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

Charlotte Wickham

she/her

Part-time instructor at
Oregon State University

@cvwickham
cwickham@gmail.com

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!

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

Ingrid Rodriguez

she/her

RStudio

https://www.linkedin.com/in/rodrin/

Julia Blum

she/her

@jcblum on RStudio Community

François Michonneau

he/him

The Carpentries

@fmic_

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

Outline for **today**

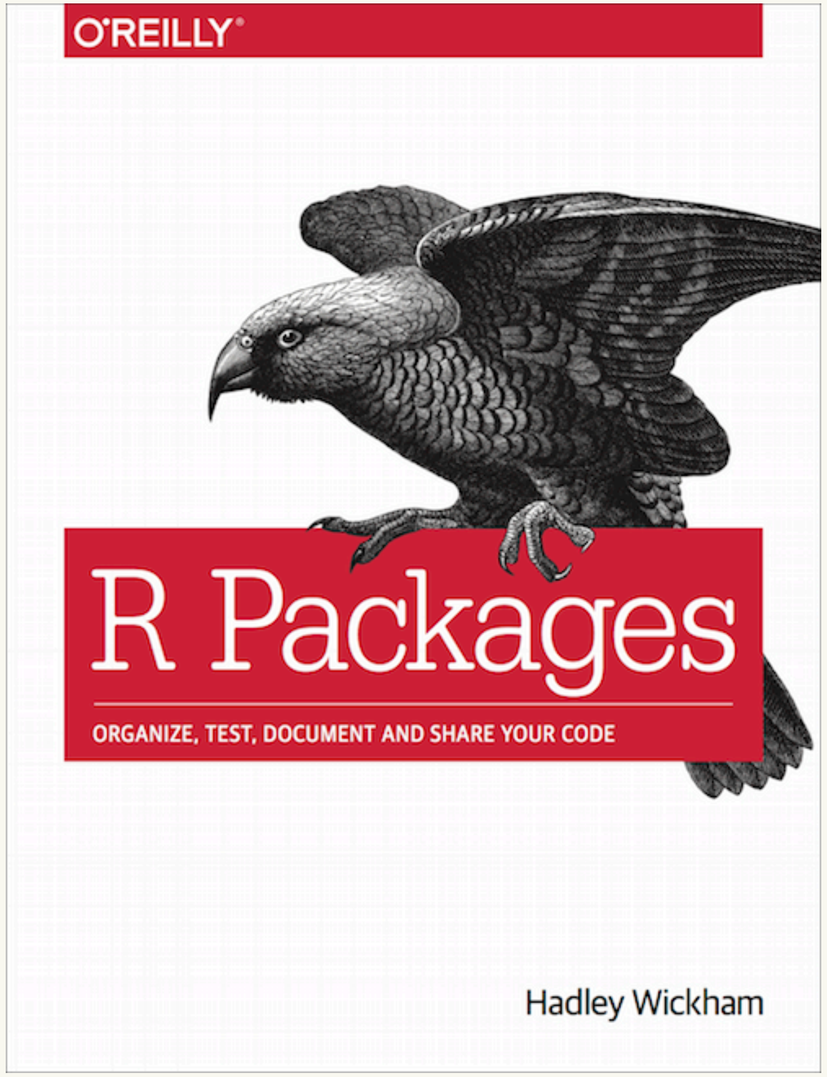
9:00-10:30	Introduction / "The Whole Game"	Charlotte
11:00-12:30	Testing	Charlotte
13:30-15:00	Documentation / Sharing	Charlotte
15:30-17:00	Dependencies	Hadley

