1. Organising raw camera trap images in camtrapR

Juergen Niedballa (camtrapr@gmail.com)

2026-02-08

Contents

Introduction

camtrapR provides functions for organising and managing camera trap images and for the preparation of occupancy and (spatial) capture-recapture analyses. The package handles JPG images and supports video files.

The five vignettes demonstrate the camtrapR workflow using the sample data set that is included in the package. Sample data were generated in a camera trapping survey in Sabah, Malaysian Borneo (A. Mohamed *et al. J. Mammal.*, 94, 82-89; 2013).

camtrapR is compatible with other software for camera trap data management. If you prefer to manage images and identify species using other software, you can usually do so. For example, Camelot and TRAPPER can export csv files that can be used as input for camtrapR. To maximize compatibility, camtrapR uses a simple data structure and standard data types (data frames and matrices) to store information about the camera trap survey and the detected species and individuals in separate data frames.

Installation

You can install the newest stable version from CRAN via:

```
install.packages("camtrapR")
```

Alternatively, you can install the development version with the latest features and bugfixes from GitHub (requires package remotes):

```
library(remotes)
install_github("jniedballa/camtrapR")
```

Afterwards, load camtrapR via:

```
library(camtrapR)
```

Documentation

The latest vignettes, function reference and changelog can be found on the camtrapR GitHub pages or the camtrapR page on CRAN. Source code can be found on GitHub.

Workflow

The camtrapR workflow consists of a series of functions that build upon one another in a logical sequence. Below is a list of functions grouped by work steps. Detailed information about each function can be found in the sections below, as well as in the other vignettes and the function help files.

- Organising raw images

1. `createStationFolders`
2. `fixDateTimeOriginal` (only if necessary)
3. `timeShiftImages` (only if necessary)
4. `imageRename`
5. `addCopyrightTag`

- Species identification and data extraction

1. `checkSpeciesNames`
2. `createSpeciesFolders`
3. `checkSpeciesIdentification`
4. `appendSpeciesNames`
5. `exifTagNames`
6. `recordTable`

- Data exploration and visualisation (species-level)

1. `detectionMaps`
2. `activityHistogram`
3. `activityDensity`
4. `activityRadial`
5. `activityOverlap`
6. `surveyReport`

- Individual identification and data extraction

1. `getSpeciesImages`
2. `recordTableIndividual`

- Preparing input for subsequent analyses

1. `cameraOperation`
2. `detectionHistory` (for occupancy analyses)
3. `spatialDetectionHistory` (for spatial capture-recapture analyses)
4. `communityModel` (for multi-species occupancy models)

General sampling situation

camtrapR was designed for camera trapping surveys in which an array of camera trap stations (each consisting of one or more camera traps per station) is operated in a study area. The vignettes will provide a guideline for how to use camtrapR in different sampling situations.

Video support (since version 2.0.0)

Since version 2.0.0, the functions `recordTable` and `recordTableIndividual` support video files. In addition `exifTagNames` can read and display video metadata via ExifTool.

Video support allows you to process video files and identify species and individuals using the same workflow as for images. With minimal effort, you can extract and tabulate these information using the familiar functions within camtrapR.

Since this is a new feature, please be sure to read the vignette sections on video and the help files of the relevant functions and check the example code. Also, please test the workflow before working on larger amounts of data. Unfortunately, I could not include a sample video in the package due to file size constraints, but I hope you will get the idea. And please, let me know about your experiences working with videos in camtrapR and possible problems. I would really appreciate the feedback to inmprove and extend video support.

Video date/time

The main problem in working with videos is the lack of a standardized date/time tag in the metadata. Only if such a tag is present, camtrapR can successfully extract the video date/time. Not all cameras save date and time in a suitable format in the metadata, so before conducting a study using videos, please make sure that your cameras record a suitable (i.e., persistent) date/time tag in the metadata. You can check if there is a suitable tag available in the video metadata:

1. Use `exifTagNames` to identify a usable video date/time tag
2. provide the name of the video date/time tag to `recordTable`

When using `exifTagNames`, make sure the date/time tag you choose is identical to the date/time shown in the data field of the video. In the example videos I tests, tags such as "QuickTime:CreateDate" worked well. The tags "File Creation Date/Time", "File Modification Date/Time" and "File Access Date/Time" are not suitable, since they are updated when the video files are moved or copied.

