Package ‘camtrapR’

July 29, 2020

Type Package

Title Camera Trap Data Management and Preparation of Occupancy and Spatial Capture-Recapture Analyses

Version 2.0.3

Date 2020-07-29

Depends R (>= 3.1.0)

Imports methods, sp, overlap, secr, data.table

Suggests knitr, lubridate, raster, rgdal, ritis, rmarkdown, taxize, testthat, tibble, unmarked, zip, RSQLite

VignetteBuilder knitr

SystemRequirements ExifTool
  (http://www.sno.phy.queensu.ca/~phil/exiftool/)

Description Management of and data extraction from camera trap data in wildlife studies. The package provides a workflow for storing and sorting camera trap photos (and videos), tabulates records of species and individuals, and creates detection/non-detection matrices for occupancy and spatial capture-recapture analyses with great flexibility. In addition, it can visualise species activity data and provides simple mapping functions with GIS export.

URL https://github.com/jniedballa/camtrapR,
    https://jniedballa.github.io/camtrapR,
    https://groups.google.com/forum/#!forum/camtrapr

BugReports https://groups.google.com/forum/#!forum/camtrapr

Encoding UTF-8

License GPL (>= 2)

NeedsCompilation no

Author Juergen Niedballa [aut, cre] (<https://orcid.org/0000-0002-9187-2116>), Alexander Courtiol [aut] (<https://orcid.org/0000-0003-0637-2959>), Rahel Sollmann [aut] (<https://orcid.org/0000-0002-1607-2039>), John Mathai [ctb], Seth Timothy Wong [ctb], An The Truong Nguyen [ctb],
Azlan bin Mohamed [ctb] (<https://orcid.org/0000-0003-3788-4383>),
Andrew Tilker [ctb] (<https://orcid.org/0000-0003-3630-8691>),
Andreas Wilting [ctb, ths] (<https://orcid.org/0000-0001-5073-9186>)

Maintainer Juergen Niedballa <camtrapr@gmail.com>

Repository CRAN

Date/Publication 2020-07-29 12:20:02 UTC

R topics documented:

- camtrapR-package ........................................... 3
- activityDensity ............................................. 6
- activityHistogram .......................................... 8
- activityOverlap ............................................ 10
- activityRadial ............................................. 12
- addCopyrightTag ........................................... 15
- appendSpeciesNames ....................................... 16
- cameraOperation .......................................... 18
- camtraps .................................................... 21
- camtrapsMultiSeason ...................................... 22
- checkSpeciesIdentification ............................... 24
- checkSpeciesNames ......................................... 26
- createSpeciesFolders ..................................... 28
- createStationFolders ..................................... 30
- detectionHistory .......................................... 31
- detectionMaps ............................................. 36
- exifTagNames ............................................... 39
- exiftoolPath ............................................... 41
- fixDateTimeOriginal ...................................... 42
- getSpeciesImages .......................................... 43
- imageRename ............................................... 46
- recordTable ............................................... 48
- recordTableIndividual .................................... 54
- recordTableIndividualSample ............................ 59
- recordTableIndividualSampleMultiSeason ............... 60
- recordTableSample ......................................... 61
- recordTableSampleMultiSeason ........................... 62
- spatialDetectionHistory .................................. 63
- surveyReport .............................................. 68
- timeShiftImages .......................................... 71
- timeShiftTable ............................................ 74

Index 75
Description

This package provides a streamlined workflow for processing data generated in camera trap-based wildlife studies and prepares input for further analyses, particularly in occupancy and spatial capture-recapture frameworks. It suggests a simple data structure and provides functions for managing digital camera trap photographs (and videos), generating record tables, maps of species richness and species detections and species activity diagrams. It further helps prepare subsequent analyses by creating detection/non-detection matrices for occupancy analyses, e.g. in the unmarked package, and capthist objects for spatial capture-recapture analyses in the secr package. In addition, basic survey statistics are computed. The functions build on one another in a logical sequence. The only manual input needed is species (and individual) identification, which is achieved by moving images into species directories or by tagging images in image management software. Besides, a table holding basic information about camera trap station IDs, locations and trapping periods must be created in spreadsheet software.

Details

Image metadata (such as date and time or user-assigned tags) are extracted from the images using Phil Harvey’s ExifTool (available from https://exiftool.org/) and the information is stored in a record table. An adjustable criterion for temporal independence of records can be applied. Maps of species presence and species richness can be generated. Several functions are available for plotting single- and two-species activity patterns. Information about the camera-specific trapping periods (and periods of malfunction) are summarized into a matrix about camera trap operability. These, together with the record table, are used to generate species detection histories for occupancy and spatial capture-recapture analyses. The user has considerable freedom in generating the detection histories; sampling occasion length, beginning date and and occasion start times are adjustable. In addition, trapping effort (i.e. active trap nights per station and occasion) can be computed for use as a covariate/offset on detection probability.

User support

The camtrapR Google group is an online support and help forum for camtrapR users. You can find it here: https://groups.google.com/forum/#!forum/camtrapr.

Image organisation and management

The functions in this section set up a directory structure for storing camera trap images and identifying species and individuals from images. They build on one another and can be run in sequential order as needed.

- **createStationFolders**: Create camera trap station directories for raw images
- **fixDateTimeOriginal**: Fix DateTimeOriginal Exif metadata tag in Reconyx Hyperfire cameras
- **timeShiftImages**: Apply time shifts to JPEG images
- **imageRename**: Copy and rename images based on station ID and image creation date
Species / individual identification

These functions assist in species identification and prepare individual identification of animals.

- `checkSpeciesNames`: Check species names against the ITIS taxonomic database
- `createSpeciesFolders`: Create directories for species identification
- `checkSpeciesIdentification`: Consistency check on species image identification
- `getSpeciesImages`: Gather all images of a species in a new directory

Image data extraction

These functions use the directory structure built above (Section ‘Image management workflow’) and a table containing basic information about camera traps and/or stations (IDs, location, trapping period).

- `recordTable`: Create a species record table from camera trap images and videos
- `recordTableIndividual`: Create a single-species record table from camera trap images and videos with individual IDs
- `exifTagNames`: Return Exif metadata tags and tag names from JPEG images
- `exiftoolPath`: Add the directory containing exiftool.exe to PATH temporarily (Windows only)

Data exploration and visualisation

These plots are generated from the record table and the camera trap table.

- `detectionMaps`: Generate maps of species richness and species presence by station, export shapefiles
- `activityHistogram`: Single-species diel activity histograms
- `activityDensity`: Single-species diel activity kernel density estimation plots
- `activityRadial`: Single-species diel activity radial plot
- `activityOverlap`: Two-species diel activity overlap plots and estimates

Data export

- `cameraOperation`: Create a camera operability matrix
- `detectionHistory`: Species detection histories for occupancy analyses (single and multi-season)
- `spatialDetectionHistory`: Detection histories of individuals for spatial capture-recapture analyses
- `surveyReport`: Create a report about camera trap surveys and species detections

---

**addCopyrightTag**

Write a copyright tag into JPEG image metadata

**appendSpeciesNames**

Add or remove species names from image filenames
**Sample data**

- `camtraps` Sample camera trap station information table
- `recordTableSample` Sample species record table
- `recordTableIndividualSample` Single-species record table with individual IDs
- `camtrapsMultiSeason` Sample multi season camera trap station information table
- `recordTableSampleMultiSeason` Sample multi season species record table
- `recordTableIndividualSampleMultiSeason` Single-species multi season record table with individual IDs
- `timeShiftTable` Sample camera trap time shift information

**Vignettes**

1. Organising raw camera trap images
2. Identifying species and individuals
3. Extracting Data from Camera Trapping Images and Videos
4. Data exploration and visualisation

**Author(s)**

Juergen Niedballa
Maintainer: Juergen Niedballa <camtrapr@gmail.com>

**References**


CamtrapR Google Group [https://groups.google.com/forum/#!forum/camtrapr](https://groups.google.com/forum/#!forum/camtrapr)


Mike Meredith and Martin Ridout (2018). overlap: Estimates of coefficient of overlapping for animal activity patterns. R package version 0.3.2. [https://CRAN.R-project.org/package=overlap](https://CRAN.R-project.org/package=overlap)

Phil Harvey’s ExifTool [https://exiftool.org/](https://exiftool.org/)
See Also

overlap unmarked secr plotrix taxize ritis wiqid

---

**Description**

The function plots a kernel density estimation of species diel activity using function `densityPlot` from package `overlap`.

**Usage**

```r
activityDensity(recordTable, species, allSpecies = FALSE, speciesCol = "Species", recordDateTimeCol = "DateTimeOriginal", recordDateTimeFormat = "%Y-%m-%d %H:%M:%S", plotR = TRUE, writePNG = FALSE, plotDirectory, createDir = FALSE, pngMaxPix = 1000, add.rug = TRUE,
...)
```

**Arguments**

- `recordTable` data.frame. the record table created by `recordTable`
- `species` Name of the species for which to create an kernel density plot of activity
- `allSpecies` logical. Create plots for all species in `speciesCol` of `recordTable`? Overrides argument `species`
- `speciesCol` character. name of the column specifying species names in `recordTable`
- `recordDateTimeCol` character. name of the column specifying date and time in `recordTable`
- `recordDateTimeFormat` character. format of column `recordDateTimeCol` in `recordTable`
- `plotR` logical. Show plots in R graphics device?
- `writePNG` logical. Create pngs of the plots?
- `plotDirectory` character. Directory in which to create png plots if `writePNG = TRUE`
- `createDir` logical. Create `plotDirectory` if `writePNG = TRUE`?
- `pngMaxPix` integer. image size of png (pixels along x-axis)
- `add.rug` logical. add a rug to the plot?
- `...` additional arguments to be passed to function `densityPlot`
activityDensity

Details

species must be in the speciesCol of recordTable.

recordDateTimeFormat defaults to the "YYYY-MM-DD HH:MM:SS" convention, e.g. "2014-09-30 22:59:59". recordDateTimeFormat can be interpreted either by base-R via strptime or in lubridate via parse_date_time (argument "orders"). lubridate will be used if there are no "%" characters in recordDateTimeFormat.

For "YYYY-MM-DD HH:MM:SS", recordDateTimeFormat would be either "%Y-%m-%d %H:%M:%S" or "ymd HMS". For details on how to specify date and time formats in R see strptime or parse_date_time.

Value

Returns invisibly a vector of species record observation times in radians, i.e. scaled to \([0, 2\pi]\). If allSpecies == TRUE, all species' vectors are returned in an invisible named list.

Author(s)

Juergen Niedballa

References


See Also

activityHistogram, activityRadial, activityOverlap http://www.kent.ac.uk/smsas/personal/msr/overlap.html

Examples

# load record table
data(recordTableSample)

species4activity <- "VTA" # = Viverra tangalunga, Malay Civet

activityDensity(recordTable = recordTableSample,
species = species4activity)

# all species at once

activityDensity(recordTable = recordTableSample,
allSpecies = TRUE,
writePNG = FALSE,
plotR = TRUE,
add.rug = TRUE)
activityHistogram  

Plot histogram of single-species activity

Description

The function generates a histogram of species diel activity in 1-hour intervals.

Usage

activityHistogram(recordTable, 
species, 
allSpecies = FALSE, 
speciesCol = "Species", 
recordDateTimeCol = "DateTimeOriginal", 
recordDateTimeFormat = "%Y-%m-%d %H:%M:%S", 
plotR = TRUE, 
writePNG = FALSE, 
plotDirectory, 
createDir = FALSE, 
pngMaxPix = 1000, 
...)

Arguments

recordTable  data.frame. the record table created by recordTable
species  Name of the single species for which to create a histogram of activity
allSpecies  logical. Create plots for all species in speciesCol of recordTable? Overrides argument species
speciesCol  character. name of the column specifying species names in recordTable
recordDateTimeCol  character. name of the column specifying date and time in recordTable
recordDateTimeFormat  character. format of column recordDateTimeCol in recordTable
plotR  logical. Show plots in R graphics device?
writePNG  logical. Create pngs of the plots?
plotDirectory  character. Directory in which to create png plots if writePNG = TRUE
createDir  logical. Create plotDirectory?
pngMaxPix  integer. image size of png (pixels along x-axis)
...  additional arguments to be passed to function hist
activityHistogram

Details

Activity is calculated from the time of day of records. The date is ignored.

recordDateTimeFormat defaults to the "YYYY-MM-DD HH:MM:SS" convention, e.g. "2014-09-30 22:59:59". recordDateTimeFormat can be interpreted either by base-R via strptime or in lubridate via parse_date_time (argument "orders"). lubridate will be used if there are no "%" characters in recordDateTimeFormat.

For "YYYY-MM-DD HH:MM:SS", recordDateTimeFormat would be either "%Y-%m-%d %H:%M:%S" or "ymd HMS". For details on how to specify date and time formats in R see strptime or parse_date_time.

Value

It returns invisibly a vector of species record date and time in POSIXlt format. If allSpecies == TRUE, all species’ vectors are returned in an invisible named list.

Note

If you have a sufficiently large number of records you may wish to consider using activityDensity instead. Please be aware that this function (like the other activity... function of this package) use clock time. If your survey was long enough to see changes in sunrise and sunset times, this may result in biased representations of species activity.

Author(s)

Juergen Niedballa

See Also

activityDensity, activityRadial, activityOverlap

Examples

# load record table
data(recordTableSample)

# generate activity histogram
species4activity <- "VTA"  # = Viverra tangalunga, Malay Civet

activityHistogram (recordTable = recordTableSample,
                  species = species4activity,
                  allSpecies = FALSE)
activityOverlap

Plot overlapping kernel densities of two-species activities

Description
This function plots kernel density estimates of two species’ diel activity data by calling the function overlapPlot from package overlap. It further computes the overlap coefficient Dhat1 by calling overlapEst.

