Your First R Package

Please download these slides from TODO

Alex Hayes

2018-02-20

Pre-requisites

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

```
pkgs <- c("devtools", "usethis", "roxygen2")
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

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

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

Function 2: MEMES

```
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

Link to code

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

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

`ernst_meme` documentation

```
#' 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 me
#'   closer to center of image.
#' @param ... Other arguments passed to `meme::meme`
#'
#' @return ggplot2 meme object
#' @export
```

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

Does it work??

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

Live demo

Code from live demo

```
library(mypkg)

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

ernst_meme(lower = "something something probability re

?ols_coefs
?ernst_meme
```

Congrats: You've just created your first R package!

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.
5. Test that things work like you'd expect.

This is the tip of an iceberg: resources

Questions?

@alexpghayes on Twitter alexpghayes@gmail.com

Resources:

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

Next Steps

1. Testing your code (have as backup slides)
2. Sharing your code: Github + CRAN

Some notes on sharing your code

ADVERTISE DOCUMENT

Putting these skills to good use

Michael Weylandy opportunity to develop interface stuff

Back up slides

LICENSE

```
usethis::use_mit_license("Alex Hayes")
```