This also means that if there is no suitable date/time tag in the video metadata, camtrapR will not be able to return information for these videos and will omit them. It really depends on your camera model. So if you consider using video for your surveys, please first check (with that specific camera model) if there is a suitable date/time tag in the video metadata. A suitable date/time tag is one that doesn't change when the video file is moved, copied or otherwise modified. If a camera fails to record such a tag, camtrapR can't return proper date and time, making the videos useless.

digiKam tags & metadata

In digiKam, it is possible to assign tags to videos. Just like images, videos can be tagged with species or individual IDs, group size, sex, age group or whatever information you are interested in. BUT, in contrast to JPG images, digiKam cannot write these tags directly into the video metadata. Instead, video tags are stored in the digiKam database on your hard disk. To access tags assigned to videos, camtrapR has to read them directly from the digiKam database. This means that in order for camtrapR to read video tags, the videos need to be included in the digiKam database. Moving or copying the videos to a different location will strip the tags you assigned.

Users need to provide the filename of the digiKam database and the directory it is in (digiKam > Settings > configure digiKam > Database; the directory is stated there). In that menu, also ensure you use a SQLite database (the default). When the digiKam database information are provided to camtrapR, all the tags

you assigned to videos in digiKam will be extracted and tabulated just like image tags. See the help files of `recordTable` and `recordTableIndividual` for details.

Nomenclature

To facilitate understanding here is a glossary on how we use specific terms. It is standard camera trapping nomenclature.

- station: a station is a locality containing one or more camera traps (often a pair of two)
- camera trap: refers to the individual camera unit. One or more (often two) constitute a station
- occasion: a distinct sampling event of 1 or more days
- effort: number of active trapping days per station and occasion

ExifTool

camtrapR relies on the free and open-source software **ExifTool** written by Phil Harvey to extract metadata from images. ExifTool is freely available here and runs on Windows, Linux and MacOS. In order to use it, your system must be able to find the software. On MacOS, Linux and Unix it needs installation and should afterwards be found by the system without problems. How to set up Exiftool for Windows is detailed below. General installation instruction can be found here.

ExifTool for Windows

On Windows, ExifTool does not need installation. In order for Windows to be able to find ExifTool, follow the instructions found here. Be sure to download the stand-alone executable. The download file is a .zip which you need to unzip, rename the contained "exiftool(-k).exe" file to "exiftool.exe" and place it in, e.g., C:/Windows. Windows will be able to find ExifTool if exiftool.exe is placed in a directory that is contained in the Windows PATH variable (e.g. C:/Windows).

```
# Check which directories are in PATH (output not shown here)
Sys.getenv("PATH")
```

```
# Check if the system can find Exiftool (if output is empty "", system can't find Exiftool)
Sys.which("exiftool")
```

```
##                     exiftool
## "C:\\WINDOWS\\exiftool.exe"
```

If exiftool.exe cannot be placed in a PATH directory (e.g. due to limited user privileges), users can place exiftool.exe in any directory and then add that directory to PATH temporarily for use in the running R session using the function `addToPath`. This is necessary if the directory containing exiftool.exe is not in PATH by default.

```
# this is a dummy example assuming the directory structure: C:/Path/To/Exiftool/exiftool.exe
exiftool_dir <- "C:/Path/To/Exiftool"
addToPath(exiftool_dir)
```

Afterwards, Windows (via R) should be able to find ExifTool during the active session.

ExifTool performance

ExifTool is the performance bottleneck of the package functions. It can extract metadata at a rate of about 160 images per second (approx. 10,000 images per minute, measured on a laptop computer running Windows 7, Intel Core i5-2410M / 2.3 GHz, 6 GB RAM). Performance tends to be lower when ExifTool is first called and increases afterwards; and it decreases in situations with numerous camera trap stations with few images each (because ExifTool is called separately on every station directory and the associated overhead of loading ExifTool each time).

