Package ‘camtrapdp’

July 5, 2024

Title Read and Manipulate Camera Trap Data Packages

Version 0.3.1

Date 2024-07-05

Description Read and manipulate Camera Trap Data Packages ('Camtrap DP').
'Camtrap DP' (<https://camtrap-dp.tdwg.org>) is a data exchange format
for camera trap data. With 'camtrapdp' you can read, filter and
transform data (including to Darwin Core) before further analysis in
e.g. 'camtraptor' or 'camtrapR'.

License MIT + file LICENSE

URL https://github.com/inbo/camtrapdp,
https://inbo.github.io/camtrapdp/

BugReports https://github.com/inbo/camtrapdp/issues

Imports cli, dplyr, frictionless (>= 1.1.0), memoise, purrr, readr

Suggests lubridate, testthat (>= 3.0.0), tibble, xml2

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Peter Desmet [aut, cre] (<https://orcid.org/0000-0002-8442-8025>,
Research Institute for Nature and Forest (INBO)),
Sanne Govaert [aut] (<https://orcid.org/0000-0002-8939-1305>, Research
Institute for Nature and Forest (INBO)),
Pieter Huybrechts [aut] (<https://orcid.org/0000-0002-6658-6062>,
Research Institute for Nature and Forest (INBO)),
Damiano Oldoni [aut] (<https://orcid.org/0000-0003-3445-7562>, Research
Institute for Nature and Forest (INBO)),
Research Institute for Nature and Forest (INBO) [cph]
(https://www.vlaanderen.be/inbo/en-gb/),
LifeWatch Belgium [fnd] (https://lifewatch.be)

Maintainer Peter Desmet <peter.desmet@inbo.be>

Repository CRAN

Date/Publication 2024-07-05 14:10:05 UTC

Contents

`check_camtrapdp` *Check a Camera Trap Data Package object*

Description

Checks if an object is a Camera Trap Data Package object with the required properties.

Usage

```
check_camtrapdp(x)
```

Arguments

x	Camera Trap Data Package object, as returned by `read_camtrapdp()`.

Value

x invisibly or error.

Examples

```
x <- example_dataset()
check_camtrapdp(x) # Invisible return of x if valid
```

`deployments` *Get or set deployments*

Description

`deployments()` gets the deployments from a Camera Trap Data Package object.
`deployments<-()` is the assignment equivalent. It should only be used within other functions, where the expected data structure can be guaranteed.

Usage

```
deployments(x)

deployments(x) <- value
```

Arguments

`x`	Camera Trap Data Package object, as returned by `read_camtrapdp()`.
`value`	A data frame to assign as deployments.

Value

`tibble::tibble()` data frame with deployments.

See Also

Other accessor functions: `events()`, `locations()`, `media()`, `observations()`, `taxa()`

Examples

```
x <- example_dataset()
# Get deployments
deployments(x)

# Set deployments (not recommended outside a function)
deployments(x) <- head(deployments(x), 1)
```

`events` *Get events*

Description

Gets the (unique) events from the observations of a Camera Trap Data Package object. Only observations with `observationLevel == "event"` are considered.

Usage

```
events(x)
```

Arguments

`x`	Camera Trap Data Package object, as returned by `read_camtrapdp()`.

Value

`tibble::tibble()` data frame with the events, containing the following columns:

- `deploymentID`
- `eventID`
- `eventStart`
- `eventEnd`

See Also

Other accessor functions: `deployments()`, `locations()`, `media()`, `observations()`, `taxa()`

Examples

```
x <- example_dataset()
events(x)
```

`example_dataset` *Read the Camtrap DP example dataset*

Description

Reads the Camtrap DP example dataset. This dataset is maintained and versioned with the Camtrap DP standard.

Usage

```
example_dataset()
```

Value

Camera Trap Data Package object.

Examples

```
example_dataset()
```

`filter_deployments` *Filter deployments*

Description

Subsets deployments in a Camera Trap Data Package object, retaining all rows that satisfy the conditions.

