Package 'cleanr'

March 21, 2019

Type Package

Title Helps You to Code Cleaner

Version 1.2.0

Description Check your R code for some of the most common
layout flaws. Many tried to teach us how to write code less dreadful,
be it implicitly as B. W. Kernighan and D. M. Ritchie (1988)
<ISBN:0-13-110362-8> in 'The C Programming Language' did, be it
explicitly as R.C. Martin (2008) <ISBN:0-13-235088-2> in 'Clean Code:
A Handbook of Agile Software Craftsmanship' did. So we should check
our code for files too long or wide, functions with too many lines,
too wide lines, too many arguments or too many levels of nesting.
Note: This is not a static code analyzer like pylint or the like.
Checkout <https://cran.r-project.org/package=lintr> instead.

License BSD_2_clause + file LICENSE

URL https://gitlab.com/fvafrCU/cleanr

Depends R (>= 3.3.0)

Imports checkmate,
rprojroot,
pkgload

Suggests RUnit,
knitr,
rmarkdown,
testthat,
devtools,
usethis

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Collate 'utils.R'
'internals.R'
'checks.R'

'cleanr-package.R'
'options.R'
'throw.R'
'wrappers.R'
'zzz.R'

R topics documented:

`cleanr-package` *Helps You to Code Cleaner*

Description

Check your R code for some of the most common layout flaws.

Details

Many tried to teach us how to write code less dreadful, be it implicitly as B. W. Kernighan and D. M. Ritchie in *The C Programming Language* did, be it explicitly as R.C. Martin in *Clean Code: A Handbook of Agile Software Craftsmanship* did.

So we should check our code for files too long or wide, functions with too many lines, too wide lines, too many arguments or too many levels of nesting.

Note

This is not a static code analyzer like pylint or the like. If you're looking for a static code analyzer, check out lintr (`https://cran.r-project.org/package=lintr` or `https://github.com/jimhester/lintr`).

See Also

Packages **codetools** (https://cran.r-project.org/package=codetools), **formatR** (https://cran.r-project.org/package=formatR) and **lintr** (https://cran.r-project.org/package=lintr).

`check_directory` *Check a Directory*

Description

Run `check_file` on files in a directory.

Usage

```
check_directory(path, pattern = "\\.[rR]$", recursive = FALSE, ...)
```

Arguments

`path`	Path to the directory to be checked.
`pattern`	A pattern to search files with, see `list.files`.
`recursive`	Search the directory recursively? See `list.files`.
`...`	Arguments to be passed to `check_file`.

Details

The function catches the messages of "cleanr"-conditions `thrown` by `check_file` and, if it caught any, `throws` them.

Value

`Invisibly TRUE`, but see *Details*.

See Also

`check_package`.

Examples

```
# load internal functions first.
load_internal_functions("cleanr")
print(cleanr::check_directory(system.file("source", "R", package = "cleanr"),
                              max_num_arguments = 8, max_file_width = 90,
                              max_file_length = 350,
                              check_return = FALSE))
```

check_file *Check a File*

Description

Run check_functions_in_file and check_file_layout on a file.

Usage

```
check_file(path, ...)
```

Arguments

path	Path to the file to be checked.
...	Arguments to be passed to check_functions_in_file or check_file_layout.

Details

The function catches the messages of "cleanr"-conditions thrown by check_functions_in_file and check_file_layout and, if it caught any, throws them.

Value

invisible(TRUE), but see *Details*.

Examples

```
print(cleanr::check_file(system.file("source", "R", "utils.R",
                                     package = "cleanr")))
```

check_file_layout *Check a File's Layout*

Description

Run all file_checks on a file.

Usage

```
check_file_layout(path,
  max_file_length = get_cleanr_options("max_file_length"),
  max_file_width = get_cleanr_options("max_file_width"))
```

Arguments

path	Path to the file to be checked.
max_file_length	See check_file_length.
max_file_width	See check_file_width.

Details

The function catches the messages of "cleanr"-conditions thrown by file_checks and, if it caught any, throws them.

Value

invisible(TRUE), but see *Details*.

Examples

```
print(cleanr::check_file_layout(system.file("source", "R", "checks.R",
                                            package = "cleanr")))
```

---

check_functions_in_file

Check All Functions Defined in a File

Description

Run check_function_layout on all functions defined in a file.

Usage

```
check_functions_in_file(path, ...)
```

Arguments

path	Path to the file to be checked.
...	Arguments to be passed to check_function_layout.

Details

The functions catches the messages of "cleanr"-conditions thrown by check_function_layout and, if it caught any, throws them.

Value

invisible(TRUE), but see *Details*.

Examples

```
print(cleanr:::check_functions_in_file(system.file("source", "R", "utils.R",
                                                   package = "cleanr")))
```

---

check_function_layout *Check a Function's Layout*

Description

Run all function_checks on a function.

