

Package ‘checkhelper’

January 22, 2024

Title Deal with Check Outputs

Version 0.1.1

Description Deal with packages 'check' outputs and reduce the risk of rejection by 'CRAN' by following policies.

License MIT + file LICENSE

URL <https://thinkr-open.github.io/checkhelper/>,
<https://github.com/ThinkR-open/checkhelper>

BugReports <https://github.com/ThinkR-open/checkhelper/issues>

Depends R (>= 2.10)

Imports cli, desc, devtools, dplyr, glue, lifecycle, magrittr,
pkgbuild, purrr, rcmdcheck, roxygen2, stringr, tools, utils,
whisker (>= 0.4), withr

Suggests attachment, callr, knitr, rmarkdown, testthat, usethis

VignetteBuilder knitr

Config/fusen/version 0.5.0.9009

Config/Needs/website ThinkR-open/thinkrtemplate

Encoding UTF-8

Language en-US

RoxygenNote 7.3.0

NeedsCompilation no

Author Sebastien Rochette [aut, cre] (<<https://orcid.org/0000-0002-1565-9313>>),
Vincent Guyader [aut] (<<https://orcid.org/0000-0003-0671-9270>>),
Arthur Bréant [aut] (<<https://orcid.org/0000-0003-1668-0963>>),
Murielle Delmotte [aut] (<<https://orcid.org/0000-0002-1339-2424>>),
ThinkR [cph]

Maintainer Sebastien Rochette <sebastien@thinkr.fr>

Repository CRAN

Date/Publication 2024-01-22 09:22:46 UTC

R topics documented:

check_as_cran	2
check_clean_userspace	3
create_example_pkg	4
find_missing_tags	5
get_data_info	6
get_notes	7
get_no_visible	8
print_globals	9
use_data_doc	10

Index	11
--------------	-----------

check_as_cran	<i>Check your package with real CRAN default environment variables and check strategy</i>
---------------	---

Description

[Experimental]

Usage

```
check_as_cran(
  pkg = ".",
  check_output = tempfile("check_output"),
  scratch = tempfile("scratch_dir"),
  Ncpus = 1,
  as_command = FALSE,
  clean_before = TRUE,
  open = FALSE
)
```

Arguments

pkg	pkg directory to check
check_output	Where to store check outputs. Default is a temporary directory
scratch	Where to store temporary files (cleaned after). Default is another temporary directory
Ncpus	Number of CPU used to build the package
as_command	Whether to run the check as Linux command line, instead of directly in R
clean_before	Whether to delete the previous check_output
open	Whether to open the check dir at the end of the process

Details

When you send your package on CRAN, there are multiple options set before running the checks. Here we use the CRAN settings and way of managing incoming packages used for Linux in this function `check_as_cran()`.

Scripts and options used are directly issued from the GitHub mirror repository of the CRAN machines: <https://github.com/r-devel/r-dev-web/tree/master/CRAN/>. Although `check_as_cran()` should run on any OS, it will run CRAN parameters originally set up for Linux machines.

In the `check_output`, you will get the same outputs, in the same format as used by CRAN, for the pre-test of incoming packages.

Value

An object containing errors, warnings, and notes.

References

<https://github.com/r-devel/r-dev-web/tree/master/CRAN/>

Examples

```
## Not run:  
# This runs a check of the current package  
# Directory to store the check outputs  
check_output <- tempfile("example")  
# Check the current package  
check_as_cran(check_output = check_output)  
# Open directory with all outputs  
utils::browseURL(check_output)  
  
## End(Not run)
```

`check_clean_userspace` *check_clean_userspace*

Description

[Experimental]

Usage

```
check_clean_userspace(pkg = ".", check_output = tempfile("dircheck"))
```

Arguments

<code>pkg</code>	Path to package to check
<code>check_output</code>	Path to directory where to store check results

Value

data.frame of files that are left after checks

Examples

```
## Not run:  
# This runs a check of the current package  
all_files <- check_clean_userspace()  
all_files  
  
## End(Not run)
```

create_example_pkg *Create a package example producing notes and errors*

Description

Create a package example producing notes and errors

Usage

```
create_example_pkg(  
  path = tempfile(pattern = "pkg-"),  
  with_functions = TRUE,  
  with_extra_notes = FALSE  
)
```

Arguments

path Path where to store the example package
with_functions Logical. Whether there will be functions or not (with notes)
with_extra_notes Logical. Whether there are extra notes or not

Value

Path where the example package is stored.

Examples

```
create_example_pkg()
```

find_missing_tags	<i>Find missing 'return' tag when function exported</i>
-------------------	---

Description

Find missing 'return' tag when function exported

Usage

```
find_missing_tags(  
  package.dir = ".",  
  roclets = NULL,  
  load_code = NULL,  
  clean = FALSE  
)
```

Arguments

package.dir	Location of package top level directory. Default is working directory.
roclets	Character vector of roclet names to use with package. The default, NULL, uses the roxygen roclets option, which defaults to <code>c("collate", "namespace", "rd")</code> .
load_code	A function used to load all the R code in the package directory. The default, NULL, uses the strategy defined by the <code>load</code> roxygen option, which defaults to <code>load_pkgload()</code> . See <code>load</code> for more details.
clean	If TRUE, roxygen will delete all files previously created by roxygen before running each roclet.

