3. Data Extraction from Images and videos, creating occupancy & secr input

Juergen Niedballa (camtrapr@gmail.com)

2024-02-01

Contents

Tabulating species and individual records: the recordTable functions **1**

`recordTable`: tabulating species records 2

`recordTableIndividual`: tabulating individuals of a species 8

Camera Operation **9**

Dates as input 9

Date-times as input 10

Differences between version 2.1 and previous versions 11

Camera days vs. camera nights 11

Combining multiple cameras per station 12

Plotting camera operation matrices 14

Saving and loading camera operation matrices 18

Input for subsequent analyses (single-season) **18**

Occupancy analyses 19

Spatial Capture-Recapture analyses 22

Input for subsequent analyses (multi-season) **24**

Multi-season occupancy analyses 24

Multi-season spatial capture-recapture 27

```r
library(camtrapR)
library(secr)
```

Tabulating species and individual records: the recordTable functions

There are 2 function to tabulate species records after identification of species from images and videos.

- `recordTable` tabulates records of all images after species-level identification and
- `recordTableIndividual` tabulates individuals of one species.

Nevertheless, the underlying idea is the same. For each image, the date and time it was taken are read from the image's Exif metadata using ExifTool. Species or individual ID are read from the directory structure or image metadata (see vignette "Species and Individual Identification"). Video data are extracted analogously.

`recordTable`: tabulating species records

`recordTable` is typically run after identifying species from images. It reads species IDs from the directory structure the images are placed in or from image metadata tags.

First we define the directory containing our renamed, identified images

```
# find the directory with sample images contained in the package
wd_images_ID <- system.file("pictures/sample_images_species_dir", package = "camtrapR", lib.loc = .libPa
```

and see how many JPG images we have (this is not necessary, but informative here).

```
length(list.files(wd_images_ID, pattern = "JPG", recursive = TRUE))
```

```
## [1] 68
```

Now we can run `recordTable`. Here is a minimal example:

```
rec.db.species0 <- recordTable(inDir  = wd_images_ID,
                               IDfrom = "directory")
```

```
## timeZone is not specified. Assuming UTC

## StationA:      8 images    0 duplicates removed      |=======             |   33%

## StationB:     23 images    6 duplicates removed      |=============       |   67%

## StationC:     37 images    6 duplicates removed      |====================|  100%
```

```
head(rec.db.species0)
```

```
##    Station Species    DateTimeOriginal       Date     Time delta.time.secs delta.time.mins delta.time
## 1 StationA     PBE 2009-04-21 00:40:00 2009-04-21 00:40:00               0               0
## 2 StationA     PBE 2009-04-22 20:19:00 2009-04-22 20:19:00          157140            2619
## 3 StationA     PBE 2009-04-22 20:21:00 2009-04-22 20:21:00             120               2
## 4 StationA     PBE 2009-04-23 00:07:00 2009-04-23 00:07:00           13560             226
## 5 StationA     PBE 2009-04-23 00:09:00 2009-04-23 00:09:00             120               2
## 6 StationA     PBE 2009-05-07 17:11:00 2009-05-07 17:11:00         1270920           21182
##
## 1 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_specie
## 2 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_specie
## 3 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_specie
## 4 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_specie
## 5 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_specie
## 6 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_specie
```

Argument `IDfrom` tells the function to look for species directories within the station directories and to take species IDs from these. `IDfrom` must be set to "metadata" if metadata tagging was used for species identification.

By default, the function returns all records. Nevertheless, we get a data frame containing 56 records, less than the number of images in the image directory (68). This is because a number of images were taken at the same time (measured with 1 minute precision by these cameras) and the function removes duplicate records.

It may suffice to illustrate this with the Pig-tailed macaque images from stationB:

```
list.files(file.path(wd_images_ID, "StationB", "MNE"))
```

```
##  [1] "StationB__2009-04-15__07-21-00(1).JPG" "StationB__2009-04-15__07-23-00(1).JPG" "StationB__2009-
##  [5] "StationB__2009-04-28__17-48-00(1).JPG" "StationB__2009-04-28__17-48-00(2).JPG" "StationB__2009-
##  [9] "StationB__2009-04-28__17-49-00(2).JPG" "StationB__2009-04-28__17-49-00(3).JPG"
```

There are, amongst others, 3 images taken at 17:48:00 on the 28th of April 2009. Of these, only 1 will be returned.

Here is what the columns of the record table contain:

column	content
`Station`	the station the image is from
`Species`	species name
`DateTimeOriginal`	Date and time of record in R-readable format
`Date`	record date
`Time`	record time of day
`delta.time.secs`	time difference between record and last (independent) record of same species at same station / camera* (in seconds)
`delta.time.mins`	time difference between record and last (independent) record of same species at same station / camera* (in minutes)
`delta.time.hours`	time difference between record and last (independent) record of same species at same station / camera* (in hours)
`delta.time.days`	time difference between record and last (independent) record of same species at same station / camera* (in days)
`Directory`	directory the image is in
`FileName`	image file name

*see below: Independence between cameras within stations

Temporal independence between records

Imagine a species that loves to hang out in front of your cameras. You will end up with hundreds of shots of the same species, maybe even the same animal. Therefore, images can be filtered using an adjustable criterion for temporal independence between subsequent records of the same species in an attempt to remove non-independent records. This is achieved via argument `minDeltaTime`. It is the minimum time difference (in minutes) between two records of the same species at the same station which are to be considered independent. The default is 0, causing the function to return all records. Setting it to a higher number, e.g. 60 (i.e., 1 hour), is commonly done to thin the number of records. Note that you will not lose records of different species, even if they fall within the specified time interval from a record.

The argument `deltaTimeComparedTo` further controls how independence between records is assessed. Setting it to "lastRecord" returns only records taken `minDeltaTime` minutes after the last record. Setting it to "lastIndependentRecord" returns only records taken `minDeltaTime` minutes after the last *independent* record.

Let's now apply an 1-hour independence criterion and define a time zone.

```
rec.db.species60 <- recordTable(inDir               = wd_images_ID,
                                IDfrom              = "directory",
                                minDeltaTime        = 60,
                                deltaTimeComparedTo = "lastRecord",
                                timeZone            = "Asia/Kuala_Lumpur")
```

```
## StationA:      8 images    0 duplicates removed      |=======              |   33%

## StationB:     23 images    6 duplicates removed      |=============        |   67%

## StationC:     37 images    6 duplicates removed      |=====================|  100%
```

```
nrow(rec.db.species60)
```

```
## [1] 40
```

Now 40 records were returned instead of 56. The missing records were taken less than 1 hour after the prior record of the same species at the same station and therefore omitted.

Time zones and daylight saving time

Note the warning about the missing time zone in the minimal example above. By default, camtrapR will assume UTC time zone. This should work well in most situations (even though the time zone may not be correct, strictly speaking), because UTC does not use daylight saving time (DST, aka 'summer time'), and camera traps normally don't use time zones (hence, camera traps will normally not respect DST of the area you work in and will not adjust image timestamps accordingly).

Now, if you work in an area that uses DST, and your camera does not know about DST, things can go wrong in various ways and you may end up with a systematic 1-hour offset. If you set your cameras up during DST (in summer) and set the internal clock accordingly, all records taken in winter (non-DST period) will be recorded as 1 hour later than according to actual clock time. If you set your cameras up during winter (not DST) and set the internal clock accordingly, all records taken in summer (DST period) will be recorded as 1 hour earlier than according to actual clock time.

In addition if you specify a time zone with DST and your cameras don't respect it, records may fall into the non-existing hour when clocks are advanced in spring, leading to an error in interpreting the date/time.

So, the question whether or not your cameras record the time zone becomes very important. Here is some recommendations on how to use the argument `timeZone`.

If your cameras don't save the time zone, and your time zone does not have DST, set argument `timeZone` to your study area time zone If your cameras don't save the time zone, and your time zone has DST, leave argument `timeZone` at the default UTC. If your cameras save the time zone and adjust image timestamps accordingly, set argument `timeZone` to your study area time zone (both if your area has DST or not).

In other words, it is advised to set argument `timeZone` to your study area's time zone (one of `OlsonNames()`), *unless* the time zone of your study area has DST, but your cameras don't record it.

Independence between cameras within stations

The issue of temporal independence between records becomes slightly more complex if more than one camera was operated at stations. That information can and should be included in the output of `recordTable` (and `recordTableIndividual`). Users can then decide whether temporal independence is to be assessed within stations or within cameras at each station. In the first case, argument `camerasIndependent` must be set to FALSE. 2 images taken at different 2 cameras at the same station within `minDeltaTime` minutes will be reported as 1 record in the record table (suitable if cameras were places in pairs). In the second case `camerasIndependent` must be set to TRUE and 2 images taken at different 2 cameras at the same station within `minDeltaTime` minutes will be reported as 2 record in the record table (suitable e.g., if cameras were located at some distance to one another and faced different trails).

The `cameraID` argument controls where camtrapR will look for camera IDs: in the file names (after renaming using `imageRename`, e.g. "renamedImages/StationA/StationA___**Camera1**___2015-12-31___23-59-59(1).JPG") or in the directory structure (e.g. renamedImages/StationA/**Camera1**/StationA___Camera1___2015-12-31___23-59-59(1).JPG"). If missing, it will be assumed there was only 1 camera per station.