Usage
activityOverlap(recordTable, speciesA, speciesB, speciesCol = "Species", recordDateTimeCol = "DateTimeOriginal", recordDateTimeFormat = "%Y-%m-%d %H:%M:%S", plotR = TRUE, writePNG = FALSE, addLegend = TRUE, legendPosition = "topleft", plotDirectory, createDir = FALSE, pngMaxPix = 1000, add.rug = TRUE, overlapEstimator = c("Dhat1", "Dhat4", "Dhat5"), ...
)

Arguments
recordTable data.frame. the record table created by recordTable
speciesA Name of species 1 (as found in speciesCol of recordTable)
speciesB Name of species 2 (as found in speciesCol of recordTable)
speciesCol character. name of the column specifying species names in recordTable
recordDateTimeCol character. name of the column specifying date and time in recordTable
recordDateTimeFormat character. format of column recordDateTimeCol in recordTable
plotR logical. Show plots in R graphics device?
writePNG logical. Create pngs of the plots?
addLegend logical. Add a legend to the plots?
legendPosition character. Position of the legend (keyword)
plotDirectory character. Directory in which to create png plots if writePNG = TRUE
activityOverlap

createDir logical. Create plotDirectory?
pngMaxPix integer. image size of png (pixels along x-axis)
add.rug logical. add a rug to the plot?
overlapEstimator character. Which overlap estimator to return (passed on to argument type in overlapEst)

... additional arguments to be passed to function overlapPlot

Details

... can be graphical parameters passed on to function overlapPlot, e.g. linetype, linewidth, linecol (see example below).

recordDateTimeFormat defaults to the "YYYY-MM-DD HH:MM:SS" convention, e.g. "2014-09-30 22:59:59". recordDateTimeFormat can be interpreted either by base-R via strptime or in lubridate via parse_date_time (argument "orders"). lubridate will be used if there are no "%" characters in recordDateTimeFormat.

For "YYYY-MM-DD HH:MM:SS", recordDateTimeFormat would be either "%Y-%m-%d %H:%M:%S" or "ymd HMS". For details on how to specify date and time formats in R see strptime or parse_date_time.

Value

Returns invisibly the data.frame with plot coordinates returned by overlapPlot.

Note

Please be aware that the function (like the other activity... function of this package) use clock time, not solar time. If your survey was long enough to see changes in sunrise and sunset times, this may result in biased representations of species activity.

Author(s)

Juergen Niedballa

References


See Also

activityDensity
http://www.kent.ac.uk/smsas/personal/msr/overlap.html
Examples

```r
# load record table
data(recordTableSample)

# define species of interest
speciesA_for_activity <- "VTA"  # = Viverra tangalunga, Malay Civet
speciesB_for_activity <- "PBE"  # = Prionailurus bengalensis, Leopard Cat

# create activity overlap plot (basic)
activityOverlap (recordTable = recordTableSample,
                speciesA = "VTA",  # = Viverra tangalunga, Malay Civet
                speciesB = "PBE",  # = Prionailurus bengalensis, Leopard Cat
                writePNG = FALSE,
                plotR = TRUE)

# create activity overlap plot (prettier and with some overlapPlot arguments set)
activityOverlap (recordTable = recordTableSample,
                speciesA = speciesA_for_activity,
                speciesB = speciesB_for_activity,
                writePNG = FALSE,
                plotR = TRUE,
                createDir = FALSE,
                pngMaxPix = 1000,
                linecol = c("black", "blue"),
                linewidth = c(5,3),
                linetype = c(1, 2),
                olapcol = "darkgrey",
                add.rug = TRUE,
                extend = "lightgrey",
                ylim = c(0, 0.25),
                main = paste("Activity overlap between ",
                            speciesA_for_activity, " and",
                            speciesB_for_activity))
```

---

**activityRadial**  
*Radial plots of single-species activity*

**Description**

The function generates a radial plot of species diel activity using an adapted version of function `radial.plot` from package `plotrix` (without the need to install the package). Records are aggregated by hour. The number of independent events is used as input, which in turn is based on the argument `minDeltaTime` in `recordTable`. 
activityRadial

Usage

```
activityRadial(recordTable, 
species, 
allSpecies = FALSE, 
speciesCol = "Species", 
recordDateTimeCol = "DateTimeOriginal", 
recordDateTimeFormat = "%Y-%m-%d %H:%M:%S", 
byNumber = FALSE, 
plotR = TRUE, 
writePNG = FALSE, 
plotDirectory, 
createDir = FALSE, 
pngMaxPix = 1000,
... 
)
```

Arguments

- `recordTable` data.frame. the record table created by `recordTable`
- `species` Name of the species for which to create an kernel density plot of activity
- `allSpecies` logical. Create plots for all species in `speciesCol` of `recordTable`? Overrides argument `species`
- `speciesCol` character. name of the column specifying species names in `recordTable`
- `recordDateTimeCol` character. name of the column specifying date and time in `recordTable`
- `recordDateTimeFormat` character. format of column `recordDateTimeCol` in `recordTable`
- `byNumber` logical. If FALSE, plot proportion of records. If TRUE, plot number of records
- `plotR` logical. Show plots in R graphics device?
- `writePNG` logical. Create pngs of the plots?
- `plotDirectory` character. Directory in which to create png plots if `writePNG = TRUE`
- `createDir` logical. Create `plotDirectory`?
- `pngMaxPix` integer. image size of png (pixels along x-axis)
- `...` additional arguments to be passed to function `radial.plot`

Details

`radial.plot` was adjusted to show a clockwise 24-hour clock face. It is recommended to set argument `lwd` to a value >= 2. You may also wish to add argument `rp.type="p"` to show a polygon instead of bars.

`recordDateTimeFormat` defaults to the "YYYY-MM-DD HH:MM:SS" convention, e.g. "2014-09-30 22:59:59". `recordDateTimeFormat` can be interpreted either by base-R via `strptime` or in `lubridate` via `parse_date_time` (argument "orders"). `lubridate` will be used if there are no "%" characters in `recordDateTimeFormat`.
For "YYYY-MM-DD HH:MM:SS", `recordDateTimeFormat` would be either "\%Y-\%m-\%d \%H:\%M:\%S" or "ymd HMS". For details on how to specify date and time formats in R see `strptime` or `parse_date_time`.

**Value**

Returns invisibly a data.frame containing all information needed to create the plot: radial position, lengths, hour (for labels). If `allSpecies == TRUE`, all species’ data frames are returned in an invisible named list.

**Author(s)**

Juergen Niedballa

**References**

https://CRAN.R-project.org/package=plotrix

**See Also**

`activityDensity`, `activityHistogram`, `activityOverlap`

**Examples**

```r
# load record table
data(recordTableSample)

species4activity <- "PBE" # = Prionailurus bengalensis, Leopard Cat

activityRadial(recordTable = recordTableSample,  
                species = species4activity,  
                allSpecies = FALSE,  
                speciesCol = "Species",  
                recordDateTimeCol = "DateTimeOriginal",  
                plotR = TRUE,  
                writePNG = FALSE,  
                lwd = 5)

# plot type = polygon

activityRadial(recordTable = recordTableSample,  
                species = species4activity,  
                allSpecies = FALSE,  
                speciesCol = "Species",  
                recordDateTimeCol = "DateTimeOriginal",  
                plotR = TRUE,  
                writePNG = FALSE,  
                lwd = 5,  
                rp.type = "p"
```

addCopyrightTag

Write a copyright tag into JPEG image metadata

Description

This function writes a copyright tag into the copyright field of JPEG image Exif metadata. It does so recursively, so it works both for images that are sorted into subdirectories and unsorted images. Note that all images in subdirectories of inDir will be tagged. It is not required to run this function in the camtrapR workflow, but may be desired for data sharing or publishing.

Usage

addCopyrightTag(inDir, copyrightTag, askFirst = TRUE, keepJPG_original = TRUE)

Arguments

inDir character. Name of the directory containing camera trap images.
copyrightTag character. The tag to be written into the Exif Copyright field
askFirst logical. Ask user to confirm before execution?
keepJPG_original logical. Keep original JPG files as .JPG_original files (TRUE) or overwrite JPGs (FALSE)?

Details

If askFirst = TRUE, the function will show a menu and asks the user to confirm the action before execution. Type “1” to write copyright tags and “2” to abort.

By default Exiftool creates a copy of each JPG image and preserves the original images (without the copyright tag) as .JPG_original files. Note that this behaviour will instantly double the number of images in inDir and the disk space required. If this is not desired, set keepJPG_original = FALSE.

Value

An invisible list of Exiftool output.

More importantly, the specified copyright tag is written into the Copyright field of the Exif metadata of all images in inDir.

Author(s)

Juergen Niedballa
appendSpeciesNames

Add or remove species names from JPEG image filenames

Description

Add or remove species names from JPEG image filenames. It makes it easier to find images of a species.

Usage

appendSpeciesNames(inDir,
IDfrom,
hasCameraFolders,
metadataSpeciesTag,
metadataHierarchyDelimiter = "|",
removeNames = FALSE,
writecsv = FALSE)
appendSpeciesNames

Arguments

inDir character. Directory containing camera trap images sorted into station subdirectories (e.g. inDir/StationA/)

IDfrom character. Read species ID from image metadata ("metadata") of from species directory names ("directory")?

hasCameraFolders logical. Do the station subdirectories of inDir have camera-subdirectories (e.g. inDir/StationA/CameraA1; inDir/StationA/CameraA2)?

metadataSpeciesTag character. The species ID tag name in image metadata (if IDfrom = "metadata").

metadataHierarchyDelimiter character. The character delimiting hierarchy levels in image metadata tags in field "HierarchicalSubject". Either "|" or ":".

removeNames logical. remove appended species names?

writecsv logical. write csv table containing old and new file names into inDir?

Details

Species names can be appended or removed from image filenames. Before running the function, you may want to run checkSpeciesIdentification to detect possible misidentifications. As an example, the function would change an image file name from "StationA__2015-05-41__20-59-59(1).JPG" to "StationA__2015-05-41__20-59-59(1)__Species Name.JPG". If species names were appended several times by accident, they can all be removed by running the function with removeNames = TRUE

Value

A data.frame containing the old and new file names and directories.

Author(s)

Juergen Niedballa

Examples

## Not run:

# copy sample images to another location (so we don't mess around in the package directory)
wd_images_ID <- system.file("pictures/sample_images_species_dir", package = "camtrapR")
file.copy(from = wd_images_ID, to = getwd(), recursive = TRUE)
wd_images_ID_copy <- file.path(getwd(), "sample_images_species_dir")

# append species names
SpecNameAppend1 <- appendSpeciesNames(inDir = wd_images_ID_copy, 
IDfrom = "directory", 
hasCameraFolders = FALSE, 
removeNames = FALSE, 
writecsv = FALSE)
SpecNameAppend1

# remove species names
SpecNameRemove1 <- appendSpeciesNames(inDir = wd_images_ID_copy,
                                      IDfrom = "directory",
                                      hasCameraFolders = FALSE,
                                      removeNames = TRUE,
                                      writecsv = FALSE)

SpecNameRemove1

## End(Not run)

cameraOperation

Create a camera trap station operability matrix

Description

Construct a matrix of daily camera trap station operability for use in detectionHistory and spatialDetectionHistory, where it is needed for calculating trapping effort per occasion. If several cameras were deployed per station, the matrix can contain camera- or station-specific trap operation information.

Usage

cameraOperation(CTtable,
                stationCol = "Station",
                cameraCol,
                sessionCol,
                setupCol,
                retrievalCol,
                hasProblems = FALSE,
                byCamera,
                allCamsOn,
                camerasIndependent,
                dateFormat = "%Y-%m-%d",
                writecsv = FALSE,
                outDir
               )

Arguments

CTtable  data.frame containing information about location and trapping period of camera trap stations
stationCol character. name of the column specifying Station ID in CTtable
cameraCol  character. name of the column specifying Camera ID in CTtable (optional). If empty, 1 camera per station is assumed.
**sessionCol** character. Name of the column specifying session ID in CTtable (optional). Use it for creating multi-session / multi-season detection histories (unmarked: `unmarkedMultFrame`; sec: `capthist`)

**setupCol** character. Name of the column containing camera setup dates in CTtable

**retrievalCol** character. Name of the column containing camera retrieval dates in CTtable

**hasProblems** logical. If TRUE, function will look for columns specifying malfunction periods in CTtable (naming convention: `ProblemX_from` and `ProblemX_to`, where X is a number)

**byCamera** logical. If TRUE, camera operability matrix is computed by camera, not by station (requires `cameraCol`)

**allCamsOn** logical. Takes effect only if `cameraCol` is defined and if `byCamera` is FALSE. If `allCamsOn` = TRUE, all cameras at a station need to be operational for the station to be operational (e.g. 1 camera out of 2 malfunctioning renders the station inoperational). Output values can be 1/0/NA only (all cameras at a station operational/ at least 1 camera not operational/ no camera set up). If `allCamsOn` = FALSE, at least 1 active camera makes a station operational.

**camerasIndependent** logical. Return number of active camera traps by station? Only if `byCamera` is FALSE and `allCamsOn` is FALSE. If `camerasIndependent` is TRUE, output values will be the number of operational cameras at a station. If `camerasIndependent` is FALSE, the value is 1 if at least 1 camera was operational, otherwise 0. In both cases, values are NA if no camera was set up.

**dateFormat** character. The format of columns `setupCol` and `retrievalCol` (and potential problem columns) in CTtable. Must be interpretable by either `as.Date` or the "orders" argument `parse_date_time` in `lubridate`.

**writecsv** logical. Should the camera operability matrix be saved as a .csv?

**outDir** character. Directory into which csv is saved

**Details**

cameraCol is NULL by default. The function then assumes there was 1 camera per station CTtable. In more than 1 camera was deployed per station, cameraCol needs to be specified to identify individual cameras within a station. `dateFormat` defaults to "YYYY-MM-DD", e.g. "2014-10-31". It can be specified either in the format required by `strptime` or the 'orders' argument in `parse_date_time` in `lubridate`. In the example above, "YYYY-MM-DD" would be specified as "%Y-%m-%d" or "ymd".

If `hasProblems` is TRUE, the function tries to find columns `ProblemX_from` and `ProblemX_to` in CTtable. X is a consecutive number from 1 to n, specifying periods in which a camera or station was not operational. If `hasProblems` is FALSE, cameras are assumed to have been operational uninterruptedly from setup to retrieval (see `camtraps` for details). `allCamsOn` only has an effect if there was more than 1 camera at a station. If TRUE, for the station to be considered operational, all cameras at a station need to be operational. If FALSE, at least 1 active camera renders the station operational. Argument `camerasIndependent` defines if cameras record animals independently (it thus only has an effect if there was more than 1 camera at a station). This is the case if an observation at one camera does not increase the probability for detection at another camera (cameras face
different trails at a distance of one another). Non-independence occurs if an animal is likely to trigger both cameras (as would be the case with 2 cameras facing each other). If `camerasIndependent` is TRUE, 2 active cameras at a station will result in a station operation value of 2 in the resulting matrix, i.e., 2 independent trap days at 1 station and day. If `camerasIndependent` is FALSE, 2 active cameras will return value 1, i.e., 1 trap night at 1 station per day.

Row names depend on the input arguments and contain the station name and potentially session and camera names (if `sessionCol` and/or `cameraCol` are defined).

