Package 'auk'

July 6, 2017

Title eBird Data Extraction with AWK

Version 0.0.2

Date 2017-07-05

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

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

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.

Depends R (>= 3.3.3)

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

Imports assertthat, stringr, stringi, magrittr, countrycode, tidyr

RoxygenNote 6.0.1

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

VignetteBuilder knitr

NeedsCompilation no

Author Matthew Strimas-Mackey [aut, cre], Eliot Miller [aut], Wesley Hochachka [aut], Cornell Lab of Ornithology [cph]

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

Repository CRAN

Date/Publication 2017-07-05 22:32:02 UTC

R topics documented:

---

auk *auk: eBird Data Processing with AWK*

Description

Use the command line utility AWK to process eBird data.

---

auk_clean *Clean an EBD file*

Description

Some rows in the eBird Basic Dataset (EBD) may have an incorrect number of columns, often resulting from tabs embedded in the comments field. This function drops these problematic records. **Note that this function typically takes at least 3 hours to run on the full EBD.**

Usage

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

Arguments

f_in	character; input file.
f_out	character; output file.
sep	character; the input field separator, the EBD 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_blank	logical; whether the trailing blank should be removed from the end of each row. The EBD comes with an extra tab at the end of each line, which causes a extra blank column.
overwrite	logical; overwrite output file if it already exists

Details

This function can clean an EBD file or an EBD sampling file.

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

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

Examples

```
## Not run:
# example data with errors
f <- system.file("extdata/ebd-sample_messy.txt", package = "auk")
tmp <- tempfile()

# clean file to remove problem rows
auk_clean(f, tmp)
# number of lines in input
length(readLines(f))
# number of lines in output
length(readLines(tmp))

# note that the extra blank column has also been removed
ncol(read.delim(f, nrows = 5, quote = ""))
ncol(read.delim(tmp, nrows = 5, quote = ""))
unlink(tmp)

## End(Not run)
```

auk_complete *Filter out incomplete checklists from the EBD*

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 object; reference to EBD file created by `auk_ebd()`.

Value

An auk_ebd object.

Examples

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

auk_country *Filter the EBD 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)
```

Arguments

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

Value

An auk_ebd object.

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)
```

---

auk_date *Filter the EBD 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 object; reference to EBD file created by auk_ebd().
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.

Value

An auk_ebd object.

Examples

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

---

auk_duration *Filter the EBD 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.

Usage

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

Arguments

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

Value

An auk_ebd object.

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))
```

---

auk_ebd *Reference to EBD 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.
file_sampling	character; optional input sampling event data file, required if you intend to zero-fill the data to produce a presence-absence data set. The sampling 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 EBD to ensure they are in sync.
sep	character; the input field separator, the EBD is 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

The EBD 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 EBD. 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.

Examples

```
# set up reference to sample EBD file
f <- system.file("extdata/ebd-sample.txt", package = "auk")
auk_ebd(f)
# to produce zero-filled data, provide a sampling event data file
f_ebd <- system.file("extdata/zerofill-ex_ebd.txt", package = "auk")
f_smpl <- system.file("extdata/zerofill-ex_sampling.txt", package = "auk")
auk_ebd(f_ebd, file_sampling = f_smpl)
```

---

auk_extent *Filter the EBD by spatial extent*

Description

Define a filter for the eBird Basic Dataset (EBD) based on spatial extent. 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_extent(x, extent)
```

Arguments

x	auk_ebd object; reference to EBD file created by auk_ebd().
extent	numeric; spatial extent expressed as the range of latitudes and longitudes: c(lng_min, lat_min, lng_ma

Value

An auk_ebd object.

Examples

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

---

auk_filter *Filter the EBD 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 EBD.**

Usage

```
auk_filter(x, file, file_sampling, awk_file, sep, filter_sampling, execute,
  overwrite)
```

Arguments

x	auk_ebd object; reference to EBD file created by auk_ebd() with filters defined.
file	character; output file.
file_sampling	character; optional output file for EBD sampling data.
awk_file	character; output file to optionally save the awk script to.
sep	character; the input field separator, the EBD 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 EBD 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 an EBD sampling file is provided in the auk_ebd object, this function will filter both the EBD 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.

Examples

```
# define filters
filters <- system.file("extdata/ebd-sample.txt", package = "auk") %>%
  auk_ebd() %>%
  auk_species(species = c("Gray Jay", "Blue Jay")) %>%
  auk_country(country = c("US", "Canada")) %>%
  auk_extent(extent = c(-100, 37, -80, 52)) %>%
  auk_date(date = c("2012-01-01", "2012-12-31")) %>%
  auk_time(time = c("06:00", "09:00")) %>%
  auk_duration(duration = c(0, 60)) %>%
  auk_complete()
## Not run:
# temp output file
out_file <- tempfile()
auk_filter(filters, file = out_file) %>%
  read_ebd() %>%
  str()
# clean
unlink(out_file)

## End(Not run)
```

auk_getpath *OS specific path to AWK*

Description

Return the OS specific path to AWK, or highlights if it's not installed.