Ignoring species

Argument `exclude` can be used to ignore certain species. This is useful for omitting images in directories like "team" or "unidentified". Here is an example:

```
# see what species  we recorded
table(rec.db.species60$Species)
```

```
##
##   EGY  MNE  PBE  TRA UNID  VTA
##     6    2   18    8    1    5
```

```
# remove "UNID" by setting argument exclude = "UNID"
rec.db.species60.exclude <- recordTable(inDir                = wd_images_ID,
                                        IDfrom               = "directory",
                                        minDeltaTime         = 60,
                                        deltaTimeComparedTo  = "lastIndependentRecord",
                                        timeZone             = "Asia/Kuala_Lumpur",
                                        exclude              = "UNID")
```

```
## StationA:      8 images    0 duplicates removed      |=======             |   33%

## StationB:     22 images    6 duplicates removed      |=============       |   67%

## StationC:     37 images    6 duplicates removed      |====================|  100%
```

```
# note that "UNID" is gone now
table(rec.db.species60.exclude$Species)
```

```
##
## EGY MNE PBE TRA VTA
##   6   2  18   8   5
```

Extracting image metadata

`recordTable` and `recordTableIndividual` can both extract additional metadata from images (apart from date and time). For example, some camera models record ambient temperature or moon phase, which may be of interest. Metadata tags are stored in the images at the time they are taken and can be accessed and extracted if their tag names are known. Some tag names are standardised (e.g. "DateTimeOriginal") while others are manufacturer-specific. Therefore, function `exifTagNames` returns all Exif metadata it finds in a sample image. Users can then choose which to extract with `recordTable` and `recordTableIndividual`.

```r
wd_images_ID <- system.file("pictures/sample_images_species_dir", package = "camtrapR")
exifTagNames(inDir = wd_images_ID)
```

```
## Metadata of:
## C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_species_d

##    tag_group                tag_name
## 1   ExifTool          ExifToolVersion
## 2   ExifTool                  Warning
## 3       File                 FileName
## 4       File                Directory C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camt
## 5       File                 FileSize
## 6       File           FileModifyDate
## 7       File           FileAccessDate
## 8       File           FileCreateDate
## 9       File          FilePermissions
## 10      File                 FileType
## 11      File        FileTypeExtension
## 12      File                 MIMEType
## 13      File            ExifByteOrder
## 14      File               ImageWidth
## 15      File              ImageHeight
## 16      File          EncodingProcess
## 17      File            BitsPerSample
## 18      File          ColorComponents
## 19      File         YCbCrSubSampling
## 20      EXIF                     Make
## 21      EXIF                    Model
## 22      EXIF              XResolution
## 23      EXIF              YResolution
## 24      EXIF               ModifyDate
## 25      EXIF           ResolutionUnit
## 26      EXIF         YCbCrPositioning
## 27      EXIF              ExifVersion
## 28      EXIF ComponentsConfiguration
## 29      EXIF          FlashpixVersion
## 30      EXIF               ColorSpace
## 31      EXIF           ExifImageWidth
## 32      EXIF          ExifImageHeight
## 33      EXIF         DateTimeOriginal
## 34      EXIF               CreateDate
## 35      EXIF              UserComment
## 36      EXIF        ShutterSpeedValue
## 37      EXIF              Compression
```

```
## 38      EXIF          ThumbnailOffset
## 39      EXIF          ThumbnailLength
## 40      EXIF           ThumbnailImage
## 41 Composite                ImageSize
## 42 Composite               Megapixels
## 43 Composite             ShutterSpeed
```

The output table contains tag groups, tag descriptions, tag names and tag values. The values in `tag_group` help to unambiguously identify specific metadata tags.

Use the entries in `tag_name` (not `tag_description`) when providing values to the argument `additionalMetadataTags` in the`recordTable` functions.

Of the tags shown here, "DateTimeOriginal" contains the date and time that camtrapR reads out. Apart from that, there is little information of ecological interest in the example data. However, for demonstration purposes, let's extract information about the camera model and make. To be as precise as possible, we provide tags in the format: tag_group:tag_name (e.g. EXIF:Model)

```
rec.db.species.metadata1 <- recordTable(inDir                  = wd_images_ID,
                                        IDfrom                 = "directory",
                                        timeZone               = "Asia/Kuala_Lumpur",
                                        additionalMetadataTags = c("EXIF:Model", "EXIF:Make"))
```

```
## StationA:      8 images    0 duplicates removed      |=======             |   33%

## StationB:     23 images    6 duplicates removed      |=============       |   67%

## StationC:     37 images    6 duplicates removed      |====================|  100%
```

```
head(rec.db.species.metadata1)
```

```
##    Station Species    DateTimeOriginal       Date     Time delta.time.secs delta.time.mins delta.time
## 1 StationA     PBE 2009-04-21 00:40:00 2009-04-21 00:40:00               0               0
## 2 StationA     PBE 2009-04-22 20:19:00 2009-04-22 20:19:00          157140            2619
## 3 StationA     PBE 2009-04-22 20:21:00 2009-04-22 20:21:00             120               2
## 4 StationA     PBE 2009-04-23 00:07:00 2009-04-23 00:07:00           13560             226
## 5 StationA     PBE 2009-04-23 00:09:00 2009-04-23 00:09:00             120               2
## 6 StationA     PBE 2009-05-07 17:11:00 2009-05-07 17:11:00         1270920           21182
##
## 1 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_species
## 2 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_species
## 3 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_species
## 4 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_species
## 5 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_species
## 6 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_species
##   n_images
## 1        1
## 2        1
## 3        1
## 4        1
## 5        1
## 6        1
```

There are 2 additional columns containing the information from the metadata we requested.

`recordTableIndividual`: tabulating individuals of a species

Individual identification of species is a prerequiste for running (spatial) capture-recapture models. `recordTableIndividual` and the subsequent `spatialDetectionHistory` function prepare data for these models.

```
# find the directory with tagged sample images contained in the package
wd_images_individual_ID <- system.file("pictures/sample_images_indiv_tag/LeopardCat", package = "camtrap
 # missing space in species = "LeopardCat" is because of CRAN package policies

 rec.db.pbe <- recordTableIndividual(inDir                  = wd_images_individual_ID,
                                     IDfrom                 = "metadata",
                                     minDeltaTime           = 60,
                                     deltaTimeComparedTo    = "lastIndependentRecord",
                                     hasStationFolders      = FALSE,          # images are not in statio
                                     metadataIDTag          = "individual",  # the name of the metadata
                                     timeZone               = "Asia/Kuala_Lumpur"
 )
```

```
## StationA, StationB, StationC:     26 images    3 duplicates removed      |====================|  100
```

Extracting custom image metadata

In addition to the metadata that were saved when the image was taken (see above), custom metadata tags assigned in image management software can also be extracted. These may contain species ID tags if species were identified using metadata tags (instead of moving images into species directories), but also individual IDs, animal counts, sex of animals recorded, etc..

```
head(rec.db.pbe)
```

```
##    Station    Species Individual    DateTimeOriginal       Date     Time delta.time.secs delta.time.
## 1 StationA LeopardCat          1 2009-05-07 17:11:00 2009-05-07 17:11:00               0
## 2 StationA LeopardCat          2 2009-04-21 00:40:00 2009-04-21 00:40:00               0
## 3 StationA LeopardCat          2 2009-04-22 20:19:00 2009-04-22 20:19:00          157140            2
## 4 StationA LeopardCat          2 2009-04-23 00:07:00 2009-04-23 00:07:00           13680
## 5 StationB LeopardCat          1 2009-04-07 00:23:00 2009-04-07 00:23:00               0
## 6 StationB LeopardCat          1 2009-04-14 06:13:00 2009-04-14 06:13:00          625800           10
##
## 1 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_indiv_
## 2 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_indiv_
## 3 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_indiv_
## 4 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_indiv_
## 5 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_indiv_
## 6 C:/Users/niedballa/AppData/Local/Programs/R/R-4.3.2/library/camtrapR/pictures/sample_images_indiv_
##   metadata_individual n_images               HierarchicalSubject
## 1                   1        1 Species|Leopard Cat, individual|1
## 2                   2        1 Species|Leopard Cat, individual|2
## 3                   2        2 Species|Leopard Cat, individual|2
## 4                   2        2 Species|Leopard Cat, individual|2
## 5                   1        1 Species|Leopard Cat, individual|1
## 6                   1        1 Species|Leopard Cat, individual|1
```

Camera Operation

The camera operation matrix is a day-by-station matrix that states how many cameras were active at a station on a given day. Rows represent stations and columns days, beginning with the day the first camera was set up and ending the day the last camera was retrieved. It is calculated from the camera trap table mention in vignette 1 (see ?camtraps for an example). Users can give the setup / retrieval / problem columns in the input table as dates or date-times (since version 2.1.0).

Important: Prior to version 2.1.0, the values were integer numbers, with 1 indicating the camera was active on a day (or possibly higher values if multiple cameras at a station are combined).

From version 2.1.0, the values are the fraction of the day a camera was active. If only dates are provided in the camera trap table, the function will assume 12 noon as setup/retrieval time Problem periods will be assumed to last the entire day.

If imput is provided as date-times, output will be the exact fraction of day a station or camera was active (taking into account setup / retrieval time as well as time of Problem periods).

Dates as input

```
 # first load the camera trap station table
data(camtraps)

dateFormat <- "dmy"    # requires lubridate package
# alternatively, use "%d/%m/%Y" (from base R)

camop_problem <- cameraOperation(CTtable      = camtraps,
                                 stationCol   = "Station",
                                 setupCol     = "Setup_date",
                                 retrievalCol = "Retrieval_date",
                                 writecsv     = FALSE,
                                 hasProblems  = TRUE,
                                 dateFormat   = dateFormat
)

# as a reminder, these are the dates in our station information table
camtraps[,-which(colnames(camtraps) %in% c("utm_y", "utm_x"))]
```

```
##    Station Setup_date Retrieval_date Problem1_from Problem1_to
## 1 StationA 02/04/2009     14/05/2009
## 2 StationB 03/04/2009     16/05/2009
## 3 StationC 04/04/2009     17/05/2009    12/05/2009  17/05/2009
```

```
# now let's have a look at the first few columns of the camera operation matrix
camop_problem[, 1:5]
```

```
##          2009-04-02 2009-04-03 2009-04-04 2009-04-05 2009-04-06
## StationA        0.5        1.0        1.0          1          1
## StationB         NA        0.5        1.0          1          1
## StationC         NA         NA        0.5          1          1
```

```r
# and the last few
camop_problem[, (ncol(camop_problem)-6):ncol(camop_problem)]
```

```
##          2009-05-11 2009-05-12 2009-05-13 2009-05-14 2009-05-15 2009-05-16 2009-05-17
## StationA          1          1          1        0.5         NA         NA         NA
## StationB          1          1          1        1.0          1        0.5         NA
## StationC          1          0          0        0.0          0        0.0          0
```

If stations were not set up, values are NA. Operational stations get value 1. If cameras were set up but malfunctioning, it is 0 (if hasProblems = TRUE).

