

Creating reproducible examples with reprex

2018 September
rstd.io/reprex

Jennifer Bryan



 @jennybc
 @JennyBryan

`rstd.io/reprex`

`https://reprex.tidyverse.org`

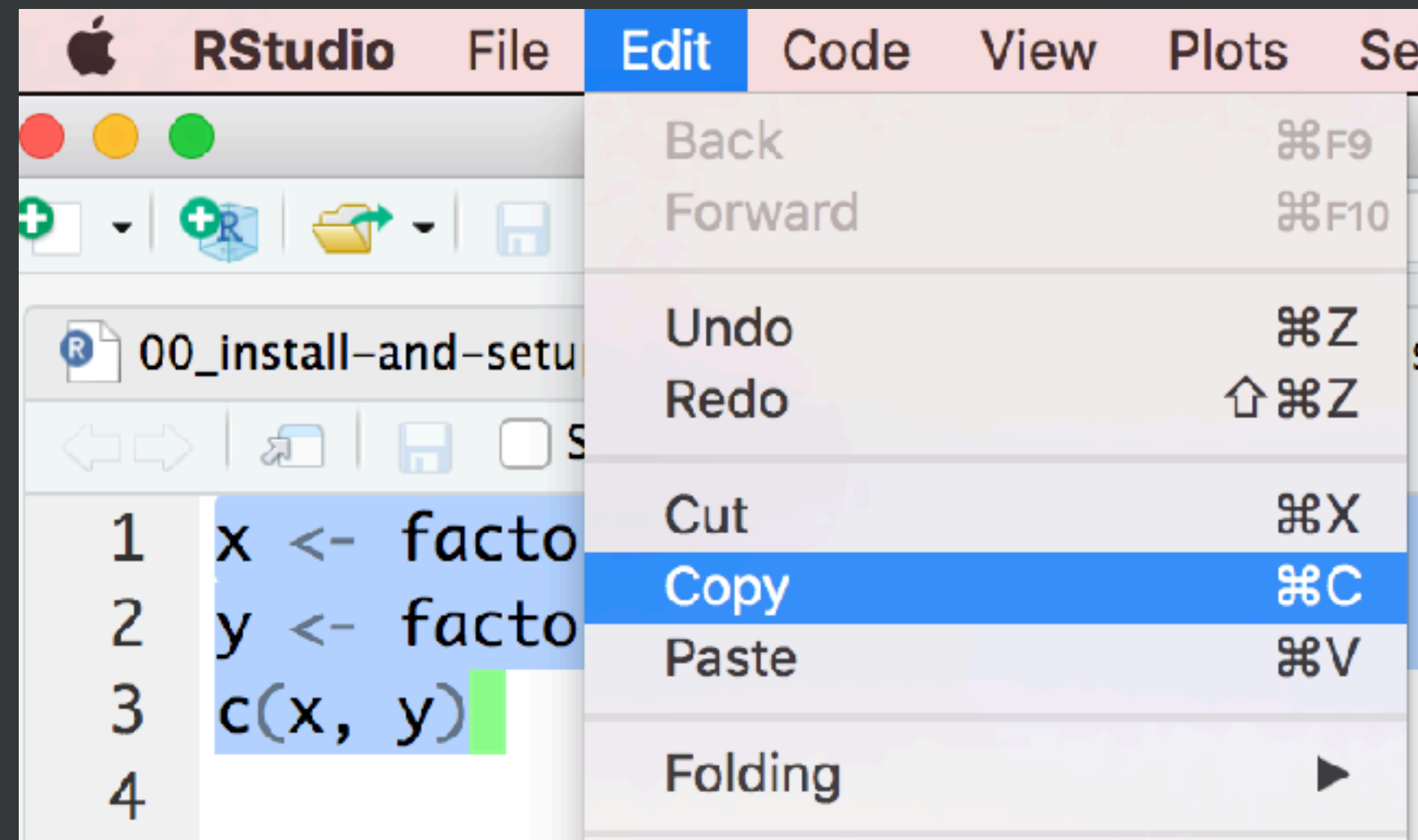
This work is licensed under a **Creative Commons Attribution-ShareAlike 4.0 International License**.

