Building Tidy Tools

@ Roche

September 2020

Charlotte Wickham & Sara Altman

Welcome!

Building Tidy Tools

Charlotte Wickham and Sara Altman

```
# We'll get started at 8:35am

# Materials on GitHub: http://bit.ly/build-tt
# 0. Skim the README
# 1. Check you've completed
#    the Setup section
# 2. Get these slides from the
#    Schedule section
# Questions? Ask in Slido
```


Join at
slido.com
#80875
Passcode: **tidytools**

Charlotte Wickham

she/her

Part-time instructor at
Oregon State University

@cvwickham
cwickham@gmail.com

Sara Altman

Your turn: Getting to know you

How much experience do you have building R packages?

This means that you have to work!

Your turn: Breakout Rooms

This course is hands-on and, while we're here to help, there are many of you, and only two of us.

For some activities, we'll put you in breakout rooms, so you can help each other.

Instructions

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

2. Discuss as a group: How do you install R packages? Add your answers in slido.

3. Once you've got an answer for #2, indicate you are *ready to move on* in the Google Doc: bit.ly/built-tt-breakout

Take a note of the slide number

Getting Help

slido: Ask a question in the Q&A at anytime.
Also vote on other peoples' questions.

In Breakout Rooms:

1. Ask your roommates

2. If your room is stuck, change your status to "Send Help" in Google Doc

Zoom chat: Reserved for urgent technical matters (e.g. "we can't hear you")

Workflow

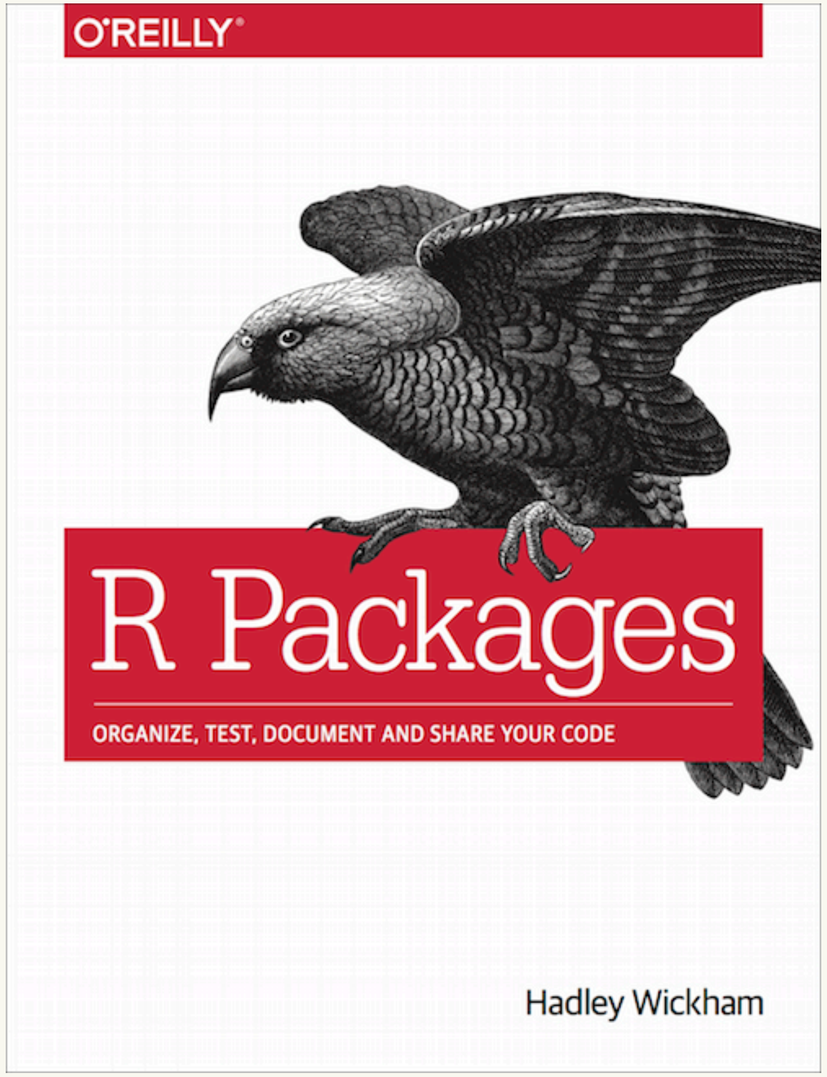
Interface

Implementation

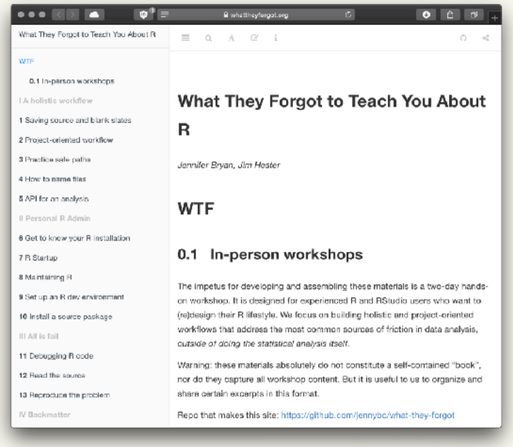
Outline for **today:** Workflow

8:30-10:00	Introduction / "The Whole Game"	Charlotte
10:00-10:30	Break	
10:30-12:00	Testing	Sara

