

# object of type closure is not subsettable

[rstd.io/debugging](https://rstd.io/debugging)

Jennifer Bryan

RStudio

 @jennybc

 @JennyBryan

>





Fun!



Fun!



Less fun.

## WHAT'S YOUR MAIN DEBUGGING METHOD?

- A. I don't have a method. Why do you think I'm here?!?
- B. I ask in an online forum or consult my local expert.
- C. Google is my BFF.
- D. Print debugging is my jam.
- E. I use R's debugging tools, like `browser()`.

The background of the slide features a woman in a dark wetsuit swimming underwater. A bright, glowing light beam originates from her right hand and extends towards the bottom right corner of the frame. The water is a deep teal color.

1. RESET

2. REPREX

3. DEBUG

4. DETER



# RESET

A close-up, low-angle shot of a vehicle's front wheel and fender. The ground is thick, dark brown mud. The wheel is almost entirely submerged, with only the hubcap and a thin strip of tire visible above the surface. Mud tracks lead away from the wheel towards the right. The background is dark and out of focus.

If at first you don't succeed...  
**try exactly the same thing again.**

A close-up photograph of a person's index finger hovering directly above a power button on a dark-colored computer keyboard. The power button is a standard white-on-black rectangular key. To the left of the power button is a key labeled 'F12' with a speaker icon. To the right is a key with a series of small dots. The background is slightly blurred, emphasizing the action of the finger.

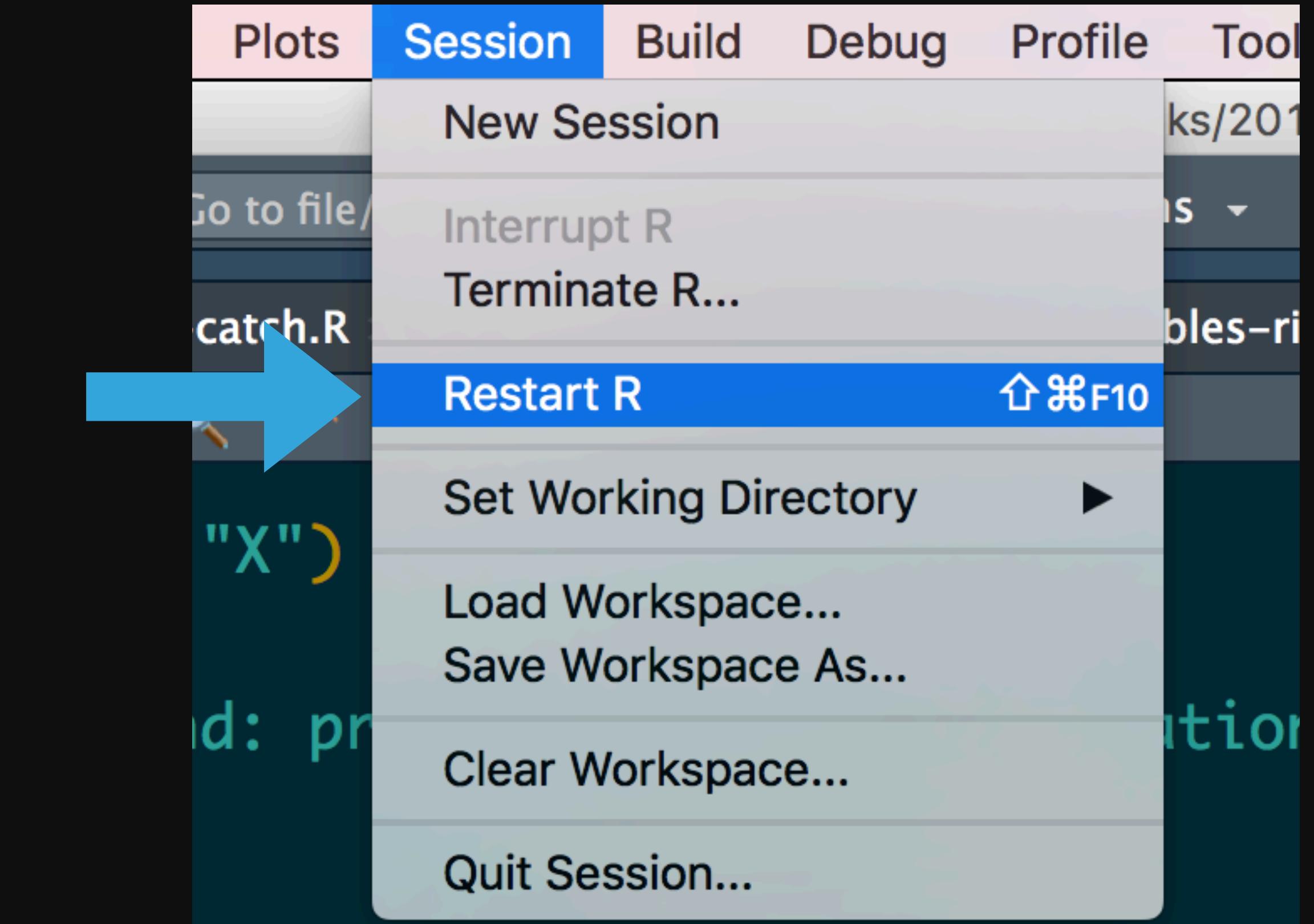
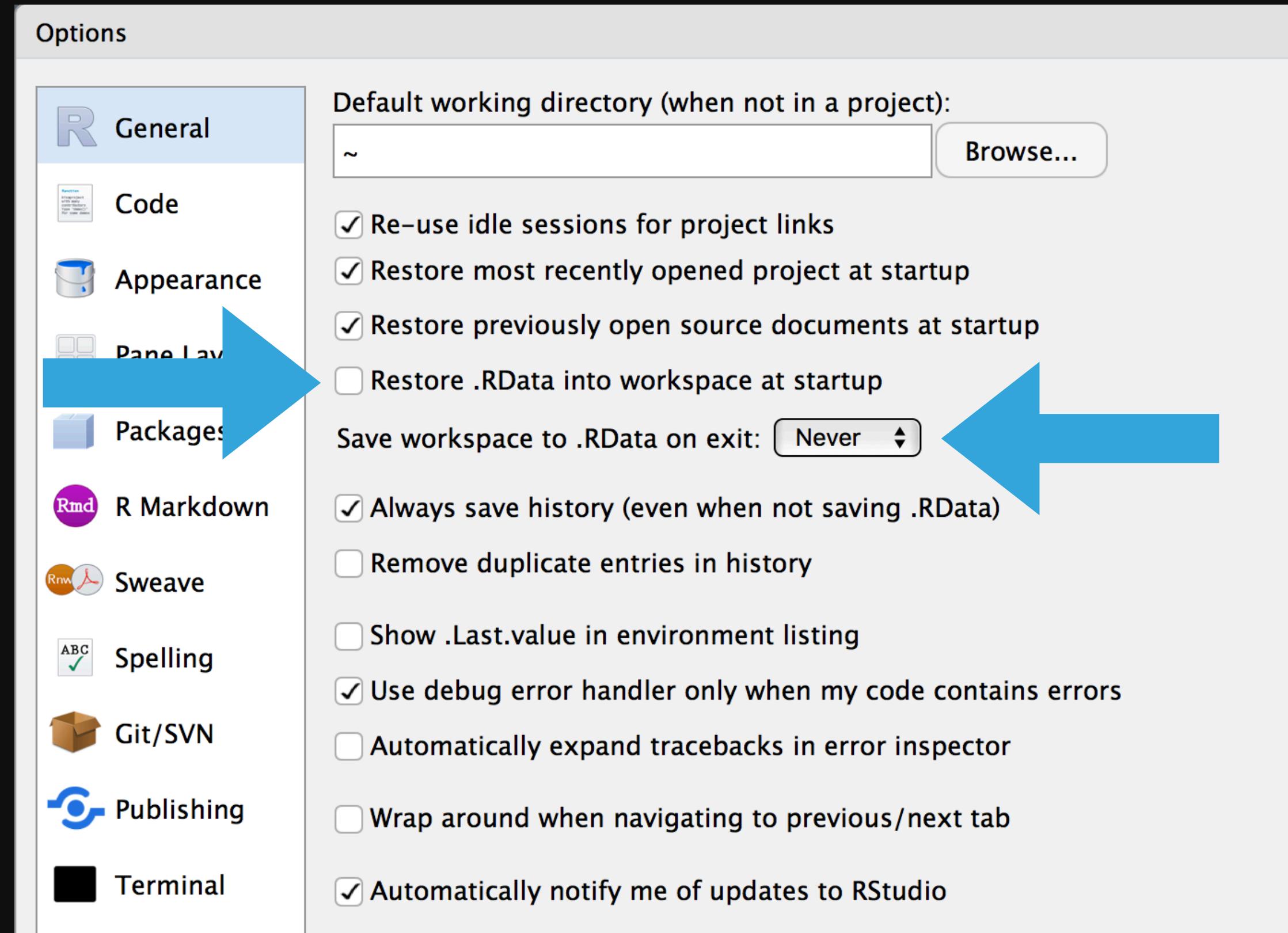
Have you tried turning it **OFF** and **ON** again?