To view a copy of this license, visit
<http://creativecommons.org/licenses/by-sa/4.0/>



basic usage

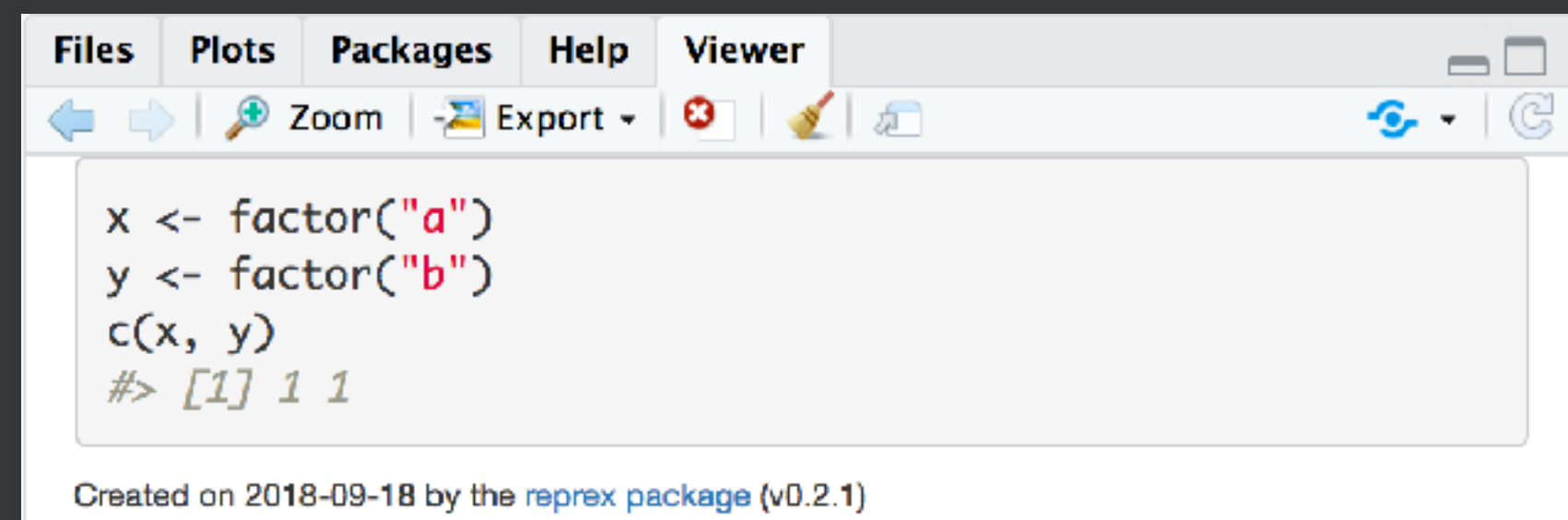
1. Copy code.