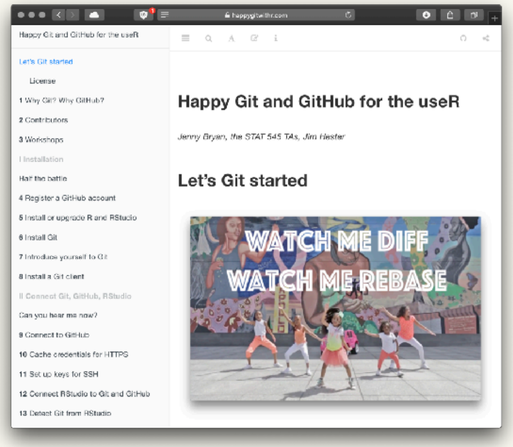
Materials: http://bit.ly/build-tt



https://r-pkgs.org/



https://rstats.wtf



https://happygitwithr.com

Materials: http://bit.ly/build-tt

The whole game

Materials: http://bit.ly/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/wickhamc/Desktop/foofactors/'
✓ Setting active project to '/Users/wickhamc/Desktop/foofactors'
✓ Creating 'R/'
✓ Writing 'DESCRIPTION'
Package: foofactors
Title: What the Package Does (One Line, Title Case)
Version: 0.0.0.9000
Authors@R (parsed):
    * First Last <first.last@example.com> [aut, cre] (YOUR-ORCID-ID)
Description: What the package does (one paragraph).
License: `use_mit_license()`, `use_gpl3_license()` or friends to
    pick a license
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.1.1
✓ Writing 'NAMESPACE'
✓ Writing 'foofactors.Rproj'
✓ Adding '.Rproj.user' to '.gitignore'
✓ Adding '^foofactors\\.Rproj$', '^\\.Rproj\\.user$' to '.Rbuildignore'
