

Package ‘cam2model’

October 14, 2026

Title Preprocessing Trap Camera Data for Machine Learning Models

Version 0.3.0

Description Provides tools for preprocessing camera trap image data to prepare it for training machine learning models. It automates tasks such as extracting EXIF metadata, processing and analyzing image attributes, selecting relevant images (e.g., daytime photos), sampling subsets, and managing file organization. These features streamline the workflow for ecological research and wildlife monitoring projects. Additionally, the package includes tools for generating exploratory visualizations and creating annotation datasets.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Imports dplyr, tidyr, lubridate, stringr, ggplot2, readr, fs, exifr, rlang, tibble, progressr

Suggests future, future.apply, knitr, rmarkdown, testthat

Depends R (>= 4.1.0)

VignetteBuilder knitr

NeedsCompilation no

Author Edlin Guerra-Castro [aut, cph] (ORCID:
<<https://orcid.org/0000-0003-3529-4507>>),
Arturo Sanchez-Porras [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-1691-286X>>)

Maintainer Arturo Sanchez-Porras <sp.arturo@gmail.com>

Repository CRAN

Date/Publication 2026-10-14 15:32:11 UTC

Contents

copy_in_batches	2
generate_plot	4
process_metadata	6
process_training_images	7
rename_images	9

copy_in_batches	<i>Copy files into batch folders using fs</i>
-----------------	---

Description

Copies files listed in a data frame into destination subfolders of fixed size. This is useful when a large image collection must be reorganized into evenly sized groups (for example, 250 files per folder) while preserving a record of the operation.

Usage

```
copy_in_batches(  
  df,  
  output_dir,  
  batch_size = 250L,  
  folder_prefix = "batch_",  
  rename_with_id = TRUE,  
  overwrite = FALSE,  
  dry_run = FALSE,  
  verbose = TRUE  
)
```

Arguments

df	A data.frame containing at least the columns id and path.
output_dir	Character scalar. Root directory where batch folders will be created.
batch_size	Integer scalar. Number of files per destination folder. Default is 250.
folder_prefix	Character scalar. Prefix used to name batch folders. Default is "batch_".
rename_with_id	Logical. If TRUE (default), output file names are prefixed with id to reduce the risk of name collisions.
overwrite	Logical. If TRUE, existing files in the destination are overwritten. Default is FALSE.
dry_run	Logical. If TRUE, no files are copied, but the returned report shows the intended destination structure. Default is FALSE.
verbose	Logical. If TRUE, prints progress messages. Default is TRUE.

Details

Files are assigned to folders according to their row order in df.

Expected minimum input columns (contract): df must include id and path.

Produced columns: returns one row per valid input record with id, path, source_exists, batch_id, batch_folder_name, dest_file_name, dest_path, copied, and status.

I/O side-effects: creates output_dir and batch subfolders when dry_run = FALSE; copies files into those subfolders unless dry_run = TRUE.

The function creates one folder per batch using zero-padded sequential numbering, for example:

- batch_001
- batch_002
- batch_003

Destination file names are generated from the source file name. When rename_with_id = TRUE, the output name becomes:

"{id}_{original_filename}"

If duplicates remain within a batch, they are disambiguated with `base::make.unique()`.

The function returns a data frame describing the outcome for each row, including whether the source existed, the assigned batch, the destination path, and the copy status.

Value

A data.frame with one row per valid input file and the following columns:

id Identifier from the input data frame.

path Original source path.

source_exists Logical; whether the source file existed.

batch_id Integer batch number assigned to the file.

batch_folder_name Name of the destination batch folder.

dest_file_name Destination file name.

dest_path Full destination path.

copied Logical indicating whether the copy succeeded. NA in dry_run mode.

status Character status label such as "copied", "source_not_found", "destination_exists_not_overwritten", "copy_failed", or "dry_run".

