Package 'auk'

February 4, 2019

Title eBird Data Extraction and Processing in R

Version 0.3.2

Description Extract and process bird sightings records from eBird (<http://ebird.org>), an online tool for recording bird observations. Public access to the full eBird database is via the eBird Basic Dataset (EBD; see <http://ebird.org/ebird/data/download> for access), a downloadable text file. This package is an interface to AWK for extracting data from the EBD based on taxonomic, spatial, or temporal filters, to produce a manageable file size that can be imported into R.

License GPL-3

URL https://github.com/CornellLabofOrnithology/auk, http://CornellLabofOrnithology.github.io/auk/

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

Depends R (>= 3.1.2)

Imports assertthat, countrycode (>= 1.0.0), dplyr (>= 0.7.8), httr, magrittr, rlang (>= 0.3.0), stringi, stringr, tidyr (>= 0.8.0), utils

Suggests covr, data.table, knitr, readr, rmarkdown, testthat, unmarked

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Matthew Strimas-Mackey [aut, cre] (<https://orcid.org/0000-0001-8929-7776>), Eliot Miller [aut], Wesley Hochachka [aut], Cornell Lab of Ornithology [cph]

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

Repository CRAN

Date/Publication 2019-02-04 15:44:49 UTC

R topics documented:

auk *auk: eBird Data Extraction and Processing in R*

Description

Tools for extracting and processing eBird data from the eBird Basic Dataset (EBD).

auk_bbox *Filter the eBird data by spatial bounding box*

Description

Define a filter for the eBird Basic Dataset (EBD) based on spatial bounding box. This function only defines the filter and, once all filters have been defined, `auk_filter()` should be used to call AWK and perform the filtering.

Usage

```
auk_bbox(x, bbox)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by `auk_ebd()` or `auk_sampling()`.
bbox	numeric; spatial bounding box expressed as the range of latitudes and longitudes in decimal degrees: `c(lng_min, lat_min, lng_max, lat_max)`. Note that longitudes in the Western Hemisphere and latitudes sound of the equator should be given as negative numbers.

Details

This function can also work with on an auk_sampling object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: `auk_bcr`, `auk_breeding`, `auk_complete`, `auk_country`, `auk_date`, `auk_distance`, `auk_duration`, `auk_extent`, `auk_filter`, `auk_last_edited`, `auk_project`, `auk_protocol`, `auk_species`, `auk_state`, `auk_time`

Examples

```
# fliter to locations roughly in the Pacific Northwest
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_bbox(bbox = c(-125, 37, -120, 52))

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_bbox(ebd, bbox = c(-125, 37, -120, 52))
```

---

auk_bcr *Filter the eBird data by Bird Conservation Region*

Description

Define a filter for the eBird Basic Dataset (EBD) to extract data for a set of Bird Conservation Regions (BCRs). BCRs are ecologically distinct regions in North America with similar bird communities, habitats, and resource management issues. This function only defines the filter and, once all filters have been defined, `auk_filter()` should be used to call AWK and perform the filtering.

Usage

```
auk_bcr(x, bcr, replace = FALSE)
```

Arguments

`x`	auk_ebd or auk_sampling object; reference to file created by `auk_ebd()` or `auk_sampling()`.
`bcr`	integer; BCRs to filter by. BCRs are identified by an integer, from 1 to 66, that can be looked up in the bcr_codes table.
`replace`	logical; multiple calls to `auk_state()` are additive, unless `replace = FALSE`, in which case the previous list of states to filter by will be removed and replaced by that in the current call.

Details

This function can also work with on an `auk_sampling` object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: `auk_bbox`, `auk_breeding`, `auk_complete`, `auk_country`, `auk_date`, `auk_distance`, `auk_duration`, `auk_extent`, `auk_filter`, `auk_last_edited`, `auk_project`, `auk_protocol`, `auk_species`, `auk_state`, `auk_time`

Examples

```
# bcr codes can be looked up in bcr_codes
dplyr::filter(bcr_codes, bcr_name == "Central Hardwoods")
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_bcr(bcr = 24)

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_bcr(ebd, bcr = 24)
```

---

auk_breeding *Filter to only include observations with breeding codes*

Description

eBird users have the option of specifying breeding bird atlas codes for their observations, for example, if nesting building behaviour is observed. Use this filter to select only those observations with an associated breeding code. This function only defines the filter and, once all filters have been defined, auk_filter() should be used to call AWK and perform the filtering.

Usage

```
auk_breeding(x)
```

Arguments

x	auk_ebd object; reference to basic dataset file created by auk_ebd().

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_complete, auk_country, auk_date, auk_distance, auk_duration, auk_extent, auk_filter, auk_last_edited, auk_project, auk_protocol, auk_species, auk_state, auk_time

Examples

```
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_breeding()
```

auk_clean *Clean an eBird data file (Deprecated)*

Description

This function is no longer required by current versions of the eBird Basic Dataset (EBD).

Usage

```
auk_clean(f_in, f_out, sep = "\t", remove_text = FALSE,
  overwrite = FALSE)
```

Arguments

f_in	character; input file. If file is not found as specified, it will be looked for in the directory specified by the EBD_PATH environment variable.
f_out	character; output file.
sep	character; the input field separator, the basic dataset is tab separated by default. Must only be a single character and space delimited is not allowed since spaces appear in many of the fields.
remove_text	logical; whether all free text entry columns should be removed. These columns include comments, location names, and observer names. These columns cause import errors due to special characters and increase the file size, yet are rarely valuable for analytical applications, so may be removed. Setting this argument to TRUE can lead to a significant reduction in file size.
overwrite	logical; overwrite output file if it already exists.

Value

If AWK ran without errors, the output filename is returned, however, if an error was encountered the exit code is returned.

See Also

Other text: auk_select, auk_split

Examples

```
## Not run:
# get the path to the example data included in the package
f <- system.file("extdata/ebd-sample.txt", package = "auk")
# output to a temp file for example
# in practice, provide path to output file
# e.g. f_out <- "output/ebd_clean.txt"
f_out <- tempfile()

# clean file to remove problem rows
```

```
# note: this function is deprecated and no longer does anything
auk_clean(f, f_out)

## End(Not run)
```

---

auk_complete *Filter out incomplete checklists from the eBird data*

Description

Define a filter for the eBird Basic Dataset (EBD) to only keep complete checklists, i.e. those for which all birds seen or heard were recorded. These checklists are the most valuable for scientific uses since they provide presence and absence data.This function only defines the filter and, once all filters have been defined, auk_filter() should be used to call AWK and perform the filtering.

Usage

```
auk_complete(x)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by auk_ebd() or auk_sampling().

Details

This function can also work with on an auk_sampling object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_country, auk_date, auk_distance, auk_duration, auk_extent, auk_filter, auk_last_edited, auk_project, auk_protocol, auk_species, auk_state, auk_time

Examples

```
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_complete()
```

auk_country *Filter the eBird data by country*

Description

Define a filter for the eBird Basic Dataset (EBD) based on a set of countries. This function only defines the filter and, once all filters have been defined, auk_filter() should be used to call AWK and perform the filtering.

