Package 'camtrapR'

June 24, 2016

Type Package

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

Version 0.99.2

Date 2016-06-22

Depends R (>= 3.1.0)

Imports methods, overlap, secr, rgdal, sp, taxize

Suggests unmarked, knitr, rmarkdown

VignetteBuilder knitr

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

Description Management of and data extraction from camera trap photographs in wildlife studies. The package provides a workflow for storing and sorting camera trap photographs, computes record databases and detection/non-detection matrices for occupancy and spatial capture-recapture analyses with great flexibility. In addition, it provides simple mapping functions (number of species, number of independent species detections by station) and can visualise activity data.

License GPL (>= 2)

NeedsCompilation no

Author Juergen Niedballa [aut, cre],
Alexandre Courtiol [aut],
Rahel Sollmann [aut],
John Mathai [ctb],
Seth Timothy Wong [ctb],
An The Truong Nguyen [ctb],
Azlan bin Mohamed [ctb],
Andrew Tilker [ctb],
Andreas Wilting [ctb, ths]

Maintainer Juergen Niedballa `<niedballa@izw-berlin.de>`

Repository CRAN

Date/Publication 2016-06-24 07:39:57

R topics documented:

`camtrapR-package` *Overview of the functions in the camtrapR package*

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, 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 it 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 http://www.sno.phy.queensu.ca/~phil/exiftool/) 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 information 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.

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
`timeShiftImages`	Apply time shifts to JPEG images
`imageRename`	Copy and rename images based on station ID and image creation date
`appendSpeciesNames`	Add or remove species names from image filenames

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 function 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
`recordTableIndividual`	Create a single-species record table from camera trap images 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
`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
`spatialDetectionHistory`	Detection histories of individuals for spatial capture-recapture analyses
`surveyReport`	Create a report about camera trap surveys and species detections

Sample data

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

Author(s)

Juergen Niedballa

Maintainer:Juergen Niedballa <niedballa@izw-berlin.de>

References

Lemon, J. (2006) Plotrix: a package in the red light district of R. R-News, 6(4): 8-12.
Mike Meredith and Martin Ridout (2014). overlap: Estimates of coefficient of overlapping for animal activity patterns. R package version 0.2.4. http://CRAN.R-project.org/package=overlap
Phil Harvey's ExifTool http://www.sno.phy.queensu.ca/~phil/exiftool/

See Also

overlap unmarked secr plotrix taxize

`activityDensity` *Plot kernel density estimation of single-species activity*

Description

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

Usage

```
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`

Details

`species` must be in the `speciesCol` of `recordTable`.

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

Martin Ridout and Matthew Linkie (2009). Estimating overlap of daily activity patterns from camera trap data. Journal of Agricultural, Biological and Environmental Statistics, 14(3), 322-337
Mike Meredith and Martin Ridout (2014). overlap: Estimates of coefficient of overlapping for animal activity patterns. R package version 0.2.4. http://CRAN.R-project.org/package=overlap

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`

Details

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

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,
  ...
)
```

Arguments

`recordTable`	data.frame. the record table created by `recordTable`
`speciesA`	Name of species 1
`speciesB`	Name of species 2
`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`
`createDir`	logical. Create `plotDirectory`?
`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 `overlapPlot`

Details

`...` can be graphical parameters, e.g. `linetype`, `linewidth`, `linecol`.

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

Mike Meredith and Martin Ridout (2014). overlap: Estimates of coefficient of overlapping for animal activity patterns. R package version 0.2.4. http://CRAN.R-project.org/package=overlap
Ridout, M.S. and Linkie, M. (2009) Estimating overlap of daily activity patterns from camera trap data. Journal of Agricultural, Biological and Environmental Statistics, 14, 322-337.

See Also

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

Examples

```
# 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
activityOverlap (recordTable = recordTableSample,
                 speciesA    = speciesA_for_activity,
                 speciesB    = speciesB_for_activity,
                 writePNG    = FALSE,
                 plotR       = TRUE,
                 createDir   = FALSE,
                 pngMaxPix   = 1000,
                 linecol     = c("red", "blue"),
                 linewidth   = c(3,3),
                 add.rug     = TRUE
)
```

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.

