

Package ‘standardizeSnapshot’

December 20, 2022

Title Standardize and clean Snapshot data

Version 0.1.0

Description standardizeSnapshot is a R package to standardize camera trap records files from the Snapshot Safari project. Snapshot Safari data comes in different standards, following the method that was used to classify pictures (Zooniverse, TrapTagger or Digikam). This package allows to standardize all data sources to a unique file format and then cleans the files to homogenize records.

Depends R (>= 3.4.3)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Roxygen list(markdown = TRUE)

Imports log4r, data.table, chron, lubridate, dplyr, magrittr, stringr, tidyselect, utils

R topics documented:

clean_species	2
create_logger	2
digikam	3
get_camnames	4
get_csv_files_and_folders	5
get_final_filename	6
guess_classifier	6
read_snapshot_files	7
standard	8
standardize_snapshot_df	8
standardize_snapshot_list	9
traptagger	11
write_log_message	11
write_standardized_df	12
write_standardized_list	13
zooniverse	15

clean_species	<i>Standardize species</i>
---------------	----------------------------

Description

Eliminate species duplicate names (things like birdofprey and birdsofprey)

Usage

```
clean_species(species)
```

Arguments

species	vector of species names
---------	-------------------------

Value

the vector of species names with standardized names, according to the species names that have already been encountered in the past datasets.

Examples

```
species <- c("zebraplains", "zebraburchells", "duiker",  
            "duikercommon", "aardvark", "lionfemale")  
clean_species(species)
```

create_logger	<i>Create logger</i>
---------------	----------------------

Description

Initializes a logger with a given logfile.

Usage

```
create_logger(  
  my_logfile,  
  threshold = c("INFO", "DEBUG", "WARNING", "ERROR", "FATAL")  
)
```

Arguments

my_logfile	path to the log file (character).
threshold	logging levels to write to the logger (see loglevel documentation in log4r). Defaults to INFO.

Value

Returns a logger and creates a logfile at the given path. If the path given does not exist, also creates this path.

Examples

```
logger <- create_logger(tempfile())
```

digikam	<i>Digikam sample data</i>
---------	----------------------------

Description

A dataset mimicking typical Digikam data (randomized rows)

Usage

```
digikam
```

Format

A data frame with 100 rows and 22 variables:

X integer Row names (read by R when reading the csv file)

Station character Camera

Species character Species

DateTimeOriginal character Date and time

Date character Date

Time character Time

delta.time.secs integer Time elapsed since this species was last seen at this camera (seconds).
Not relevant because rows were permuted.

delta.time.mins double Time elapsed since this species was last seen at this camera (minutes).
Not relevant because rows were permuted.

delta.time.hours double Time elapsed since this species was last seen at this camera (hours).
Not relevant because rows were permuted.

delta.time.days double Time elapsed since this species was last seen at this camera (days). Not
relevant because rows were permuted.

Directory character Local directory where the original photo is

FileName character Name of the original photo on the local storage

EXIF.Model character Exif info

EXIF.Make character Exif info

metadata_Species character Species (other column)

metadata_Number character Species count

metadata_Behaviour character Tagged behaviors
 metadata_Sex character Tagged sex
 n_images integer Number of pictures associated to this event?
 metadata_young_present character Tagged youngs (Yes/No)
 metadata_Numberofindividuals logical Tagged number of individuals on the picture
 HierarchicalSubject character Summary column for all metadata_...

get_camnames	<i>Get camera names</i>
--------------	-------------------------

Description

Subsets the locations IDs from the camera names vector.

Usage

```
get_camnames(cameras, locations, silence_warning = FALSE)
```

Arguments

cameras a character vector of camera names
 locations a character vector of locations (must be the same length as cameras)
 silence_warning
 print a warning if some locations are NA?

Value

A vector of camera names without the location prefix (if it was present)

Examples

```
get_camnames(c("APN_A01", "MAD_B01"), c("APN", "MAD"))
```

`get_csv_files_and_folders`*Get csv files and folder*

Description

Get the names of all csv files from the input character vector.

Usage

```
get_csv_files_and_folders(input, except, basepath)
```

Arguments

<code>input</code>	a character vector of valid paths: can be files and/or folders
<code>except</code>	files to ignore (optional): all paths for (a part of) which match the exact expression in <code>except</code> will be ignored.
<code>basepath</code>	the part of the path that should be ignored when copying final files (i.e. absolute path inside one's computer that should not be copied in final file.)

Details

`input` can be a vector of files and/or folders. If an element of `input` is a folder, the function will list all files within `input`; if the element is a file, the function will only list this file.

Value

A dataframe with columns `folders` and `files`, where `folders` are the paths up to a given file (corresponding to the relative path to the file from `basepath`), and `files` are the files paths from `folders`.

