formatR: Format R Code Automatically

Yihui Xie*

March 30, 2011

The package **formatR** (Xie, 2011) was designed to help users tidy (reformat) their source code. Features include:

1. automatically add spaces and indent where appropriate;
2. preserve comments while tidying the code;
3. can use a GUI to format the code too;

1 The command line: *tidy.source()*

The main function 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)
> args(tidy.source)
```

```
function (source = "clipboard", keep.comment, keep.blank.line,
    keep.space, output = TRUE, text = NULL, width.cutoff = 0.75 *
        getOption("width"), ...)
NULL
```

There are three options which can affect the final output: `keep.comment`, `keep.blank.line` and `keep.space`. 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 the option like this:

```
> options(keep.blank.line = FALSE)
```

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  # comments begin with at least 2 spaces!", "}else{", "x=2;print('Oh no...
ask the right bracket to go away!')}",
+     "1*3 # this comment will be dropped!", "2+2+2    # 'short comments'",
+     "   ", "lm(y~x1+x2)  ### only 'single quotes' are allowed in comments",
+     "\t\t## 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
```

*Department of Statistics, Iowa State University. Email: xie@yihui.name

```
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+1+1+1+1+1+1+1+1+1+1+1  ## comments after a long line",
+     paste("## here is a", paste(rep("long", 20), collapse = " "),
+         "comment"))
> ## source code
> cat(src, sep = "\n")

    # a single line of comments is preserved
1+1
if(TRUE){
x=1  # comments begin with at least 2 spaces!
}else{
x=2;print('Oh no... ask the right bracket to go away!')}
1*3 # this comment will be dropped!
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          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+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 lon

> ## the formatted version
> tidy.source(text = src)

# a single line of comments is preserved
1 + 1
if (TRUE) {
    x = 1  # comments begin with at least 2 spaces!
} else {
    x = 2
    print("Oh no... ask the right bracket to go away!")
}
1 * 3
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 + 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 long comment

> ## other options: preserve leading spaces
> tidy.source(text = src, keep.space = TRUE)

    # a single line of comments is preserved
1 + 1
if (TRUE) {
    x = 1  # comments begin with at least 2 spaces!
} else {
```

```
    x = 2
    print("Oh no... ask the right bracket to go away!")
}
1 * 3
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          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 lon
```

```
> ## preserve blank lines: note the 11th line!
> tidy.source(text = src, keep.blank.line = TRUE)
```

```
# a single line of comments is preserved
1 + 1
if (TRUE) {
    x = 1  # comments begin with at least 2 spaces!
} else {
    x = 2
    print("Oh no... ask the right bracket to go away!")
}
1 * 3
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 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 + 1
```

2 The GUI: *formatR()*

We can either use the command line version (*tidy.source()*) or the GUI version to format the code: *formatR()* is a function which depends the **gWidgets** package to create a graphical interface (see Figure 1).

3 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 **roxygen** is not well-formatted; **pgfSweave** (Bracken and Sharpsteen, 2010) can tidy the Sweave code chunks when the Sweave option `tidy` is `TRUE` (just like the code in this vignette).

4 Further notes

The tricks used in this packages are very dirty. There might be dangers in using the functions in **formatR**. Please read the documentation carefully to know exactly what kinds of comments could be preserved.

A known issue is that when we write comments in a string (i.e. the comments are not real comments but a part of the string) which is broken into several lines, **formatR** might not work correctly. For example, the code below will make **formatR** fail:

```
mod = "1+1
## comments here can bring troubles
2+2"
```

We need to write like this:

```
mod = paste("1+1", "## comments here can bring troubles", "2+2",
    sep = "\n")
```

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 **pgfSweave** (Bracken and Sharpsteen, 2010). The real vignette is in LyX, which can be found here:

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

Read this blog entry for details and how to reproduce the vignette: `http://yihui.name/en/?p=602`.