Usage

```
auk_country(x, country, replace = FALSE)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by auk_ebd() or auk_sampling().
country	character; countries to filter by. Countries can either be expressed as English names or ISO 2-letter country codes. English names are matched via regular expressions using countrycode, so there is some flexibility in names.
replace	logical; multiple calls to auk_country() are additive, unless replace = FALSE, in which case the previous list of countries to filter by will be removed and replaced by that in the current call.

Details

This function can also work with on an auk_sampling object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_complete, auk_date, auk_distance, auk_duration, auk_extent, auk_filter, auk_last_edited, auk_project, auk_protocol, auk_species, auk_state, auk_time

Examples

```
# country names and ISO2 codes can be mixed
# not case sensitive
country <- c("CA", "United States", "mexico")
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_country(country)
```

```
# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_country(ebd, country)
```

---

auk_date *Filter the eBird data by date*

Description

Define a filter for the eBird Basic Dataset (EBD) based on a range of dates. This function only defines the filter and, once all filters have been defined, auk_filter() should be used to call AWK and perform the filtering.

Usage

```
auk_date(x, date)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by auk_ebd() or auk_sampling().
date	character or date; date range to filter by, provided either as a character vector in the format "2015-12-31" or a vector of Date objects. To filter on a range of dates, regardless of year, use "*" in place of the year.

Details

To select observations from a range of dates, regardless of year, the wildcard "*" can be used in place of the year. For example, using `date = c("*-05-01", "*-06-30")` will return observations from May and June of *any year*.

This function can also work with on an auk_sampling object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_complete, auk_country, auk_distance, auk_duration, auk_extent, auk_filter, auk_last_edited, auk_project, auk_protocol, auk_species, auk_state, auk_time

Examples

```
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_date(date = c("2010-01-01", "2010-12-31"))

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_date(ebd, date = c("2010-01-01", "2010-12-31"))

# the * wildcard can be used in place of year to select dates from all years
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  # may-june records from all years
  auk_date(date = c("*-05-01", "*-06-30"))
```

---

auk_distance *Filter eBird data by distance travelled*

Description

Define a filter for the eBird Basic Dataset (EBD) based on the distance travelled on the checklist. This function only defines the filter and, once all filters have been defined, `auk_filter()` should be used to call AWK and perform the filtering. Note that stationary checklists (i.e. point counts) have no distance associated with them, however, since these checklists can be assumed to have 0 distance they will be kept if 0 is in the range defined by `distance`.

Usage

```
auk_distance(x, distance, distance_units)
```

Arguments

`x`	auk_ebd or auk_sampling object; reference to file created by `auk_ebd()` or `auk_sampling()`.
`distance`	integer; 2 element vector specifying the range of distances to filter by. The default is to accept distances in kilometers, use `distance_units = "miles"` for miles.
`distance_units`	character; whether distances are provided in kilometers (the default) or miles.

Details

This function can also work with on an `auk_sampling` object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_complete, auk_country, auk_date, auk_duration, auk_extent, auk_filter, auk_last_edited, auk_project, auk_protocol, auk_species, auk_state, auk_time

Examples

```
# only keep checklists that are less than 10 km long
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_distance(distance = c(0, 10))

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_distance(ebd, distance = c(0, 10))
```

---

auk_duration *Filter the eBird data by duration*

Description

Define a filter for the eBird Basic Dataset (EBD) based on the duration of the checklist. This function only defines the filter and, once all filters have been defined, auk_filter() should be used to call AWK and perform the filtering. Note that checklists with no effort, such as incidental observations, will be excluded if this filter is used since they have no associated duration information.

Usage

```
auk_duration(x, duration)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by auk_ebd() or auk_sampling().
duration	integer; 2 element vector specifying the range of durations in minutes to filter by.

Details

This function can also work with on an auk_sampling object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_complete, auk_country, auk_date, auk_distance, auk_extent, auk_filter, auk_last_edited, auk_project, auk_protocol, auk_species, auk_state, auk_time

Examples

```
# only keep checklists that are less than an hour long
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_duration(duration = c(0, 60))

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_duration(ebd, duration = c(0, 60))
```

---

auk_ebd *Reference to eBird data file*

Description

Create a reference to an eBird Basic Dataset (EBD) file in preparation for filtering using AWK.

Usage

```
auk_ebd(file, file_sampling, sep = "\t")
```

Arguments

file	character; input file. If file is not found as specified, it will be looked for in the directory specified by the EBD_PATH environment variable.
file_sampling	character; optional input sampling event data (i.e. checklists) file, required if you intend to zero-fill the data to produce a presence-absence data set. This file consists of just effort information for every eBird checklist. Any species not appearing in the EBD for a given checklist is implicitly considered to have a count of 0. This file should be downloaded at the same time as the basic dataset to ensure they are in sync. If file is not found as specified, it will be looked for in the directory specified by the EBD_PATH environment variable.
sep	character; the input field separator, the eBird data are tab separated so this should generally not be modified. Must only be a single character and space delimited is not allowed since spaces appear in many of the fields.

Details

eBird data can be downloaded as a tab-separated text file from the eBird website after submitting a request for access. As of February 2017, this file is nearly 150 GB making it challenging to work with. If you're only interested in a single species or a small region it is possible to submit a custom download request. This approach is suggested to speed up processing time.

There are two potential pathways for preparing eBird data. Users wishing to produce presence only data, should download the eBird Basic Dataset and reference this file when calling `auk_ebd()`. Users wishing to produce zero-filled, presence absence data should additionally download the sampling event data file associated with the basic dataset This file contains only checklist information and can be used to infer absences. The sampling event data file should be provided to `auk_ebd()` via the `file_sampling` argument. For further details consult the vignettes.

Value

An auk_ebd object storing the file reference and the desired filters once created with other package functions.

See Also

Other objects: `auk_sampling`

Examples

```
# get the path to the example data included in the package
# in practice, provide path to ebd, e.g. f <- "data/ebd_relFeb-2018.txt
f <- system.file("extdata/ebd-sample.txt", package = "auk")
auk_ebd(f)
# to produce zero-filled data, provide a checklist file
f_ebd <- system.file("extdata/zerofill-ex_ebd.txt", package = "auk")
f_cl <- system.file("extdata/zerofill-ex_sampling.txt", package = "auk")
auk_ebd(f_ebd, file_sampling = f_cl)
```

---

auk_ebd_version *Get the EBD version and associated taxonomy version*

Description

Based on the filename of eBird Basic Dataset (EBD) or sampling event data, determine the version (i.e. release date) of this EBD. Also determine the corresponding taxonomy version. The eBird taxonomy is updated annually in August.

Usage

```
auk_ebd_version(x, check_exists = TRUE)
```

Arguments

x	filename of EBD of sampling event data file, auk_ebd object, or auk_sampling object.
check_exists	logical; should the file be checked for existence before processing. If check_exists = TRUE and the file does not exists, the function will raise an error.

Value

A list with two elements:

- ebd_version: a date object specifying the release date of the EBD.
- taxonomy_version: the year of the taxonomy used in this EBD.

Both elements will be NA if an EBD version cannot be extracted from the filename.

See Also

Other helpers: auk_version, ebird_species, get_ebird_taxonomy

Examples

```
auk_ebd_version("ebd_relAug-2018.txt", check_exists = FALSE)
```

---

auk_extent *Filter the eBird data by spatial extent*

Description

Deprecated, use auk_bbox() instead.

Usage

```
auk_extent(x, extent)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by auk_ebd() or auk_sampling().
extent	numeric; spatial extent expressed as the range of latitudes and longitudes in decimal degrees: c(lng_min, lat_min, lng_max, lat_max). Note that longitudes in the Western Hemisphere and latitudes sound of the equator should be given as negative numbers.