See Also

`rename_images()`, `process_metadata()`, `generate_plot()`, `process_training_images()`

Examples

```
# Create temporary source and output directories
src_dir <- tempfile("cam2model_batch_src")
out_dir <- tempfile("cam2model_batch_out")
dir.create(src_dir, recursive = TRUE, showWarnings = FALSE)
dir.create(out_dir, recursive = TRUE, showWarnings = FALSE)

file1 <- file.path(src_dir, "img1.jpg")
file2 <- file.path(src_dir, "img2.jpg")
file3 <- file.path(src_dir, "img3.jpg")

writeBin(charToRaw("a"), file1)
```

```

writeBin(charToRaw("b"), file2)
writeBin(charToRaw("c"), file3)

df <- data.frame(
  id = c("A01", "A02", "A03"),
  path = c(file1, file2, file3),
  stringsAsFactors = FALSE
)

# Dry run: inspect the planned organization without copying
report_dry <- copy_in_batches(
  df = df,
  output_dir = out_dir,
  batch_size = 2,
  dry_run = TRUE,
  verbose = FALSE
)

report_dry[, c("id", "batch_folder_name", "dest_file_name", "status")]

# Actual copy
report <- copy_in_batches(
  df = df,
  output_dir = out_dir,
  batch_size = 2,
  dry_run = FALSE,
  verbose = FALSE
)

report[, c("id", "dest_path", "status")]

```

generate_plot

Generate Custom Plots from Metadata

Description

Generates ggplot2 visualizations for processed camera-trap metadata, with flexible x/y mappings, optional color/faceting, and optional filtering.

Usage

```

generate_plot(
  processed_metadata,
  x,
  y,
  color = NULL,
  facets = NULL,
  filter_expr = NULL,
  angle = 0
)

```

Arguments

processed_metadata	A data frame containing processed metadata.
x	Bare column name to use for the x-axis.
y	Bare column name to use for the y-axis.
color	Bare column name (optional) to map to point/line color.
facets	Bare column name (optional) used in facet_wrap().
filter_expr	Expression (optional) used to filter the metadata before plotting.
angle	Numeric. Angle for x-axis labels. Default is 0.

Details

Expected minimum input columns (contract): the columns referenced in x and y, and optionally those used in color, facets, and filter_expr.

Produced columns/artifacts: none (no data is modified on disk). The function returns a ggplot object combining geom_line() and geom_point().

I/O side-effects: none.

Value

A ggplot object.

See Also

[rename_images\(\)](#), [process_metadata\(\)](#), [process_training_images\(\)](#), [copy_in_batches\(\)](#)

Examples

```
demo_metadata <- data.frame(  
  Image_dttm = as.POSIXct("2024-08-01 00:00:00", tz = "UTC") + 0:5 * 3600,  
  bLuma = c(100, 120, 140, 130, 150, 160),  
  Camera = rep(c("C1", "C2"), each = 3),  
  stringsAsFactors = FALSE  
)  
  
p <- generate_plot(  
  processed_metadata = demo_metadata,  
  x = Image_dttm,  
  y = bLuma,  
  color = Camera,  
  facets = Camera  
)  
p
```

process_metadata	<i>Process Camera Trap Metadata</i>
------------------	-------------------------------------

Description

Extracts EXIF metadata from images in a directory, processes the UserComment column, and saves the resulting metadata to a CSV file.

Usage

```
process_metadata(output_dir, output_file, UserComment = FALSE)
```

Arguments

output_dir	Character. Path to the main directory containing camera folders with renamed images.
output_file	Character. Relative file path (from output_dir) for the output CSV where metadata will be saved. The parent directory for this file must already exist inside output_dir.
UserComment	Logical. Default FALSE. When TRUE, attempts to parse and expand the EXIF UserComment field into additional metadata columns. Set to FALSE if parsing fails for your files.

Details

Expected minimum input columns (internal contract): none. This function reads EXIF tags directly from image files under output_dir and appends a Camera column derived from each first-level folder name.

