Package: camtrapReport (via r-universe)

July 1, 2026

Type Package

Title Camera-Trap Report Generator

Version 1.0.28

Date 2026-06-29

Description Provides an extensible workflow for processing camera-trap data in Camtrap-DP format and automatically generating reproducible data-status and ecological reports. The package supports data-quality checks, ecological summaries, visualisations, maps, tables and modular report sections for wildlife-monitoring applications.

License MIT

URL https://spatialecology.github.io/camtrapReport/, https://github.com/spatialecology/camtrapReport

BugReports https://github.com/spatialecology/camtrapReport/issues

Depends R (>= 4.1.0)

Imports data.table, dplyr, shiny, glue, htmltools, jsonlite, knitr, lubridate, methods, rlang, rmarkdown, sf, sp, taxize, terra, tibble, tidyr

Suggests activity, corrplot, DT, dygraphs, ggplot2, ggrepel, gridExtra, gt, htmlwidgets, iNEXT, kableExtra, leaflet, leaflet.extras, magick, purrr, remotes, rgbif, scales, shiny, spOccupancy, spatstat, stringr, stats, suncalc, taxadb, testthat (>= 3.0.0), unmarked, xml2

VignetteBuilder knitr

Encoding UTF-8

Config/testthat/edition 3

RoxygenNote 7.3.3

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libglpk-dev make libicu-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

Repository https://camera-traps.r-universe.dev

Date/Publication 2026-07-01 10:55:05 UTC

RemoteUrl https://github.com/spatialecology/camtrapReport

RemoteRef HEAD

RemoteSha 8be191c127eefb224adbd099fac60409c29f8d7e

Contents

`add_Module` *Add a report module*

Description

Add a new YAML report module to the camtrapReport module library.

Usage

```
add_Module(x, before, after, test, object)

## S4 method for signature 'character'
add_Module(x, before, after, test, object)
```

Arguments

x	Path to a YAML module file.
before	Optional module name before which the new module should be inserted.
after	Optional module name after which the new module should be inserted.
test	Logical. If TRUE, test the module before adding it.
object	Optional camReport object used for testing the module.

Value

Invisibly returns information about the added module.

---

add_Module *Managing modules*

Description

A list of functions are available to manage modules in the package including listing the existing modules, adding a new one or deleting an existing one, or moving the position of a module in a report

Usage

```
add_Module(x,before,after,test)

move_Module(name,before,after,parent,level0)

remove_Module(name,recursive)

empty_trash(name,id)

list_Modules(tree,brief,include_trash,validate)
```

Arguments

x	The new module YAML filename
before	The name of module before which the new module should be added
after	The name of module after which the new module should be added
test	logical; specifying whether the module should be tested
name	The module name
parent	The parent name of module (modules are organised hierarchically as parent and childs)
level0	Optional; names of the modules in root: c("introduction", "methods", "results", "acknowledgement", "appendix")

recursive	logical; when it is TRUE, by deleting a parent module, its child modules are also deleted to avoid any inconsistencies in the report
id	The module id
tree	logical; specifies whether the list of modules should represents their hierarchical tree structure
brief	logical; if tree is FALSE, it specifies whether brief module information should be returned
include_trash	logical; if tree is FALSE, it specifies whether deleted modules in trash should also be listed
validate	logical; whether the modules should be tested for their validity

Details

Given that the package is extensible, designed based on a modular architecture, a set of functions are provided to facilitate module management.

A module refers to an object designed to contain the contents of a section in the report. These contents can be texts, tables, executable R codes, etc. Multiple interfaces are available for a user to define a module.

An easy way is to use a YAML file (a template can be used) to define a module. While the file is created, it can be added to the package using the `add_Module` function.

Other functions can also be used to list the existing modules, delete or recover a module, or move a module to a different location within the hierarchical structure the modules are organised.

Author(s)

Elham Ebrahimi

References

Ebrahimi et al. (2026) camtrapReport: xxx

Examples

```
list_Modules()
```

`camData` *Create camera trap data object*

Description

The camera-trap data, read as camtraptor data package, is the input.

