

Package: camtrapReport (via r-universe)

October 8, 2026

Type Package

Title Camera-Trap Report Generator

Version 1.0.64

Description Provides an extensible workflow for processing camera-trap data in the 'Camtrap DP' format and automatically generating reproducible data-status and ecological reports in HTML or PDF format. The package supports data-quality checks, ecological summaries, visualisations, maps, tables and modular report sections for wildlife-monitoring applications. Optional taxonomic enrichment queries the Global Biodiversity Information Facility Species API <https://techdocs.gbif.org/en/openapi/v1/species> through 'taxize'.

License MIT + file LICENSE

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, jsonlite, lubridate, knitr,
methods, rmarkdown, terra

Suggests roxygen2, pak, pagedown, testthat (>= 3.1.7), xml2

VignetteBuilder knitr

SystemRequirements Google Chrome or Microsoft Edge (optional, for PDF output)

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs chromium cmake libgdal-dev gdal-bin libgeos-dev make libuv1-dev libproj-dev libsqlite3-dev zlib1g-dev

Repository <https://ropensci-staging.r-universe.dev>
Date/Publication 2026-10-08 04:09:23 UTC
RemoteUrl <https://github.com/spatialecology/camtrapReport>
RemoteRef HEAD
RemoteSha 9adc2f742e4843ad24af436d1c314da6972956f0

Contents

camData,character-method	2
camReport-classes	3
gui,camReport-method	5
info,camReport-method	6
install_all	8
modules	9
report	11
reportProfile	13
reportSection,character-method	14
section_names	16
testSection,.textSection-method	18
updateReportSection	19
Index	22

camData,character-method
<i>Read camera-trap data in Camtrap DP format</i>

Description

Create a [camReport](#) object from a Camtrap DP dataset.

Usage

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

Arguments

data	A character string giving the path to a Camtrap DP dataset, provided as a ZIP file or an extracted dataset directory.
habitat	An optional data frame containing habitat information for camera locations. The default is NULL.
study_area	An optional study-area boundary provided as a spatial file path, a SpatVector, or an sf object. The default is NULL.
update	A logical value (default FALSE) specifying whether to recreate a previously saved camReport object.
...	Additional arguments. These are currently reserved for future use.

Details

The function reads the input Camtrap DP dataset and creates a `camReport` object. The resulting object contains processed camera-trap data, metadata, summaries, report text, report modules, and data-status information used to generate automated reports.

If habitat information is provided, it is linked to camera locations and used in habitat-related summaries. If a study-area boundary is provided, it is stored with the object and used in spatial summaries where relevant.

The `camReport` object is saved within an extracted dataset directory as `__camReport_Object.rds`. A later call using that directory reuses the saved object unless `update = TRUE`, in which case the object is recreated.

Value

A `camReport` object.

See Also

[report\(\)](#), [status\(\)](#), [info\(\)](#), [gui\(\)](#)

Examples

```
source_dataset <- system.file(
  "external",
  "dataset",
  package = "camtrapReport"
)
example_dataset <- tempfile("camtrapReport-example-")
dir.create(example_dataset)
invisible(file.copy(
  list.files(source_dataset, full.names = TRUE),
  example_dataset,
  recursive = TRUE
))

cm <- camData(example_dataset)

cm

unlink(example_dataset, recursive = TRUE, force = TRUE)
```

`camReport-classes`*camReport classes*

Description

Class definitions used by `camtrapReport` to store camera-trap data, report metadata, report sections, and intermediate report objects.

Value

This is a documentation-only topic and does not return a value.

camReport class

The `camReport` class is the main object class used by `camtrapReport`. Objects of this class are usually created with `camData()`. A `camReport` object stores the camera-trap dataset, report metadata, processed summaries, report modules, status-report modules, and settings used for report generation.