Date-times as input

```r
camtraps_hrs <- camtraps

# assign hours for setup and retrieval
camtraps_hrs$Setup_date     <- paste(camtraps_hrs$Setup_date, c("12", "15", "18"))
camtraps_hrs$Retrieval_date <- paste(camtraps_hrs$Retrieval_date, c("18", "15", "12"))

# assign a random hours for begin of Problem at station 3
camtraps_hrs$Problem1_from[3]  <- paste(camtraps_hrs$Problem1_from[3], "20")
# Problem ends with retrieval (i.e., malfunction until camera retrieved)
camtraps_hrs$Problem1_to[3]  <- camtraps_hrs$Retrieval_date[3]




# create camera operation matrix
camop_hours <- cameraOperation(CTtable = camtraps_hrs,
                                         stationCol    = "Station",
                                         setupCol      = "Setup_date",
                                         retrievalCol = "Retrieval_date",
                                         hasProblems  = TRUE,
                                         dateFormat    = "dmy H"
)

# as a reminder, these are the date-times in our station information table
camtraps_hrs
```

```
##    Station  utm_y  utm_x    Setup_date Retrieval_date Problem1_from   Problem1_to
## 1 StationA 604000 526000 02/04/2009 12  14/05/2009 18
## 2 StationB 606000 523000 03/04/2009 15  16/05/2009 15
## 3 StationC 607050 525000 04/04/2009 18  17/05/2009 12 12/05/2009 20 17/05/2009 12
```

```r
# now let's have a look at the first few columns of the camera operation matrix
camop_hours[, 1:5]
```

```
##          2009-04-02 2009-04-03 2009-04-04 2009-04-05 2009-04-06
## StationA        0.5      1.000       1.00          1          1
## StationB         NA      0.375       1.00          1          1
## StationC         NA         NA       0.25          1          1
```

```r
# and the last few
camop_hours[, (ncol(camop_hours)-6):ncol(camop_hours)]
```

```
##          2009-05-11 2009-05-12 2009-05-13 2009-05-14 2009-05-15 2009-05-16 2009-05-17
## StationA          1     1.0000          1       0.75         NA         NA         NA
## StationB          1     1.0000          1       1.00          1      0.625         NA
## StationC          1     0.8333          0       0.00          0      0.000          0
```

Differences between version 2.1 and previous versions

Version 2.1 introduced support of cameraOperation() for date-times in the setup / retrieval / problem columns. By giving the setup / retrieval / problem columns as date-times (and adjusting argument dateFormat accordingly), one can calculate precise daily effort.

If the times are unknown and users provide dates only to argument dateFormat, camtrapR will assume that setup and retrieval were at 12 noon. Malfunction periods (indicated by the problem columns) will be interpreted as the cameras having malfunctioned the whole day.

This is how the handling of setup/retrieval days differs between dates and date-times in dateFormat:

version	dateFormat is date	dateFormat is date-time
prior to 2.1	1	not implemented
2.1 and later	0.5	calculated precisely

This is how the handling of problems / malfunction periods differs between dates and date-times in dateFormat:

version	dateFormat is date	dateFormat is date-time
prior to 2.1	0	not implemented
2.1 and later	0	calculated precisely

Camera days vs. camera nights

By default, the camera operation matrix refers to days, centered on noon and lasting from midnight to midnight the next day. In some situations it can make sense to shift the time frame for analyses, so the camera operation matrix describes trap nights (centered on midnight, lasting from noon to noon the next day).

To that end, the argument occasionStartTime can be set. Prior to version 2.1.0, it was an argument of detectionHistory and spatialDetectionHistory, but was moved to cameraOperation in v2.1.0. It can be set to an hour between 0 and 23, describing the hour the occasions begin.

```r
# create camera operation matrix with occasions / trap days starting on noon (until noon the next day)
camop_hours_12 <- cameraOperation(CTtable = camtraps_hrs,
                                  stationCol   = "Station",
                                  setupCol     = "Setup_date",
                                  retrievalCol = "Retrieval_date",
                                  hasProblems  = TRUE,
                                  dateFormat   = "dmy H",
                                  occasionStartTime = 12
```

```
)

# now let's have a look at the first few columns of the camera operation matrix
camop_hours_12[, 1:5]
```

```
##          2009-04-02+12h 2009-04-03+12h 2009-04-04+12h 2009-04-05+12h 2009-04-06+12h
## StationA              1          1.000           1.00              1              1
## StationB             NA          0.875           1.00              1              1
## StationC             NA             NA           0.75              1              1
```

```
# and the last few
camop_hours_12[, (ncol(camop_hours_12)-6):ncol(camop_hours_12)]
```

```
##          2009-05-11+12h 2009-05-12+12h 2009-05-13+12h 2009-05-14+12h 2009-05-15+12h 2009-05-16+12h 20
## StationA              1         1.0000              1           0.25             NA             NA
## StationB              1         1.0000              1           1.00              1          0.125
## StationC              1         0.3333              0           0.00              0          0.000
```

The column names now indicate the shift, "+12h" in the example above. That means that the first column shows the effort from 12 noon on 2009-04-02 until 12 noon 2009-04-03.

When using this camera operation matrix in detectionHistory or spatialDetectionHistory, the shift will be extracted and trapping effort will be calculated accordingly. Therefore, the argument occasionStartTime in these two functions is deprecated from version 2.1.0.

Combining multiple cameras per station

Often multiple cameras are set at a station, but data are to be analysed on a station-level, not at the level of individual cameras. cameraOperation can combine these effort of individual cameras according to how they were set up.

First, let's create a simple artifial data set. Note how the columns contain an hour (e.g. 12 noon in this case: "2020-01-01 12"). Other formats for date-time are possible and can be specified with argument dateFormat in cameraOperation.

```
camtraps_by_camera <- data.frame(Station = c("A", "A"),
                        camera = c("cam1", "cam2"),
                        setup = c("2020-01-01 12"),
                        retrieval = c("2020-01-08 18", "2020-01-07 18"),
                        Problem1_from = c("2020-01-03 12", NA),
                        Problem1_to = c("2020-01-05 18", NA),
                        Problem2_from = c("2020-01-06 12", NA),
                        Problem2_to = c("2020-01-07 12", NA))

camtraps_by_camera
```

```
##   Station camera         setup     retrieval Problem1_from   Problem1_to Problem2_from   Problem2_to
## 1       A   cam1 2020-01-01 12 2020-01-08 18 2020-01-03 12 2020-01-05 18 2020-01-06 12 2020-01-07 12
## 2       A   cam2 2020-01-01 12 2020-01-07 18          <NA>          <NA>          <NA>          <NA>
```

To aggregate data from cameras to station, you need a camera column ("camera" in the example above). Furthermore, set argument byCamera = FALSE. Then, the combinations of the arguments camerasIndependent and allCamsOn define how effort from individual cameras is combined. allCamsOn means all cameras need to be active for a station to be considered active. camerasIndepndent specifies whether the cameras accumulate effort independently or now. See ?cameraoperation for details.

Below we'll run cameraOperation of the little sample table camtraps_by_camera, for all possible combinations of allCamsOn, cameraOperation and hasProblems.

```
# list of common arguments for the different function runs
 args.common <- list(CTtable = camtraps_by_camera,
  stationCol = "Station",
  cameraCol  = "camera",
  setupCol   = "setup",
  retrievalCol = "retrieval",
  dateFormat = "ymd H",
  byCamera = FALSE,
  writecsv = FALSE
  )

# combinations of the arguments of interest
 fun.args0 <- expand.grid(allCamsOn = c(T,F),
                                camerasIndependent = c(T,F),
                                hasProblems = c(T,F))

 # create a list of function arguments with the varying arguments
   fun.args1 <- apply(fun.args0, MARGIN = 1, FUN = function(x) modifyList(args.common, as.list(x)))

   # run cameraOperation on each combination of arguments, and combine into an output table with the fu
   camOp_station_aggregation <- cbind(fun.args0, do.call(rbind, lapply(fun.args1, FUN = function(x)
     do.call(cameraOperation, x))), row.names = NULL)
```

Now let's see how the daily effort was combined for the different combinations of allCamsOn and camerasIndependent when hasProblems is FALSE.

```
   camOp_station_aggregation [camOp_station_aggregation$hasProblems == FALSE, -3]
```

```
##   allCamsOn camerasIndependent 2020-01-01 2020-01-02 2020-01-03 2020-01-04 2020-01-05 2020-01-06 2020
## 5      TRUE               TRUE        1.0          2          2          2          2          2
## 6     FALSE               TRUE        1.0          2          2          2          2          2
## 7      TRUE              FALSE        0.5          1          1          1          1          1
## 8     FALSE              FALSE        0.5          1          1          1          1          1
```

sS you can see, allCamsOn has no effect if all cameras were active (due to hasProblems = FALSE). The differences on the setup/retrieval day are due to camerasIndependent.

Now with hasProblems = TRUE:

```
   camOp_station_aggregation [camOp_station_aggregation$hasProblems == TRUE, -3]
```

```
##   allCamsOn camerasIndependent 2020-01-01 2020-01-02 2020-01-03 2020-01-04 2020-01-05 2020-01-06 2020
## 1      TRUE               TRUE        1.0          2       1.00        0.0      0.500       1.00
## 2     FALSE               TRUE        1.0          2       1.50        1.0      1.250       1.50
## 3      TRUE              FALSE        0.5          1       0.50        0.0      0.250       0.50
## 4     FALSE              FALSE        0.5          1       0.75        0.5      0.625       0.75
```

Here's how the daily aggregates are calculate for the different combinations of allCamsOn and camerasIndependent, and depending on whether input columns are date or date-time (from v2.1.0). In the table below, x is a vector of values at individual cameras for one day (e.g. c(1,0.5) when one camera was active the whole day, the other cameras half a day).