Usage

```
check_function_layout(object, function_name = NULL,
  max_lines_of_code = get_cleanr_options("max_lines_of_code"),
  max_lines = get_cleanr_options("max_lines"),
  max_num_arguments = get_cleanr_options("max_num_arguments"),
  max_nesting_depth = get_cleanr_options("max_nesting_depth"),
  max_line_width = get_cleanr_options("max_line_width"),
  check_return = get_cleanr_options("check_return"))
```

Arguments

object	The function to be checked.
function_name	The name to be used for reporting. Stick with the default: If NULL, it is taken from the object given. Argument is used internally to pass function names retrieved via get in the wrapper function check_functions_in_file.
max_lines_of_code	See check_num_lines_of_code.
max_lines	See check_num_lines.
max_num_arguments	See check_num_arguments.
max_nesting_depth	See check_nesting_depth.
max_line_width	See check_line_width.
check_return	See check_return.

Details

The functions catches the messages of "cleanr"-conditions thrown by function_checks and, if it caught any, throws them.

Value

invisible(TRUE), but see *Details*.

Examples

```
print(cleanr::check_function_layout(cleanr::check_num_lines))
```

---

check_package *Check a Package*

Description

Run `check_file` on a package's source.

Usage

```
check_package(path, pattern = "\\.[rR]$", ...)
```

Arguments

`path`	Path to the package to be checked.
`pattern`	A pattern to search files with, see `list.files`.
`...`	Arguments to be passed to `check_file`.

Details

The function catches the messages of "cleanr"-conditions `thrown` by `check_file` and, if it caught any, `throws` them.

Value

`Invisibly TRUE`, but see *Details*.

Examples

```
# create a fake package first:
package_path <- file.path(tempdir(), "fake")
usethis::create_package(package_path, fields = NULL,
                        rstudio = FALSE, open = FALSE)
directory <- system.file("runit_tests", "source", "R_s4",
                         package = "cleanr")
file.copy(list.files(directory, full.names = TRUE), file.path(package_path,
                                                              "R"))
RUnit::checkTrue(cleanr::check_package(package_path, check_return = FALSE))
```

file_checks *File Checks*

Description

A set of tiny functions to check that files adhere to a layout style. A file should have a clear layout, it should

- not have too many lines and
- not have lines too wide.

Usage

```
check_file_width(path, max_file_width = gco("max_file_width"))

check_file_length(path, max_file_length = gco("max_file_length"))
```

Arguments

path	Path to the file to be checked.
max_file_width	The maximum line width accepted. Set (preferably via set_cleanr_options) to NULL or FALSE to disable the check.
max_file_length	The maximum number of lines accepted. Set (preferably via set_cleanr_options) to NULL or FALSE to disable the check.

Details

In case of a fail all file_checks throw a condition of class c("cleanr", "error", "condition").

Value

Invisibly TRUE, but see *Details*.

Examples

```
print(cleanr::check_file_width(system.file("source", "R", "checks.R",
                                           package = "cleanr")))
print(cleanr::check_file_length(system.file("source", "R", "checks.R",
                                            package = "cleanr"),
                                            max_file_length = 300))
```

function_checks *Function Checks*

Description

A set of tiny functions to check that functions adhere to a layout style. A function should have a clear layout, it should

- not have too many arguments,
- not have too many levels of nesting,
- neither have too many lines nor
- have too many lines of code,
- not have lines too wide and
- explicitly `return` an object.

Usage

```
check_num_arguments(object, max_num_arguments = gco("max_num_arguments"))

check_nesting_depth(object, max_nesting_depth = gco("max_nesting_depth"))

check_num_lines(object, max_lines = gco("max_lines"))

check_num_lines_of_code(object,
  max_lines_of_code = gco("max_lines_of_code"))

check_line_width(object, max_line_width = gco("max_line_width"))

check_return(object, check_return = gco("check_return"))
```

Arguments

`object`	The function to be checked. Should have been sourced with keep.source = TRUE (see `get_function_body`).
`max_num_arguments`	The maximum number of arguments accepted. Set (preferably via `set_cleanr_options`) to `NULL` or `FALSE` to disable the check.
`max_nesting_depth`	The maximum nesting depth accepted. Set (preferably via `set_cleanr_options`) to `NULL` or `FALSE` to disable the check.
`max_lines`	The maximum number of lines accepted. Set (preferably via `set_cleanr_options`) to `NULL` or `FALSE` to disable the check.
`max_lines_of_code`	The maximum number of lines of code accepted. Set (preferably via `set_cleanr_options`) to `NULL` or `FALSE` to disable the check.

max_line_width	The maximum line width accepted. Set (preferably via set_cleanr_options) to NULL or FALSE to disable the check.
check_return	Set (preferably via set_cleanr_options) to NULL or FALSE to disable the check.

Details

In case of a fail all function_checks throw a condition of class c("cleanr", "error", "condition").

Value

Invisibly TRUE, but see *Details*.

Warning

check_return just greps for a for a line starting with a return statement (ah, see the code for the real thing). This does not ensure that *all* return paths from the function are explicit and it may miss a return path after a semicolon. It just checks if you use return at all.