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_complete, auk_country, auk_date, auk_distance, auk_duration, auk_filter, auk_last_edited, auk_project, auk_protocol, auk_species, auk_state, auk_time

Examples

```
# fliter to locations roughly in the Pacific Northwest
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_bbox(bbox = c(-125, 37, -120, 52))

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_bbox(ebd, bbox = c(-125, 37, -120, 52))
```

---

auk_filter *Filter the eBird file using AWK*

Description

Convert the filters defined in an auk_ebd object into an AWK script and run this script to produce a filtered eBird Reference Dataset (ERD). The initial creation of the auk_ebd object should be done with auk_ebd() and filters can be defined using the various other functions in this package, e.g. auk_species() or auk_country(). **Note that this function typically takes at least a couple hours to run on the full dataset**

Usage

```
auk_filter(x, file, ...)

## S3 method for class 'auk_ebd'
auk_filter(x, file, file_sampling, keep, drop, awk_file,
  sep = "\t", filter_sampling = TRUE, execute = TRUE,
  overwrite = FALSE, ...)

## S3 method for class 'auk_sampling'
auk_filter(x, file, keep, drop, awk_file,
  sep = "\t", execute = TRUE, overwrite = FALSE, ...)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by auk_ebd() or auk_sampling().
file	character; output file.
...	arguments passed on to methods.
file_sampling	character; optional output file for sampling data.

keep	character; a character vector specifying the names of the columns to keep in the output file. Columns should be as they appear in the header of the EBD; however, names are not case sensitive and spaces may be replaced by underscores, e.g. "COMMON NAME", "common name", and "common_NAME" are all valid.
drop	character; a character vector of columns to drop in the same format as keep. Ignored if keep is supplied.
awk_file	character; output file to optionally save the awk script to.
sep	character; the input field separator, the eBird file is tab separated by default. Must only be a single character and space delimited is not allowed since spaces appear in many of the fields.
filter_sampling	logical; whether the sampling event data should also be filtered.
execute	logical; whether to execute the awk script, or output it to a file for manual execution. If this flag is FALSE, awk_file must be provided.
overwrite	logical; overwrite output file if it already exists

Details

If a sampling file is provided in the auk_ebd object, this function will filter both the eBird Basic Dataset and the sampling data using the same set of filters. This ensures that the files are in sync, i.e. that they contain data on the same set of checklists.

The AWK script can be saved for future reference by providing an output filename to awk_file. The default behavior of this function is to generate and run the AWK script, however, by setting execute = FALSE the AWK script will be generated but not run. In this case, file is ignored and awk_file must be specified.

Calling this function requires that the command line utility AWK is installed. Linux and Mac machines should have AWK by default, Windows users will likely need to install Cygwin.

Value

An auk_ebd object with the output files set. If execute = FALSE, then the path to the AWK script is returned instead.

Methods (by class)

- auk_ebd: auk_ebd object
- auk_sampling: auk_sampling object

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_complete, auk_country, auk_date, auk_distance, auk_duration, auk_extent, auk_last_edited, auk_project, auk_protocol, auk_species, auk_state, auk_time

Examples

```
# get the path to the example data included in the package
# in practice, provide path to ebd, e.g. f <- "data/ebd_relFeb-2018.txt"
f <- system.file("extdata/ebd-sample.txt", package = "auk")
# define filters
filters <- auk_ebd(f) %>%
  auk_species(species = c("Canada Jay", "Blue Jay")) %>%
  auk_country(country = c("US", "Canada")) %>%
  auk_bbox(bbox = c(-100, 37, -80, 52)) %>%
  auk_date(date = c("2012-01-01", "2012-12-31")) %>%
  auk_time(start_time = c("06:00", "09:00")) %>%
  auk_duration(duration = c(0, 60)) %>%
  auk_complete()

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
filters <- auk_species(ebd, species = c("Canada Jay", "Blue Jay"))
filters <- auk_country(filters, country = c("US", "Canada"))
filters <- auk_bbox(filters, bbox = c(-100, 37, -80, 52))
filters <- auk_date(filters, date = c("2012-01-01", "2012-12-31"))
filters <- auk_time(filters, start_time = c("06:00", "09:00"))
filters <- auk_duration(filters, duration = c(0, 60))
filters <- auk_complete(filters)

# apply filters
## Not run:
# output to a temp file for example
# in practice, provide path to output file
# e.g. f_out <- "output/ebd_filtered.txt"
f_out <- tempfile()
filtered <- auk_filter(filters, file = f_out)
str(read_ebd(filtered))

## End(Not run)
```

---

auk_get_awk_path *OS specific path to AWK executable*

Description

Return the OS specific path to AWK (e.g. `"C:/cygwin64/bin/gawk.exe"` or `"/usr/bin/awk"`), or highlights if it's not installed. To manually set the path to AWK, set the `AWK_PATH` environment variable in your `.Renviron` file, which can be accomplished with the helper function `auk_set_awk_path(path)`.

Usage

```
auk_get_awk_path()
```

Value

Path to AWK or NA if AWK wasn't found.

See Also

Other paths: auk_get_ebd_path, auk_set_awk_path, auk_set_ebd_path

Examples

```
auk_get_awk_path()
```

---

auk_get_ebd_path *Return EBD data path*

Description

Returns the environment variable EBD_PATH, which users are encouraged to set to the directory that stores the eBird Basic Dataset (EBD) text files.

Usage

```
auk_get_ebd_path()
```

Value

The path stored in the EBD_PATH environment variable.

See Also

Other paths: auk_get_awk_path, auk_set_awk_path, auk_set_ebd_path

Examples

```
auk_get_ebd_path()
```

auk_last_edited *Filter the eBird data by last edited date*

Description

Define a filter for the eBird Basic Dataset (EBD) based on a range of last edited dates. Last edited date is typically used to extract just new or recently edited data. This function only defines the filter and, once all filters have been defined, auk_filter() should be used to call AWK and perform the filtering.

Usage

```
auk_last_edited(x, date)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by auk_ebd() or auk_sampling().
date	character or date; date range to filter by, provided either as a character vector in the format "2015-12-31" or a vector of Date objects.

Details

This function can also work with on an auk_sampling object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_complete, auk_country, auk_date, auk_distance, auk_duration, auk_extent, auk_filter, auk_project, auk_protocol, auk_species, auk_state, auk_time

Examples

```
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_last_edited(date = c("2010-01-01", "2010-12-31"))
```

auk_project *Filter the eBird data by project code*

Description

Some eBird records are collected as part of a particular project (e.g. the Virginia Breeding Bird Survey) and have an associated project code in the eBird dataset (e.g. EBIRD_ATL_VA). This function only defines the filter and, once all filters have been defined, auk_filter() should be used to call AWK and perform the filtering.

Usage

```
auk_project(x, project)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by auk_ebd() or auk_sampling().
project	character; project code to filter by (e.g. "EBIRD_MEX"). Multiple codes are accepted.

Details

This function can also work with on an auk_sampling object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_complete, auk_country, auk_date, auk_distance, auk_duration, auk_extent, auk_filter, auk_last_edited, auk_protocol, auk_species, auk_state, auk_time

Examples

```
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_project("EBIRD_MEX")

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_project(ebd, "EBIRD_MEX")
```

auk_protocol *Filter the eBird data by protocol*

Description

Filter to just data collected following a specific search protocol: stationary, traveling, or casual. This function only defines the filter and, once all filters have been defined, auk_filter() should be used to call AWK and perform the filtering.

Usage

```
auk_protocol(x, protocol)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by auk_ebd() or auk_sampling().
protocol	character. Many protocols exist in the database, however, this function only extracts the following protocols: • Stationary • Traveling • Area • Incidental • Nocturnal Flight Call Count • PROALAS Multiple protocols are allowed at the same time.

