Packages, projects, and code

April 2019

Hadley Wickham

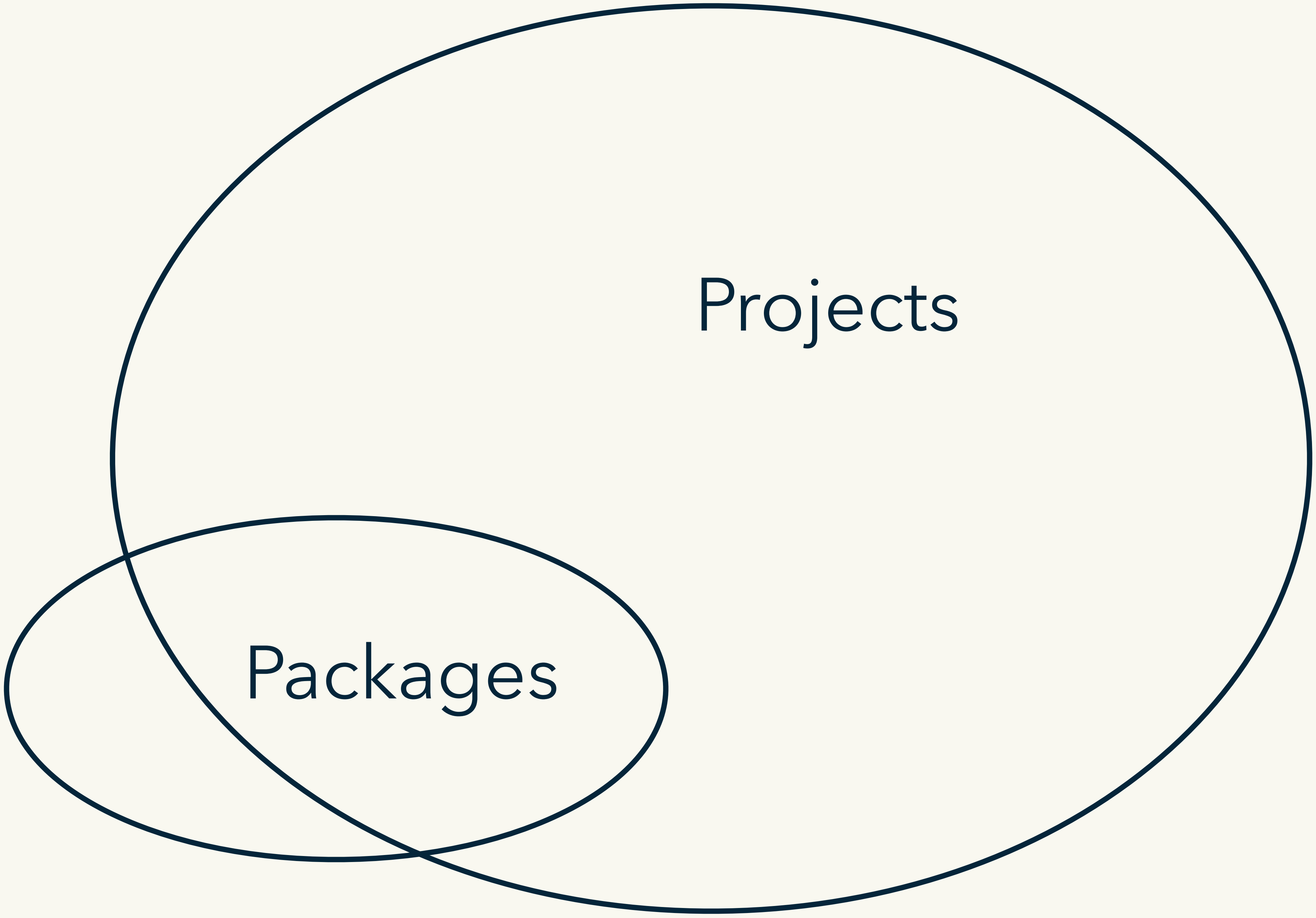
@hadleywickham

Motivation

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

> “Seriously, it doesn’t have to be about sharing your code (although that is an added benefit!). It is about saving yourself time.”
>
> — Hilary Parker