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.

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

Lemon, J. (2006) Plotrix: a package in the red light district of R. R-News, 6(4): 8-12.
http://CRAN.R-project.org/package=plotrix

See Also

activityDensity, activityHistogram, activityOverlap

Examples

```
# 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"
)
```

---

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,
  metadataHierarchyDelimitor = "|",
  removeNames = FALSE,
  writecsv = FALSE
)
```

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").
`metadataHierarchyDelimitor`	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", package = "camtrapR")
file.copy(from = wd_images_ID, to = getwd(), recursive = TRUE)
wd_images_ID_copy <- file.path(getwd(), "sample_images")

# 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,
  cameraCol,
  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.
`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`. Should be interpretable by `as.Date`
`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". See `strptime` for formatting options. 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 camers (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.

Value

A matrix. Row names indicate Station IDs (camera ID 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"
)

camop_no_problem
camop_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.

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

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

Arguments

`inDir`	character. Directory containing identified 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 directories in `inDir` have camera subdirectories (e.g. "inDir/StationA/Camera1" or "inDir/StationA/Camera1/Species1")?
`metadataSpeciesTag`	character. The species ID tag name in image metadata (if IDfrom = "metadata").
`metadataSpeciesTagToCompare`	character. A second species ID tag name in image metadata (if IDfrom = "metadata"). For comparing double observer species identification.

`metadataHierarchyDelimitor`	character. The character delimiting hierarchy levels in image metadata tags in field "HierarchicalSubject". Either "|" or ":"
`maxDeltaTime`	numeric. Maximum time interval between images to be returned (in seconds)
`excludeSpecies`	character. vector of species to exclude from checks
`stationsToCheck`	character. vector of stations to be checked (optionally)
`writecsv`	logical. Should the resulting data.frame be saved as a .csv?

Details

Images may accidentally be misidentified by assigning wrong species tags or by moving them into wrong species directories. Imagine your cameras take sequences of images each time they are triggered and one image of the sequence is misidentified. The time difference between these images (that have different species assigned to them) will be very small, usually a few seconds. This function will return all these images for you to check if they were identified correctly.

If multiple observers identify images independently 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.

Value

A `list` containing 2 data frames. The first contains a data frame with images file names, directories, time stamp and species ID that were taken within `maxDeltaTime` seconds of another species image at a particular station. The second data frame contains images with conflicting species IDs (if `IDfrom = "metadata"` and `metadataSpeciesTagToCompare` is defined)

Note

The function will not be able to find "isolated" images, i.e. images that were misidentified, but were not part of a sequence of images. Likewise, if all images of a sequence were misidentified, they cannot be found either. From version 0.99.0, the function can also handle images identied with metadata tags.

Author(s)

Juergen Niedballa

Examples

```
wd_images_ID <- system.file("pictures/sample_images", package = "camtrapR")

if (Sys.which("exiftool") != ""){        # only run this example if ExifTool is available
check.folders <- checkSpeciesIdentification(inDir             = wd_images_ID,
                                            IDfrom            = "directory",
                                            hasCameraFolders  = FALSE,
                                              maxDeltaTime      = 120,
                                              writecsv          = FALSE)

check.folders   # In the example, 2 different species were photographed within 2 minutes.

# now exclude one of these 2 species
check.folders2 <- checkSpeciesIdentification(inDir             = wd_images_ID,
                                             IDfrom            = "directory",
                                             hasCameraFolders  = FALSE,
                                             maxDeltaTime      = 120,
                                             excludeSpecies    = "EGY",
                                             writecsv          = FALSE)

check.folders2   # the data frame is empty

# now we check only one station
check.folders3 <- checkSpeciesIdentification(inDir             = wd_images_ID,
                                             IDfrom            = "directory",
                                             hasCameraFolders  = FALSE,
                                             maxDeltaTime      = 120,
                                             stationsToCheck   = "StationB",
                                             writecsv          = FALSE)
check.folders3   # the data frame is empty

}
```

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,
                                          ask          = 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

```
## 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.