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

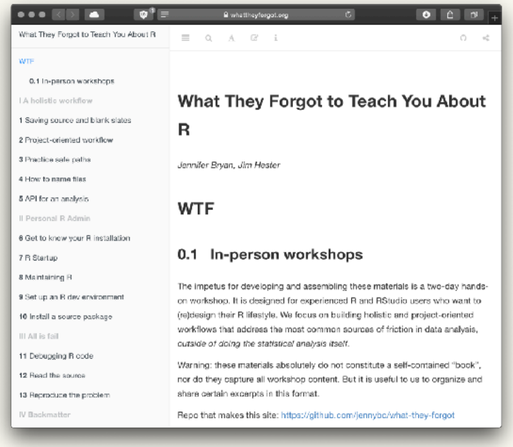
Outline for **tomorrow**

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

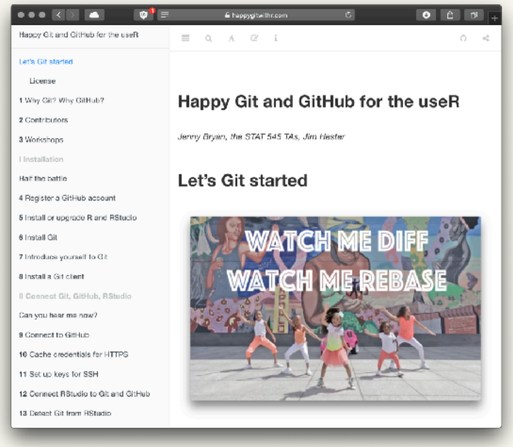
Materials: http://rstd.io/build-tt



https://r-pkgs.org/



https://rstats.wtf



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

Sticky notes

I'm stuck,
send help

I'm done,
let's move on

Your turn 

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

# Green sticky when you and your neighbour(s)
# have one answer for each.
```

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()
)
```

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

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?