Naming convention is (since version 1.2) **Bold** information are from the columns `stationCol`, `sessionCol` and `cameraCol` in `CTtable`:

```
Station  
Station__SESS_SessionID  
Station__CAM_CameraID  
Station__SESS_SessionID__CAM_CameraID  
```

Session are designated with prefix "__SESS_", cameras with prefix "__CAM_". Therefore, these are reserved words and may not be part of station, session or camera names. Here’s what it may look like in real life:

```
Station1  
Station1__SESS_2019  
Station1__CAM_1024152  
Station1__SESS_2019__CAM_1024152  
```

Functions `detectionHistory` and `spatialDetectionHistory` recognize these and use the information accordingly.

**Value**

A matrix. Row names always indicate Station IDs. If `sessionCol` and/or `cameraCol` are defined, they are contained in the row names also (camera ID only if `byCamera = TRUE`). Column names are dates.

Legend: NA: camera(s) not set up, 0: camera(s) not operational, 1 (or higher): number of operational camera(s) or an indicator for whether the station was operational (depending on `camerasIndependent` and `allCamsOn`)

**Note**

Setting `camerasIndependent` according to the sampling situation is important for the functions `detectionHistory` and `spatialDetectionHistory`, if sampling effort (the number of active trap nights in a occasion) is to be computed and returned.

**Author(s)**

Juergen Niedballa

**Examples**

```
data(camtraps)  
# no problems/malfunction  
```
camop_no_problem <- cameraOperation(CTtable = camtraps, 
    stationCol = "Station",
    setupCol = "Setup_date",
    retrievalCol = "Retrieval_date",
    writecsv = FALSE,
    hasProblems = FALSE,
    dateFormat = "%d/%m/%Y"
)

# with problems/malfunction
camop_problem <- cameraOperation(CTtable = camtraps, 
    stationCol = "Station",
    setupCol = "Setup_date",
    retrievalCol = "Retrieval_date",
    writecsv = FALSE,
    hasProblems = TRUE,
    dateFormat = "%d/%m/%Y"
)

# with problems/malfunction / dateFormat in lubridate format
camop_problem_lubridate <- cameraOperation(CTtable = camtraps, 
    stationCol = "Station",
    setupCol = "Setup_date",
    retrievalCol = "Retrieval_date",
    writecsv = FALSE,
    hasProblems = TRUE,
    dateFormat = "dmy"
)

camop_no_problem

### camtraps

**Sample camera trap station information**

#### Description
Example camera trap station information table

#### Usage

data(camtraps)

#### Format
A data frame with 3 rows and 7 variables
Details

This is a general example of how information about camera trap stations are arranged in camtrapR. It contains setup and retrieval dates and coordinates. If more than 1 camera was set up at a station (e.g. 2 cameras facing each other), a camera ID column must be added, with camera-specific information instead of station-specific information. If cameras malfunctioned repeatedly, additional pairs of problem columns can be added, e.g. "Problem2_from" and "Problem2_to" etc..

The variables are as follows:

- Station. Camera trap station ID
- utm_y. y coordinate of station (northing)
- utm_x. x coordinate of station (easting)
- Setup_date. camera trap setup date
- Retrieval_date. camera trap retrieval date
- Problem1_from. first day of camera malfunction
- Problem1_to. last day of camera malfunction

Note

The coordinates can be in the units of any coordinate system. UTM was chosen as an example, but it could be latlong or anything else, too. capthist objects (as created by spatialDetectionHistory for spatial capture-recapture analyses) expect the unit to be meters.

---

camtrapsMultiSeason  Sample multi-season camera trap station information

Description

Example multi-season camera trap station information table

Usage

data(camtrapsMultiSeason)

Format

A data frame with 7 rows and 8 variables

Details

This is a general example of how information about camera trap stations from multiple seasons are arranged in camtrapR. It contains setup and retrieval dates, coordinates and a season identifier. If more than 1 camera was set up at a station (e.g. 2 cameras facing each other), a camera ID column must be added, with camera-specific information instead of station-specific information. If cameras malfunctioned repeatedly, additional pairs of problem columns can be added, e.g. "Problem2_from" and "Problem2_to" etc..
Note that season 2010 has an additional station (StationD). This is to simulate a situation where a station was not set up during an entire season.

The variables are as follows:

- Station. Camera trap station ID
- utm_y. y coordinate of station (northing)
- utm_x. x coordinate of station (easting)
- Setup_date. camera trap setup date
- Retrieval_date. camera trap retrieval date
- Problem1_from. first day of camera malfunction
- Problem1_to. last day of camera malfunction
- session. Identified for trapping session / season

Note

The coordinates can be in the units of any coordinate system. UTM was chosen as an example, but it could be latlong or anything else, too. capthist objects (as created by spatialDetectionHistory for spatial capture-recapture analyses) expect the unit to be meters. capthist also require session information as integer numbers starting with 1.

"Season" and "session" are used synonymously here. secr nomenclature is "session", in unmarked it is "season".

Examples

```r
# data were created with the following code:
data(camtraps)
camtraps_season2 <- camtraps

# change 2009 to 2010
camtraps_season2[, "Setup_date"] <- gsub("2009", "2010", camtraps_season2[, "Setup_date"])
camtraps_season2[, "Retrieval_date"] <- gsub("2009", "2010", camtraps_season2[, "Retrieval_date"])
camtraps_season2[, "Problem1_from"] <- gsub("2009", "2010", camtraps_season2[, "Problem1_from"])
camtraps_season2[, "Problem1_to"] <- gsub("2009", "2010", camtraps_season2[, "Problem1_to"])

# add an extra station with different dates in session 2010
camtraps_season2 <- rbind(camtraps_season2, NA)
camtraps_season2$Station[4] <- "StationD"
camtraps_season2$utm_y[4] <- 607050
camtraps_season2$utm_x[4] <- 525000
camtraps_season2$Setup_date[4] <- "04/04/2010"
camtraps_season2$Retrieval_date[4] <- "17/06/2010"
camtraps_season2$Problem1_from[4] <- "20/05/2010"
camtraps_season2$Problem1_to[4] <- "30/05/2010"

# add season column
camtraps$session <- 2009
camtraps_season2$session <- 2010
```
# combine the tables for 2 seasons
camtrapsMultiSeason <- rbind(camtraps, camtraps_season2)

---

**checkSpeciesIdentification**

*Consistency check on species image identification*

**Description**

This function serves 2 purposes: 1) it assesses possible misidentification of species and 2) compares double observer species identification (only if metadata tagging was used for species identification). Within each station, it assesses whether there are images of a species taken within a given time interval of another species. Often, it is unlikely that different species are encountered within a very short time intervals at the same location. This type of misidentification can arise easily if some images belonging to a sequence of images were accidentally moved into different species directories or tagged incorrectly.

Double observer identification may be desirable to increase reliability of species identification. The function returns conflicts in species identification between 2 observers. These conflicts can then be corrected.

**Usage**

```r
checkSpeciesIdentification(inDir, 
  IDfrom, 
  hasCameraFolders, 
  metadataSpeciesTag, 
  metadataSpeciesTagToCompare, 
  metadataHierarchyDelimiter = "\|", 
  maxDeltaTime, 
  excludeSpecies, 
  stationsToCheck, 
  writecsv = FALSE
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inDir</td>
<td>character. Directory containing identified camera trap images sorted into station subdirectories (e.g. inDir/StationA/)</td>
</tr>
<tr>
<td>IDfrom</td>
<td>character. Read species ID from image metadata (&quot;metadata&quot;) of from species directory names (&quot;directory&quot;)?</td>
</tr>
<tr>
<td>hasCameraFolders</td>
<td>logical. Do the station directories in inDir have camera subdirectories (e.g. &quot;inDir/StationA/Camera1&quot; or &quot;inDir/StationA/Camera1/Species1&quot;)?</td>
</tr>
</tbody>
</table>
The function `checkSpeciesIdentification` can be used to identify images that were accidentally misidentified or moved into wrong species directories. It works by comparing images that are triggered by the same camera at a particular station within a specified time interval (`maxDeltaTime`). If multiple observers independently identify images using metadata tagging, their identifications can be compared by setting `metadataSpeciesTagToCompare`. Conflicting or missing identifications will be reported. This feature is only available if images were identified by metadata tagging. Species like "blank" or "team" can be ignored using `excludeSpecies`. If only specific stations are to be checked, `stationsToCheck` can be set.

The function returns a list containing 2 data frames. The first data frame contains images with conflicting species IDs (if `IDfrom = "metadata"` and `metadataSpeciesTagToCompare` is defined) and the second data frame contains images with conflicting species IDs (if `IDfrom = "metadata"` and `metadataSpeciesTagToCompare` is defined).

**Author(s)**

Juergen Niedballa
checkSpeciesNames

Check species names against the ITIS taxonomic database

Description

The function checks species names (common or scientific names) provided by the user with the ITIS taxonomic database (http://www.itis.gov/) via functions from the package taxize. It returns both common and scientific names, the taxon authors, taxon rank name and status, the TSN (taxonomic serial numbers) and ITIS urls.
Usage

checkSpeciesNames(speciesNames, searchtype, accepted = TRUE, ask = TRUE)

Arguments

speciesNames character. Vector of species names to check. Either common names or scientific names.
searchtype character. Type of names specified in speciesNames. One of 'scientific' or 'common'.
accepted logical. Return only accepted valid names? If TRUE, invalid names are returned as NA. Set to FALSE to return both accepted and unaccepted names.
ask logical. Should the function be run in interactive mode? If TRUE and more than one TSN is found for a species, the user is asked to choose one. If FALSE, NA is returned for multiple matches.

Details

Arguments searchtype, accepted and ask are passed on to get_tsn.

Value

A data.frame with the names supplied by the user, matching common and scientific names, taxon author and year, taxonomic rank, status, TSNs (taxonomic serial numbers) and ITIS urls.

Author(s)

Juergen Niedballa

References

http://www.itis.gov/

Examples

## Not run:

species_common <- c("Leopard Cat", "moonrat")

# ask = TRUE. Multiple matches for leopard cat will cause menu to pop up asking user input.

species.names.check1 <- checkSpeciesNames(speciesNames = species_common, searchtype = "common", accepted = TRUE,
2   # we choose entry 2
species.names.check1

# ask = FALSE. Multiple matches for leopard cat will cause NA.

species.names.check2 <- checkSpeciesNames(speciesNames = species_common,
searchtype = "common",
accepted = TRUE,
ask = FALSE)

species.names.check2

# search for scientific names

species_scientific <- c("Tragulus", "Prionailurus bengalensis")

species.names.check3 <- checkSpeciesNames(speciesNames = species_scientific,
searchtype = "scientific",
accepted = TRUE,
ask = TRUE)

species.names.check3

## End(Not run)

createSpeciesFolders

Create species directories for species identification

Description

This function creates species subdirectories within station directories. They can be used for species identification by manually moving images into the respective species directories. The function can also delete empty species directories (if species were not detected at sites). It is not necessary to run this function if animals will be identified by metadata tagging.

Usage

createSpeciesFolders(inDir,
hasCameraFolders,
species,
removeFolders = FALSE
)

Arguments

inDir character. Directory containing camera trap images sorted into station subdirectories (e.g. inDir/StationA/)
hasCameraFolders logical. Do the station directories in `inDir` have camera-subdirectories (e.g. `inDir/StationA/CameraA1`; `inDir/StationA/CameraA2`)?

species character. names of species directories to be created in every station (or station/camera) subdirectory of `inDir`.

removeFolders logical. Indicating whether to create (TRUE) or remove (FALSE) species directories.

Details

This function should be run after `imageRename`. Empty directories can be created as containers for species identification if images are identified with the drag & drop method. After species identification is complete, empty species directories can be deleted using `removeFolders = TRUE`. The function will delete only directories which are specified in `species`. If `hasCameraFolders` was set to `TRUE` in function `imageRename`, `hasCameraFolders` must be set to `TRUE` here too. Species directories will then be created within each camera subdirectory of each station directory. If the user wishes to identify species by metadata tagging, running this function is not needed.

Value

A data.frame with directory names and an indicator for whether directories were created or deleted.

Author(s)

Juergen Niedballa

Examples

```r
## Not run:
# create dummy directories for tests
# (normally, you'd use directory containing renamed, unsorted images)

# this will be used as inDir
wd_createDirTest <- file.path(getwd(), "createSpeciesFoldersTest")

# now we create 2 station subdirectories
dirs_to_create <- file.path(wd_createDirTest, c("StationA", "StationB"))
sapply(dirs_to_create, FUN = dir.create, recursive = TRUE)

# species names for which we want to create subdirectories
species <- c("Sambar Deer", "Bay Cat")

# create species subdirectories
SpecFolderCreate1 <- createSpeciesFolders (inDir = wd_createDirTest,
                                          species = species,
                                          hasCameraFolders = FALSE,
                                          removeFolders = FALSE)

SpecFolderCreate1
```
# check if directories were created
list.dirs(wd_createDirTest)

# delete empty species directories
SpecFolderCreate2 <- createSpeciesFolders (inDir = wd_createDirTest,
species = species,
hasCameraFolders = FALSE,
removeFolders = TRUE)

SpecFolderCreate2

# check if species directories were deleted
list.dirs(wd_createDirTest)

## End(Not run)

---

createStationFolders create camera trap station directories for raw camera trap images

Description

This function creates camera trap station directories, if needed with camera subdirectories. They can be used as an initial directory structure for storing raw camera trap images.

Usage

createStationFolders(inDir, stations, cameras, createInDir)

Arguments

inDir character. Directory in which station directories are to be created
stations character. Station IDs to be used as directory names within inDir
cameras character. Camera trap IDs to be used as subdirectory names in each station directory (optionally)
createInDir logical. If inDir does not exist, create it?

Details

The empty directories serve as containers for saving raw camera trap images. If more than 1 camera was set up at a station, specifying cameras is required in order to keep images from different cameras separate. Otherwise, generic filenames (e.g., IMG0001.JPG) from different cameras may lead to accidental overwriting of images if images from these cameras are saved in one station directory.
detectionHistory

Value

A data.frame with station (and possibly camera) directory names and an indicator for whether they were created successfully.

Author(s)

Juergen Niedballa

Examples

## Not run:

```r
# create dummy directory for tests (this will be used as inDir)
# (normally, you'd set up an empty directory, e.g. .../myStudy/rawImages)
wd_createStationDir <- file.path(tempdir(), "createStationFoldersTest")

# now we load the sample camera trap station data frame
data(camtraps)

data(camtraps)