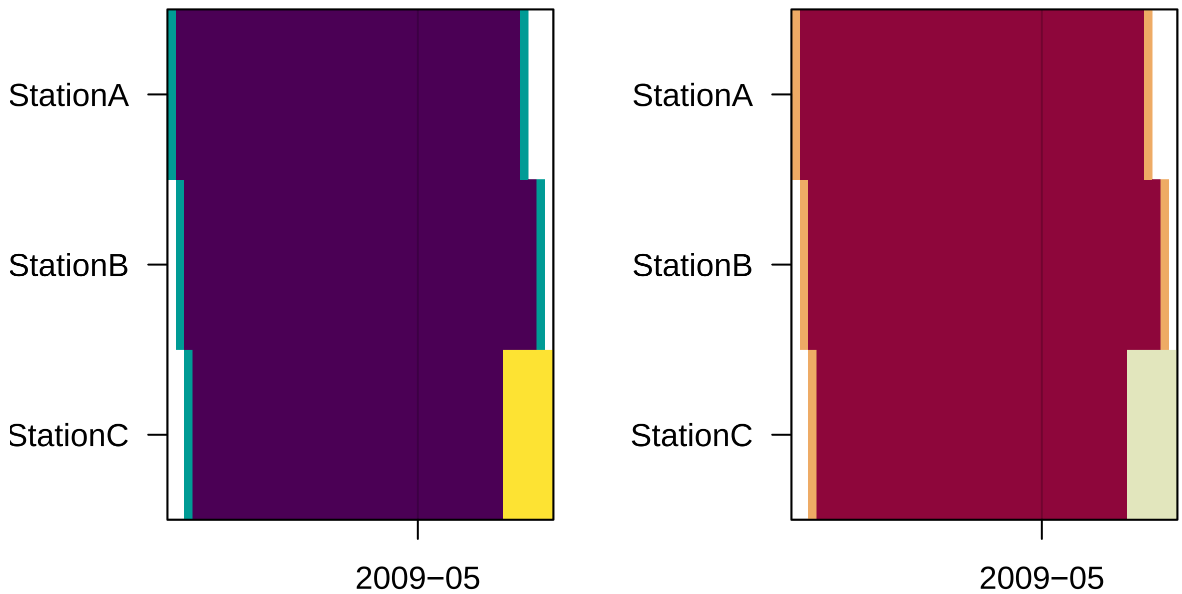
camerasIndependent	allCamsOn	input columns	aggregate with	example c(1, 0.5)	example c(1, 1)
TRUE	TRUE	date-time	ifelse(all(x == 1), sum(x), min(x) * length(x))	1	2
TRUE	TRUE	date	ifelse(all(x == 1), sum(x), 0)	0	2
TRUE	FALSE	date-time	sum	1.5	2
TRUE	FALSE	date	sum	1.5	2
FALSE	TRUE	date-time	min	0.5	1
FALSE	TRUE	date	ifelse(all(x == 1), 1, 0)	0	1
FALSE	FALSE	date-time	mean	0.75	2
FALSE	FALSE	date	ifelse(sum(x, na.rm = TRUE) >= 1, 1, sum(x, na.rm = TRUE))	1	1

Plotting camera operation matrices

There is a little function for plotting the camera operation matrix included in the package, but not exported in the Namespace. It is accessible with camtrapR:::camopPlot(). Imagine there is a typo in one of your date fields and the setup or retrieval year is wrong. You will easily be able to spot it this way.

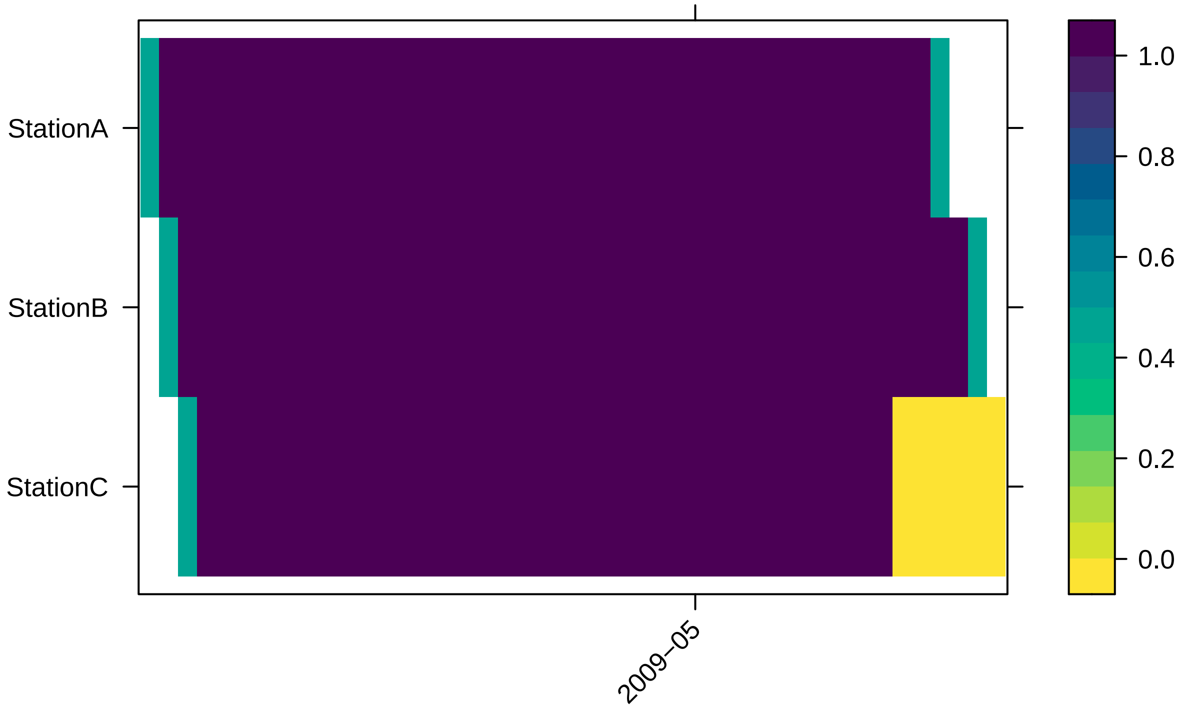
Here is the camera operation matrix calculated above (in two different color palettes, the default being the viridis palette - requires R 3.6.0 or higher).

```
par(mfrow = c(1,2))
camtrapR:::camopPlot(camOp = camop_problem)
camtrapR:::camopPlot(camOp = camop_problem, palette = "Heat")
```

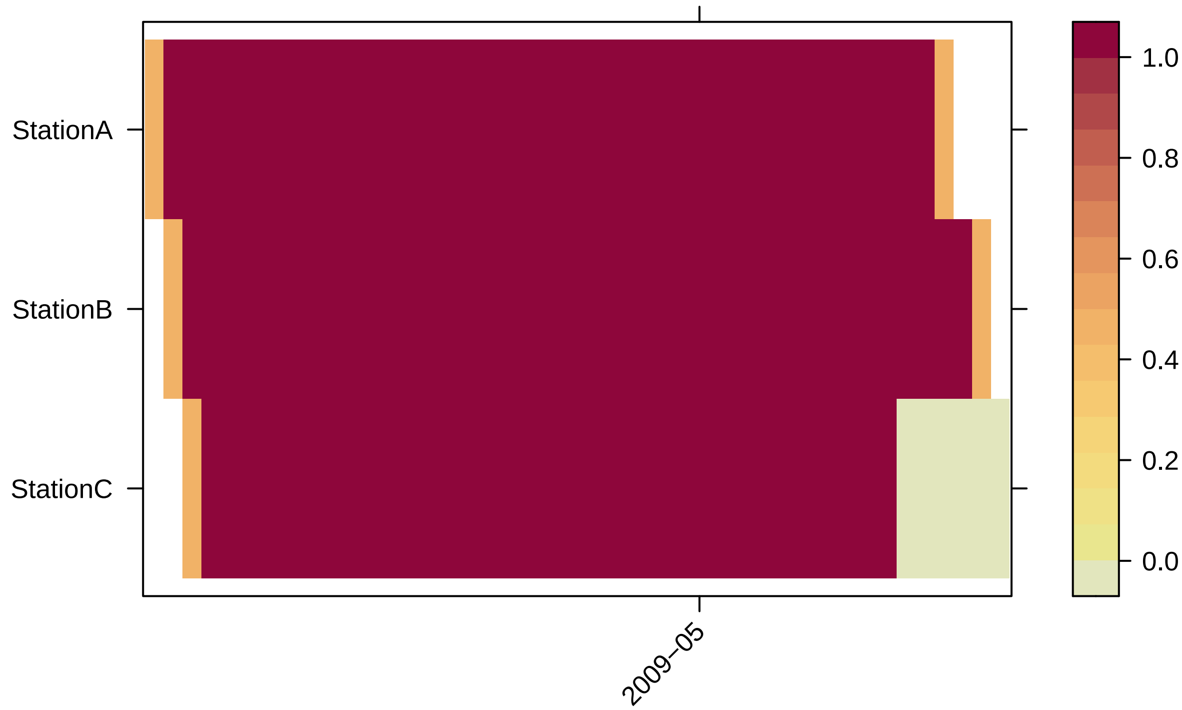


One can also plot with the levelplot function from the lattice package:

```
camtrapR:::camopPlot(camOp = camop_problem, lattice = TRUE)
```