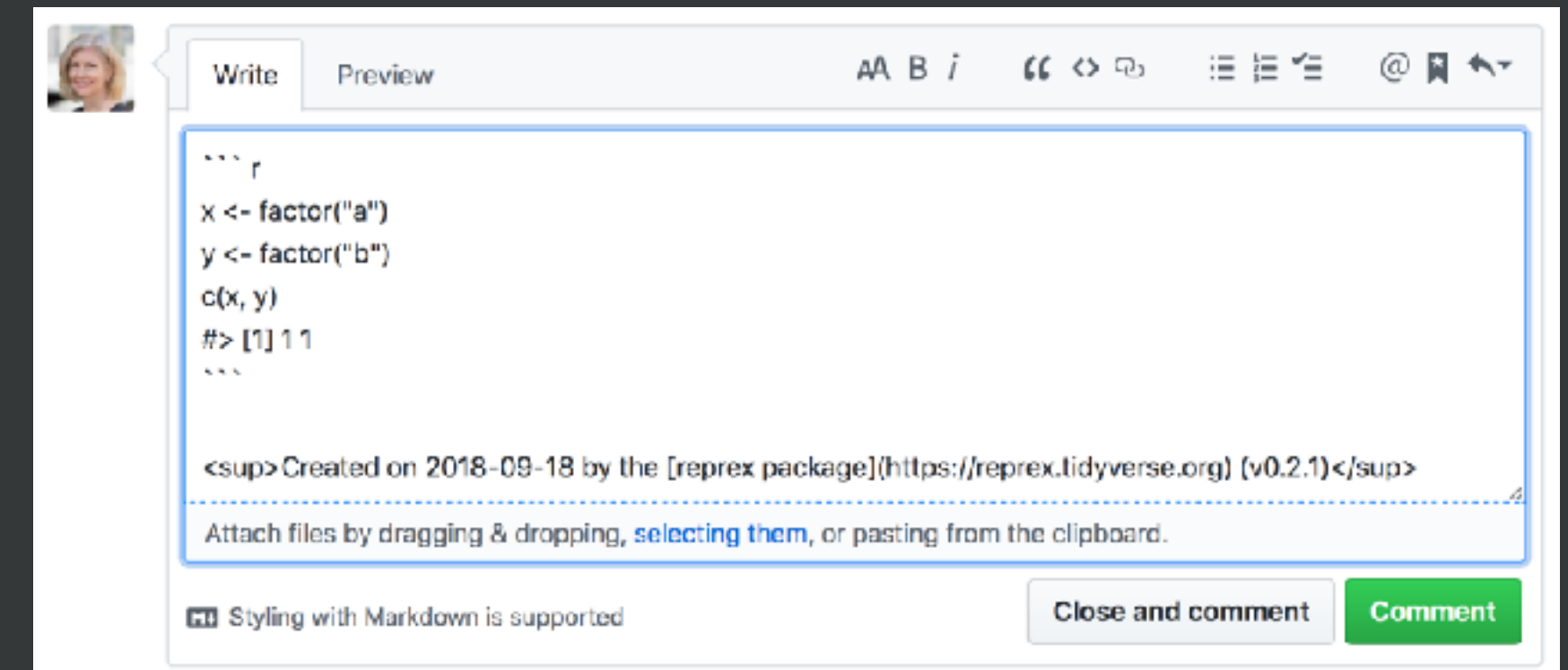
2. Run `reprex()`.

```
> reprex()
Rendering reprex...
Rendered reprex is on the clipboard.
```

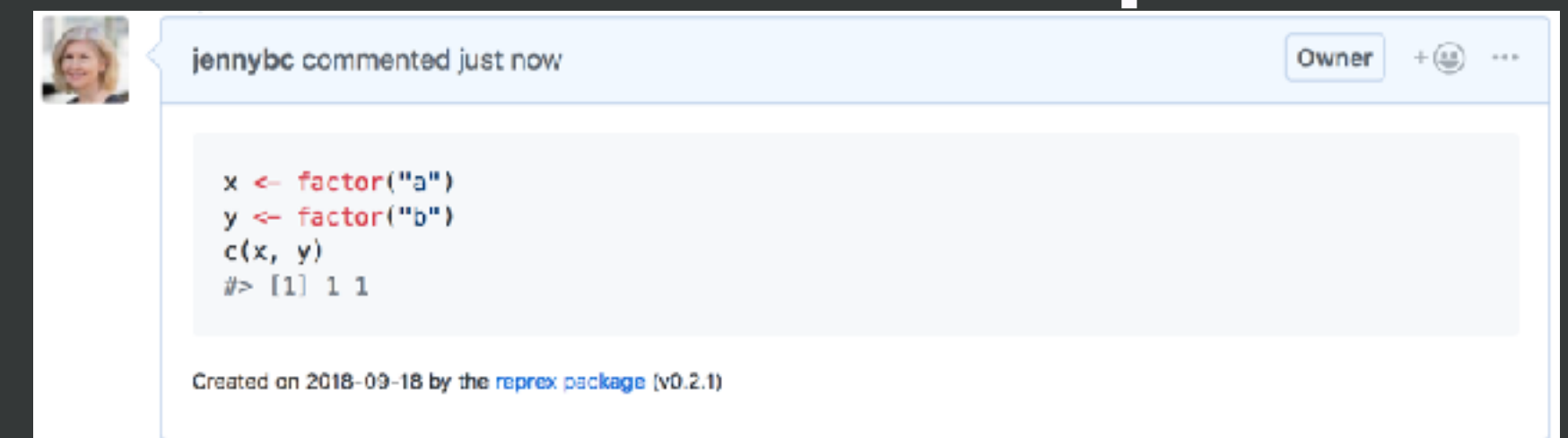
3. Admire, locally.