Produced columns: varies by EXIF availability. Output always includes available EXIF fields plus Camera; when UserComment = TRUE and parsing is successful, it may also include Image_dttm, File_date, File_hms, File_hour, File_minute, and parsed fields such as temp/bLuma.

I/O side-effects: writes one CSV to file.path(output_dir, output_file). The function does not create missing parent directories for output_file.

Value

A tibble with processed metadata. Typical contract columns used by downstream functions include Directory and FileName (required by [process_training_images\(\)](#)).

See Also

[rename_images\(\)](#), [generate_plot\(\)](#), [process_training_images\(\)](#), [copy_in_batches\(\)](#)

Examples

```
## Not run:
# Requires real image files with EXIF metadata.
renamed_dir <- tempfile("cam2model_renamed_")
dir.create(renamed_dir, recursive = TRUE)
dir.create(file.path(renamed_dir, "camera_A"), recursive = TRUE)

# Copy real camera-trap images into `renamed_dir/camera_A` before running.
meta <- process_metadata(
  output_dir = renamed_dir,
  output_file = "metadata.csv",
  UserComment = FALSE
)

file.exists(file.path(renamed_dir, "metadata.csv"))
head(meta)

## End(Not run)
```

process_training_images

Process and Select Images for Training

Description

Filters metadata according to one or more rules, samples a per-directory proportion of images, optionally copies sampled images to a destination directory, and generates CSV files for bookkeeping and annotation.

Usage

```
process_training_images(
  metadata_file,
  dest_dir,
  output_sample_file,
  output_annotation_file,
  sample_proportion = 0.5,
  filters = NULL,
  min_per_dir = 1,
  max_per_dir = Inf,
  copy_images = FALSE,
  parallel = FALSE,
  workers = NULL,
  show_progress = TRUE
)
```

Arguments

metadata_file	Character. Path to the CSV file containing metadata.
dest_dir	Character. Path to the directory where outputs are written.
output_sample_file	Character. Name of the CSV file storing the sampled image list.
output_annotation_file	Character. Name of the CSV file storing the annotation template.
sample_proportion	Numeric in (0, 1]. Proportion of images to sample within each directory.
filters	A list of filtering rules. Each rule must be a list with: • var: character, name of the metadata column to filter by. • min: lower threshold (or NULL). • max: upper threshold (or NULL). If NULL, no filtering is applied.
min_per_dir	Integer. Minimum number of images to sample per directory.
max_per_dir	Integer. Maximum number of images to sample per directory.
copy_images	Logical. If TRUE, sampled images are copied to dest_dir.
parallel	Logical. If TRUE, file copying is done in parallel.
workers	Integer or NULL. Number of workers for parallel copying.
show_progress	Logical. If TRUE, show a progress bar using progressr.

Details

Expected minimum input columns (contract): Directory and FileName must be present in metadata_file.

Produced columns: returns sampled rows with at least Directory, FileName, optional File_date/File_hour (if present in input), and file_path; adds dest_file and copied when copy_images = TRUE.

I/O side-effects:

- Creates dest_dir if it does not exist.
- Writes sampled CSV to file.path(dest_dir, output_sample_file).
- Writes annotation template CSV to file.path(dest_dir, output_annotation_file).
- Optionally copies sampled files into dest_dir when copy_images = TRUE.

Value

A tibble with sampled images and file paths used for training.

See Also

[rename_images\(\)](#), [process_metadata\(\)](#), [generate_plot\(\)](#), [copy_in_batches\(\)](#)