Usage

```
filter_deployments(x, ...)
```

Arguments

`x`	Camera Trap Data Package object, as returned by `read_camtrapdp()`.
`...`	Filtering conditions, see `dplyr::filter()`.

Details

- Media are filtered on associated `deploymentID`.
- Observations are filtered on associated `deploymentID`.

Value

`x` filtered.

See Also

Other filter functions: `filter_media()`, `filter_observations()`

Examples

```
x <- example_dataset()

# Filtering returns x, so pipe with deployments() to see the result
x %>%
  filter_deployments(deploymentID == "62c200a9") %>%
  deployments()

# Filtering on deployments also affects associated media and observations
x_filtered <- filter_deployments(x, deploymentID == "62c200a9")
media(x_filtered)
observations(x_filtered)

# Filtering on multiple conditions (combined with &)
x %>%
  filter_deployments(latitude > 51.0, longitude > 5.0) %>%
  deployments()
```

```
# Filtering on dates is easiest with lubridate
library(lubridate, warn.conflicts = FALSE)
x %>%
  filter_deployments(
    deploymentStart >= lubridate::as_date("2020-06-19"),
    deploymentEnd <= lubridate::as_date("2020-08-30")
  ) %>%
  deployments()
```

`filter_media` *Filter media*

Description

Subsets media in a Camera Trap Data Package object, retaining all rows that satisfy the conditions.

Usage

```
filter_media(x, ...)
```

Arguments

`x`	Camera Trap Data Package object, as returned by `read_camtrapdp()`.
`...`	Filtering conditions, see `dplyr::filter()`.

Details

- Deployments are not filtered.
- Observations are filtered on associated `mediaID` (for media-based observations) and `eventID` (for event-based observations).

Value

`x` filtered.

See Also

Other filter functions: `filter_deployments()`, `filter_observations()`

Examples

```
x <- example_dataset()

# Filtering returns x, so pipe with media() to see the result
x %>%
  filter_media(captureMethod == "timeLapse") %>%
  media()

# Filtering on media also affects associated observations, but not deployments
```

```
x_filtered <- filter_media(x, favorite == TRUE)
observations(x_filtered)

# Filtering on multiple conditions (combined with &)
x %>%
  filter_media(captureMethod == "activityDetection", filePublic == FALSE) %>%
  media()

# Filtering on datetimes is easiest with lubridate
library(lubridate, warn.conflicts = FALSE)
x %>%
  filter_media(
    timestamp >= lubridate::as_datetime("2020-08-02 05:01:00"),
    timestamp <= lubridate::as_datetime("2020-08-02 05:02:00")
  ) %>%
  media()
```

`filter_observations` *Filter observations*

Description

Subsets observations in a Camera Trap Data Package object, retaining all rows that satisfy the conditions.

Usage

```
filter_observations(x, ...)
```

Arguments

`x`	Camera Trap Data Package object, as returned by `read_camtrapdp()`.
`...`	Filtering conditions, see `dplyr::filter()`.

Details

- Deployments are not filtered.
- Media are filtered on associated `mediaID` (for media-based observations) and `eventID` (for event-based observations). Filter on `observationLevel == "media"` to only retain directly linked media.

Value

`x` filtered.

See Also

Other filter functions: `filter_deployments()`, `filter_media()`

Examples

```
x <- example_dataset()

# Filtering returns x, so pipe with observations() to see the result
x %>%
  filter_observations(observationType == "animal") %>%
  observations()

# Filtering on observations also affects associated media, but not deployments
x %>%
  filter_observations(scientificName == "Vulpes vulpes", observationLevel == "event") %>%
  media()
x %>%
  filter_observations(scientificName == "Vulpes vulpes", observationLevel == "media") %>%
  media()

# Filtering on multiple conditions (combined with &)
x %>%
  filter_observations(
    deploymentID == "577b543a",
    scientificName %in% c("Martes foina", "Mustela putorius")
  ) %>%
  observations()

# Filtering on datetimes is easiest with lubridate
library(lubridate, warn.conflicts = FALSE)
x %>%
  filter_observations(
    eventStart >= lubridate::as_datetime("2020-06-19 22:00:00"),
    eventEnd <= lubridate::as_datetime("2020-06-19 22:10:00")
  ) %>%
  observations()
```

`locations` *Get locations*

Description

Gets the (unique) locations from the deployments of a Camera Trap Data Package object.

