

Package ‘ebirdst’

April 4, 2019

Type Package

Title Access and Analyze eBird Status and Trends Data

Version 0.1.0

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

License GPL-3

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

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

Depends R (>= 3.3.0)

Imports car, data.table, dplyr (>= 0.7.0), fasterize, gbm (>= 2.1.5), ggplot2, grDevices, gridExtra, methods, mgcv, PresenceAbsence, rappdirs, raster, scales, sf, stats, stringr, tidyr, tools, utils, viridisLite, xml2, rgdal

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

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

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

Maintainer Tom Auer <mta45@cornell.edu>

Repository CRAN

Date/Publication 2019-04-04 09:50:03 UTC

R topics documented:

abundance_palette	2
bernoulli_dev	3
calc_bins	4
calc_effective_extent	5
calc_full_extent	6
compute_ppms	6
ebirdst	7
ebirdst_download	8
ebirdst_extent	9
ebirdst_predictors	11
ebirdst_runs	12
ebirdst_subset	13
label_raster_stack	14
load_pds	15
load_pis	16
load_raster	17
load_test_data	17
map_centroids	18
parse_raster_dates	19
plot_all_ppms	20
plot_binary_by_time	20
plot_pds	21
plot_pis	23
poisson_dev	24
project_extent	25
sample_grid	26
stixelize	27
Index	29

abundance_palette	<i>eBird Status and Trends color palettes for abundance data</i>
-------------------	--

Description

Generate the color palettes used in 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 color hex codes.

Examples

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

bernoulli_dev	<i>Bernoulli deviance</i>
---------------	---------------------------

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)
```

calc_bins	<i>Calculates bins (breaks) based on standard deviations of Box-Cox power-transformed data for mapping</i>
-----------	--

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, we selected a method (described by Maciejewski et al. 2013) that first selects an optimal power (the Box-Cox method) for normalizing the data, then power transforms the entire year of non-zero data, constructs bins with the power-transformed data using standard-deviations, and then un-transforms the bins.

Usage

```
calc_bins(x)
```

Arguments

x RasterStack or RasterBrick; original eBird Status and Trends product raster GeoTIFF with 52 bands, one for each week.

Value

A list with two elements: bins is a vector containing the break points of the bins and power is the optimal power used to transform data when calculating bins.

References

Ross Maciejewski, Avin Pattah, Sungahn Ko, Ryan Hafen, William S. Cleveland, David S. Ebert. Automated Box-Cox Transformations for Improved Visual Encoding. IEEE Transactions on Visualization and Computer Graphics, 19(1): 130-140, 2013.

Examples

```
# download and load example abundance data
sp_path <- ebirdst_download("example_data")
abd <- load_raster("abundance_umean", sp_path)

# calculate bins for a single week for this example
year_bins <- calc_bins(abd)
```

calc_effective_extent *Calculate and map effective extent of selected centroids*

Description

The selection of stixel centroids for analysis of PIs and/or PDs yields an effective footprint, or extent, showing the effective location of where the information going into the analysis with PIs and/or PDs is based. While a bounding box or polygon may be used to select a set of centroids, due to the models being fit within large rectangular areas, the information from a set of centroids often comes from the core of the selected area. This function calculates where the highest proportion of information is coming from, returns a raster and plots that raster, with the selected area and centroids for reference. The legend shows, for each pixel, what percentage of the selected stixels are contributing information, ranging from 0 to 1.

Usage

```
calc_effective_extent(path, ext, pi_pd = c("pi", "pd"), plot = TRUE)
```

Arguments

path	character; full path to directory containing the eBird Status and Trends products for a single species.
ext	ebirdst_extent object (optional); the spatiotemporal extent to filter the data to.
pi_pd	character; whether to use predictor importance ("pi") or partial dependence ("pd") for stixel centroids.
plot	logical; whether to plot the results or just return a raster without plotting.

Value

A raster showing the percentage of the selected stixels that are contributing to each grid cell. In addition, if `plot = TRUE` this raster will be plotted along with centroid locations and [ebirdst_extent](#) boundaries.

Examples

```
# download and load example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)

# define a spatiotemporal extent
bb_vec <- c(xmin = -86, xmax = -84, ymin = 41.5, ymax = 43.5)
e <- ebirdst_extent(bb_vec, t = c("05-01", "05-31"))