Camera trap station information

Before looking at how images are handled, let's first familiarise ourselves with how information about camera trap stations are stored. Camera trap station information are tabulated in a simple data frame with 1 row per station (if there was 1 camera per station) or 1 row per camera (if there was more than 1 camera per station). Information to be stored in this table are: station (and camera) IDs, geographic coordinates, setup and retrieval dates and possible dates when camera were not operational (because of malfunction, animal interference, empty batteries, full memory cards etc.). The table can be generated standard spreadsheet software and loaded into R.

```
# here is a sample camera trap station table
data("camtraps")
camtraps
```

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

Note that the sample data contain only one camera per station. Thus, there is no need to differentiate between station and camera IDs. If there was more than one camera per station, an additional column for camera IDs would be needed. Consequently, there would be one row for each camera and thus several rows per station.

Station / Camera IDs

Station and camera IDs can be any combination of characters and numbers. Please don't use underscores though, and more specifically, don't use double underscores ("___"), as these are used in a number of functions to separate bits of information (e.g. in file names created by `imageRename` or row names in `cameraOperation`).

Column names

Column names in camtrapR are not prescribed, allowing users the freedom to name their data as they wish, with one exception detailed below. In the sample table, 'Station' contains station IDs, 'utm_y' and 'utm_x' contain the station's geographic coordinates. 'Setup_date' and 'Retrieval_date' are the dates the stations were set up and retrieved from the field. All of these column names can be user-defined.

'Problem1_from' and 'Problem1_to' are optional columns to specify periods in which cameras/stations were not operational. If cameras or stations malfunctioned repeatedly, additional pairs of columns such as 'Problem2_from' and 'Problem2_to' can be added. These column names must follow the given pattern.

Date format

All dates in this table (setup, retrieval and problem columns) should be of data type "character" and can be formatted in any way that function `as.Date` can interpret. Since version 1.2, camtrapR also supports the more flexible and convenient naming conventions of package `lubridate`, which makes it easier to specify dates correctly.

camtrapR by default assumes the standard format "YYYY-MM-DD", e.g. "2015-12-31", and we recommend to use that format (it is also best for sorting by date). In any case, the date format must be consistent between the various date columns.

Two functions (`cameraOperation`, `surveyReport`) utilize these date information and therefore contain arguments to specify the date format (`dateFormat` and `CTDateFormat`, respectively). These arguments are used to interpret the character string in the table (e.g. "2015-12-31") as a date.

If these arguments contain the character "%" (characteristic of the format argument to `strptime`), they are passed to function `as.Date` as argument `format`. If they don't contain character "%", they are interpreted

as `orders` arguments to the function `parse_date_time` in the `lubridate` package.

Details on how to set it correctly can be found in the Details section of the help files of functions `as.Date` and `strptime` or `parse_date_time`.

Here is some information about date format in base R.

Here is some information about date format in lubridate.

Here are a few examples for December 1st 2015 ("generic" is the abstract representation of your date string (Y = year, M = month, D = day), "example" is a formatted example and "dateFormat argument" is what you'd specify in `dateFormat` and `CTDateFormat` in either the base R or `lubridate` naming convention):

generic	example	dateFormat argument (base)	dateFormat argument (lubridate)
"YYYY-MM-DD"	"2015-12-01"	"%Y-%m-%d"	"ymd"
"YYYY/MM/DD"	"2015/12/01"	"%Y/%m/%d"	"ymd"
"YYYY/MM/DD"	"2015/12/1"	"%Y/%m/%d"	"ymd"
"MM/DD/YY"	"12/01/15"	"%m/%d/%y"	"mdy"
"YYYYMMDD"	"20151201"	"%Y%m%d"	"ymd"
"DD.MM.YY"	"1.12.15"	"%d.%m.%y"	"dmy"
"DD-MMM-YY"	"01-Dec-15"	"%d-%b-%y"	"dmy"

In the sample table (`camtraps`), we use another date format for demonstration purposes.

Data type of the ID and date columns in the table must be `character` or `factor`. Please don't use data type `Date`. You can turn data type `Date` into `character` using `as.character`.

Station-/ Camera-level covariates

If needed, station-/camera-level covariates can be specified in this table, e.g. trail or habitat type, camera model etc. They can later be used as covariates, e.g. in occupancy models.