Usage

```
camData(data,habitat,study_area,...)
```

Arguments

data	either character which is the filename of the cameratrap data (as ZIP or Json file), or a datapackage object read through camtraptor (the read_camtrap_dp function)
habitat	data.frame of habitat types
study_area	either name of a shapefile or a SpatVector object defining spatial boundary of a study site.
...	additional arguments

Details

The records of the input data package are used to build a camReport object, containing processed camera-trap data and texts, graphs, etc. used to automate generating a report.

Value

a ReferenceClass

Author(s)

Elham Ebrahimi <eebrahimi.bio@gmail.com>

References

ebrahimni et al. XXX

Examples

```
## Not run:

# filename of dataset: "veleuw.zip"

habitat <- read.csv('hanitat.csv')

cm <- camData("veleuw.zip",habitat,study_area='study_area.shp')

cm


## End(Not run)
```

camReport-classes *camReport classes*

Description

A set of S4 and Reference classes to manage camera-trap data and information.

Slots

Slots for camReport objects:

`title` a camReport object

`subtitle` contains the species data

`reportObjects` a list of modules

`statusReportObjects` a list of modules

Author(s)

Elham Ebrahimi

References

Ebrahimi et al. (2025) xxx

empty_trash *Empty the module trash*

Description

Permanently remove deleted modules from the trash folder.

Usage

```
empty_trash(name, id)

## S4 method for signature 'ANY'
empty_trash(name, id)
```

Arguments

`name`	Optional module name to purge.
`id`	Optional deletion batch ID to purge.

Value

Invisibly returns the updated trash index.

gui *Launch the camtrapReport GUI*

Description

Opens an interactive Shiny interface for configuring and generating camtrapReport outputs from an existing ‘camReport‘ object.

Usage

```
gui(object,launch.browser,max_upload_mb, ...)
```

Arguments

object	A ‘camReport‘ object, usually created with ‘camData()‘
launch.browser	Logical. If ‘TRUE‘, the application is opened in the default web browser.
max_upload_mb	Numeric. Maximum file-upload size in megabytes. The default is 2000.
...	Additional arguments passed to [shiny::runApp()]

Details

The function starts a shiny-based web-server and lunch a GUI interface.

Author(s)

Elham Ebrahimi

References

Ebrahimi et al. (2026) xxx

Examples

```
## Not run:
cm <- camData("data-folder")

gui(cm)



## End(Not run)
```

`info` *Get or set field values in a* `camReport` *object*

Description

This function facilitates to extract specific fields (e.g., title, authors, etc.) from a `camReport` object, or update the value of a field.

Usage

```
## S4 method for signature 'camReport'
info(x, name)

## S4 replacement method for signature 'camReport'
info(x, name)<- value
```

Arguments

`x`	A `camReport` object
`name`	character; names of the field from/to which the information is retrieved/assigned
`value`	the new value to assign to the specified field (name)

Details

The camReport object,created by the `camData` function, contains information extracted from the camera-trap dataset. To control some of these details go to report (e.g., title, subtitle, authors, institute, siteName, description, sampling,...), a user can check their values using the `info` function or update the value of a certain field.

Author(s)

Elham Ebrahimi

References

Ebrahimi et al. (2025) xxx

See Also

camData

Examples

```
## Not run:
cm <- camData("data-folder")

info(cm)

xi <- info(cm, name=c('title','authors'))

xi

info(cm,'title') <- 'This is NEW title...'


## End(Not run)
```

`install_All` *Install packages required by camtrapReport*

Description

Installs packages required for the full camtrapReport workflow, including packages used by optional report modules.

Usage

```
install_All(
  pkgs = NULL,
  update = FALSE,
  include_dev = FALSE,
  include_github = TRUE,
  include_gitlab = TRUE,
  ...
)
```

Arguments

`pkgs`	Optional character vector of package names to install. If 'NULL', the default camtrapReport package list is used.
`update`	Logical. If 'TRUE', reinstall GitHub/GitLab packages and attempt to install or update CRAN packages.
`include_dev`	Logical. If 'TRUE', also install developer packages such as 'testthat', 'pkgdown', 'covr', 'usethis' and 'devtools'.
`include_github`	Logical. If 'TRUE', install required GitHub packages.
`include_gitlab`	Logical. If 'TRUE', install required GitLab packages if a GitLab package list is available.
`...`	Additional arguments passed to 'install.packages()'.