# calculate effective extent map
eff <- calc_effective_extent(path = sp_path, ext = e, pi_pd = "pi")
```

calc_full_extent	<i>Calculates spatial extent of non-zero data from Raster* object for plotting</i>
------------------	--

Description

After loading a RasterStack of results, there are lots of NA values and plots of individual raster layers will display at the full extent of the study extent. To show an ideal extent, this function trims away 0 and NA values and checks to make sure it returns a reasonable extent for plotting. The returned Extent object can then be used for plotting.

Usage

```
calc_full_extent(x)
```

Arguments

x Raster* object; either full RasterStack or subset.

Value

raster Extent object

Examples

```
# download and load example abundance data
sp_path <- ebirdst_download("example_data")
abd <- load_raster("abundance_umean", sp_path)

# calculate full extent
plot_extent <- calc_full_extent(abd)

# plot
raster::plot(abd[[1]], axes = FALSE, ext = plot_extent)
```

compute_ppms	<i>Computes the Predictive Performance Metrics for a spatiotemporal extent</i>
--------------	--

Description

Loads test data and ensemble support values and then calculates the predictive performance metrics (PPMs) within a spatiotemporal extent defined by an [ebirdst_extent](#) object. Use this function directly to access the computed metrics, or use `plot_all_ppms()` or `plot_binary_by_time()` to summarize the metrics.

Usage

```
compute_ppms(path, ext)
```

Arguments

path character; full path to directory containing the eBird Status and Trends products for a single species.

ext [ebirdst_extent](#) object (optional); the spatiotemporal extent to filter the data to.

Value

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 occupancy 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.

Examples

```
# download and load example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)

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

# compute predictive performance metrics
ppms <- compute_ppms(path = sp_path, ext = e)
```

ebirdst	<i>ebirdst: Tools to Load, Map, Plot, and Analyze eBird Status and Trends Data Products</i>
---------	---

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. 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.

Usage

```
ebirdst_download(species, path = rappdirs::user_data_dir("ebirdst"),
  tifs_only = TRUE, force = FALSE)
```

```
download_data(species, path = rappdirs::user_data_dir("ebirdst"),
  tifs_only = TRUE, force = FALSE)
```

Arguments

species	character; a single species given as a scientific name, common name or six-letter species code (e.g. <code>woothr</code>). 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 <code>"example_data"</code> .
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 <code>rappdirs::user_data_dir("ebirdst")</code> .
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.

Value

Path to the run-specific root of the downloaded files.

Functions

- `download_data`: deprecated, use `ebirdst_download()`

Examples

```
# download the example data
ebirdst_download("example_data")

## Not run:
# download the full data package for wood thrush
ebirdst_download("woothr")

## End(Not run)
```

ebirdst_extent	<i>Construct a spatiotemporal extent object to subset Status and Trends data</i>
----------------	--

Description

ebirdst_extent objects 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 <code>sf::st_bbox()</code> object) or a polygon (an <code>sf::sf()</code> 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. <code>c("2016-12-01", "2016-01-31")</code> 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 `sf::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 `sf::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 `sf::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 (e.g. "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("2016-12-01", "2016-01-31")` 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 `sf::st_bbox()`
- `numeric`: bounding box given as edges
- `sfc`: polygons as `sf` spatial feature column
- `sf`: polygons as `sf` object

Examples

```
# bounding box of NE United States 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("2016-01-01", "2016-01-31"))

# dates can wrap around, e.g. to use dec-jan
ebirdst_extent(bb, t = c("2016-12-01", "2016-01-31"))

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

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 travelled, 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 four FRAGSTATS metrics are calculated within a 1.5 km buffer around each checklist: patch index (LPI), and patch density (PD).