4. Paste into target.



5. Wait for help.





HELP ME HELP YOU



Romain François

@romain_francois

New favorite word smash: reprex, for
{rep}roducible {ex}ample. Will use it everywhere,
starting [github.com/hadley/dplyr/i...](https://github.com/hadley/dplyr/issues) cc
[@JennyBryan](#)

reprex (noun)

a reproducible example

reprex

an R package available on CRAN

reprex::reprex()

an R function in **reprex** to make a **reprex**

Include a **reprex** when you ...

1. Seek R help on community.rstudio.com
2. Ask an `[r]` question on stackoverflow.com
3. Report a bug in an R package on github.com
4. Talk about R stuff in Slack or in email

reprex :: **reprex**() makes this easier!



installation & setup

Pick one, do once per machine

```
## install JUST reprex
```

```
install.packages("reprex")
```

```
## install reprex,
```

```
## as part of the tidyverse
```

```
install.packages("tidyverse")
```

Do once per R session

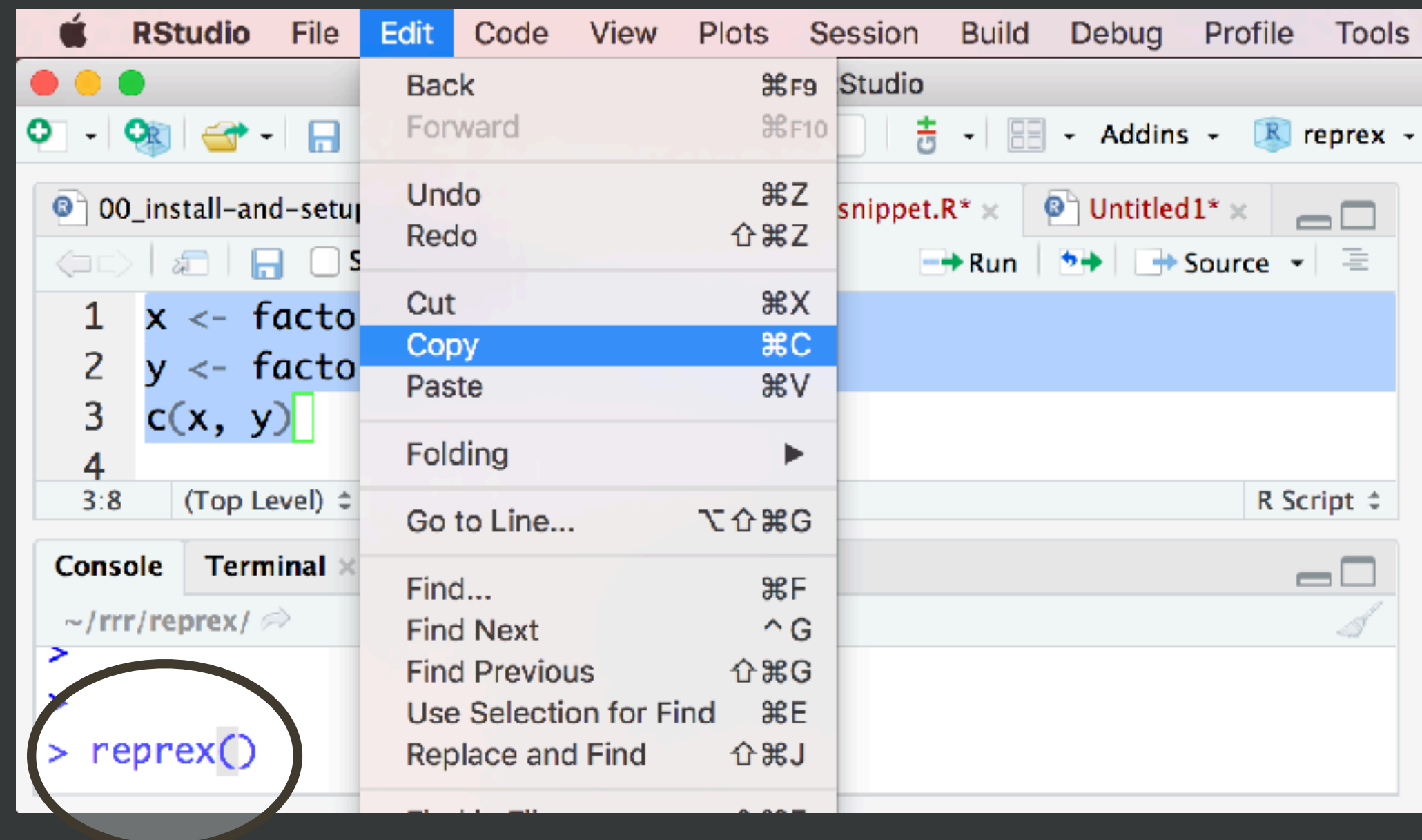
```
library(reprex)
```


Or ... do this once per machine

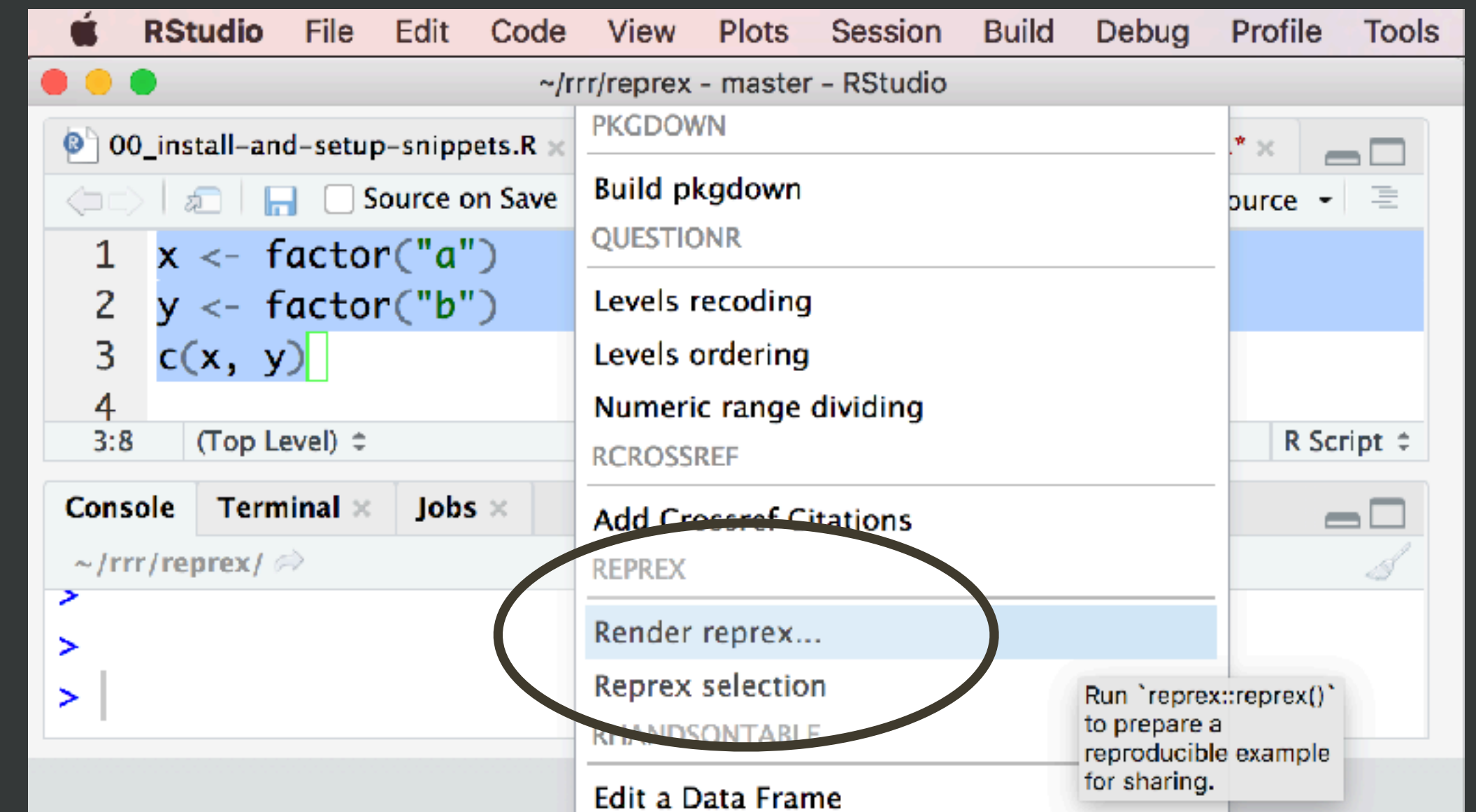
```
## put this in ~/.Rprofile to make reprex  
## available 24/7  
if (interactive()) {  
  suppressMessages(require(reprex))  
}  
  
## one way to create or open your .Rprofile  
## install.packages("usethis")  
usethis::edit_r_profile()
```