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:

# 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)

# 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

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

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`
`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`	format of column `recordDateTimeCol` in `recordTable`
`occasionLength`	integer. occasion length in days
`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 an argument of `OlsonNames`
`writecsv`	logical. Should the detection history be saved as a .csv?
`outDir`	character. Directory into which detection history .csv file is saved

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 "

`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 relevant 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. `DateTimeFormat` defaults to "%Y-%m-%d %H:%M:%S", e.g. "2014-09-30 22:59:59". For details on how to specify date and time formats in R see `strptime`.

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`

Author(s)

Juergen Niedballa

Examples

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

# load station information
data(camtraps)

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

```
if (Sys.which("exiftool") != ""){       # only run this function if ExifTool is available
recordTableSample <- recordTable(inDir               = wd_images_ID,
                                 IDfrom              = "directory",
                                 minDeltaTime        = 60,
                                 deltaTimeComparedTo = "lastRecord",
                                 exclude             = "NO_ID",
                                 timeZone            = "Asia/Kuala_Lumpur"
)
} else {
data(recordTableSample)
}

# 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

# with effort
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[[1]]  # detection history
DetHist2[[2]]  # effort
```

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

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

```
detectionMaps(CTtable,
  recordTable,
  Xcol,
  Ycol,
  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`
`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**. `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
)
```

exifTagNames   *Show Exif metadata tags and tag names from JPEG images*

Description

The function will return sample metadata and tag names of Exif metadata of JPEG images. It uses the first JPEG image it finds in a subdirectory of the specified directory.

Usage

```
exifTagNames(inDir,
  whichSubDir = 1,
  returnMetadata = FALSE)
```

Arguments

`inDir`	character. Directory containing camera trap images sorted into station subdirectories (e.g. inDir/StationA/)
`whichSubDir`	integer. The number of the subdirectory of `inDir` in which to look for an image
`returnMetadata`	logical. Return actual metadata (TRUE) or metadata tag names only (FALSE)

Details

Many digital cameras record information such as ambient temperature or moon phase under maker-specific tag names in Exif metadata of JPEG images. The tag names must be known to be passed to the functions `recordTable` and `recordTableIndividual` via the `additionalMetadataTags` argument to extract those information from images and add them to the record tables.

Value

A character vector containing available metadata or metadata tag names.

Author(s)

Juergen Niedballa

References

Phil Harvey's ExifTool http://www.sno.phy.queensu.ca/~phil/exiftool/

See Also

recordTable

Examples

```
if (Sys.which("exiftool") != ""){        # only run this example if ExifTool is available
wd_images_ID <- system.file("pictures/sample_images", package = "camtrapR")

# return tag names only
exifTagNames(inDir          = wd_images_ID,
             returnMetadata = FALSE)

# return tag names and metadata
exifTagNames(inDir          = wd_images_ID,
             returnMetadata = TRUE)
}
```

---

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

```
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

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

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

`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. Species IDs can be read from species directories or from metadata. Earlier in the workflow, images should have been renamed (with `imageRename`) to give images unique file names based on station ID and date/time.

Usage

```
getSpeciesImages(species,
  inDir,
  outDir,
  createStationSubfolders = FALSE,
  IDfrom,
  metadataSpeciesTag,
  metadataHierarchyDelimitor = "|"
  )
```

Arguments

`species`	character. Species whose images are to be fetched
`inDir`	character. Directory containing identified (species level) camera trap images sorted into station subdirectories (e.g. inDir/StationA/)
`outDir`	character. Directory in which to save species images (in a species subdirectory)

`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")?
`metadataSpeciesTag`	character. The species ID tag name in image metadata (if IDfrom = "metadata").
`metadataHierarchyDelimitor`	character. The character delimiting hierarchy levels in image metadata tags in field "HierarchicalSubject". Either "|" or ":" (if IDfrom = "metadata").

Details

The function can derive species IDs both from a directory structure like this > inDir/Station/Species or > inDir/Station/Camera/Species,

or from species metadata tags. In the latter case, only station directories are needed. In any case, the argument `species` must match species IDs (either 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 file locations and the copy status (`copy_ok`; TRUE if copying was successful).

Author(s)

Juergen Niedballa

Examples

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

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 — *Copy and rename images based on camera trap station ID and creation date*

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,
  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")?
`copyImages`	logical. Copy images to `outDir`?
`writecsv`	logical. Save a data frame with a summary as a .csv?

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 `http://www.sno.phy.queensu.ca/~phil/exiftool/`

Examples

