

Package ‘formatR’

June 11, 2019

Type Package

Title Format R Code Automatically

Version 1.7

Description Provides a function `tidy_source()` to format R source code. Spaces and indent will be added to the code automatically, and comments will be preserved under certain conditions, so that R code will be more human-readable and tidy. There is also a Shiny app as a user interface in this package (see `tidy_app()`).

Depends R ($\geq 3.0.2$)

Suggests codetools, shiny, testit, rmarkdown, knitr

License GPL

URL <https://github.com/yihui/formatR>

BugReports <https://github.com/yihui/formatR/issues>

VignetteBuilder knitr

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation no

Author Yihui Xie [aut, cre] (<<https://orcid.org/0000-0003-0645-5666>>),
Eugene Ha [ctb],
Kohske Takahashi [ctb],
Ed Lee [ctb]

Maintainer Yihui Xie <xie@yihui.name>

Repository CRAN

Date/Publication 2019-06-11 20:00:03 UTC

R topics documented:

<code>tidy_app</code>	2
<code>tidy_dir</code>	2
<code>tidy_eval</code>	3
<code>tidy_source</code>	4
<code>usage</code>	6

Index**8**

tidy_app	<i>A Shiny app to format R code</i>
----------	-------------------------------------

Description

This function calls `tidy_source()` to format R code in a Shiny app. The arguments of `tidy_source()` are presented in the app as input widgets such as checkboxes.

Usage

```
tidy_app()
```

Examples

```
if (interactive()) formatR::tidy_app()
```

tidy_dir	<i>Format all R scripts under a directory, or specified R scripts</i>
----------	---

Description

`tidy_dir()` first looks for all the R scripts under a directory (using the pattern "[.]?[RrSsQq]\$", then uses `tidy_source` to tidy these scripts. The original scripts will be overwritten with reformatted code if reformatting was successful. You may need to back up the original directory first if you do not fully understand the tricks used by `tidy_source`. `tidy_file()` formats specified R scripts.

Usage

```
tidy_dir(path = ".", recursive = FALSE, ...)
```

```
tidy_file(file, ...)
```

Arguments

path	the directory
recursive	whether to recursively look for R scripts under path
...	other arguments to be passed to <code>tidy_source</code>
file	a vector of filenames

Value

Invisible NULL.

Author(s)

Yihui Xie ([tidy_dir](#)) and Ed Lee ([tidy_file](#))

See Also

[tidy_source](#)

Examples

```
library(formatR)

path = tempdir()
file.copy(system.file("demo", package = "base"), path, recursive = TRUE)
tidy_dir(path, recursive = TRUE)
```

tidy_eval

Evaluate R code and mask the output by a prefix

Description

This function is designed to insert the output of each chunk of R code into the source code without really breaking the source code, since the output is masked in comments.

Usage

```
tidy_eval(source = "clipboard", ..., file = "", prefix = "## ", envir = parent.frame())
```

Arguments

source	the input filename (by default the clipboard; see tidy_source)
...	other arguments passed to tidy_source
file	the file to write by cat ; by default the output is printed on screen
prefix	the prefix to mask the output
envir	the environment in which to evaluate the code (by default the parent environment; if we do not want to mess up with the parent environment, we can set <code>envir = NULL</code> or <code>envir = new.env()</code>)

Value

Evaluated R code with corresponding output (printed on screen or written in a file).

References

<https://yihui.name/formatR>

Examples

```
library(formatR)
## evaluate simple code as a character vector
tidy_eval(text = c("a<-1+1;a", "matrix(rnorm(10),5)"))

## evaluate a file
tidy_eval(system.file("format", "messy.R", package = "formatR"))
```

tidy_source

Reformat R code while preserving blank lines and comments

Description

This function returns reformatted source code; it tries to preserve blank lines and comments, which is different with [parse](#) and [deparse](#). It can also replace = with <- where = means assignments, and reindent code by a specified number of spaces (default is 4).