Main fields

Important fields in a `camReport` object include:

- `data`: A list containing the camera-trap data tables, including observations, deployments, media, locations, sequences, and taxonomy.
- `habitat`: Optional habitat information linked to camera locations.
- `study_area`: Optional spatial object or path describing the study area.
- `siteName`: Name of the study site.
- `title`: Title used in the generated report.
- `subtitle`: Subtitle used in the generated report.
- `authors`: Author or contributor text used in the generated report.
- `institute`: Institute or organisation text used in the generated report.
- `description`: Study-area or project description used in the generated report.
- `years`: Years included in the report summaries.
- `group_definition`: Definitions of species or observation groups used in the report.
- `setting`: Report settings, including selected focus groups, `rem_parameter_scope` ("pooled" or "annual"), `rem_reps`, `rem_seed`, and `map_basemap` ("offline", "cartodb", "esri", or "openstreetmap"). The offline mode is the default; the OpenStreetMap mode includes an OpenTopoMap fallback for failed tiles. Bundled report modules select their REM scope explicitly; the standard profile is pooled and the EOW profile is annual.
- `data_status`: Data-status summaries generated from the input dataset.
- `reportObjectElements`: Report modules and related objects used to generate the ecological report.
- `statusReportObjects`: Objects used to generate the data-status report.

Supporting classes

`camtrapReport` also defines supporting classes and class unions used internally, including `camInfo`, `.Rchunk`, `.textSection`, `characterORnull`, `characterORlist`, `characterORlistORnull`, `listORnull`, and `data.frameORnull`.

See Also

`camData()`, `report()`, `status()`, `info()`, `reportSection()`

Examples

```
# Inspect the formal definition of the main report class.
methods::getClass("camReport")

# Check whether the class is registered.
methods::isClass("camReport")
```

gui,camReport-method *Launch the camtrapReport GUI*

Description

Launch 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	A logical value (default TRUE) indicating whether the application is opened in the default web browser.
max_upload_mb	A numeric value (default 2000) giving the maximum file-upload size in megabytes.
...	Additional arguments passed to shiny::runApp() .

Details

The function starts a Shiny-based graphical user interface for editing report metadata, selecting report sections, defining focus groups, and generating data-status or ecological reports.

The GUI can be used with an existing `camReport` object and provides an interactive alternative to configuring reports directly in R.

Value

Invisibly returns the supplied `camReport` object after launching the application.

See Also

[camData\(\)](#), [report\(\)](#), [status\(\)](#), [info\(\)](#)

Other report generation: [report\(\)](#)

Examples

```
source_dataset <- system.file(
  "external",
  "dataset",
  package = "camtrapReport"
)
example_dataset <- tempfile("camtrapReport-example-")
dir.create(example_dataset)
invisible(file.copy(
  list.files(source_dataset, full.names = TRUE),
  example_dataset,
  recursive = TRUE
))

cm <- camData(example_dataset)

gui(cm)

unlink(example_dataset, recursive = TRUE, force = TRUE)
```

info,camReport-method *Get or set information in a camReport object*

Description

Get selected information fields from a [camReport](#) object or update the value of a field used in report generation.

Usage

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

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

Arguments

x	A camReport object created by camData() .
name	A character vector naming fields to retrieve, or a single character string naming the field to update. When omitted from info() , the default report-information fields are returned.
value	The new value to assign to the specified field.

Details

A `camReport` object contains metadata and report information extracted from the camera-trap dataset, such as title, subtitle, authors, institute, site name, description, and acknowledgement text.

The `info` function can be used to inspect selected fields. The replacement form, `info(x, name) <- value`, can be used to update report information before generating the report.

Value

`info()` returns an object of class `camInfo` containing the requested fields. The replacement method returns the updated `camReport` object invisibly.

See Also

[camData\(\)](#), [report\(\)](#), [status\(\)](#), [updateReportSection\(\)](#)

Examples

```
source_dataset <- system.file(
  "external",
  "dataset",
  package = "camtrapReport"
)
example_dataset <- tempfile("camtrapReport-example-")
dir.create(example_dataset)
invisible(file.copy(
  list.files(source_dataset, full.names = TRUE),
  example_dataset,
  recursive = TRUE
))

cm <- camData(example_dataset)

# Retrieve all report information
info(cm)

# Retrieve selected fields
selected_info <- info(
  cm,
  name = c("title", "authors", "institute")
)

selected_info

# Update individual fields
info(cm, "title") <- "Camera-trap monitoring report"
info(cm, "institute") <- "Example Wildlife Research Institute"

# Inspect the updated fields
info(
  cm,
  name = c("title", "institute")
)
```

```
)

unlink(example_dataset, recursive = TRUE, force = TRUE)
```

install_all

Install packages required by camtrapReport

Description

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

Usage

```
install_all(pkgs = NULL, update = FALSE, github = TRUE,
            gitlab = FALSE, ...)

## S4 method for signature 'ANY'
install_all(pkgs = NULL, update = FALSE, github = TRUE, gitlab = FALSE, ...)

install_All(pkgs = NULL, update = FALSE, github = TRUE, gitlab = FALSE, ...)

## S4 method for signature 'ANY'
install_All(pkgs = NULL, update = FALSE, github = TRUE, gitlab = FALSE, ...)
```