# create station directories in wd_createStationDir
StationFolderCreate1 <- createStationFolders (inDir = wd_createStationDir,
                                               stations = as.character(camtraps$Station),
                                               createInDir = TRUE)

StationFolderCreate1

# check if directories were created
list.dirs(wd_createStationDir)

## End(Not run)
```

detectionHistory

Species detection histories for occupancy analyses

Description

This function generates species detection histories that can be used in occupancy analyses, e.g. with package `unmarked`. It generates detection histories in different formats, with adjustable occasion length and occasion start time.

Usage

```r
detectionHistory(recordTable, 
                 species, 
                 camOp, 
                 output = c("binary", "count"), 
                 stationCol = "Station", 
                 speciesCol = "Species",
```
recordDateTimeCol = "DateTimeOriginal",
recordDateTimeFormat = "%Y-%m-%d %H:%M:%S",
occasionLength,
minActiveDaysPerOccasion,
maxNumberDays,
day1,
buffer,
includeEffort = TRUE,
scaleEffort = FALSE,
occasionStartTime = 0,
datesAsOccasionNames = FALSE,
timeZone,
writecsv = FALSE,
outDir,
unmarkedMultFrameInput
)

Arguments

recordTable data.frame. the record table created by recordTable
species character. the species for which to compute the detection history
camOp The camera operability matrix as created by cameraOperation
output character. Return binary detections ("binary") or counts of detections ("count")
stationCol character. name of the column specifying Station ID in recordTable
speciesCol character. name of the column specifying species in recordTable
recordDateTimeCol character. name of the column specifying date and time in recordTable
recordDateTimeFormat character. Format of column recordDateTimeCol in recordTable
occasionLength integer. occasion length in days
minActiveDaysPerOccasion integer. minimum number of active trap days for occasions to be included (optional)
maxNumberDays integer. maximum number of trap days per station (optional)
day1 character. When should occasions begin: station setup date ("station"), first day of survey ("survey"), a specific date (e.g. "2015-12-31")?
buffer integer. Makes the first occasion begin a number of days after station setup. (optional)
includeEffort logical. Compute trapping effort (number of active camera trap days per station and occasion)?
scaleEffort logical. scale and center effort matrix to mean = 0 and sd = 1?
occasionStartTime integer. time of day (the full hour) at which to begin occasions.
datesAsOccasionNames
    If day1 = "survey", occasion names in the detection history will be composed
    of first and last day of that occasion.

timeZone
    character. Must be a value returned by OlsonNames

writecsv
    logical. Should the detection history be saved as a .csv?

outDir
    character. Directory into which detection history .csv file is saved

unmarkedMultFrameInput
    logical. Return input for multi-season occupancy models in unmarked (argument "y" in unmarkedMultFrame)?

Details

The function computes a species detection matrix, either as a detection-by-date or a detection-by-
occasion matrix. day1 defines if each stations detection history will begin on that station’s setup
day (day1 = "station") or if all station’s detection histories have a common origin (the day the
first station was set up if day1 = "survey" or a fixed date if, e.g. day1 = "2015-12-31"). If day1 is
a date, as.Date must be able to understand it. The most suitable format is "YYYY-MM-DD", e.g.
"2015-12-31".

output is analogous to spatialDetectionHistory. It makes the function return either counts of
detections during occasions, or a binary indicator for whether the species was detected.

includeEffort controls whether an additional effort matrix is computed or not. This also affects
the detection matrices. If includeEffort = FALSE, all occasions in which a station was not set up
or malfunctioning (NA or 0 in camOp) will result in NAs in the detection history. If includeEffort
= TRUE, the record history will only contain 0 and 1, and no NAs. The effort matrix can then
be included in occupancy models as a (continuous) observation covariate to estimate the effect of
effort on detection probability.

The number of days that are aggregated is controlled by occasionLength. occasionStartTime
can be used to make occasions begin another hour than midnight (the default). This may be rele-
vant for nocturnal animals, in which 1 whole night would be considered an occasion. The values
of stationCol in recordTable must be matched by the row names of camOp (case-insensitive),
otherwise an error is raised.

recordDateTimeFormat defaults to the "YYYY-MM-DD HH:MM:SS" convention, e.g. "2014-
09-30 22:59:59". recordDateTimeFormat can be interpreted either by base-R via strptime or in
lubridate via parse_date_time (argument "orders"). lubridate will be used if there are no "%" characters in recordDateTimeFormat.

For "YYYY-MM-DD HH:MM:SS", recordDateTimeFormat would be either "%Y-%m-%d %H:%M:%S"
or "ymd HMS". For details on how to specify date and time formats in R see strptime or
parse_date_time.

If the camera operation matrix (camOp) was created for a multi-season study (argument sessionsCol
in cameraOperation was set, it will be detected automatically. Output can be for unmarked-
MultiFrame by setting unmarkedMultFrameInput = TRUE. Each row corresponds to a site, and the
columns are in season-major, occasion-minor order, e.g. season1-occasion1, season1-occasion2,
etc.).
Value

Depending on the value of includeEffort and scaleEffort, a list with either 1, 2 or 3 elements. The first element is the species detection history. The second is the optional effort matrix and the third contains the effort scaling parameters.

detection_history
  A species detection matrix

effort
  A matrix giving the number of active camera trap days per station and occasion (= camera trapping effort). It is only returned if includeEffort = TRUE

effort_scaling_parameters
  Scaling parameters of the effort matrix. It is only returned if includeEffort and scaleEffort are TRUE

Warning

Setting output = "count" returns a count of detections, not individuals. We strongly advise against using it as input for models of animal abundance (such as N-Mixture models) models which use counts as input.

Please note the section about defining argument timeZone in the vignette on data extraction (accessible via vignette("DataExtraction") or online (https://cran.r-project.org/package=camtrapR/vignettes/camtrapr3.html)).

Author(s)

Juergen Niedballa

Examples

# define image directory
wd_images_ID <- system.file("pictures/sample_images_species_dir", package = "camtrapR")

# load station information
data(camtraps)

data(camtraps)

# create camera operation matrix
ccamop_no_problem <- cameraOperation(CTtable = camtraps,
stationCol = "Station",
setupCol = "Setup_date",
retrievalCol = "Retrieval_date",
hasProblems = FALSE,
dateFormat = "%d/%m/%Y"
)

## Not run:
if (Sys.which("exiftool") != ""){
  "only run this function if ExifTool is available"
recordTableSample <- recordTable(inDir = wd_images_ID,
IDfrom = "directory",
minDeltaTime = 60,
deltaTimeComparedTo = "lastRecord",}
data(recordTableSample) # load the record history, as created above

# compute detection history for a species

# without trapping effort
DetHist1 <- detectionHistory(recordTable = recordTableSample,
camOp = camop_no_problem,
stationCol = "Station",
speciesCol = "Species",
recordDateTimeCol = "DateTimeOriginal",
species = "VTA",
occasionLength = 7,
day1 = "station",
datesAsOccasionNames = FALSE,
includeEffort = FALSE,
timeZone = "Asia/Kuala_Lumpur"
)

DetHist1 # this is a list with 1 element
DetHist1$detection_history # this is the contained detection/non-detection matrix

# with effort / using base R to define recordDateTimeFormat
DetHist2 <- detectionHistory(recordTable = recordTableSample,
camOp = camop_no_problem,
stationCol = "Station",
speciesCol = "Species",
recordDateTimeCol = "DateTimeOriginal",
species = "VTA",
occasionLength = 7,
day1 = "station",
datesAsOccasionNames = FALSE,
includeEffort = TRUE,
scaleEffort = FALSE,
timeZone = "Asia/Kuala_Lumpur"
)

DetHist2$detection_history # detection history (alternatively, use: DetHist2[[1]])
DetHist2$effort # effort (alternatively, use: DetHist2[[2]])

# with effort / using lubridate package to define recordDateTimeFormat
DetHist2_lub <- detectionHistory(recordTable = recordTableSample,
camOp = camop_no_problem,
stationCol = "Station",
speciesCol = "Species",
recordDateTimeCol = "DateTimeOriginal",
exclude = "UNID",
timeZone = "Asia/Kuala_Lumpur"
)

## End(Not run)
recordDateTimeFormat = "ymd HMS",
species = "VTA",
ocasionLength = 7,
day1 = "station",
datesAsOccasionNames = FALSE,
includeEffort = TRUE,
scaleEffort = FALSE,
timeZone = "Asia/Kuala_Lumpur"
)

DetHist2_lub$detection_history # detection history (alternatively, use: DetHist2_lub[[1]])
DetHist2_lub$effort # effort (alternatively, use: DetHist2_lub[[2]])

# multi-season detection history

# load multi-season data
data(camtrapsMultiSeason)
data(recordTableSampleMultiSeason)

data(CTtable)

camop_season <- cameraOperation(CTtable = camtrapsMultiSeason,
    stationCol = "Station",
    setupCol = "Setup_date",
    sessionCol = "session",
    retrievalCol = "Retrieval_date",
    hasProblems = TRUE,
    dateFormat = "%d/%m/%Y"
)

detHist_multi <- detectionHistory(recordTable = recordTableSampleMultiSeason,
    camOp = camop_season,
    stationCol = "Station",
    speciesCol = "Species",
    species = "VTA",
    occasionLength = 10,
    day1 = "station",
    recordDateTimeCol = "DateTimeOriginal",
    includeEffort = TRUE,
    scaleEffort = FALSE,
    timeZone = "UTC",
    unmarkedMultFrameInput = TRUE
)

DetHist_multi

detectionMaps Generate maps of observed species richness and species presences by station
**detectionMaps**

**Description**

Generates maps of observed species richness and species presence by species and station. Output can be R graphics, PNG graphics or a shapefile for use in GIS software.

**Usage**

```r
detectionMaps(CTtable, recordTable, Xcol, Ycol, backgroundPolygon, stationCol = "Station", speciesCol = "Species", speciesToShow, richnessPlot = TRUE, speciesPlots = TRUE, addLegend = TRUE, printLabels = FALSE, smallPoints, plotR = TRUE, writePNG = FALSE, plotDirectory, createPlotDir = FALSE, pngMaxPix = 1000, writeShapefile = FALSE, shapefileName, shapefileDirectory, shapefileProjection
)
```

**Arguments**

- **CTtable**: data.frame. contains station IDs and coordinates
- **Xcol**: character. name of the column specifying x coordinates in CTtable
- **Ycol**: character. name of the column specifying y coordinates in CTtable
- **backgroundPolygon**: SpatialPolygons or SpatialPolygonsDataFrame. Polygon to be plotted in the background of the map (e.g. project area boundary)
- **stationCol**: character. name of the column specifying station ID in CTtable and recordTable
- **recordTable**: data.frame. the record table created by recordTable
- **speciesCol**: character. name of the column specifying species in recordTable
- **speciesToShow**: character. Species to include in the maps. If missing, all species in recordTable will be included.
- **writePNG**: logical. Create PNGs of the plots?
- **plotR**: logical. Create plots in R graphics device?
- **plotDirectory**: character. Directory in which to save the PNGs
createPlotDir  logical. Create plotDirectory?
richnessPlot  logical. Generate a species richness plot?
speciesPlots  logical. Generate plots of all species number of independent events?
printLabels  logical. Add station labels to the plots?
smallPoints  numeric. Number by which to decrease point sizes in plots (optional).
addLegend  logical. Add legends to the plots?
pngMaxPix  integer. number of pixels in pngs on the longer side
writeShapefile  logical. Create a shapefile from the output?
shapefileName  character. Name of the shapefile to be saved. If empty, a name will be generated automatically.
shapefileDirectory  character. Directory in which to save the shapefile.
shapefileProjection  character. A character string of projection arguments to use in the shapefile.

Details

The column name stationCol must be identical in CTtable and recordTable and station IDs must match.

Shapefile creation depends on the packages sp and rgdal. Argument shapefileProjection must be a valid argument of CRS. If shapefileProjection is undefined, the resulting shapefile will lack a coordinate reference system.

Value

An invisible data.frame with station coordinates, numbers of events by species at each station and total species number by station. In addition and optionally, R graphics or png image files.

Author(s)

Juergen Niedballa

References

A great resource for CRS arguments is http://spatialreference.org/. Use the Proj4 string as shapefileProjection argument.

Examples

# load station information
data(camtraps)

# load record table
data(recordTableSample)
# create maps
Mapstest <- detectionMaps(CTtable = camtraps,
recordTable = recordTableSample,
Xcol = "utm_x",
Ycol = "utm_y",
stationCol = "Station",
speciesCol = "Species",
writePNG = FALSE,
plotR = TRUE,
printLabels = TRUE,
richnessPlot = TRUE,
addLegend = TRUE)

# with a polygon in the background, and for one species only

# make a dummy polygon for the background
library(sp)
poly1 <- Polygon(cbind(c(521500,526500,527000, 521500),c(607500, 608000, 603500, 603500)))
poly2 <- Polygons(list(poly1), "s1")
poly3 <- SpatialPolygons(list(poly2))

Mapstest2 <- detectionMaps(CTtable = camtraps,
recordTable = recordTableSample,
Xcol = "utm_x",
Ycol = "utm_y",
backgroundPolygon = poly3, # this was added
speciesToShow = c("PBE", "VTA"), # this was added
stationCol = "Station",
speciesCol = "Species",
writePNG = FALSE,
plotR = TRUE,
printLabels = TRUE,
richnessPlot = TRUE,
addLegend = TRUE)

```

exifTagNames

Show Exif metadata of JPEG images or other image or video formats

Description

The function will return metadata values, metadata tag names and group names of Exif metadata of JPEG images or other formats.
Usage

```r
exifTagNames(inDir, 
  whichSubDir = 1, 
  fileName, 
  returnMetadata = "DEPRECATED", 
  returnTagGroup = "DEPRECATED")
```

Arguments

- `inDir` character. Directory containing camera trap images sorted into station subdirectories (e.g. `inDir/StationA/`)
- `whichSubDir` integer or character. Either number or name of subdirectory of `inDir` in which to look for an image
- `fileName` character. A filename, either the file name of an image in `inDir` or a full path with file name (in which case `inDir` is not needed)
- `returnMetadata` deprecated and ignored
- `returnTagGroup` deprecated and ignored

Details

Many digital cameras record information such as ambient temperature or moon phase under makerspecific tag names in Exif metadata of JPEG images. In addition, many technical information are stored in Exif metadata. In order to extract those information from images and add them to the record tables created by the functions `recordTable` and `recordTableIndividual`, the tag names must be known so they can be passed to these functions via the `additionalMetadataTags` argument.