Usage

```
ebirdst_predictors
```

Format

A data frame with 87 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 variable 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 four FRAGSTATS metrics a single name for the landcover class.

ebirdst_runs	<i>Data frame of available eBird Status and Trends species</i>
--------------	--

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 107 rows and 14 variables:

species_code Six letter eBird code in eBird Taxonomy v2016

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

scientific_name Scientific name from eBird Taxonomy v2016

common_name English common name from eBird Taxonomy v2016

breeding_start_dt Breeding season start date

breeding_end_dt Breeding season start date

nonbreeding_start_dt Non-breeding season start date

nonbreeding_end_dt Non-breeding season start date

postbreeding_migration_start_dt Post-breeding season start date

postbreeding_migration_end_dt Post-breeding season start date

prebreeding_migration_start_dt Pre-breeding season start date

prebreeding_migration_end_dt Pre-breeding season start date

year_round_start_dt For resident species, the year-round start date

year_round_end_dt For resident species, the year-round end date

ebirdst_subset	<i>Subset eBird Status and Trends data spatiotemporally</i>
----------------	---

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 Raster 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.

Methods (by class)

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

Examples

```
# 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"))

# download and load example data
sp_path <- ebirdst_download("example_data")
pis <- load_pis(sp_path)
abd <- load_raster(product = "abundance_umean", sp_path)
```

```
# subset
abd_ss <- ebirdst_subset(abd, ext = e)
pis_ss <- ebirdst_subset(pis, ext = e)
```

label_raster_stack *Labels 52 week RasterStack with the dates for each band*

Description

The raster package does not allow layer names to be saved with the bands of a multi-band GeoTIFF. Accordingly, all eBird Status and Trends products raster results cover the entire 52 week temporal extent of analysis. For convenience, this function labels the RasterStack once it has been loaded with the dates for each band.

Usage

```
label_raster_stack(x)
```

Arguments

x RasterStack or RasterBrick; original eBird Status and Trends product raster GeoTIFF with 52 bands, one for each week.

Value

A RasterStack or RasterBrick with names assigned for the dates in the format of "YYYYY.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.

Examples

```
# download and load example abundance data
sp_path <- ebirdst_download("example_data")
abd <- load_raster("abundance_umean", sp_path)

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

load_pds	<i>Load partial dependencies for single species eBird Status and Trends products</i>
----------	--

Description

Loads the partial dependency data (from pd.txt), joins with stixel summary data, sets the names, and cleans up the data.frame. This is one of the slower functions in the package, due to the size of the pd.txt file (usually multiple GB).

Usage

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

Arguments

path	character; full path to the directory containing single species eBird Status and Trends products.
return_sf	logical; whether to return an sf object of spatial points rather than the default data frame.

Value

data.frame containing partial dependency values for each stixel, as well as stixel summary information. To make these data more compact, they're stored in a wide format. Each row corresponds to the partial dependence relationship of one predictor for a single stixel. There are 50 columns (x1-x50) at which the partial dependence is measured and 50 columns (y1-y50) that give the resulting probability of occurrence on the logit scale.

Examples

```
# download example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)

# load partial dependence
pds <- load_pds(sp_path)

# plot partial dependence for effort hours
# 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, "effort_hrs", ext = e)
```

load_pis	<i>Load predictor importances for single species eBird Status and Trends products</i>
----------	---

Description

Loads the predictor importance data (from pi.txt), joins with stixel summary data, sets column names, and cleans up the data.frame.

Usage

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

Arguments

path	character; full path to the directory containing single species eBird Status and Trends products.
return_sf	logical; whether to return an sf object of spatial points rather than the default data frame.

Value

data.frame containing predictor importance values for each stixel, as well as stixel summary information.

Examples

```
# download example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)

# load predictor importance
pis <- load_pis(sp_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)
```

load_raster	<i>Load eBird Status and Trends raster data</i>
-------------	---

Description

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

Usage

```
load_raster(product = c("abundance_umean", "occurrence_umean",
  "abundance_lower", "abundance_upper", "template"), path)
```

Arguments

product	character; status and trends product to load, options are relative abundance, occurrence, and upper and lower bounds on relative abundance. It is also possible to return a template raster with no data.
path	character; full path to the directory containing single species eBird Status and Trends products.

Value

A RasterStack of data for the given product.

Examples

```
# download example data
sp_path <- ebirdst_download("example_data")

# load data
load_raster("abundance_umean", sp_path)
```

load_test_data	<i>Test data loader</i>
----------------	-------------------------

Description

Internal function used by `compute_ppms()` to get the test data for calculating predictive performance metrics. This file contains the observed counts and model predicted relative occurrence and abundance values.

Usage

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

Arguments

path	character; full path to the directory containing single species eBird Status and Trends products.
return_sf	logical; whether to return an sf object of spatial points rather than the default data frame.

Value

data.frame containing test data, including checklist locations, observed counts, and predicted mean and associated error for relative occurrence and abundance.

Examples

