

# formatR: Format R Code Automatically

Yihui Xie\*

April 2, 2012

The package **formatR** (Xie, 2012a) was designed to help users tidy (reformat) their source code. This vignette serves as a showcase of the function *tidy.source()*, and a broader introduction can be found in <https://github.com/yihui/formatR/wiki/>.

## 1 The workhorse *tidy.source()*

The main function in this package is *tidy.source()*, which can take a file as input, parse it and write the formatted code to the console or a file.

```
library(formatR)
usage(tidy.source, width = 0.73)

## tidy.source(source = "clipboard", keep.comment = getOption("keep.comment",
## TRUE), keep.blank.line = getOption("keep.blank.line",
## TRUE), keep.space = getOption("keep.space", FALSE),
## replace.assign = getOption("replace.assign", FALSE),
## reindent.spaces = getOption("reindent.spaces", 4),
## output = TRUE, text = NULL, width.cutoff = getOption("width"),
## ...)
```

There are five options which can affect the output: `keep.comment`, `keep.blank.line`, `keep.space`, `reindent.spaces` and `replace.assign`. They are explained in the help page; see `?tidy.source`. For example, if we do not want to keep the blank lines in the code, we can first specify a global option like this:

```
options(keep.blank.line = TRUE) # not really need to do so; default is TRUE
```

The option `width` will affect the width of the output, e.g. we can specify a narrow width:

```
options(width = 85)
```

Here are some examples taken from the help page:

```
library(formatR)
## use the 'text' argument
src = c("    # a single line of comments is preserved", "1+1", "if(TRUE){",
        "x=1 # inline comments", "}else{", "x=2;print('Oh no... ask the right bracket",
        "to go away!')})",
```

---

\*Department of Statistics, Iowa State University. Email: [xie@yihui.name](mailto:xie@yihui.name)



```
tidy.source(text = src[1:2], keep.space = TRUE)

# a single line of comments is preserved
1 + 1
```

## Discard blank lines

```
## note the 11th line (was a blank line)!
tidy.source(text = src, keep.blank.line = FALSE)

# a single line of comments is preserved
1 + 1
if (TRUE) {
  x = 1 # inline comments
} else {
  x = 2
  print("Oh no... ask the right bracket to go away!")
}
1 * 3 # one space before this comment will become two!
2 + 2 + 2 # 'short comments'
lm(y ~ x1 + x2) ### only 'single quotes' are allowed in comments
## tabs/spaces before comments: use keep.space=TRUE to keep them
"a character string with \t in it"
# note tabs will be converted to spaces when keep.space=TRUE
1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 +
  1 + 1 + 1 + 1 ## comments after a long line
## here is a long long long long long long long long long long long long long
## long long long long long long comment
```

## Reindent code

```
tidy.source(text = src, reindent.spaces = 2)

# a single line of comments is preserved
1 + 1
if (TRUE) {
  x = 1 # inline comments
} else {
  x = 2
  print("Oh no... ask the right bracket to go away!")
}
1 * 3 # one space before this comment will become two!
2 + 2 + 2 # 'short comments'

lm(y ~ x1 + x2) ### only 'single quotes' are allowed in comments
## tabs/spaces before comments: use keep.space=TRUE to keep them
"a character string with \t in it"
# note tabs will be converted to spaces when keep.space=TRUE
```

```
1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 +
  1 + 1 + 1 + 1  ## comments after a long line
## here is a long long long long long long long long long long long long long
## long long long long long long long comment
```

## Discard comments

```
tidy.source(text = src, keep.comment = FALSE)

1 + 1
if (TRUE) {
  x = 1
} else {
  x = 2
  print("Oh no... ask the right bracket to go away!")
}
1 * 3
2 + 2 + 2
lm(y ~ x1 + x2)
"a character string with \t in it"
1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 +
  1 + 1 + 1 + 1
```

## 2 Applications

This package has been used in a few other R packages. For example, **Rd2roxygen** (Wickham and Xie, 2011) uses **formatR** to reformat the code in the usage and examples sections in Rd files, since the code generated by **roxygen2** is not well-formatted; **pgfSweave** (Bracken and Sharpsteen, 2011) can tidy the Sweave code chunks when the Sweave option `tidy` is `TRUE` (just like the code in this vignette).

## About this vignette

You might be curious about how this vignette was generated, because it looks different from other Sweave-based vignettes. The answer is **knitr** (Xie, 2012b). The real vignette is in L<sub>Y</sub>X, which can be found here:

```
system.file("doc", "formatR.lyx", package = "formatR")
```

Instructions on how to use **knitr** with L<sub>Y</sub>X can be found at <https://github.com/yihui/lyx>.

## References

- Bracken C, Sharpsteen C (2011). *pgfSweave: Quality speedy graphics compilation and caching with Sweave*. R package version 1.2.1, URL <http://CRAN.R-project.org/package=pgfSweave>.
- Wickham H, Xie Y (2011). *Rd2roxygen: Convert Rd to roxygen documentation and utilities to improve documentation*. R package version 1.0-7, URL <https://github.com/yihui/Rd2roxygen>.

Xie Y (2012a). *formatR: Format R Code Automatically*. R package version 0.4, URL <https://github.com/yihui/formatR/wiki>.

Xie Y (2012b). *knitr: A general-purpose package for dynamic report generation in R*. R package version 0.4.1, URL <http://yihui.name/knitr/>.