Examples

```
## Not run:
get_csv_files_and_folders(input = c("path/to/datafolder/KGA",
                                     "path/to/datafolder/ATH_Roll1_Snapshot.csv"),
                          except = "KGA/KGA-KHO_together",
                          basepath = "path/to/datafolder")

## End(Not run)
```

get_final_filename	<i>Get final filemane</i>
--------------------	---------------------------

Description

Return the filename from the file columns.

Usage

```
get_final_filename(df)
```

Arguments

df	The dataframe to be copied. Must have columns locationID, season, roll.
----	---

Value

The filename for this file in the format locationID_Sseason_Rroll.csv It there are several values in locationID, season or roll, they are separated by a dash in the filename: locationID1-locationID2...

Examples

```
zooniverse_std <- standardize_snapshot_df(df = zooniverse, standard = standard)
get_final_filename(zooniverse_std)
```

guess_classifier	<i>Guess classifier</i>
------------------	-------------------------

Description

Guesses the classifier used to annotate the data based on the column names given in colnames_df.

Usage

```
guess_classifier(colnames_df)
```

Arguments

colnames_df	A character vector of column names.
-------------	-------------------------------------

Value

The classifier: either zooniverse, digikam or traptagger.

Examples

```
guess_classifier(colnames(zooniverse))
guess_classifier(colnames(traptagger))
guess_classifier(colnames(digikam))
```

read_snapshot_files	<i>Read Snapshot files</i>
---------------------	----------------------------

Description

Reads files from a vector of files and/or folders into a list of dataframes.

Usage

```
read_snapshot_files(input, except, basepath)
```

Arguments

input	a character vector of valid paths: can be files and/or folders
except	files to ignore (optional): all paths for (a part of) which match the exact expression in except will be ignored.
basepath	the part of the path that should be ignored when copying final files (i.e. absolute path inside one's computer that should not be copied in final file.)

Details

The files are assumed to be comma or semicolon-separated CSV. If the filename is MAD_S2_full_report_0-50__agreement_corrected_fin.csv, the function only reads the first 26 columns as this file has empty columns.

Value

A named list of dataframes. Each element of the list is a dataframe with the contents of a file read from the files list given in input. The names of the list are the file names from the root of input: If the element from input is a file, it is a filename. If the element from input is a folder, it is the relative path from basepath to the file.

Examples

```
## Not run:
read_snapshot_files("path/to/datafolder/DHP",
                    basepath = "path/to/datafolder",
                    except = "DHP/DHP+OVE_same_file/*")

## End(Not run)
```

standard	<i>Standard column names</i>
----------	------------------------------

Description

A dataframe listing the standard column names for the different Snapshot data formats.

Usage

standard

Format

A data frame with 39 rows and 4 variables:

zooniverse	character	Expected columns names for Zooniverse data
traptagger	character	Expected columns names for TrapTagger data
digikam	character	Expected columns names for Digikam data
new	character	Columns names for the output standardized data

standardize_snapshot_df	<i>Standardize dataframe</i>
-------------------------	------------------------------

Description

Standardizes a dataframe to the Snapshot standard.

Usage

standardize_snapshot_df(df, standard_df, locationID_digikam, classifier)

Arguments

- | | |
|--------------------|--|
| df | The dataframe to standardize. It is expected to match the data format for either Zooniverse, TrapTagger or Digikam processed data (i.e. have column names defined in standard_df). |
| standard_df | The standard dataframe to match column names to the new standard. A dataframe with >= 2 columns, one of which must be named zooniverse, digikam or traptagger and another one must be named new. |
| locationID_digikam | Optional character locationID to use for Digikam data (will display a warning if not provided for Digikam data.) Indeed, for Digikam data, the locationID cannot be inferred from other columns. |
| classifier | Optional character for the classifier. |

Details

Dates and times are standardized to YYYY-MM-DD and HH:MM:SS. The columns locationID, cameraID, roll, capture and season (if classifier is zooniverse) are filled with meaningful information. Species names (snapshotName) are standardized to match the existing, known species names. For column locationID:

- The location code DHP is replaced with OVE if the corresponding camera code starts with 'O'.
- The location code KGA is replaced with KHO if the corresponding camera code starts with 'KHO'.

For column cameraID:

- For TrapTagger data: will remove the leading location code part for all data (eg if location is ATH, will change cameras ATH_A01 -> A01). Also, if the location code is KHO, SAM or TSW: will remove the dash in the camera name (e.g KHO_E_A01 -> EA01)
- For Zooniverse data: if the location code is KHO, will replace KHOG with E and KHOL with M in cameras. If the location code is DHP, will remove leading D in cameraID. If the location code is OVE, will remove leading O in cameraID.
- For column eventID: the event ID formatted as season#cam_site#roll#event_no.