Usage

```
auk_getpath()
```

Value

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

Examples

```
auk_getpath()
```

auk_last_edited *Filter the EBD 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 object; reference to EBD file created by auk_ebd().
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.

Value

An auk_ebd object.

Examples

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

auk_species *Filter the EBD 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, replace)
```

Arguments

`x`	auk_ebd object; reference to EBD file 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).
`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.

Value

An auk_ebd object.

Examples

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

auk_time *Filter the EBD 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, time)
```

Arguments

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

Value

An auk_ebd object.

Examples

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

`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.

`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 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.

Value

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

Examples

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

---

auk_version_date *Dates of EBD and Taxonomy in package version*

Description

This package depends on the version of the EBD and on the eBird taxonomy. Use this function to determine the version dates for which the package is suitable.

Usage

```
auk_version_date()
```

Value

A date vector with the EBD date and taxonomy date that this version of the package corresponds to.

Examples

```
auk_version_date()
```

---

auk_zerofill *Read and zero-fill an EBD file*

Description

Read an eBird Basic Dataset 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, unique = TRUE,
  collapse = FALSE, setclass = c("tbl", "data.frame", "data.table"), ...)

## S3 method for class 'character'
auk_zerofill(x, sampling_events, species, sep = "\t",
  unique = TRUE, collapse = FALSE, setclass = c("tbl", "data.frame",
  "data.table"), ...)

## S3 method for class 'auk_ebd'
auk_zerofill(x, species, sep = "\t", unique = TRUE,
  collapse = FALSE, setclass = c("tbl", "data.frame", "data.table"), ...)

collapse_zerofill(x, setclass = c("tbl", "data.frame", "data.table"))
```

Arguments

x	filename, `data.frame` of EBD 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, don't pass anything to this argument.
unique	logical; should auk_unique() be run on the input data if it hasn't already. The check that auk_unique() has been run is simplistic: is there a checklist_id field or not.
collapse	logical; whether to call zerofill_collapse() to return a data frame rather than an auk_zerofill object.
setclass	tbl, data.frame, or data.table; optionally set additional classes to set on the output data. All return objects are data frames, but may additionally be tbl (for use with dplyr and the tidyverse) or data.table (for use with data.table). The default is to return a tibble.
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.

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().

Examples

```
# read and zero-fill the sampling 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)
```

`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, or common names if `scientific = FALSE`. Un-matched species are returned as `NA`.

Usage

```
ebird_species(x, scientific = TRUE)
```

Arguments

`x`	character; species to look up, provided as scientific or English common names, or a mixture of both. Case insensitive.
`scientific`	logical; whether to return scientific (`TRUE`) or English common names (`FALSE`).

Value

Character vector of scientific names or common names if names if `scientific = FALSE`.

Examples

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

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 six variables and 15,251 rows:

- `category`: whether the entry is for a species or another field-identifiable taxon, such as spuh, `slash`, `hybrid`, etc.
- `species_code`: a unique alphanumeric code identifying each species.
- `name_common`: the common name of the species as used in eBird.
- `name_scientific`: the scientific name of the species.
- `order`: the scientific name of the order that the species belongs to.
- `family`: the family of the species, in the form "Parulidae (New World Warblers)".

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

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, unique, setclass)

## S3 method for class 'character'
read_ebd(x, reader, sep = "\t", unique = TRUE,
  setclass = c("tbl", "data.frame", "data.table"))

## S3 method for class 'auk_ebd'
read_ebd(x, reader, sep = "\t", unique = TRUE,
  setclass = c("tbl", "data.frame", "data.table"))

read_sampling(x, reader, sep, unique, setclass)

## S3 method for class 'character'
read_sampling(x, reader, sep = "\t", unique = TRUE,
  setclass = c("tbl", "data.frame", "data.table"))

## S3 method for class 'auk_ebd'
read_sampling(x, reader, sep = "\t", unique = TRUE,
  setclass = c("tbl", "data.frame", "data.table"))
```

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.
setclass	tbl, data.frame, or data.table; optionally set additional classes to set on the output data. All return objects are data frames, but may additionally be tbl (for use with dplyr and the tidyverse) or data.table (for use with data.table). The default is to return a tibble.

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 with additional class tbl unless setclass is used, in which case a standard data.frame or data.table can be returned. 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().

Examples

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

Index