Examples

```
print(cleanr::check_num_arguments(cleanr::check_num_arguments))
print(cleanr::check_nesting_depth(cleanr::check_nesting_depth))
print(cleanr::check_num_lines(cleanr::check_num_lines))
print(cleanr::check_num_lines_of_code(cleanr::check_num_lines_of_code))
print(cleanr::check_return(cleanr::check_return))
# R reformats functions on import (see
# help(get_function_body, package = "cleanr")), so we need 90 characters:
print(cleanr::check_line_width(cleanr::check_line_width,
                               max_line_width = 90))
```

---

get_cleanr_options *Get options for* **cleanr**

Description

A convenience function for getOption.

Usage

```
get_cleanr_options(..., remove_names = FALSE, flatten_list = TRUE)

gco(..., remove_names = FALSE, flatten_list = TRUE)
```

Arguments

...	See getOption.
remove_names	[boolean(1)] Remove the names?
flatten_list	[boolean(1)] Return a vector?

Value

a (possibly named) list or a vector.

Note

`gco` is just an alias for `get_cleanr_options`.

Examples

```
cleanr::get_cleanr_options("max_lines")
cleanr::get_cleanr_options("max_lines", remove_names = TRUE)
cleanr::get_cleanr_options("max_lines", flatten_list = TRUE)
cleanr::get_cleanr_options("max_lines", flatten_list = TRUE, remove_names = TRUE)
cleanr::get_cleanr_options(flatten_list = TRUE, remove_names = TRUE)
cleanr::get_cleanr_options(c("max_lines", "max_lines_of_code"))
```

`is_not_false` *Test if an Object is not False*

Description

Sometimes you need to know whether or not an object exists and is not set to FALSE (and possibly not NULL).

Usage

```
is_not_false(object, null_is_false = TRUE, ...)
```

Arguments

`object`	The object to be tested.
`null_is_false`	[boolean(1)] Should NULL be treated as FALSE?
`...`	Parameters passed to `exists`. See Details and Examples.

Details

Pass an environment if you call the function elsewhere than from `.GlobalEnv`.

Value

TRUE if the object is set to something different than FALSE, FALSE otherwise.

Examples

```
a  <- 1
is_not_false(a)
f <- function() {
    a <- NULL
    should_be_true <- ! is_not_false(a, null_is_false = TRUE,
                                     where = environment())
    return(should_be_true)
}
print(f())
```

`load_internal_functions` *Load a Package's Internals*

Description

Load objects not exported from a package's namespace.

Usage

```
load_internal_functions(package, ...)
```

Arguments

package	[character(1)] The name of the package as a string.
...	Arguments passed to `ls`, all.names = TRUE could be a good idea.

Details

The files to be checked get sourced, which means they have to contain R code producing no errors. If we want to check the source code of a package, we need to load the package *and* be able to run all its internals in our environment.

Value

invisible(TRUE)

See Also

`checkUsageEnv in codetools`.

Examples

```
load_internal_functions("cleanr")
```

set_cleanr_options *Set Options for* **cleanr**

Description

A convenience function for options.

Usage

```
set_cleanr_options(..., reset = FALSE, overwrite = TRUE)
```

Arguments

...	See options. See function_checks and file_checks for options to be set with **cleanr**. Use get_cleanr_options to get the current values. **cleanr**'s standard defaults are: max_file_width=80 max_file_length=300 max_lines=65 max_lines_of_code=50 max_num_arguments=5 max_nesting_depth=3 max_line_width=80 check_return=TRUE
reset	[boolean(1)] Reset all **cleanr** options to the package's defaults?
overwrite	[boolean(1)] Overwrite options already set? Is set to FALSE on package loading to ensure your previously set **cleanr** options won't get overridden. Just ignore that argument.

Details

cleanr loads a couple of options as defaults for its functions. The defaults are stored in a list element of options. All checks (see function_checks and file_checks) can be disabled by setting the corresponding option list item to NULL or FALSE.

Value

Invisibly TRUE.

Examples

```
# R.C. Martin's Clean Code recommends monadic argument lists.
cleanr::set_cleanr_options(max_num_arguments = 1)
# R.C. Martin's Clean Code recommends functions less than 20 lines long.
cleanr::set_cleanr_options(max_lines = 30, max_lines_of_code = 20)
# same as above:
```

```
cleanr::set_cleanr_options(list(max_lines = 30, max_lines_of_code = 20))
cleanr::get_cleanr_options(flatten_list = TRUE)
# we delete all options and set some anew
options("cleanr" = NULL)
options("cleanr" = list(max_lines = 30, max_lines_of_code = 20))
# fill the missing options with the package's defaults:
cleanr::set_cleanr_options(overwrite = FALSE)
cleanr::get_cleanr_options(flatten_list = TRUE)
# reset to the package's defaults:
cleanr::set_cleanr_options(reset = TRUE)
cleanr::get_cleanr_options(flatten_list = TRUE)
```

Index