Arguments

pkgs	An optional character vector of additional package names. The default is NULL.
update	A logical value. If TRUE, optional packages are force-reinstalled. The default is FALSE.
github	A logical value (default TRUE) specifying whether packages configured in GitHub repositories are included.
gitlab	A logical value (default FALSE) specifying whether packages configured in GitLab repositories are included.
...	Additional named arguments passed to pak::pkg_install() , such as lib, ask, dependencies, or upgrade.

Details

The function checks the package list used by camtrapReport and installs packages that are not currently available. Packages configured as GitHub or GitLab dependencies are installed from their corresponding repositories. All package installation is handled by [pak::pkg_install\(\)](#).

Unlike a direct call to pak, install_all() discovers the packages declared by the currently available YAML report modules, including modules added or modified by users, and combines them

with any additional packages supplied through `pkgs`. It then delegates resolution and installation to `pak`. The function is opt-in and is never called when the package is loaded or a report is rendered. `install_All()` is retained as a deprecated compatibility alias. New code should use `install_all()`. When `update = TRUE`, the requested optional packages are force-reinstalled from their configured sources using the `pak reinstall` parameter. Dependencies are upgraded only when required by the requested packages, unless `upgrade = TRUE` is supplied through `...`.

Value

Called for its side effects. The function returns `NULL` invisibly.

See Also

`camData()`, `report()`, `status()`

Examples

```
# Package installation changes the user's R library, so these examples are
# deliberately restricted to interactive sessions.
if (interactive()) {
  install_all()
  install_all(pkgs = "readr")
  install_all(update = TRUE)

  # Reinstall the requested packages and also update their dependencies.
  install_all(update = TRUE, upgrade = TRUE)
}
```

modules

Manage report modules

Description

Manage report modules in `camtrapReport`. These functions can be used to list existing modules, add new modules, move modules within the report structure, remove modules, restore deleted modules, and empty the module trash.

Usage

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

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

remove_Module(name, recursive, dir)

restore_Module(name, batch_id, test, dir)

empty_trash(name, id, dir)
```

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

Arguments

x	A character string giving the path to a module YAML file to add to the package.
before	An optional character string naming the module before which the new or moved module should be placed. The default is NULL.
after	An optional character string naming the module after which the new or moved module should be placed. The default is NULL.
test	A logical value specifying whether a module is tested before being added or restored. For <code>add_Module()</code> , the default is FALSE when object is not supplied and TRUE when a <code>camReport</code> object is supplied. For <code>restore_Module()</code> , the default is TRUE.
object	An optional <code>camReport</code> object used when testing a module that requires access to report data. The default is NULL.
formats	Output compatibility for the module: "html", "pdf", or "both". For <code>add_Module()</code> , an omitted value is read from an optional <code>formats</code> field in the YAML file and otherwise defaults to "both". For <code>move_Module()</code> , an omitted value retains the current setting.
dir	An optional path to a writable module registry containing <code>__modulesList.csv</code> . When omitted, the package's bundled report-module registry is used.
name	A character string naming the module to move, remove, restore, or permanently delete from the trash. For <code>empty_trash()</code> , the default is NULL.
parent	An optional character string naming the parent module. If omitted, the current parent is retained when moving a module.
level0	An optional character vector specifying the top-level module order. The default is <code>c("introduction", "methods", "results", "acknowledgements", "appendix")</code> .
recursive	A logical value (default TRUE) specifying whether child modules are removed together with the selected parent module.
batch_id	An optional character string identifying a deleted-module batch to restore. The default is NULL.
id	An optional character string identifying a deleted-module batch to remove permanently from the trash. The default is NULL.
tree	A logical value (default TRUE). If TRUE, modules are returned as a hierarchical tree.
brief	A logical value (default TRUE). If TRUE, a shortened module summary is returned when <code>tree = FALSE</code> .
include_trash	A logical value (default FALSE) specifying whether deleted modules in the trash are also listed.
validate	A logical value (default FALSE) specifying whether module YAML files are checked for validity when listing modules.

Details

camtrapReport uses a modular report architecture in which each report section is stored as a module. A module can contain section metadata, text, executable R code, required packages, and rendering settings.

Because modules can execute arbitrary R code with the permissions of the current R process, only install modules from trusted sources and inspect their contents before running them.

add_Module() adds a new YAML-defined module to the package module directory. The module can optionally be inserted before or after an existing module.

move_Module() changes the position or parent of an existing module within the hierarchical report structure.

remove_Module() moves a module to the module trash. If recursive = TRUE, child modules are also moved to the trash.

restore_Module() restores a previously removed module from the trash.

empty_trash() permanently removes modules from the trash.

list_Modules() lists active modules and can optionally include deleted modules stored in the trash.

Value

list_Modules() returns a data frame, a hierarchical module tree, or a list containing active and trashed modules. The other functions return information about the performed operation invisibly.

See Also

[reportSection\(\)](#), [updateReportSection\(\)](#), [testSection\(\)](#), [report\(\)](#)

Examples