“

If you love something, set it free.  
If it comes back, it is yours.  
If it does not, it never was.

# RESTART R

Especially when things get weird



R --no-save --no-restore-data

R --no-save --no-restore-data

!=

rm(list = ls())

# WHICH PERSIST AFTER rm(list = ls())?

- A. library(dplyr)
- B. summary <- head
- C. options(stringsAsFactors = FALSE)
- D. Sys.setenv(LANGUAGE = "fr")
- E. x <- 1:5
- F. attach(iris)

# WHICH PERSIST AFTER rm(list = ls())?

- A. library(dplyr) ✓
- B. summary <- head ✗
- C. options(stringsAsFactors = FALSE) ✓
- D. Sys.setenv(LANGUAGE = "fr") ✓
- E. x <- 1:5 ✗
- F. attach(iris) ✓

```
R --no-save --no-restore-data
```

```
>>>
```

```
rm(list = ls())
```

# A FRESH START

Cleans workspace

Resets options and env vars

Clears search path

R sessions are like crops,



not houseplants.



2

REPREX

A close-up profile photograph of a woman's face. She is wearing round, light-colored glasses and has long, dark hair. Her gaze is directed downwards and to her left, with a contemplative or sad expression. Her hand is resting against her chin, supporting her head.

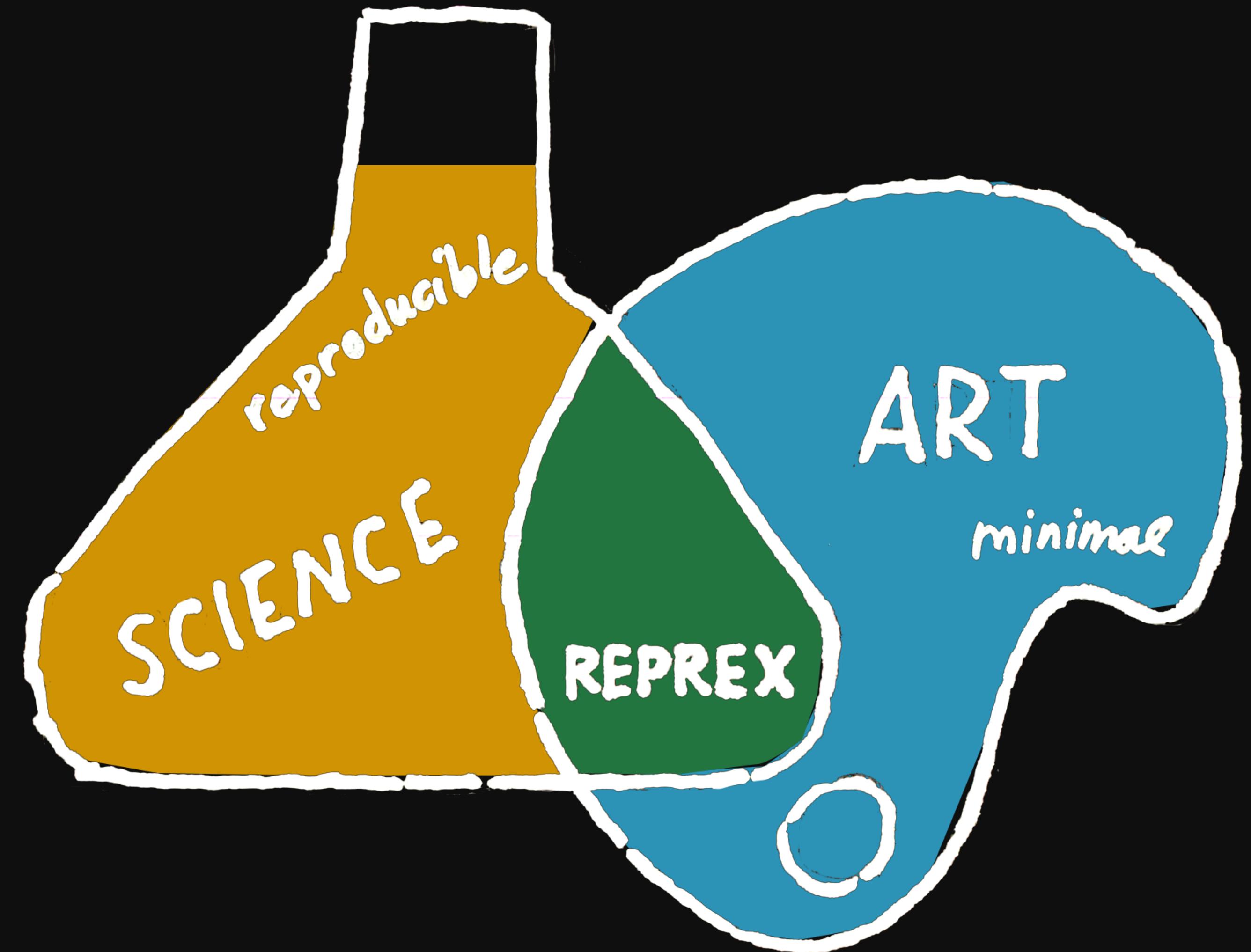
If at first you don't succeed...  
**brood, dither, and fret.**

**“ Don't wring hands and speculate.  
Work a small concrete example that  
reveals, confirms, or eliminates  
something.**

reprex

minimum  
reproducible  
example





```
template <- "${EXCLAMATION} - your reprex is ${adjective}!"  
praise(template)  
#> Error in praise(template): could not find function "praise"
```

```
library(praise)
praise(template)
#> Error in grepl(template_pattern, x): object 'template' not found
```

```
library(praise)
template <- "{$EXCLAMATION} - your reprex is ${adjective}!"
praise(template)
#> [1] "WOWIE - your reprex is glorious!"
```