Value

Invisibly returns ‘TRUE‘ if all required packages are available, otherwise ‘FALSE‘.

`install_All` *Install packages required by camtrapReport*

Description

Installs packages required for the full camtrapReport workflow, including packages used by optional report modules.

Usage

```
install_All(pkgs,update,...)
```

Arguments

`pkgs`	Optional character vector of package names to install. If ‘NULL‘, the default camtrapReport package list is used.
`update`	Logical. If ‘TRUE‘, reinstall GitHub/GitLab packages and attempt to install or update CRAN packages.
`...`	Additional arguments passed to ‘install.packages()‘

Details

The function extracts the list of packages specified in the modules and make sure they are installed, otherwise, some report sections may be excluded from the report.

Author(s)

Elham Ebrahimi <eebrahimi.bio@gmail.com>

References

ebrahimni et al. (under-review) camtrapReport: an R package for...

Examples

```
## Not run:

install_All()


## End(Not run)
```

list_Modules *List available report modules*

Description

List modules available in the camtrapReport module library.

Usage

```
list_Modules(tree, brief, include_trash, validate)

## S4 method for signature 'ANY'
list_Modules(tree, brief, include_trash, validate)
```

Arguments

tree	Logical. If TRUE, return modules as a tree table.
brief	Logical. If TRUE, return a brief table.
include_trash	Logical. If TRUE, include active trash records.
validate	Logical. If TRUE, validate module YAML files.

Value

A data frame, or a list containing modules and trash records.

listReportSections *List report sections*

Description

Lists report sections currently stored in a ‘camReport‘ object.

Usage

```
listReportSections(x)

## S4 method for signature 'camReport'
listReportSections(x)
```

Arguments

x	A ‘camReport‘ object.

Value

A data frame describing report section names, titles, parents and paths.

move_Module *Move a report module*

Description

Move an existing module before or after another module, or under a new parent.

Usage

```
move_Module(name, before, after, parent, level0)

## S4 method for signature 'character'
move_Module(name, before, after, parent, level0)
```

Arguments

name	Name of the module to move.
before	Optional module name before which to move the module.
after	Optional module name after which to move the module.
parent	Optional new parent module.
level0	Character vector defining the root-level module order.

Value

Invisibly returns the updated module information table.

remove_Module *Remove a report module*

Description

Move a module, and optionally its child modules, to the module trash folder.

Usage

```
remove_Module(name, recursive)

## S4 method for signature 'character'
remove_Module(name, recursive)
```

Arguments

name	Name of the module to remove.
recursive	Logical. If TRUE, remove child modules as well.

Value

Invisibly returns information about the deleted module batch.

`report` *Generating automated reports based on a* `camReport` *object*

Description

These functions facilitate generating two reports (data quality status, and ecological insights) automatically. Each report is an HTML rendered by creating an RMarkdown file based on the modules added to the `camReport` object.

Usage

```
report(object,filename,view, test)

status(object,filename,view)
```

Arguments

`object`	The camReport object created by the `camData` function
`filename`	optional (default is "report"); a character specifying filename or path/filename to write the report and rmarkdown files. The default location to write the files is the data folder
`view`	logical (default = TRUE); specifies whether the renderred html file should be displayed!
`test`	logical (default = FALSE); specifies whether the sections should be tested!

Details

By default (if filename is not specified), two files including "report.html" and "report.rmd" ("data_status.html" and "data_status.rmd" for the data status report) will be written in the folder of the camera-trap dataset.

If ‘test = TRUE‘ is used, in case any error is caused in the report generation procedure, a testing procedure is started to test each section and exclude the problematic section.