```
### "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")

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

# because copyImages = FALSE, outDir does not need to be defined
renaming.table <- imageRename(inDir               = wd_images_raw,
                              hasCameraFolders = FALSE,
                              copyImages          = FALSE,
                              writecsv            = FALSE
  )
  } else {
  message("ExifTool is not available. Cannot test function")
  }
  ## Not run:

  # define image directories

  # raw image location
wd_images_raw <- system.file("pictures/raw_images", package = "camtrapR")
  # destination for renamed images
wd_images_raw_renamed <- file.path(tempdir(), "raw_images_renamed")


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

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

  list.files(wd_images_raw_renamed, recursive = TRUE)


## End(Not run)
```

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

Description

Generates a record table from camera trap images. Images 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

```
recordTable(inDir,
  IDfrom,
  cameraID,
  camerasIndependent,
  exclude,
  minDeltaTime = 0,
  deltaTimeComparedTo,
  timeZone,
  stationCol,
  writecsv = FALSE,
  outDir,
  metadataHierarchyDelimitor = "|",
  metadataSpeciesTag,
  additionalMetadataTags
)
```

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 an argument of `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`.
`metadataHierarchyDelimitor`	character. The character delimiting hierarchy levels in image metadata tags in field "HierarchicalSubject". Either "|" or ":".
`metadataSpeciesTag`	character. In custom image metadata, the species ID tag name.
`additionalMetadataTags`	character. Additional camera model-specific metadata tags to be extracted.

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.

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, e.g.

- Species # set `metadataSpeciesTag` to "Species" * Leopard Cat * Malay Civet * Moonrat

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.

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 http://www.sno.phy.queensu.ca/~phil/exiftool/

Examples

```
wd_images_ID <- system.file("pictures/sample_images", package = "camtrapR")

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

rec.db1 <- recordTable(inDir                  = wd_images_ID,
                       IDfrom                 = "directory",
                       minDeltaTime           = 60,
                       deltaTimeComparedTo    = "lastRecord",
                       writecsv               = FALSE,
                       additionalMetadataTags = c("Model", "Make")
)

rec.db2 <- recordTable(inDir                  = wd_images_ID,
                       IDfrom                 = "directory",
                       minDeltaTime           = 60,
                       deltaTimeComparedTo    = "lastRecord",
                       exclude                = "NO_ID",
                       writecsv               = FALSE,
                       timeZone               = "Asia/Kuala_Lumpur",
                   additionalMetadataTags = c("Model", "Make", "NonExistingMetadataTag")
)

any(rec.db1$Species == "NO_ID")
any(rec.db2$Species == "NO_ID")

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

recordTableIndividual *Generate a single-species record table with individual identification from camera trap images*

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,
  stationCol,
  writecsv = FALSE,
  outDir,
  metadataHierarchyDelimitor = "|",
  metadataIDTag,
  additionalMetadataTags
)
```

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") of 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. `timeZone` must be an argument of `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`.
`metadataHierarchyDelimitor`	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.

Details

The function can handle a number of different ways of storing images. 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 allowed. 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. `metadataHierarchyDelimitor` 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.

Value

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

Warning

Custom image metadata must be organised hierarchically, e.g.

- Individual * Female1 * Female2 * Male1 * Male2

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 of your image management software accordingly. In DigiKam, go to Settings/Configure digiKam/Metadata. There, make sure "Write to sidecar files" is unchecked.

Author(s)

Juergen Niedballa

References

Phil Harvey's ExifTool `http://www.sno.phy.queensu.ca/~phil/exiftool/`

Examples