# REPRODUCIBLE

No reliance on hidden state

what you  
THINK/SAY  
you are doing

what you are  
ACTUALLY DOING



the Problem

To FIND a NEEDLE



Try looking in a  
**SMALLER STACK**



```
fs::dir_info(path = raw_data_path,          # list xls[s] files
             regexp = "[.]xls[x]?$") %>%
  dplyr::mutate(sheets = purrr::map(        # create list-col of
    path, ~ readxl::excel_sheets)) %>%      # worksheets
  tidyverse::unnest(sheets) %>%              # get one row per worksheet
  dplyr::mutate(data = purrr::map2(          # read data into a list-col
    path, sheets,                            # of data frames
    ~ readxl::read_excel(.x, .y) %>%       # call `as.character()`
    dplyr::mutate_all(as.character))         # on each column
  ))
```

#> New names:

```
#> * ` ` → `..2`  
#> * ` ` → `..3`  
#> * ` ` → `..4`  
#> * ` ` → `..5`  
#> * ` ` → `..6`  
#> Error: the ... list does not contain 3 elements
```

```
fs::dir_info(path = raw_data_path,          # list xls[s] files
             regexp = "[.]xls[x]?$") %>%
  dplyr::mutate(sheets = purrr::map(        # create list-col of
    path, ~ readxl::excel_sheets)) %>%      # worksheets
  tidyverse::unnest(sheets) %>%              # get one row per worksheet
  dplyr::mutate(data = purrr::map2(          # read data into a list-col
    path, sheets,                            # of data frames
    ~ readxl::read_excel(.x, .y) %>%        # call `as.character()`
    dplyr::mutate_all(as.character))          # on each column
  ))
#> New names:
#> * `..` → `..2`
#> * `..` → `..3`
#> * `..` → `..4`
#> * `..` → `..5`
#> * `..` → `..6`
#> Error: the ... list does not contain 3 elements
```

```
fs::dir_info(path = raw_data_path,  
            regexp = "[.]xls[x]?$") %>%  
  
dplyr::mutate(sheets = purrr::map(  
    path, ~ readxl::excel_sheets)) %>%  
  
tidyr::unnest(sheets) %>%  
  
dplyr::mutate(data = purrr::map2(  
    path, sheets,  
    ~ readxl::read_excel(.x, .y) %>%  
        dplyr::mutate_all(as.character)  
)  
#> New names:  
#> * ` ` → `..2`  
#> * ` ` → `..3`  
#> * ` ` → `..4`  
#> * ` ` → `..5`  
#> * ` ` → `..6`  
#> Error: the ... list does not contain 3 elements
```

**private xlsx files**

**~10 lines of code**

**5 packages**

**8 functions**

## inlined data

```
dat <- data.frame(`..1` = 1)  
dplyr::mutate_all(dat, as.character)  
#> Error: Column 1 must not have names  
#> of the form ... or ..j.  
#> Use .name_repair to specify repair.
```

