# Develop Packages in R
## Part 1: Make code into R package

*April 2019*

Isabella Gollini
Isabella.Gollini@ucd.ie
@IsabellaGollini

Bruna Wundervald @bwundervald
Chiara Cotroneo @selenocysteina
Jo Nieć @joannaniec
@RLadiesDublin

https://github.com/forwards/workshops/

https://rladies.org/
https://forwards.github.io

# Preliminaries

# The material is mostly drawn from

**http://r-pkgs.had.co.nz**

https://amzn.com/1491910399

© Allie Brosh

# The bad news:

It’s going to be frustrating

http://hyperboleandahalf.blogspot.com/2010/09/four-levels-of-social-entrapment.html

© Allie Brosh

**The good news:**

Frustration is typical and temporary

http://hyperboleandahalf.blogspot.com/2010/06/this-is-why-ill-never-be-adult.html

# Why make a package?

- You want to **test**
- You want to **generalise**
- You want to **document**
- You want to **share**

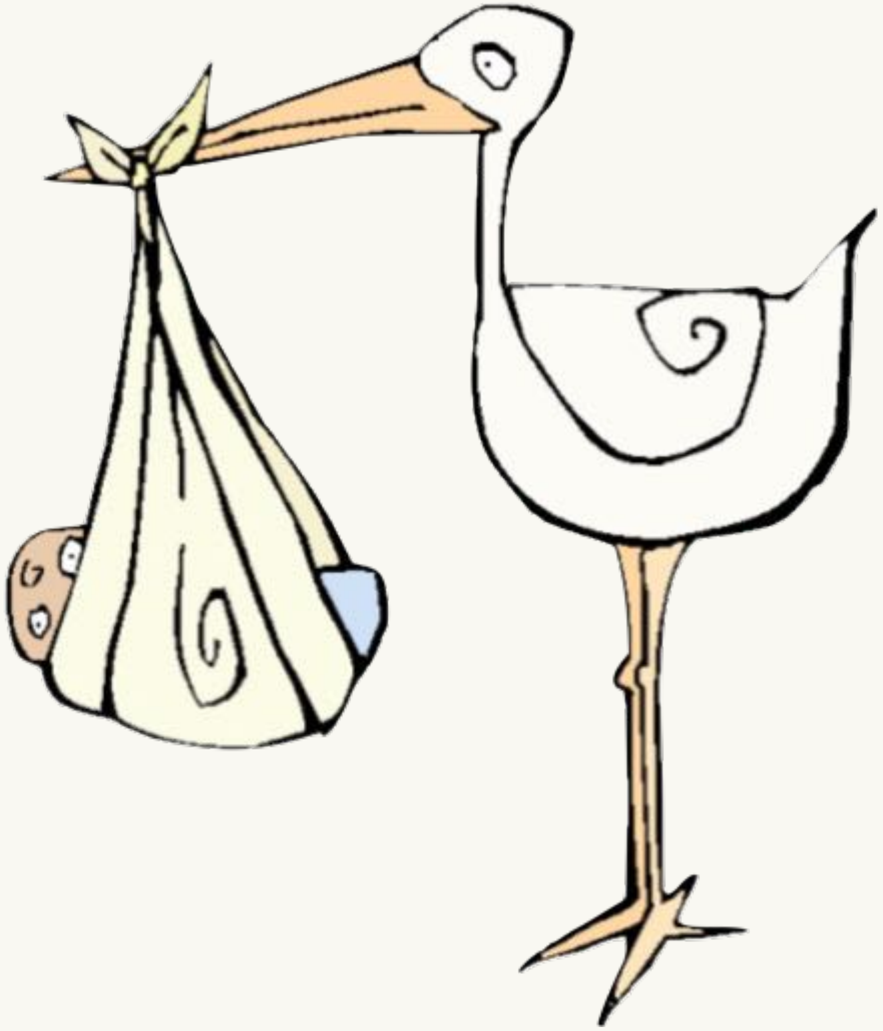
# Get to know your R installation

```r
R.home()

list.files(R.home())

R.version
```