You are now ready to use `reprex::reprex()`

call in the R Console



use RStudio addin



reprer is a workflow package

you use it interactively

not in scripts, Rmd's, packages, Shiny apps

therefore, it is safe to attach via .Rprofile

do not do this with dplyr, ggplot2, etc.



reprex

www.rstudio.com

What drove
me to this?

STAT

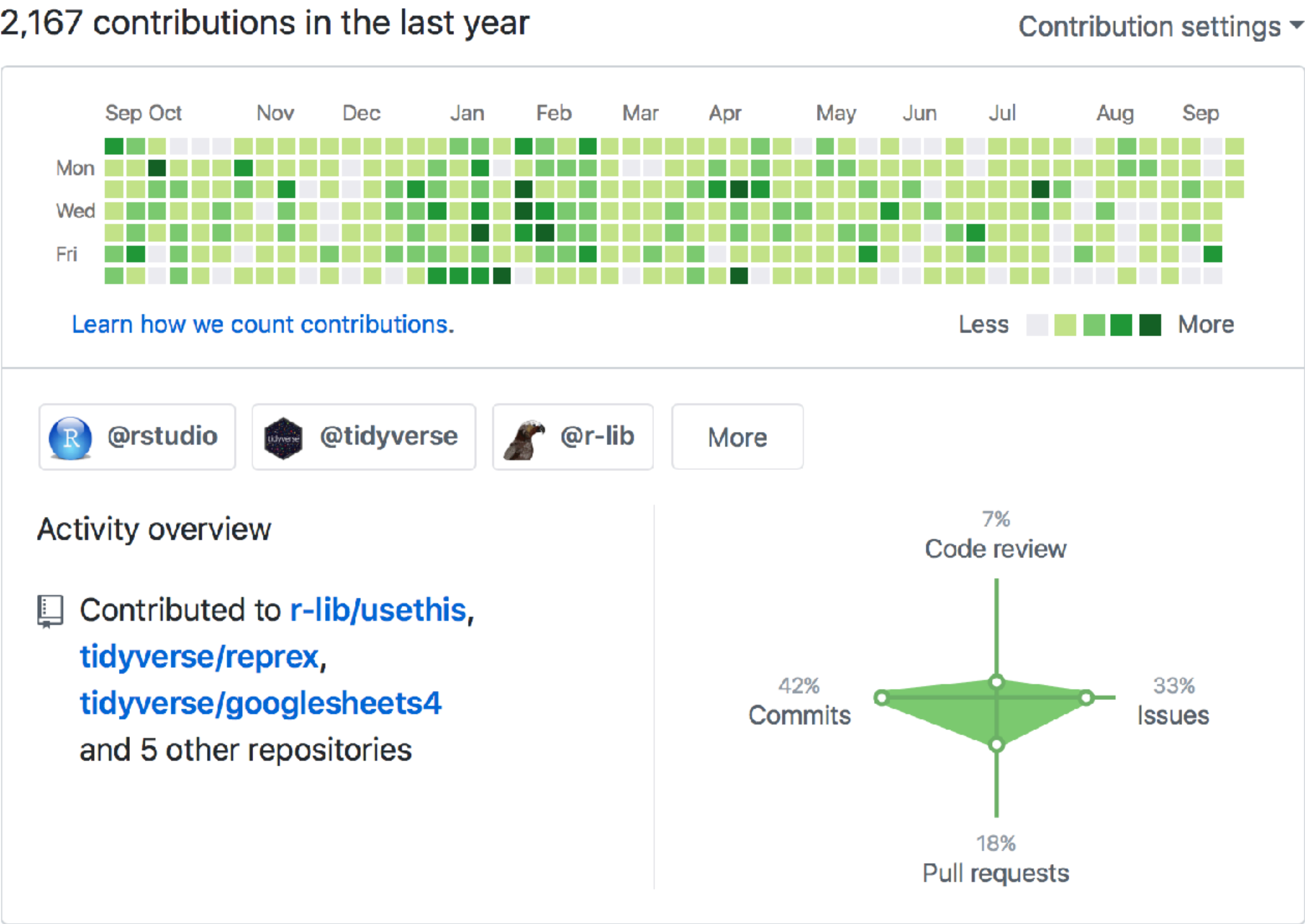
545

stat545.com

I participated in
300 - 500 R-heavy
GitHub issue threads
in Sept-Nov each year



Now, I work with R a lot on GitHub and in Slack (not shown)





reprex

www.rstudio.com

reprex
philosophy

conversations about code are more productive with:

code that **actually runs**

code that **I don't have to run**

code that **I can easily run**

code that **actually runs**

code is run in new R session

ergo, it must be self-contained!

must load all necessary packages

must create all necessary objects

not self-contained — forgot to attach necessary package

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

not self-contained — forgot to define template object

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