```
wd_images_ID <- system.file("pictures/sample_images_tagged/LeopardCat", package = "camtrapR")
# missing space in species = "LeopardCat" is because of CRAN package policies

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

rec.db.pbe <- recordTableIndividual(inDir                  = wd_images_ID,
                                    minDeltaTime           = 60,
                                    deltaTimeComparedTo    = "lastRecord",
                                    hasStationFolders      = FALSE,
                                    IDfrom                 = "metadata",
                                    camerasIndependent     = FALSE,
                                    writecsv               = FALSE,
                                    metadataIDTag          = "individual",
                                    additionalMetadataTags = c("Model", "Make"),
                                    timeZone               = "Asia/Kuala_Lumpur"
)

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

`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 Hierarchical-Subject)

`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

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

```
spatialDetectionHistory(recordTableIndividual,
  species,
  camOp,
  CTtable,
  output,
  stationCol = "Station",
  speciesCol = "Species",
  Xcol,
  Ycol,
  stationCovariateCols,
  individualCol,
  individualCovariateCols,
  recordDateTimeCol = "DateTimeOriginal",
  recordDateTimeFormat = "%Y-%m-%d %H:%M:%S",
  occasionLength,
  occasionStartTime = 0,
  maxNumberDays,
  day1,
  buffer,
  includeEffort = TRUE,
  scaleEffort = FALSE,
  binaryEffort,
  timeZone,
  makeRMarkInput
)
```

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`
`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`	integer. occasion length in days
`occasionStartTime`	integer. time of day (the full hour) at which to begin occasions.
`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. 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?
`binaryEffort`	logical. Should effort be binary (1 if >1 active day per occasion, 0 otherwise)?
`timeZone`	character. must be an argument of `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). scaleEffort and binaryEffort further define the behaviour.

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").

capthist objects (as created by spatialDetectionHistory for spatial capture-recapture analyses) expect the units of coordinates (Xcol and col in CTtable) to be meters.

Value

Output depends on argument makeRMarkInput:

makeRMarkInput = FALSE
A capthist object

makeRMarkInput = TRUE
A data frame for use in RMark

Author(s)

Juergen Niedballa

See Also

secr RMark

Examples

```
data(recordTableIndividualSample)
data(camtraps)

# 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)
```

`surveyReport` *Create a report about s 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 calculated by `recordTable`
`CTtable`	data.frame containing information about location and trapping period of camera trap stations
`speciesCol`	character. name of the column specifying Species ID in `CTtable`
`stationCol`	character. name of the column specifying Station ID in `CTtable`
`cameraCol`	character. name of the column specifying Camera ID in `CTtable`
`setupCol`	character. name of the column containing camera setup dates in `CTtable`
`retrievalCol`	character. name of the column containing camera retrieval dates in `CTtable`
`CTDateFormat`	character. The format of columns `setupCol` and `retrievalCol` in `CTtable`.
`CTHasProblems`	logical. Are there periods of camera malfunction specified in `CTtable`?
`recordDateTimeCol`	character. The names of the column containing date and time of record 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

The value of `CTDateFormat` should be interpretable by `as.Date`. `CTDateFormat` defaults to "YYYY-MM-DD", e.g. "2014-10-31". See `strptime` for how to format date and time strings in R. 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.

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",
                                        CTHasProblems        = TRUE)

reportTest_problem$survey_dates

## Not run:
# run again with sinkpath defined
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",
                            sinkpath             = getwd())

# have a look at the text file
readLines(list.files(getwd(), pattern = paste("survey_report_", Sys.Date(), ".txt", sep = ""),
 full.names = TRUE))

## End(Not run)
```

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 extentsion of the original images (.JPG) will be renamed to ".JPG_original".

Usage

```
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`.

Value

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

Author(s)

Juergen Niedballa

References

http://www.sno.phy.queensu.ca/~phil/exiftool/Shift.html

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", package = "camtrapR")
file.copy(from = wd_images_ID, to = tempdir(), recursive = TRUE)
wd_images_ID_copy <- file.path(tempdir(), "sample_images")

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

Index