

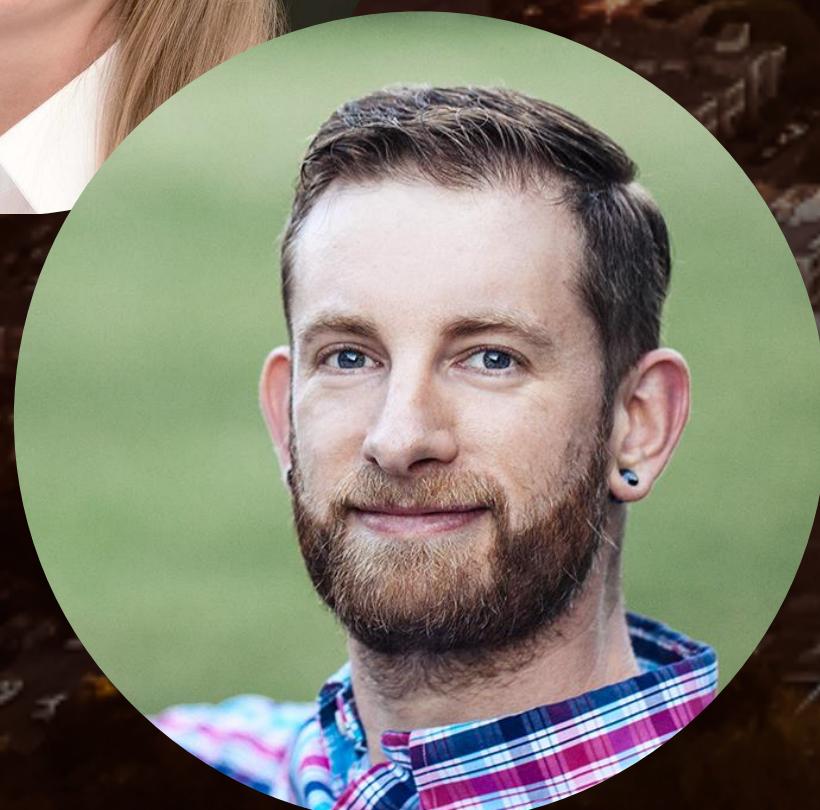
Building Tidy Tools

Charlotte Wickham &
Hadley Wickham



rstudio::conf
SAN FRANCISCO // JANUARY 27 - 30, 2020

from RStudio



Welcome!

Building Tidy Tools

Charlotte Wickham and Hadley Wickham

Wifi: rstudio20

Password: tidyverse20

Please get set up using the instructions at:

<http://rstd.io/build-tt>

Need Help? Questions?

Raise a **PINK** sticky note on your laptop

Materials: <http://rstd.io/build-tt>



Workshop Policies

- Identify the exits closest to you in case of emergency
- Please review the `rstudio::conf` code of conduct that applies to all workshops. Issues can be addressed three ways:
 - In person: contact any `rstudio::conf` staff member or the conference registration desk
 - By email: send a message to conf@rstudio.com
 - By phone: call 844-448-1212
- Please do not photograph people wearing red lanyards
- A chill-out room is available for neurologically diverse attendees on the 4th floor of tower 1

Your turn

This course is hands-on and, while we're here to help, the best resource may be the person sitting next to you.

Introduce yourself to your neighbours. Who are you and what are you using R for?

This means that
you have to work!

Ingrid
Rodriguez

Julia
Blum

François
Michonneau

Materials: <http://rstd.io/build-tt>

Outline for today

9:00-10:30

Introduction /
"The Whole Game"