Value

a list with the 3 data.frames with the missing tags

Examples

```
## Not run:  
# What you will do from inside your package  
find_missing_tags()  
  
## End(Not run)  
# A reproducible example on a test package  
pkg_path <- create_example_pkg()  
find_missing_tags(pkg_path)
```

`get_data_info`*Get information of a .Rda file stored inside the 'data' folder*

Description

Get information of a .Rda file stored inside the 'data' folder

Usage

```
get_data_info(name, description, source)
```

Arguments

<code>name</code>	name of the file that exists in "data/"
<code>description</code>	description for the data
<code>source</code>	source of data

Value

list of information from a data.frame

See Also

[use_data_doc\(\)](#) to add the information directly as roxygen documentation in your package.

Examples

```
# Store a dataset as rda file
path_project <- tempfile(pattern = "data-")
path_data <- file.path(path_project, "data")
dir.create(path_data, recursive = TRUE)
path_rda <- file.path(path_data, "iris.rda")
save(iris, file = path_rda)

# Get its information
withr::with_dir(
  path_project,
  {
    get_data_info("iris", "Iris data frame", source = "ThinkR")
  }
)

# Clean userspace
unlink(path_project, recursive = TRUE)
```

get_notes	<i>List notes from check and identify global variables</i>
-----------	--

Description

List notes from check and identify global variables

Usage

```
get_notes(path = ".", checks, ...)
```

Arguments

path	Path to a package tarball or a directory.
checks	Output of rcmdcheck if already computed
...	Other parameters for rcmdcheck

Value

A tibble with notes and information about the global variables

Examples

```
## Not run:
# This runs a check of the example package
tempdir <- tempdir()
# Create fake package
usethis::create_package(file.path(tempdir, "checkpackage"), open = FALSE)

# Create function no visible global variables and missing documented functions
cat("
#' Function
#' @importFrom dplyr filter
#' @export
my_fun <- function() {
  data %>%
  filter(col == 3) %>%
  mutate(new_col = 1) %>%
  ggplot() +
    aes(x, y, colour = new_col) +
    geom_point()
}
", file = file.path(tempdir, "checkpackage", "R", "function.R"))

path <- file.path(tempdir, "checkpackage")
attachment::att_to_description(path = path)
get_notes(path)

## End(Not run)
```

get_no_visible	<i>List no visible globals from check and separate by category</i>
----------------	--

Description

List no visible globals from check and separate by category

Usage

```
get_no_visible(path = ".", checks, ...)
```

Arguments

path	Path to a package tarball or a directory.
checks	Output of <code>rcmdcheck</code> if already computed
...	Other parameters for <code>rcmdcheck</code>

Value

A list with no visible globals

Examples

```
## Not run:
# This runs a check of the example package
tempdir <- tempdir()
# Create fake package
usethis::create_package(file.path(tempdir, "checkpackage"), open = FALSE)

# Create function no visible global variables and missing documented functions
cat("
#' Function
#' @importFrom dplyr filter
#' @export
my_fun <- function() {
  data %>%
  filter(col == 3) %>%
  mutate(new_col = 1) %>%
  ggplot() +
    aes(x, y, colour = new_col) +
    geom_point()
}
", file = file.path(tempdir, "checkpackage", "R", "function.R"))

path <- file.path(tempdir, "checkpackage")
attachment::att_to_description(path = path)
get_no_visible(path)

## End(Not run)
```

print_globals	<i>Print no visible globals from check and separate by category</i>
---------------	---

Description

Print no visible globals from check and separate by category

Usage

```
print_globals(globals, path = ".", ..., message = TRUE)
```

Arguments

globals	A list as issued from get_no_visible or empty
path	Path to a package tarball or a directory.
...	Other parameters for rcmdcheck
message	Logical. Whether to return message with content (Default) or return as list

Value

A message with no visible globals or a list with no visible globals

Examples

```
## Not run:
# This runs a check of the example package
tempdir <- tempdir()
# Create fake package
usethis::create_package(file.path(tempdir, "checkpackage"), open = FALSE)

# Create function no visible global variables and missing documented functions
cat("
#' Function
#' @importFrom dplyr filter
#' @export
my_fun <- function() {
  data %>%
  ggplot2::ggplot() +
    aes(x, y, colour = new_col) +
    geom_point()
}
", file = file.path(tempdir, "checkpackage", "R", "function.R"))

path <- file.path(tempdir, "checkpackage")
attachment::att_to_description(path = path)
globals <- get_no_visible(path)
print_globals(globals = globals)

## End(Not run)
```

`use_data_doc`*Create documentation of a rda / RData dataset in a package*

Description

Create documentation of a rda / RData dataset in a package

Usage

```
use_data_doc(  
  name,  
  prefix = "doc_",  
  description = "Description",  
  source = "Source"  
)
```

Arguments

<code>name</code>	Name of your data without extension
<code>prefix</code>	Add prefix for the name of the output R script
<code>description</code>	Description of the dataset that will be added in the documentation
<code>source</code>	Source of the dataset that will be presented in the documentation

Value

Creates a data documentation in a R file and returns path to the file

See Also

[get_data_info\(\)](#) to only retrieve information without writing the documentation

Examples

```
## Not run:  
# This adds a R file in the current user directory  
# This works if there is a "my_data.rda" file in your "data/" directory  
use_data_doc("my_data", description = "Description of my_data", source = "Here the source")  
  
## End(Not run)
```

Index

`check_as_cran`, 2
`check_clean_userspace`, 3
`create_example_pkg`, 4

`find_missing_tags`, 5

`get_data_info`, 6
`get_data_info()`, 10
`get_no_visible`, 8, 9
`get_notes`, 7

`load`, 5
`load_pkgload()`, 5

`print_globals`, 9

`rcmdcheck`, 7–9

`use_data_doc`, 10
`use_data_doc()`, 6