2 lines of code

1 package

1 function

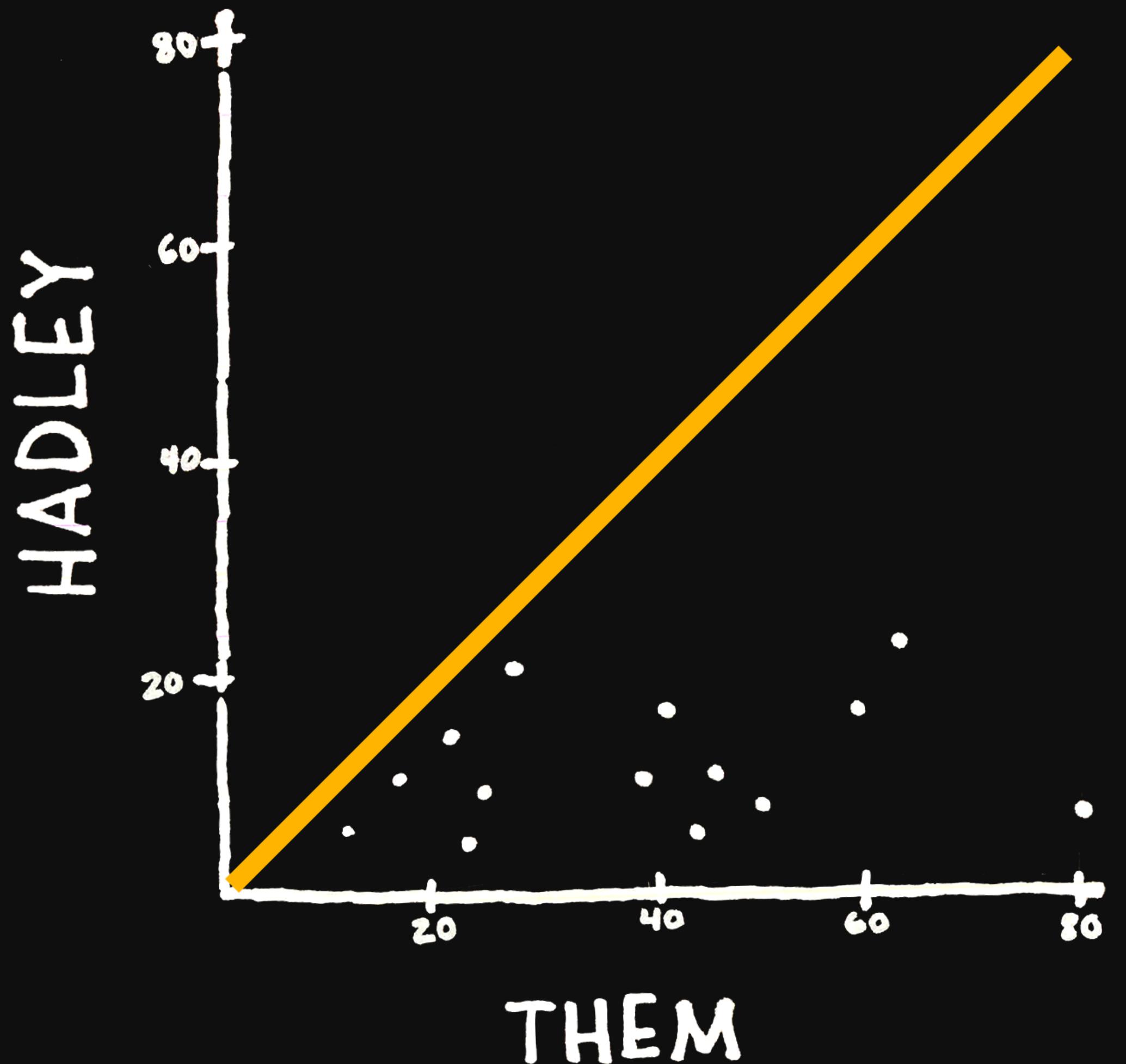
# MINIMAL

Inputs are small and simple

~~Extraneous packages~~

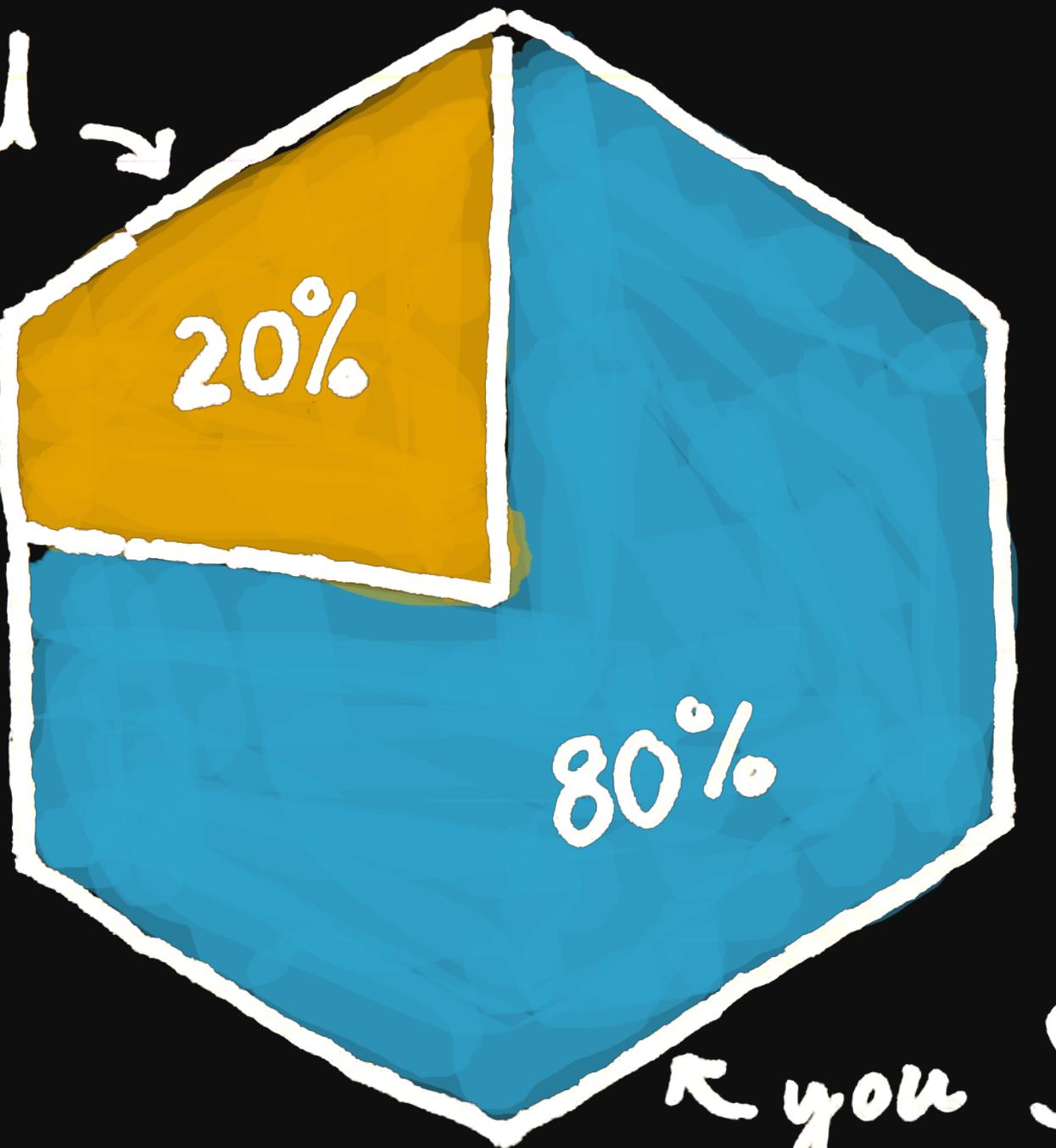
~~Unnecessary function calls~~

# SIZE of REPREX



# WHY make a REPREX?

helps them  
HELP YOU →



→ you SOLVE  
your own  
PROBLEM



**3 DEBUG**

# HAVE YOU GOTTEN STUCK IN THE DEBUGGER?

- A. R has a debugger? What is a debugger?
- B. No.
- C. Yes.
- D. Yes and I'm not even sure how I got there.

```
# Error : .onLoad failed in loadNamespace() for 'rJava', details:  
# call: dyn.load(file, DLLpath = DLLpath, ...)  
# error: unable to load shared object '/Users/janedoe/Library/R/3.6/library/rJava/libs/rJava.so':  
# libjvm.so: cannot open shared object file: No such file or directory  
# Error: loading failed  
# Execution halted  
# ERROR: loading failed  
# * removing '/Users/janedoe/Library/R/3.6/library/rJava/'  
# Warning in install.packages :  
# installation of package 'rJava' had non-zero exit status
```

```
# Error : blah bl failed blah blah blah blah blah blah blah:  
# blah: bla.blah(blah, blahbla = blahbla, ...)  
# error: unable to blah blah blah bla '/blahb/blahbla/blahbla/b/b.b/blahbla/blahb/blah/blahb.so':  
# blahbl.so: cannot blah blah blah bla: No blah blah blah blah bl  
# Error: blah bl failed  
# Blah blah blah b  
# ERROR: blah bl failed  
# * removing '/blahb/blahbla/blahbla/b/b.b/blahbla/blahb/'  
# Warning in blah blah blah b :  
# blahblahblah bl blahbla 'blahb' bla bla-blah blah blahbl
```

```
dat
#>          blackberry blueberry peach   plum
#> calories           4            1     59    30
#> weight            9            2    150    78
#> yumminess         6            8     10     5
```

```
fruit_avg(dat, pattern = "berry")
#> Found 2 fruits!
#>   calories      weight  yumminess
#>       2.5        5.5     7.0
```

```
dat
#>      blackberry blueberry peach   plum
#> calories          4            1     59    30
#> weight           9            2    150    78
#> yumminess        6            8     10     5

fruit_avg(dat, pattern = "melon")
#> Found 0 fruits!
#>   calories      weight  yumminess
#>       NaN         NaN      NaN
```

```
dat
#>          blackberry blueberry peach   plum
#> calories           4            1      59     30
#> weight            9            2     150     78
#> yumminess         6            8      10      5

fruit_avg(dat, pattern = "black")
#> Found fruits!
#> Error in rowMeans(mini_dat): 'x' must be an array
#> of at least two dimensions
```

```
dat
#>          blackberry blueberry peach   plum
#> calories           4             1     59    30
#> weight            9             2    150    78
#> yumminess         6             8     10     5

fruit_avg(dat, pattern = "black")
#> Found fruits!
#> Error in rowMeans(mini_dat): 'x' must be an array
#> of at least two dimensions
```

# Certificate of Death

16

WILLIE

First Name	Middle Name	Last Name
PARTICULARS <i>(Medical Examiner.)</i>		
MEDICAL CERTIFICATE <i>(To be filled in by Medical Examiner)</i>		
16. PLACE OF DEATH:		
(a) NEW YORK CITY: (a)		
(c) Name of Hospital or Institution Flat <i>(If not in hospital)</i>		
(d) If elsewhere than in hospital, name place of death, as: hotel, a store, street, taxicab, etc.		
17. DATE AND HOUR OF DEATH (Month)		
18. SEX   19. Color		