# Where do R packages come from?



# CRAN and GitHub, mostly

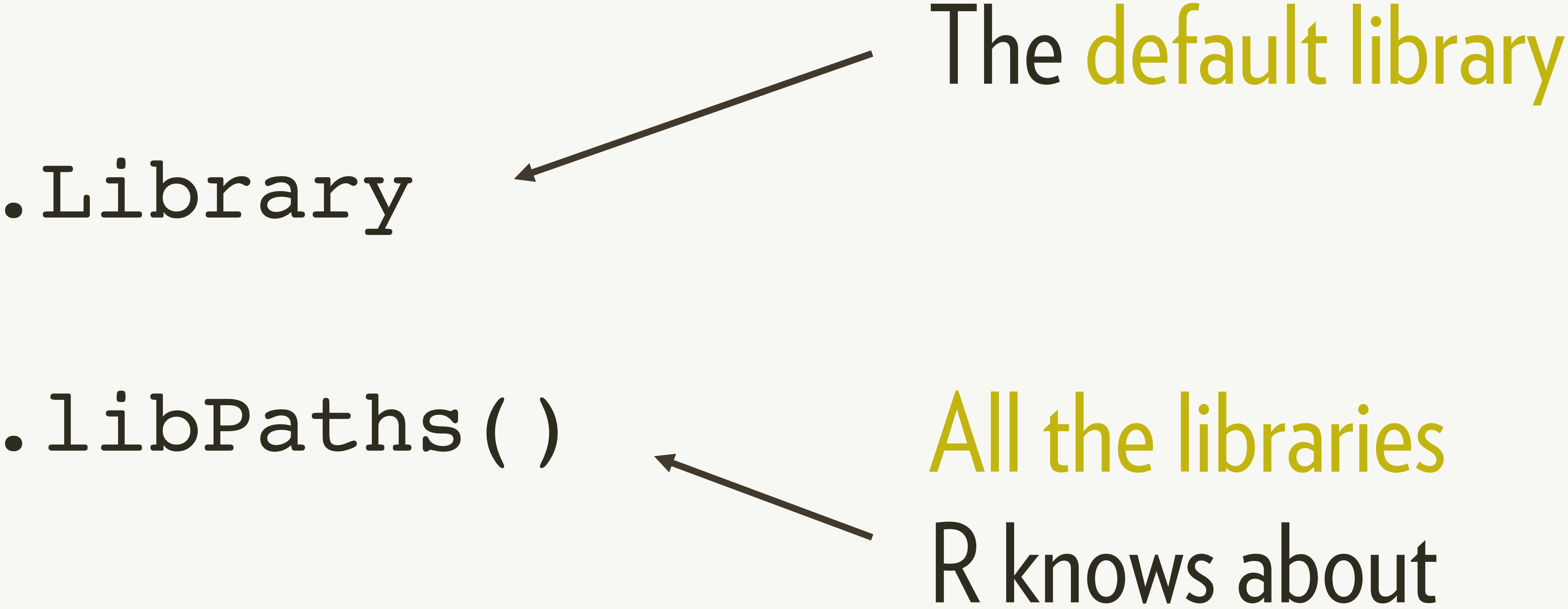
```
install.packages("foo")


library(devtools)
install_github("jane/foo")
```

# Where do R packages live on your computer?

# R packages live in a library

# Get to know your R library(ies?)



```
.Library
```

The default library

```
.libPaths()
```

All the libraries R knows about

`.Library` $\overset{?}{==}$ `.libPaths()`

For many useRs, these are same

Other useRs maintain multiple libraries

E.g., You can put add-on packages in a user-level library:

```
/Users/isabella/resources/R/library
```

# Make sure you have recent versions of these packages

```
old.packages()
packageVersion()

install.packages("devtools")
install.packages("pkgdown")
install.packages("roxygen2")
install.packages("testthat")
install.packages("tidyverse")
install.packages("usethis")
```