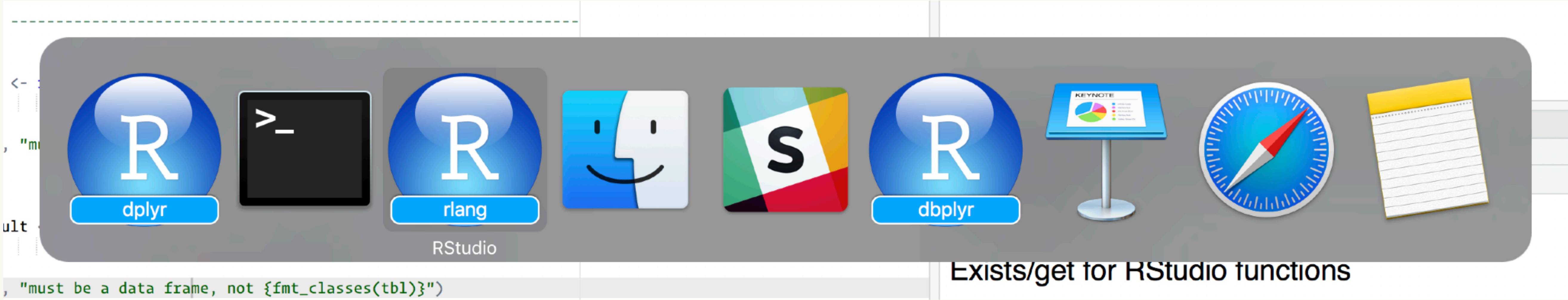
RStudio projects


Projects
Packages

Why use an RStudio project?

3 reasons

Multiple RStudio sessions



```
, "must be a data frame, not {fmt_classes(tbl)}")
```

Exists/get for RStudio functions

Work on multiple projects simultaneously and independently

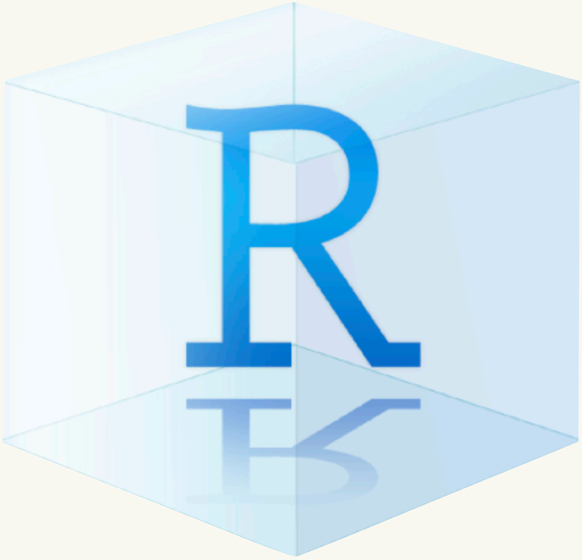
Manage working directories

> If the first line of your R script is
>
> `setwd(”C:\path\that\only\I\have”)`
>
> I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.
>
> – *Jenny Bryan*