Details

This function can also work with on an auk_sampling object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_complete, auk_country, auk_date, auk_distance, auk_duration, auk_extent, auk_filter, auk_last_edited, auk_project, auk_species, auk_state, auk_time

Examples

```
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_protocol("Stationary")

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_protocol(ebd, "Stationary")
```

---

auk_rollup *Roll up eBird taxonomy to species*

Description

The eBird Basic Dataset (EBD) includes both true species and every other field-identifiable taxon that could be relevant for birders to report. This includes taxa not identifiable to a species (e.g. hybrids) and taxa reported below the species level (e.g. subspecies). This function produces a list of observations of true species, by removing the former and rolling the latter up to the species level. In the resulting EBD data.frame, `category` will be `"species"` for all records and the subspecies fields will be dropped. By default, `read_ebd()` calls `ebd_rollup()` when importing an eBird data file.

Usage

```
auk_rollup(x, taxonomy_version, drop_higher = TRUE)
```

Arguments

`x`	data.frame; data frame of eBird data, typically as imported by `read_ebd()`
`taxonomy_version`	integer; the version (i.e. year) of the taxonomy. In most cases, this should be left empty to use the version of the taxonomy included in the package. See `get_ebird_taxonomy()`.
`drop_higher`	logical; whether to remove taxa above species during the rollup process, e.g. "spuhs" like "duck sp.".

Details

When rolling observations up to species level the observed counts are summed across any taxa that resolve to the same species. However, if any of these taxa have a count of "X" (i.e. the observer did not enter a count), then the rolled up record will get an "X" as well. For example, if an observer saw 3 Myrtle and 2 Audubon's Warblers, this will roll up to 5 Yellow-rumped Warblers. However, if an "X" was entered for Myrtle, this would roll up to "X" for Yellow-rumped Warbler.

The eBird taxonomy groups taxa into eight different categories. These categories, and the way they are treated by `auk_rollup()` are as follows:

- **Species:** e.g., Mallard. Combined with lower level taxa if present on the same checklist.

- **ISSF or Identifiable Sub-specific Group:** Identifiable subspecies or group of subspecies, e.g., Mallard (Mexican). Rolled-up to species level.
- **Intergrade:** Hybrid between two ISSF (subspecies or subspecies groups), e.g., Mallard (Mexican intergrade. Rolled-up to species level.
- **Form:** Miscellaneous other taxa, including recently-described species yet to be accepted or distinctive forms that are not universally accepted (Red-tailed Hawk (Northern), Upland Goose (Bar-breasted)). If the checklist contains multiple taxa corresponding to the same species, the lower level taxa are rolled up, otherwise these records are left as is.
- **Spuh:** Genus or identification at broad level – e.g., duck sp., dabbling duck sp.. Dropped by `auk_rollup()`.
- **Slash:** Identification to Species-pair e.g., American Black Duck/Mallard). Dropped by `auk_rollup()`.
- **Hybrid:** Hybrid between two species, e.g., American Black Duck x Mallard (hybrid). Dropped by `auk_rollup()`.
- **Domestic:** Distinctly-plumaged domesticated varieties that may be free-flying (these do not count on personal lists) e.g., Mallard (Domestic type). Dropped by `auk_rollup()`.

The rollup process is based on the eBird taxonomy, which is updated once a year in August. The `auk` package includes a copy of the eBird taxonomy, current at the time of release; however, if the EBD and `auk` versions are not aligned, you may need to explicitly specify which version of the taxonomy to use, in which case the eBird API will be queried to get the correct version of the taxonomy.

Value

A data frame of the eBird data with taxonomic rollup applied.

References

Consult the eBird taxonomy page for further details.

See Also

Other pre: `auk_unique`

Examples

```
# get the path to the example data included in the package
# in practice, provide path to ebd, e.g. f <- "data/ebd_relFeb-2018.txt
f <- system.file("extdata/ebd-rollup-ex.txt", package = "auk")
# read in data without rolling up
ebd <- read_ebd(f, rollup = FALSE)
# rollup
ebd_ru <- auk_rollup(ebd)
# keep higher taxa
ebd_higher <- auk_rollup(ebd, drop_higher = FALSE)

# all taxa not identifiable to species are dropped
unique(ebd$category)
unique(ebd_ru$category)
```

```
unique(ebd_higher$category)

# yellow-rump warbler subspecies rollup
library(dplyr)
# without rollup, there are three observations
ebd %>%
  filter(common_name == "Yellow-rumped Warbler") %>%
  select(checklist_id, category, common_name, subspecies_common_name,
         observation_count)
# with rollup, they have been combined
ebd_ru %>%
  filter(common_name == "Yellow-rumped Warbler") %>%
  select(checklist_id, category, common_name, observation_count)
```

`auk_sampling` *Reference to eBird sampling event file*

Description

Create a reference to an eBird sampling event file in preparation for filtering using AWK. For working with the sightings data use `auk_ebd()`, only use `auk_sampling()` if you intend to only work with checklist-level data.

Usage

```
auk_sampling(file, sep = "\t")
```

Arguments

`file`	character; input sampling event data file, which contains checklist data from eBird.
`sep`	character; the input field separator, the eBird data are tab separated so this should generally not be modified. Must only be a single character and space delimited is not allowed since spaces appear in many of the fields.

Details

eBird data can be downloaded as a tab-separated text file from the eBird website after submitting a request for access. In the eBird Basic Dataset (EBD) each row corresponds to a observation of a single bird species on a single checklist, while the sampling event data file contains a single row for every checklist. This function creates an R object to reference only the sampling data.

Value

An `auk_sampling` object storing the file reference and the desired filters once created with other package functions.

See Also

Other objects: auk_ebd

Examples

```
# get the path to the example data included in the package
# in practice, provide path to the sampling event data
# e.g. f <- "data/ebd_sampling_relFeb-2018.txt"
f <- system.file("extdata/zerofill-ex_sampling.txt", package = "auk")
auk_sampling(f)
```

---

auk_select *Select a subset of columns*

Description

Select a subset of columns from the eBird Basic Dataset (EBD) or the sampling events file. Subsetting the columns can significantly decrease file size.

Usage

```
auk_select(x, select, file, sep = "\t", overwrite = FALSE)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by auk_ebd() or auk_sampling().
select	character; a character vector specifying the names of the columns to select. Columns should be as they appear in the header of the EBD; however, names are not case sensitive and spaces may be replaced by underscores, e.g. "COMMON NAME", "common name", and "common_NAME" are all valid.
file	character; output file.
sep	character; the input field separator, the eBird file is tab separated by default. Must only be a single character and space delimited is not allowed since spaces appear in many of the fields.
overwrite	logical; overwrite output file if it already exists

Value

Invisibly returns the filename of the output file.

See Also

Other text: auk_clean, auk_split

Examples

```
## Not run:
# select a minimal set of columns
out_file <- tempfile()
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
cols <- c("latitude", "longitude",
          "group identifier", "sampling event identifier",
          "scientific name", "observation count")
selected <- auk_select(ebd, select = cols, file = out_file)
str(read_ebd(selected))

## End(Not run)
```

---

auk_set_awk_path *Set a custom path to AWK executable*

Description

If AWK has been installed in a non-standard location, the environment variable AWK_PATH must be set to specify the location of the executable. Use this function to set AWK_PATH in your .Renviron file. **Most users should NOT set** AWK_PATH**, only do so if you have installed AWK in non-standard location and** auk **cannot find it.**

Usage

```
auk_set_awk_path(path, overwrite = FALSE)
```

Arguments

path	character; path to the AWK executable on your system, e.g. "C:/cygwin64/bin/gawk.exe" or "/usr/bin/awk".
overwrite	logical; should the existing AWK_PATH be overwritten if it has already been set in .Renviron.

Value

Edits .Renviron, then returns the AWK path invisibly.

See Also

Other paths: auk_get_awk_path, auk_get_ebd_path, auk_set_ebd_path

Examples

```
## Not run:
auk_set_awk_path("/usr/bin/awk")