Charlotte

11:00-12:30

Testing

Charlotte

13:30-15:00

Documentation /
Sharing

Charlotte

15:30-17:00

Dependencies

Hadley

Materials: <http://rstd.io/build-tt>

Outline for tomorrow

9:00-10:30

Using the tidyverse in
packages

Charlotte

11:00-12:30

Interface

Hadley

13:30-15:00

Interface

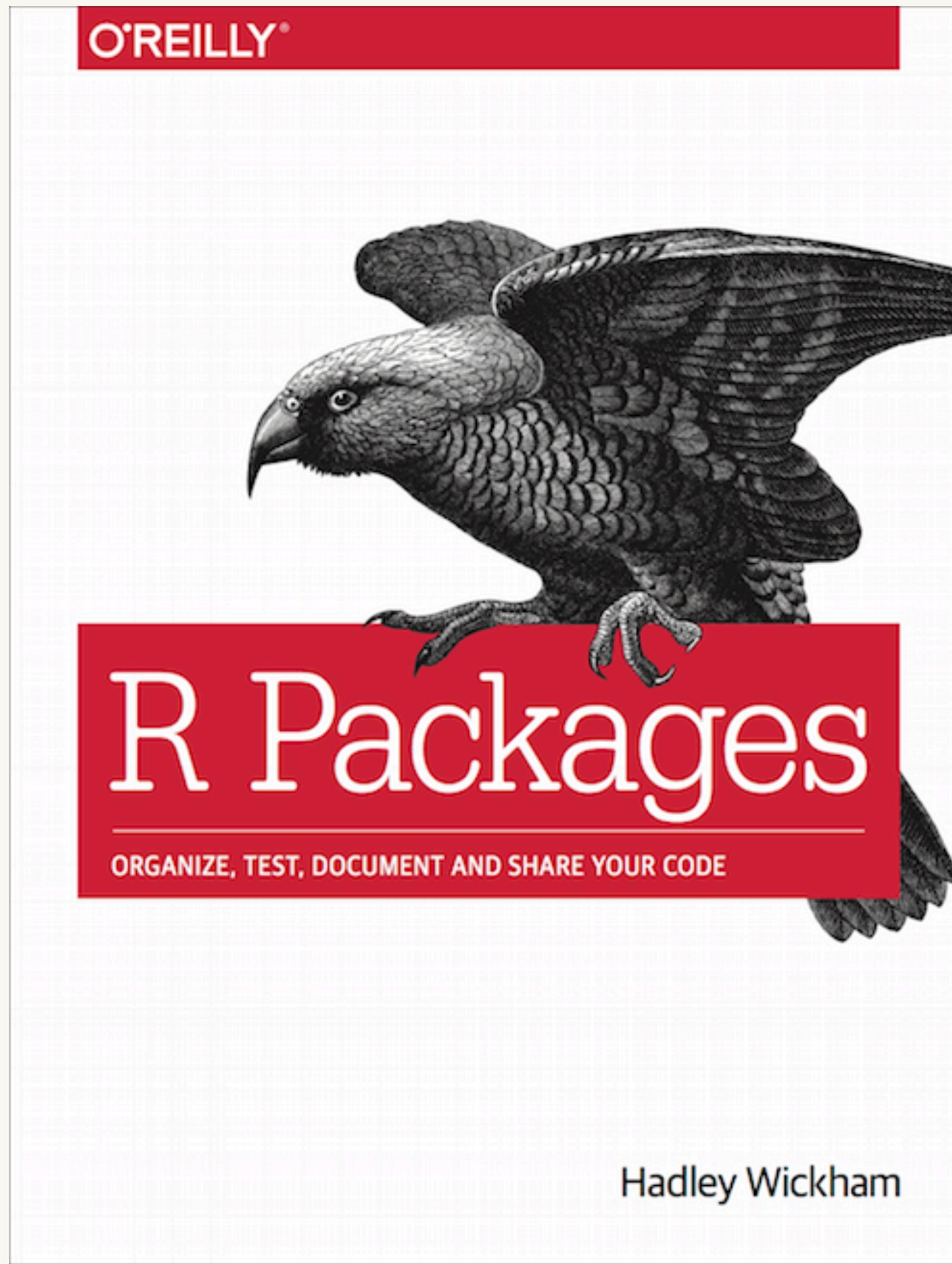
Hadley

15:30-17:00

OO programming/S3
Wrap Up

Charlotte

Materials: <http://rstd.io/build-tt>



<https://r-pkgs.org/>

What They Forgot to Teach You About R

Jennifer Bryan, Jim Hester

WTF

0.1 In-person workshops

- I A holistic workflow
- 2 Project-oriented workflow
- 3 Practice safe paths
- 4 How to name files
- 5 API for an analysis
- II Personal R Admin
- 6 Get to know your R installation
- 7 R Startup
- 8 Maintaining R
- 9 Set up an R dev environment
- 10 Install a source package
- III All is fall
- 11 Debugging R code
- 12 Read the sources
- 13 Reproduce the problem
- IV Backmatter

The impetus for developing and assembling these materials is a two-day hands-on workshop. It is designed for experienced R and RStudio users who want to (re)design their R lifestyle. We focus on building holistic and project-oriented workflows that address the most common sources of friction in data analysis, outside of doing the statistical analysis itself.

