Creating a simple R Package

Gaston Sanchez

October 19, 2016

Creating a minimalist R package

More information can be found in Hadley Wickham's **R Packages** book:

http://r-pkgs.had.co.nz/intro.html

Required Packages

In order to create a package make sure you have the following packages:

```
install.packages(c("devtools", "roxygen2", "testthat", "knitr"))
```

Creating a package as an R Project

Here's the list of suggested steps to create an R package from scratch

- Open RStudio
- Go to `File` in the menu bar
- Select `New Project`
- Choose `New Directory`
- Choose `R Package`
- Specify a name for the package (e.g. `oski`), and choose directory
- Click on button `Create Project`

What's in the package structure?

The series of previous steps should initialize an "R project" with the following contents:

```
.Rbuildignore
oski.Rproj
DESCRIPTION
NAMESPACE
R/
man/
.Rproj.user/
```

- Open the `DESCRIPTION` file and customize its fields with your information
- Don't touch `NAMESPACE` (this will be updated via `devtools`)
- Don't modify the contents in `man/`
- Add R code (functions) in the directory `R/`

Adding code and Roxygen comments

- Go to the `R/` directory and add your own R scripts (containing the functions).
- Document your code using Roxygen comments
- Add examples to the main functions

Here's an example of code for a function `coin()` (in file `coin.R`)

```
#' @title Coin
#' @description Creates an object of class \code{"coin"}
#' @param sides vector of coin sides
#' @param prob vector of side probabilities
#' @return an object of class coin
#' @export
#' @examples
#' # default
#' coin1 <- coin()
#'
#' # another coin
#' coin2 <- coin(c('h', 't'))
#'
#' # us cent
#' cent1 <- coin(c('lincoln', 'shield'))
#'
#' # loaded coin
#' loaded <- coin(prob = c(0.7, 0.3))
#'
coin <- function(sides = c("heads", "tails"), prob = c(0.5, 0.5)) {
  object <- list(
    sides = sides,
    prob = prob)
  class(object) <- "coin"
  object
}
```

Workflow with `"devtools"`

The typical package development workflow with `"devtools"` consists of:

- generating the documentation: `.Rd` files in `man/` directory
- checking that the documentation is OK
- build the bundled package: creates the compressed file (a.k.a. *tarball* `.tar.gz` file); this file ca be shared and installed in any platform.
- install the package: installs the bundled package.

We are ssuming that you are using an R project to work in your package. In this way, R's working directory is actually the directory of your project. Here's the `devtools` commands you should use:

```
library(devtools)

# creating documentation (i.e. the Rd files in man/)
devtools::document()

# checking documentation
devtools::check_man()

# building tarball (e.g. oski_0.1.tar.gz)
devtools::build()

# checking install
devtools::install()
```

More elements: tests and vignettes

You should also include test and vignettes.

To include unit tests:

http://r-pkgs.had.co.nz/tests.html

To include vignettes:

http://r-pkgs.had.co.nz/vignettes.html

Devtools workflow

```
library(devtools)

# creating documentation (i.e. the Rd files in man/)
devtools::document()

# checking documentation
devtools::check_man()

# run tests
devtools::test()

# checking documentation
devtools::build_vignettes()

# building tarball (e.g. oski_0.1.tar.gz)
devtools::build()

# checking install
devtools::install()
```