Organising raw camera trap images

After collecting images from camera traps, the images are saved into a directory structure like this:

rawImages/stationA

rawImages/stationB

If there was more than 1 camera per station, the station directories must contain camera subdirectories, e.g.

rawImages/stationA/camera1

rawImages/stationA/camera2

If you have more than 1 camera per station but don't separate the images from different cameras at this stage, you will not be able to do so at a later point. Later in the workflow, you can decide whether you would like to keep them separate or merge them (in the function `imageRename`).

Generally, you should not work on your raw data and instead keep them as a backup. If you rename you images with `imageRename`, the whole camtrapR workflow will take place in a copy of the images to prevent data loss.

Another important point is not to save any other data besides images in image directories. It may interfere with the operation of the package and ExifTool.

Saving raw images on your hard disk

The directories for the raw images can be created automatically using the function `createStationFolders` and the camera trap station table. As mentioned above, images can either be stored in station directories (if there was 1 camera per station) or in station/camera directories (if there was >1 camera per station). The behaviour is controlled by the function argument `cameras`. It specifies the camera ID column in the camera trap information table. If it is defined, camera subdirectories will be created within the station directories.

Here is an example in which station directories without camera subdirectories are created.

```
# create a temporary dummy directory for tests
# (normally, you'd set up an empty directory in a proper place, e.g. .../myStudy/rawImages)
wd_createStationDir <- file.path(normalizePath(tempdir(), winslash = "/"), "createStationFoldersTest")

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

```
## created 3 directories
```

```
StationFolderCreate1
```

```
##    station                                                                    directory created
## 1 StationA C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/createStationFoldersTest/StationA    TRUE
## 2 StationB C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/createStationFoldersTest/StationB    TRUE
## 3 StationC C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/createStationFoldersTest/StationC    TRUE
```

IMPORTANT: Please note that in this vignette station directories are set up in a temporary directory. That does not make any sense in real life and is done here solely to demonstrate how the functions works. So please don't do this at home! Instead, always work in permanent directories!

Once the directories are created, you can copy over your images from the memory cards.

Shifting date and time of images

There are situations in which the date and time of your images may be incorrect. Imagine you forgot to set the system time in one of your cameras or reset the time accidentally. Or think of a bug in the camera software that causes years to be wrong as happened in the cameras of a major manufacturer at the turn of the year 2015/2016. Another situation in which images times may need a little shift is when users wish to synchronise the record times between camera pairs.

In any case, systematic offsets of date and time recorded by cameras can be corrected easily. The function `timeShiftImages` does just that by utilising the date/time shift module of ExifTool. All you need is a table containing the time offset in a certain format, i.e.

```
data(timeShiftTable)
timeShiftTable
```

```
##    Station camera    timeshift sign
## 1 StationA     NA  1:0:0 0:0:0    -
## 2 StationB     NA 0:0:0 12:0:0    +
```

There is a `station` column with station IDs, `camera` is NA because there was 1 camera per station only in our example, `timeshift` is the amount by which to shift the time of all images at that station, and `sign` specifies the direction in which to shift time. Setting `sign` to `+` sets image time ahead by the amount specified in `timeshift`, `-` sets image times back. `timeshift` format is "year:month:day hour:minute:second", i.e. "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. In the above table, time stampes of images taken at StationA are shifted back by 1 year, and images taken at StationB are shifted ahead by 12 hours.

The function `timeShiftImages` should be run directly after saving the raw images on the hard disk.

```
# copy sample images to another location (so we don't mess around in the package directory)
wd_images_raw <- system.file("pictures/raw_images", package = "camtrapR")
file.copy(from = wd_images_raw, to = normalizePath(tempdir(), winslash = "/") , recursive = TRUE)
```

```
## [1] TRUE
```

```
wd_images_raw_copy <- file.path(normalizePath(tempdir(), winslash = "/"), "raw_images")

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

```
## C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images/StationA
```

```
## C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images/StationB
```

```
timeshift_run
```

```
##                                                      directory             n_directories
## 1 C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images/StationA     1 directories scanned     3
## 2 C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images/StationB     1 directories scanned     3
```

ExifTool has now updated the time tag of all images in the directories specified in the `timeShiftTable`. The original images are preserved as XYZ.JPG_original files (XYZ being the original file name). By setting argument `undo = TRUE`, the orginals are restored and the modified images are deleted. The function will check if the number of JPG_original files and JPG files in folders is identical. If not, the function will not work. It does not check for consistency of file names.