<b>Certificate of Death</b>		
: 16		
<b>WILLIE</b>		
First Name	Middle Name	Last Name
<b>Particulars</b> <i>(Medical Examiner.)</i>		
<b>MEDICAL CERTIFICATE</b> <i>(To be filled in by Medical Examiner.)</i>		
<b>16. PLACE OF DEATH:</b>		
(a) NEW YORK CITY: (a)		
(c) Name of Hospital or Institution Flat <i>(If not in hospital)</i>		
(d) If elsewhere than in hospital, name place of death, as: hotel, a store, street, taxicab, etc.		
<b>17. DATE AND HOUR OF DEATH</b> (Month) December		
18. SEX	19. Color	
Married		



traceback()

browser()

options(error = recover)

```
fruit_avg(dat, pattern = "black")
#> Found fruits!
#> Error in rowMeans(mini_dat): 'x' must be an array
#> of at least two dimensions

traceback()
#> 3: stop("'x' must be an array of at least two dimensions")
#> 2: rowMeans(mini_dat) at fruit_avg.R#5
#> 1: fruit_avg(dat, pattern = "black")
```

call  
stack  
trace  
back

call stack  
trace back  
stack trace  
back trace  
call back

call  
stack  
trace  
back

call stack  
trace back  
stack trace  
back trace  
~~call back~~

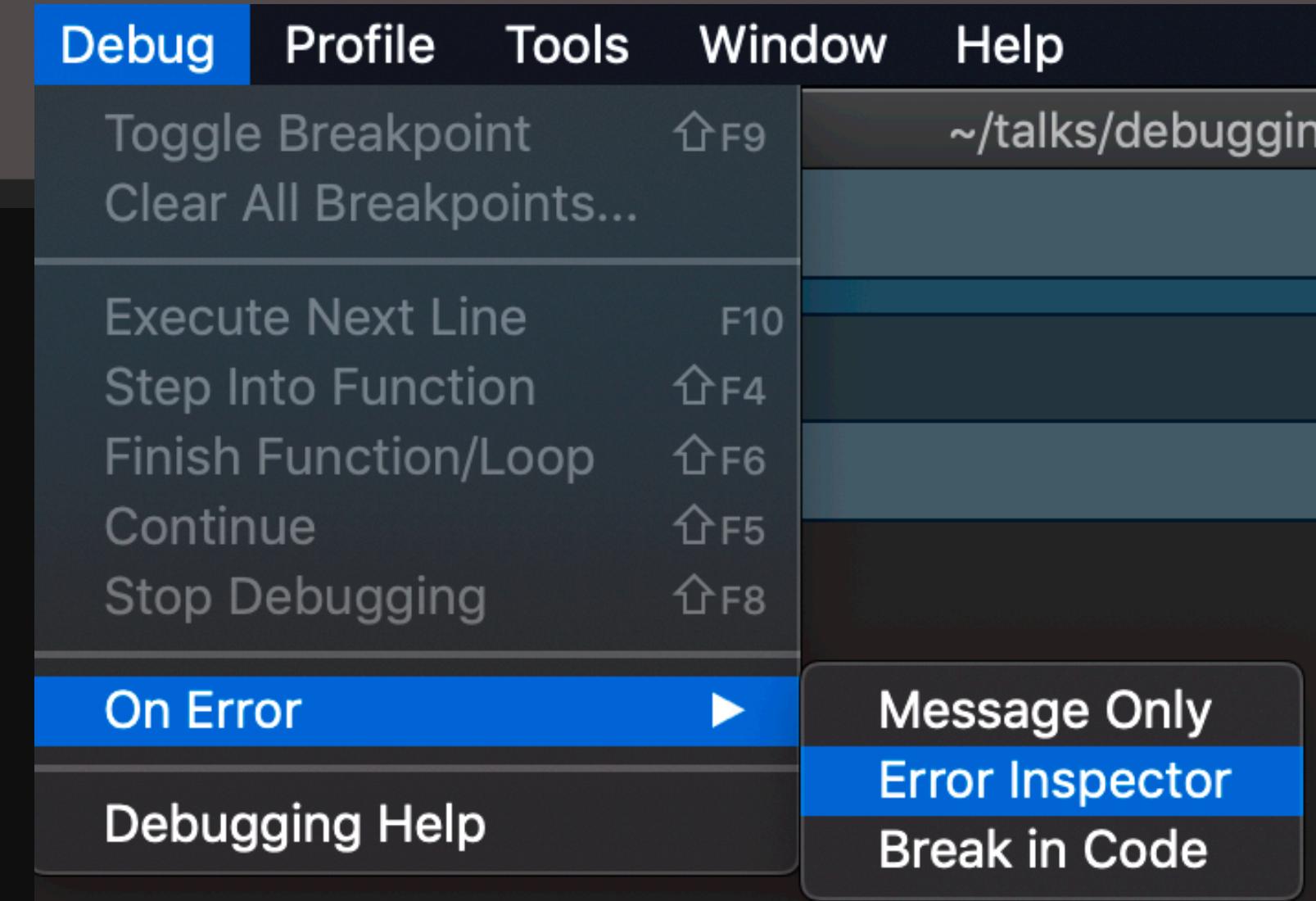
```
fruit_avg(dat, pattern = "black")
#> Found fruits!
#> Error in rowMeans(mini_dat): 'x' must be an array
#> of at least two dimensions

rlang::last_trace()
#> <error/rlang_error>
#> 'x' must be an array of at least two dimensions
#> Backtrace:
#>   1. global::fruit_avg(dat, pattern = "black")
#>   2. base::rowMeans(mini_dat) R/fruit_avg.R:5:2
```

```
> fruit_avg(dat, pattern = "black")
Found fruits!

Error in rowMeans(mini_dat) :
  'x' must be an array of at least two dimensions
3. stop("'x' must be an array of at least two dimensions")
2. rowMeans(mini_dat) at fruit_avg.R#5
1. fruit_avg(dat, pattern = "black")
```

▲ Hide Traceback  
● Rerun with Debug





**Video of a tiny room hidden behind an electrical outlet.**

**By Mozu Studios**

<https://www.mozustudios.com>

<https://www.instagram.com/p/B6NvxK7JnpU>

```
options(error = recover)

fruit_avg(dat, "black")
#> Found fruits!
#> Error in rowMeans(mini_dat): 'x' must be an array
#> of at least two dimensions
```

Enter a frame number, or 0 to exit

```
1: fruit_avg(dat, "black")
2: fruit_avg.R#5: rowMeans(mini_dat)
```

Selection: 1

Enter a frame number, or 0 to exit

1: fruit\_avg(dat, "black")

2: fruit\_avg.R#5: rowMeans(mini\_dat)

Selection: 1

Browse[1]> ls.str()

cols : int 1

dat : 'data.frame': 3 obs. of 4 variables:

\$ blackberry: int 4 9 6

\$ blueberry : int 1 2 8

\$ peach : int 59 150 10

\$ plum : int 30 78 5

mini\_dat : int [1:3] 4 9 6

pattern : chr "black"



Enter a frame number, or 0 to exit

1: fruit\_avg(dat, "black")  
2: fruit\_avg.R#5: rowMeans(mini\_dat)

Selection: 1

Browse[1]> ls.str()  
cols : int 1  
dat : 'data.frame': 3 obs. of 4 variables:  
 \$ blackberry: int 4 9 6  
 \$ blueberry : int 1 2 8  
 \$ peach : int 59 150 10  
 \$ plum : int 30 78 5  
mini\_dat : int [1:3] 4 9 6  
pattern : chr "black"

Enter a frame number, or 0 to exit

1: fruit\_avg(dat, "black")

2: fruit\_avg.R#5: rowMeans(mini\_dat)