https://www.tidyverse.org/articles/2017/12/workflow-vs-script/

INSTRUCTIONS
BC
KIDDE

Manage working directories

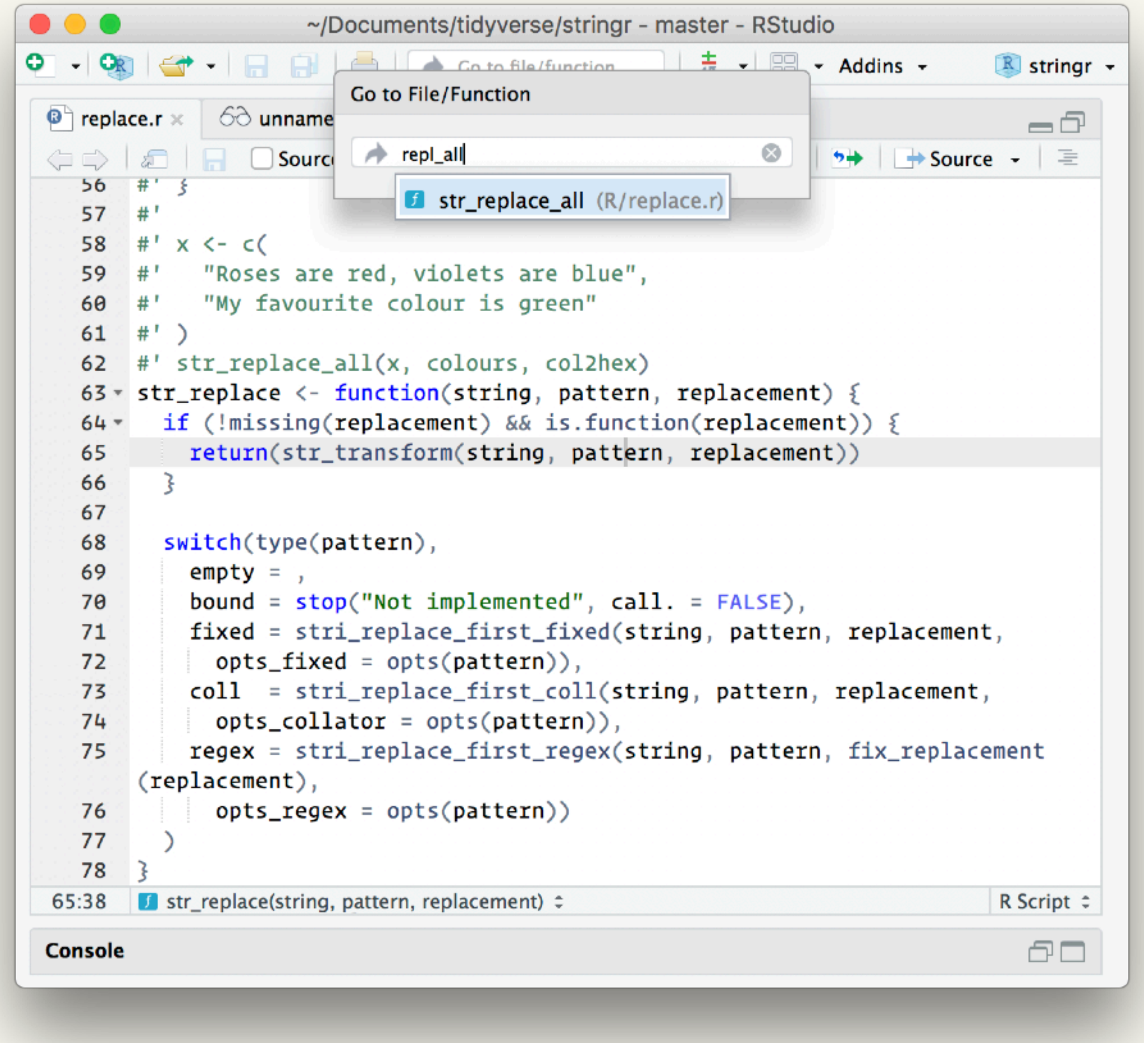


Double click to open project

`mylittlepony.Rproj`

Enhanced navigation

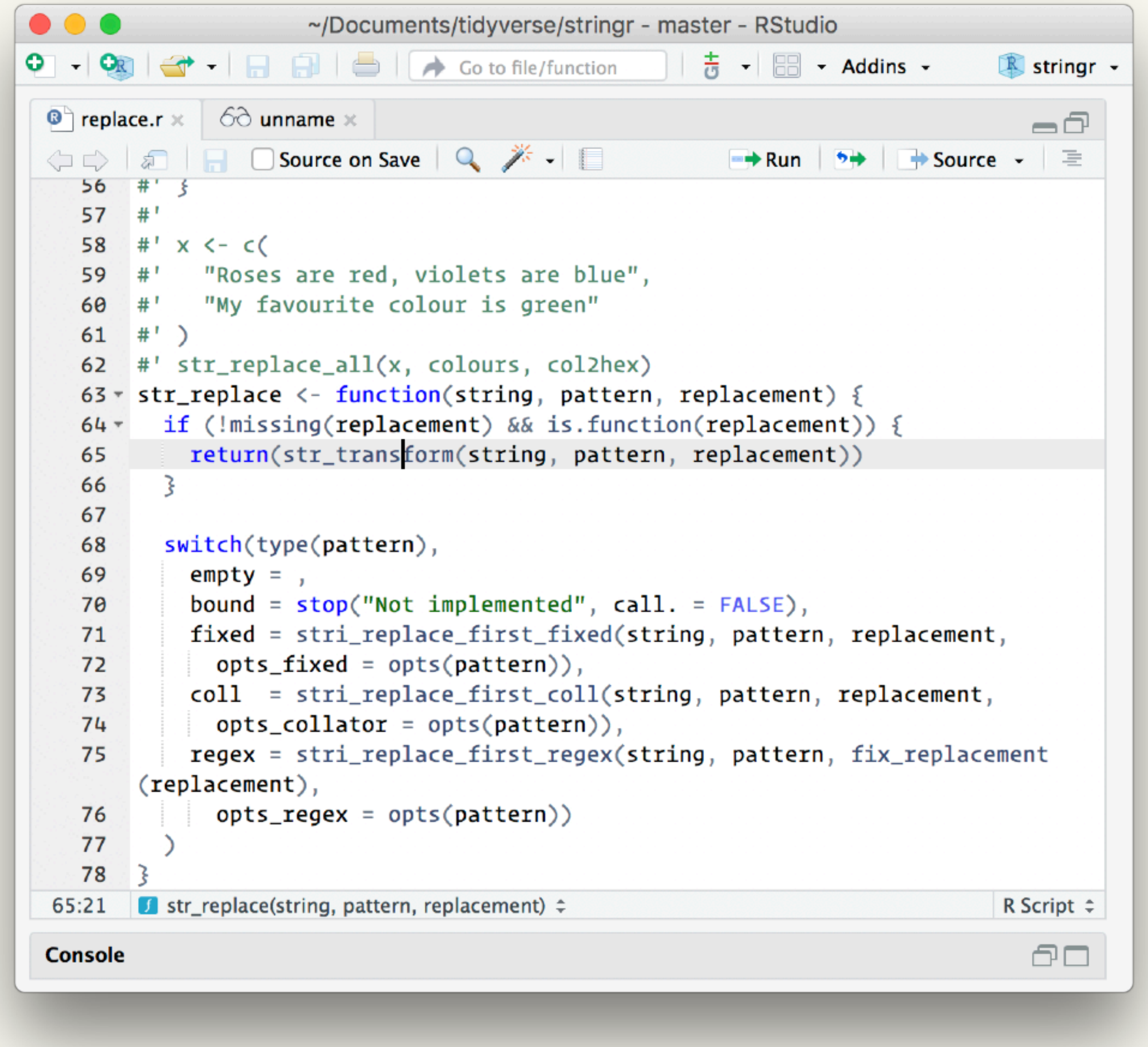
Ctrl + . = find functions/files



```
repl_all
str_replace_all (R/replace.r)
```

```
#' }
#'
#' x <- c(
#'   "Roses are red, violets are blue",
#'   "My favourite colour is green"
#' )
#' str_replace_all(x, colours, col2hex)
str_replace <- function(string, pattern, replacement) {
  if (!missing(replacement) && is.function(replacement)) {
    return(str_transform(string, pattern, replacement))
