

Package: formatR (via r-universe)

August 23, 2024

Type Package
Title Format R Code Automatically
Version 1.14.1
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 (>= 3.2.3)
Suggests rstudioapi, shiny, testit, rmarkdown, knitr
License GPL
URL <https://github.com/yihui/formatR>
BugReports <https://github.com/yihui/formatR/issues>
VignetteBuilder knitr
RoxygenNote 7.2.3
Encoding UTF-8
Repository <https://yihui.r-universe.dev>
RemoteUrl <https://github.com/yihui/formatR>
RemoteRef HEAD
RemoteSha 819562627bd9931b4e98fd762ebe5688f1b47da8

Contents

tidy_app	2
tidy_dir	2
tidy_eval	3
tidy_pipe	4
tidy_rstudio	4
tidy_source	5
usage	8
Index	10

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

Description

Run a Shiny app that formats R code via `tidy_source()`. This app uses input widgets, such as checkboxes, to pass arguments to `tidy_source()`.

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

Look for all R scripts under a directory (using the pattern "[.][RrSsQq]\$", then tidy them with `tidy_source()`. If successful, the original scripts will be overwritten with reformatted ones. Please back up the original directory first if you do not fully understand the tricks used by `tidy_source()`. `tidy_file()` formats scripts specified by file names.

Usage

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

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

Arguments

path	The path to a directory containing R scripts.
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`*Insert output to source code*

Description

Evaluate R code by chunks, then insert the output to each chunk. As the output is masked in comments, the source code will not break.

Usage

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

Arguments

<code>source</code>	The input file name (by default the clipboard; see tidy_source()).
<code>...</code>	Other arguments passed to tidy_source() .
<code>file</code>	The file name to write to via cat() .
<code>prefix</code>	The prefix to mask the output.
<code>envir</code>	The environment in which to evaluate the code. By default the parent frame; set <code>envir = NULL</code> or <code>envir = new.env()</code> to avoid the possibility of contaminating the parent frame.

Value

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

References

<https://yihui.org/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_pipe

*Substitute the **magrittr** pipe with R's native pipe operator*

Description

Parse the R code in the RStudio editor, identify %>%, and substitute with |>.

Usage

```
tidy_pipe()
```

Note

Currently this function only works inside the RStudio IDE, and may be extended in future to deal with arbitrary R code elsewhere.

Examples

```
formatR::tidy_pipe()
```

tidy_rstudio

Reformat R code in RStudio IDE

Description

If any R code is selected in the RStudio source editor, this function reformats the selected code; otherwise it reformats the current open file (if it is unsaved, it will be automatically saved).

Usage

```
tidy_rstudio(...)
```

Arguments

... Arguments to be passed to [tidy_source\(\)](#), among which the indent argument will respect the value you set for the number of spaces for indentation in RStudio.

Note

If the output is not what you want, you can undo the change in the editor (Ctrl + Z or Command + Z).

Examples

```
formatR::tidy_rstudio()
formatR::tidy_rstudio(args.newline = TRUE)
```

tidy_source

*Reformat R code***Description**

Read R code from a file or the clipboard and reformat it. This function is based on [parse\(\)](#) and [deparse\(\)](#), but it does several other things, such as preserving blank lines and comments, substituting the assignment operator = with <-, and re-indenting code with a specified number of spaces.

Usage

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

Arguments

source	A character string: file path to the source code (defaults to the clipboard).
comment	Whether to keep comments.
blank	Whether to keep blank lines.
arrow	Whether to substitute the assignment operator = with <-.
pipe	Whether to substitute the magrittr pipe %>% with R's native pipe operator >.
brace.newline	Whether to put the left brace { to a new line.

<code>indent</code>	Number of spaces to indent the code.
<code>wrap</code>	Whether to wrap comments to the linewidth determined by <code>width.cutoff</code> (roxygen comments will never be wrapped).
<code>width.cutoff</code>	An integer in <code>[20, 500]</code> : if a line's character length is at or over this number, the function will try to break it into a new line. In other words, this is the <i>lower bound</i> of the line width. See 'Details' if an upper bound is desired instead.
<code>args.newline</code>	Whether to start the arguments of a function call on a new line instead of after the function name and <code>(</code> when the arguments cannot fit one line.
<code>output</code>	Whether to output to the console or a file using <code>cat()</code> .
<code>text</code>	An alternative way to specify the input: if <code>NULL</code> , the function will use the source argument; if a character vector containing the source code, the function will use this and ignore the source argument.
<code>...</code>	Other arguments passed to <code>cat()</code> , e.g. <code>file</code> (this can be useful for batch-processing R scripts, e.g. <code>tidy_source(source = 'input.R', file = 'output.R')</code>).

Details

A value of the argument `width.cutoff` wrapped in `I()` (e.g., `I(60)`) will be treated as the *upper bound* of the line width. The corresponding argument to `deparse()` is a lower bound, so the function will perform a binary search for a width value that can make `deparse()` return code with line width smaller than or equal to the `width.cutoff` value. If the search fails, a warning will signal, suppressible by global option `options(formatR.width.warning = FALSE)`.

Value

A list with components

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

Note

Be sure to read the reference to know other limitations.

Author(s)

Yihui Xie <<https://yihui.org>> with substantial contribution from Yixuan Qiu <<https://yixuan.blog>>

References

<https://yihui.org/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 from 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

<code>FUN</code>	The function name.
<code>width</code>	The width of the output.
<code>tidy</code>	Whether to reformat the usage code.
<code>output</code>	Whether to print the output to the console (via cat()).
<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 represents the action taken when the width constraint is unfulfillable. "warn" and "stop" will signal warnings and errors, while "none" will do nothing.

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

Reformatted usage code of a function, in character strings (invisible).

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, [8](#)

cat, [3](#), [6](#), [8](#)

deparse, [5](#), [6](#)

getOption, [8](#)

I, [6](#)

parse, [5](#), [6](#)

tidy_app, [2](#)

tidy_dir, [2](#)

tidy_eval, [3](#)

tidy_file(tidy_dir), [2](#)

tidy_pipe, [4](#)

tidy_rstudio, [4](#)

tidy_source, [2–4](#), [5](#), [8](#), [9](#)

usage, [8](#)