Selection: 1

The screenshot shows the RStudio interface with the Environment tab selected. The top navigation bar includes tabs for Environment, History, Connections, Build, and Git. Below the tabs are various icons for file operations like Import Dataset and a search bar. The main pane displays the environment variables:

Data	
dat	3 obs. of 4 variables
Values	
cols	1L
mini_dat	int [1:3] 4 9 6
pattern	"black"

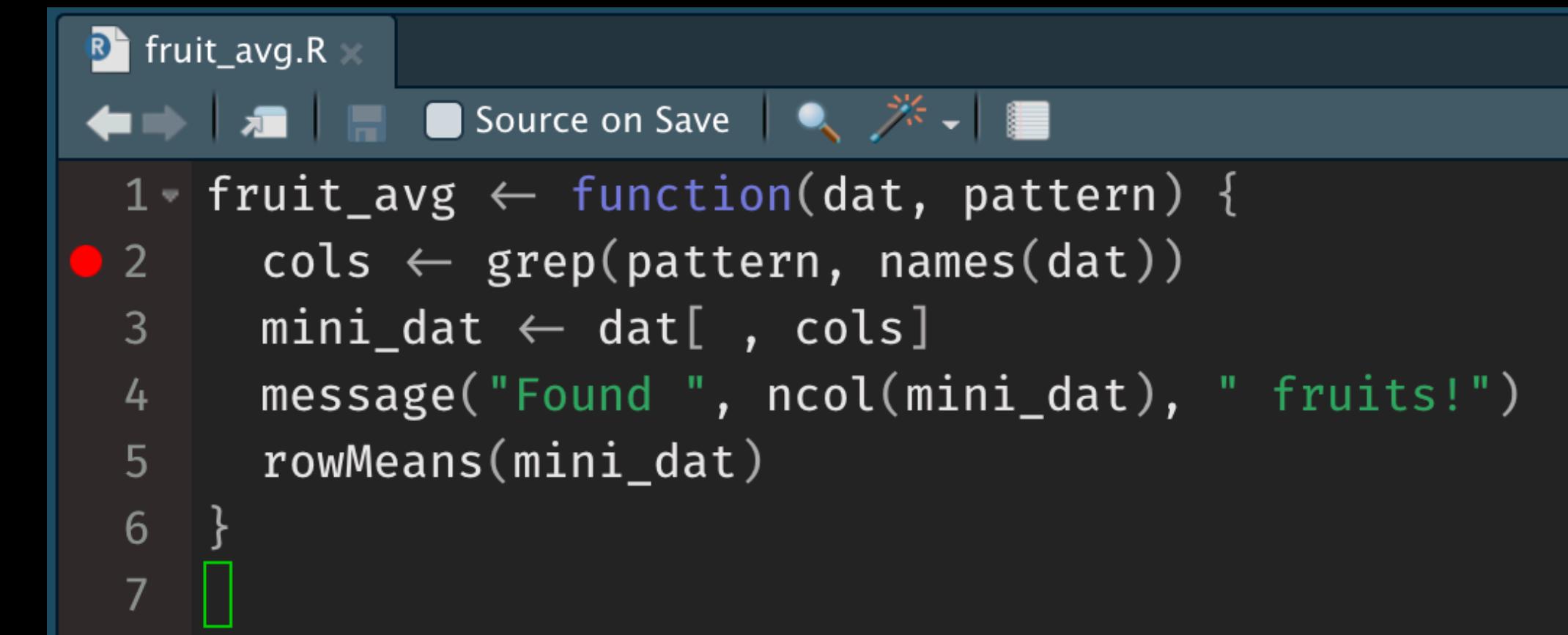
```
fruit_avg <- function(dat, pattern) {  
  cols <- grep(pattern, names(dat))  
  mini_dat <- dat[ , cols]  
  message("Found ", ncol(mini_dat), " fruits!")  
  rowMeans(mini_dat)  
}
```

```
fruit_avg <- function(dat, pattern) {  
  browser()  
  cols <- grep(pattern, names(dat))  
  mini_dat <- dat[ , cols]  
  message("Found ", ncol(mini_dat), " fruits!")  
  rowMeans(mini_dat)  
}
```

```
fruit_avg <- function(dat, pattern) {  
  browser()  
  cols <- grep(pattern, names(dat))  
  mini_dat <- dat[, cols]  
  message("Found ", ncol(mini_dat), " fruits!")  
  rowMeans(mini_dat)  
}
```

```
fruit_avg <- function(dat, pattern) {  
  browser()  
  cols <- grep(pattern, names(dat))  
  mini_dat <- dat[, cols]  
  message("Found ", ncol(mini_dat), " fruits!")  
  rowMeans(mini_dat)  
}
```

```
debug(fruit_avg)
```



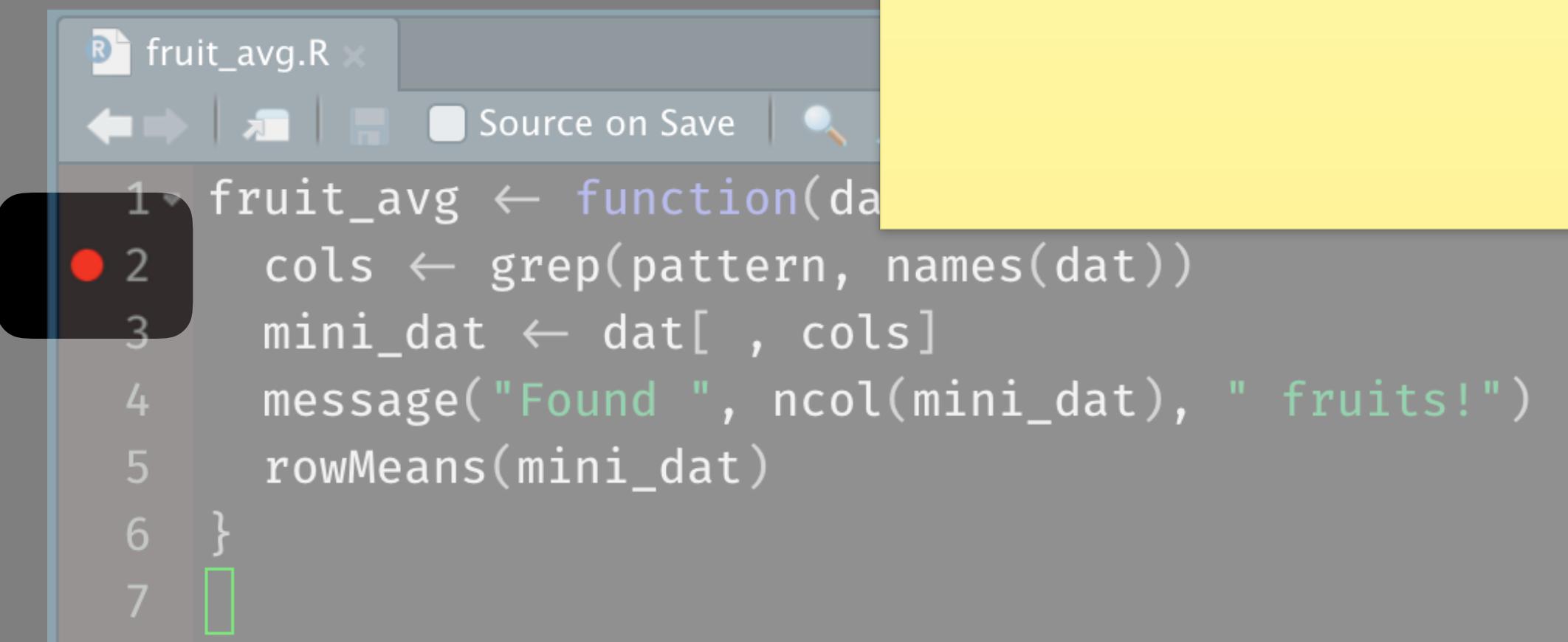
The screenshot shows an R script named "fruit\_avg.R" in an IDE. The script contains the following code:

```
1 fruit_avg <- function(dat, pattern) {  
2   cols <- grep(pattern, names(dat))  
3   mini_dat <- dat[, cols]  
4   message("Found ", ncol(mini_dat), " fruits!")  
5   rowMeans(mini_dat)  
6 }  
7
```

A red dot is positioned at the start of line 2, indicating that the debugger is currently executing that line. The IDE interface includes a toolbar with icons for back, forward, and save, and a status bar showing "Source on Save".