  }

  switch(type(pattern),
    empty = ,
    bound = stop("Not implemented", call. = FALSE),
    fixed = stri_replace_first_fixed(string, pattern, replacement,
      opts_fixed = opts(pattern)),
    coll  = stri_replace_first_coll(string, pattern, replacement,
      opts_collator = opts(pattern)),
    regex = stri_replace_first_regex(string, pattern, fix_replacement
(replacement),
      opts_regex = opts(pattern))
  )
}
```

```
65:38   str_replace(string, pattern, replacement)   R Script
```



```
#' }
#'
#' x <- c(
#'   "Roses are red, violets are blue",
#'   "My favourite colour is green"
#' )
#' str_replace_all(x, colours, col2hex)
str_replace <- function(string, pattern, replacement) {
  if (!missing(replacement) && is.function(replacement)) {
    return(str_transform(string, pattern, replacement))
  }

  switch(type(pattern),
    empty = ,
    bound = stop("Not implemented", call. = FALSE),
    fixed = stri_replace_first_fixed(string, pattern, replacement,
      opts_fixed = opts(pattern)),
    coll  = stri_replace_first_coll(string, pattern, replacement,
      opts_collator = opts(pattern)),
    regex = stri_replace_first_regex(string, pattern, fix_replacement
(replacement),
      opts_regex = opts(pattern))
  )
}
```

```
65:21   str_replace(string, pattern, replacement)   R Script
```