Warning: these materials absolutely do not constitute a self-contained "book", nor do they capture all workshop content. But it is useful to us to organize and share certain excerpts in this format.

Repo that makes this site: <https://github.com/jennybc/what-they-forgot>

<https://rstats.wtf>

Happy Git and GitHub for the useR

Jenny Bryan, the STAT 545 TAs, Jim Hester

Let's Git started

1 Why Git? Why GitHub?

- 2 Contributions
- 3 Workshops
- 4 Installation
- 5 Har the battle
- 6 Register a GitHub account
- 7 Install or upgrade R and RStudio
- 8 Install Git
- 9 Introduce yourself to Git
- 10 Connect Git, GitHub, RStudio
- 11 Can you hear me now?
- 12 Connect to GitHub
- 13 Cache credentials for HTTPS
- 14 Set up keys for SSH
- 15 Connect RStudio to Git and GitHub
- 16 Detect Git from RStudio

WATCH ME DIFF
WATCH ME REBASE

<https://happygitwithr.com>

Materials: <http://rstd.io/build-tt>

Warmup

Getting to know your R installation!

Read more: <https://r-pkgs.org/package-structure-state.html>

Materials: <http://rstd.io/build-tt>

Your turn



```
# How do you install a package from CRAN?  
# How do you install a package from GitHub?
```

Materials: <http://rstd.io/build-tt>

Handful of ways of installing packages

```
install.packages("devtools")
pak::pkg_install("devtools")
```

```
devtools::install_github("r-lib/itdepends")
remotes::install_github("r-lib/itdepends")
pak::pkg_install("r-lib/itdepends")
```

```
# What's the difference between devtools
# and remotes?
```

Your turn



```
# How does installing a package change your  
# computer?  
# What is a library? How many libraries do you  
# have? Which is the default?
```

.Library

.libPaths()

Materials: <http://rstd.io/build-tt>

Library = directory of R packages

base R =

14 **base** packages+

29 **recommended** (also on CRAN) packages

Automatically installed with R.

Your turn

```
# How does running library(dplyr) affect your  
# computer? How is it connected to your  
# libraries?  
  
# Hint: try comparing this code before and  
# after  
data.frame(  
  env = search(),  
  path = searchpaths()  
)
```

`library(pkg)` **attaches** a package

7 base packages are always attached

Use R --vanilla to check

The whole game

Materials: <http://rstd.io/build-tt>

What follows is adapted from

The Whole Game

chapter in the revised version of R Packages.

<https://r-pkgs.org/whole-game.html>

A proper package for the care and feeding of factors:

forcats

<https://forcats.tidyverse.org>

A package is a set of
conventions that
(with the right tools)
makes your life easier

usethis::create_package()

What does `create_package()` do?

- ✓ Creating '/Users/jenny/tmp/foofactors2/'
- ✓ Setting active project to '/Users/jenny/tmp/foofactors2'
- ✓ Creating 'R/'
- ✓ Writing 'DESCRIPTION'

Package: foofactors2

Title: What the Package Does (One Line, Title Case)

Version: 0.0.0.9000

Authors@R (parsed):

* Jennifer Bryan <jenny@rstudio.com> [aut, cre]

Description: What the package does (one paragraph).

License: MIT + file LICENSE

Encoding: UTF-8

LazyData: true

✓ Writing 'NAMESPACE'

✓ Writing 'foofactors2.Rproj'

✓ Adding '.Rproj.user' to '.gitignore'

✓ Adding '^foofactors2\\.Rproj\$', '^\\.Rproj\\\\.user\$' to '.Rbuildignore'

✓ Opening '/Users/jenny/tmp/foofactors2/' in new RStudio session

✓ Setting active project to '<no active project>'

use_git()

Not going to teach it,
but diffs are helpful

Factors can be vexing

```
(a <- factor(c("character", "in", "the", "streets")))
#> [1] character in the streets
#> Levels: character in streets the
(b <- factor(c("integer", "in", "the", "sheets")))
#> [1] integer in the sheets
#> Levels: in integer sheets the

c(a, b)
#> [1] 1 2 4 3 2 1 4 3
```

Factors can be vexing

```
factor(c(as.character(a), as.character(b)))
#> [1] character in      the      streets    integer   in
#> [7] the      sheets
#> Levels: character in integer sheets streets the
```

Let's turn this into our first function:

`fbind()`

use_r()

Where do we define functions?