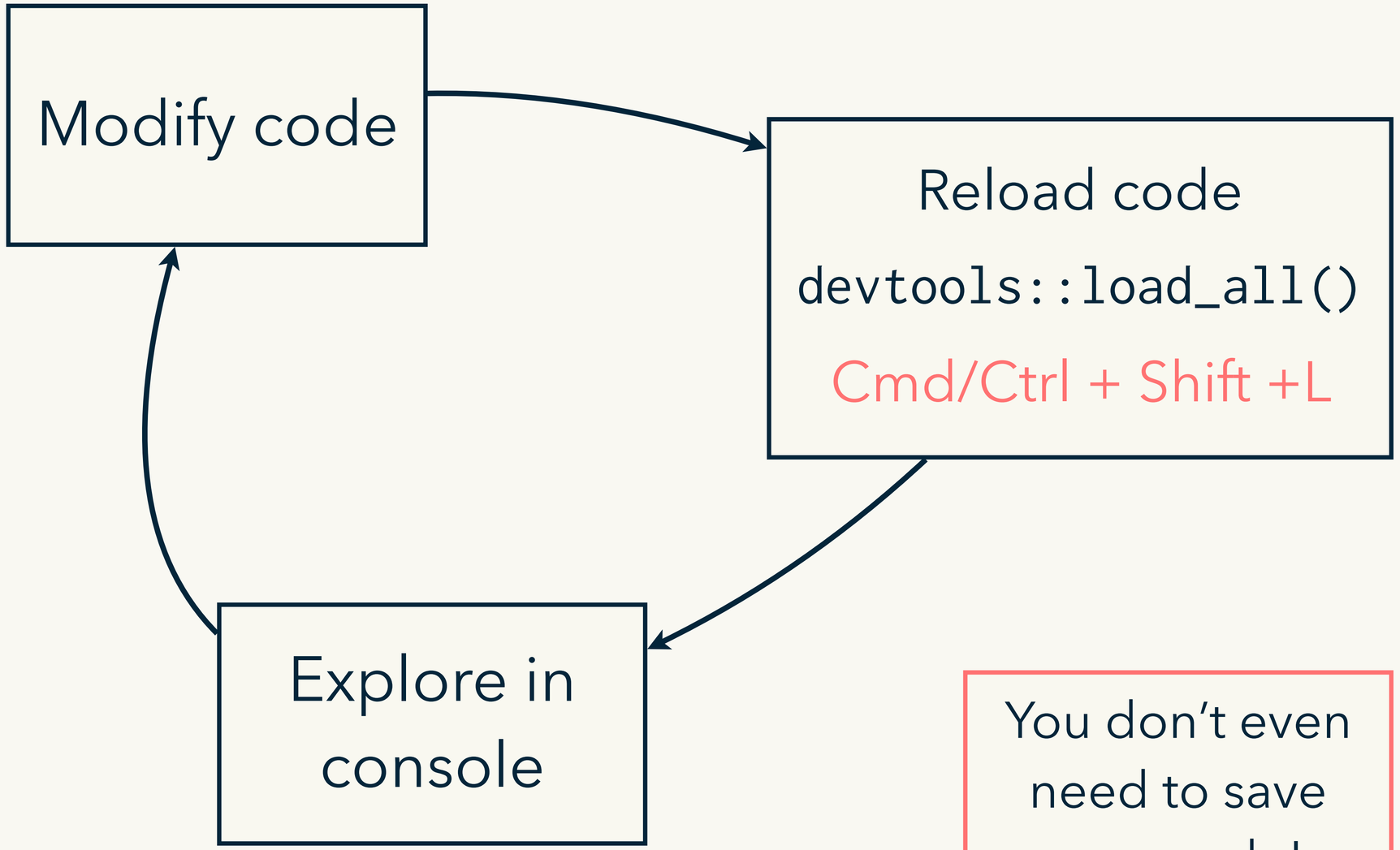
~~`source("R/fbind.R")`~~

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

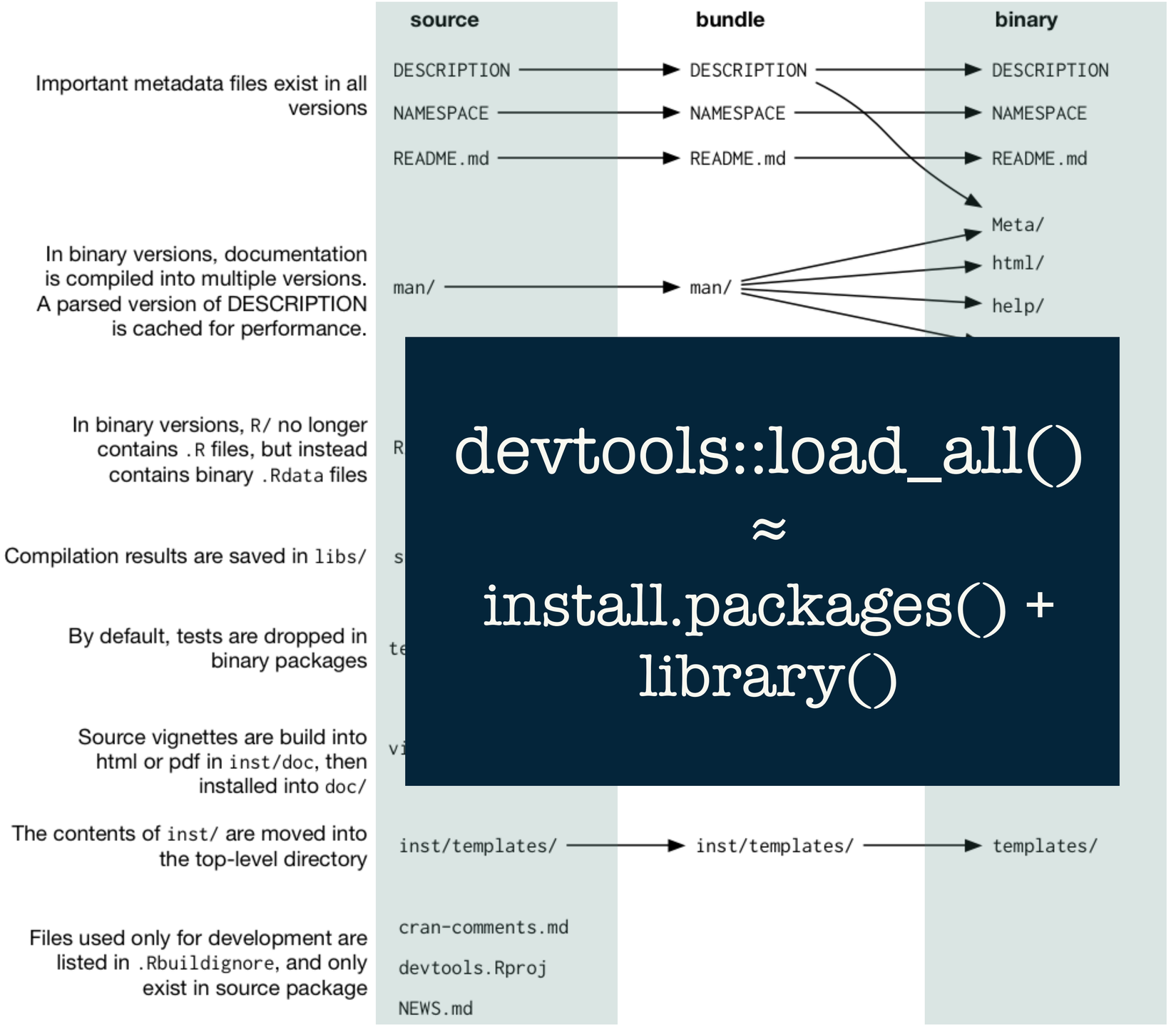
`devtools::load_all()`

devtools::load_all()

Why do we love devtools? Workflow!



You don’t even need to save your code!