F2 = jump to definition

Packages

Why use a package?

Easy to share

Standardised conventions

Helpful workflows

```
package.skeleton()
```

Never use this!

Your turn

```
# Verify that you can create a package with:
usethis::create_package(”~/Desktop/mypackage”)
```

Omit if you are
on rstudio.cloud

```
# What files and directories are created?
# You can also create new project using RStudio
# but it has some slight differences that will
# cause friction today (but not in general)

# What are the rules for package names?
# Can you create an invalid name?
```

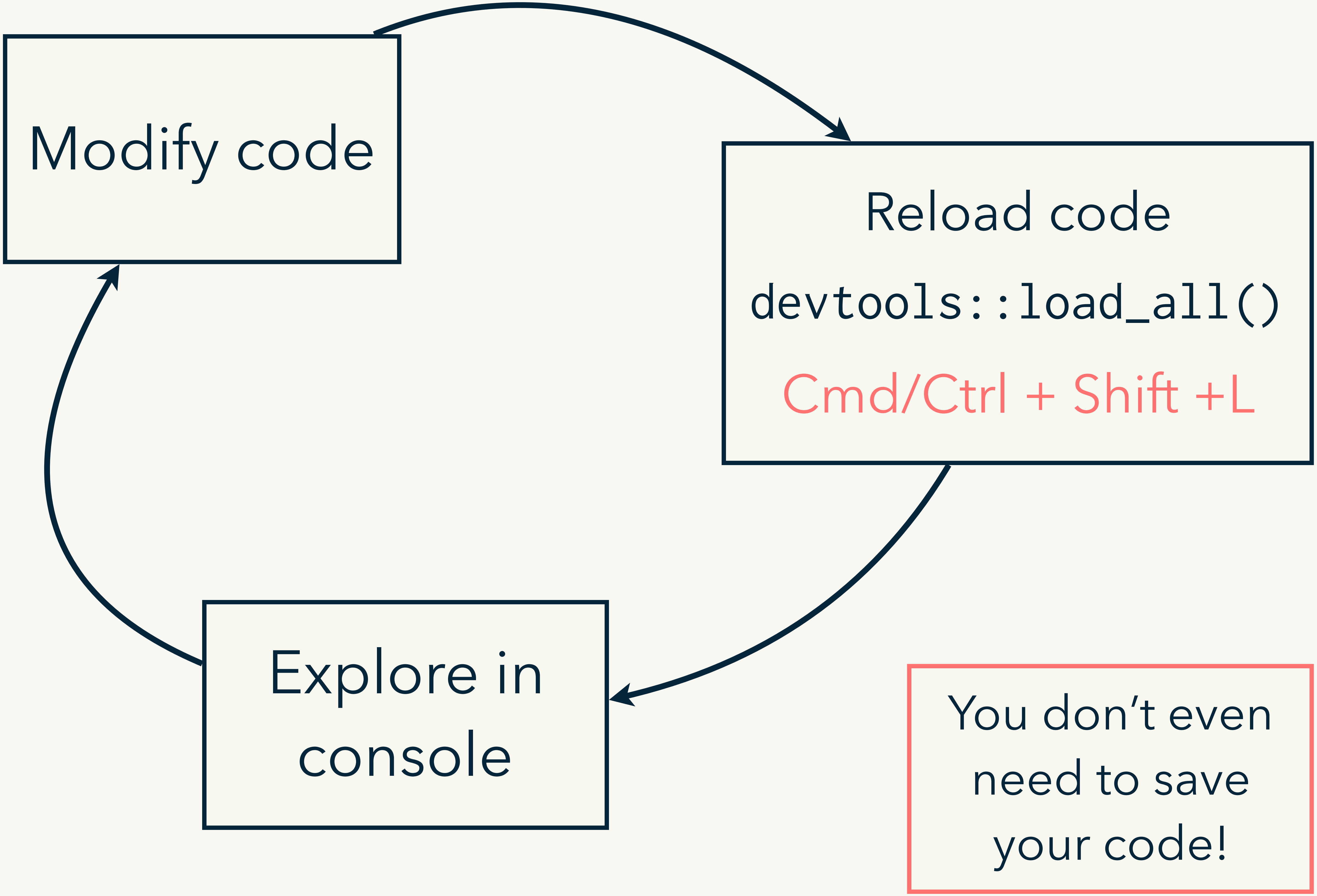
What does create_package() do?

```
✔ Creating '/Users/hadley/Desktop/mypackage/'
✔ Setting active project to '/Users/hadley/Desktop/mypackage'
✔ Creating 'R/'
✔ Writing 'DESCRIPTION'
...
✔ Writing 'NAMESPACE'
✔ Writing 'mypackage.Rproj'
✔ Adding '.Rproj.user' to '.gitignore'
✔ Adding '^mypackage\\.Rproj$', '^\\.Rproj\\.user$' to '.Rbuildignore'
✔ Opening '/Users/hadley/Desktop/mypackage/' in new RStudio session
✔ Setting active project to '<no active project>'
```