✓ Opening '/Users/wickhamc/Desktop/foofactors/' in new RStudio session
✓ Setting active project to '<no active project>'
```

An R package is just a set of conventions

foofactors/

R/

Functions go in .R files in here

DESCRIPTION

General package info

... some other stuff we won't worry about (yet)

usethis::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()`

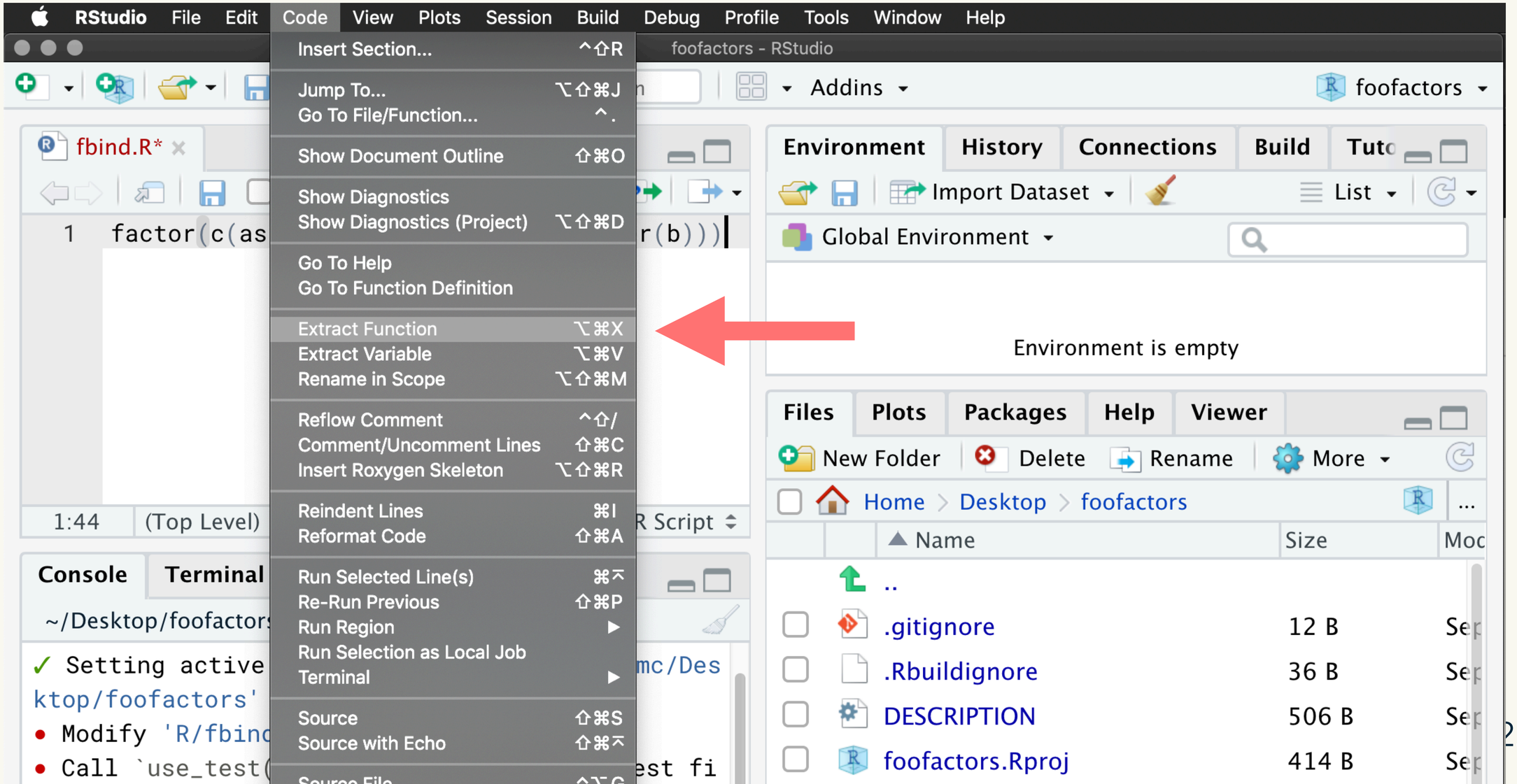
usethis::use_r()

Where do we define functions?

File name

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

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

How do we write fbind()?



Now what?

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

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

Now what?

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

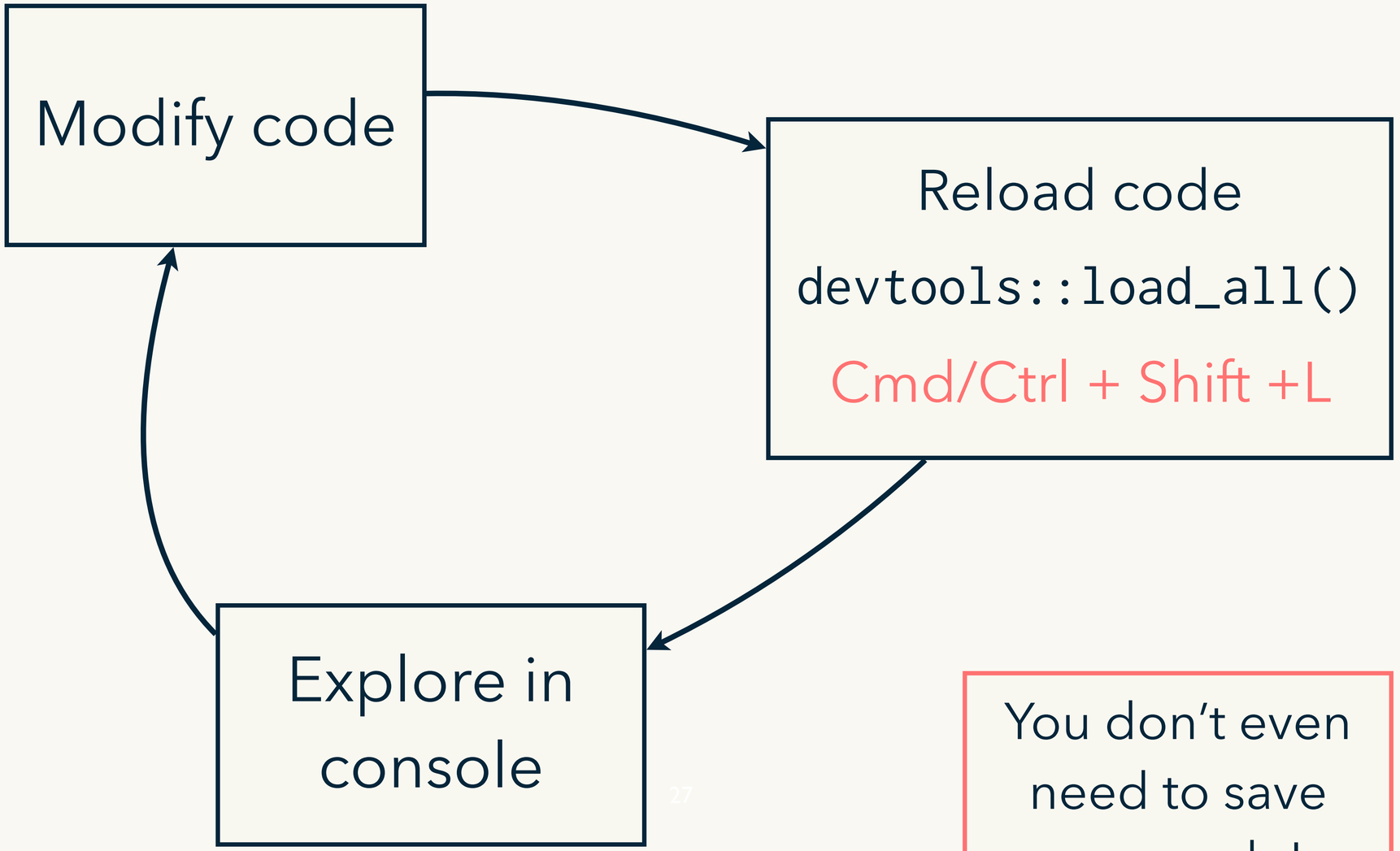
~~Use IDE tricks to send definition of fbind() to the R Console~~

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

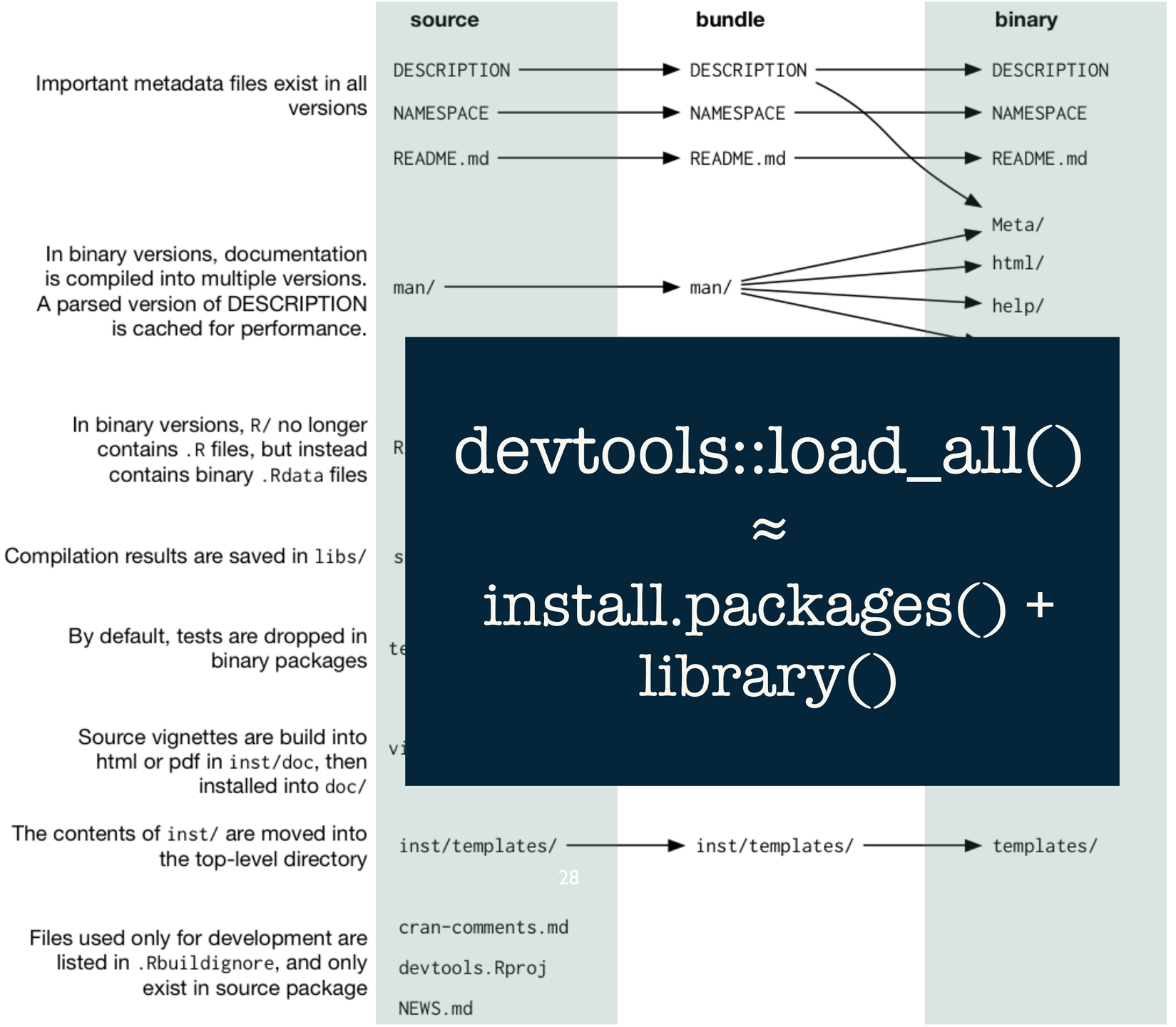
devtools::load_all()