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

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:

```
# 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",
       email = "hadley@rstudio.com",
       role = c("aut", "cre"),
       comment = c(ORCID = "YOUR-ORCID-ID")
    )'
  )
)
```

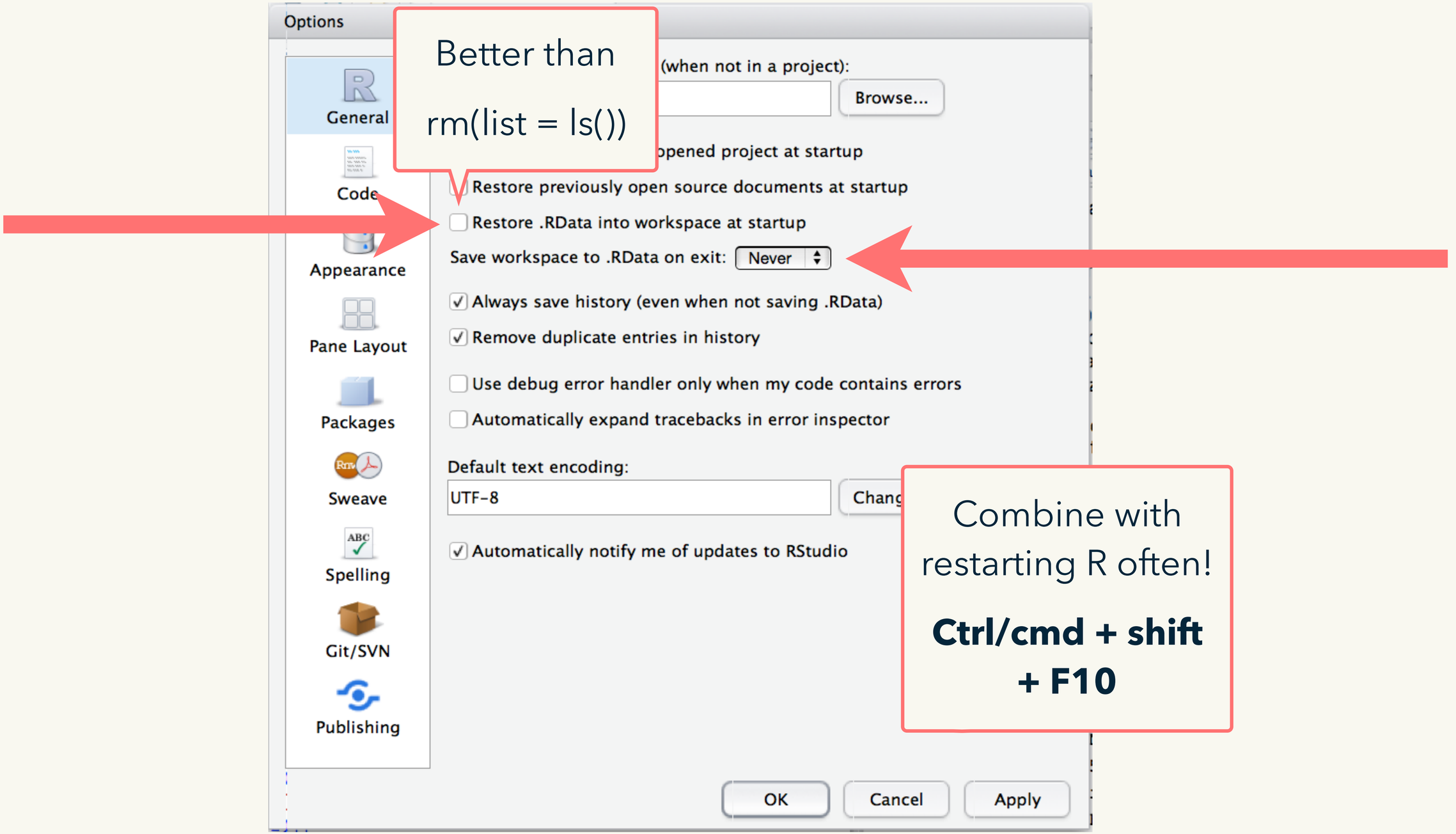
Your first and last names

Your email

Leave as is

Delete this line if you don't have an ORCID

https://usethis.r-lib.org/articles/articles/usethis-setup.html


Options
General
Code
Appearance
Pane Layout
Packages
Sweave
Spelling
Git/SVN
Publishing
(when not in a project):
Browse...
opened project at startup
Restore previously open source documents at startup
Restore .RData into workspace at startup
Save workspace to .RData on exit:
Never
Always save history (even when not saving .RData)
Remove duplicate entries in history
Use debug error handler only when my code contains errors
Automatically expand tracebacks in error inspector
Default text encoding:
UTF-8
Chang
Automatically notify me of updates to RStudio
OK
Cancel
Apply
Better than
rm(list = ls())
Combine with restarting R often!
Ctrl/cmd + shift + F10

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

Substitute your preferred location.

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

# 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()
```