```
# download example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)

# test data
test_data <- load_test_data(sp_path)
dplyr::glimpse(test_data)
```

map_centroids

Map PI and PD centroid locations

Description

Creates a map showing the stixel centroid locations for PIs and/or PDs, with an optional spatiotemporal subset using an [ebirdst_extent](#) object

Usage

```
map_centroids(path, ext, plot_pis = TRUE, plot_pds = TRUE)
```

Arguments

path	character; full path to directory containing the eBird Status and Trends products for a single species.
ext	ebirdst_extent object (optional); the spatiotemporal extent to filter the data to.
plot_pis	logical; whether to show PI stixel centroid locations.
plot_pds	logical; whether to show PD stixel centroid locations.

Value

Plot showing locations of PIs and/or PDs.

Examples

```
# download and load example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)

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

map_centroids(path = sp_path, ext = e)
```

parse_raster_dates *Parses the names attached to a Raster from label_raster_stack()*

Description

The `label_raster_stack()` function labels the dates of the estimate rasters in the format of "YYYYY.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; full or subset of original eBird Status and Trends product raster GeoTIFF.

Value

Date vector.

Examples

```
# download and load example abundance data
sp_path <- ebirdst_download("example_data")
abd <- load_raster("abundance_umean", sp_path)

# label
abd <- label_raster_stack(abd)

# parse dates
parse_raster_dates(abd)
```

plot_all_ppms *Plot all predictive performance metrics*

Description

For a spatiotemporal extent, plots bar plots for all available predictive performance metrics within three categories: Binary Occurrence, Occurrence Probability, and Abundance.

Usage

```
plot_all_ppms(path, ext)
```

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.

Value

Plot of metric box plots by category

Examples

```
# download example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)

# define a spatiotemporal extent to plot data from
bb_vec <- c(xmin = -83, xmax = -82, ymin = 41, ymax = 48)
e <- ebirdst_extent(bb_vec, t = c("04-01", "06-30"))

# plot ppms within extent
plot_all_ppms(path = sp_path, ext = e)
```

plot_binary_by_time *Plot binary occurrence metrics by time*

Description

For a specified number of time periods (ideally weeks or months), plots one of four (Kappa, AUC, Sensitivity, Specificity) box plots. Provide an `ebirdst_extent` object to see performance within a spatiotemporal extent, otherwise rangewide performance will be shown.

Usage

```
plot_binary_by_time(path, metric = c("kappa", "auc", "sensitivity",
  "specificity"), ext, n_time_periods = 52)
```

Arguments

path character; full path to directory containing the eBird Status and Trends products for a single species.

metric character; the PPM to plot, either "kappa", "auc", "sensitivity", or "specificity".

ext [ebirdst_extent](#) object (optional); the spatiotemporal extent to filter the data to. The temporal component will be ignored since `n_time_periods` defines the temporal periods over which to calculate the PPMs.

n_time_periods integer; number of periods to divide the year into to calculate the PPMs, e.g. use 52 to divide into weeks and 12 to divide into months.

Value

Boxplot of PPM over time.

Examples

```
# download and load example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)

# plot monthly kappa
plot_binary_by_time(path = sp_path, metric = "kappa", n_time_periods = 12)
```

plot_pds

Plot partial dependency line plots

Description

For all of the available predictors in a single species set of eBird Status and Trends products, this function makes a line plot of a single partial dependency (PD), with two options for smoothing.

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 to filter the data to. 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. Defaults to TRUE.
show_stixel_pds	logical; whether to plot the individual stixel PD values as semi-transparent lines. Defaults to FALSE.
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. Defaults to FALSE.
n_bs	int; number of GAM bootstrap iterations when estimating PD confidence intervals. Ignored if bootstrap_smooth = FALSE. Default is 100.
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. Defaults to 10.
k	integer; number of knots to use in the GAM when smooth the PD relationship. Default is 25.
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 and returns a data frame of the smoothed curve with confidence intervals.

Examples