Beautiful pairing:
use_r() & use_test()

```
# There's a usethis helper for that too!  
usethis::use_r("file-name")
```

```
# Organise files so that related code  
# lives together. If you can give a file  
# a concise and informative name, it's  
# probably about right
```

Now what?

```
source("R/fbind.R")
```

Use IDE tricks to send definition of
fbind() to the R Console

Now what?

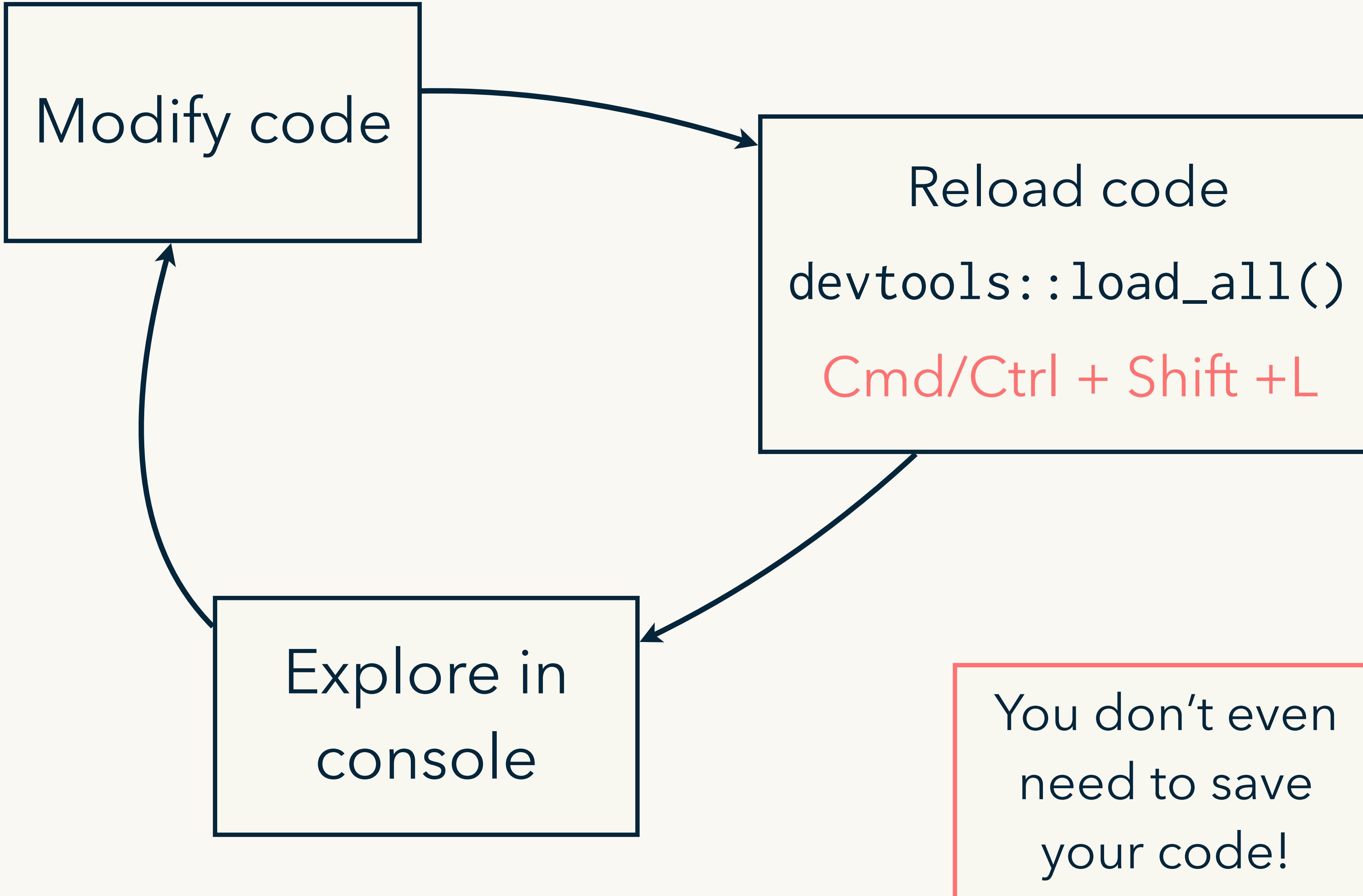
~~source("R/fbind.R")~~

~~Use IDE tricks to send definition of
fbind() to the R Console~~

devtools::load_all()

devtools::load_all()

Why do we love devtools? Workflow!



