Rd2roxygen: Convert Rd to **roxygen** documentation and utilities to enhance **roxygen**

Yihui Xie*

December 18, 2010

The package **Rd2roxygen** (Wickham and Xie, 2010) helps R package developers who used to write R documentation in the raw LaTeX-like commands but want to switch their documentation to **roxygen** (Danenberg and Eugster, 2009), which is a convenient tool for developers, since we can write documentation as inline comments, e.g.

```
>   ## the source code of the function 'parse_and_save'
> ex.file = system.file("examples", "parse_and_save.R", package = "Rd2roxygen")
> cat(readLines(ex.file), sep = "\n")
```

```
##' Parse the input Rd file and save the roxygen documentation into a file.
##'
##' @param path the path of the Rd file
##' @param file the path to save the roxygen documentation
##' @param usage logical: whether to include the usage section in the output
##' @return a character vector if \code{file} is not specified, or write the vector
##' into a file
##' @export
##' @author Hadley Wickham; modified by Yihui Xie <\url{http://yihui.name}>
parse_and_save <- function(path, file, usage = FALSE) {
  parsed <- parse_file(path)
  output <- create_roxygen(parsed, usage = usage)
  if (missing(file)) output else
        cat(paste(output, collapse = "\n"), file = file)
}
```

With **roxygen** (typically using *roxygenize()*), we can create the real Rd file from the above source code like this:

```
> rd.file = system.file("examples", "parse_and_save.Rd", package = "Rd2roxygen")
> cat(readLines(rd.file), sep = "\n")
```

```
\name{parse_and_save}
\alias{parse_and_save}
\title{Parse the input Rd file and save the roxygen documentation into a file.}
\usage{parse_and_save(path, file, usage=FALSE)}
\description{Parse the input Rd file and save the roxygen documentation into a file.}
```

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

```
\value{a character vector if \code{file} is not specified, or write the vector
into a file}
\author{Hadley Wickham; modified by Yihui Xie <\url{http://yihui.name}>}
\arguments{\item{path}{the path of the Rd file}
\item{file}{the path to save the roxygen documentation}
\item{usage}{logical: whether to include the usage section in the output}}
```

The **Rd2roxygen** package goes exactly in the *opposite* way – it parses the Rd files and turns them back to roxygen comments. We can either do this job on single Rd files, or just convert the whole package. The latter might be more useful for developers who are considering the switch.

1 Convert a whole package

The function *Rd2roxygen()* can take a path of a source package, parse all the Rd files under the `man` directory, and write the roxygen comments right above the source code of the functions under the `R` directory.

```
> library(Rd2roxygen)
> args(Rd2roxygen)
```

```
function (pkg, nomatch, usage = FALSE)
NULL
```

```
>   ## e.g. Rd2roxygen('somewhere/to/source/pkg')
>   ## there must be 'man' and 'R' directories under this path
```

2 Parse a single Rd file

We can parse a single Rd file and create the roxygen comments as well with *parse_file()* and *create_roxygen()*, e.g.:

```
>   ## we can specify the roxygen comments prefix (#' by default)
> options(roxygen.comment = "##' ")
> (info = parse_file(rd.file))
```

```
$title
[1] "Parse the input Rd file and save the roxygen documentation into a file."

$usage
[1] "parse_and_save(path, file, usage=FALSE)"

$desc
[1] "Parse the input Rd file and save the roxygen documentation into a file."

$value
[1] "a character vector if \\code{file} is not specified, or write the vector\ninto a file"

$author
[1] "Hadley Wickham; modified by Yihui Xie <\\url{http://yihui.name}>"
```

```
$name
[1] "parse_and_save"

$keywords
list()

$params
[1] "path the path of the Rd file"
[2] "file the path to save the roxygen documentation"
[3] "usage logical: whether to include the usage section in the output"
```

```
> create_roxygen(info)
```

```
 [1] "##' Parse the input Rd file and save the roxygen documentation into a file."
 [2] "##' Parse the input Rd file and save the roxygen documentation into a file."
 [3] "##' "
 [4] "##' "
 [5] "##' @param path the path of the Rd file"
 [6] "##' @param file the path to save the roxygen documentation"
 [7] "##' @param usage logical: whether to include the usage section in the output"
 [8] "##' @return a character vector if \\code{file} is not specified, or write the"
 [9] "##'   vector into a file"
[10] "##' @author Hadley Wickham; modified by Yihui Xie <\\url{http://yihui.name}>"
[11] "\n"
```

```
>   ## parse_and_save() combines the above two steps
```

3 Roxygenize and build a package

This package also provides a tool *roxygen_and_build()* (or in short *rab()*) to help us build the package.

```
> args(roxygen_and_build)
```

```
function (pkg, roxygen.dir = NULL, install = FALSE, check = FALSE,
    check.opts = "", remove.check = TRUE, escape = TRUE, reformat = TRUE,
    use.Rd2 = TRUE, ...)
NULL
```

By default, **roxygen** will generate Rd files for all the objects in the package, which is sometimes not necessary, e.g. the functions which are not exported to the user. These Rd files are removed from the roxygenized package; in fact, we can also use Rd2 in roxygen and specify the tag `@nord` (no Rd) to suppress Rd creation. Another problem is, we often forget to escape the percent symbol `%` in our documentation[1], which will make R treat such texts as comments; *rab()* will escape `%` by default (can be turned off by `escape = FALSE`). Besides, *rab()* also provides options to install or check the package.

[1] One situation which made me crazy for a long time is, it is hard to debug the R documentation when we have `%` in the arguments of the function or in the examples. Image you write a function like this: `f = function(a = '%03d') {}`, and it is really easy to forget to escape the percent symbol correctly in your documentation as `\usage{f(a = '\%03d')}`! Besides, roxygen will generate the documentation without escaping percent symbols, which is often error-prone. That is the reasoning behind the default value `escape = TRUE`.

Yet another feature to mention about *rab()* is that it has an option to "reformat" the code in the usage and example sections; this is due to the fact that roxygen will remove all the leading spaces and indent in the R code, which makes it difficult to read especially when the code is long. If we specify `reformat = TRUE` in *rab()*, the code will be reformated like this:

```
## original code
roxygen_and_build=function(pkg,roxygen.dir=NULL,install=FALSE,check=FALSE,
check.opts='',remove.check=TRUE,escape=TRUE,reformat=TRUE,...){}
```

```
  ## the reformatted code
roxygen_and_build = function(pkg, roxygen.dir = NULL, install = FALSE,
    check = FALSE, check.opts = "", remove.check = TRUE, escape = TRUE,
    reformat = TRUE, ...) {
}
```

Note this functionality depends on the package **formatR** (Xie, 2010), and sometimes it might be not be appropriate to reformat the code, e.g. the `\dontrun{}` command in Rd can contain arbitrary texts, which means there could be illegal R expressions and **formatR** will be unable to format the code. In case of errors, we can consider turning this feature off.

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", "Rd2roxygen.lyx", package = "Rd2roxygen")
```

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 1.1.1, URL `http://CRAN.R-project.org/package=pgfSweave`.

Danenberg P, Eugster M (2009). *roxygen: Literate Programming in R*. R package version 0.1-2, URL `http://CRAN.R-project.org/package=roxygen`.

Wickham H, Xie Y (2010). *Rd2roxygen: Convert Rd to roxygen documentation*. R package version 0.1-3, URL `https://github.com/yihui/Rd2roxygen`.

Xie Y (2010). *formatR: Format R Code Automatically*. R package version 0.1-5, URL `http://CRAN.R-project.org/package=formatR`.