Usage

```
locations(x)
```

Arguments

`x`	Camera Trap Data Package object, as returned by `read_camtrapdp()`.

Value

`tibble::tibble()` data frame with the locations, containing the following columns:

- `locationID`
- `locationName`
- `latitude`
- `longitude`
- `coordinateUncertainty`

See Also

Other accessor functions: `deployments()`, `events()`, `media()`, `observations()`, `taxa()`

Examples

```
x <- example_dataset()
locations(x)
```

`media` *Get or set media*

Description

`media()` gets the media from a Camera Trap Data Package object.
`media<-()` is the assignment equivalent. It should only be used within other functions, where the expected data structure can be guaranteed.

Usage

```
media(x)

media(x) <- value
```

Arguments

`x`	Camera Trap Data Package object, as returned by `read_camtrapdp()`.
`value`	A data frame to assign as media.

Value

`tibble::tibble()` data frame with media.

See Also

Other accessor functions: `deployments()`, `events()`, `locations()`, `observations()`, `taxa()`

Examples

```
x <- example_dataset()
# Get media
media(x)

# Set media (not recommended outside a function)
media(x) <- head(media(x), 1)
```

`observations` *Get observations*

Description

`observations()` gets the observations from a Camera Trap Data Package object.
`observations<-()` is the assignment equivalent. It should only be used within other functions, where the expected data structure can be guaranteed.

Usage

```
observations(x)

observations(x) <- value
```

Arguments

x	Camera Trap Data Package object, as returned by `read_camtrapdp()`.
value	A data frame to assign as observations.

Value

`tibble::tibble()` data frame with observations.

See Also

Other accessor functions: `deployments()`, `events()`, `locations()`, `media()`, `taxa()`

Examples

```
x <- example_dataset()
# Get the observations
observations(x)

# Set observations (not recommended outside a function)
observations(x) <- head(observations(x), 1)
```

print.camtrapdp *Print a Camera Trap Data Package*

Description

Prints a human-readable summary of a Camera Trap Data Package, as an extension of `frictionless::print.datapackage(`

Usage

```
## S3 method for class 'camtrapdp'
print(x, ...)
```

Arguments

`x`	Camera Trap Data Package object, as returned by `read_camtrapdp()`.
`...`	Further arguments, they are ignored by this function.

Value

`print()` with a summary of the Camera Trap Data Package object.

Examples

```
x <- example_dataset()

# Print a summary
print(x)

# Print a summary after filtering
filter_deployments(x, deploymentID == "62c200a9")
```

read_camtrapdp *Read a Camera Trap Data Package*

Description

Reads files from a Camera Trap Data Package (Camtrap DP) into memory.

Usage

```
read_camtrapdp(file)
```

Arguments

`file`	Path or URL to a `datapackage.json` file.

Value

Camera Trap Data Package object.

Assign taxonomic information

Camtrap DP metadata has a `taxonomic` property that can contain extra information for each `scientificName` found in observations. Such information can include higher taxonomy (`family`, `order`, etc.) and vernacular names in multiple languages.

This function **will automatically include this taxonomic information in observations**, as extra columns starting with `taxon.`.

Assign eventIDs

Observations can contain two classifications at two levels:

Media-based observations (`observationLevel = "media"`) are based on a single media file and are directly linked to it via `mediaID`.

Event-based observations (`observationLevel = "event"`) are based on an event, defined as a combination of `eventID`, `eventStart` and `eventEnd`. This event can consist of one or more media files, but is not directly linked to these.