Important metadata files exist in all versions

In binary versions, documentation is compiled into multiple versions. A parsed version of DESCRIPTION is cached for performance.

In binary versions, R/ no longer contains .R files, but instead contains binary .Rdata files

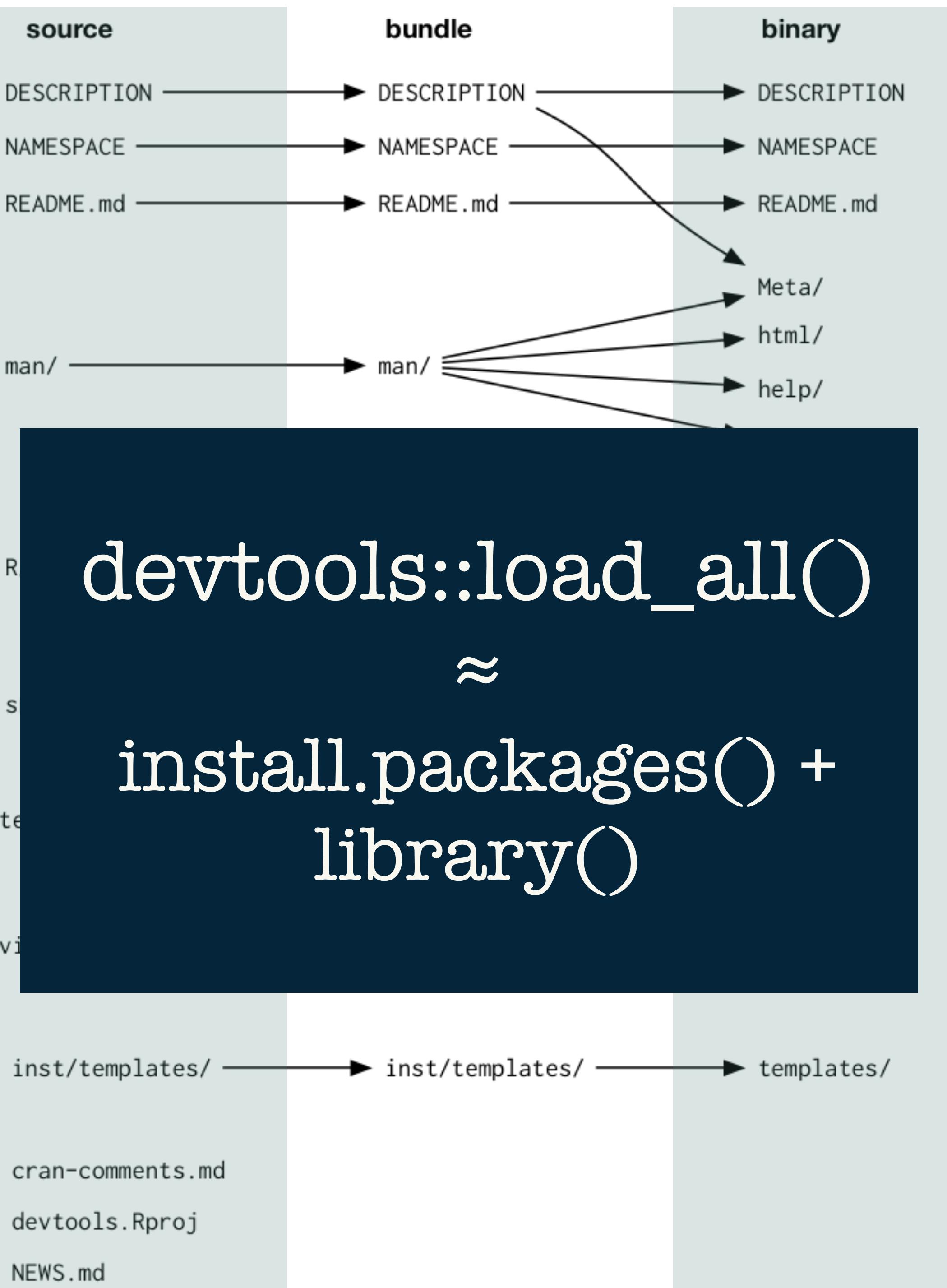
Compilation results are saved in libs/

By default, tests are dropped in binary packages

Source vignettes are build into html or pdf in inst/doc, then installed into doc/