```
# List modules as a hierarchical tree
module_tree <- list_Modules()
head(module_tree)

# List modules in table form
module_table <- list_Modules(tree = FALSE)
head(module_table)
```

report

Generate ecological and data-status reports

Description

Generate reports from a [camReport](#) object. report() creates an ecological report, while [status\(\)](#) creates a data-status report. A named profile can select an ordered set of modules immediately before rendering.

Usage

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

```
status(object, filename, view, profile, format)
```

Arguments

object	A camReport object created by camData() .
filename	An optional output filename or path. A .html or .pdf extension selects the format when format is omitted. Without an extension, the defaults are "report" and "data_status".
view	A logical value (default FALSE) specifying whether to open the generated report.
test	A logical value (default FALSE). If TRUE, ecological report modules are tested when HTML rendering fails.
profile	An optional profile name or reportProfile object. When omitted, the sections currently attached to object are rendered.
format	Output format, either "html" or "pdf". The default is inferred from filename, falling back to "html".

Details

HTML is the default output. PDF output first renders the existing HTML report and then prints it through a headless Chrome or Edge browser using the optional pagedown package. This route preserves modules that contain HTML widgets or HTML-formatted tables. Interactive controls are naturally static in the resulting PDF. PDF output omits the table of contents and excludes modules whose registry entry does not support PDF. The formats column shown by [list_Modules\(\)](#) records module compatibility.

Value

Invisibly returns the generated report path.

See Also

[reportProfile\(\)](#), [sections\(\)](#), [section_names\(\)](#), [camData\(\)](#)

Other report generation: [gui,camReport-method](#)

Examples

```
if (rmarkdown::pandoc_available()) {
  source_dataset <- system.file(
    "external", "dataset", package = "camtrapReport"
  )
  example_dataset <- tempfile("camtrapReport-example-")
  dir.create(example_dataset)
  invisible(file.copy(
    list.files(source_dataset, full.names = TRUE),
    example_dataset,
```

```

    recursive = TRUE
  ))
  cm <- camData(example_dataset)
  cm <- sections(cm, "introduction")
  report_file <- report(
    cm,
    filename = tempfile("camtrapReport-report-"),
    view = FALSE
  )
  file.exists(report_file)
  unlink(c(report_file, sub("\\.html$", ".Rmd", report_file)), force = TRUE)
  unlink(example_dataset, recursive = TRUE, force = TRUE)
}

```

reportProfile	<i>Define, import, and export report profiles</i>
---------------	---

Description

A report profile is an ordered selection of modules used to produce an ecological report and a data-status report. Modules normally come from the corresponding module pool. A module from the other pool can be selected with a qualified reference such as "status::spatial".

Usage