Examples

```

src_dir <- tempfile("cam2model_train_src")
dest_dir <- tempfile("cam2model_train_out")
dir.create(src_dir, recursive = TRUE)

# Create minimal files that can be referenced from metadata
f1 <- file.path(src_dir, "img1.jpg")
f2 <- file.path(src_dir, "img2.jpg")
writeBin(charToRaw("a"), f1)
writeBin(charToRaw("b"), f2)

metadata_tbl <- data.frame(
  Directory = c(src_dir, src_dir),
  FileName = c("img1.jpg", "img2.jpg"),
  File_hour = c(10, 14),
  stringsAsFactors = FALSE
)
metadata_csv <- tempfile("cam2model_metadata_", fileext = ".csv")
readr::write_csv(metadata_tbl, metadata_csv)

sampled <- process_training_images(
  metadata_file = metadata_csv,
  dest_dir = dest_dir,
  output_sample_file = "sampled_images.csv",
  output_annotation_file = "annotation_data.csv",
  sample_proportion = 1,
  copy_images = FALSE,
  show_progress = FALSE,
  parallel = FALSE
)

sampled
file.exists(file.path(dest_dir, "sampled_images.csv"))
file.exists(file.path(dest_dir, "annotation_data.csv"))

```

rename_images

Rename and flatten camera trap images

Description

This wrapper function builds a renaming plan and immediately executes it, allowing the user to flatten and rename all images from a sampling-site directory in a single call.

Usage

```

rename_images(
  input_dir,
  output_dir,
  site_name = NULL,

```

```

recursive = TRUE,
extensions = c("jpg", "jpeg", "png", "tif", "tiff"),
keep_relative_path = TRUE,
date_fields = c("DateTimeOriginal", "FileModifyDate"),
parallel = FALSE,
workers = NULL,
overwrite = FALSE,
return_plan = FALSE
)

```

Arguments

input_dir	Character. Path to the root directory of a sampling site.
output_dir	Character. Path to the root output directory.
site_name	Character or NULL. Name of the sampling site. If NULL, the basename of input_dir is used.
recursive	Logical. Should files be searched recursively? Default TRUE.
extensions	Character vector. Allowed image extensions.
keep_relative_path	Logical. If TRUE, part of the relative path is included in the new file name.
date_fields	Character vector. EXIF date fields to try in order.
parallel	Logical. Should the renaming plan be built in parallel? Default FALSE.
workers	Integer or NULL. Number of workers when parallel = TRUE.
overwrite	Logical. Should existing files be overwritten? Default FALSE.
return_plan	Logical. If TRUE, returns a list with both plan and result. If FALSE, returns only the executed plan with copy status.

Details

Expected minimum input columns (internal contract): none. The function reads image files directly from input_dir and builds metadata internally.

Produced columns:

- If return_plan = FALSE, returns the execution table with columns from the renaming plan plus copied.
- If return_plan = TRUE, returns a list with:
 - plan: planning table that includes original_path, new_name, and new_path.
 - result: the same rows plus copied.

I/O side-effects: creates <output_dir>/<site_name> when needed and copies files to that folder using standardized names.

Value

A tibble with copy results, or a list containing both the plan and the execution result.

See Also

[process_metadata\(\)](#), [generate_plot\(\)](#), [process_training_images\(\)](#), [copy_in_batches\(\)](#)

Examples

```
## Not run:
# Requires real image files on disk.
raw_dir <- tempfile("cam2model_raw_")
out_dir <- tempfile("cam2model_out_")
dir.create(raw_dir, recursive = TRUE)
dir.create(out_dir, recursive = TRUE)

# Copy real camera-trap images into `raw_dir` before running.
result <- rename_images(
  input_dir = raw_dir,
  output_dir = out_dir,
  site_name = "site_demo",
  overwrite = FALSE
)
head(result)

## End(Not run)
```

Index

`base::make.unique()`, [3](#)

`copy_in_batches`, [2](#)

`copy_in_batches()`, [5](#), [6](#), [8](#), [11](#)

`generate_plot`, [4](#)

`generate_plot()`, [3](#), [6](#), [8](#), [11](#)

`process_metadata`, [6](#)

`process_metadata()`, [3](#), [5](#), [8](#), [11](#)

`process_training_images`, [7](#)

`process_training_images()`, [3](#), [5](#), [6](#), [11](#)

`rename_images`, [9](#)

`rename_images()`, [3](#), [5](#), [6](#), [8](#)