## End(Not run)
```

auk_set_ebd_path *Set the path to EBD text files*

Description

Users of auk are encouraged to set the path to the directory containing the eBird Basic Dataset (EBD) text files in the EBD_PATH environment variable. All functions referencing the EBD or sampling event data files will check in this directory to find the files, thus avoiding the need to specify the full path every time. This will increase the portability of your code. Use this function to set EBD_PATH in your .Renviron file; it is also possible to manually edit the file.

Usage

```
auk_set_ebd_path(path, overwrite = FALSE)
```

Arguments

path	character; directory where the EBD text files are stored, e.g. "/home/matt/ebd".
overwrite	logical; should the existing EDB_PATH be overwritten if it has already been set in .Renviron.

Value

Edits .Renviron, then returns the AWK path invisibly.

See Also

Other paths: auk_get_awk_path, auk_get_ebd_path, auk_set_awk_path

Examples

```
## Not run:
auk_set_ebd_path("/home/matt/ebd")

## End(Not run)
```

auk_species *Filter the eBird data by species*

Description

Define a filter for the eBird Basic Dataset (EBD) based on species. This function only defines the filter and, once all filters have been defined, auk_filter() should be used to call AWK and perform the filtering.

Usage

```
auk_species(x, species, taxonomy_version, replace = FALSE)
```

Arguments

x	auk_ebd object; reference to object created by auk_ebd().
species	character; species to filter by, provided as scientific or English common names, or a mixture of both. These names must match the official eBird Taxomony (ebird_taxonomy).
taxonomy_version	integer; the version (i.e. year) of the taxonomy. In most cases, this should be left empty to use the version of the taxonomy included in the package. See get_ebird_taxonomy().
replace	logical; multiple calls to auk_species() are additive, unless replace = FALSE, in which case the previous list of species to filter by will be removed and replaced by that in the current call.

Details

The list of species is checked against the eBird taxonomy for validity. This taxonomy is updated once a year in August. The auk package includes a copy of the eBird taxonomy, current at the time of release; however, if the EBD and auk versions are not aligned, you may need to explicitly specify which version of the taxonomy to use, in which case the eBird API will be queried to get the correct version of the taxonomy.

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_complete, auk_country, auk_date, auk_distance, auk_duration, auk_extent, auk_filter, auk_last_edited, auk_project, auk_protocol, auk_state, auk_time

Examples

```
# common and scientific names can be mixed
species <- c("Canada Jay", "Pluvialis squatarola")
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_species(species)

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_species(ebd, species)
```

auk_split *Split an eBird data file by species*

Description

Given an eBird Basic Dataset (EBD) and a list of species, split the file into multiple text files, one for each species. This function is typically used after `auk_filter()` has been applied if the resulting file is too large to be read in all at once.

Usage

```
auk_split(file, species, taxonomy_version, prefix = "", ext = "txt",
  sep = "\t", overwrite = FALSE)
```

Arguments

`file`	character; input file.
`species`	species character; species to filter and split by, provided as scientific or English common names, or a mixture of both. These names must match the official eBird Taxomony (ebird_taxonomy).
`taxonomy_version`	integer; the version (i.e. year) of the taxonomy. In most cases, this should be left empty to use the version of the taxonomy included in the package. See `get_ebird_taxonomy()`.
`prefix`	character; a file and directory prefix. For example, if splitting by species "A" and "B" and `prefix = "data/ebd_"`, the resulting files will be "data/ebd_A.txt" and "data/ebd_B.txt".
`ext`	character; file extension, typically "txt".
`sep`	character; the input field separator, the eBird file is tab separated by default. Must only be a single character and space delimited is not allowed since spaces appear in many of the fields.
`overwrite`	logical; overwrite output files if they already exists.

Details

The list of species is checked against the eBird taxonomy for validity. This taxonomy is updated once a year in August. The `auk` package includes a copy of the eBird taxonomy, current at the time of release; however, if the EBD and `auk` versions are not aligned, you may need to explicitly specify which version of the taxonomy to use, in which case the eBird API will be queried to get the correct version of the taxonomy.

Value

A vector of output filenames, one for each species.

See Also

Other text: auk_clean, auk_select

Examples

```
## Not run:
species <- c("Canada Jay", "Cyanocitta stelleri")
# get the path to the example data included in the package
# in practice, provide path to a filtered ebd file
# e.g. f <- "data/ebd_filtered.txt
f <- system.file("extdata/ebd-sample.txt", package = "auk")
# output to a temporary directory for example
# in practice, provide the path to the output location
# e.g. prefix <- "output/ebd_"
prefix <- file.path(tempdir(), "ebd_")
species_files <- auk_split(f, species = species, prefix = prefix)

## End(Not run)
```

---

auk_state *Filter the eBird data by state*

Description

Define a filter for the eBird Basic Dataset (EBD) based on a set of states. This function only defines the filter and, once all filters have been defined, auk_filter() should be used to call AWK and perform the filtering.

Usage

```
auk_state(x, state, replace = FALSE)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by auk_ebd() or auk_sampling().
state	character; states to filter by. eBird uses 4 to 6 character state codes consisting of two parts, the 2-letter ISO country code and a 1-3 character state code, separated by a dash. For example, "US-NY" corresponds to New York State in the United States. Refer to the data frame ebird_states for look up state codes.
replace	logical; multiple calls to auk_state() are additive, unless replace = FALSE, in which case the previous list of states to filter by will be removed and replaced by that in the current call.

Details

It is not possible to filter by both country and state, so calling `auk_state()` will reset the country filter to all countries, and vice versa.

This function can also work with on an `auk_sampling` object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: `auk_bbox`, `auk_bcr`, `auk_breeding`, `auk_complete`, `auk_country`, `auk_date`, `auk_distance`, `auk_duration`, `auk_extent`, `auk_filter`, `auk_last_edited`, `auk_project`, `auk_protocol`, `auk_species`, `auk_time`

Examples

```
# state codes for a given country can be looked up in ebird_states
dplyr::filter(ebird_states, country == "Costa Rica")
# choose texas, united states and puntarenas, cost rica
states <- c("US-TX", "CR-P")
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_state(states)

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_state(ebd, states)
```

---

auk_time *Filter the eBird data by checklist start time*

Description

Define a filter for the eBird Basic Dataset (EBD) based on a range of start times for the checklist. This function only defines the filter and, once all filters have been defined, `auk_filter()` should be used to call AWK and perform the filtering.

Usage

```
auk_time(x, start_time)
```

Arguments

x	auk_ebd or auk_sampling object; reference to file created by `auk_ebd()` or `auk_sampling()`.
start_time	character; 2 element character vector giving the range of times in 24 hour format, e.g. `"06:30"` or `"16:22"`.

Details

This function can also work with on an auk_sampling object if the user only wishes to filter the sampling event data.

Value

An auk_ebd object.

See Also

Other filter: auk_bbox, auk_bcr, auk_breeding, auk_complete, auk_country, auk_date, auk_distance, auk_duration, auk_extent, auk_filter, auk_last_edited, auk_project, auk_protocol, auk_species, auk_state

Examples

```
# only keep checklists started between 6 and 8 in the morning
system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_time(start_time = c("06:00", "08:00"))

# alternatively, without pipes
ebd <- auk_ebd(system.file("extdata/ebd-sample.txt", package = "auk"))
auk_time(ebd, start_time = c("06:00", "08:00"))
```

---

auk_unique *Remove duplicate group checklists*

Description

eBird checklists can be shared among a group of multiple observers, in which case observations will be duplicated in the database. This functions removes these duplicates from the eBird Basic Dataset (EBD) or the EBD sampling event data (with checklists_only = TRUE), creating a set of unique bird observations. This function is called automatically by read_ebd() and read_sampling().

