
Building Tidy Tools
Charlotte Wickham &
Hadley Wickham
rstudio::conf
SAN FRANCISCO // JANUARY 27 - 30, 2020
from RStudio

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?

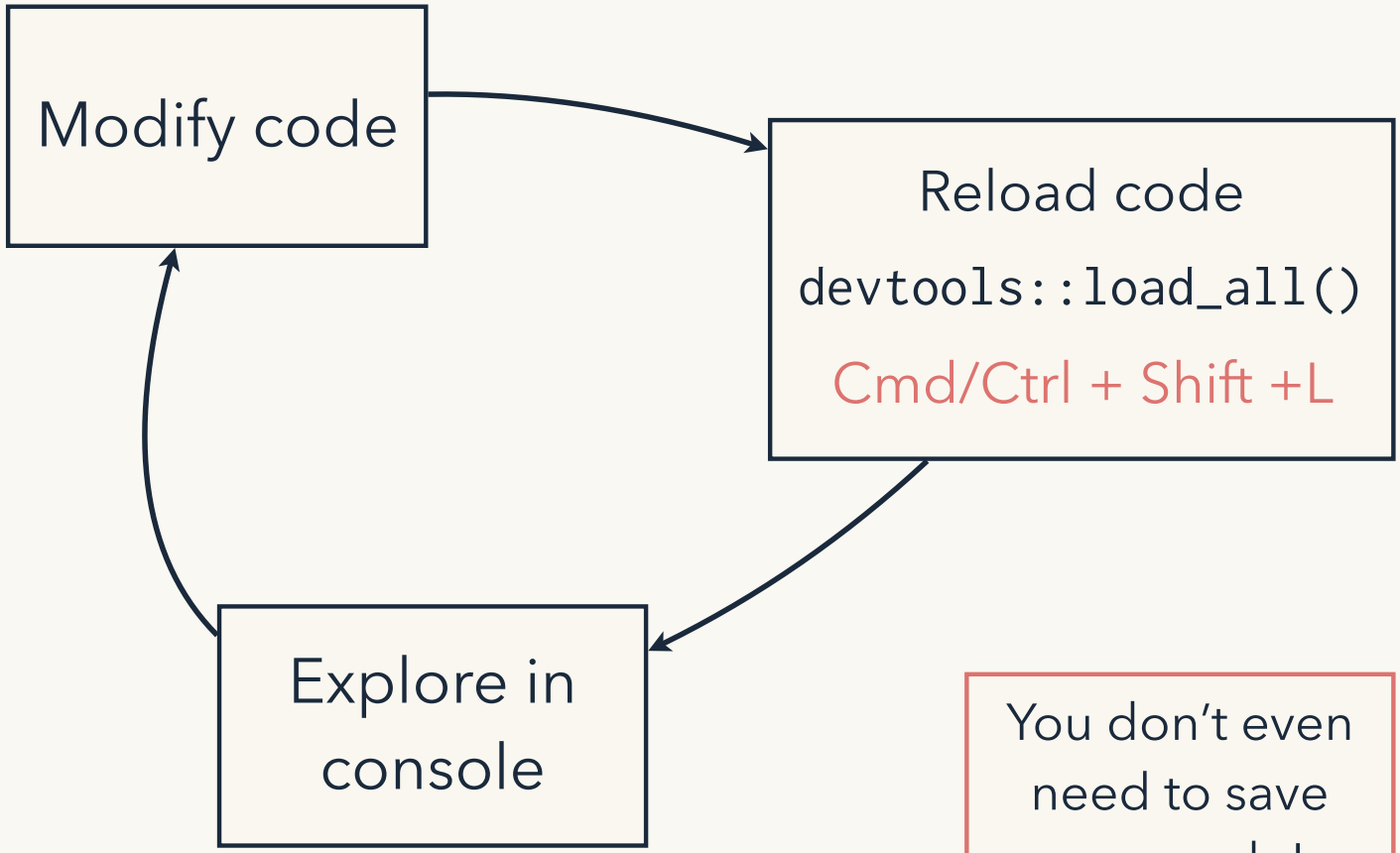
~~`source("R/fbind.R")`~~

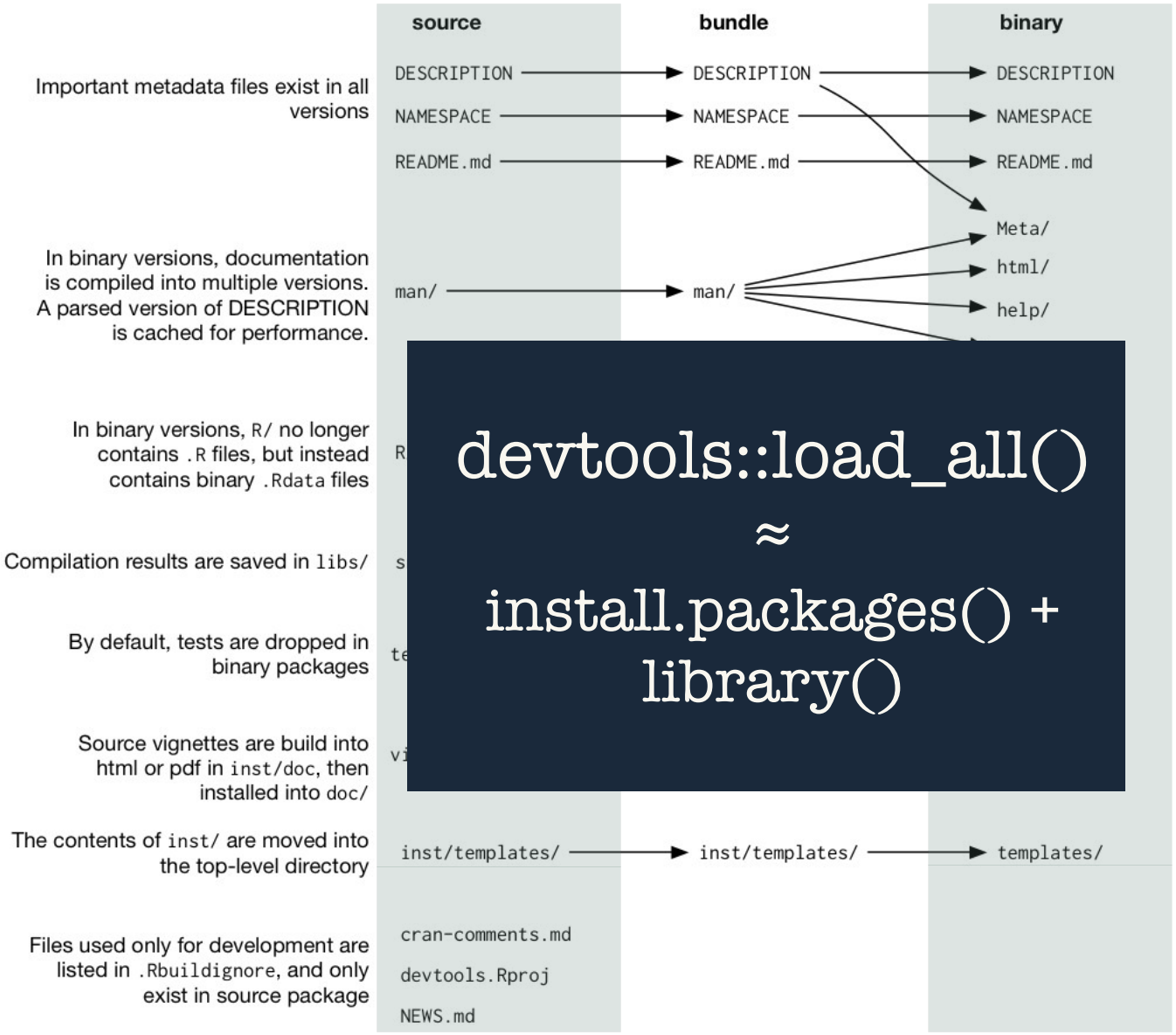
~~Use IDE tricks to send definition of fbind() to the R Console~~

```
devtools::load_all()
```

devtools::load_all()

Why do we love devtools? Workflow!




source
bundle
binary
Important metadata files exist in all versions
DESCRIPTION
NAMESPACE
README.md
In binary versions, documentation is compiled into multiple versions. A parsed version of DESCRIPTION is cached for performance.
man/
Meta/
html/
help/
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
inst/templates/
templates/
Files used only for development are listed in .Rbuildignore, and only exist in source package
cran-comments.md
devtools.Rproj
NEWS.md
devtools::load_all()
≈
install.packages() +
library()

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() ≈ 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