Your First R Package

Download these slides from goo.gl/aiuNFp

Alex Hayes

2018-02-20

Pre-requisites

1. Install RStudio + R
2. Install some important packages:

```r
pkgs <- c("devtools", "usethis",
          "roxygen2", "testthat")
install.packages(pkgs)
```

Why Should You Use R Packages

- Keep code that you use frequently in one place: hayeslib
- Fewer copy-paste errors
- Easy to share code with others
- Understand why packages work the way they do
- Learn where documentation lives
- Nerd cred

Documentation pop-quiz

1. How you find the documentation for the `lm` function?
2. How do you see the source code for the `lm` function?

Take 2 minutes

Our Goal Today

Create a personal package with two functions

Function 1: Get OLS coefficients

```r
ols_coefs <- function(X, y) {
  solve(t(X) %*% X) %*% t(X) %*% y
}
```

Function 2: MEMES

```r
ernst_meme <- function(upper = "", lower = "",
                       vjust = 0.25, ...) {
  if (.Platform$OS.type == "windows") {
    windowsFonts(Impact = windowsFont("Impact"),
                 Courier = windowsFont("Courier"))
  }
  u <- system.file("extdata", "ernst.jpg",
                   package = "mypkg")
  meme::meme(u, upper = upper, lower = lower,
             vjust = vjust, ...)
}
```

Putting these functions into an R package

Live demo: create an R package skeleton with RStudio

What are these files?

`.gitignore`: Makes using `git` nice.

`.Rbuildignore`: You can ignore this for now.

`DESCRIPTION`: This is where all the meta-data about your package goes. More in a slide.

`mypkg.Rproj`: Turns the directory into an RStudio project and allows you to save RStudio settings specific to the package.

`NAMESPACE`: Controls which functions your package shows (“exports”) to users, and which functions it depends on (“imports”). You can ignore this file since `devtools` will create it for you.

`R`: A folder where your R code goes.

DESCRIPTION (template)

We fill this in with the relevant info:

```
Package: mypkg
Title: What the Package Does (one line, title case)
Version: 0.0.0.9000
Authors@R: person("First", "Last",
                  email = "first.last@example.com",
                  role = c("aut", "cre"))
Description: What the package does (one paragraph).
Depends: R (>= 3.4.1)
License: What license is it under?
Encoding: UTF-8
LazyData: true
```

DESCRIPTION (template filled in)

```
Package: mypkg
Title: Calculate OLS Coefficients and Make Memes
Version: 0.0.0.9000
Authors@R: person("Alex", "Hayes",
                  email = "aph3@rice.edu",
                  role = c("aut", "cre"))
Description: Provides function to calculate OLS
    coefficients and make memes of
    professors in the Rice statistics dept.
Depends: R (>= 3.4.1)
License: MIT
Encoding: UTF-8
LazyData: true
```

Putting `ols_coefs` into our package

```r
#' Get estimates of OLS coefficients
#'
#' @param X A data matrix.
#' @param y A response vector.
#'
#' @return A vector of coefficients
#' @export

ols_coefs <- function(X, y) {
  solve(t(X) %*% X) %*% t(X) %*% y
}
```

Live Demo

Putting `ernst_meme` into our package

1. Download this picture of Ernst.
2. Put it in `inst/extdata/ernst.jpg`
3. Add a dependency on the `meme` package

```
usethis::use_package("meme")
```

`ernst_meme` documentation

```r
#' Create a meme of professor Ernst
#'
#' @param upper Text to display at top of image.
#' @param lower Text to display at bottom of image.
#' @param vjust Vertical adjustment. Higher number
#'   means text closer to center of image.
#' @param ... Other arguments passed to `meme::meme`
#'
#' @return ggplot2 meme object
#' @export
ernst_meme <- function(upper = "", lower = "",
                       vjust = 0.25, ...) {
  if (.Platform$OS.type == "windows") {
    windowsFonts(Impact = windowsFont("Impact"),
```

DOCUMENT! DOCUMENT! DOCUMENT!!!

Why:

1. Need to tell R to export our functions.
2. Need to describe what the functions do for when we inevitably forget in two days.

Live demo adding in a third function from scratch!

Does it work??

1. Compile the documentation!
2. Build and install!

Live demo

Code from live demo

```r
library(mypkg)

X <- model.matrix(mpg ~ hp, mtcars)
y <- mtcars$mpg
ols_coefs(X, y)

ernst_meme(
  lower = "something something probability related"
)

?ols_coefs
?ernst_meme
```

Congrats!

Workflow: refresher

1. Put functions into `my_functions.R` files.
2. Documentation the functions in `my_functions.R`.
3. Compile the documentation.
4. Build and install the package. (or load the functions!)
5. Test that things work like you'd expect.

Loading vs building a package

Loading: makes the package functions available in the current session

Building and installing: Installs the package on your computer, after which you can access the functions with `library(mypkg)`

Add a simple test

1. Start with:

```r
usethis::use_testthat()
usethis::use_test("ols")
```

Testing continued

2. Change `tests/testthat/test-ols.R` to

```r
context("test-ols.R")

test_that("multiplication works", {

  X <- model.matrix(mpg ~ hp, mtcars)
  y <- mtcars$mpg

  result <- ols_coefs(X, y)
  expected <- coef(lm(mpg ~ hp, mtcars))

  expect_equivalent(result, expected)
})
```

Even more testing

3. Run the tests
4. Change your functions until the tests pass

Next Steps (details in Hadley's book)

- Learn the RStudio keyboard shortcuts
- Run the CRAN checks with `devtools::check()`
- Sharing your package on Github + CRAN (add a LICENSE!)
- Make a website for your package with `pkgdown`
- Advertise your package
- Provide vignettes (examples) showing how to use your package

Questions?

Resources:

- Hilary Parker's Writing an R package from Scratch blog post
- Hadley Wickham's book R Packages
- `#rstats` on Twitter

@alexpghayes on Twitter
alexpghayes@gmail.com

Summer opportunity: `MoMA` package with Michael Weylandt and Dr. Allen