Usage

```
auk_unique(x, group_id = "group_identifier",
  checklist_id = "sampling_event_identifier",
  species_id = "scientific_name", observer_id = "observer_id",
  checklists_only = FALSE)
```

Arguments

x	data.frame; the EBD data frame, typically as imported by read_ebd().
group_id	character; the name of the group ID column.

`checklist_id` character; the name of the checklist ID column, each checklist within a group will get a unique value for this field. The record with the lowest `checklist_id` will be picked as the unique record within each group. In the output dataset, this field will be updated to have a full list of the checklist IDs that went into this group checklist.

`species_id` character; the name of the column identifying species uniquely. This is required to ensure that removing duplicates is done independently for each species. Note that this will not treat sub-species independently and, if that behavior is desired, the user will have to generate a column uniquely identifying species and sub-species and pass that column's name to this argument.

`observer_id` character; the name of the column identifying the owner of this instance of the group checklist. In the output dataset, the full list of observer IDs will be stored (comma separated) in the new `observer_id` field. The order of these IDs will match the order of the comma separated checklist IDs.

`checklists_only` logical; whether the dataset provided only contains checklist information as with the sampling event data file. If this argument is `TRUE`, then the `species_id` argument is ignored and removing of duplicated records is done at the checklist level not the species level.

Details

This function chooses the checklist within in each that has the lowest value for the field specified by `checklist_id`. A new column is also created, `checklist_id`, whose value is the taken from the field specified in the `checklist_id` parameter for non-group checklists and from the field specified by the `group_id` parameter for grouped checklists.

All the checklist and observer IDs for the checklists that comprise a given group checklist will be retained as a comma separated string ordered by checklist ID.

Value

A data frame with unique observations, and an additional field, `checklist_id`, which is a combination of the sampling event and group IDs.

See Also

Other pre: `auk_rollup`

Examples

```
# read in an ebd file and don't automatically remove duplicates
f <- system.file("extdata/ebd-sample.txt", package = "auk")
ebd <- read_ebd(f, unique = FALSE)
# remove duplicates
ebd_unique <- auk_unique(ebd)
nrow(ebd)
nrow(ebd_unique)
```

auk_version *Versions of auk, the EBD, and the eBird taxonomy*

Description

This package depends on the version of the EBD and on the eBird taxonomy. Use this function to determine the currently installed version of auk, the version of the EBD that this auk version works with, and the version of the eBird taxonomy included in the packages. The EBD is update quarterly, in March, June, September, and December, while the taxonomy is updated annually in August or September. To ensure proper functioning, always use the latest version of the auk package and the EBD.

Usage

```
auk_version()
```

Value

A list with three elements:

- auk_version: the version of auk, e.g. "auk 0.3.0".
- ebd_version: a date object specifying the release date of the EBD version that this auk version is designed to work with.
- taxonomy_version: the year of the taxonomy built in to this version of auk, i.e. the one stored in ebird_taxonomy.

See Also

Other helpers: auk_ebd_version, ebird_species, get_ebird_taxonomy

Examples

```
auk_version()
```

auk_zerofill *Read and zero-fill an eBird data file*

Description

Read an eBird Basic Dataset (EBD) file, and associated sampling event data file, to produce a zero-filled, presence-absence dataset. The EBD contains bird sightings and the sampling event data is a set of all checklists, they can be combined to infer absence data by assuming any species not reported on a checklist was had a count of zero.

Usage

```
auk_zerofill(x, ...)

## S3 method for class 'data.frame'
auk_zerofill(x, sampling_events, species,
  taxonomy_version, collapse = FALSE, unique = TRUE, rollup = TRUE,
  drop_higher = TRUE, complete = TRUE, ...)

## S3 method for class 'character'
auk_zerofill(x, sampling_events, species,
  taxonomy_version, collapse = FALSE, unique = TRUE, rollup = TRUE,
  drop_higher = TRUE, complete = TRUE, sep = "\t", ...)

## S3 method for class 'auk_ebd'
auk_zerofill(x, species, taxonomy_version,
  collapse = FALSE, unique = TRUE, rollup = TRUE,
  drop_higher = TRUE, complete = TRUE, sep = "\t", ...)

collapse_zerofill(x)
```

Arguments

x	filename, data.frame of eBird observations, or auk_ebd object with associated output files as created by auk_filter(). If a filename is provided, it must point to the EBD and the sampling_events argument must point to the sampling event data file. If a data.frame is provided it should have been imported with read_ebd(), to ensure the variables names have been set correctly, and it must have been passed through auk_unique() to ensure duplicate group checklists have been removed.
...	additional arguments passed to methods.
sampling_events	character or data.frame; filename for the sampling event data or a data.frame of the same data. If a data.frame is provided it should have been imported with read_sampling(), to ensure the variables names have been set correctly, and it must have been passed through auk_unique() to ensure duplicate group checklists have been removed.
species	character; species to include in zero-filled dataset, provided as scientific or English common names, or a mixture of both. These names must match the official eBird Taxomony (ebird_taxonomy). To include all species, leave this argument blank.
taxonomy_version	integer; the version (i.e. year) of the taxonomy. In most cases, this should be left empty to use the version of the taxonomy included in the package. See get_ebird_taxonomy().
collapse	logical; whether to call collapse_zerofill() to return a data frame rather than an auk_zerofill object.
unique	logical; should auk_unique() be run on the input data if it hasn't already.

rollup	logical; should auk_rollup() be run on the input data if it hasn't already.
drop_higher	logical; whether to remove taxa above species during the rollup process, e.g. "spuhs" like "duck sp.". See auk_rollup().
complete	logical; if TRUE (the default) all checklists are required to be complete prior to zero-filling.
sep	character; single character used to separate fields within a row.

Details

auk_zerofill() generates an auk_zerofill object consisting of a list with elements observations and sampling_events. observations is a data frame giving counts and binary presence/absence data for each species. sampling_events is a data frame with checklist level information. The two data frames can be connected via the checklist_id field. This format is efficient for storage since the checklist columns are not duplicated for each species, however, working with the data often requires joining the two data frames together.

To return a data frame, set collapse = TRUE. Alternatively, zerofill_collapse() generates a data frame from an auk_zerofill object, by joining the two data frames together to produce a single data frame in which each row provides both checklist and species information for a sighting.

The list of species is checked against the eBird taxonomy for validity. This taxonomy is updated once a year in August. The auk package includes a copy of the eBird taxonomy, current at the time of release; however, if the EBD and auk versions are not aligned, you may need to explicitly specify which version of the taxonomy to use, in which case the eBird API will be queried to get the correct version of the taxonomy.

Value

By default, an auk_zerofill object, or a data frame if collapse = TRUE.

Methods (by class)

- data.frame: EBD data frame.
- character: Filename of EBD.
- auk_ebd: auk_ebd object output from auk_filter(). Must have had a sampling event data file set in the original call to auk_ebd().

See Also

Other import: read_ebd

Examples

```
# read and zero-fill the ebd data
f_ebd <- system.file("extdata/zerofill-ex_ebd.txt", package = "auk")
f_smpl <- system.file("extdata/zerofill-ex_sampling.txt", package = "auk")
auk_zerofill(x = f_ebd, sampling_events = f_smpl)

# use the species argument to only include a subset of species
auk_zerofill(x = f_ebd, sampling_events = f_smpl,
```

```
              species = "Collared Kingfisher")

# to return a data frame use collapse = TRUE
ebd_df <- auk_zerofill(x = f_ebd, sampling_events = f_smpl, collapse = TRUE)
```

`bcr_codes` *BCR Codes*

Description

A data frame of Bird Conservation Region (BCR) codes. BCRs are ecologically distinct regions in North America with similar bird communities, habitats, and resource management issues. These codes are required to filter by BCR using `auk_bcr()`.

