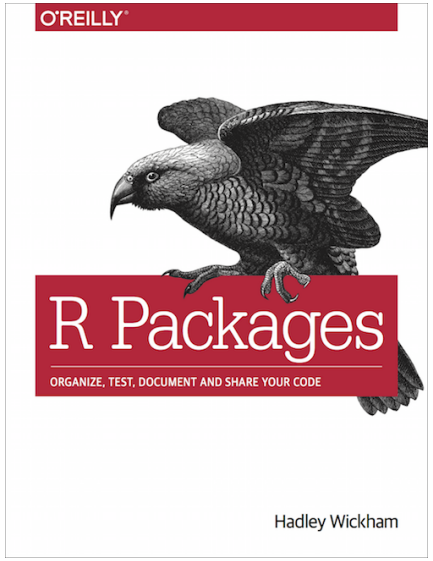
Intro to R Packages

Eric Fox

August 27, 2018

Main reference is Hadley Wickam's book "R Packages", which is available for free online:

`http://r-pkgs.had.co.nz/`



Motivation

Why learn how to create R packages in RStudio:

- Easily share code and collaborate with others.
- Bundle R functions, data sets, and documentation together.
- Provides a consistent template and set of tools for organizing large research projects.

Getting started

First, make sure you have the following packages installed:

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

Getting started

The process of setting up an R package in RStudio is documented here:

`http://r-pkgs.had.co.nz/package.html`

If you are using a government computer make sure you use the `C:/` drive as the directory for your package. R packages will not work on the network.

Components of an R package

- **R functions**: all R functions are placed in the `R/` directory.
- **Documentation**: write documentation for your functions so that others (and future you!) can easily use your package. Documentation exists in the `man/` directory, but can be included with your R function code using `roxygen2`.
- **Data**: place data sets in the `data/` directory for easy access.
- **Installed files**: `inst/` is the 'anything goes' directory; you are free to put anything you like in this folder (e.g., R markdown reports, manuscript drafts, etc.).

Components of an R package

The automatically generated `Description` and `Namespace` files are sufficient for most purposes. If you would like to learn more about these topics I suggest reading:

`http://r-pkgs.had.co.nz/namespace.html`
`http://r-pkgs.had.co.nz/description.html`

Although, understanding these components is not necessary to start creating your own R packages.

Components of an R package

Default directory structure for package created in RStudio.

Name	Date modified	Type
.git	5/15/2017 5:11 PM	File folder
.Rproj.user	5/15/2017 5:10 PM	File folder
man	5/15/2017 5:10 PM	File folder
R	5/15/2017 5:10 PM	File folder
.gitignore	5/15/2017 5:10 PM	Text Document
.Rbuildignore	5/15/2017 5:10 PM	RBUILDIGNORE File
DESCRIPTION	5/15/2017 5:10 PM	File
NAMESPACE	5/15/2017 5:10 PM	File
testpkg.Rproj	5/15/2017 5:10 PM	R Project

Components of an R package

I usually add `data/` and `inst/` folders since these are useful components of a package, especially for research projects.

Name	Date modified	Type
.git	5/15/2017 5:12 PM	File folder
.Rproj.user	5/15/2017 5:10 PM	File folder
data	5/15/2017 5:12 PM	File folder
inst	5/15/2017 5:12 PM	File folder
man	5/15/2017 5:10 PM	File folder
R	5/15/2017 5:10 PM	File folder
.gitignore	5/15/2017 5:10 PM	Text Document
.Rbuildignore	5/15/2017 5:10 PM	RBUILDIGNORE File
DESCRIPTION	5/15/2017 5:10 PM	File
NAMESPACE	5/15/2017 5:10 PM	File
testpkg.Rproj	5/15/2017 5:10 PM	R Project

Example 1: R function and documentation

```
#' Add together two numbers.
#'
#' @param x A number.
#' @param y A number.
#' @return The sum of \code{x} and \code{y}.
#'
#' @examples
#' add(1, 1)
#' add(10, 1)
#'
#' @export
add <- function(x, y) {
  x + y
}
```

Loading R functions

To load all R functions in your package use the following command:

```
devtools::load_all(".")
```

Or use the shortcut *Ctl* + *Shift* + *L*

Each time your change or debug your R functions run this command.