```

reportProfile(
  name,
  report = character(),
  status = character(),
  description = ""
)

read_profile(file)

write_profile(profile, file, overwrite = FALSE, object = NULL)

profile_names(object = NULL)

add_profile(object, profile, overwrite = FALSE)

```

Arguments

name	A non-empty profile name.
report	An ordered set of modules for <code>report()</code> .
status	An ordered set of modules for <code>status()</code> .
description	Optional descriptive text.

file	Path to a profile YAML file.
profile	A reportProfile object, a profile name registered in object, or a bundled profile name. write_profile() accepts either an object or a name.
overwrite	A logical value controlling whether an existing profile or file can be replaced.
object	A camReport object.

Details

Profile entries may be character vectors or data frames. Character entries use "source: :module" notation; unqualified entries default to the report pool for report and the status pool for status. A data frame can also contain a parent column to override the parent declared by a module. This makes it possible, for example, to place status modules below an ecological report's appendix without modifying the shared module YAML files.

Bundled profiles are registered in each object created by [camData\(\)](#). User profiles can be registered with add_profile() and shared as YAML files with read_profile() and write_profile().

Value

reportProfile() and read_profile() return a reportProfile object. profile_names() returns a character vector. add_profile() invisibly returns the modified camReport object, and write_profile() invisibly returns the output path.

See Also

[sections\(\)](#), [section_names\(\)](#), [report\(\)](#), [status\(\)](#)

Examples

```
profile <- reportProfile(
  name = "brief",
  report = c("introduction", "results"),
  status = c("abstract", "availability")
)
profile

profile_file <- tempfile(fileext = ".yaml")
write_profile(profile, profile_file)
identical(read_profile(profile_file), profile)
unlink(profile_file)
```

Description

Create a report-section object that stores text and optional R code for a section of a [camReport](#) report. Report sections can be added to a report to make report generation more flexible, modular, and extensible.

Usage

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

Arguments

name	A character string giving a unique name for the report section.
title	A character string giving the section title displayed in the report. The default is "".
parent	A character string naming the parent section, or NULL (default) for a top-level section.
txt	A character vector or list containing text to include in the report section. The default is NULL.
code_setting	Optional R Markdown chunk options for the code chunk. For example, {c(echo = FALSE, results = "asis")}. The default is NULL.
packages	An optional character vector giving packages required by the code chunk. The default is NULL.
code	Optional R code supplied inside braces, for example, { plot(1:10) }.

Details

Each report section must have a unique name. If a new section with the same name as an existing section is added to a report, the existing section is replaced.

A report section can contain text only, code only, or both text and code. When the code chunk needs access to the associated `camReport` object, the object can be referred to as `object` inside the code block.

This function is useful for creating custom report modules or extending the default `camtrapReport` workflow with project-specific text, tables, figures, or analyses.

Value

A `.textSection` object.

See Also

[camData\(\)](#), [report\(\)](#), [updateReportSection\(\)](#), [testSection\(\)](#)

Other report sections: [section_names\(\)](#), [testSection](#), [.textSection-method](#), [updateReportSection\(\)](#)

Examples

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

tx

# Create a section containing text and an R code chunk
tx2 <- reportSection(
  name = "methods",
  title = "Methods",
  parent = NULL,
  txt = "This section summarises the camera deployment design.",
  code_setting = {
    c(echo = FALSE, results = "asis", warning = FALSE)
  },
  code = {
    object$camera_setup |>
    summary()
  }
)

tx2
```

section_names

Select report sections

Description

Get module names or update which sections are used by an ecological or data-status report. Profiles provide named, reusable selections without moving or duplicating the YAML module files.

Usage

```
section_names(keep, exclude, object, profile, report_type)
```

```
sections(x, n, profile, report_type)
```

Arguments

keep	An optional character vector of section names to keep.
exclude	An optional character vector of section names to exclude.
object	An optional camReport object. It is needed to inspect user-defined profiles registered in that object.
profile	An optional profile name or reportProfile object.

report_type	Which output selection to use: "report", "status", or, for section_names(), "all".
x	A camReport object created by camData() .
n	An optional character vector of modules to include. Qualified names such as "status::spatial" can select from the other module pool when a profile is being edited.

Details

section_names() lists modules in a pool or in a profile. Set report_type = "all" to list both pools. Names from both pools, and names returned for a profile, use the unambiguous "source::module" notation.

sections() updates the transient module tree used for the next render. When profile is supplied, it can also inspect or update that registered profile. Set report_type = "status" to manage the data-status selection.

Value

section_names() and sections(x) return character vectors. sections(x, n) updates x and returns it invisibly.