By default the function returns both metadata tag names and the metadata group they belong to (via argument `returnTagGroup`). This is helpful to unambiguously address specific metadata tags, because different groups can contain tags of identical names, which may cause problems executing the functions `recordTable` and `recordTableIndividual`. The format is "GROUP:tag", e.g. "EXIF:Flash".

Value

A data frame containing three columns: metadata tag group, tag name, and values.

Author(s)

Juergen Niedballa

References

Phil Harvey’s ExifTool [https://exiftool.org/](https://exiftool.org/)

See Also

`recordTable`
Examples

```r
## Not run:
wd_images_ID <- system.file("pictures/sample_images_species_dir", package = "camtrapR")

# specify directory, camtrapR will automatically take first image from first subdirectory
exifTagNames(inDir = wd_images_ID)

# specify subdirectory by name, camtrapR will use first image
exifTagNames(inDir = wd_images_ID,
              whichSubDir = "StationA")

# specifying fileName only (line break due to R package policy)
exifTagNames(fileName = file.path(wd_images_ID, "StationC", "TRA",
                                "StationC__2009-05-02__00-10-00(1).JPG"))

# specify inDir and fileName
exifTagNames(inDir = wd_images_ID,
              fileName = file.path("StationC", "TRA", "StationC__2009-05-02__00-10-00(1).JPG"))

# it also works this way
exifTagNames(inDir = file.path(wd_images_ID, "StationC", "TRA"),
              fileName = "StationC__2009-05-02__00-10-00(1).JPG")

# with tagged sample images
wd_images_ID_tagged <- system.file("pictures/sample_images_indiv_tag", package = "camtrapR")
exifTagNames(inDir = wd_images_ID_tagged)

## End(Not run)
```

---

**exiftoolPath**

*Add a directory to PATH temporarily*

---

**Description**

Temporarily adds a directory to the environmental variable PATH for system calls from within R. This allows Windows users to store exiftool.exe anywhere on their hard drive. It is not needed on Linux or MacOS machines.

**Usage**

```r
exiftoolPath(exiftoolDir)
```

**Arguments**

- `exiftoolDir` character, the directory in the file system containing exiftool.exe.
Details

Several functions within this package depend on ExifTool. Under Windows, exiftool.exe cannot be used if it is not in a directory path specified in PATH. This can be solved by adding the directory containing exiftool.exe for temporary use within the running R process.

Value

invisible logical indicating whether `exiftoolDir` was added to PATH successfully (in the running R process).

Note

The directories in PATH can be queried by `Sys.getenv("PATH")`.

Author(s)

Juergen Niedballa

Examples

```r
exiftool_dir <- "C:/Path/To/Exiftool"
exiftoolPath(exiftoolDir = exiftool_dir)

# check if it has been added to PATH
grepl(exiftool_dir, Sys.getenv("PATH"))
```

Description

Some camera models don’t store the date/time information in the standard Exif metadata tag. Consequently, camtrapR cannot find that information. This function uses Exiftool to update the DateTimeOriginal metadata tag in all images within a directory to make them readable with camtrapR (and other software).

Usage

```r
fixDateTimeOriginal(inDir, recursive = TRUE)
```

Arguments

- `inDir` character. Name of the directory containing images to be fixed
- `recursive` logical. Recursively find images in subdirectories of `inDir`?
getSpeciesImages

Details

Some Reconyx Hyperfire cameras (e.g. HC500) are known to show this problem.

Value

Returns invisibly the messages returned by the Exiftool call (warnings etc.).

Warning

Please make a backup of your images before running this function.

Author(s)

Juergen Niedballa

References

This function uses the code from:

Examples

```r
## Not run:
# a hypothetical example

wd_images_hyperfire <- "C:/Some/Directory"

fixDateTimeOriginal(inDir = wd_images_hyperfire, 
                   recursive = TRUE)

## End(Not run)
```

getSpeciesImages     Collect all images of a species

Description

This function will fetch all images of a particular species from all camera trap stations and copies these images to a new location. The images which are to be copied are found in one of 2 possible ways, 1) by providing an existing record table (created with recordTable) or 2) by reading species IDs from species directories or from metadata (calling ExifTool). Earlier in the workflow, i.e., before running this function, images should have been renamed (with imageRename) to give images unique file names based on station ID and date/time.
Usage

getSpeciesImages(species, recordTable, speciesCol = "Species", stationCol = "Station", inDir, outDir, createStationSubfolders = FALSE, IDfrom, metadataSpeciesTag, metadataHierarchyDelimiter = "|")

Arguments

species character. Species whose images are to be fetched
recordTable data frame. A data frame as returned by function *recordTable*. If you specify this argument, do not specify inDir
speciesCol character. Name of the column specifying species ID in recordTable. Only required if recordTable is defined
stationCol character. Name of the column specifying station ID in recordTable. Only required if recordTable is defined
inDir character. Directory containing identified (species level) camera trap images sorted into station subdirectories (e.g. inDir/StationA/). If you specify this argument, do not specify recordTable.
outDir character. Directory in which to save species images. A species subdirectory will be created in outDir automatically.
createStationSubfolders logical. Save images in station directories within the newly created species directory in outDir?
IDfrom character. Read species ID from image metadata ("metadata") of from species directory names ("directory")? Only required if inDir is defined.
metadataSpeciesTag character. The species ID tag name in image metadata (if IDfrom = "metadata"). Only required if inDir is defined.
metadataHierarchyDelimiter character. The character delimiting hierarchy levels in image metadata tags in field "HierarchicalSubject". Either ":" or ":" (if IDfrom = "metadata"). Only required if inDir is defined and IDfrom = "metadata".

Details

The function finds the images to be copied by either consulting a record table created with *recordTable* or by reading species IDs from images. The former is considerably faster because ExifTool is not called, but requires images to be in precisely the location given by the columns Directory and FileName in recordTable. To use this feature, provide the function with a record table in argument recordTable.
If you’d rather read species IDs from images within the function (to make sure all file paths are correct), images need to be in the directory structure required by the package, e.g.

> inDir/Station/Species

or

> inDir/Station/Camera/Species

if using species directories for species IDs, and

> inDir/Station

or

> inDir/Station/Camera

if reading IDs from species metadata tags. In the latter case, only station directories are needed. In any case, the argument species must match species IDs (either the speciesCol in recordTable, species directory names or species metadata tags).

Before running the function, first rename the images using function imageRename to provide unique file names and prevent several images from having the same name (if generic names like "IMGP0001.jpg" are used). The function will not copy images if there are duplicate filenames to prevent overwriting images unintentionally.

Value

A data.frame with old and new directories and file names and the copy status (copy_ok; TRUE if copying was successful, FALSE if not).

Author(s)

Juergen Niedballa

Examples

```r
## Not run:
# define image directory
wd_images_ID <- system.file("pictures/sample_images_species_dir", package = "camtrapR")
wd_images_ID_copy <- file.path(tempdir(), "sample_images_species_dir")

species_to_copy <- "VTA"  # = Viverra tangalunga, Malay Civet

specImagecopy <- getSpeciesImages(species = species_to_copy,
                                   inDir = wd_images_ID,
                                   outDir = wd_images_ID_copy,
                                   createStationSubfolders = FALSE,
                                   IDfrom = "directory")
```

## End(Not run)
imageRename

Description

The function renames and copies raw camera trap images into a new location where they can be identified. Images are renamed with camera trap station ID, camera ID (optional), creation date and a numeric identifier for images taken within one minute of each other at a given station. Station ID and camera ID are derived from the raw image directory structure. The creation date is extracted from image metadata using ExifTool.

Usage

imageRename(inDir, outDir, hasCameraFolders, keepCameraSubfolders, createEmptyDirectories = FALSE, copyImages = FALSE, writecsv = FALSE)

Arguments

inDir character. Directory containing camera trap images sorted into station subdirectories (e.g. inDir/StationA/)

outDir character. Directory into which the renamed images will be copied

hasCameraFolders logical. Do the station directories in inDir have camera subdirectories (e.g. "inDir/StationA/Camera1")?

keepCameraSubfolders logical. Should camera directories be preserved as subdirectories of outDir (e.g. "outDir/StationA/CameraA1")?

createEmptyDirectories logical. If station or camera directories are empty, should they be copied nevertheless (causing empty directories in inDir, but preserving the whole directory structure)?

copyImages logical. Copy images to outDir?

writecsv logical. Save a data frame with a summary as a .csv? The csv will be saved in outDir.

Details

Setting up the correct raw image directory structure is necessary for running the function successfully. inDir is the main directory that contains camera trap station subdirectories (e.g. inDir/StationA). If one camera was deployed per station and no camera subdirectories are used within
station directories, hasCameraFolders can be set to FALSE. If more than one camera was deployed at stations, there must be subdirectories for the individual camera traps within the station directories (e.g. "inDir/StationA/CameraA1" and "inDir/StationA/CameraA2"). Even if only some stations had multiple cameras, all station will need camera subdirectories. The argument hasCameraFolders must be TRUE. Within the camera subdirectories, the directory structure is irrelevant.

Renaming of images follows the following pattern: If hasCameraFolders is TRUE, it is: "StationID__CameraID__Date__Time(Number).JPG", e.g. "StationA__CameraA1__2015-01-31__18-59-59(1).JPG". If hasCameraFolders is FALSE, it is: "StationID__Date__Time(Number).JPG", e.g. "StationA__2015-01-31__18-59-59(1).JPG".

The purpose of the number in parentheses is to prevent assigning identical file names to images taken at the same station (and camera) in the same second, as can happen if cameras take sequences of images. It is a consecutive number given to all images taken at the same station by the same camera within one minute. The double underscore "__" in the image file names is for splitting and extracting information from file names in other functions (e.g. for retrieving camera IDs in recordTable if camera subdirectories are not preserved (keepCameraSubfolders = FALSE)).

The function finds all JPEG images and extracts the image timestamp from the image metadata using ExifTool and copies the images (with new file names) into outDir, where it will set up a directory structure based on the station IDs and, if required by keepCameraSubfolders = TRUE, camera IDs (e.g. outDir/StationA/ or outDir/StationA/CameraA1).

copyImages can be set to FALSE to simulate the renaming and check the file names of the renamed images without copying. If you are handling large number of images (>e.g., 100,000), the function may take some time to run.

Value
A data.frame with original directory and file names, new directory and file names and an indicator for whether images were copied successfully.

Author(s)
Juergen Niedballa

References
Phil Harvey’s ExifTool https://exiftool.org/

Examples

```r
## Not run:
### "trial" run. create a table with file names after renaming, but don't copy images.

# first, find sample image directory in package directory:
wd_images_raw <- system.file("pictures/raw_images", package = "camtrapR")

# because copyImages = FALSE, outDir does not need to be defined
renaming_table <- imageRename(inDir = wd_images_raw,
```
### a real example in which images are copied and renamed

```r
# define raw image location
wd_images_raw <- system.file("pictures/raw_images", package = "camtrapR")

# define destination for renamed images
wd_images_raw_renamed <- file.path(tempdir(), "raw_images_renamed")

# now we have to define outDir because copyImages = TRUE
renaming.table2 <- imageRename(
inDir = wd_images_raw,
outDir = wd_images_raw_renamed,
hasCameraFolders = FALSE,
copyImages = TRUE,
writecsv = FALSE
)

# show output files
list.files(wd_images_raw_renamed, recursive = TRUE)

# output table
renaming.table2
```

### End(Not run)

---

**recordTable**  
*Generate a species record table from camera trap images and videos*

**Description**

Generates a record table from camera trap images or videos. Images/videos must be sorted into station directories at least. The function can read species identification from a directory structure (Station/Species or Station/Camera/Species) or from image metadata tags.