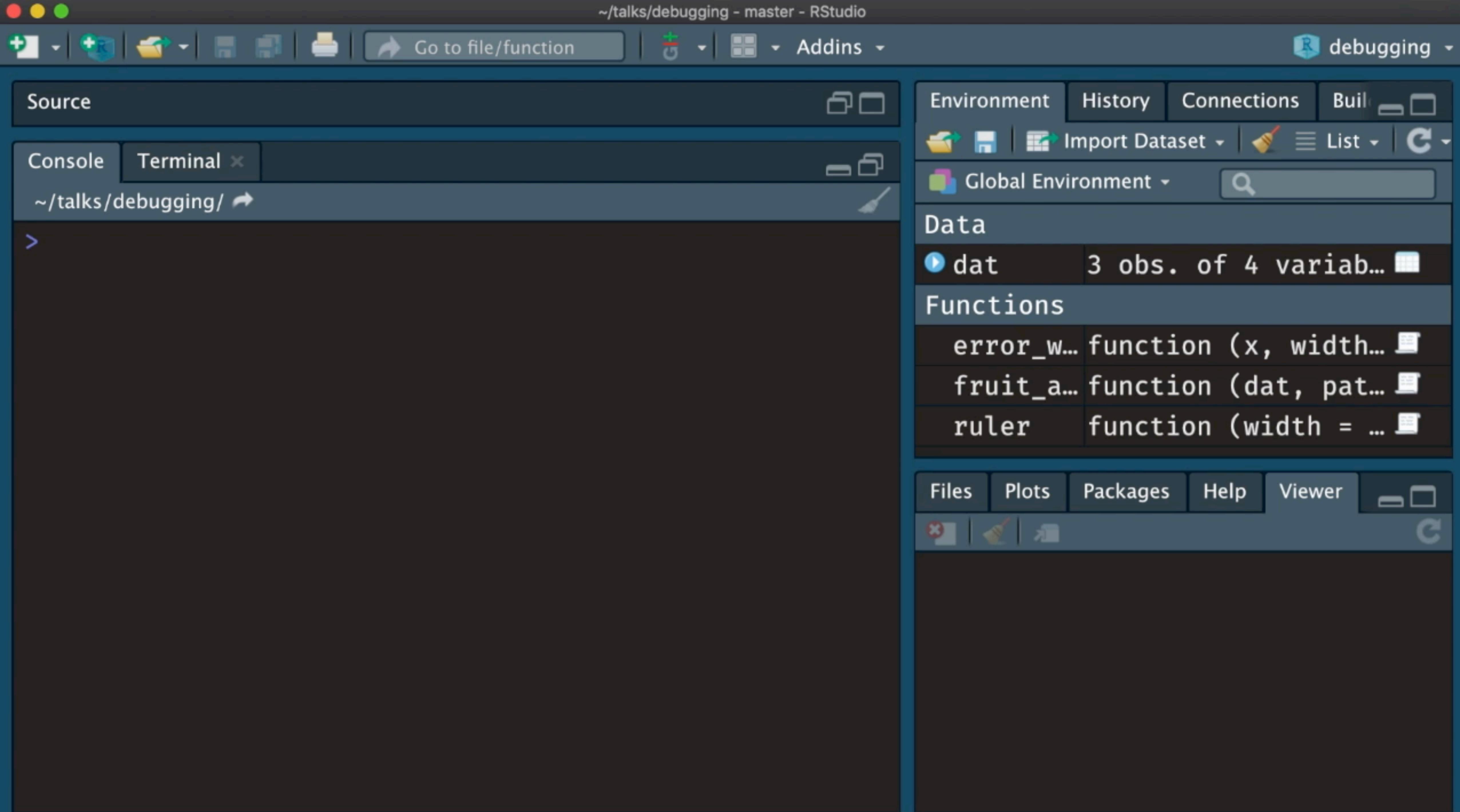
```
fruit_avg <- function(dat, pattern) {  
  browser()  
  cols <- grep(pattern, names(dat))  
  mini_dat <- dat[, cols]  
  message("Found ", ncol(mini_dat), " fruits!")  
  rowMeans(mini_dat)  
}
```

```
debug(fruit_avg)
```



```
R fruit_avg.R x  
◀ ▶ | ⚡ | ⌂ Source on Save | ⚡  
1 fruit_avg <- function(da  
2   cols <- grep(pattern, names(dat))  
3   mini_dat <- dat[, cols]  
4   message("Found ", ncol(mini_dat), " fruits!")  
5   rowMeans(mini_dat)  
6 }  
7 ┌─
```

Here's what I did: Duplicated the previous slide. Put a 50% white shape over this slide. Exported the previous slide as a PNG. Inserted the PNG on this slide. Copied it to use again later. Inserted a rounded rectangle and placed it over the first "spotlight." Selected PNG and shape. Format > Image > Mask with Selection. Pasted the PNG again. Repeat shape-mask for second spotlight. Repeat paste-shape-mask for third spotlight.



# HELP!

- ▶ Q
- ▶ RStudio ■ stop button
- ▶ debug() + undebug()
- ▶ debugonce()

I'm stuck in  
browser() and I can't  
get out!





4 DETER

**FIX IT ONCE? KEEP IT FIXED**

# ADD A TEST

```
# https://github.com/OWNER/REPO/issues/666
test_that("fruit_avg() works for 0, 1, ≥2 matches", {
  dat ← data.frame(a = 1:2, ab = 3:4, row.names = c("one", "two"))
  expect_equal(fruit_avg(dat, "abc"), c(one = NaN, two = NaN))
  expect_equal(fruit_avg(dat, "ab"), c(one = 3, two = 4))
  expect_equal(fruit_avg(dat, "a"), c(one = 2, two = 3))
})
```

# ADD AN ASSERTION

```
dat ← read.csv("fruit.csv")

if (!all(vapply(dat, is.numeric, logical(1)))) {
  stop("All columns of `dat` must be numeric")
}

fruit_avg(dat, pattern = "berry")
```