See Also

[reportProfile\(\)](#), [report\(\)](#), [status\(\)](#), [listReportSections\(\)](#)

Other report sections: [reportSection](#), [character-method](#), [testSection](#), [.textSection-method](#), [updateReportSection\(\)](#)

Examples

```
section_names()
section_names(profile = "default")
section_names(report_type = "all")

source_dataset <- system.file(
  "external", "dataset", package = "camtrapReport"
)
example_dataset <- tempfile("camtrapReport-example-")
dir.create(example_dataset)
invisible(file.copy(
  list.files(source_dataset, full.names = TRUE),
  example_dataset,
  recursive = TRUE
))
cm <- camData(example_dataset)
sections(cm, profile = "default")
cm <- sections(cm, c("introduction", "methods"))
unlink(example_dataset, recursive = TRUE, force = TRUE)
```

testSection,.textSection-method

Test a report section

Description

Render a .textSection object as a temporary HTML report to check whether the section text and code can be rendered successfully.

Usage

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

Arguments

x	A .textSection object, usually created with reportSection() .
object	An optional camReport object created by camData() . The default is NULL.
view	A logical value (default TRUE) specifying whether the rendered temporary HTML file is opened after rendering.

Details

This function is mainly used to test custom report sections before they are added to a full [camReport](#) report. It checks whether the section text, R Markdown chunk settings, required packages, and R code can be rendered successfully.

The object argument is only needed when the R code in the section refers to a [camReport](#) object.

Value

Invisibly returns the path to the rendered temporary HTML file.

See Also

[camData\(\)](#), [reportSection\(\)](#), [updateReportSection\(\)](#), [report\(\)](#)

Other report sections: [reportSection](#), [character-method](#), [section_names\(\)](#), [updateReportSection\(\)](#)

Examples

```
tx <- reportSection(
  name = "introduction",
  title = "Introduction",
  parent = NULL,
  txt = "This is an introduction section.",
  code = {
    plot(1:10)
  }
)
```

```
# Render the section without opening a browser
test_file <- testSection(
  tx,
  view = FALSE
)

file.exists(test_file)

# Remove the generated HTML file
unlink(test_file)
```

updateReportSection	<i>Update report sections</i>
---------------------	-------------------------------

Description

Update the content of a report section in a [camReport](#) object, or list the report sections currently attached to the object.

Usage

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

listReportSections(x)
```

Arguments

x	A camReport object created by camData() .
section	A single character string giving the name or title of the report section to update.
text	An optional character value used to update the text body of the report section. The default is NULL, which leaves the text unchanged.
title	An optional character value giving a new title for the report section. The default is NULL, which leaves the title unchanged.

code	Optional R code used to update or add a code chunk. The code can be supplied inside braces.
code_name	An optional character string naming the R code chunk. The default is NULL.
code_setting	Optional R Markdown chunk options for the code chunk. The default is NULL.
packages	An optional character vector giving the R packages required by the code chunk. The default is NULL.
append_text	A logical value (default FALSE). If TRUE, the new text is appended to the existing section text; otherwise, the existing text is replaced.
append_code	A logical value (default FALSE). If TRUE, the new code is appended to the existing code chunk; otherwise, the existing code is replaced.

Details

Report sections can be identified using either their name or title. `listReportSections()` lists the sections attached to a `camReport` object and can be used to find the exact section names before updating them.

`updateReportSection()` is useful for adapting the default report content to project-specific needs, including changing section titles, replacing or appending text, and adding or updating R code chunks.

Value

`updateReportSection()` returns the updated `camReport` object invisibly. `listReportSections()` returns a data frame describing the report sections attached to the object.

See Also

[camData\(\)](#), [report\(\)](#), [reportSection\(\)](#), [info\(\)](#)

Other report sections: [reportSection](#), [character-method](#), [section_names\(\)](#), [testSection](#), [.textSection-method](#)

Examples

```
source_dataset <- system.file(
  "external",
  "dataset",
  package = "camtrapReport"
)
example_dataset <- tempfile("camtrapReport-example-")
dir.create(example_dataset)
invisible(file.copy(
  list.files(source_dataset, full.names = TRUE),
  example_dataset,
  recursive = TRUE
))

cm <- camData(example_dataset)

# List sections currently attached to the report
section_catalog <- listReportSections(cm)
```

```
head(section_catalog)

# Update the introduction
cm <- updateReportSection(
  cm,
  section = "introduction",
  title = "Project introduction",
  text = paste(
    "This report summarises camera-trap monitoring",
    "data for the example study site."
  )
)