# My first package

# Your turn

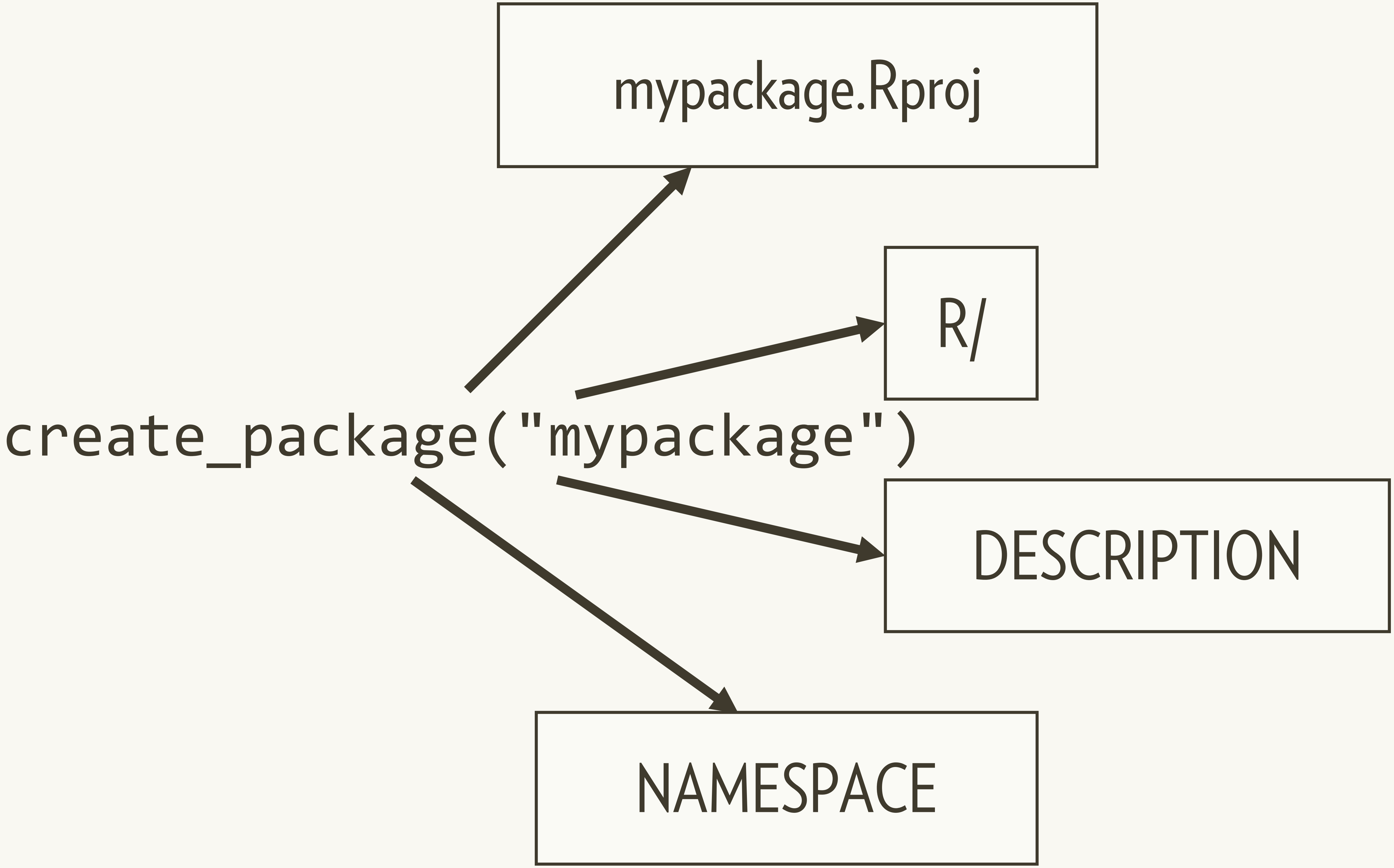
```
# Create a package on your Desktop:
usethis::create_package("~/Desktop/mypackage")

# Or on RStudio.cloud (create within main project)
usethis::create_package("mypackage")
```

This will create a new R project and start a new session.