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

timeshift_undo
```

```
##                                                      directory n_images_undo
## 1 C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images/StationA             3
## 2 C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images/StationB             3
```

Date/Time problems with Reconyx Hyperfire cameras

Some Reconyx Hyperfire cameras (e.g. Hyperfire HC500) don't store date and time in the standard Exif format. Consequently, camtrapR cannot read the DateTimeOriginal tag from these images. This can be fixed easily by using the camtrapR function `fixDateTimeOriginal` or by following the instructions given by Mathias Tobler in the CameraBase documentation (page 7).

Renaming images

Raw images are copied from the raw image directories into a new location and renamed at the same time. The renamed file names follow this pattern (depending on whether there was one or more cameras per station):

StationID___Date___Time(X).JPG

StationID___CameraID___Date___Time(X).JPG

Station ID and camera ID are derived from the raw image directory structure created earlier with `createStationFolders` Date and time are taken from image Exif metadata (read using ExifTool). (X) is a numeric identifier that is assigned to all images taken at the same station (and camera, if applicable) within one minute. This is to avoid identical image names if images were taken in the same second (or in the same minute, if cameras record time in hours and minutes only).

Again, please be aware that temporary directories are used in the example. Set `outDir` to a permanent directory. In addition, images are not actually copied (argument `copyImages = FALSE`), but copying and renaming is mererly simulated in out example. So, in real world data, set `copyImages = TRUE`.

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

renaming.table2 <- imageRename(inDir              = wd_images_raw,
                               outDir             = wd_images_raw_renamed,
                               hasCameraFolders   = FALSE,
                               copyImages         = FALSE
  )
```

```
## StationA:    3 images      |==========          |   50%
## StationB:    3 images      |====================|  100%
```

```
# here is the information for a few images
head(renaming.table2)
```

```
##                                                                Directory    FileName  Station
## 1 C:/Users/Juergen/Documents/GitHub/camtrapR/inst/pictures/raw_images/StationA IMG0001.JPG StationA
## 2 C:/Users/Juergen/Documents/GitHub/camtrapR/inst/pictures/raw_images/StationA IMG0002.JPG StationA
## 3 C:/Users/Juergen/Documents/GitHub/camtrapR/inst/pictures/raw_images/StationA IMG0003.JPG StationA
## 4 C:/Users/Juergen/Documents/GitHub/camtrapR/inst/pictures/raw_images/StationB IMG0001.JPG StationB
## 5 C:/Users/Juergen/Documents/GitHub/camtrapR/inst/pictures/raw_images/StationB IMG0002.JPG StationB
## 6 C:/Users/Juergen/Documents/GitHub/camtrapR/inst/pictures/raw_images/StationB IMG0003.JPG StationB
##                                                                   outDir
## 1 C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images_renamed/StationA StationA__2009-04-10__05
## 2 C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images_renamed/StationA StationA__2009-04-21__00
## 3 C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images_renamed/StationA StationA__2009-04-22__20
## 4 C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images_renamed/StationB StationB__2009-04-05__00
## 5 C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images_renamed/StationB StationB__2009-04-05__23
## 6 C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/raw_images_renamed/StationB StationB__2009-04-07__00
```

So, the first image, "IMG0001.JPG" was renamed to "StationA___2009-04-10___05-07-00(1).JPG". Here is what the columns mean:

column	content
`Directory`	the raw image directory
`FileName`	raw image file name
`Station`	the station the image was taken at

column	content
`Camera`	the camera ID (here NA because there were no camera subdirectories)
`DateTimeOriginal`	Date and time in R-readable format
`DateReadable`	whether the date could be read from the metadata or not (TRUE = yes)
`outDir`	the directory the renamed image was copied to
`filename_new`	the new file name
`CopyStatus`	whether the images was copied successfully (TRUE = yes)

At this point you created a copy of your raw images and renamed them with station ID, date and time.

Writing a copyright tag into image metadata

The function `addCopyrightTag` can write a copyright tag into image Exif metadata. While this can be useful when sharing or publishing data, it is not required in the camtrapR workflow. The function will recursively find all images in `inDir` and its subdirectories and writes a user-defined text into the Copyright field of the Exif metadata.