```
# Restart R

devtools::load_all()

a <- factor(c("character", "in", "the", "streets"))
b <- factor(c("integer", "in", "the", "sheets"))

fbind(a, b)
#> [1] character in        the       streets
#> [5] integer   in        the       sheets
#> 6 Levels: character in integer ... the
```

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/
devtools::load_all()
≈
install.packages() + library()
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

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: Breakout Rooms

Follow the steps on the **following slides** to:

1. Create the foofactors package
2. Add the fbind() function to the package
3. Practice the workflow with devtools::load_all()
4. Let me know you are done

10mins

Your turn: 1. Create a package

Or another location that works for you

```
usethis::create_package("~/Desktop/foofactors")
# usethis will create the necessary folders and files
# then open a new session in the foofactors project
```

Your turn: 2. Add a function

```
usethis::use_r("fbind")
# usethis will create the file fbind.R and open it
# Copy and paste the code snippet:
factor(c(as.character(a), as.character(b)))
# Then turn it into a function.
# RStudio helper: Code -> Extract function
```

Your turn: 3. Practice the workflow

```
# Restart R


# Load the package functions
devtools::load_all()


# Test our example
a <- factor(c("character", "in", "the", "streets"))
b <- factor(c("integer", "in", "the", "sheets"))
fbind(a, b)
# If you get the right output, you are done!
```

Your turn: 4. Ready to move on?

```
# Check in with your breakout room.
# Is everyone ready to move on?
# If no, help them out!
# If yes, edit your status in the Google Doc:
# bit.ly/built-tt-breakout

# While you wait, try running:
devtools::check()

# What happens?
```

Questions?

devtools::check()

check() output...

```
Updating foofactors documentation
Loading foofactors
── Building ──────────────────────────── foofactors ──
Setting env vars:
• CFLAGS    : -Wall -pedantic -fdiagnostics-color=always
• CXXFLAGS  : -Wall -pedantic -fdiagnostics-color=always
• CXX11FLAGS: -Wall -pedantic -fdiagnostics-color=always
────────────────────────────────────────────────────────
✓  checking for file ‘/Users/wickhamc/Desktop/foofactors/DESCRIPTION’ ...
─  preparing ‘foofactors’:
✓  checking DESCRIPTION meta-information ...
─  checking for LF line-endings in source and make files and shell scripts
─  checking for empty or unneeded directories
   Removed empty directory ‘foofactors/man’
─  building ‘foofactors_0.0.0.9000.tar.gz’

── Checking ──────────────────────────── foofactors ──
Setting env vars:
• _R_CHECK_CRAN_INCOMING_USE_ASPELL_: TRUE
• _R_CHECK_CRAN_INCOMING_REMOTE_    : FALSE
• _R_CHECK_CRAN_INCOMING_           : FALSE
• _R_CHECK_FORCE_SUGGESTS_          : FALSE
• NOT_CRAN                          : true
── R CMD check ───────────────────────────────────────────
─  using log directory ‘/private/var/folders/d0/gkc929kn4cn61pz_chjl6hn00000gn/T/Rtmp0ifEJD/foofactors.Rcheck’
─  using R version 4.0.2 (2020-06-22)
─  using platform: x86_64-apple-darwin17.0 (64-bit)
```

check() output...

```
...
── R CMD check results ──────────────────── foofactors 0.0.0.9000 ────
Duration: 9.3s

> checking DESCRIPTION meta-information ... WARNING
  Non-standard license specification:
    `use_mit_license()`, `use_gpl3_license()` or friends to pick a
    license
  Standardizable: FALSE

> checking for future file timestamps ... NOTE
  unable to verify current time

0 errors ✓ | 1 warning x | 1 note x
```

```
# Fix the warning with, e.g.:
usethis::use_mit_license("Charlotte Wickham")
```

`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: Breakout Rooms

Follow the steps on the **following slides** to:

1. Remove the warning from check()
2. Add documentation to the fbind()
3. Install the package with install()

Same process: ask for help, or indicate your room is done in Google Doc

Your turn: 1. Remove the warning

Your name

```
# Fix the warning with, e.g.:
usethis::use_mit_license("Charlotte Wickham")

# Verify the warning is fixed
devtools::check()
```

Your turn: 2. Document fbind

```
# Put your cursor inside fbind() in fbind.R
# Then add a Roxygen skeleton
# RStudio helper: Code -> Insert Roxygen Skeleton

# Edit the skeleton to include a title, and short
# description for the arguments and return value

# Build the documentation
devtools::document()

# Verify the documentation
?fbind

# Check no errors, no warnings, and no notes
devtools::check()
```

Your turn: 3. Install

```
# Install the package
devtools::install()
# Restart R

# Load the package
library(foofactors)

# Test our example
a <- factor(c("character", "in", "the", "streets"))
b <- factor(c("integer", "in", "the", "sheets"))
fbind(a, b)
# If you get the right output, you are done!
```

Your turn: Ready to move on?

```
# Check in with your breakout room.
# Is everyone ready to move on?
# If no, help them out!
# If yes, edit your status in the Google Doc:
# bit.ly/built-tt-breakout