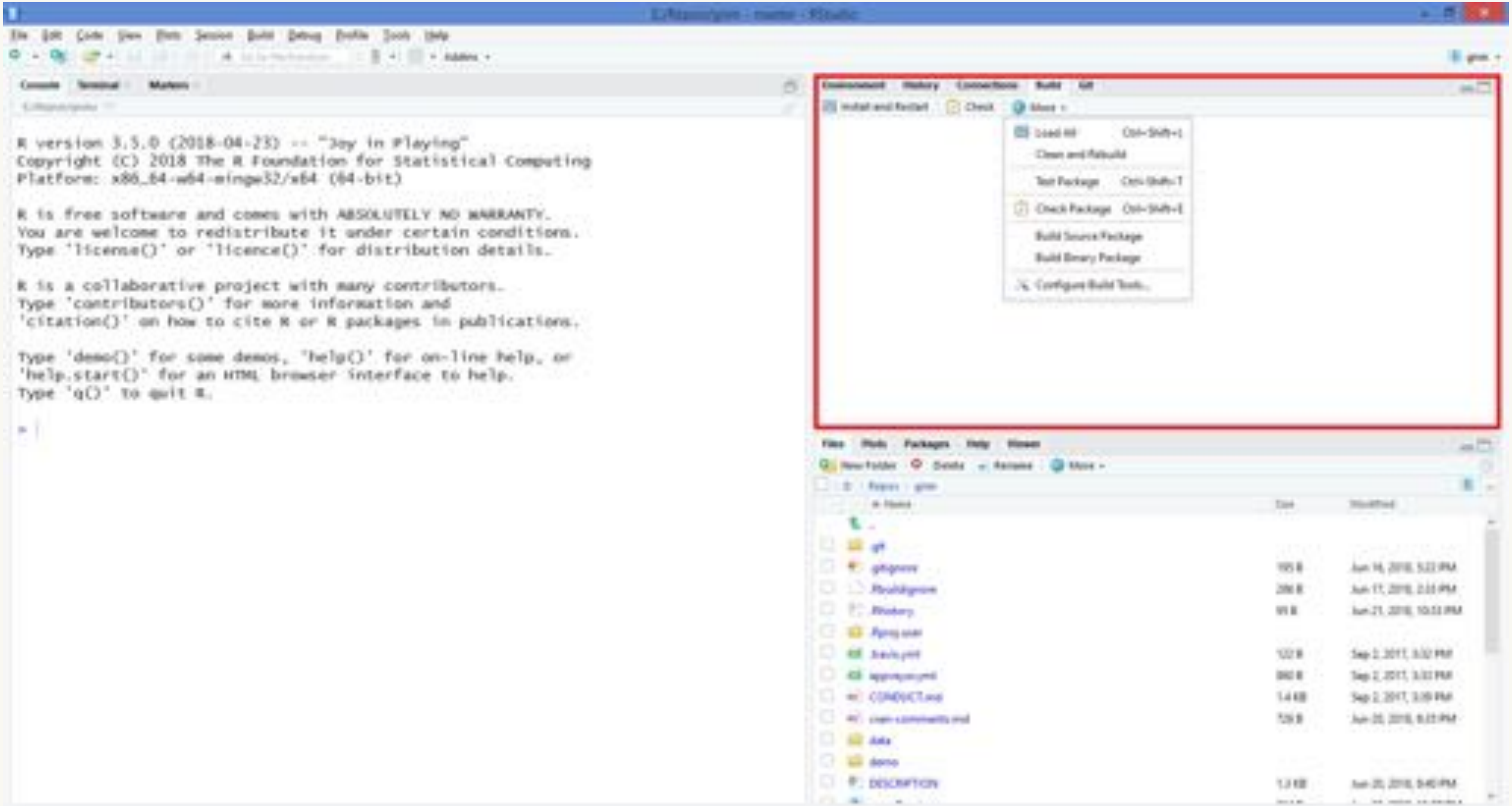
What other files and directories are created?