What’s a valid name?

```
usethis::create_package(”~/Desktop/mypac-kage”)
#> Error: ’mypac-kage’ is not a valid package
#> name. It should:
#> * Contain only ASCII letters, numbers, and ’.’
#> * Have at least two characters
#> * Start with a letter
#> * Not end with ’.’
```

Why bother? Improved workflow



Modify code

Reload code

`devtools::load_all()`

Cmd/Ctrl + Shift +L

Explore in console

You don’t even need to save your code!

Change project to:

[mylittlepony.Rproj]

mylittlepony.Rproj

Double click
to open

Your turn

Jump to rpony() using only the keyboard.

Load all the functions (using a keyboard shortcut), then run rpony(10).

Uhoh! I have forgotten to include Fluttershy in the list of ponies. Add her, reload the code, and verify that your change worked.

What if you need to create a new file?

```
# 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
```

Your turn

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

Add a function that prints all ponies in alphabetical order.

Practice using the keyboard shortcuts.

R/RStudio setup

Workflow setup: your .Rprofile

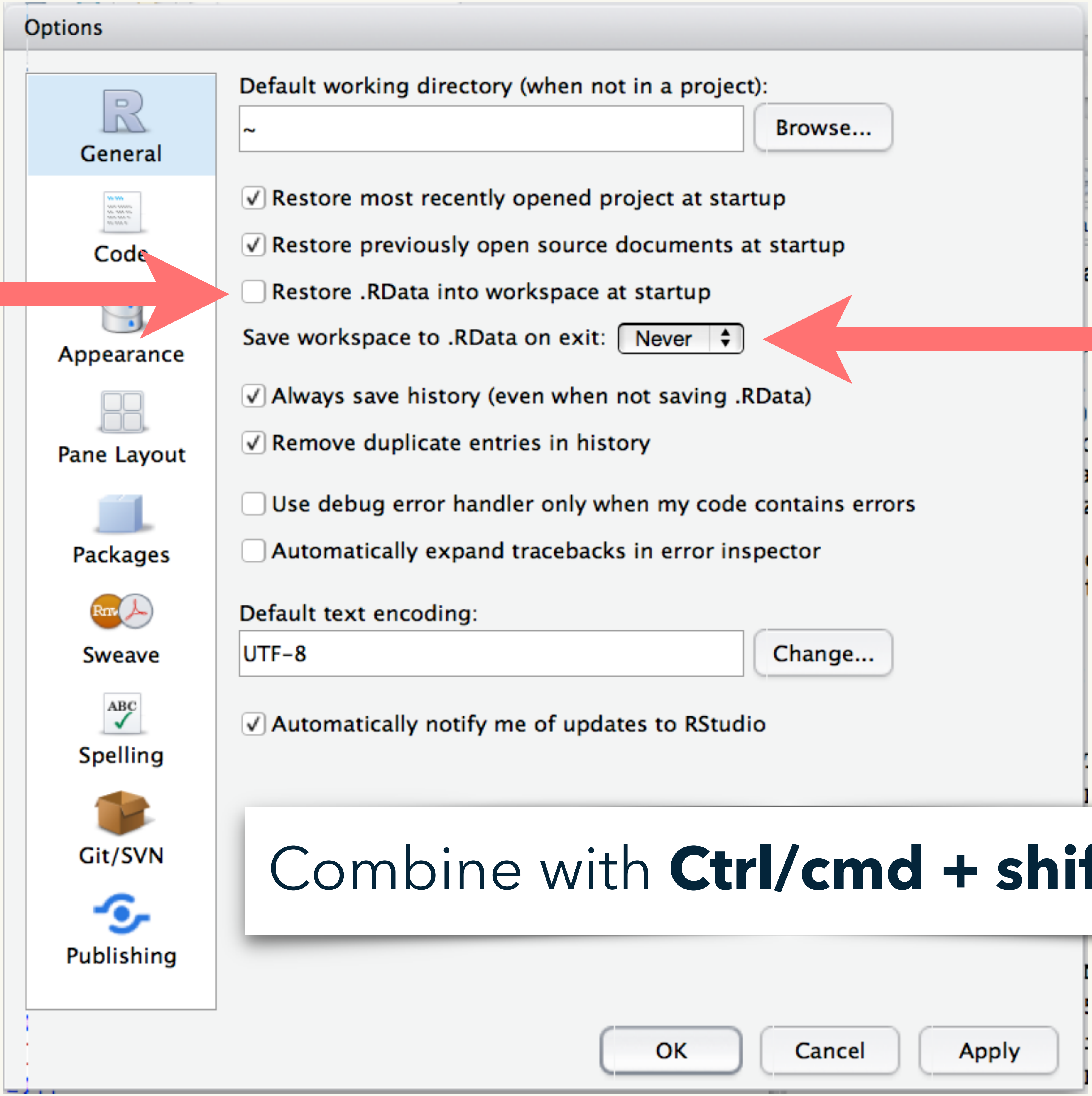
```r
# Setup some code that is run every time
# you start R: usethis::edit_r_profile()

if (interactive()) {
  suppressMessages(require(devtools))
  suppressMessages(require(usethis))
  suppressMessages(require(testthat))
}
```

Never include analysis packages here

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

While you're in there, also add

```
options(
  warnPartialMatchArgs = TRUE,
  warnPartialMatchDollar = TRUE,
  warnPartialMatchAttr = TRUE
)
```



Default working directory (when not in a project):

~

☑ Restore most recently 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

☑ Automatically notify me of updates to RStudio

Combine with **Ctrl/cmd + shift + F10**

Your turn

Follow the instructions in previous slides and make sure that you're optimally configured.

Read *https://www.tidyverse.org/articles/2017/12/workflow-vs-script/* for more justification/motivation