```
# download and load example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)
pds <- load_pds(sp_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 this example, run with 5 bootstrap iterations for speed
pd_smooth <- plot_pds(pds, "time", ext = e, n_bs = 5)
dplyr::glimpse(pd_smooth)

## Not run:

# in practice, it's best to run with the default number of iterations (100)
plot_pds(pds, predictor = "time", ext = e)

## End(Not run)

```

plot_pis

Plot predictor importances boxplot

Description

For all of the available predictors in a single set of species eBird Status and Trends products, this function makes a bar plot of those relative importances, from highest to lowest. Many function parameters allow for customized plots.

Usage

```

plot_pis(pis, ext, by_cover_class = FALSE, n_top_pred = 50,
         pretty_names = TRUE, plot = TRUE)

```

Arguments

pis	data.frame; predictor importance data from load_pis() .
ext	ebirdst_extent object; the spatiotemporal extent to filter the data to. Required, since results are less meaningful over large spatiotemporal extents.
by_cover_class	logical; whether to aggregate the four FRAGSTATS metrics 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

```
# download and load example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)
pis <- load_pis(sp_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
```

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	<i>Transform a spatiotemporal extent to a different CRS</i>
----------------	---

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 sf::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=6371007.181 +b=6371007.181 +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

*Spatial grid sampling methods***Description**

Methods for subsampling data to deal with spatiotemporal bias in observations. These functions define a grid in space and time, then sample 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 <code>sf</code> point object; the points to subsample
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, $7 / 375$ 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

```
# download example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)

# test data to sample
test_data <- load_test_data(sp_path, return_sf = TRUE)

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

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

```
# grid sampling preserves the presence/absence ratio
table(test_data$obs > 0) / nrow(test_data)
table(td_grid$obs > 0) / nrow(td_grid)
# while case control sampling increases the prevalence of presences
table(td_cc$obs > 0) / nrow(td_cc)

## Not run:

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

## End(Not run)
```

stixelize

Generate stixel polygons from PI/PD data

Description

All predictor importance and partial dependence data are provided at the stixel level. In these files, the stixel is defined based on a centroid, width, and height. 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)

## S3 method for class 'data.frame'
stixelize(x)

## S3 method for class 'sf'
stixelize(x)
```

Arguments

`x` `data.frame` or `sf` object; PI or PD data loaded with `load_pis()` or `load_pds()`, or any other data frame with fields `lon`, `lat`, `stixel_width`, and `stixel_high`.

Value

`sf` object with geometry column storing polygons representing the stixel boundaries.

Methods (by class)

- data.frame: PI or PD data
- sf: PI or PD data as sf object

Examples

```
# download example data
sp_path <- ebirdst_download("example_data", tifs_only = FALSE)

# load predictor importance
pis <- load_pis(sp_path)

stixelize(pis)

# also works on sf objects
pis_sf <- sf::st_as_sf(pis, coords = c("lon", "lat"), crs = 4326)
stixelize(pis_sf)
```

Index

*Topic **datasets**

- ebirdst_predictors, 11
- ebirdst_runs, 12

- abundance_palette, 2

- bernoulli_dev, 3

- calc_bins, 4
- calc_effective_extent, 5
- calc_full_extent, 6
- compute_ppms, 6
- compute_ppms(), 17

- download_data (ebirdst_download), 8

- ebirdst, 7
- ebirdst-package (ebirdst), 7
- ebirdst_download, 8
- ebirdst_extent, 5–7, 9, 13, 18, 20–23, 25
- ebirdst_extent(), 13
- ebirdst_predictors, 11, 22
- ebirdst_runs, 8, 12
- ebirdst_subset, 13

- label_raster_stack, 14
- label_raster_stack(), 19
- load_pds, 15
- load_pds(), 27
- load_pis, 16
- load_pis(), 23, 27
- load_raster, 17
- load_test_data, 17

- map_centroids, 18

- parse_raster_dates, 19
- plot_all_ppms, 20
- plot_binary_by_time, 20
- plot_pds, 21
- plot_pis, 23

- poisson_dev, 24
- project_extent, 25

- sample_case_control (sample_grid), 26
- sample_case_control(), 26
- sample_grid, 26
- sf, 15, 16, 18, 26, 27
- sf::sf(), 9
- sf::st_bbox(), 9, 10
- sf::st_crs(), 10, 25
- stixelize, 27