**Usage**

```r
recordTable(inDir,
IDfrom,
cameraID,
camerasIndependent,
exclude,
minDeltaTime = 0,
hasCameraFolders = FALSE,
copyImages = FALSE,
writecsv = FALSE
)```
Arguments

inDir character. Directory containing station directories. It must either contain images in species subdirectories (e.g. inDir/StationA/SpeciesA) or images with species metadata tags (without species directories, e.g. inDir/StationA).

IDfrom character. Read species ID from image metadata ("metadata") of from species directory names ("directory")?

cameraID character. Where should the function look for camera IDs: 'filename', 'directory'. 'filename' requires images renamed with imageRename. 'directory' requires a camera subdirectory within station directories (station/camera/species). Can be missing.

camerasIndependent logical. If TRUE, species records are considered to be independent between cameras at a station.

exclude character. Vector of species names to be excluded from the record table

minDeltaTime integer. Time difference between records of the same species at the same station to be considered independent (in minutes)

deltaTimeComparedTo character. For two records to be considered independent, must the second one be at least minDeltaTime minutes after the last independent record of the same species ("lastIndependentRecord"), or minDeltaTime minutes after the last record ("lastRecord")?

timeZone character. Must be a value returned by OlsonNames

stationCol character. Name of the camera trap station column. Assuming "Station" if undefined.

writecsv logical. Should the record table be saved as a .csv?

outDir character. Directory to save csv to. If NULL and writecsv = TRUE, recordTable will be written to inDir.

metadataHierarchyDelimiter character. The character delimiting hierarchy levels in image metadata tags in field "HierarchicalSubject". Either "i" or ":".
metadataSpeciesTag
character. In custom image metadata, the species ID tag name.

additionalMetadataTags
character. Additional camera model-specific metadata tags to be extracted. (If possible specify tag groups as returned by `exifTagNames`)

removeDuplicateRecords
logical. If there are several records of the same species at the same station (also same camera if `cameraID` is defined) at exactly the same time, show only one?

returnFileNamesMissingTags
logical. If species are assigned with metadata and images are not tagged, return a few file names of these images as a message?

eventSummaryColumn
character. A column in the record table (e.g. from a metadata tag) by to summarise non-independent records (those within `minDeltaTime` of a given record) with a user-defined function (`eventSummaryFunction`)

eventSummaryFunction
character. The function by which to summarise `eventSummaryColumn` of non-independent records, e.g. "sum", "max" (optional)

video
list. Contains information on how to handle video data (optional). See details.

Details

The function can handle a number of different ways of storing images, and supports species identification by moving images into species directories as well as metadata tagging. In every case, images need to be stored into station directories. If images are identified by moving them into species directories, a camera directory is optional: "Station/Species/XY.JPG" or "Station/Camera/Species/XY.JPG". Likewise, if images are identified using metadata tagging, a camera directory can be used optionally: "Station/XY.JPG" or "Station/Camera/XY.JPG".

If images are identified by metadata tagging, `metadataSpeciesTag` specifies the metadata tag group name that contains species identification tags. `metadataHierarchyDelimitor` is "|" for images tagged in DigiKam and images tagged in Adobe Bridge / Lightroom with the default settings. It is only necessary to change it if the default was changed in these programs.

`minDeltaTime` is a criterion for temporal independence of species recorded at the same station. Setting it to 0 will make the function return all records. Setting it to a higher value will remove records that were taken less than `minDeltaTime` minutes after the last record (deltaTimeComparedTo = "lastRecord") or the last independent record (deltaTimeComparedTo = "lastIndependentRecord").

camerasIndependent defines if the cameras at a station are to be considered independent. If TRUE, records of the same species taken by different cameras are considered independent (e.g. if they face different trails). Use FALSE if both cameras face each other and possibly TRUE).

`exclude` can be used to exclude "species" directories containing irrelevant images (e.g. "team", "blank", "unidentified"). `stationCol` can be set to match the station column name in the camera trap station table (see `camtraps`).

Many digital images contain Exif metadata tags such as "AmbientTemperature" or "MoonPhase" that can be extracted if specified in `metadataTags`. Because these are manufacturer-specific and not standardized, function `exifTagNames` provides a vector of all available tag names. Multiple names
can be specified as a character vector as: `c(Tag1,Tag2,...)`. The metadata tags thus extracted may be used as covariates in modelling species distributions.

`eventSummaryColumn` and `eventSummaryFunction` can be used to extract summary statistics for independent sampling events. For example, you assigned a "count" tag to your images, indicating the number of individuals in a picture. In a sequence of pictures taken within 1 minute, most pictures show one individual, but one image shows two individuals. You tagged the images accordingly (count = 1 or count = 2) and run `recordTable`. Set `eventSummaryColumn = "count"` and `eventSummaryFunction = "max"` to obtain the maximum number of count in all images within `minDeltaTime` minutes of a given record. The results is in a new column, in this example `count_max`. You can also calculate several statistics at the same time, by supplying vectors of values, e.g. `eventSummaryColumn = c("count","count","camera")` and `eventSummaryFunction = c("min","max","unique")` to get minimum and maximum count and all unique camera IDs for that event. Note that `eventSummaryColumn` and `eventSummaryFunction` must be of same length.

Argument `video` is a named list with 2 or 4 items. 2 items (file_formats, dateTimeTag) are always required, and are sufficient if `IDfrom = "directory"`. In that case, no digiKam tags will be returned. To return digiKam tags, two additional items are required (db_directory, db_filename). This is essential when using `IDfrom = "metadata"`. When using `IDfrom = "directory"`, it is optional, but allows to extract metadata tags assigned to videos in digiKam. This workaround is necessary because digiKam tags are not written into video metadata, but are only saved in the digiKam database. So in contrast to JPG images, they can not be extracted with ExifTool. It also requires that inDir is in your digiKam database.

The items of argument `video` are:

- **file_formats**: The video formats to extract (include "jpg" if you want JPG image metadata)
- **dateTimeTag**: the metadata tag to extract date/time from (use `exifTagNames` to find out which tag is suitable)
- **db_directory**: The directory containing digiKam database (optional if `IDfrom = "directory"`)
- **db_filename**: The digiKam database file in `db_directory` (optional if `IDfrom = "directory"`)

See the examples below for how to specify the argument `video`.

**Value**

A data frame containing species records and additional information about stations, date, time and (optionally) further metadata.

**Warning**

Custom image metadata must be organised hierarchically (tag group - tag; e.g. "Species": "Leopard Cat"). Detailed information on how to set up and use metadata tags can be found in vignette 2: Species and Individual Identification.

Custom image metadata tags must be written to the images. The function cannot read tags from .xmp sidecar files. Make sure you set the preferences accordingly. In DigiKam, go to Settings/Configure digiKam/Metadata. There, make sure "Write to sidecar files" is unchecked.

Please note the section about defining argument `timeZone` in the vignette on data extraction (accessible via vignette("DataExtraction") or online (https://cran.r-project.org/package=camtrapR/vignettes/camtrapr3.html)).
Note

The results of a number of other function will depend on the output of this function (namely on the arguments exclude for excluding species and minDeltaTime/ deltaTimeComparedTo for temporal independence):

- `detectionMaps`
- `detectionHistory`
- `activityHistogram`
- `activityDensity`
- `activityRadial`
- `activityOverlap`
- `activityHistogram`
- `surveyReport`

Author(s)

Juergen Niedballa

References

Phil Harvey's ExifTool https://exiftool.org/

Examples

```r
## Not run: # the examples take too long to pass CRAN tests

# set directory with camera trap images in station directories
wd_images_ID_species <- system.file("pictures/sample_images_species_dir",
  package = "camtrapR")

if (Sys.which("exiftool") !=""){
  # only run these examples if ExifTool is available

  rec_table1 <- recordTable(inDir = wd_images_ID_species,
    IDfrom = "directory",
    minDeltaTime = 60,
    deltaTimeComparedTo = "lastRecord",
    writecsv = FALSE,
    additionalMetadataTags = c("EXIF:Model", "EXIF:Make")
  )

  # note argument additionalMetadataTags: it contains tag names as returned by function exifTagNames

  rec_table2 <- recordTable(inDir = wd_images_ID_species,
    IDfrom = "directory",
    minDeltaTime = 60,
    deltaTimeComparedTo = "lastRecord",
    exclude = "UNID",
    writecsv = FALSE,
    additionalMetadataTags = c("EXIF:Model", "EXIF:Make")
  )

  # note argument additionalMetadataTags: it contains tag names as returned by function exifTagNames
```
writecsv = FALSE, timeZone = "Asia/Kuala_Lumpur",
additionalMetadataTags = c("EXIF:Model", "EXIF:Make", "NonExistingTag"),
eventSummaryColumn = "EXIF:Make",
eventSummaryFunction = "unique"
)

# note the warning that the last tag in "additionalMetadataTags" ("NonExistingTag") was not found

any(rec_table1$Species == "UNID")  # TRUE
any(rec_table2$Species == "UNID")  # FALSE

# here's how the removeDuplicateRecords argument works

rec_table3a <- recordTable(inDir = wd_images_ID_species,
IDfrom = "directory",
minDeltaTime = 0,
exclude = "UNID",
timeZone = "Asia/Kuala_Lumpur",
removeDuplicateRecords = FALSE)

rec_table3b <- recordTable(inDir = wd_images_ID_species,
IDfrom = "directory",
minDeltaTime = 0,
exclude = "UNID",
timeZone = "Asia/Kuala_Lumpur",
removeDuplicateRecords = TRUE)

anyDuplicated(rec_table3a[, c("Station", "Species", "DateTimeOriginal")])  # got duplicates
anyDuplicated(rec_table3b[, c("Station", "Species", "DateTimeOriginal")])  # no duplicates

# after removing duplicates, both are identical:
whichAreDuplicated <- which(duplicated(rec_table3a[,c("Station", "Species", "DateTimeOriginal")]))
all(rec_table3a[-whichAreDuplicated,] == rec_table3b)

### extracting species IDs from metadata

wd_images_ID_species_tagged <- system.file("pictures/sample_images_species_tag",
package = "camtrapR")

rec_table4 <- recordTable(inDir = wd_images_ID_species_tagged,
IDfrom = "metadata",
metadataSpeciesTag = "Species",
exclude = "unidentified")

### Including videos
recordTableIndividual

Generate a single-species record table with individual identification from camera trap images or videos

Description

The function generates a single-species record table containing individual IDs, e.g. for (spatial) capture-recapture analyses. It prepares input for the function spatialDetectionHistory.

Usage

recordTableIndividual(inDir, hasStationFolders, IDfrom, cameraID, camerasIndependent, minDeltaTime = 0, deltaTimeComparedTo, timeZone, ...)

# sample videos are not included in package

# with videos, IDfrom = "directory", not extracting digiKam metadata

rec_table4 <- recordTable(inDir = wd_images_ID_species,
                          IDfrom = "directory",
                          video = list(file_formats = c("jpg", "mp4"), dateTimeTag = "QuickTime:CreateDate")
)

# with videos, IDfrom = "metadata", extracting digiKam metadata

rec_table5 <- recordTable(inDir = wd_images_ID_species,
                          IDfrom = "metadata",
                          metadataSpeciesTag = "Species",
                          video = list(file_formats = c("jpg", "mp4", "avi", "mov"), dateTimeTag = "QuickTime:CreateDate",
                                      db_directory = "C:/Users/YourName/Pictures",
                                      db_filename = "digikam4.db")
)

} else {
  # show function output if ExifTool is not available
  message("ExifTool is not available. Cannot test function. Loading recordTableSample instead")
  data(recordTableSample)
}

## End(Not run)
Arguments

inDir character. Directory containing images of individuals. Must end with species name (e.g. "/speciesImages/Clouded Leopard")

hasStationFolders logical. Does inDir have station subdirectories? If TRUE, station IDs will be taken from directory names. If FALSE, they will be taken from image filenames (requires images renamed with imageRename).

IDfrom character. Read individual ID from image metadata ("metadata") or from directory names ("directory")?

cameraID character. Should the function look for camera IDs in the image file names? If so, set to 'filename'. Requires images renamed with imageRename. If missing, no camera ID will be assigned and it will be assumed there was 1 camera only per station.

camerasIndependent logical. If TRUE, cameras at a station are assumed to record individuals independently. If FALSE, cameras are assumed to be non-independent (e.g. in pairs). Takes effect only if there was more than 1 camera per station and cameraID = "filename".

minDeltaTime numeric. time difference between observation of the same individual at the same station/camera to be considered independent (in minutes)

deltaTimeComparedTo character. For two records to be considered independent, must the second one be at least minDeltaTime minutes after the last independent record of the same individual ("lastIndependentRecord"), or minDeltaTime minutes after the last record ("lastRecord")?

timeZone character. Must be a value returned by OlsonNames

stationCol character. Name of the camera trap station column in the output table.

writecsv logical. Should the individual record table be saved as a .csv file?

outDir character. Directory to save csv file to. If NULL and writecsv = TRUE, the output csv will be written to inDir.

metadataHierarchyDelimiter character. The character delimiting hierarchy levels in image metadata tags in field "HierarchicalSubject". Either "|" or ":".
metadataIDTag character. In custom image metadata, the individual ID tag name.
additionalMetadataTags character. additional camera model-specific metadata tags to be extracted. (If possible specify tag groups as returned by `exifTagNames`)
removeDuplicateRecords logical. If there are several records of the same individual at the same station (also same camera if cameraID is defined) at exactly the same time, show only one?
returnFileNamesMissingTags logical. If species are assigned with metadata and images are not tagged, return a few file names of these images as a message?
eventSummaryColumn character. A column in the record table (e.g. from a metadata tag) by to summarise non-independent records (those within `minDeltaTime` of a given record) with a user-defined function (`eventSummaryFunction`)
eventSummaryFunction character. The function by which to summarise `eventSummaryColumn` of non-independent records, e.g. "sum", "max" (optional)
video list. Contains information on how to handle video data (optional). See details.

Details

The function can handle a number of different ways of storing images and videos. In every case, images need to be stored in a species directory first (e.g. using function `getSpeciesImages`). Station subdirectories are optional. Camera subdirectories are not supported. This directory structure can be created easily with function `getSpeciesImages`.

As with species identification, individuals can be identified in 2 different ways: by moving images into individual directories ("Species/Station/Individual/XY.JPG" or "Species/Individual/XY.JPG") or by metadata tagging (without the need for individual directories: "Species/XY.JPG" or "Species/Station/XY.JPG").

`minDeltaTime` is a criterion for temporal independence of records of an individual at the same station/location. Setting it to 0 will make the function return all records. `camerasIndependent` defines if the cameras at a station are to be considered independent (e.g. FALSE if both cameras face each other and possibly TRUE if they face different trails). `stationCol` is the station column name to be used in the resulting table. Station IDs are read from the station directory names if `hasStationFolders = TRUE`. Otherwise, the function will try to extract station IDs from the image filenames (requires images renamed with `imageRename`).

If individual IDs were assigned with image metadata tags, `metadataIDTag` must be set to the name of the metadata tag group used for individual identification. `metadataHierarchyDelimiter` is "|" for images tagged in DigiKam and images tagged in Adobe Bridge/ Lightroom with the default settings. Manufacturer-specific Exif metadata tags such as "AmbientTemperature" or "MoonPhase" can be extracted if specified in `additionalMetadataTags`. Multiple names can be specified as a character vector as: `c(Tag1,Tag2,...)`. Because they are not standardized, function `exifTagNames` provides a vector of all available tag names. The metadata tags thus extracted may be used as individual covariates in spatial capture-recapture models.

`eventSummaryColumn` and `eventSummaryFunction` can be used to extract summary statistics for independent sampling events. For example, you assigned a "count" tag to your images, indicating the number of individuals in a picture. In a sequence of pictures taken within 1 minute,
most pictures show one individual, but one image shows two individuals. You tagged the images accordingly (count = 1 or count = 2) and run `recordTable`. Set `eventSummaryColumn = "count"` and `eventSummaryFunction = "max"` to obtain the maximum number of count in all images within `minDeltaTime` minutes of a given record. The results is in a new column, in this example `count_max`. You can also calculate several statistics at the same time, by supplying vectors of values. e.g. `eventSummaryColumn = c("count","count","camera")` and `eventSummaryFunction = c("min","max","unique")` to get minimum and maximum count and all unique camera IDs for that event. Note that `eventSummaryColumn` and `eventSummaryFunction` must be of same length.

Argument `video` is analogous to `recordTable`, a named list with 2 or 4 items. 2 items (`file_formats`, `dateTimeTag`) are always required, and are sufficient if `IDfrom = "directory"`. In that case, no digiKam tags will be returned. To return digiKam tags, two additional items are required (`db_directory`, `db_filename`). This is essential when using `IDfrom = "metadata"`. When using `IDfrom = "directory"`, it is optional, but allows to extract metadata tags assigned to videos in digiKam. This workaround is necessary because digiKam tags are not written into video metadata, but are only saved in the digiKam database. So in contrast to JPG images, they can not be extracted with ExifTool. It also requires that `inDir` is in your digiKam database.

The items of argument `video` are:

- `file_formats` The video formats to extract (include "jpg" if you want JPG image metadata)
- `dateTimeTag` the metadata tag to extract date/time from (use `exifTagNames` to find out which tag is suitable)
- `db_directory` The directory containing digiKam database (optional if `IDfrom = "directory"`)
- `db_filename` The digiKam database file in `db_directory` (optional if `IDfrom = "directory"`)

See the example below for how to specify the argument `video`.

**Value**

A data frame containing species records with individual IDs and additional information about stations, date, time and (optionally) further metadata.

**Warning**

Be sure to read the section on individual identification in the package vignette (https://CRAN.R-project.org/package=camtrapR/vignettes/camtrapr2.html).

If you use image metadata tags for identification, the tags must be written to the image metadata. The function cannot read tags from .xmp sidecar files. Make sure you set the preferences of your image management software accordingly. In DigiKam, go to Settings/Configure digiKam/Metadata. There, make sure "Write to sidecar files" is unchecked.

Please note the section about defining argument `timeZone` in the vignette on data extraction (accessible via vignette("DataExtraction") or online (https://cran.r-project.org/package=camtrapR/vignettes/camtrapr3.html)).

**Author(s)**

Juergen Niedballa
recordTableIndividual

References

Phil Harvey’s ExifTool [https://exiftool.org/](https://exiftool.org/)

Examples