Usage

```
bcr_codes
```

Format

A data frame with two variables and 66 rows:

- `bcr_code`: integer code from 1 to 66.
- `bcr_name`: name of BCR.

See Also

Other data: `ebird_states`, `ebird_taxonomy`

`ebird_species` *Lookup species in eBird taxonomy*

Description

Given a list of common or scientific names, check that they appear in the official eBird taxonomy and convert them all to scientific names, common names, or species codes. Un-matched species are returned as `NA`.

Usage

```
ebird_species(x, type = c("scientific", "common", "code", "all"),
  taxonomy_version)
```

Arguments

x	character; species to look up, provided as scientific or English common names, or a mixture of both. Case insensitive.
type	character; whether to return scientific names (`scientific`), English common names (`common`), or 6-letter eBird species codes (`code`). Alternatively, use `all` to return a data frame with the all the taxonomy information.
taxonomy_version	integer; the version (i.e. year) of the taxonomy. Leave empty to use the version of the taxonomy included in the package. See `get_ebird_taxonomy()`.

Value

Character vector of species identified by scientific name, common name, or species code. If `type = "all"` a data frame of the taxonomy of the requested species is returned.

See Also

Other helpers: `auk_ebd_version`, `auk_version`, `get_ebird_taxonomy`

Examples

```
# mix common and scientific names, case-insensitive
species <- c("Blackburnian Warbler", "Poecile atricapillus",
             "american dipper", "Caribou")
# note that species not in the ebird taxonomy return NA
ebird_species(species)

# use taxonomy_version to query older taxonomy versions
ebird_species("Cordillera Azul Antbird")
ebird_species("Cordillera Azul Antbird", taxonomy_version = 2017)
```

---

ebird_states *eBird States*

Description

A data frame of state codes used by eBird. These codes are 4 to 6 characters, consisting of two parts, the 2-letter ISO country code and a 1-3 character state code, separated by a dash. For example, `"US-NY"` corresponds to New York State in the United States. These state codes are required to filter by state using `auk_state()`.

Usage

```
ebird_states
```

Format

A data frame with four variables and 3,145 rows:

- `country`: short form of English country name.
- `country_code`: 2-letter ISO country code.
- `state`: state name.
- `state_code`: 4 to 6 character state code.

Details

Note that some countries are not broken into states in eBird and therefore do not appear in this data frame.

See Also

Other data: `bcr_codes`, `ebird_taxonomy`

`ebird_taxonomy` *eBird Taxonomy*

Description

A simplified version of the taxonomy used by eBird. Includes proper species as well as various other categories such as `spuh` (e.g. *duck sp.*) and *slash* (e.g. *American Black Duck/Mallard*). This taxonomy is based on the Clements Checklist, which is updated annually, typically in the late summer. Non-ASCII characters (e.g. those with accents) have been converted to ASCII equivalents in this data frame.

Usage

```
ebird_taxonomy
```

Format

A data frame with eight variables and 16,248 rows:

- `scientific_name`: scientific name.
- `common_name`: common name, defaults to English, but different languages can be selected using the `locale` parameter.
- `species_code`: a unique alphanumeric code identifying each species.
- `category`: whether the entry is for a species or another field-identifiable taxon, such as `spuh`, `slash`, `hybrid`, etc.
- `taxon_order`: numeric value used to sort rows in taxonomic order.
- `order`: the scientific name of the order that the species belongs to.
- `family`: the scientific name of the family that the species belongs to.

- `report_as`: for taxa that can be resolved to true species (i.e. species, subspecies, and recognizable forms), this field links to the corresponding species code. For taxa that can't be resolved, this field is `NA`.

For further details, see http://help.ebird.org/customer/en/portal/articles/1006825-the-ebird-taxonomy

See Also

Other data: `bcr_codes`, `ebird_states`

`filter_repeat_visits` *Filter observations to repeat visits for hierarchical modeling*

Description

Hierarchical modeling of abundance and occurrence requires repeat visits to sites to estimate detectability. These visits should be all be within a period of closure, i.e. when the population can be assumed to be closed. eBird data, and many other data sources, do not explicitly follow this protocol; however, subsets of the data can be extracted to produce data suitable for hierarchical modeling. This function extracts a subset of observation data that have a desired number of repeat visits within a period of closure.

Usage

```
filter_repeat_visits(x, min_obs = 2L, max_obs = 10L, n_days = 14L,
  annual_closure = FALSE, date_var = "observation_date",
  site_vars = c("locality_id", "observer_id"))
```

Arguments

`x`	`data.frame`; observation data, e.g. data from the eBird Basic Dataset (EBD) zero-filled with `auk_zerofill()`. This function will also work with an `auk_zerofill` object, in which case it will be converted to a data frame with `collapse_zerofill()`. **Note that these data must for a single species**.
`min_obs`	integer; minimum number of observations required for each site.
`max_obs`	integer; maximum number of observations allowed for each site.
`n_days`	integer; number of days defining the temporal length of closure. Ignored if `annual_closure = TRUE`.
`annual_closure`	logical; whether the entire year should be treated as the period of closure. This can be useful, for example, if data are from one season (e.g. breeding) across multiple years.
`date_var`	character; column name of the variable in `x` containing the date. This column should either be in `Date` format or convertible to `Date` format with `as.Date()`.
`site_vars`	character; names of one of more columns in `x` that define a site, typically the location and observer IDs.

Details

In addition to specifying the minimum and maximum number of observations per site, users must specify the variables in the dataset that define a "site". This is typically a combination of IDs defining the geographic site and the unique observer (repeat visits are meant to be conducted by the same observer). Finally, the number of days defining the period of closure is required. A default value of 14 days is used; however, users should choose a suitable period for their species within which the population can reasonably be assumed to be closed.

Value

A `data.frame` filtered to only retain observations from sites with the allowed number of observations within the period of closure. The results will be sorted such that sites are together and in chronological order. The following variables are added to the data frame:

- `site`: a unique identifier for each "site" corresponding to all the variables in `site_vars` and `closure_id` concatenated together with underscore separators.
- `closure_id`: a unique ID for each closure period. If `annual_closure = TRUE`, this will be the year. Otherwise, it will be the number of blocks of `n_days` days since the earliest observation. Note that in this latter case, there may be gaps in the IDs.
- `n_observations`: number of observations at each site after all filtering.

See Also

Other modeling: `format_unmarked_occu`

Examples

```
# read and zero-fill the ebd data
f_ebd <- system.file("extdata/zerofill-ex_ebd.txt", package = "auk")
f_smpl <- system.file("extdata/zerofill-ex_sampling.txt", package = "auk")
# data must be for a single species
ebd_zf <- auk_zerofill(x = f_ebd, sampling_events = f_smpl,
                       species = "Collared Kingfisher",
                       collapse = TRUE)
filter_repeat_visits(ebd_zf, n_days = 30)
```

`format_unmarked_occu` *Format EBD data for occupancy modeling with* `unmarked`

Description

Prepare a data frame of species observations for ingestion into the package `unmarked` for hierarchical modeling of abundance and occurrence. The function `unmarked::formatWide()` takes a data frame and converts it to one of several `unmarked` objects, which can then be used for modeling. This function converts data from a format in which each row is an observation (e.g. as in the eBird Basic Dataset) to the esoteric format required by `unmarked::formatWide()` in which each row is a site.