References

Bracken C, Sharpsteen C (2010). *pgfSweave: Quality speedy graphics compilation with Sweave*. R package version 1.1.3, URL `http://CRAN.R-project.org/package=pgfSweave`.

Wickham H, Xie Y (2011). *Rd2roxygen: Convert Rd to roxygen documentation*. R package version 0.1-8, URL `https://github.com/yihui/Rd2roxygen`.

Xie Y (2011). *formatR: Format R Code Automatically*. R package version 0.2-1, URL `http://CRAN.R-project.org/package=formatR`.

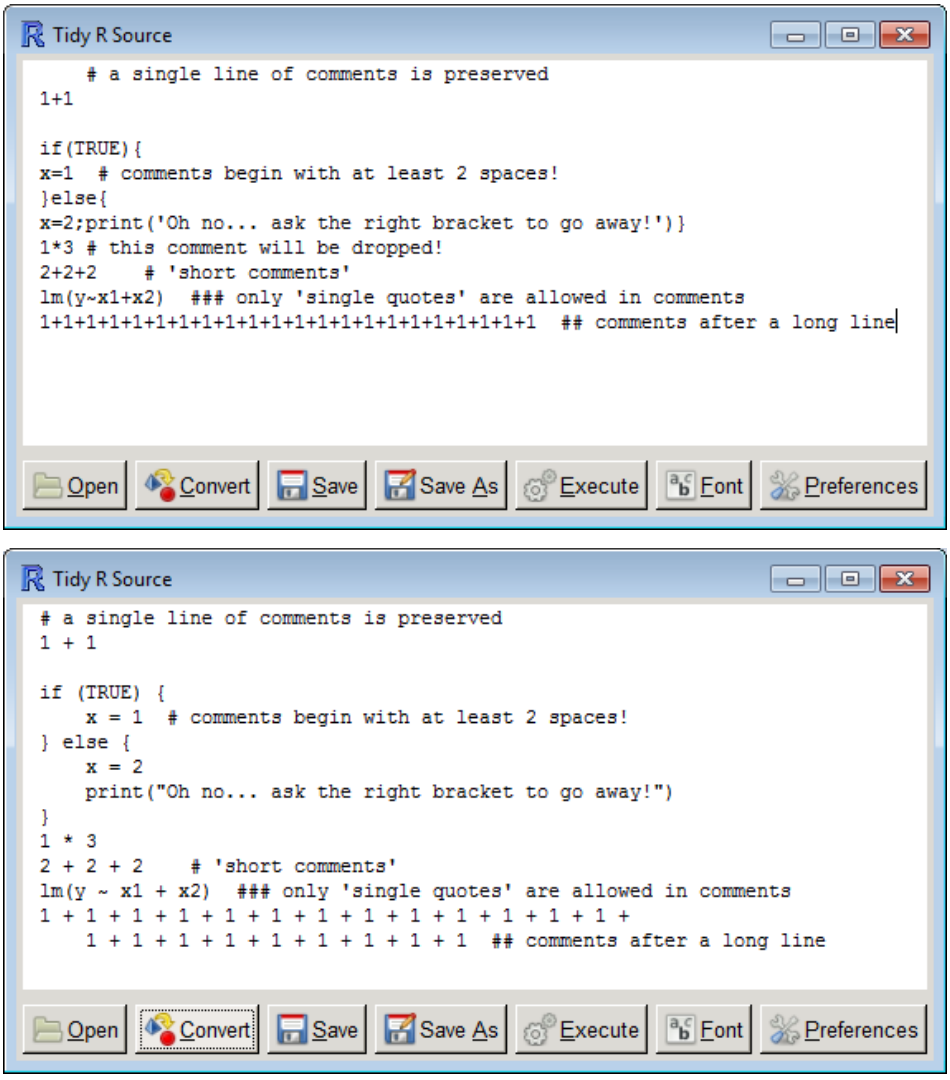

Figure 1: The graphical interface to format R code. Top: the original code; bottom: the formatted code after clicking the Convert button.