```r
## Not run: # the examples run too long to pass CRAN tests

wd_images_ID_individual <- system.file("pictures/sample_images_indiv_tag/LeopardCat", package = "camtrapR")
# missing space in species = "LeopardCat" is because of CRAN package policies
# note argument additionalMetadataTags: contains tag names as returned by function exifTagNames

if (Sys.which("exiftool") != ""){
  # only run these examples if ExifTool is available
  rec_table_pbe <- recordTableIndividual(inDir = wd_images_ID_individual,
                                          minDeltaTime = 60,
                                          deltaTimeComparedTo = "lastRecord",
                                          hasStationFolders = FALSE,
                                          IDfrom = "metadata",
                                          camerasIndependent = FALSE,
                                          writecsv = FALSE,
                                          metadataIDTag = "individual",
                                          additionalMetadataTags = c("EXIF:Model", "EXIF:Make"),
                                          timeZone = "Asia/Kuala_Lumpur")
}

# extracting some example summary stats too
# a nonsensical example, get all unique cameras with which the event was photographed

rec_table_pbe2 <- recordTableIndividual(inDir = wd_images_ID_individual,
                                        minDeltaTime = 60,
                                        deltaTimeComparedTo = "lastRecord",
                                        hasStationFolders = FALSE,
                                        IDfrom = "metadata",
                                        camerasIndependent = FALSE,
                                        writecsv = FALSE,
                                        metadataIDTag = "individual",
                                        additionalMetadataTags = c("EXIF:Model", "EXIF:Make"),
                                        timeZone = "Asia/Kuala_Lumpur",
                                        eventSummaryColumn = "EXIF:Make",
                                        eventSummaryFunction = "unique")

### Video example (the sample data don’t contain a video, this is hypothetical)
# with JPG, video mp4, avi, mov, ID = metadata

rec_table_ind_video <- recordTableIndividual(inDir = wd_images_ID_individual,
                                          hasStationFolder = FALSE,
                                          IDfrom = "metadata",
                                          metadataIDTag = "individual")
```
recordTableIndividualSample

Sample single-species record table with custom metadata from camera trap images

Description

Sample single-species record table with individual IDs from the tagged sample images in the package. Generated with function recordTableIndividual.

Usage

data(recordTableIndividualSample)

Format

A data frame with 21 rows and 17 variables

Details

The variables are as follows:

- Station. Camera trap station ID
- Species. Species ID
- Individual. Individual ID
- DateTimeOriginal. Date and time as extracted from image
- Date. record date
- Time. record time of day
- delta.time.secs. time difference to first species record at a station (seconds)
- delta.time.mins. time difference to first species record at a station (minutes)
- `delta.time.hours`. time difference to first species record at a station (hours)
- `delta.time.days`. time difference to first species record at a station (days)
- `Directory`. Image directory
- `FileName`. Image filename
- `HierarchicalSubject`. Content of the HierarchicalSubject image metadata tag
- `Model`. Camera model extracted from image metadata
- `Make`. Camera make extracted from image metadata
- `metadata_Species`. Content of custom image metadata tag "Species" (see HierarchicalSubject)
- `metadata_individual`. Content of custom image metadata tag "individual" (see HierarchicalSubject)

---

**Description**

Sample single-species multi-season record table with individual IDs from the tagged sample images in the package. Generated with function `recordTableIndividual`, then duplicated to simulate a second year.

**Usage**

data(recordTableIndividualSampleMultiSeason)

**Format**

A data frame with 31 rows and 17 variables

**Details**

The variables are as follows:

- `Station`. Camera trap station ID
- `Species`. Species ID
- `Individual`. Individual ID
- `DateTimeOriginal`. Date and time as extracted from image
- `Date`. Record date
- `Time`. Record time of day
- `delta.time.secs`. Time difference to first species record at a station (seconds)
- `delta.time.mins`. Time difference to first species record at a station (minutes)
- `delta.time.hours`. Time difference to first species record at a station (hours)
recordTableSample

- delta.time.days. time difference to first species record at a station (days)
- Directory. Image directory
- FileName. image filename
- HierarchicalSubject. content of the HierarchicalSubject image metadata tag
- Model. camera model extracted from image metadata
- Make. camera make extracted from image metadata
- metadata_Species. content of custom image metadata tag "Species" (see HierarchicalSubject)
- metadata_individual. content of custom image metadata tag "individual" (see HierarchicalSubject)

Examples

# example data were created as follows:
data(recordTableIndividualSample)

recordTableIndividualSample_season2 <- recordTableIndividualSample[1:10,]
recordTableIndividualSample_season2$DateTimeOriginal <- gsub("2009", "2010",
recordTableIndividualSample_season2$DateTimeOriginal)
recordTableIndividualSampleMultiSeason <- rbind(recordTableIndividualSample,
recordTableIndividualSample_season2)

________________________________________________________________________

recordTableSample

Sample species record table from camera trap images

Description

Sample species record table from camera trap images generated from the sample images in the package with the function recordTable.

Usage

data(recordTableSample)

Format

A data frame with 39 rows and 11 variables

Details

The variables are as follows:

- Station. Camera trap station ID
- Species. Species ID
- DateTimeOriginal. Date and time as extracted from image
- Date. record date
• Time. record time of day
• delta.time.secs. time difference to first species record at a station (seconds)
• delta.time.mins. time difference to first species record at a station (minutes)
• delta.time.hours. time difference to first species record at a station (hours)
• delta.time.days. time difference to first species record at a station (days)
• Directory. Image directory
• FileName. image filename

recordTableSampleMultiSeason

Sample multi-season species record table from camera trap images

Description
Sample multi-season species record table from camera trap images generated from the sample images in the package with the function recordTable. Season 2009 is the same as recordTableSample, season 2010 was simulated by adding 1 year to these records.

Usage
data(recordTableSampleMultiSeason)

Format
A data frame with 78 rows and 11 variables

Details
The variables are as follows:
• Station. Camera trap station ID
• Species. Species ID
• DateTimeOriginal. Date and time as extracted from image
• Date. record date
• Time. record time of day
• delta.time.secs. time difference to first species record at a station (seconds)
• delta.time.mins. time difference to first species record at a station (minutes)
• delta.time.hours. time difference to first species record at a station (hours)
• delta.time.days. time difference to first species record at a station (days)
• Directory. Image directory
• FileName. image filename
spatialDetectionHistory

**Examples**

```r
# data were created with the following code:

data(recordTableSample)
recordTableSample_season2 <- recordTableSample

# substitute 2009 with 2010
recordTableSample_season2$DateTimeOriginal <- gsub("2009", "2010",
  recordTableSample_season2$DateTimeOriginal)

# combine with season 2009
recordTableSampleMultiSeason <- rbind(recordTableSample, recordTableSample_season2)
```

**spatialDetectionHistory**

Generate a capthist object for spatial capture-recapture analyses from camera-trapping data

**Description**

This function generates spatial detection histories of individuals of a species for spatial capture-recapture analyses with package `secr`. Data are stored in a capthist object. The capthist object contains detection histories, camera-trap station location and possibly individual and station-level covariates. Detection histories can have adjustable occasion length and occasion start time (as in the function detectionHistory).

**Usage**

```r
spatialDetectionHistory(recordTableIndividual,
  species, camOp, CTtable, output,
  stationCol = "Station",
  speciesCol = "Species",
  sessionCol, Xcol, Ycol,
  stationCovariateCols, individualCol, individualCovariateCols,
  recordDateTimeCol = "DateTimeOriginal",
  recordDateTimeFormat = "%Y-%m-%d %H:%M:%S",
  occasionLength, minActiveDaysPerOccasion,
  occasionStartTime = 0, maxNumberDays,
  day1,
```
Arguments

recordTableIndividual  
data.frame. the record table with individual IDs created by `recordTableIndividual`

species  
character. the species for which to compute the detection history

camOp  
The camera operability matrix as created by `cameraOperation`

CTtable  
data.frame. contains station IDs and coordinates. Same as used in `cameraOperation`.

output  
character. Return individual counts ("count") or binary observations ("binary")?

stationCol  
character. name of the column specifying Station ID in `recordTableIndividual` and `CTtable`

speciesCol  
character. name of the column specifying species in `recordTableIndividual`

sessionCol  
character. name of the column specifying session IDs, either in `recordTableIndividual`  
or in `CTtable`. See 'Details' for more information. Session ID values must be a  
sequence of integer numbers beginning with 1 (i.e., 1,2,3,....).

Xcol  
character. name of the column specifying x coordinates in `CTtable`

Ycol  
character. name of the column specifying y coordinates in `CTtable`

stationCovariateCols  
character. name of the column(s) specifying station-level covariates in `CTtable`

individualCol  
character. name of the column specifying individual IDs in `recordTableIndividual`

individualCovariateCols  
character. name of the column(s) specifying individual covariates in `recordTableIndividual`

recordDateTimeCol  
character. name of the column specifying date and time in `recordTableIndividual`

recordDateTimeFormat  
format of column `recordDateTimeCol` in `recordTableIndividual`

occasionLength  
icenter. occasion length in days

minActiveDaysPerOccasion  
icenter. minimum number of active trap days for occasions to be included (optional)

occasionStartTime  
icenter. time of day (the full hour) at which to begin occasions.

maxNumberDays  
icenter. maximum number of trap days per station (optional)

day1  
character. When should occasions begin: station setup date ("station"), first day  
of survey ("survey"), a specific date (e.g. "2015-12-31")?

buffer  
icenter. Makes the first occasion begin a number of days after station setup. 
(optional)
spatialDetectionHistory

includeEffort logical. Include trapping effort (number of active camera trap days per station and occasion) as usage in capthist object?

scaleEffort logical. scale and center effort matrix to mean = 0 and sd = 1? Currently not used. Must be FALSE.

binaryEffort logical. Should effort be binary (1 if >1 active day per occasion, 0 otherwise)?

timeZone character. Must be a value returned by OlsonNames

makeRMarkInput logical. If FALSE, output will be a data frame for RMark. If FALSE or not specified, a secr capthist object

Details

The function creates a capthist object by combining three different objects: 1) a record table of identified individuals of a species, 2) a camera trap station table with station coordinates and 3) a camera operation matrix computed with cameraOperation. The record table must contain a column with individual IDs and optionally individual covariates. The camera trap station table must contain station coordinates and optionally station-level covariates. The camera operation matrix provides the dates stations were active or not and the number of active stations.

day1 defines if each stations detection history will begin on that station’s setup day (day1 = "station") or if all station’s detection histories have a common origin (the day the first station was set up if day1 = "survey" or a fixed date if, e.g. day1 = "2015-12-31").

includeEffort controls whether an effort matrix is computed or not. If TRUE, effort will be used for object usage information in a traps. binaryEffort makes the effort information binary. scaleEffort is currently not used and must be set to FALSE. The reason is that usage can only be either binary, or nonnegative real values, whereas scaling effort would return negative values.

The number of days that are aggregated is controlled by occasionLength. occasionStartTime can be used to make occasions begin another hour than midnight (the default). This may be relevant for nocturnal animals, in which 1 whole night would be considered an occasion. Output can be returned as individual counts per occasion (output = "count") or as binary observation (output = "binary").

Argument sessionCol can be used to a create multi-session capthist object. There are two different ways in which the argument is interpreted. It depends on whether a column with the name you specify in argument sessionCol exists in recordTableIndividual or in CTtable. If sessionCol is found in recordTableIndividual, the records will be assigned to the specified sessions, and it will be assumed that all camera trap station were used in all sessions. Alternatively, if sessionCol is found inCTtable, it will be assumed that only a subset of stations was used in each session, and the records will be assigned automatically (using the station IDs to identify which session they belong into). In both cases, session information must be provided as a sequence of integer numbers beginning with 1, i.e., you provide the session number directly in sessionCol. See session for more information about sessions in secr.

capthist objects (as created by spatialDetectionHistory for spatial capture-recapture analyses) expect the units of coordinates (Xcol and col in CTtable) to be meters. Therefore, please use a suitable coordinate system (e.g. UTM).

recordDateTimeFormat defaults to the "YYYY-MM-DD HH:MM:SS" convention, e.g. "2014-09-30 22:59:59". recordDateTimeFormat can be interpreted either by base-R via strptime or in lubridate via parse_date_time (argument "orders"). lubridate will be used if there are no "%" characters in recordDateTimeFormat.
For "YYYY-MM-DD HH:MM:SS", recordDateTimeFormat would be either ">%Y-%m-%d %H:%M:%S" or "ymd HMS". For details on how to specify date and time formats in R see `strptime` or `parse_date_time`.

**Value**

Output depends on argument `makeRMarkInput`:

- `makeRMarkInput = FALSE`
  - A `capthist` object
- `makeRMarkInput = TRUE`
  - A data frame for use in RMark

**Warning**

Please note the section about defining argument `timeZone` in the vignette on data extraction (accessible via `vignette("DataExtraction")` or online ([https://cran.r-project.org/package=camtrapR/vignettes/camtrapr3.html](https://cran.r-project.org/package=camtrapR/vignettes/camtrapr3.html))).

**Author(s)**

Juergen Niedballa

**See Also**

- secr
- RMark

**Examples**

data(recordTableIndividualSample)
data(camtraps)