Usage

```
tidy_source(source = "clipboard", comment = getOption("formatR.comment", TRUE),
  blank = getOption("formatR.blank", TRUE), arrow = getOption("formatR.arrow",
    FALSE), brace.newline = getOption("formatR.brace.newline", FALSE),
  indent = getOption("formatR.indent", 4), wrap = getOption("formatR.wrap",
    TRUE), output = TRUE, text = NULL, width.cutoff = getOption("width"),
  ...)
```

Arguments

source	a character string: location of the source code (default to be the clipboard; this means we can copy the code to clipboard and use tidy_source() without specifying the argument source)
comment	whether to keep comments (TRUE by default)
blank	whether to keep blank lines (TRUE by default)
arrow	whether to replace the assign operator = with <-
brace.newline	whether to put the left brace { to a new line (default FALSE)
indent	number of spaces to indent the code (default 4)
wrap	whether to wrap comments to the linewidth determined by width.cutoff (note that roxygen comments will never be wrapped)
output	output to the console or a file using cat ?
text	an alternative way to specify the input: if it is NULL, the function will read the source code from the source argument; alternatively, if text is a character vector containing the source code, it will be used as the input and the source argument will be ignored
width.cutoff	passed to deparse : integer in [20, 500] determining the cutoff at which line-breaking is tried (default to be getOption("width"))
...	other arguments passed to cat , e.g. file (this can be useful for batch-processing R scripts, e.g. tidy_source(source = 'input.R', file = 'output.R'))

Value

A list with components

text.tidy	the reformatted code as a character vector
text.mask	the code containing comments, which are masked in assignments or with the weird operator

Note

Be sure to read the reference to know other limitations.

Author(s)

Yihui Xie <<https://yihui.name>> with substantial contribution from Yixuan Qiu <<http://yixuan.cos.name>>

References

<https://yihui.name/formatR> (an introduction to this package, with examples and further notes)

See Also

[parse](#), [deparse](#)

Examples

```
library(formatR)

## a messy R script
messy = system.file("format", "messy.R", package = "formatR")
tidy_source(messy)

## use the 'text' argument
src = readLines(messy)

## source code
cat(src, sep = "\n")

## the formatted version
tidy_source(text = src)

## preserve blank lines
tidy_source(text = src, blank = TRUE)

## indent with 2 spaces
tidy_source(text = src, indent = 2)

## discard comments!
tidy_source(text = src, comment = FALSE)

## wanna see the gory truth??
```

```

tidy_source(text = src, output = FALSE)$text.mask

## tidy up the source code of image demo
x = file.path(system.file(package = "graphics"), "demo", "image.R")

# to console
tidy_source(x)

# to a file
f = tempfile()
tidy_source(x, blank = TRUE, file = f)

## check the original code here and see the difference
file.show(x)
file.show(f)

## use global options
options(comment = TRUE, blank = FALSE)
tidy_source(x)

## if you've copied R code into the clipboard
if (interactive()) {
  tidy_source("clipboard")
  ## write into clipboard again
  tidy_source("clipboard", file = "clipboard")
}

## the if-else structure
tidy_source(text = c("{if(TRUE)1 else 2; if(FALSE){1+1", "## comments", "} else 2}"))

```

usage

Show the usage of a function

Description

Print the reformatted usage of a function. The arguments of the function are searched by [argsAnywhere](#), so the function can be either exported or non-exported in a package. S3 methods will be marked.

Usage

```
usage(FUN, width = getOption("width"), tidy = TRUE, output = TRUE,
      indent.by.FUN = FALSE, fail = c("warn", "stop", "none"))
```

Arguments

FUN	the function name
width	the width of output
tidy	whether to reformat the usage code

<code>output</code>	whether to write the output to the console (via <code>cat</code>)
<code>indent.by.FUN</code>	whether to indent subsequent lines by the width of the function name (see “Details”)
<code>fail</code>	a character string that determines whether to generate a warning, stop, or do neither, if the width constraint is unfulfillable (default is “warn”)

Details

Line breaks in the output occur between arguments. In particular, default values of arguments will not be split across lines.

When `indent.by.FUN` is `FALSE`, indentation is set by the option `getOption("formatR.indent", 4L)`, the default value of the `indent` argument of `tidy_source`.

Value

The R code for the usage is returned as a character string (invisibly).

See Also

[tidy_source](#)

Examples

```
library(formatR)
usage(var)

usage(plot)

usage(plot.default) # default method
usage("plot.lm") # on the 'lm' class

usage(usage)

usage(barplot.default, width = 60) # output lines have 60 characters or less

# indent by width of 'barplot('
usage(barplot.default, width = 60, indent.by.FUN = TRUE)

## Not run:
# a warning is raised because the width constraint is unfulfillable
usage(barplot.default, width = 30)

## End(Not run)
```

Index

argsAnywhere, [6](#)

cat, [3](#), [4](#), [7](#)

deparse, [4](#), [5](#)

getOption, [7](#)

parse, [4](#), [5](#)

tidy_app, [2](#)

tidy_dir, [2](#)

tidy_eval, [3](#)

tidy_file(tidy_dir), [2](#)

tidy_source, [2](#), [3](#), [4](#), [7](#)

usage, [6](#)