# What happens we run create_package()?



```
create_package("mypackage")
```

# RStudio helpers: build (& git) menus



```
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

| Name | Size | Modified |
|---|---|---|
| .. | | |
| .git | | |
| .gitignore | 195 B | Jun 16, 2018, 5:22 PM |
| .Rbuildignore | 286 B | Jun 17, 2018, 2:23 PM |
| .Rhistory | 93 B | Jun 21, 2018, 10:33 PM |
| .Rproj.user | | |
| .travis.yml | 122 B | Sep 2, 2017, 3:32 PM |
| appveyor.yml | 880 B | Sep 2, 2017, 3:32 PM |
| CONDUCT.md | 1.4 KB | Sep 2, 2017, 3:39 PM |
| cran-comments.md | 726 B | Jun 20, 2018, 8:23 PM |
| data | | |
| demo | | |
| DESCRIPTION | 1.3 KB | Jun 20, 2018, 9:40 PM |

# DESCRIPTION

```
Package: mypackage
Version: 0.0.0.9000
Title: What the Package Does (One Line, Title Case)
Description: What the package does (one paragraph).
Authors@R: person("First", "Last", ,
"first.last@example.com", c("aut", "cre"))
License: What license it uses
Encoding: UTF-8
LazyData: true
ByteCompile: true
```

# Your turn

Use the package `available` to check if the package name `mypackage` is available:

```
available::available("mypackage")
```

Try with other names for your package.

# Development workflow

How is developing a
package
same / different
from developing a script?

## How same?

Iterate early and often!

Change it, try it, change it, try it, ***ad nauseum***

## How different?

Write functions, not “top-level” code.

Dependencies are different,

no library() calls

Install & Restart (or simulate that),

don’t source()

You write this

... but you use this

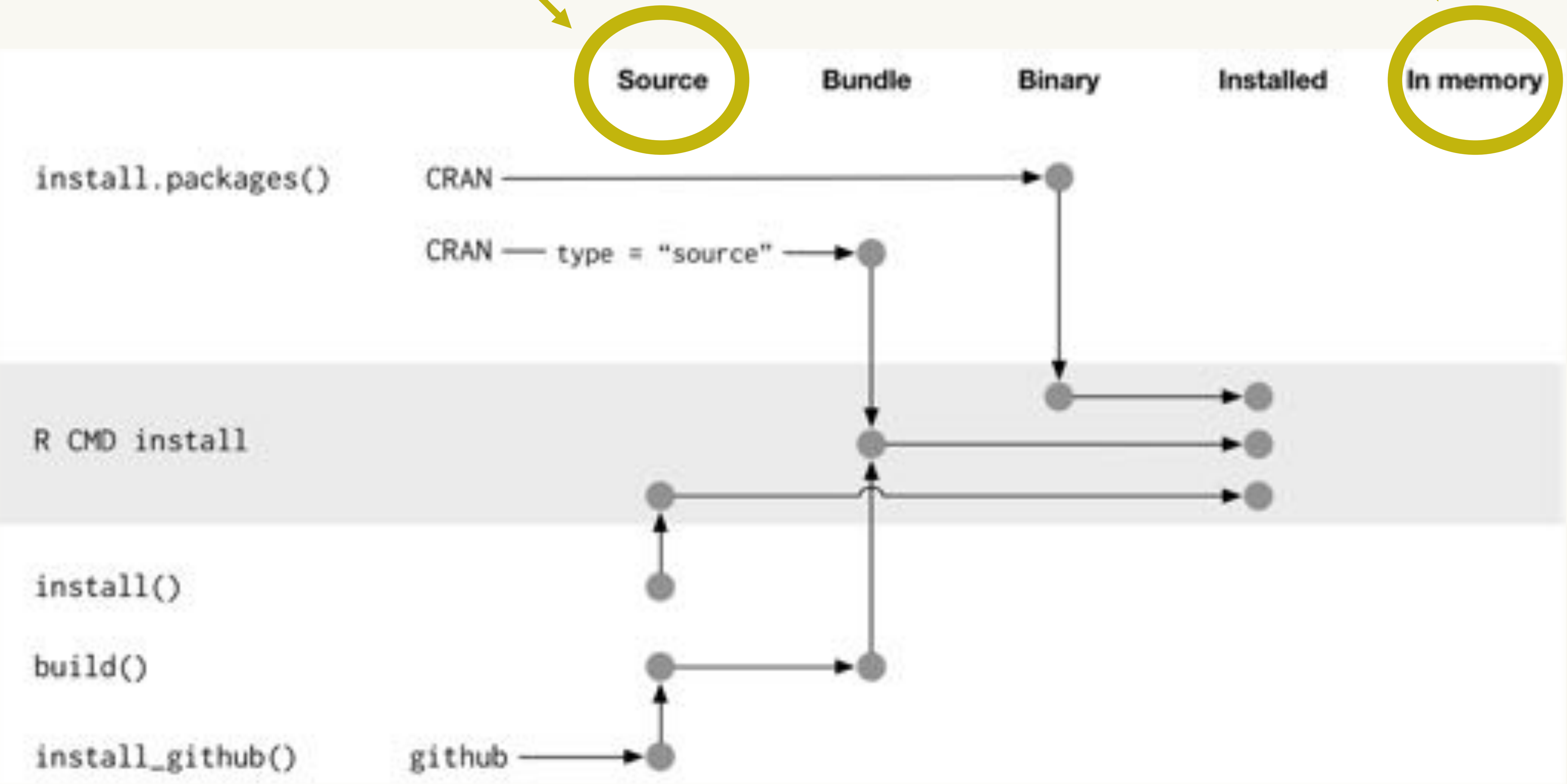


Figure from Hadley Wickham's book, R packages

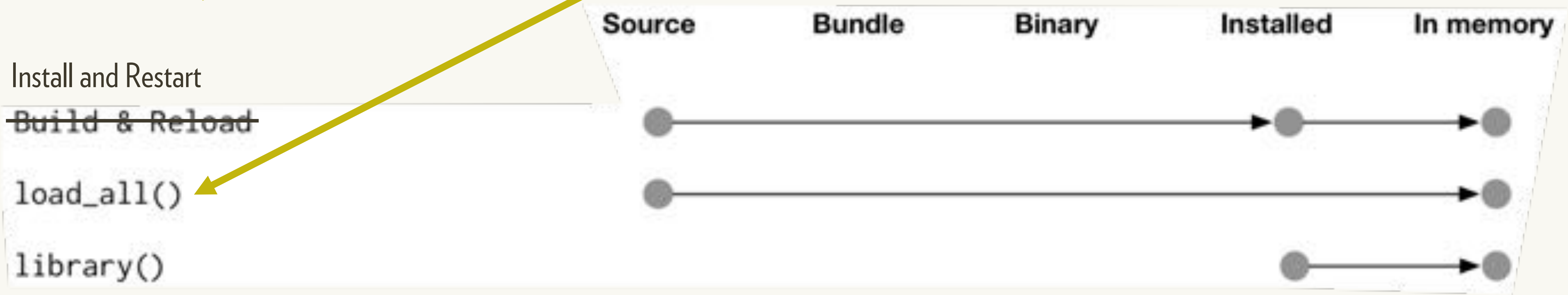
http://r-pkgs.had.co.nz

https://github.com/hadley/r-pkgs/blob/master/diagrams/installation.png