YAAAASSSSSS

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

<https://reprex.tidyverse.org/articles/reprex-dos-and-donts.html>

- Use the smallest, simplest, most built-in data possible.
- Include commands on a strict "need to run" basis.
- Pack it in, pack it out, and don't take liberties with other people's computers.

standard tricks for the inline
creation of a small data frame


```
x <- read.csv(text = "a,b\n1,2\n3,4")
```

```
x
```

```
#>      a b
```

```
#> 1  1 2
```

```
#> 2  3 4
```

```
x <- data.frame(
```

```
  a = c(1, 2),
```

```
  b = c(3, 4)
```

```
)
```

```
x
```

```
#>      a b
```

```
#> 1  1 3
```

```
#> 2  2 4
```

```
library(readr)
x <- read_csv("a,b\n1,2\n3,4")
x
#> # A tibble: 2 x 2
#>       a      b
#>   <dbl> <dbl>
#> 1     1     2
#> 2     3     4
```

```
library(tibble)
x <- tribble(
  ~a, ~b,
  1, 2,
  3, 4
)
x
#> # A tibble: 2 x 2
#>       a     b
#>   <dbl> <dbl>
#> 1     1     2
#> 2     3     4
```

```
x <- tibble(
  a = c(1, 2),
  b = c(3, 4)
)
x
#> # A tibble: 2 x 2
#>       a     b
#>   <dbl> <dbl>
#> 1     1     3
#> 2     2     4
```


