R_Packages

Ryan Womack

2023-08-15

Table of contents

1 R Packages

1.1 The R packages book, by Wickham and Bryan

This is a brief introduction to R package creation, taking as its text the book R Packages, 2nd edition by Hadley Wickham and Jennifer Bryan. This book is freely available on the web, so please consult for further details of topics that are only outlined here. The second edition has been finalized in June 2023.

1.2 Install and load packages

The fundamental packages required to create an R package are *devtools*, *roxygen2*, and *testthat*. Please install these with the following commands if they are not already available on your system. These will not install by default if you just attempt to run this R markdown file.

```
install.packages("devtools", dependencies=TRUE)
install.packages("roxygen2", dependencies=TRUE)
install.packages("testthat", dependencies=TRUE)

devtools::session_info()
```

Let's load those libraries now.

```
library(devtools)
```

```
Loading required package: usethis
```

```
library(roxygen2)
library(testthat)
```

```
Attaching package: 'testthat'
```

```
The following object is masked from 'package:devtools':

    test_file
```

We can check that we're running recent enough versions of our software with the *packageVersion* command for indivual packages, or *session_info* for our entire setup.

```
packageVersion("devtools")
```

```
[1] '2.4.5'
```

```
devtools::session_info()
```

```
- Session info -------------------------------------------------------------
 setting  value
 version  R version 4.2.2 Patched (2022-11-10 r83330)
 os       Debian GNU/Linux 12 (bookworm)
 system   x86_64, linux-gnu
 ui       X11
 language (EN)
 collate  en_US.UTF-8
 ctype    en_US.UTF-8
 tz       America/New_York
 date     2023-08-16
 pandoc   3.1.1 @ /usr/lib/rstudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown)

- Packages -----------------------------------------------------------------
 package     * version date (UTC) lib source
 brio          1.1.3   2021-11-30 [2] CRAN (R 4.0.4)
```

```
cachem        1.0.8   2023-05-01 [2] CRAN (R 4.2.2)
callr         3.7.3   2022-11-02 [2] CRAN (R 4.2.1)
cli           3.6.1   2023-03-23 [2] CRAN (R 4.2.2)
crayon        1.5.2   2022-09-29 [2] CRAN (R 4.2.1)
devtools    * 2.4.5   2022-10-11 [2] CRAN (R 4.2.2)
digest        0.6.33  2023-07-07 [2] CRAN (R 4.2.2)
ellipsis      0.3.2   2021-04-29 [2] CRAN (R 4.0.4)
evaluate      0.21    2023-05-05 [2] CRAN (R 4.2.2)
fastmap       1.1.1   2023-02-24 [2] CRAN (R 4.2.2)
fs            1.6.3   2023-07-20 [2] CRAN (R 4.2.2)
glue          1.6.2   2022-02-24 [2] CRAN (R 4.1.2)
htmltools     0.5.5   2023-03-23 [2] CRAN (R 4.2.2)
htmlwidgets   1.6.2   2023-03-17 [2] CRAN (R 4.2.2)
httpuv        1.6.11  2023-05-11 [2] CRAN (R 4.2.2)
jsonlite      1.8.7   2023-06-29 [2] CRAN (R 4.2.2)
knitr         1.43    2023-05-25 [2] CRAN (R 4.2.2)
later         1.3.1   2023-05-02 [2] CRAN (R 4.2.2)
lifecycle     1.0.3   2022-10-07 [2] CRAN (R 4.2.1)
magrittr      2.0.3   2022-03-30 [2] CRAN (R 4.1.3)
memoise       2.0.1   2021-11-26 [2] CRAN (R 4.0.4)
mime          0.12    2021-09-28 [2] CRAN (R 4.0.4)
miniUI        0.1.1.1 2018-05-18 [2] CRAN (R 4.2.2)
pkgbuild      1.4.2   2023-06-26 [2] CRAN (R 4.2.2)
pkgload       1.3.2.1 2023-07-08 [2] CRAN (R 4.2.2)
prettyunits   1.1.1   2020-01-24 [2] CRAN (R 4.0.4)
processx      3.8.2   2023-06-30 [2] CRAN (R 4.2.2)
profvis       0.3.8   2023-05-02 [2] CRAN (R 4.2.2)
promises      1.2.0.1 2021-02-11 [2] CRAN (R 4.0.4)
ps            1.7.5   2023-04-18 [2] CRAN (R 4.2.2)
purrr         1.0.1   2023-01-10 [2] CRAN (R 4.2.2)
R6            2.5.1   2021-08-19 [2] CRAN (R 4.0.4)
Rcpp          1.0.11  2023-07-06 [2] CRAN (R 4.2.2)
remotes       2.4.2.1 2023-07-18 [2] CRAN (R 4.2.2)
rlang         1.1.1   2023-04-28 [2] CRAN (R 4.2.2)
rmarkdown     2.23    2023-07-01 [2] CRAN (R 4.2.2)
roxygen2    * 7.2.3   2022-12-08 [2] CRAN (R 4.2.2)
rstudioapi    0.15.0  2023-07-07 [2] CRAN (R 4.2.2)
sessioninfo   1.2.2   2021-12-06 [2] CRAN (R 4.0.4)
shiny         1.7.4.1 2023-07-06 [2] CRAN (R 4.2.2)
stringi       1.7.12  2023-01-11 [2] CRAN (R 4.2.2)
stringr       1.5.0   2022-12-02 [2] CRAN (R 4.2.2)
testthat    * 3.1.10  2023-07-06 [2] CRAN (R 4.2.2)
urlchecker    1.0.1   2021-11-30 [2] CRAN (R 4.2.2)
usethis     * 2.2.2   2023-07-06 [2] CRAN (R 4.2.2)
vctrs         0.6.3   2023-06-14 [2] CRAN (R 4.2.2)
```

```
xfun          0.39    2023-04-20 [2] CRAN (R 4.2.2)
xml2          1.3.5   2023-07-06 [2] CRAN (R 4.2.2)
xtable        1.8-4   2019-04-21 [2] CRAN (R 4.0.4)
yaml          2.3.7   2023-01-23 [2] CRAN (R 4.2.2)

[1] /home/ryan/R/x86_64-pc-linux-gnu-library/4.2
[2] /usr/local/lib/R/site-library
[3] /usr/lib/R/site-library
[4] /usr/lib/R/library
```