Author(s)

Elham Ebrahimi

References

Ebrahimi et al. (2025) xxx

See Also

camData

Examples

```
## Not run:
cm <- camData("data-folder")

report(cm, view=T) # ecological insights

status(cm, view=T) #  data quality status



## End(Not run)
```

`reportSection` *Creates a reproducible subsection of the report.*

Description

This function creates an object which keeps and manages the contents of a subsection in the report. Each subsection (report-section) may keep both text and a chunk of R codes. The object can be created by users and be added to the report to make the report generation flexible and extensible.

Usage

```
reportSection(name,title,parent,txt,code_setting,packages,code)
```

Arguments

`name`	a character which is used as a unique name of the object.
`title`	a character which specifies the title header of the subsection, appeared in the report
`parent`	if NULL, the subsection is considered as the level 1, but a subsection can be added as the second or third level of previously added subsections (i.e., subsection of a subsection)
`txt`	a character or a list with texts.
`code_setting`	if a chunk of R code is provided in the `code` argument, the setting of the chunk can be provided here within a brackt; example: `{ c(echo=FALSE,results="asis") }`
`packages`	a character vector specifying the name of required packages
`code`	a chunk of R code can be provided as an R script within bracket { }

Details

The name of the subsections should be unique, meaning two subsections can not have the same name. If a new subsection with the same name as previously added subsections is added to the report, the old one will be replaced by the new one.

In the code chuck, to get access the camReport object and its fields or methods, you need to use "object" in the code (see examples).

Author(s)

Elham Ebrahimi

References

Ebrahimi et al. (2025) xxx

See Also

camData

Examples

```
tx <- reportSection(name='introduction',title='Introduction',parent=NULL,txt="This is introduction section...")

tx




# in the following text section, we add an R chunk code which gets access to the camReport object
# through using the object keyword:

tx2 <- reportSection(name='method',title='Methods',parent=NULL,txt="Here, we show the usage of ...",
                     code_setting={c(echo=FALSE,results="asis",warn=FALSE)},
                     packages=c("gt","dplyr"),
                     code = {

                       object$camera_setup |>
                       gt() |>
                       tab_header(
                       title = md("**Camera Deployment Summary**"),
                  subtitle = md("**Table 1.** Details of camera deployments per year")) |>
                       cols_label(
                       year = "Year",
                       number_camtraps = "Camera Traps",
                       deployment_period = "Deployment Period",
                       setup_period = "Setup Period",
                       pickup_period = "Pickup Period") |>
                       tab_options(
                       table.font.size = px(14),
                       heading.title.font.size = px(16),
```

```
                        heading.subtitle.font.size = px(12),
                        row.striping.include_table_body = TRUE) |>
                        opt_row_striping() |>
                        tab_style(style = list(
                        cell_borders(sides = "bottom", color = "gray", weight = px(1))),
                        locations = cells_body()) |>
                        tab_style(
                  style = list(cell_text(weight = "bold")),locations = cells_column_labels())


                       }
                       )

  tx2
```

---

restore_Module *Restore a deleted report module*

Description

Restore a module from the module trash folder.

Usage

```
restore_Module(name, batch_id, test)

## S4 method for signature 'character'
restore_Module(name, batch_id, test)
```

Arguments

name	Name of the module to restore.
batch_id	Optional deletion batch ID.
test	Logical. If TRUE, test the restored module.

Value

Invisibly returns information about restored modules.

section_names *Get available report section names*

Description

Returns the names of available report sections/modules. Users can optionally keep only selected sections or exclude selected sections.

Usage

```
section_names(keep, exclude)

## S4 method for signature 'ANY'
section_names(keep, exclude)
```

Arguments

keep	Optional character vector of section/module names to keep.
exclude	Optional character vector of section/module names to exclude.

Value

A character vector of section/module names.

section_names *Return the name of available report sections or update/specify the report sections*

Description

This function can help to select a subset of sections for generating the report.