Value

The standardized dataframe: it has the same columns as specified in standard_df\$new, dates and times are standardized and some columns regarding information on the capture are filled. Species names (snapshotName) and cameraID and locationID are standardized (see details). Columns are in the same order as provided in standard_df\$new and the rows are ordered by ascending camera, date and time.

Examples

```
standardize_snapshot_df(zooniverse, standard)
standardize_snapshot_df(traptagger, standard)
standardize_snapshot_df(digikam, standard, locationID_digikam = "MOK")
```

```
standardize_snapshot_list
```

Standardize a list of dataframes

Description

Standardizes a list of dataframes to the Snapshot standard.

Usage

```
standardize_snapshot_list(df_list, standard_df, classifier)
```

Arguments

<code>df_list</code>	a list of dataframes.
<code>standard_df</code>	The standard dataframe to match column names to the new standard. A dataframe with ≥ 2 columns, one of which must be named <code>zooniverse</code> , <code>digikam</code> or <code>traptagger</code> and another one must be named <code>new</code> .
<code>classifier</code>	Optional character or vector of characters for the classifier.

Details

Dates and times are standardized to YYYY-MM-DD and HH:MM:SS. The columns `locationID`, `cameraID`, `roll`, `capture` and `season` (if classifier is `zooniverse`) are filled with meaningful information. Species names (`snapshotName`) are standardized to match the existing, known species names. For column `locationID`:

- The location code DHP is replaced with OVE if the corresponding camera code starts with 'O'.
- The location code KGA is replaced with KHO if the corresponding camera code starts with 'KHO'.

For column `cameraID`:

- For TrapTagger data: will remove the leading location code part for all data (eg if location is ATH, will change cameras ATH_A01 -> A01). Also, if the location code is KHO, SAM or TSW: will remove the dash in the camera name (e.g KHO_E_A01 -> EA01)
- For Zooniverse data: if the location code is KHO, will replace KHOG with E and KHOL with M in cameras. If the location code is DHP, will remove leading D in cameraID. If the location code is OVE, will remove leading O in cameraID.
- For column `eventID`: the event ID formatted as `season#cam_site#roll#event_no`.

Value

The list of standardized dataframes: they the same columns as specified in `standard_df$new`, dates and times are standardized and some columns regarding information on the capture are filled. Species names (`snapshotName`) and `cameraID` and `locationID` are standardized (see details). Columns are in the same order as provided in `standard_df$new` and the rows are ordered by ascending camera, date and time.

Note

If the file name is `Marinmane NR _record_table_0min_deltaT_2021-06-10.csv`, will apply the location code MAR.

Examples

```
df_list <- list(zooniverse, digikam, traptagger)

# Digikam data must be named with at least the locationID code
names(df_list)[2] <- "MOK"
standardize_snapshot_list(df_list, standard)
```

```
# The name for Digikam data can also a filename starting with the locationID code
names(df_list)[2] <- "MOK_record_table_0min_deltaT_2021-05-07.csv"
standardize_snapshot_list(df_list, standard)
```

traptagger	<i>TrapTagger sample data</i>
------------	-------------------------------

Description

A dataset mimicking typical TrapTagger data (randomized rows)

Usage

```
traptagger
```

Format

A data frame with 100 rows and 9 variables:

Capture_ID character ID for the capture composed of locationcapture#roll#Cam.Site

Cam.Site character Camera

id integer Picture ID

latitude double Camera latitude (mock data)

longitude double Camera longitude (mock data)

timestamp character Date time

capture_labels character Species seen on the picture

capture_sighting_count integer Species count

capture_url character Picture url (mock data)

write_log_message	<i>Write log message</i>
-------------------	--------------------------

Description

Writes a log message. If a logger is provided, writes to that logger; if it is NA, displays a message.

Usage

```
write_log_message(message, logger = NA, level = "info")
```

Arguments

message	The message to display/write to the logger
logger	Logger to write to (log4r object of class logger) (defaults to NA)
level	Logging level: either info, warn, debug or error.

Details

- if logger is not NA, writes a message to the logger file with the specified level.
- if logger is NA, the output depends on level. If level is info returns a message(), if it is warn returns a warning(), if error returns an error() and if debug returns a print.

Value

Will either write the message specified in message to the logger or returns a print, message, warning or stop depending on level (see details).

Examples

```
# With no logger
write_log_message("Test", level = "debug")
write_log_message("Test", level = "info")
## Not run:
write_log_message("Test", level = "warn") # Returns a warning
write_log_message("Test", level = "error") # Returns an error

## End(Not run)
# With a logger
logger <- create_logger(tempfile()) # Create a logger
write_log_message("Test", level = "debug", logger = logger)
write_log_message("Test", level = "info", logger = logger)
## Not run:
write_log_message("Test", level = "warn",
                  logger = logger) # Writes the log message and returns a warning
write_log_message("Test", level = "error",
                  logger = logger) # Writes the log message and returns an error

## End(Not run)
```

write_standardized_df *Write the standardized file*

Description

Writes a file to a given location. If to does not exist, it is created, and if filename is not provided, a default standardized name is chosen.