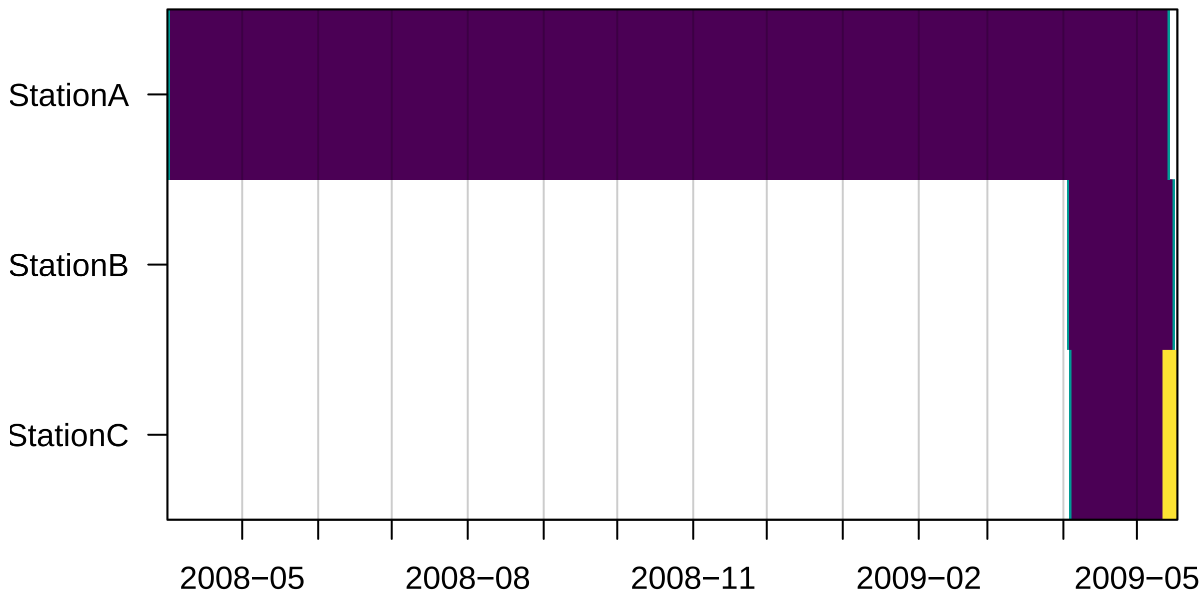



```
camtrapR:::camopPlot(camOp = camop_problem, palette = "Heat", lattice = TRUE)
```



Here is what it would look like if the year is wrong for one station.

```
camtraps_typo <- camtraps
# replace 2009 with 2008 in setup date of station A
camtraps_typo$Setup_date[1] <- gsub("2009", "2008", camtraps_typo$Setup_date[1])

camop_typo <- cameraOperation(CTtable      = camtraps_typo,
                              stationCol   = "Station",
                              setupCol     = "Setup_date",
                              retrievalCol = "Retrieval_date",
                              writecsv     = FALSE,
                              hasProblems  = TRUE,
                              dateFormat   = dateFormat
)
camtrapR:::camopPlot(camop_typo)
```



Note that the colors are stretched between the lowest and highest value. There is no dedicated color for 0, designating non-operational cameras.

For a list of color palettes, see here.

Saving and loading camera operation matrices

The camera operation matrix can easily be saved as a csv file (by setting argument `writecsv = TRUE`, `check.names = FALSE` and defining `outdir`). In order to load the csv into R again, it is necessary to tell R to use the station IDs (the first column) as row names:

```
camOp <- read.csv(file = ..., row.names = 1, check.names = FALSE)
```

`check.names = FALSE` ensures that column names (the dates) are read back into R as they are (e.g. "2015-12-01"). Otherwise one may end up with unreadable column names (at least for camtrapR) such as "X2015.12.01".

Input for subsequent analyses (single-season)

It is very easy to prepare input for single-season occupancy and spatial capture-recapture (SCR) analyses in camtrapR. All one needs it a record table and the camera operation matrix. Making input for SCR analyses further requires the camera trap station table.

Occupancy analyses

Occupancy models use detection/non-detection matrices in which for every station and every occasion "1"" signifies a detection of a given species, "0"" signifies non-detecion, and NA missing data.

Here is how to obtain a detection/non-detection matrix using function `detectionHistory`. Because the function builds on prior functions (`recordTable` and `cameraOperation`) we also show these function here to provide the context.

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

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

# make record table
recordTableSample <- recordTable(inDir               = wd_images_ID,
                                 IDfrom              = "directory",
                                 minDeltaTime        = 60,
                                 deltaTimeComparedTo = "lastIndependentRecord",
                                 timeZone            = "Asia/Kuala_Lumpur"
)
```

```
## StationA:      8 images    0 duplicates removed      |=======             |  33%

## StationB:     23 images    6 duplicates removed      |=============       |  67%

## StationC:     37 images    6 duplicates removed      |====================| 100%
```

```
# make detection history (without trapping effort)
DetHist1 <- detectionHistory(recordTable         = recordTableSample,
                             camOp               = camop_no_problem,
                             stationCol          = "Station",
                             speciesCol          = "Species",
                             recordDateTimeCol   = "DateTimeOriginal",
                             species             = "VTA",
                             occasionLength      = 7,
                             day1                = "station",
                             includeEffort       = FALSE
)
```

```
## Warning: timeZone is not specified. Assuming UTC
```

```
DetHist1
```

```
## $detection_history
```

```
##           o1 o2 o3 o4 o5 o6 o7
## StationA  0  1  0  0  1  0 NA
## StationB  0  1  0  1  0  0 NA
## StationC  0  0  1  0  0  0 NA
```

Note the warning about the missing time zone (as in the functions `recordTable` and `recordTable`). Normally, it should be fine, but to be on the safe side, better set it to your study area's time zone.

If trapping effort is thought to influence detection probability, it can be returned by setting `includeEffort = TRUE`. This way the number of active trapping days per occasion and station is returned.

```
# make detection history (with trapping effort)
DetHist2 <- detectionHistory(recordTable          = recordTableSample,
                             camOp                = camop_no_problem,
                             stationCol           = "Station",
                             speciesCol           = "Species",
                             recordDateTimeCol    = "DateTimeOriginal",
                             species              = "VTA",
                             timeZone             = "Asia/Kuala_Lumpur",
                             occasionLength       = 7,
                             day1                 = "station",
                             includeEffort        = TRUE,
                             scaleEffort          = FALSE
)

DetHist2[[1]]  # detection history
```

```
##           o1 o2 o3 o4 o5 o6 o7
## StationA  0  1  0  0  1  0  0
## StationB  0  1  0  1  0  0  0
## StationC  0  0  1  0  0  0  0
```

```
DetHist2[[2]]  # effort (in days per occasion)
```

```
##            o1 o2 o3 o4 o5 o6  o7
## StationA 6.5  7  7  7  7  7 0.5
## StationB 6.5  7  7  7  7  7 1.5
## StationC 6.5  7  7  7  7  7 1.5
```

To help with convergence of models, the effort matrix can be scaled to mean = 0 and sd = 1 by setting `scaleEffort = TRUE`. If `writecsv = TRUE`, the scaling parameters will also be saved in a separate csv file.

```
DetHist3 <- detectionHistory(recordTable          = recordTableSample,
                             camOp                = camop_no_problem,
                             stationCol           = "Station",
                             speciesCol           = "Species",
                             recordDateTimeCol    = "DateTimeOriginal",
                             species              = "VTA",
                             timeZone             = "Asia/Kuala_Lumpur",
                             occasionLength       = 7,
                             day1                 = "station",
                             includeEffort        = TRUE,
```

```
                              scaleEffort          = TRUE
)

DetHist3[[1]]  # detection history (same as above)
```

```
##           o1 o2 o3 o4 o5 o6 o7
## StationA  0  1  0  0  1  0  0
## StationB  0  1  0  1  0  0  0
## StationC  0  0  1  0  0  0  0
```

```
DetHist3[[2]]  # effort (scaled)
```

```
##                 o1        o2        o3        o4        o5        o6        o7
## StationA 0.1948432 0.4355319 0.4355319 0.4355319 0.4355319 0.4355319 -2.693421
## StationB 0.1948432 0.4355319 0.4355319 0.4355319 0.4355319 0.4355319 -2.212043
## StationC 0.1948432 0.4355319 0.4355319 0.4355319 0.4355319 0.4355319 -2.212043
```

```
DetHist3[[3]]  # scaling parameters for back-transformation
```

```
##   effort.scaled.center effort.scaled.scale
## 1             6.095238            2.077372
```

```
# backtransform scaled effort like this if needed
(DetHist3[[2]] * DetHist3[[3]]$effort.scaled.scale) + DetHist3[[3]]$effort.scaled.center
```

```
##            o1 o2 o3 o4 o5 o6  o7
## StationA 6.5  7  7  7  7  7 0.5
## StationB 6.5  7  7  7  7  7 1.5
## StationC 6.5  7  7  7  7  7 1.5
```

Handling of incomplete occasions

The following table shows the behaviour of the `detectionHistory` function for different combinations of the function arguments `includeEffort` and `minActiveDaysPerOccasion` and different occasion-level camera operation values (table head, column 3-7). Depending on these arguments, incomplete occasions will either NA or have values of 0/1 (depending on whether there was a detection) in the output detection matrix .

includeEffort	minActiveDaysPerOccasion	camera operation all 1	at least one 1	all 0	0 and NA	all NA
TRUE	not defined	0/1	0/1	NA	NA	NA
TRUE	defined	0/1	0/1/NA*	NA	NA	NA
FALSE	not defined	0/1	NA	NA	NA	NA
FALSE	defined	0/1	0/1/NA*	NA	NA	NA

*: NA if there were less active days in an occasion than `minActiveDaysPerOccasion`

The same applies to generation of input for spatial capture-recapture analyses using `spatialDetectionHistory` as described below.

Saving and loading detection histories

The detection history and effort matrices can easily be saved as csv files (by setting argument `writecsv = TRUE` and defining `outdir`). In order to load the csv into R again, it is necessary to tell R to use the station IDs as row names:

```
detHist <- read.csv(file = ..., row.names = 1)
effort  <- read.csv(file = ..., row.names = 1)
```

Spatial Capture-Recapture analyses

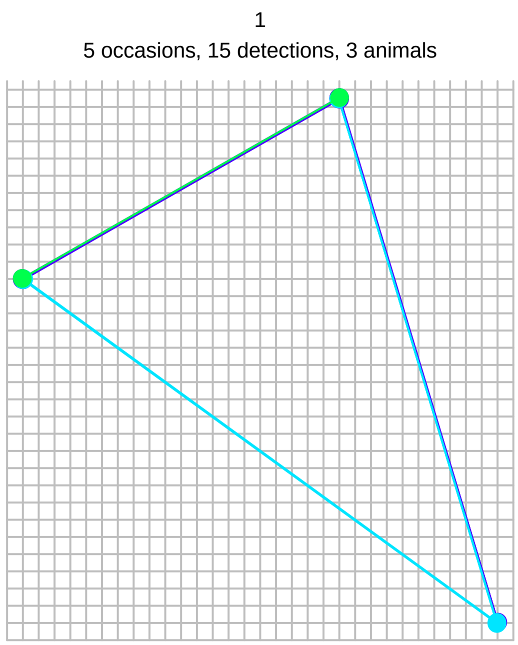
Input for spatial capture-recapture analyses can be generated in the form of `capthist`-objects as defined in the `secr` package with the function `spatialDetectionHistory`. Output can be in the form of counts (number of individual detections per occasion, argument, argument `output = "count"`) or binary (was an individual detected during an occasion, argument `output = "binary"`). note that the detector type will change accordingly: "proximity" if `output = "binary"` and "count" if `output = "count"`.