This function **will automatically assign `eventID`s to media**, using `media.deploymentID = event.deploymentID` and `eventStart <= media.timestamp <= eventEnd`. Note that this can result in media being linked to multiple events (and thus being duplicated), for example when events and sub-events were defined.

Examples

```
file <- "https://raw.githubusercontent.com/tdwg/camtrap-dp/1.0/example/datapackage.json"
x <- read_camtrapdp(file)
x
```

`taxa` *Get taxa*

Description

Gets the (unique) scientific names and associated taxonomic information from the observations of a Camera Trap Data Package object.

Usage

```
taxa(x)
```

Arguments

`x`	Camera Trap Data Package object, as returned by `read_camtrapdp()`.

Value

`tibble::tibble()` data frame with the taxonomic information, containing at least a `scientificName` column.

See Also

Other accessor functions: `deployments()`, `events()`, `locations()`, `media()`, `observations()`

Examples

```
x <- example_dataset()
taxa(x)
```

`version` *Get Camtrap DP version*

Description

Extracts the version number used by a Camera Trap Data Package object. This version number indicates what version of the Camtrap DP standard was used.

Usage

```
version(x)
```

Arguments

`x`	Camera Trap Data Package object, as returned by `read_camtrapdp()`. Also works on a Frictionless Data Package, as returned by `frictionless::read_package()`.

Details

The version number is derived as follows:

1. The `version` attribute, if defined.
2. A version number contained in `x$profile`, which is expected to contain the URL to the used Camtrap DP standard.
3. `x$profile` in its entirety (can be `NULL`).

Value

Camtrap DP version number (e.g. `1.0`).

Examples

```
x <- example_dataset()
version(x)
```

`write_dwc` *Transform a Camera Trap Data Package to a Darwin Core Archive*

Description

Transforms a Camera Trap Data Package object to a Darwin Core Archive.

Usage

```
write_dwc(x, directory)
```

Arguments

`x`	Camera Trap Data Package object, as returned by `read_camtrapdp()`.
`directory`	Path to local directory to write files to.

Value

CSV and `meta.xml` files written to disk. And invisibly, a list of data frames with the transformed data.

Transformation details

This function **follows recommendations** in Reyserhove et al. (2023) doi:10.35035/doc0qzp2x37 and transform data to:

- An Occurrence core.
- An Audubon/Audiovisual Media Description extension.
- A `meta.xml` file.

Key features of the Darwin Core transformation:

- The Occurrence core contains one row per observation (`dwc:occurrenceID` = `observationID`).
- Only observations with `observationType` = `"animal"` and `observationLevel` = `"event"` are included, thus excluding observations that are (of) humans, vehicles, blanks, unknowns, unclassified and media-based.
- Deployment information is included in the Occurrence core, such as location, habitat, `dwc:samplingProtocol`, deployment duration in `dwc:samplingEffort` and `dwc:parentEventID` = `deploymentID` as grouping identifier.
- Event information is included in the Occurrence core, as event duration in `dwc:eventDate` and `dwc:eventID` = `eventID` as grouping identifier.
- Media files are included in the Audubon/Audiovisual Media Description extension, with a foreign key to the observation. A media file that is used for more than one observation is repeated.
- Metadata is used to set the following record-level terms:
  - `dwc:datasetID` = `id`.

- `dwc:datasetName` = `title`.
- `dwc:collectionCode`: first source in `sources`.
- `dcterms:license`: license (`name`) in `licenses` with scope `data`. The license (`name`) with scope `media` is used as `dcterms:rights` in the Audubon Media Description extension.
- `dcterms:rightsHolder`: first contributor in `contributors` with role `rightsHolder`.
- `dwc:dataGeneralizations`: set if `coordinatePrecision` is defined.

Examples

```
x <- example_dataset()
write_dwc(x, directory = "my_directory")

# Clean up (don't do this if you want to keep your files)
unlink("my_directory", recursive = TRUE)
```

Index