Processing documentation

Run the following command so that roxygen documentation can be previewed in the help pane:

```
devtools::document()
```

Technically, this command converts the roxygen comments into R documentation files (.Rd) which are placed in the `man/` directory.

Each time you edit your function's roxygen documentation run this command and then preview documentation in the help menu.

Example 1: R function and documentation

C:/Users/efox02/Documents/intro_r_packages/testpkg - RStudio

File Edit Code View Plots Session Build Debug Tools Help

add.R

```r
#' Add together two numbers.
#'
#' @param x A number.
#' @param y A number.
#' @return The sum of \code{x} and \code{y}.
#'
#' @examples
#' add(1, 1)
#' add(10, 1)
#'
#' @export
add <- function(x, y) {
  x + y
}
```

Console C:/Users/efox02/Documents/intro_r_packages/testpkg/

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> devtools::load_all(".")
Loading testpkg
> devtools::document()
Updating testpkg documentation
Loading testpkg
Writing NAMESPACE
Writing add.Rd
> add(1,3)
[1] 4
> add(5,-2)
[1] 3
> ?add
Using development documentation for add
>
```

add {testpkg} R Documentation

Add together two numbers.

Description

Add together two numbers.

Usage

```
add(x, y)
```

Arguments

x A number.

y A number.

Value

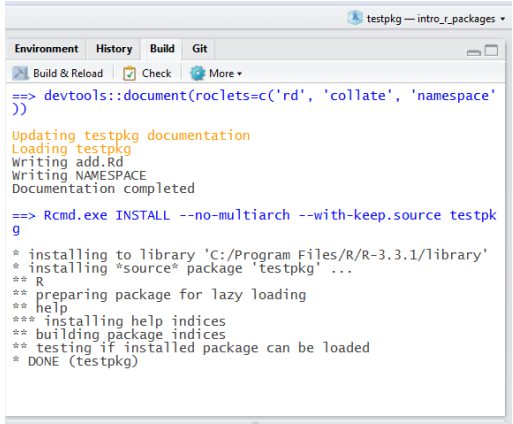
The sum of x and y.

Examples

```
add(1, 1)
add(10, 1)
```

Build a R packages

- Click 'Build & Reload' in the RStudio Build pane to build, install, and then reload your R package.
- This will enable you to use the `library()` function to load your package.
- Building a package can be slow, so if you are in the development stage I recommend using the `devtools` functions (`load_all()` and `document()`) instead.



Data/

- Data sets placed in the `data/` directory will become readily available when you load your package (e.g., using `load_all()` or a manually building and using `library()`).
- Save data sets in `data/` as `.RData` files.
- You can also document data sets using roxygen style comments. See 'Documenting datasets' section of `http://r-pkgs.had.co.nz/data.html` for an example.

Inst/

- You can place anything you like in the `inst/` directory.
- Some examples:
  - R markdown reports,
  - manuscript drafts,
  - R scripts, etc.
- Only exception is don't use an existing package directory name (e.g., avoid `inst/R` or `inst/data`).

Example 2

Write an R function and documentation for computing the root-mean-square error defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where y_i is an observed value and $\hat{y}_i$ is a predicted value (e.g., from a linear model).

Example 2

```
#' RMSE
#'
#' A function that computes the root-mean-square error (RMSE).
#'
#' @param y Numeric vector of observed values.
#' @param yhat Numeric vector of predictions.
#' @return RMSE
#'
#' @export
compute_rmse <- function(y, yhat) {
  n <- length(y)
  sqrt((1 / n) * sum((y - yhat)^2))
}
```