```
data(recordTableIndividualSample)

# 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   = dateFormat
)

sdh <- spatialDetectionHistory(recordTableIndividual = recordTableIndividualSample,
                               species               = "LeopardCat",
                               output                = "binary",
                               camOp                 = camop_problem,
                               CTtable               = camtraps,
                               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 own data you can have spaces in your directory names.

  summary(sdh)
```

```
## Object class       capthist
## Detector type      proximity
## Detector number    3
## Average spacing    2258.871 m
## x-range            523000 526000 m
## y-range            604000 607050 m
##
##  Usage range by occasion
##       1  2  3  4   5
## min 7.5 10 10 10 0.0
## max 9.5 10 10 10 4.5
##
## Counts by occasion
##                    1 2 3 4 5 Total
## n                  2 2 3 2 1    10
## u                  2 0 1 0 0     3
## f                  0 1 0 2 0     3
## M(t+1)             2 2 3 3 3     3
## losses             0 0 0 0 0     0
## detections         3 4 4 3 1    15
## detectors visited  2 3 3 2 1    11
## detectors used     3 3 3 3 2    14
```

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

```
## Warning in plot.capthist(sdh, tracks = TRUE): track for repeat detections on same occasion joins poi
```



Input for subsequent analyses (multi-season)

Since version 1.2, camtrapR can prepare input for multi-season occupancy and spatial capture-recapture analyses (in unmarked and secr). The process is very similar to the one for single-season models.

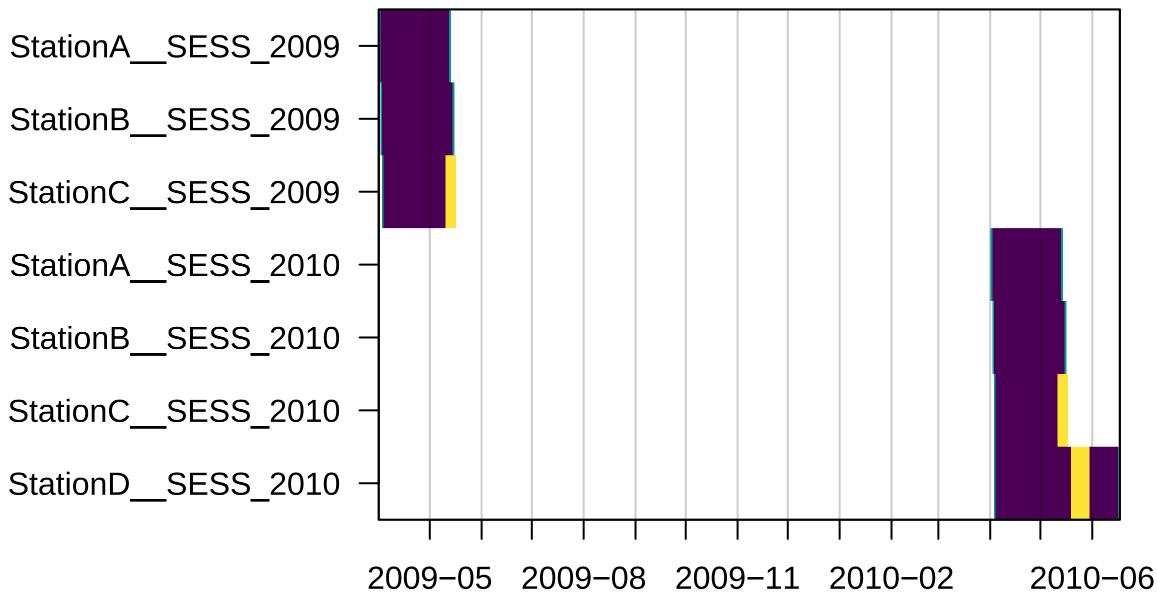
Multi-season occupancy analyses

camtrapR prepares detection histories that can be used as input for argument `y` in function `unmarkedMultFrame`. `unmarkedMultFrame` creates the input for function `colext` which fits multi-season occupancy models.

For multi-season occupancy, we need a session column in the camera trap table. Records will be assigned to session automatically based on their dates.

First we simulate data for 2 seasons by duplicating the existing sample data and adjusting the dates (add 1 year)

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

# also, for clarity, lets remove all unnecessary columns from the record table
recordTableSampleMultiSeason <- recordTableSampleMultiSeason[, c("Station", "Species", "DateTimeOrigina

# create camera operation matrix
camop_season <- cameraOperation(CTtable         = camtrapsMultiSeason,
                                    stationCol   = "Station",
                                    setupCol     = "Setup_date",
                                    sessionCol   = "session",
                                    retrievalCol = "Retrieval_date",
                                    hasProblems  = TRUE,
                                    dateFormat   = dateFormat
)

# plot camera operation matrix
par(oma = c(0,7,0,0))
camtrapR:::camopPlot(camop_season)
```




```
# make multi-season detection history
DetHist_multi <- detectionHistory(recordTable   = recordTableSampleMultiSeason,
                             camOp                  = camop_season,
                             stationCol             = "Station",
                             speciesCol             = "Species",
                             species                = "VTA",
                             occasionLength         = 10,
                             day1                   = "station",
                             recordDateTimeCol      = "DateTimeOriginal",
                             includeEffort          = TRUE,
                             scaleEffort            = FALSE,
                             timeZone               = "UTC",
                             unmarkedMultFrameInput = TRUE
)

DetHist_multi
```

```
## $detection_history
##          o1 o2 o3 o4 o5 o6 o7 o8 o1 o2 o3 o4 o5 o6 o7 o8
## StationA  1  0  0  1  0 NA NA NA  1  0  0  1  0 NA NA NA
## StationB  0  1  1  0  0 NA NA NA  0  1  1  0  0 NA NA NA
## StationC  0  0  1  0 NA NA NA NA  0  0  1  0 NA NA NA NA
## StationD NA NA NA NA NA NA NA NA  0  0  0  0  0  0  0  0
##
## $effort
##           o1 o2 o3 o4  o5 o6 o7 o8  o1 o2 o3 o4  o5 o6 o7  o8
```

```
## StationA 9.5 10 10 10 2.5 NA NA NA 9.5 10 10 10 2.5 NA NA  NA
## StationB 9.5 10 10 10 3.5 NA NA NA 9.5 10 10 10 3.5 NA NA  NA
## StationC 9.5 10 10  8  NA NA NA NA 9.5 10 10  8  NA NA NA  NA
## StationD  NA NA NA NA  NA NA NA NA 9.5 10 10 10 6.0  3 10 4.5
```

Note that the function makes sure that all seasons have identical number of occasions, and that all stations are represented, even if they were not sampled in all seasons.

From here, the resulting object can be used in unmarked.

```
year_matrix <- matrix(unique(as.character(camtrapsMultiSeason$session)),
               ncol = length(unique(as.character(camtrapsMultiSeason$session))),
               nrow = length(unique(camtrapsMultiSeason$Station)),
               byrow = TRUE)

# this is a made up example table with station covariates for demonstration
site_covariates <- data.frame(Station = rownames(DetHist_multi$detection_history),
                              elevation = c(100, 200, 500, 300),
                              treecover = c(80, 100, 50, 10))

umf <- unmarked::unmarkedMultFrame(y = DetHist_multi$detection_history,
                                   siteCovs = site_covariates,
                                   yearlySiteCovs = list(year = year_matrix),
                                   obsCovs = list(effort = DetHist_multi$effort),
                                   numPrimary = 2)
```

```
## Warning: siteCovs contains characters. Converting them to factors.

## Warning: yearlySiteCovs contains characters. Converting them to factors.
```

```
colext_example <- unmarked::colext(psiformula = ~ treecover,    # First-year occupancy
                                   gammaformula = ~ 1,          # Colonization
                                   epsilonformula = ~ 1,        # Extinction
                                   pformula = ~ effort,         # Detection
                                   data = umf,
                                   method="BFGS")
summary(colext_example)
```

```
##
## Call:
## unmarked::colext(psiformula = ~treecover, gammaformula = ~1,
##     epsilonformula = ~1, pformula = ~effort, data = umf, method = "BFGS")
##
## Initial (logit-scale):
##             Estimate    SE      z P(>|z|)
## (Intercept)   -9.504 28.60 -0.332    0.74
## treecover      0.342  1.23  0.279    0.78
##
## Colonization (logit-scale):
##  Estimate   SE      z P(>|z|)
##      -6.6 27.3 -0.242   0.809
##
## Extinction (logit-scale):
```

```
##  Estimate   SE      z P(>|z|)
##     -7.89 29.4 -0.269   0.788
##
## Detection (logit-scale):
##             Estimate    SE     z P(>|z|)
## (Intercept)   -17.46 14.73 -1.19   0.236
## effort          1.75  1.49  1.17   0.241
##
## AIC: 42.03042
## Number of sites: 4
## optim convergence code: 0
## optim iterations: 45
## Bootstrap iterations: 0
```