# AUTOMATE YOUR CHECKS

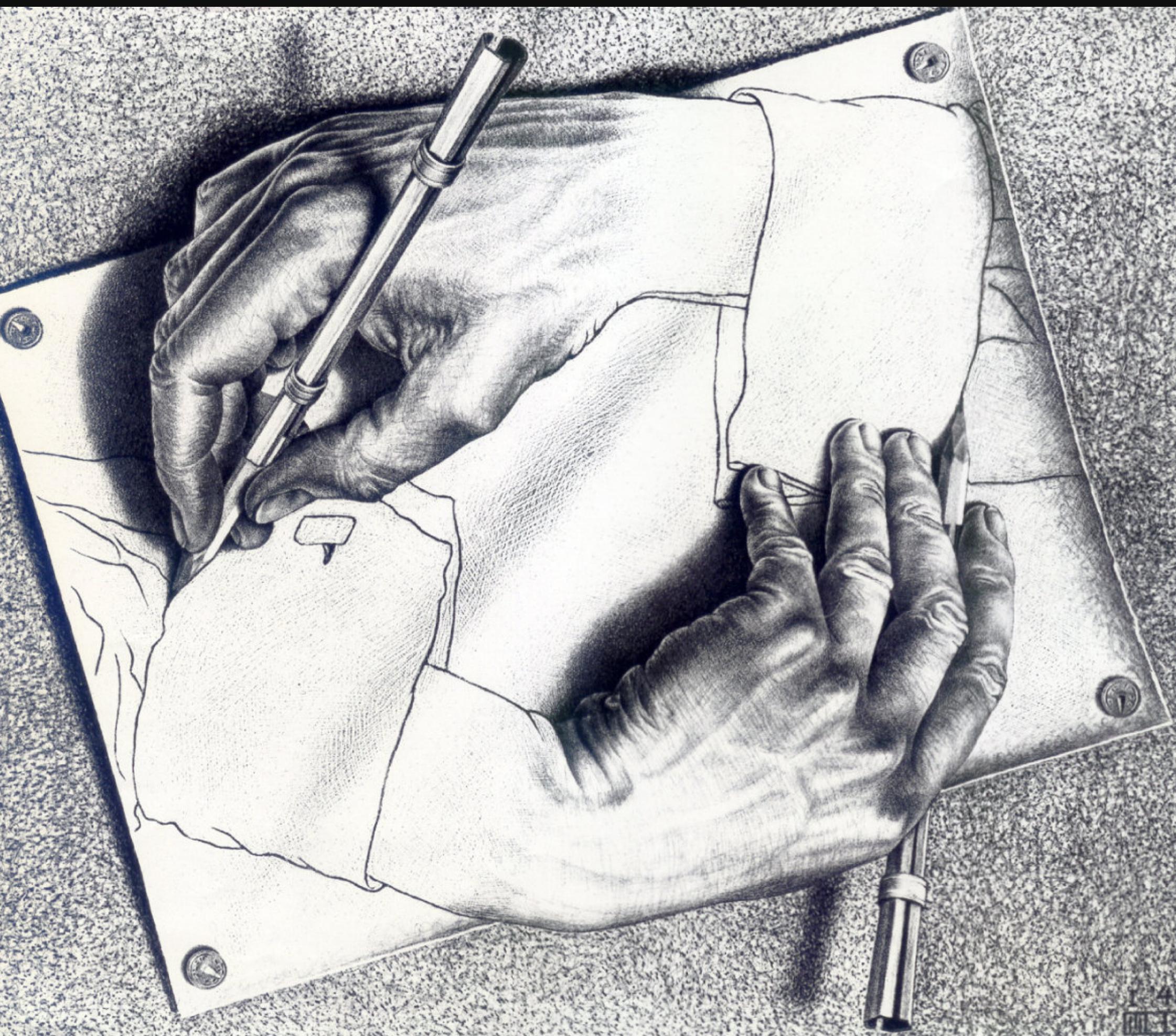
R CMD check

`testthat::test_check()`

# RUN YOUR CHECKS ON THEIR MACHINE

Continuous integration

# USE MIND BENDY STUFF IN MODERATION



**“ The major difference between a thing that might go wrong and a thing that cannot possibly go wrong is that when a thing that cannot possibly go wrong goes wrong it usually turns out to be impossible to get at and repair.**

Douglas Adams

# LEAVE ACCESS PANELS

```
readxl + xls2csv, excelgesis  
httr::with_verbose()  
rlang::qq_show()  
options(future.debug = TRUE)  
options(gargle_quiet = FALSE)  
curl::handle_setopt(h, verbose = TRUE)  
options(internet.info = 0)
```



# WRITE ERROR MESSAGES FOR HUMANS

```
dat <- data.frame(x = 1, y = 2)
```

```
df$x  
#> Error: object of type 'closure' is not subsettable
```

# WRITE ERROR MESSAGES FOR HUMANS

```
dat <- data.frame(x = 1, y = 2)
```

```
df$x  
#> Error: object of type 'function' is not subsettable
```

# WRITE ERROR MESSAGES FOR HUMANS

```
dat ← data.frame(x = 1, y = 2)

df$x
#> Error: Can't subset a function.
#> Have you forgotten to define a variable named `df`?
```

# WRITE ERROR MESSAGES FOR HUMANS

```
library(dplyr)  
  
filter(iris, Species = "setosa")  
#> Error: `Species` (`Species = "setosa"` ) must not be  
#> named, do you need `==`?
```

Turn it off and on again

Make a reprex

Dig into the error

Plan for the unexpected

Thanks:

Tidyverse team

Christine Kuper

[rstd.io/debugging](http://rstd.io/debugging)

Jennifer Bryan

RStudio

 @jennybc

 @JennyBryan

# IMAGE SOURCES

- Fret: <https://unsplash.com/photos/OsC8HauR0e0>
- Do same thing again: <https://unsplash.com/photos/uxUUENpp01I>
- Diver: <https://unsplash.com/photos/wVvxjiLJr-g>
- Ocean horizon background: <https://unsplash.com/photos/sYzFlusQp3Q>
- Calm sea background: <https://unsplash.com/photos/lZ01rjXOXQA>
- Coral reef background: <https://unsplash.com/photos/T1Wru10gKhq>
- Seaweed background: <https://unsplash.com/photos/nAkC-KS444M>
- Orchid: <https://unsplash.com/photos/Ug6z9PCwr58>
- Corn field: <https://unsplash.com/photos/nCQXxsSg3oo>
- On/off key: [https://unsplash.com/photos/cw\\_uvISXkCI](https://unsplash.com/photos/cw_uvISXkCI)

# IMAGE SOURCES CONTINUED

- Sunlight under water background: [https://unsplash.com/photos/K785Da4A\\_JA](https://unsplash.com/photos/K785Da4A_JA)
- Garnishing with sauce: <https://unsplash.com/photos/YaiY50wzWzl>
- death certificate (modified): Public Domain, <https://commons.wikimedia.org/w/index.php?curid=214170>
- washing pot: <https://unsplash.com/photos/-VhH4S1Lur8>
- The Night King: <https://cnet4.cbsistatic.com/img/vugy5MvUVBvwcJf0JvKIBd1RwJE=/1200x675/2019/04/22/2b2fee8d-111a-4d19-ae83-4e61899cf47/1nightking.jpg> (probably copyright HBO)
- Autopsy painting by Rembrandt: <https://www.mauritshuis.nl/en/explore/the-collection/artworks/the-anatomy-lesson-of-dr-nicolaes-tulp-146/detailgegevens/> Public Domain, <https://commons.wikimedia.org/w/index.php?curid=64281722>
- Gray cube abstract wallpaper vector art <https://unsplash.com/photos/1CVy8JStf3A>