Package 'formatR'

August 25, 2014

Type Package

Title Format R Code Automatically

Version 1.0

Date 2014-08-25

Author Yihui Xie

Maintainer Yihui Xie <xie@yihui.name>

Description This package 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()).

Suggests codetools, shiny, testit, knitr

License GPL

URL http://yihui.name/formatR

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

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-25 00:55:00

R topics documented:

tidy_app *A Shiny app to format R code*

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 *Format the R scripts under a directory*

Description

This function 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 `tidy_source` is using.

Usage

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

Arguments

`path`	the directory
`recursive`	whether to recursively look for R scripts under `path`
`...`	other arguments to be passed to `tidy_source`

Value

Invisible `NULL`.

Author(s)

Yihui Xie <http://yihui.name>

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 envir = NULL or envir = new.env())

Value

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

References

http://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(file.path(system.file(package = "stats"), "demo", "nlm.R"))
```

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), 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.souce() 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)
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 <http://yihui.name> with substantial contribution from Yixuan Qiu <http://yixuan.cos.name>

References

http://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)
```

Arguments

`FUN`	the function name
`width`	the width of output (passed to `width.cutoff` in `tidy_source`)
`tidy`	whether or not to reformat the usage code

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)  # narrower output
```

Index