The contents of inst/ are moved into the top-level directory

Files used only for development are listed in .Rbuildignore, and only exist in source package



devtools::check()



`check()` ≈ R CMD check

Checks package for technical validity

Do from R (or RStudio Ctrl/cmd + shift + e)

`check()` early, `check()` often

Get it working, keep it working

Necessary (but not sufficient) for CRAN

Excellent way to run your tests (and more)

devtools::document()

roxygen2 turns comments into help

```
#' Bind two factors
#'
#' Create a new factor from two existing factors, where the new
#' factor's levels are the union of the levels of the input
#' factors.
#'
#' @param a factor
#' @param b factor
#'
#' @return factor
#' @export
#' @examples
#'
#> fbind(factor(letters[1:3]), factor(letters[26:24]))
fbind <- function(a, b) {
  factor(c(as.character(a), as.character(b)))
}
```

RStudio helper:

Code > Insert roxygen skeleton

devtools::check()



devtools::install()



`install()` \approx R CMD install

- Makes an *installed* pkg from your source pkg
- Do from R (or RStudio *Install and Restart*)
- `install()` less often than you `load_all()` or `check()`
- Marks transition from **developing** your package
to **using** your package

Your turn

R/RStudio setup

Workflow setup: your .Rprofile

```
# Setup code that is run at R startup:  
# usethis::edit_r_profile()  
  
# Helper to add devtools specifically:  
# usethis::use_devtools()  
  
if (interactive()) {  
  suppressMessages(require(devtools))  
  suppressMessages(require(testthat))  
}  
devtools makes  
usethis available too!
```

Never include analysis packages here

```
if (interactive()) {  
  suppressMessages(require(ggplot2))  
  suppressMessages(require(dplyr))  
}
```

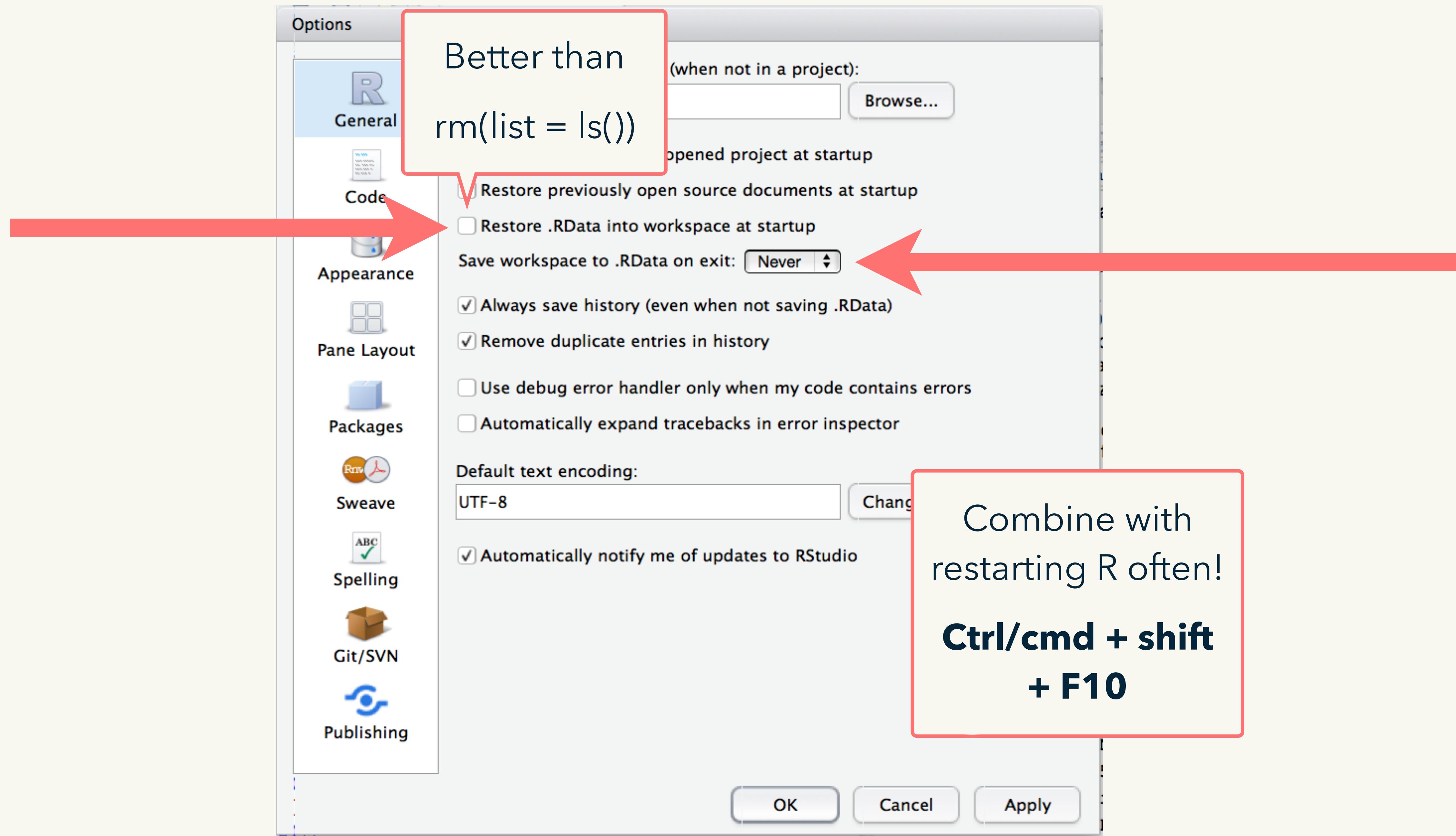
While you're in there, also add:

```
# Helper to add devtools specifically:  
# usethis::use_partial_warnings()  
  
options(  
  warnPartialMatchArgs = TRUE,  
  warnPartialMatchDollar = TRUE,  
  warnPartialMatchAttr = TRUE  
)
```

