; whether to plot predictor importance or just return top predictors.

Value

Plots a boxplot of predictor importance and invisibly returns a named vector of top predictors, and their median predictor importance, based on the n_top_pred parameter.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# load predictor importance
pis <- load_pis(path)

# define a spatiotemporal extent to plot data from
bb_vec <- c(xmin = -86, xmax = -83, ymin = 41.5, ymax = 43.5)
e <- ebirdst_extent(bb_vec, t = c("05-01", "05-31"))

top_pred <- plot_pis(pis, ext = e, by_cover_class = TRUE, n_top_pred = 10)
top_pred

## End(Not run)
```

---

poisson_dev *Poisson deviance*

Description

Poisson deviance

Usage

```
poisson_dev(obs, pred)
```

Arguments

obs	numeric; observed values.
pred	numeric; predicted values.

Value

A named numeric vector with three elements: model deviance, mean deviance, and deviance explained.

Examples

```
obs <- c(0, 0, 1, 3, 5, 2)
pred <- c(0.5, 0.1, 2.5, 3.3, 5.2, 2.5)
ebirdst:::poisson_dev(obs, pred)
```

---

project_extent   *Transform a spatiotemporal extent to a different CRS*

Description

Transform an eBird Status and Trends extent object to a different coordinate reference system. This is most commonly required to transform the extent to the sinusoidal CRS used by the eBird Status and Trends rasters.

Usage

```
project_extent(x, crs)
```

Arguments

x   [ebirdst_extent](#) object; a spatiotemporal extent.

crs   coordinate references system, given either as a proj4string, an integer EPSG code, or a `crs` object generated with [`st_crs()`](#).

Value

An [ebirdst_extent](#) object in the new CRS.

Examples

```
# construct an ebirdst_extent object
bb_vec <- c(xmin = -80, xmax = -70, ymin = 40, ymax = 47)
bb <- sf::st_bbox(bb_vec, crs = 4326)
bb_ext <- ebirdst_extent(bb)

# transform to sinusoidal projection of rasters
sinu <- "+proj=sinu +lon_0=0 +x_0=0 +y_0=0 +a=6371007 +b=6371007 +units=m +no_defs"
project_extent(bb_ext, crs = sinu)

# also works on polygon extents
poly <- sf::read_sf(system.file("shape/nc.shp", package="sf"))
poly_ext <- ebirdst_extent(poly)
project_extent(poly_ext, crs = sinu)
```

sample_grid

Spatiotemporal sampling of points on a grid

Description

Subsample points to deal with spatiotemporal bias in observations by defining a grid in space and time, then sampling the given number of points from each cell. sample_case_control() additionally samples presence and absence independently.

Usage

```
sample_grid(x, res, t_res, n = 1, replace = FALSE, jitter = TRUE)

sample_case_control(x, res, t_res, n = 1, replace = FALSE, jitter = TRUE)
```

Arguments

x	data frame or sf object; the points to subsample. If x is a data frame the coordiantes should be provided as columns lat and lon. The day of year should be expressed as a proportion from 0-1 and stored in the column date.
res	numeric; the size in meters of the grid to sample from. This can be a 2 element vector indicating the x and y dimensions of the cells.
t_res	numeric; the temporal resolution for sampling expressed as a proportion of the year. For example, 7 / 365 would result in sampling from each week.
n	integer; the number of points to sample from each grid cell.
replace	logical; whether to sample with replacement.
jitter	logical; to avoid always using the same grid for sampling, the grid can be jittered so that the origin is different each time this function is called.

Value

Logical vector indicating which rows are selected.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# test data to sample
preds <- load_predictions(path, return_sf = TRUE)

# sample on a 100km, 1 month grid
s <- sample_grid(preds, res = 100000, t_res = 1 / 12)
preds_grid <- preds[s, ]
```

```
# case control sampling independently samples presence and absence
s <- sample_case_control(preds, res = 100000, t_res = 1 / 12)
preds_cc <- preds[s, ]

# grid sampling preserves the presence/absence ratio
table(preds$obs > 0) / nrow(preds)
table(preds_grid$obs > 0) / nrow(preds_grid)
# while case control sampling increases the prevelance of presences
table(preds_cc$obs > 0) / nrow(preds_cc)

# plot
library(sf)
p <- par(mar = c(0, 0, 0, 0))
plot(st_geometry(preds), col = "black", pch = 19, cex = 0.2)
plot(st_geometry(preds_cc), col = "red", pch = 19, cex = 0.5, add = TRUE)

## 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()` 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)
```

---

stixelize   *Generate polygons for eBird Status and Trends stixels*

Description

eBird Status and Trends divides space and time into variably sized "stixels" within which indi-
vidual base models are fit. The process of stixelization is performed many times and the predic-
tion at any given point is the median of the predictions from all the stixels that that point falls in.
load_stixels() loads information on all the stixels that compromise a species' Status and Trends
model, with stixels identified by the location of their centroid. This function uses this information
to define polygons for each stixel and attaches them to the original data in the form of an sf object.

Usage

```
stixelize(x)
```

Arguments

x   data.frame or sf object; stixel summary data loaded with load_stixels(), or
                     any other data frame with fields lon, lat, stixel_width, and stixel_hight.

Value

sf object with geometry column storing polygons representing the stixels boundaries.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data", tifs_only = FALSE)
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")

# load stixel summary information
stixels <- load_stixels(path)

# build stixel polygons
stixelize(stixels)

## End(Not run)
```

stixel_footprint | *Calculate the spatial footprint of a set of stixels*

Description

eBird Status and Trends divides space and time into variably sized "stixels" within which individual base models are fit. The process of stixelization is performed many times and the prediction at any given point is the median of the predictions from all the stixels that that point falls in. For a given spatiotemporal extent, this function identifies the set of stixels whose centroids fall within that extent and calculates the spatial footprint of these stixels, i.e. a surface indicating the proportion of the selected stixels that contribute information to model estimates at each location. This footprint gives an estimate of where the information for the model predictions, predictor importances (PIs), and partial dependencies (PDs) come from.

Usage

```
stixel_footprint(path, ext)

## S3 method for class 'stixel_footprint'
plot(x, ...)
```

Arguments

path	character; full path to directory containing the eBird Status and Trends products for a single species.
ext	ebirdst_extent object; the spatiotemporal extent to filter the data to.
x	stixel_footprint object to map.
...	ignored.

Value

A stixel_footprint object consisting of a list with three elements:

- footprint: a RasterStack giving the percentage of the selected stixels that are contributing to each grid cell.
- centroids: an sf object containing the stixel centroids points.
- extent: an ebirdst_extent object specifying the chosen spatiotemporal extent.

The stixel footprint can be mapped by calling plot() on the returned stixel_footprint object.

Examples

```
## Not run:
# download example data
path <- ebirdst_download("example_data")
# or get the path if you already have the data downloaded
path <- get_species_path("example_data")
```

```
# define a spatiotemporal extent
bb_vec <- c(xmin = -86, xmax = -83, ymin = 42, ymax = 45)
e <- ebirdst_extent(bb_vec, t = c("05-01", "05-31"))

# calculate effective extent map
footprint <- stixel_footprint(path, ext = e)
plot(footprint)

## End(Not run)
```

Index