Usage

```
section_names(keep,exclude)

sections(x,n)
```

Arguments

keep	a character vector (default = 'all') specifies which report sections should be kept in the report
exclude	an optional character vector to specify which sections should be excluded
x	The 'camReport' object created using the camData function
n	a character vector to specify the names of report sections to update which report sections should be used in the report

Details

The ‘section_names‘ function can help to retrieve and specify the right section names in the ‘sections‘ function

Author(s)

Elham Ebrahimi

References

Ebrahimi et al. (2025) xxx

See Also

updateReportSections

Examples

```
## Not run:
cm <- camData('file.zip')

n <- section_names()

n

section_name(cm,n)

report(cm)


## End(Not run)
```

`status` *Generate a Data Status Check Report*

Description

Generates an automated data-status report from a ‘camReport‘ object. The report summarises key information about data completeness, spatial and temporal coverage, annotation quality and validation status.

Usage

```
status(object, filename, view)

## S4 method for signature 'camReport'
status(object, filename = "data_status", view)
```

Arguments

object	A ‘camReport‘ object.
filename	Output filename or file path without extension. Defaults to ‘"data_status"‘.
view	Logical. If ‘TRUE‘, the generated HTML report is opened after rendering.

Value

Invisibly returns the path to the generated HTML report.

---

testSection *Test a sub-section object (.textSection class)*

Description

If a sub-section object is created using the reportSection function, it can be tested to see how it looks like in a report.

Usage

```
testSection(x,object,view)
```

Arguments

x	The subsection object created using the reportSection function
object	The camReport object created using the camData function
view	logical (default = TRUE); specifies whether the renderred html file would be displayed!

Details

The object (camReport) is only required if the chunk of R code in the section requires the object!

Author(s)

Elham Ebrahimi

References

Ebrahimi et al. (2025) xxx

See Also

camData

Examples

```
tx <- reportSection(name='introduction',title='Introduction',parent=NULL,txt="This is introduction section...",
                    code = {

                    # R code:

                    print(plot(1:10))

                    })

tx

## Not run: testSection(tx)
```

`updateReportSection` *Updating content of report sections*

Description

This function can be used to update the content (text, code) of a section (module) within a `camReport` object.

Usage

```
updateReportSection(x,section,text,title,code,code_name,code_setting,packages,append_text,append_c

listReportSections(x)
```

Arguments

`x`	A camReport object created by the `camData` function
`section`	A character to refer to a report section (name or title of the report)
`text`	optional, to update the text body of the report section
`title`	optional, a new title for the section
`code`	optional, to update or add the R code chunk
`code_name`	optional, if R code chunk is provided, its name can be specified
`code_setting`	optional, setting for the new R chunk
`packages`	optional, name of R packages required by the new R chunk
`append_text`	optional; whether the new text should be appended to the previous text (default: FALSE)
`append_code`	optional; whether the new R code should be appended to the previous R code (default: FALSE)

Details

To find a report section, either title or name can be used in the `section` argument. The `listReportSections` is useful to list all sections added to a camReport object and check its exact name.

Author(s)

Elham Ebrahimi

References

Ebrahimi et al. (2025) xxx

See Also

info

Examples

```
## Not run:
cm <- camData("data-folder")

listReportSections(cm)



## End(Not run)
```

`updateReportSection` *Update a report section*

Description

Updates the title, text, code chunk, code settings or package list of an existing report section in a ‘camReport‘ object.

Usage

```
updateReportSection(
  x,
  section,
  text,
  title,
  code,
  code_name,
  code_setting,
```

```
  packages,
  append_text,
  append_code
)

## S4 method for signature 'camReport'
updateReportSection(
  x,
  section,
  text,
  title,
  code,
  code_name,
  code_setting,
  packages,
  append_text,
  append_code
)
```

Arguments

x	A ‘camReport‘ object.
section	A single character string identifying the section by name or title.
text	Optional replacement text.
title	Optional replacement title.
code	Optional replacement or appended R code. Code can be supplied as a character string or inside braces.
code_name	Optional code chunk name. Required if a section contains multiple chunks and a specific chunk should be updated.
code_setting	Optional R Markdown chunk settings.
packages	Optional character vector of packages required by the chunk.
append_text	Logical. If ‘TRUE‘, append text to the existing section text.
append_code	Logical. If ‘TRUE‘, append code to the existing code chunk.

Value

Invisibly returns the updated ‘camReport‘ object.

Index