1.3 Outline of the creation of a little package

We'll follow along with Chapter 1 of the R Packages book and walk through the creation of a little package, even simpler than the "toy package" presented in the text.

This will enable us to review the fundamental features of a typical package:

- functions
- version control
- documentation (*roxygen2*)
- testing (*testthat*)
- creation of a README.Rmd file

After we do this for our little package, we'll coverage some additional details relating to the steps above.

1.4 Create_package function

We call *create_package* to initiate a package. We want to start this in its own fresh directory, not a pre-existing project or git repository. The *create_package* function will set up the necessary folder structure for a package. Please *EDIT* the contents of the command below to correspond to your computer's file system. This is the one place in the code where you'll have to modify it. Note that to be a valid package name and to be allowed on CRAN, the package name should:

- Contain only ASCII letters, numbers, and '.'
- Have at least two characters
- Start with a letter
- Not end with '.'

```
create_package("/home/ryan/R/littlePackage")
```

```
v Setting active project to '/home/ryan/R/littlePackage'


Package: littlePackage
Title: What the Package Does (One Line, Title Case)
Version: 0.0.0.9000
Authors@R (parsed):
    * First Last <first.last@example.com> [aut, cre] (YOUR-ORCID-ID)
Description: What the package does (one paragraph).
License: `use_mit_license()`, `use_gpl3_license()` or friends to pick a
    license
Encoding: UTF-8
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.2.3


v Setting active project to '<no active project>'
```

This will launch a new window. To be able to continue using our script here, just close that window and navigate in the original window to the package directory (same as what you used above), either via the RStudio Files window or with the following *setwd* command (modified with the directory you used). Alternatively, you could open the markdown file in the new session, but be sure to rerun the library commands in lines 30-32 above.

```
setwd("/home/ryan/R/littlePackage")
```

The "dot files" beginning with a period (.) are used to store a history of the R session and to tell R and git to ignore certain files. Generally we can leave these files as is and let R worry about how to handle them. You shouldn't need to modify these files in most circumstances.

Likewise, the .Rproj file is usually left unmodified. This helps RStudio manage the package folder as a project.

The NAMESPACE is also a file we won't edit. It is used to keep track of relations between functions that your package will use, but you can let RStudio handle this.

One of the two locations that we *WILL* edit are the *DESCRIPTION*, which is a structured way of providing information about your package. This is what you will see when you look up the function in the R help system, or what would be displayed if your package makes it onto CRAN. For example, try typing ?testthat to see the description of that package.

The other is the *R* folder. This is the folder that we will put our functions into. Optionally we could add a *data* folder as well, if we wanted to distribute data via our package.

It is quite convenient to use *create_package* to take care of all of this for us.