Usage

```
write_standardized_df(
  df,
  to,
  filename,
  write = TRUE,
  return_path = ifelse(write, FALSE, TRUE),
  verbose = TRUE
)
```

Arguments

df	The standardized file
to	The target folder to copy data in. If it does not exist, will be created.
filename	The name to give to the file.
write	if TRUE (default), will write the df in the to folder, and df will be named filename.
return_path	Should the path be returned?
verbose	Should messages be displayed when creating a folder/file?

Value

Writes the file to the folder to/filename. Also returns the path to/filename if return_path == TRUE.

Examples

```
std_dat <- standardize_snapshot_df(zooniverse, standard)

# Don't write data
write_standardized_df(std_dat,
                      to = "/home/lnicvert/Documents/PhD/Snapshot/data/2_standardized_data",
                      write = FALSE)
# Don't write data and use custom name
write_standardized_df(std_dat,
                      to = "/home/lnicvert/Documents/PhD/Snapshot/data/2_standardized_data",
                      filename = "myname.csv",
                      write = FALSE)
# Write file to temporary location
write_standardized_df(std_dat,
                      to = tempdir())
```

```
write_standardized_list
```

Write the standardized files

Description

Writes a list of files to a given location. If to does not exist, it is created, and if filename is not provided, a default standardized name is chosen.

Usage

```
write_standardized_list(
  df_list,
  filenames,
  to,
```

```

    write = TRUE,
    return_path = ifelse(write, FALSE, TRUE),
    verbose = TRUE
  )

```

Arguments

<code>df_list</code>	The standardized files list to write. If the list is named, the subdirectory structure in the names will be used to replicate the subdirectory structure in the destination.
<code>filenames</code>	(Optional) vector of customized file names.
<code>to</code>	Destination folder to write to.
<code>write</code>	Should the result be written or only the path returned?
<code>return_path</code>	Should the path be returned?
<code>verbose</code>	Should messages be displayed when creating a folder/file?

Value

Writes the files to/`filename1`, to/`filename2`... Also returns the paths to/`filename1`, to/`filename2`... if `return_path == TRUE`.

Examples

```

# Example with a subdirectory structure (inferred from the filenames)
df_list <- list(zooniverse, digikam, traptagger)
names(df_list) <- c("APN/APN.csv", "MOK/MOK.csv", "ATH/ATH.csv")
std_list <- standardize_snapshot_list(df_list, standard)

# Don't write data
write_standardized_list(std_list,
  to = "/home/lnicvert/Documents/PhD/Snapshot/data/2_standardized_data",
  write = FALSE)
# Don't write data and use custom name
write_standardized_list(std_list,
  to = "/home/lnicvert/Documents/PhD/Snapshot/data/2_standardized_data",
  filenames = c("myname1.csv", "myname2.csv", "myname3.csv"),
  write = FALSE)
# Write files to temporary location
write_standardized_list(std_list, to = tempdir())

# Without a subdirectory structure (and without list names)
df_list <- list(zooniverse, digikam, traptagger)
names(df_list)[2] <- "MOK"
std_list <- standardize_snapshot_list(df_list, standard)

# Write files to temporary location
write_standardized_list(std_list, to = tempdir())

```

zooniverse	<i>Zooniverse sample data</i>
------------	-------------------------------

Description

A dataset mimicking typical Zooniverse data (randomized rows)

Usage

```
zooniverse
```

Format

A data frame with 100 rows and 24 variables:

```
capture_id character ID for the capture composed of season#site#roll#capture
season character Zooniverse season code
site character Camera ID
roll integer Roll (index for the camera service)
capture integer ID for the capture per camera/season/roll
capture_date_local character Date
capture_time_local character Time
zooniverse_url_0 character Url for first photo (mock URL)
zooniverse_url_1 character Url for second photo (mock URL)
zooniverse_url_2 character Url for third photo (mock URL)
subject_id integer Internal Zooniverse ID for the capture event (mock ID)
question__species character Species
question__count_max character Maximum count from the volunteers
question__count_median character Median count from the volunteers
question__count_min character Minimum count from the volunteers
question__standing double Proportion of users who declared a standing behavior
question__resting double Proportion of users who declared a resting behavior
question__moving double Proportion of users who declared a moving behaviour
question__eating double Proportion of users who declared a eating behaviour
question__interacting double Proportion of users who declared an interacting behaviour
question__young_present double Proportion of users who declared young presents
question__horns_visible double Proportion of users who declared horns on the picture
p_users_identified_this_species double Proportion of users who identified the consensus
species
pielous_evenness_index double Pielou evenness index
```