Multi-season spatial capture-recapture

For secr, session IDs must be a gapless sequence of numbers beginning with 1. So 2009 becomes season 1 and 2010 season 2

```
camtrapsMultiSeason$session[camtrapsMultiSeason$session == 2009] <- 1
camtrapsMultiSeason$session[camtrapsMultiSeason$session == 2010] <- 2
```

```
# we also want a few records in season 2
recordTableIndividualSample_season2 <- recordTableIndividualSample[1:10,]
recordTableIndividualSample_season2$DateTimeOriginal <- gsub("2009", "2010", recordTableIndividualSampl
recordTableIndividualSample_season <- rbind(recordTableIndividualSample, recordTableIndividualSample_se



# for clarity, lets remove all unnecessary columns
recordTableIndividualSample_season <- recordTableIndividualSample_season[, c("Station", "Species", "Ind
```

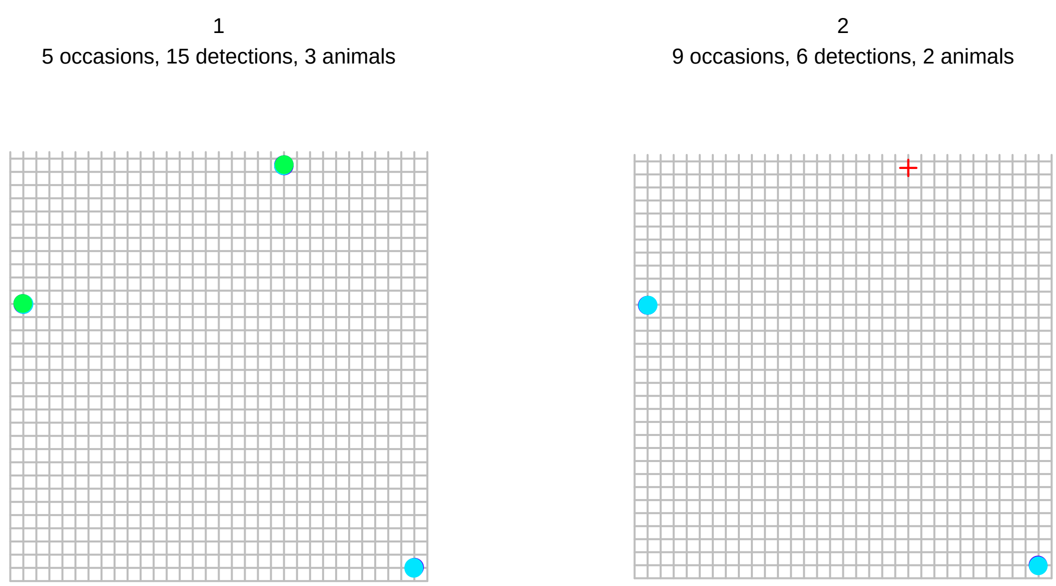
```
# create camera operation matrix (with problems/malfunction), same as above for multi-season occupancy
camop_season <- cameraOperation(CTtable         = camtrapsMultiSeason,
                                    stationCol   = "Station",
                                    setupCol     = "Setup_date",
                                    sessionCol   = "session",
                                    retrievalCol = "Retrieval_date",
                                    hasProblems  = TRUE,
                                    dateFormat   = dateFormat
)


# create capthist object
sdh_multi <- spatialDetectionHistory(recordTableIndividual = recordTableIndividualSample_season,
                               species               = "LeopardCat",
                               output                = "binary",
                               camOp                 = camop_season,
                               CTtable               = camtrapsMultiSeason,
                               stationCol            = "Station",
                               speciesCol            = "Species",
                               sessionCol            = "session",
```

```
                                    Xcol                      = "utm_x",
                                    Ycol                      = "utm_y",
                                    individualCol             = "Individual",
                                    recordDateTimeCol         = "DateTimeOriginal",
                                    recordDateTimeFormat      = "%Y-%m-%d %H:%M:%S",
                                    occasionLength            = 10,
                                    day1                      = "survey",
                                    includeEffort             = TRUE,
                                    timeZone                  = "Asia/Kuala_Lumpur",
                                    stationCovariateCols      = "utm_y",       # made up, a potential site cov
                                    individualCovariateCols = "Individual"   # made up, a potential individu
)


summary(sdh_multi)
```

```
## $`1`
## Object class       capthist
## Detector type      proximity
## Detector number    3
## Average spacing    2258.871 m
## x-range            523000 526000 m
## y-range            604000 607050 m
##
##  Usage range by occasion
##       1  2  3  4   5
## min 7.5 10 10 10 0.0
## max 9.5 10 10 10 4.5
##
## Counts by occasion
##                    1 2 3 4 5 Total
## n                  2 2 3 2 1    10
## u                  2 0 1 0 0     3
## f                  0 1 0 2 0     3
## M(t+1)             2 2 3 3 3     3
## losses             0 0 0 0 0     0
## detections         3 4 4 3 1    15
## detectors visited 2 3 3 2 1    11
## detectors used    3 3 3 3 2    14
##
## Individual covariates
##  Individual
##  1:1
##  2:1
##  3:1
##
##
## $`2`
## Object class       capthist
## Detector type      proximity
## Detector number    4
## Average spacing    2258.871 m
## x-range            523000 526000 m
```

```
## y-range            604000 607050 m
##
##  Usage range by occasion
##       1  2  3  4  5 6 7  8   9
## min 2.5 10 10 10  5 0 0  0 0.0
## max 4.5 10 10 10 10 3 6 10 1.5
##
## Counts by occasion
##                   1 2 3 4 5 6 7 8 9 Total
## n                 1 2 2 0 1 0 0 0 0     6
## u                 1 1 0 0 0 0 0 0 0     2
## f                 0 0 2 0 0 0 0 0 0     2
## M(t+1)            1 2 2 2 2 2 2 2 2     2
## losses            0 0 0 0 0 0 0 0 0     0
## detections        1 2 2 0 1 0 0 0 0     6
## detectors visited 1 1 2 0 1 0 0 0 0     5
## detectors used    4 4 4 4 4 1 1 1 1    24
##
## Individual covariates
##  Individual
##  1:1
##  2:1
```

```
par(mfrow = c(1,2))
plot(sdh_multi)
```



```
## 1  2
## 15  6
```

```
secr.fit.example <- secr.fit(capthist = sdh_multi,
                              start = c(-1, -2, 10))   # with starting values, since its only very lit
```

```
## Checking data
```

```
## Warning in verify.capthist(capthist, report = 1): Levels of factor covariate(s) differ between sessio
```

```
## Warning in secr.fit(capthist = sdh_multi, start = c(-1, -2, 10)): using default buffer width 100 m
```

```
## Preparing detection design matrices
## Preparing density design matrix
## Maximizing likelihood...
## Eval     Loglik        D       g0    sigma
##    1    -66.379  -1.0000  -2.0000  10.0000
##    2    -66.379  -1.0000  -2.0000  10.0000
##    3    -66.379  -1.0000  -2.0000  10.0000
##    4    -66.379  -1.0000  -2.0000  10.0000
##    5    -66.379  -1.0000  -2.0000  10.0000
##    6   -178.146  -1.2452 -11.9960   9.8556
##    7    -60.131  -1.1049  -6.2755   9.9382
##    8    -60.131  -1.1049  -6.2755   9.9382
##    9    -60.131  -1.1049  -6.2755   9.9382
##   10    -60.131  -1.1049  -6.2755   9.9383
##   11    -44.097  -0.9607  -5.3508   9.9516
##   12    -44.097  -0.9607  -5.3508   9.9516
##   13    -44.097  -0.9607  -5.3508   9.9516
##   14    -44.097  -0.9607  -5.3508   9.9516
##   15   -175.943  -0.6781  -1.0374  10.0124
##   16    -34.376  -0.9136  -4.6318   9.9617
##   17    -34.376  -0.9136  -4.6318   9.9617
##   18    -34.376  -0.9136  -4.6318   9.9617
##   19    -34.376  -0.9136  -4.6318   9.9617
##   20    -52.174  -1.0826  -2.2461   9.9950
##   21    -27.836  -0.9650  -3.9057   9.9718
##   22    -27.836  -0.9650  -3.9057   9.9718
##   23    -27.836  -0.9650  -3.9057   9.9718
##   24    -27.836  -0.9650  -3.9057   9.9718
##   25    -27.780  -1.1234  -3.2046   9.9816
##   26    -27.780  -1.1234  -3.2046   9.9816
##   27    -27.780  -1.1234  -3.2046   9.9816
##   28    -27.780  -1.1234  -3.2046   9.9816
##   29    -26.634  -1.0891  -3.5998   9.9762
##   30    -26.634  -1.0891  -3.5998   9.9762
##   31    -26.634  -1.0891  -3.5998   9.9762
##   32    -26.634  -1.0891  -3.5998   9.9762
##   33    -26.565  -1.1362  -3.5480   9.9770
##   34    -26.565  -1.1362  -3.5480   9.9770
##   35    -26.565  -1.1362  -3.5479   9.9770
##   36    -26.565  -1.1362  -3.5480   9.9770
##   37    -26.535  -1.1989  -3.5225   9.9775
```

```