# While you wait, open DESCRIPTION
# and edit the title, author and description.
# Check again:
devtools::check()
```

Questions?

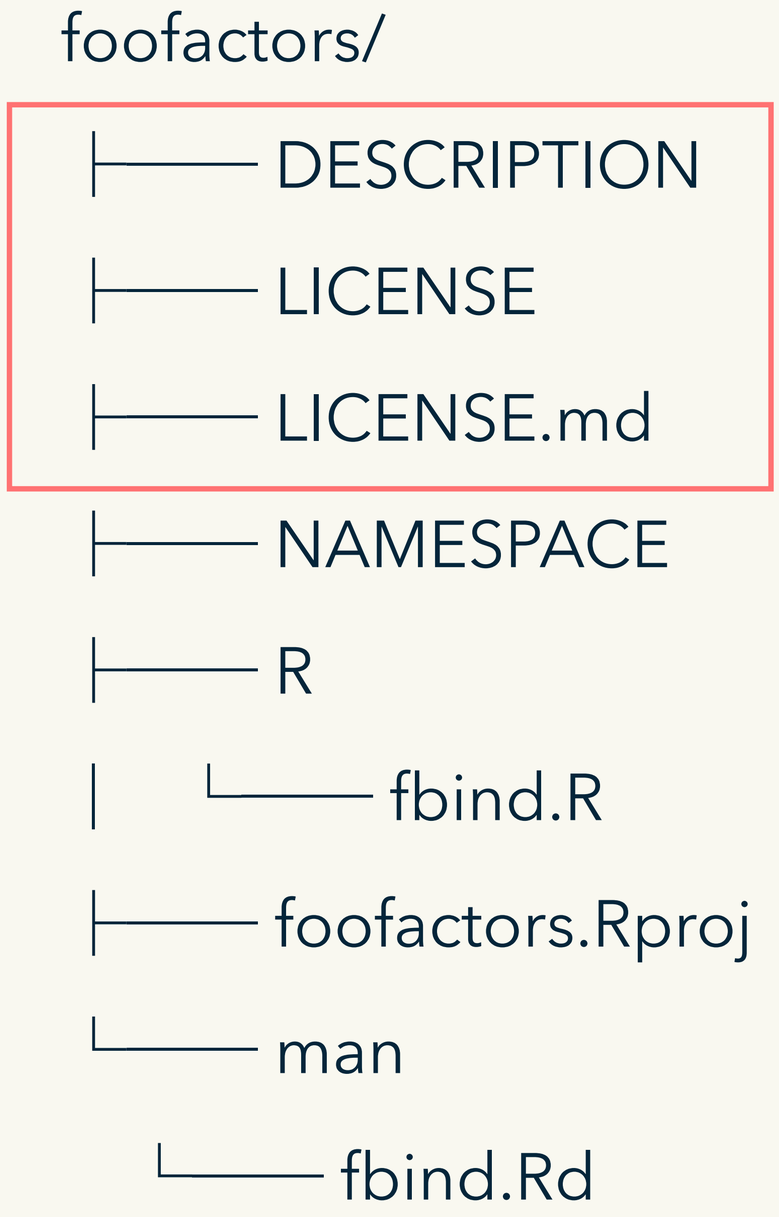
```
foofactors/
├── DESCRIPTION
├── LICENSE
├── LICENSE.md
├── NAMESPACE
├── R
│   └── fbind.R
├── foofactors.Rproj
└── man
    └── fbind.Rd
```

usethis::create_package()

```
foofactors/
├── DESCRIPTION
├── LICENSE
├── LICENSE.md
├── NAMESPACE
├── R
│   └── fbind.R
├── foofactors.Rproj
└── man
    └── fbind.Rd
```

```
usethis::create_package()
```

```
usethis::use_r()
```




```
usethis::create_package()

usethis::use_r()

usethis::use_mit_license()
```

```
foofactors/
├── DESCRIPTION
├── LICENSE
├── LICENSE.md
├── NAMESPACE
├── R
│   └── fbind.R
├── foofactors.Rproj
└── man
    └── fbind.Rd
```

```
usethis::create_package()

usethis::use_r()

usethis::use_mit_license()