Usage

```
format_unmarked_occu(x, site_id = "site",
  response = "species_observed", site_covs, obs_covs)
```

Arguments

x	data.frame; observation data, e.g. from the eBird Basic Dataset (EBD), for **a single species**, that has been filtered to those with repeat visits by filter_repeat_visits().
site_id	character; a unique idenitifer for each "site", typically identifying observations from a unique location by the same observer within a period of temporal closure. Data output from filter_repeat_visits() will have a .site_id variable that meets these requirements.
response	character; the variable that will act as the response in modeling efforts, typically a binary variable indicating presence or absence or a count of individuals seen.
site_covs	character; the variables that will act as site-level covariates, i.e. covariates that vary at the site level, for example, latitude/longitude or habitat predictors. If this parameter is missing, it will be assumed that any variable that is not an observation-level covariate (obs_covs) or the site_id, is a site-level covariate.
obs_covs	character; the variables that will act as observation-level covariates, i.e. covariates that vary within sites, at the level of observations, for example, time or length of observation.

Details

Hierarchical modeling requires repeat observations at each "site" to estimate detectability. A "site" is typically defined as a geographic location visited by the same observer within a period of temporal closure. To define these sites and filter out observations that do not correspond to repeat visits, users should use filter_repeat_visits(), then pass the output to this function.

format_unmarked_occu() is designed to prepare data to be converted into an unmarkedFrameOccu object for occupancy modeling with unmarked::occu(); however, it can also be used to prepare data for conversion to an unmarkedFramePCount object for abundance modeling with unmarked::pcount().

Value

A data frame that can be processed by unmarked::formatWide(). Each row will correspond to a unqiue site and, assuming there are a maximum of N observations per site, columns will be as follows:

1. The unique site identifier, named "site".
2. N response columns, one for each observation, named "y.1", ..., "y.N".
3. Columns for each of the site-level covariates.
4. Groups of N columns of observation-level covariates, one column per covariate per observation, names "covariate_name.1", ..., "covariate_name.N".

See Also

Other modeling: filter_repeat_visits

Examples

```
# read and zero-fill the ebd data
f_ebd <- system.file("extdata/zerofill-ex_ebd.txt", package = "auk")
f_smpl <- system.file("extdata/zerofill-ex_sampling.txt", package = "auk")
# data must be for a single species
ebd_zf <- auk_zerofill(x = f_ebd, sampling_events = f_smpl,
                       species = "Collared Kingfisher",
                       collapse = TRUE)
occ <- filter_repeat_visits(ebd_zf, n_days = 30)
# format for unmarked
# typically one would join in habitat covariates prior to this step
occ_wide <- format_unmarked_occu(occ,
                                 response = "species_observed",
                                 site_covs = c("latitude", "longitude"),
                                 obs_covs = c("effort_distance_km",
                                              "duration_minutes"))
# create an unmarked object
if (requireNamespace("unmarked", quietly = TRUE)) {
  occ_um <- unmarked::formatWide(occ_wide, type = "unmarkedFrameOccu")
  unmarked::summary(occ_um)
}

# this function can also be used for abundance modeling
abd <- ebd_zf %>%
  # convert count to integer, drop records with no count
  dplyr::mutate(observation_count = as.integer(observation_count)) %>%
  dplyr::filter(!is.na(observation_count)) %>%
  # filter to repeated visits
  filter_repeat_visits(n_days = 30)
# prepare for conversion to unmarkedFramePCount object
abd_wide <- format_unmarked_occu(abd,
                                 response = "observation_count",
                                 site_covs = c("latitude", "longitude"),
                                 obs_covs = c("effort_distance_km",
                                              "duration_minutes"))
# create an unmarked object
if (requireNamespace("unmarked", quietly = TRUE)) {
  abd_um <- unmarked::formatWide(abd_wide, type = "unmarkedFrameOccu")
  unmarked::summary(abd_um)
}
```

---

get_ebird_taxonomy *Get eBird taxonomy via the eBird API*

Description

Get the taxonomy used in eBird via the eBird API.

Usage

```
get_ebird_taxonomy(version, locale)
```

Arguments

version	integer; the version (i.e. year) of the taxonomy. The eBird taxonomy is updated once a year in August. Leave this parameter blank to get the current taxonomy.
locale	character; the locale for the common names, defaults to English.

Value

A data frame of all species in the eBird taxonomy, consisting of the following columns:

- scientific_name: scientific name.
- common_name: common name, defaults to English, but different languages can be selected using the locale parameter.
- species_code: a unique alphanumeric code identifying each species.
- category: whether the entry is for a species or another field-identifiable taxon, such as spuh, slash, hybrid, etc.
- taxon_order: numeric value used to sort rows in taxonomic order.
- order: the scientific name of the order that the species belongs to.
- family: the scientific name of the family that the species belongs to.
- report_as: for taxa that can be resolved to true species (i.e. species, subspecies, and recognizable forms), this field links to the corresponding species code. For taxa that can't be resolved, this field is NA.

See Also

Other helpers: auk_ebd_version, auk_version, ebird_species

Examples

```
## Not run:
get_ebird_taxonomy()

## End(Not run)
```

---

read_ebd *Read an EBD file*

Description

Read an eBird Basic Dataset file using data.table::fread(), readr::read_delim(), or read.delim() depending on which packages are installed. read_ebd() reads the EBD itself, while read_sampling()' reads a sampling event data file.

Usage

```
read_ebd(x, reader, sep = "\t", unique = TRUE, rollup = TRUE)

## S3 method for class 'character'
read_ebd(x, reader, sep = "\t", unique = TRUE,
  rollup = TRUE)

## S3 method for class 'auk_ebd'
read_ebd(x, reader, sep = "\t", unique = TRUE,
  rollup = TRUE)

read_sampling(x, reader, sep = "\t", unique = TRUE)

## S3 method for class 'character'
read_sampling(x, reader, sep = "\t",
  unique = TRUE)

## S3 method for class 'auk_ebd'
read_sampling(x, reader, sep = "\t", unique = TRUE)

## S3 method for class 'auk_sampling'
read_sampling(x, reader, sep = "\t",
  unique = TRUE)
```

Arguments

x	filename or auk_ebd object with associated output files as created by auk_filter().
reader	character; the function to use for reading the input file, options are "fread", "readr", or "base", for data.table::fread(), readr::read_delim(), or read.delim(), respectively. This argument should typically be left empty to have the function choose the best reader based on the installed packages.
sep	character; single character used to separate fields within a row.
unique	logical; should duplicate grouped checklists be removed. If unique = TRUE, auk_unique() is called on the EBD before returning.
rollup	logical; should taxonomic rollup to species level be applied. If rollup = TRUE, auk_rollup() is called on the EBD before returning. Note that this process can be time consuming for large files, try turning rollup off if reading is taking too long.

Details

This functions performs the following processing steps:

- Data types for columns are manually set based on column names used in the February 2017 EBD. If variables are added or names are changed in later releases, any new variables will have data types inferred by the import function used.
- Variables names are converted to snake_case.

- Duplicate observations resulting from group checklists are removed using auk_unique(), unless unique = FALSE.

Value

A data frame of EBD observations. An additional column, checklist_id, is added to output files if unique = TRUE, that uniquely identifies the checklist from which the observation came. This field is equal to sampling_event_identifier for non-group checklists, and group_identifier for group checklists.

Methods (by class)

- character: Filename of EBD.
- auk_ebd: auk_ebd object output from auk_filter()
- character: Filename of sampling event data file
- auk_ebd: auk_ebd object output from auk_filter(). Must have had a sampling event data file set in the original call to auk_ebd().
- auk_sampling: auk_sampling object output from auk_filter().

See Also

Other import: auk_zerofill

Examples

```
f <- system.file("extdata/ebd-sample.txt", package = "auk")
read_ebd(f)
# read a sampling event data file
x <- system.file("extdata/zerofill-ex_sampling.txt", package = "auk") %>%
  read_sampling()
```

Index