In doing so, Exiftool normally keeps the original images as .JPG_original files. Note that this behaviour will instantly double the number of images in inDir and the disk space required. If this is not desired, users can set the function argument `keepJPG_original = FALSE` (default is `TRUE`) in order to replace the original files without creating *.JPG_original files. I recommend to first try this on a copy of a subdataset and to then check the function output. If there are warnings like "Warning: [minor] Maker notes could not be parsed" image makernotes may be unreadable afterwards (they are proprietary, undocumented formats). On the other hand, you may never need these for analysing camera trapping data.

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

```
## [1] TRUE
```

```
wd_images_ID_copy <- file.path(normalizePath(tempdir(), winslash = "/"), "sample_images_species_dir")

# define an example copyright tag
copyrightTagToAdd <- "Your Name (Your Organisation)"

# write the tag to images
addCopyrightTag(inDir        = wd_images_ID_copy,
                copyrightTag = copyrightTagToAdd,
                askFirst     = FALSE)                  # askFirst = FALSE prevents function from asking
```

```
##    14 directories scanned    69 image files updated
```

When we check the outcome with function exifTagNames, we find field “Copyright” in line 27.

```
exiftagnames_extracted <- exifTagNames(wd_images_ID_copy)
```

```
## Metadata of:
## C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/sample_images_species_dir/StationA/PBE/StationA__2009
```

```
knitr::kable(exiftagnames_extracted, row.names = TRUE)
```

	tag_group	tag_name	value
1	ExifTool	ExifToolVersion	12.09
2	ExifTool	Warning	Invalid EXIF text encoding for UserComment
3	File	FileName	StationA___2009-04-21___00-40-00(1).JPG

	tag_group	tag_name	value
4	File	Directory	C:/Users/Juergen/AppData/Local/Temp/RtmpmiQZvg/sample_images_species_dir/S
5	File	FileSize	27 kB
6	File	FileModifyDate	2026:02:08 17:02:41+08:00
7	File	FileAccessDate	2026:02:08 17:02:41+08:00
8	File	FileCreateDate	2026:02:08 17:02:39+08:00
9	File	FilePermissions	rw-rw-rw-
10	File	FileType	JPEG
11	File	FileTypeExtension	jpg
12	File	MIMEType	image/jpeg
13	File	ExifByteOrder	Little-endian (Intel, II)
14	File	ImageWidth	100
15	File	ImageHeight	75
16	File	EncodingProcess	Baseline DCT, Huffman coding
17	File	BitsPerSample	8
18	File	ColorComponents	3
19	File	YCbCrSubSampling	YCbCr4:2:2 (2 1)
20	EXIF	Make	G4
21	EXIF	Model	CUDDEBACK
22	EXIF	XResolution	72
23	EXIF	YResolution	72
24	EXIF	ResolutionUnit	inches
25	EXIF	ModifyDate	2009:04:21 00:40:00
26	EXIF	YCbCrPositioning	Co-sited
27	EXIF	Copyright	Your Name (Your Organisation)
28	EXIF	ExifVersion	0200
29	EXIF	DateTimeOriginal	2009:04:21 00:40:00
30	EXIF	CreateDate	2009:04:21 00:40:00
31	EXIF	ComponentsConfiguration	Y, Cb, Cr, -
32	EXIF	ShutterSpeedValue	1
33	EXIF	UserComment	1 ATTEMPT G1H
34	EXIF	FlashpixVersion	0100
35	EXIF	ColorSpace	sRGB
36	EXIF	ExifImageWidth	100
37	EXIF	ExifImageHeight	75
38	EXIF	Compression	JPEG (old-style)
39	EXIF	ThumbnailOffset	450
40	EXIF	ThumbnailLength	25353
41	EXIF	ThumbnailImage	(Binary data 25353 bytes, use -b option to extract)
42	Composite	ImageSize	100x75
43	Composite	Megapixels	0.007
44	Composite	ShutterSpeed	1

The copyright tag will henceforth be visible in digiKam and other image management software.

Now that the raw images are organised, the next step is species identification, covered in the next vignette.