devtools::document()
```

Source all files in R/ for testing on the Console

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

Check package for problems

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

Install package

for use like

any other package on

your system

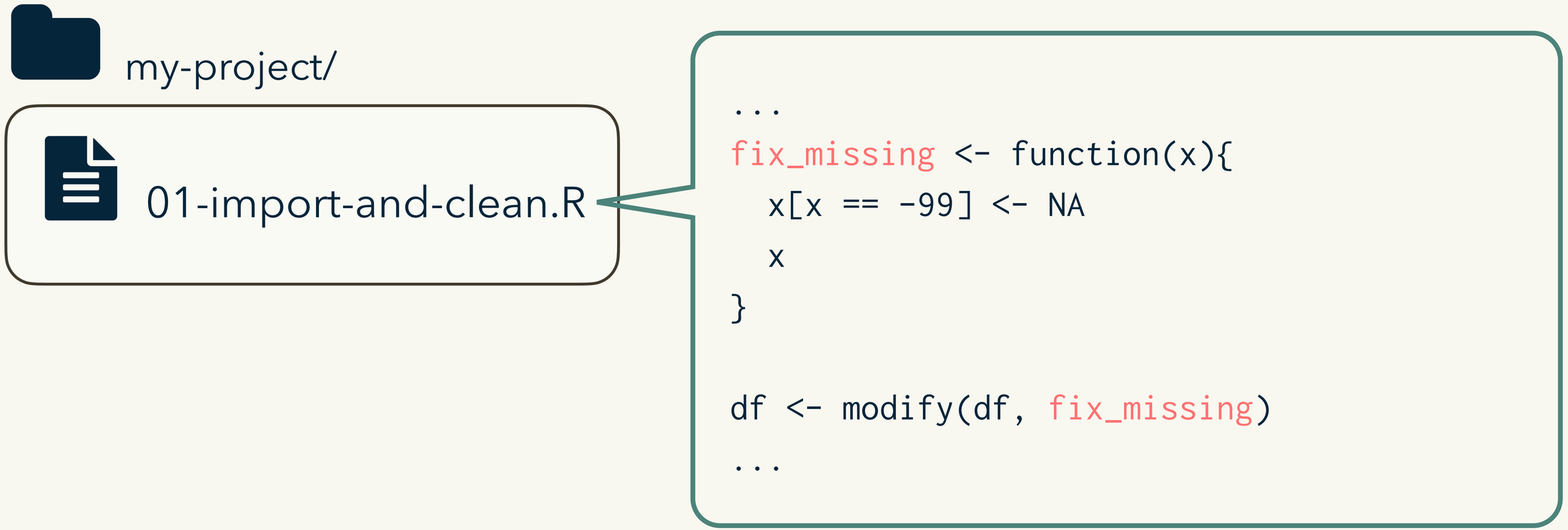
```
usethis::create_package()
usethis::use_r()
usethis::use_mit_license()
devtools::document()
devtools::load_all()
devtools::check()
devtools::install()
```

Questions?

When should I write a package?

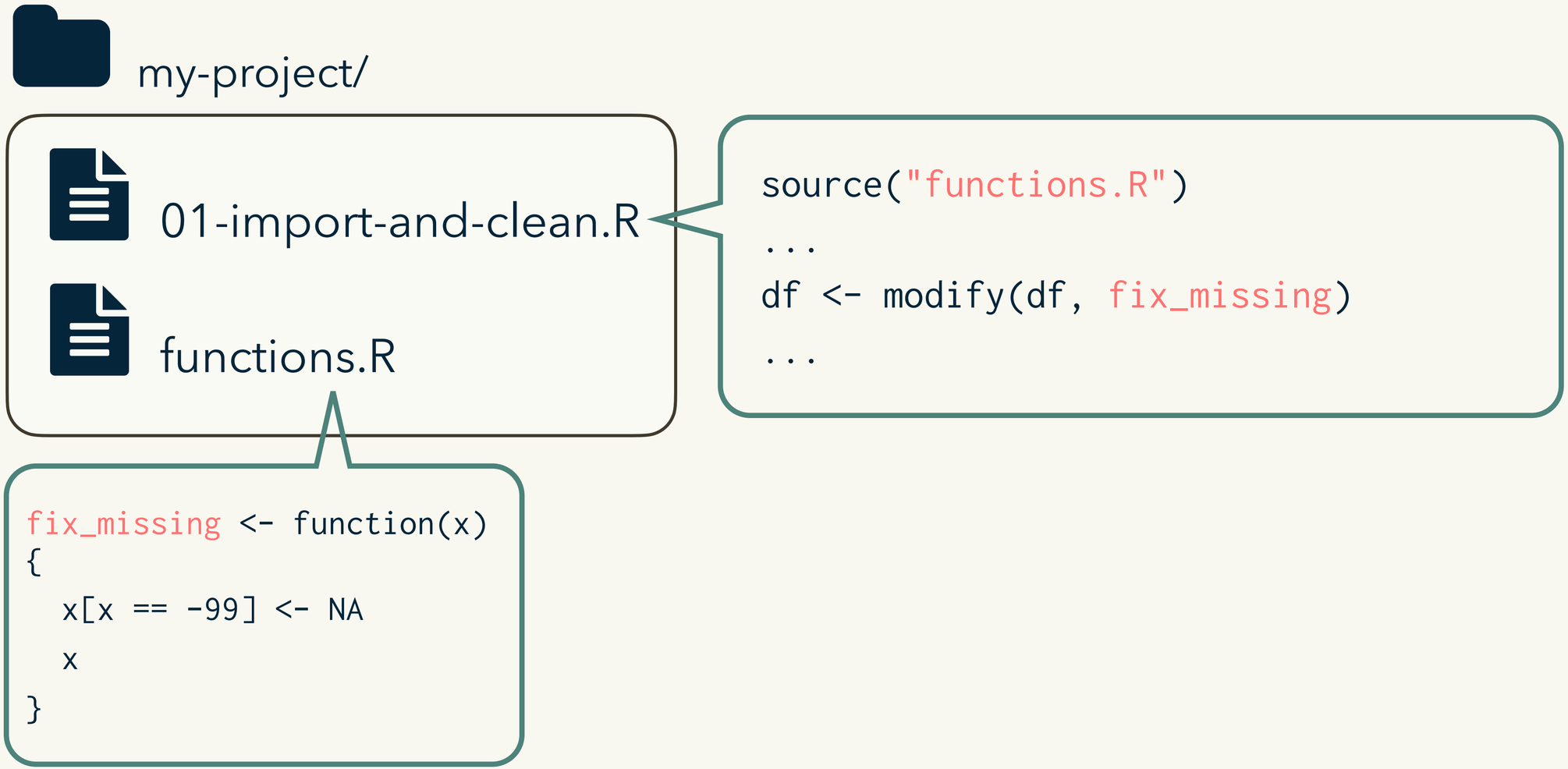
A starting workflow

Functions are defined where they are used