# How do packages get into memory?

Development: every couple hours or days?

Development: Every 2 minutes



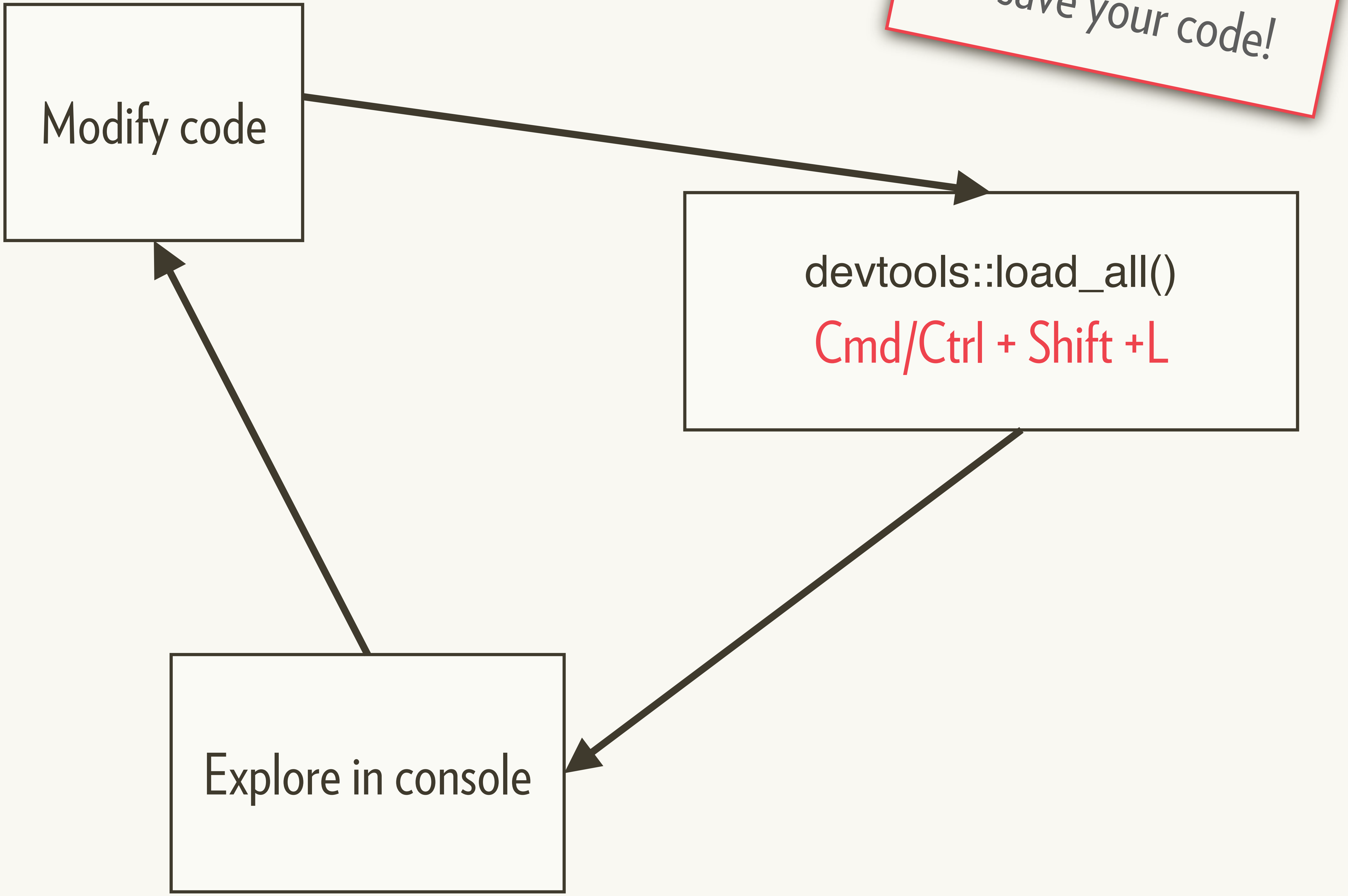
Normal day-to-day usage

Figure from Hadley Wickham's book, R packages

http://r-pkgs.had.co.nz

https://github.com/hadley/r-pkgs/blob/master/diagrams/loading.png

# Basic workflow



Modify code

devtools::load_all()

Cmd/Ctrl + Shift +L

Explore in console

You don't even need to save your code!

# Your turn

[Uses https://github.com/forwards/mylittlepony]

- Open `mylittlepony.Rproj`.
- Load all the functions using, `devtools::load_all()` 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.