# Confirm the updated section title
updated_catalog <- listReportSections(cm)
updated_catalog[
  updated_catalog$name == "introduction",
  c("name", "title"),
  drop = FALSE
]

unlink(example_dataset, recursive = TRUE, force = TRUE)
```

Index

- * **camera-trap**
 - camData, character-method, 2
- * **camtrapReport data**
 - camData, character-method, 2
- * **optional dependencies**
 - install_all, 8
- * **report generation**
 - gui, camReport-method, 5
 - report, 11
- * **report metadata**
 - info, camReport-method, 6
- * **report modules**
 - modules, 9
- * **report profiles**
 - reportProfile, 13
- * **report sections**
 - reportSection, character-method, 14
 - section_names, 16
 - testSection, .textSection-method, 18
 - updateReportSection, 19
- * **spatial**
 - camData, character-method, 2
- * **species**
 - camData, character-method, 2
 - .Rchunk-class (camReport-classes), 3
 - .textSection-class (camReport-classes), 3
- add_Module (modules), 9
- add_Module, character-method (modules), 9
- add_profile (reportProfile), 13
- camData (camData, character-method), 2
- camData(), 4–7, 9, 12, 14, 15, 17–20
- camData, character-method, 2
- camInfo-class (camReport-classes), 3
- camR (camReport-classes), 3
- camReport, 2, 3, 5, 6, 10–12, 14–19
- camReport (camReport-classes), 3
- camReport-class (camReport-classes), 3
- camReport-classes, 3
- characterORlist-class
 - (camReport-classes), 3
- characterORlistORnull-class
 - (camReport-classes), 3
- characterORnull-class
 - (camReport-classes), 3
- data.frameORnull-class
 - (camReport-classes), 3
- empty_trash (modules), 9
- empty_trash, ANY-method (modules), 9
- gui (gui, camReport-method), 5
- gui(), 3
- gui, camReport-method, 5
- info (info, camReport-method), 6
- info(), 3–5, 20
- info, camReport-method, 6
- info<- (info, camReport-method), 6
- info<- , camReport-method
 - (info, camReport-method), 6
- install_All (install_all), 8
- install_all, 8
- install_All, ANY-method (install_all), 8
- install_all, ANY-method (install_all), 8
- list_Modules (modules), 9
- list_Modules(), 12
- list_Modules, ANY-method (modules), 9
- listORnull-class (camReport-classes), 3
- listReportSections
 - (updateReportSection), 19
- listReportSections(), 17
- listReportSections, camReport-method
 - (updateReportSection), 19
- modules, 9

`move_Module(modules)`, 9
`move_Module,character-method(modules)`, 9

`pak::pkg_install()`, 8
`profile_names(reportProfile)`, 13

`read_profile(reportProfile)`, 13
`remove_Module(modules)`, 9
`remove_Module,character-method(modules)`, 9
`report`, 11
`report()`, 3–5, 7, 9, 11, 13–15, 17, 18, 20
`report,camReport-method(report)`, 11
`reportProfile`, 13
`reportProfile()`, 12, 17
`reportProfile-class`
 (`camReport-classes`), 3
`reportSection`
 (`reportSection,character-method`), 14
`reportSection()`, 4, 11, 18, 20
`reportSection,character-method`, 14
`restore_Module(modules)`, 9
`restore_Module,character-method(modules)`, 9

`section_names`, 16
`section_names()`, 12, 14, 15, 18, 20
`section_names,ANY-method`
 (`section_names`), 16
`sections(section_names)`, 16
`sections()`, 12, 14
`sections,camReport-method`
 (`section_names`), 16
`shiny::runApp()`, 5
`show,camInfo-method`
 (`camReport-classes`), 3
`show,camReport-method`
 (`camReport-classes`), 3
`status(report)`, 11
`status()`, 3–5, 7, 9, 11, 13, 14, 17
`status,camReport-method(report)`, 11

`testSection`
 (`testSection,.textSection-method`), 18
`testSection()`, 11, 15
`testSection,.textSection-method`, 18

`updateReportSection`, 19
`updateReportSection()`, 7, 11, 15, 17, 18
`updateReportSection,camReport-method`
 (`updateReportSection`), 19

`write_profile(reportProfile)`, 13