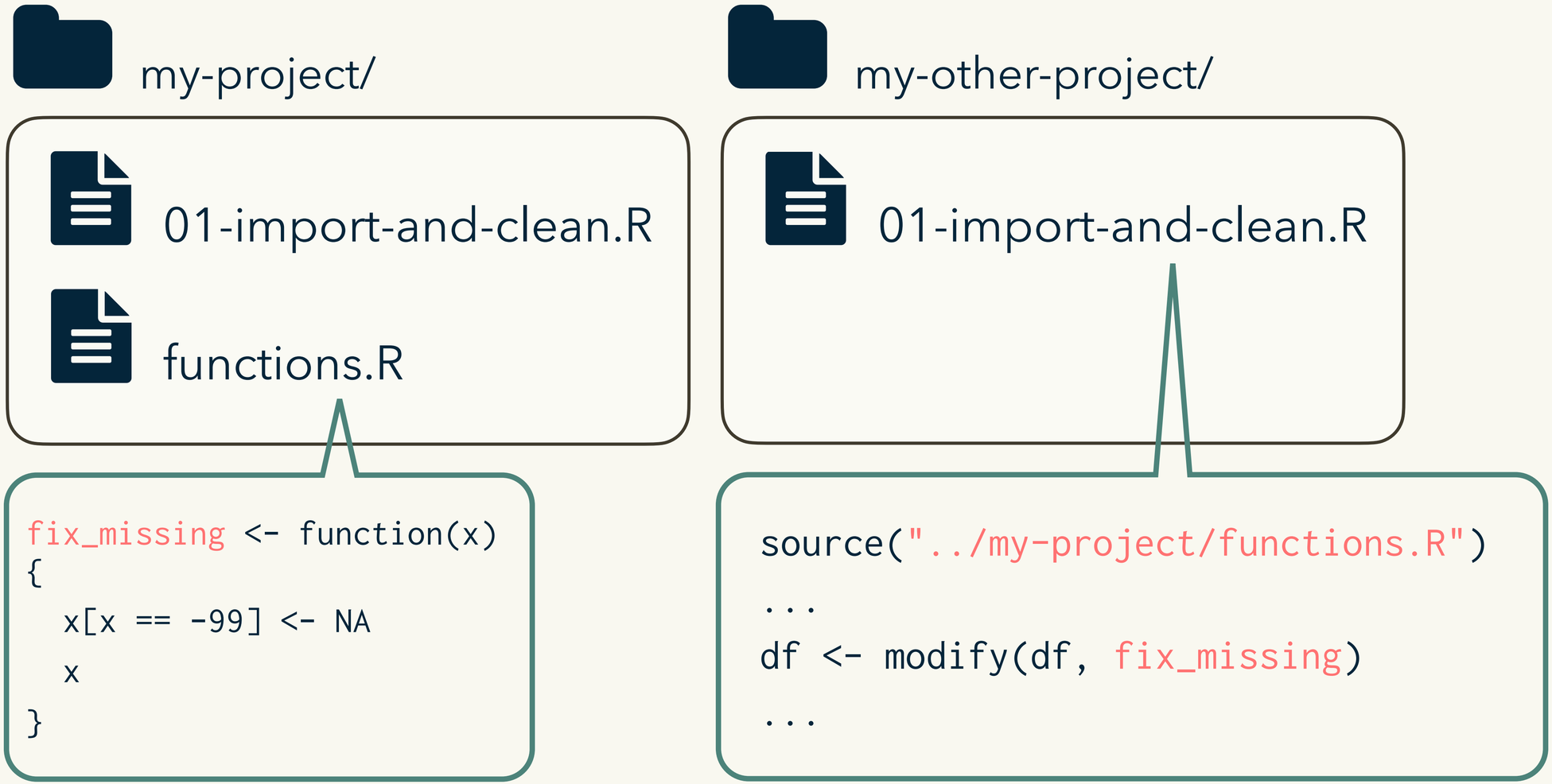
As you use functions in many places in **one** project

Functions are defined in one place



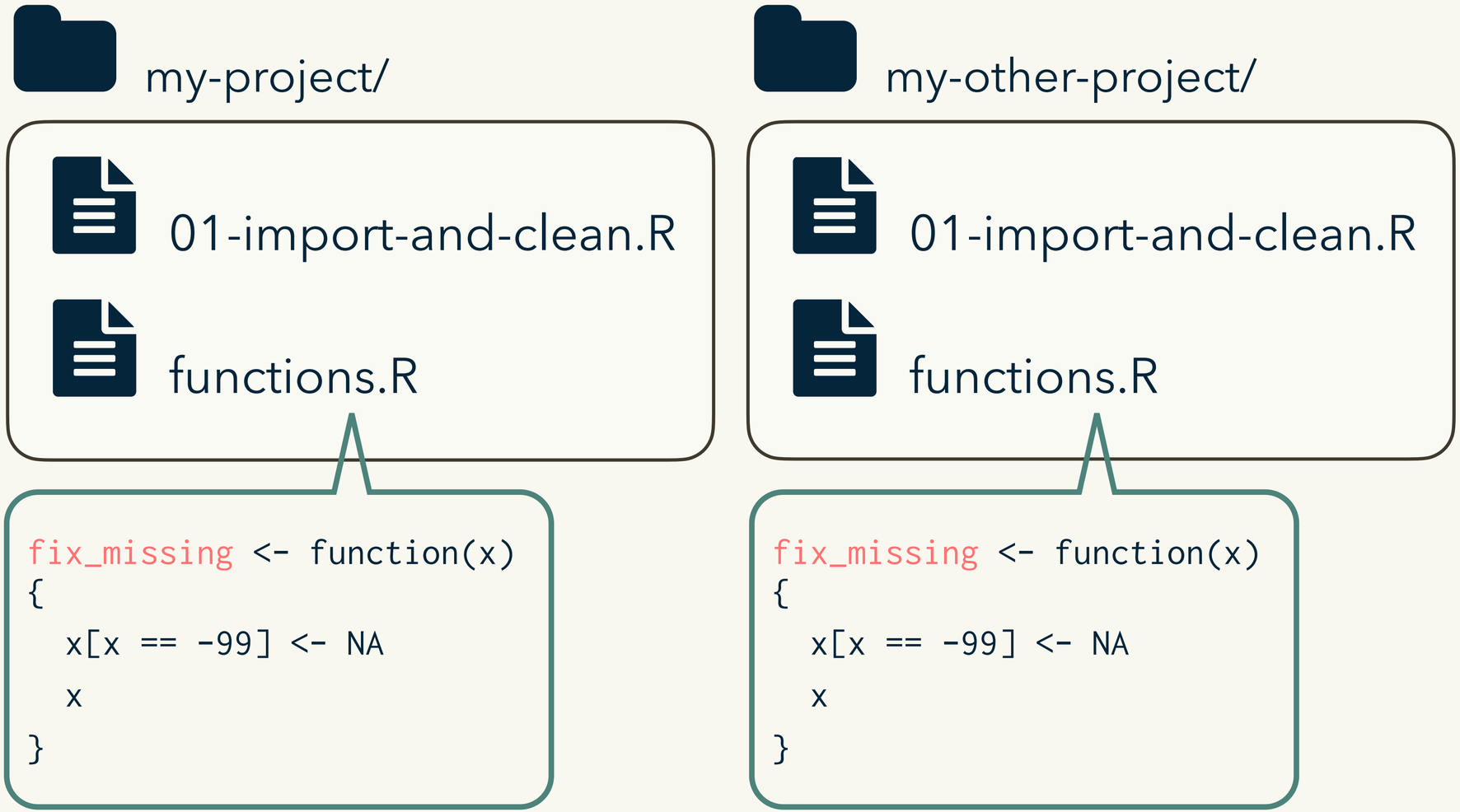
As you use functions in many projects

Sub-optimal Functions live in **one** project



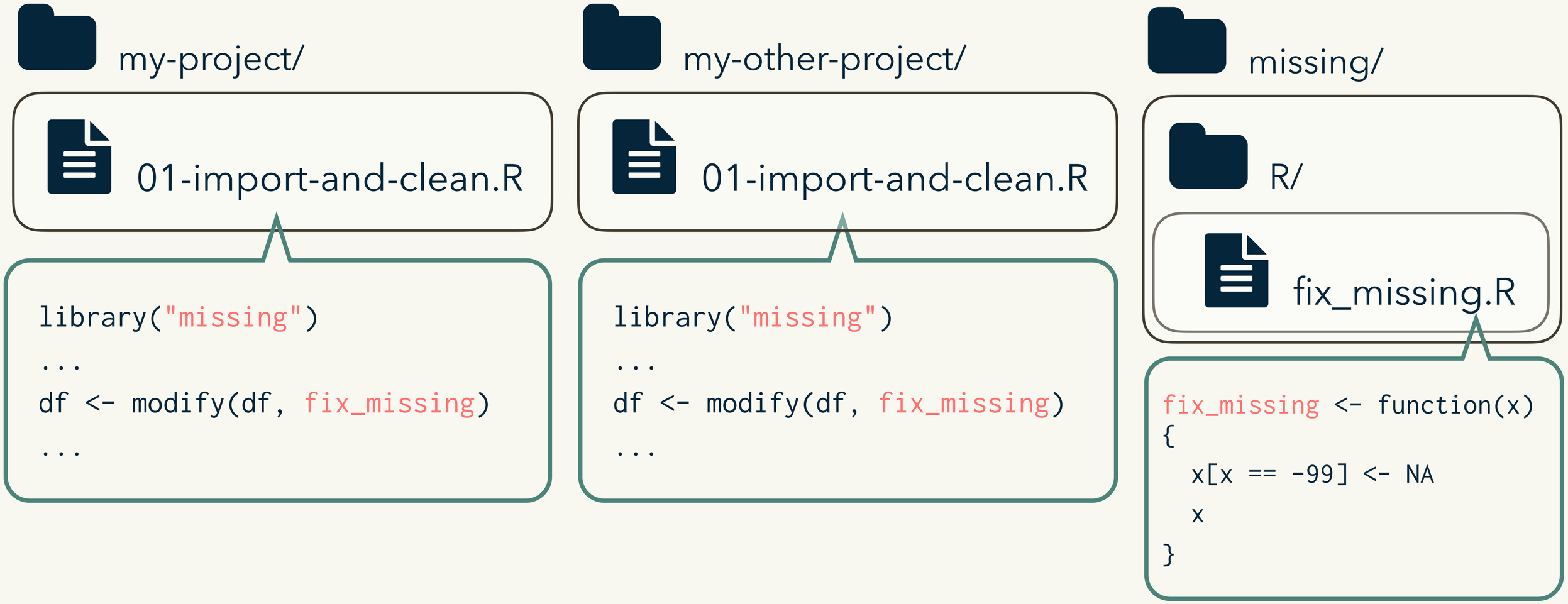
As you use functions in many projects

Sub-optimal Functions live in **both** projects



As you use functions in many projects

Optimal Functions live in a **package**



Principle:

As soon as you need the same function over more than one project, turn it into a package.

R/RStudio setup

It gets painful to type devtools::