## 38    -26.535  -1.1989  -3.5225   9.9775
## 39    -26.535  -1.1989  -3.5225   9.9775
## 40    -26.535  -1.1989  -3.5225   9.9775
## 41    -26.525  -1.2417  -3.5255   9.9776
## 42    -26.525  -1.2417  -3.5255   9.9776
## 43    -26.525  -1.2417  -3.5255   9.9776
## 44    -26.525  -1.2417  -3.5255   9.9776
## 45    -26.523  -1.2628  -3.5352   9.9777
## 46    -26.523  -1.2628  -3.5352   9.9777
## 47    -26.523  -1.2628  -3.5352   9.9777
## 48    -26.523  -1.2628  -3.5352   9.9777
## 49    -26.522  -1.2631  -3.5374   9.9777
## 50    -26.522  -1.2631  -3.5374   9.9777
## 51    -26.522  -1.2631  -3.5374   9.9777
## 52    -26.522  -1.2631  -3.5374   9.9777
## 53    -26.522  -1.2627  -3.5377   9.9778
## 54    -26.522  -1.2627  -3.5377   9.9778
## 55    -26.522  -1.2627  -3.5377   9.9778
## 56    -26.522  -1.2627  -3.5377   9.9778
## 57    -26.522  -1.2627  -3.5377   9.9779
## 58    -26.522  -1.2627  -3.5377   9.9779
## 59    -26.522  -1.2627  -3.5377   9.9779
## 60    -26.522  -1.2627  -3.5377   9.9779
## 61    -26.522  -1.2624  -3.5378   9.9787
## 62    -26.522  -1.2624  -3.5378   9.9787
## 63    -26.522  -1.2624  -3.5378   9.9787
## 64    -26.522  -1.2624  -3.5378   9.9787
## 65    -26.522  -1.2620  -3.5379   9.9805
## 66    -26.522  -1.2620  -3.5379   9.9805
## 67    -26.522  -1.2620  -3.5379   9.9805
## 68    -26.522  -1.2620  -3.5379   9.9806
## 69    -26.522  -1.2614  -3.5381   9.9864
## 70    -26.522  -1.2614  -3.5381   9.9864
## 71    -26.522  -1.2614  -3.5381   9.9864
## 72    -26.522  -1.2614  -3.5381   9.9864
## 73    -26.522  -1.2604  -3.5386  10.0016
## 74    -26.522  -1.2604  -3.5386  10.0016
## 75    -26.522  -1.2604  -3.5386  10.0016
## 76    -26.522  -1.2604  -3.5386  10.0016
## 77    -26.522  -1.2590  -3.5394  10.0407
## 78    -26.522  -1.2590  -3.5394  10.0407
## 79    -26.522  -1.2590  -3.5394  10.0407
## 80    -26.522  -1.2590  -3.5394  10.0407
## 81    -26.522  -1.2574  -3.5409  10.1281
## 82    -26.522  -1.2574  -3.5409  10.1281
## 83    -26.522  -1.2574  -3.5409  10.1281
## 84    -26.522  -1.2574  -3.5409  10.1281
## 85    -26.522  -1.2565  -3.5429  10.2897
## 86    -26.522  -1.2565  -3.5429  10.2897
## 87    -26.522  -1.2565  -3.5429  10.2897
## 88    -26.522  -1.2565  -3.5429  10.2897
## 89    -26.522  -1.2572  -3.5445  10.5337
## 90    -26.522  -1.2572  -3.5445  10.5337
## 91    -26.522  -1.2572  -3.5445  10.5337
```

```
## 92    -26.522 -1.2572 -3.5445 10.5337
## 93    -26.522 -1.2602 -3.5446 10.8393
## 94    -26.522 -1.2602 -3.5446 10.8393
## 95    -26.522 -1.2602 -3.5446 10.8393
## 96    -26.522 -1.2602 -3.5446 10.8393
## 97    -26.522 -1.2627 -3.5442 11.1149
## 98    -26.522 -1.2627 -3.5442 11.1149
## 99    -26.522 -1.2627 -3.5442 11.1149
## 100   -26.522 -1.2627 -3.5442 11.1149
## 101   -26.522 -1.2637 -3.5442 11.3617
## 102   -26.522 -1.2637 -3.5442 11.3617
## 103   -26.522 -1.2637 -3.5442 11.3617
## 104   -26.522 -1.2637 -3.5442 11.3617
## 105   -26.522 -1.2638 -3.5443 11.6604
## 106   -26.522 -1.2638 -3.5443 11.6604
## 107   -26.522 -1.2638 -3.5443 11.6604
## 108   -26.522 -1.2638 -3.5443 11.6604
## 109   -26.522 -1.2632 -3.5444 12.0166
## 110   -26.522 -1.2632 -3.5444 12.0166
## 111   -26.522 -1.2632 -3.5444 12.0166
## 112   -26.522 -1.2632 -3.5444 12.0166
## 113   -26.522 -1.2627 -3.5445 12.3560
## 114   -26.522 -1.2627 -3.5445 12.3560
## 115   -26.522 -1.2627 -3.5445 12.3560
## 116   -26.522 -1.2627 -3.5445 12.3560
## 117   -26.522 -1.2625 -3.5445 12.6624
## 118   -26.522 -1.2625 -3.5445 12.6624
## 119   -26.522 -1.2625 -3.5445 12.6624
## 120   -26.522 -1.2625 -3.5445 12.6624
## 121   -26.522 -1.2625 -3.5445 12.9888
## 122   -26.522 -1.2625 -3.5445 12.9888
## 123   -26.522 -1.2625 -3.5445 12.9888
## 124   -26.522 -1.2625 -3.5445 12.9889
## 125   -26.522 -1.2626 -3.5445 13.3421
## 126   -26.522 -1.2626 -3.5445 13.3421
## 127   -26.522 -1.2626 -3.5445 13.3421
## 128   -26.522 -1.2626 -3.5445 13.3421
## 129   -26.522 -1.2627 -3.5445 13.6813
## 130   -26.522 -1.2627 -3.5445 13.6813
## 131   -26.522 -1.2627 -3.5445 13.6813
## 132   -26.522 -1.2627 -3.5445 13.6813
## 133   -26.522 -1.2627 -3.5445 14.0030
## 134   -26.522 -1.2627 -3.5445 14.0030
## 135   -26.522 -1.2627 -3.5445 14.0030
## 136   -26.522 -1.2627 -3.5445 14.0030
## 137   -26.522 -1.2627 -3.5445 14.3403
## 138   -26.522 -1.2627 -3.5445 14.3403
## 139   -26.522 -1.2627 -3.5445 14.3403
## 140   -26.522 -1.2627 -3.5445 14.3403
## 141   -26.522 -1.2627 -3.5445 14.6932
## 142   -26.522 -1.2627 -3.5445 14.6932
## 143   -26.522 -1.2627 -3.5445 14.6932
## 144   -26.522 -1.2627 -3.5445 14.6932
## 145   -26.522 -1.2627 -3.5445 15.0328
```

```
##  146    -26.522  -1.2627  -3.5445  15.0328
##  147    -26.522  -1.2627  -3.5445  15.0328
##  148    -26.522  -1.2627  -3.5445  15.0328
##  149    -26.522  -1.2627  -3.5445  15.3652
##  150    -26.522  -1.2627  -3.5445  15.3652
##  151    -26.522  -1.2627  -3.5445  15.3652
##  152    -26.522  -1.2627  -3.5445  15.3652
##  153    -26.522  -1.2627  -3.5445  15.7107
##  154    -26.522  -1.2627  -3.5445  15.7107
##  155    -26.522  -1.2627  -3.5445  15.7107
##  156    -26.522  -1.2627  -3.5445  15.7108
##  157    -26.522  -1.2627  -3.5445  16.0514
##  158    -26.522  -1.2627  -3.5445  16.0514
##  159    -26.522  -1.2627  -3.5445  16.0514
##  160    -26.522  -1.2627  -3.5445  16.0515
##  161    -26.522  -1.2627  -3.5445  16.4049
##  162    -26.522  -1.2627  -3.5445  16.4049
##  163    -26.522  -1.2627  -3.5445  16.4049
##  164    -26.522  -1.2627  -3.5445  16.4049
##  165    -26.522  -1.2626  -3.5445  16.4049
##  166    -26.522  -1.2627  -3.5444  16.4049
##  167    -26.522  -1.2627  -3.5445  16.4065
##  168    -26.522  -1.2625  -3.5445  16.4049
##  169    -26.522  -1.2626  -3.5444  16.4049
##  170    -26.522  -1.2626  -3.5445  16.4065
##  171    -26.522  -1.2627  -3.5443  16.4049
##  172    -26.522  -1.2627  -3.5444  16.4065
##  173    -26.522  -1.2627  -3.5445  16.4082
## Completed in 78.86 seconds at 12:03:24 01 Feb 2024
```

```
summary(secr.fit.example)
```

```
## $versiontime
## [1] "4.6.4, run 12:02:05 01 Feb 2024, elapsed 78.86 s"
##
## $capthist
##              1 2
## Occasions    1 1
## Detections 15 6
## Animals      3 2
## Detectors    3 4
## Moves        5 2
## Animals2     3 2
##
## $trapsummary
## $trapsummary$`1`
## Object class       traps
## Detector type      count
## Detector number    3
## Average spacing    2258.871 m
## x-range            523000 526000 m
## y-range            604000 607050 m
##
##  Usage range by occasion
```

```
##        1
## min 37.5
## max 43.0
##
## $trapsummary$`2`
## Object class       traps
## Detector type      count
## Detector number    4
## Average spacing    2258.871 m
## x-range            523000 526000 m
## y-range            604000 607050 m
##
##  Usage range by occasion
##        1
## min 37.5
## max 63.0
##
##
## $detector
##       1       2
## "count" "count"
##
## $countmodel
## [1] "Binomial, size from usage"
##
## $mask
##   Cells Spacing Area
## 1    36      50    9
## 2    36      50    9
##
## $modeldetails
##     CL fixed distribution hcov
##  FALSE  none      poisson
##
## $AICtable
##             model   detectfn npar    logLik    AIC   AICc
##  D~1 g0~1 sigma~1 halfnormal    3 -26.52162 59.043 83.043
##
## $coef
##            beta      SE.beta       lcl        ucl
## D     -1.262694 4.474596e-01 -2.139699 -0.3856896
## g0    -3.544483 2.287552e-01 -3.992835 -3.0961309
## sigma 16.404893 9.060396e-09 16.404893 16.4048927
##
## $predicted
## $predicted$`session = 1`
##        link     estimate SE.estimate          lcl          ucl
## D       log 2.828908e-01 0.133190691 1.176903e-01 6.799816e-01
## g0    logit 2.807271e-02 0.006241503 1.811320e-02 4.326713e-02
## sigma   log 1.332154e+07 0.000000000 1.332154e+07 1.332154e+07
##
## $predicted$`session = 2`
##        link     estimate SE.estimate          lcl          ucl
## D       log 2.828908e-01 0.133190691 1.176903e-01 6.799816e-01
```

```
## g0     logit 2.807271e-02 0.006241503 1.811320e-02 4.326713e-02
## sigma    log 1.332154e+07 0.000000000 1.332154e+07 1.332154e+07
```