Tell usethis about yourself

```
options(  
  usethis.full_name = "Hadley Wickham",  
  usethis.description = list(  
    `Authors@R` = 'person(  
      "Hadley", "Wickham", Your first and last names  
      email = "hadley@rstudio.com", Your email  
      role = c("aut", "cre"), Leave as is  
      comment = c(ORCID = "YOUR-ORCID-ID")  
    )'  
  )'  
)
```

Delete this line if you don't have an ORCID



Set up your computer

`usethis::edit_r_profile():`

Load devtools & usethis

Set partial matching warnings

Tell usethis about yourself

In RStudio settings:

Don't save or reload your environment

Make a package!

<https://r-pkgs.org/whole-game.html>

Beware!

You're probably used to maintaining a .R file containing snippets of code that you use to automate various bits of your workflow.

Don't save this in R/!

What happens if you have `load_all()` inside a file inside of R/? What happens if you have `usethis::edit_r_profile()`?

Where should you save it? 🤔 I use Untitled 😊

Your turn

```
# Create a package with:  
usethis::create_package("~/Desktop/foofactors")
```

Substitute your
preferred location.

```
# Notice that you're now in a new RStudio  
# instance.  
# Continue on through the next slides to  
# repeat the actions I showed you.
```

```
# Stuck? Raise a pink post it
```

Your turn

Use `usethis::use_r("fbind")` to create a new file

In it, define a function named "fbind" that combines its inputs (presumably factors) like so:

- coerce each input to character
- combine inputs
- make output a factor

```
factor(c(as.character(a), as.character(b)))
```

Check that you can `devtools::load_all()`

Your turn

Add docs for fbind() as a roxygen comment

- RStudio helper: *Code > Insert roxygen skeleton*
- Lines MUST start with **#'**

document()

Makes an .Rd file
from the comment

Preview with ?fbind

check() again and ... rejoice!

Problems? Raise a **pink** post-it

Your turn

Edit DESCRIPTION (optional)

- Make yourself the author
- Update Title and Description

Nice to do, but skippable.

```
use_mit_license("Your Name")
```

Fixes remaining warning.

check() again, if you wish

Confused? Hoist your **pink** post-it

Your turn

Install your foofactors package

- Call `install()` in R
- RStudio *Build & Restart*
- Shell: R CMD build + R CMD install

Restart R

Attach like a "regular" package with `library()`

Call `fbind()`

Revel in your success by raising your **green** post-it

```
usethis::create_package()  
usethis::use_r()  
devtools::load_all()  
devtools::check()  
usethis::use_mit_license()  
devtools::document()  
devtools::install()
```

This work is licensed as
Creative Commons
Attribution-ShareAlike 4.0
International

To view a copy of this license, visit
[https://creativecommons.org/
licenses/by-sa/4.0/](https://creativecommons.org/licenses/by-sa/4.0/)