```
# Helper to add devtools specifically:
# usethis::use_devtools()

# Opens .Rprofile, run at startup
# Encourages you to add:
if (interactive()) {
  suppressMessages(require(devtools))
}
```

devtools makes
usethis available too!

Never include analysis packages here

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

You could also tell usethis about yourself

```
options(
  usethis.full_name = "Charlotte Wickham",
  usethis.description = list(
    `Authors@R` = 'person(
       "Charlotte", "Wickham",
       email = "cwickham@gmail.com",
       role = c("aut", "cre"),
       comment = c(ORCID = "0000-0002-6365-5499")
    )'
  )
)
```

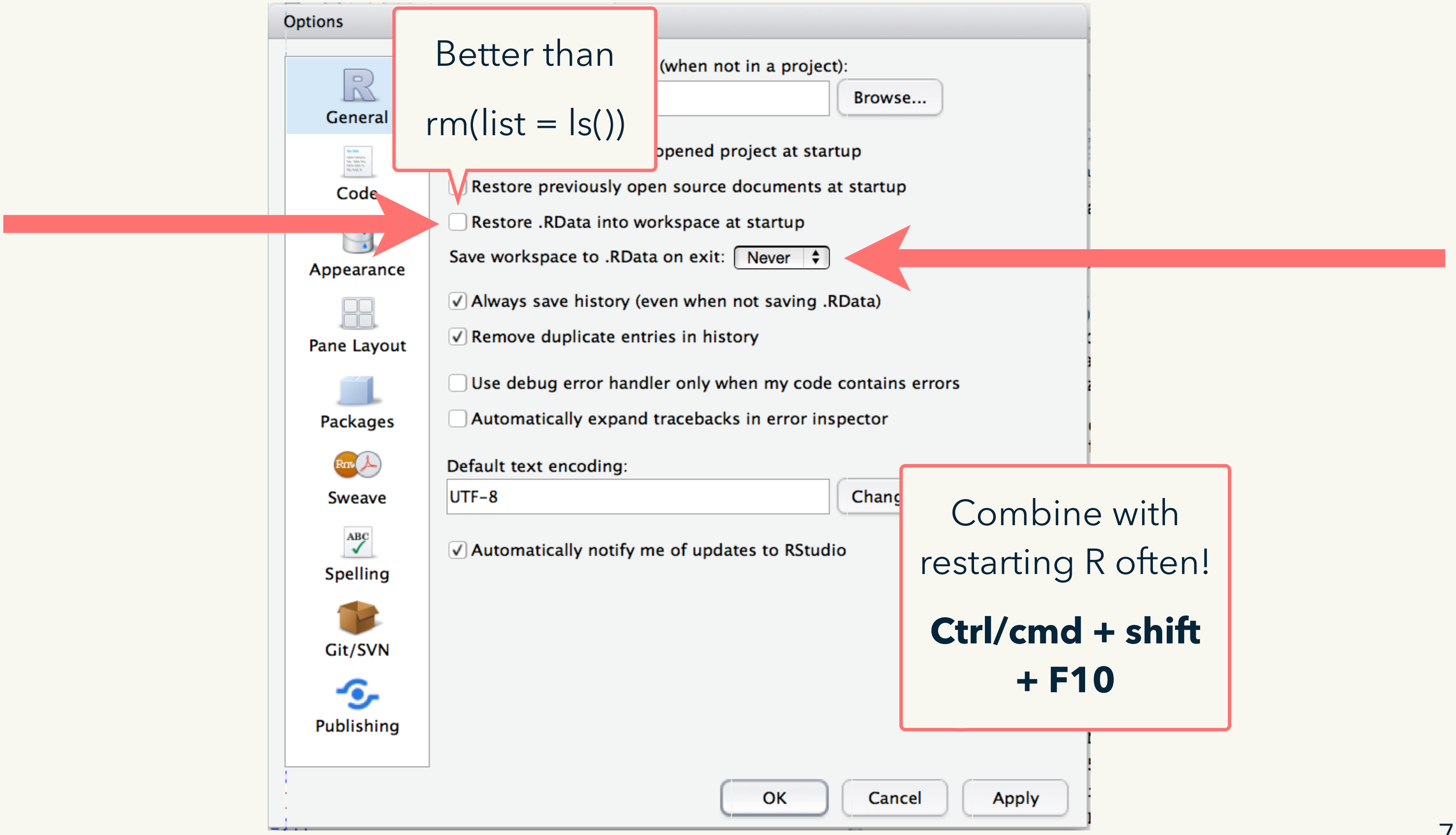
Your first and last names

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

Questions?