```r
# create camera operation matrix (with problems/malfunction)
camop_problem <- cameraOperation(CTtable = camtraps,
   stationCol = "Station",
   setupCol = "Setup_date",
   retrievalCol = "Retrieval_date",
   writecsv = FALSE,
   hasProblems = TRUE,
   dateFormat = "%d/%m/%Y"
)

sdh <- spatialDetectionHistory(recordTableIndividual = recordTableIndividualSample,
   species = "LeopardCat",
   camOp = camop_problem,
   CTtable = camtraps,
   output = "binary",
   stationCol = "Station",
   speciesCol = "Species",
   Xcol = "utm_x",
   Ycol = "utm_y",
)```

individualCol = "Individual",
recordDateTimeCol = "DateTimeOriginal",
recordDateTimeFormat = "%Y-%m-%d %H:%M:%S",
occasionLength = 10,
day1 = "survey",
includeEffort = TRUE,
timeZone = "Asia/Kuala_Lumpur"
)

# missing space in species = "LeopardCat" was introduced by recordTableIndividual
# (because of CRAN package policies.
# In your data you can have spaces in your directory names)

summary(sdh)
plot(sdh, tracks = TRUE)

## multi-season capthist object
# see vignette "3. Extracting Data from Camera Trapping Images, creating occupancy & secr input"

data(camtrapsMultiSeason)
camtrapsMultiSeason$session[camtrapsMultiSeason$session == 2009] <- 1
camtrapsMultiSeason$session[camtrapsMultiSeason$session == 2010] <- 2

data(recordTableIndividualSampleMultiSeason)

# create camera operation matrix (with problems/malfunction)
camop_session <- cameraOperation(CTtable = camtrapsMultiSeason,
stationCol = "Station",
setupCol = "Setup_date",
sessionCol = "session",
retrievalCol = "Retrieval_date",
hasProblems = TRUE,
dateFormat = "%d/%m/%Y"
)

sdh_multi <- spatialDetectionHistory(recordTableIndividual = recordTableIndividualSampleMultiSeason,
species = "LeopardCat",
output = "binary",
camOp = camop_session,
CTtable = camtrapsMultiSeason,
stationCol = "Station",
speciesCol = "Species",
sessionCol = "session",
Xcol = "utm_x",
Ycol = "utm_y",
individualCol = "Individual",
recordDateTimeCol = "DateTimeOriginal",
recordDateTimeFormat = "%Y-%m-%d %H:%M:%S",
occasionLength = 10,
day1 = "survey",
includeEffort = TRUE,
timeZone = "Asia/Kuala_Lumpur",
stationCovariateCols = "utm_y", # example
surveyReport

Create a report about a camera trapping survey and species detections

Description

This function creates a report about a camera trapping survey and species records. It uses a camera trap station information table and a record table (generated with recordTable) as input. Output tables can be saved and a zip file for simple data sharing can be created easily.

Usage

surveyReport (recordTable,
  CTtable,
  speciesCol = "Species",
  stationCol = "Station",
  cameraCol,
  setupCol,
  retrievalCol,
  CTDateFormat = "%Y-%m-%d",
  CTHasProblems = FALSE,
  recordDateTimeCol = "DateTimeOriginal",
  recordDateTimeFormat = "%Y-%m-%d %H:%M:%S",
  Xcol,
  Ycol,
  sinkpath,
  makezip
)

Arguments

- **recordTable**: data.frame containing a species record table as given by recordTable
- **CTtable**: data.frame containing information about location and trapping period of camera trap stations (equivalent to camtraps)
- **speciesCol**: character. name of the column specifying Species ID in recordTable
- **stationCol**: character. name of the column specifying Station ID in CTtable and recordTable
- **cameraCol**: character. name of the column specifying Camera ID in CTtable and recordTable
- **setupCol**: character. name of the column containing camera setup dates in CTtable
- **retrievalCol**: character. name of the column containing camera retrieval dates in CTtable
surveyReport

CTDateFormat character. The format of columns setupCol and retrievalCol (and potential problem columns) in CTtable. Must be interpretable by either as.Date or the "orders" argument parse_date_time in lubridate.

CTHasProblems logical. Are there periods of camera malfunction specified in CTtable?

recordDateTimeCol character. The name of the column containing date and time of records in recordTable

recordDateTimeFormat character. The date/time format of column recordDateTimeCol in recordTable.

Xcol character. name of the column specifying x coordinates in CTtable. Used to create detection maps if makezip is TRUE. (optional)

Ycol character. name of the column specifying y coordinates in CTtable. Used to create detection maps if makezip is TRUE. (optional)

sinkpath character. The directory into which the survey report is saved (optional)

makezip logical. Create a zip file containing tables, plots and maps in sinkpath?

Details

dateFormat defaults to "YYYY-MM-DD", e.g. "2014-10-31". It can be specified either in the format required by strptime or the 'orders' argument in parse_date_time in lubridate. In the example above, "YYYY-MM-DD" would be specified as "%Y-%m-%d" or "ymd". If CTHasProblems is set to TRUE, the function tries to find columns ProblemX_from and ProblemX_to in CTtable (X designates numbers from 1 to n in which a camera or station was not operational). If there are no such columns all stations are assumed to have been operational uninterruptedly from setup to retrieval.

recordDateTimeFormat defaults to the "YYYY-MM-DD HH:MM:SS" convention, e.g. "2014-09-30 22:59:59". recordDateTimeFormat can be interpreted either by base-R via strptime or in lubridate via parse_date_time (argument "orders"). lubridate will be used if there are no "%" characters in recordDateTimeFormat.

For "YYYY-MM-DD HH:MM:SS", recordDateTimeFormat would be either "%Y-%m-%d %H:%M:%S" or "ymd HMS". For details on how to specify date and time formats in R see strptime or parse_date_time.

Value

An invisible list containing 5 data.frames.

survey_dates station and image date ranges, number of total and active trap nights, number of cameras per station

species_by_station species numbers by station

events_by_species number of events and stations by species

events_by_station number of events for every species by station (only species that were recorded)
events_by_station2
  number of events for all species at all stations (including species that were not recorded)

The output will be saved to a .txt file if sinkpath is defined.
If makezip is TRUE, a zip file will be created in sinkpath. It contains single-species activity plots, detection maps (if Xcol and Ycol are defined), the survey report tables, the record table and the camera trap station table, and an example R script.

Author(s)
Juergen Niedballa

See Also
recordTable

Examples

```
data(camtraps)
data(recordTableSample)

reportTest <- surveyReport (recordTable = recordTableSample,
                          CTtable = camtraps,
                          speciesCol = "Species",
                          stationCol = "Station",
                          setupCol = "Setup_date",
                          retrievalCol = "Retrieval_date",
                          CTDateFormat = "%d/%m/%Y",
                          recordDateTimeCol = "DateTimeOriginal",
                          recordDateTimeFormat = "%Y-%m-%d %H:%M:%S")

class(reportTest) # a list with
length(reportTest) # 5 elements

reportTest[[1]] # camera trap operation times and image date ranges
reportTest[[2]] # number of species by station
reportTest[[3]] # number of events and number of stations by species
reportTest[[4]] # number of species events by station
reportTest[[5]] # number of species events by station including 0s (non-observed species)

# with camera problems

reportTest_problem <- surveyReport (recordTable = recordTableSample,
                                       CTtable = camtraps,
                                       speciesCol = "Species",
                                       stationCol = "Station",
                                       setupCol = "Setup_date",
                                       retrievalCol = "Retrieval_date",
                                       CTDateFormat = "%d/%m/%Y",
                                       recordDateTimeCol = "DateTimeOriginal",
                                       recordDateTimeFormat = "%Y-%m-%d %H:%M:%S")
```
## timeShiftImages

Apply time shifts to JPEG image metadata

### Description

Change the values of digital timestamps in image metadata using ExifTool. If date/time of images were set incorrectly, they can be corrected easily in batch mode for further analyses. Please, always make a backup of your data before using this function to avoid data loss or damage. This is because ExifTool will make a copy of your images and applies the time shifts to the copies. The file extension of the original images (.JPG) will be renamed to “JPG_original”.

### Usage

```r
timeShiftImages (inDir, 
    hasCameraFolders, 
    timeShiftTable, 
    stationCol, 
    cameraCol, 
    timeShiftColumn, 
    timeShiftSignColumn, 
    undo = FALSE)
```
Arguments

inDir character. Name of directory containing station directories with images
hasCameraFolders logical. Do the station directories in inDir have camera subdirectories (e.g. "inDir/StationA/Camera1")?
timeShiftTable data.frame containing information about station-/camera-specific time shifts.
stationCol character. name of the column specifying Station ID in timeShiftTable
cameraCol character. name of the column specifying Camera ID in timeShiftTable (optional)
timeShiftColumn character. The name of the column containing time shift values in timeShiftTable
timeShiftSignColumn character. The name of the column with the direction of time shifts in timeShiftTable. Can only be "-" or "+". undo logical. Undo changes and restore the original images? Please be careful, this deletes any edited images if TRUE

Details

timeShiftTable is a data frame with columns for station ID, camera ID (optional), time shift value and direction of time shift (for an example see timeShiftTable). Images in inDir must be sorted into station directories. If hasCameraFolders = TRUE, the function expects camera subdirectories in the station directories and will only apply time shifts to the camera subdirectories specified by CameraCol in timeShiftTable. If hasCameraFolders = FALSE, shifts will be applied to the whole station directory (including potential subdirectories).

The values of timeShiftColumn must adhere to the following pattern: "YYYY:MM:DD HH:MM:SS" ("year:month:day hour:minute:second"). Examples: "1:0:0 0:0:0" is a shift of exactly 1 year and "0:0:0 12:10:01" 12 hours and 10 minutes and 1 second. Note that stating "00" may cause problems, so use "0" instead if an entry is zero.

timeShiftSignColumn signifies the direction of the time shift. "+" moves image dates into the future (i.e. the image date lagged behind the actual date) and "-" moves image dates back (if the image dates were ahead of actual time).

ExifTool stores the original images as .JPG_original files in the original file location. By setting undo = TRUE, any JPG files in the directories specified by timeShiftTable will be deleted and the original JPEGs will be restored from the JPG_original files. Please make a backup before using undo.

Years can have 365 or 366 days, and months 28 to 31 days. Here is how the function handles these (from the exiftool help page): "The ability to shift dates by Y years, M months, etc, conflicts with the design goal of maintaining a constant shift for all time values when applying a batch shift. This is because shifting by 1 month can be equivalent to anything from 28 to 31 days, and 1 year can be 365 or 366 days, depending on the starting date. The inconsistency is handled by shifting the first tag found with the actual specified shift, then calculating the equivalent time difference in seconds for this shift and applying this difference to subsequent tags in a batch conversion."
Value

A data.frame containing the information about the processed directories and the number of images.

Author(s)

Juergen Niedballa

References

https://exiftool.org/#shift

Examples

## Not run:

# copy sample images to temporary directory (so we don't mess around in the package directory)
wd_images_ID <- system.file("pictures/sample_images_species_dir", package = "camtrapR")
file.copy(from = wd_images_ID, to = tempdir(), recursive = TRUE)
wd_images_ID_copy <- file.path(tempdir(), "sample_images_species_dir")
data(timeShiftTable)

timeshift_run <- timeShiftImages(inDir = wd_images_ID_copy,
timeShiftTable = timeShiftTable,
stationCol = "Station",
hasCameraFolders = FALSE,
timeShiftColumn = "timeshift",
timeShiftSignColumn = "sign",
undo = FALSE)


timeshift_undo <- timeShiftImages(inDir = wd_images_ID_copy,
timeShiftTable = timeShiftTable,
stationCol = "Station",
hasCameraFolders = FALSE,
timeShiftColumn = "timeshift",
timeShiftSignColumn = "sign",
undo = TRUE)

## End(Not run)
timeShiftTable

Sample camera trap time shift table

Description
Sample camera trap time shift table

Usage
data(timeShiftTable)

Format
A data frame with 2 rows and 4 variables

Details
If image Exif metadata timestamps are wrong systematically (e.g. because camera system time was not set after changing batteries), it can be corrected using a data frame in the following format using function `timeShiftImages`. For details on data format, please see `timeShiftImages`.

The variables are as follows:

- Station. Camera trap station ID
- camera. Camera trap ID (optional)
- timeshift. time shift amount to be applied
- sign. direction of time shift
\textbf{Index}

* datasets

- camtraps, 21
- camtrapsMultiSeason, 22
- recordTableIndividualSample, 59
- recordTableIndividualSampleMultiSeason, 60
- recordTableSample, 61
- recordTableSampleMultiSeason, 62
- timeShiftTable, 74

* package

- camtrapR-package, 3
- activityDensity, 4, 6, 9, 11, 14, 52
- activityHistogram, 4, 7, 8, 14, 52
- activityOverlap, 4, 7, 9, 10, 14, 52
- activityRadial, 4, 7, 9, 12, 52
- addCopyrightTag, 4, 15
- appendSpeciesNames, 4, 16
- as.Date, 33
- cameraOperation, 4, 18, 32, 33, 64, 65
- camtrapR (camtrapR-package), 3
- camtrapR-package, 3
- camtraps, 5, 19, 21, 50, 68
- camtrapsMultiSeason, 5, 22
- capthist, 19, 22, 23, 63, 65, 66
- checkSpeciesIdentification, 4, 17, 24
- checkSpeciesNames, 4, 26
- createSpeciesFolders, 4, 28
- createStationFolders, 3, 30
- CRS, 38
- densityPlot, 6
- detectionHistory, 4, 18, 20, 31, 52, 63
- detectionMaps, 4, 36, 52
- exifTagNames, 4, 39, 50, 51, 56, 57
- exiftoolPath, 4, 41
- fixDateTimeOriginal, 3, 42

- get_tsn, 27
- getSpeciesImages, 4, 43, 56
- hist, 8
- imageRename, 3, 29, 43, 45, 46, 49, 55, 56
- OlsonNames, 33, 49, 55, 65
- overlapEst, 10, 11
- overlapPlot, 10, 11
- parse_date_time, 7, 9, 11, 13, 14, 19, 33, 65, 66, 69
- radial.plot, 12, 13
- recordTable, 4, 6, 8, 10, 12, 13, 32, 37, 40, 43, 44, 47, 48, 57, 61, 62, 68, 70
- recordTableIndividual, 4, 40, 54, 59, 60, 64
- recordTableIndividualSample, 5, 59
- recordTableIndividualSampleMultiSeason, 5, 60
- recordTableSample, 5, 61, 62
- recordTableSampleMultiSeason, 5, 62
- secr, 63
- session, 65
- spatialDetectionHistory, 4, 18, 20, 22, 23, 33, 54, 63, 65
- strptime, 7, 9, 11, 13, 14, 19, 33, 65, 66, 69
- surveyReport, 4, 52, 68
- timeShiftImages, 3, 71, 74
- timeShiftTable, 5, 71, 74
- traps, 65
- unmarked, 31
- unmarkedMultFrame, 19, 33
- usage, 65