code that I don't have to run

many readers have lots of experience

they can often get the point w/o running code,
especially if they can see the output

reveal the output produced by your code

<https://github.com/tidyverse/readr/issues/784>

matthewarbo commented on Jan 27



If a header in a csv contains quoted newlines, extra rows appear which contain part of the headers:

```
test1 <- "\"Header\\nLine Two\"\\nValue\"  
cat(test1)  
#> "Header  
#> Line Two"  
#> Value  
readr::read_csv(test1)  
#> # A tibble: 2 x 1  
#>   `Header\\nLine Two`  
#>   <chr>  
#> 1 "Line Two\\n"  
#> 2 Value
```



1



jimhester added the

bug

label on May 4

Maintainer can label this 🐛 and fellow users can 👍 this because reprex shows the output.

code that I can easily run

do not copy/paste from the R console

do not take a screenshot of your R session



```
> test1 <- "\"Header\\nLine Two\\\"\\nValue\"  
> cat(test1)  
"Header  
Line Two"  
Value  
> readr::read_csv(test1)  
# A tibble: 2 x 1  
  `Header\\nLine Two`  
  <chr>  
1 "Line Two\\\""  
2 Value
```

Do not copy/paste from the R console.
Others must make fiddly edits to reproduce.



```
Console Terminal x Jobs x
~/rrr/reprex/ ↗
> test1 <- "\"Header\\nLine Two\\\"\\nValue\"
> cat(test1)
"Header
Line Two"
Value
> readr::read_csv(test1)
# A tibble: 2 x 1
  `Header\\nLine Two`
  <chr>
1 "Line Two\\"
2 Value
> |
```

Do not take a screenshot.

Others must retype everything to reproduce.



```
test1 <- "\"Header\\nLine Two\\\"\\nValue\"  
cat(test1)  
#> "Header  
#> Line Two"  
#> Value  
readr::read_csv(test1)  
#> # A tibble: 2 x 1  
#>   `Header\\nLine Two`  
#>   <chr>  
#> 1 "Line Two\\\""  
#> 2 Value
```

A proper reprex can be re-run via copy/paste.



Or, if you *really* want *really* clean code ...

Copy from GitHub → `reprex_clean()` → Paste.

```
test1 <- "\"Header\nLine Two\"\nValue"
cat(test1)
readr::read_csv(test1)
```

See also `reprex_invert()` and `reprex_rescue()`.



Shock and Awe



live demo of ...

automatic imgur.com upload of figs

input as expression

take control of where output goes

venues: gh, so, r, rtf

ad, session info, comment

capture std out and err

demo: figure upload to imgur.com

```
## figures are uploaded to imgur.com and linked, by default
library(gapminder)
library(ggplot2)

ggplot(subset(gapminder, continent != "Oceania"),
       aes(x = year, y = lifeExp, group = country, color = country)) +
  geom_line(lwd = 1, show.legend = FALSE) + facet_wrap(~ continent) +
  scale_color_manual(values = country_colors) +
  theme_bw() + theme(strip.text = element_text(size = rel(1.1)))

## copy the above ^^ to clipboard
reprex()
## paste into, e.g., GitHub issue
## OMG the figure is there! w00t!
```

demo: rerex from an expression
see also: the `input` argument

```
## provide input as an expression  
rerex({  
  x <- rnorm(100)  
  y <- rnorm(100)  
  cor(x, y)  
})
```


demo: outfile argument to control where things go

```
## ask to work in working directory
## (vs session temp directory)
## helpful if reprex does file I/O
reprex(
  writeLines(letters[1:6]),
  outfile = NA
)

## provide a humane base for the filename
reprex(
  writeLines(letters[21:26]),
  outfile = "shock-and-awe"
)
```

demo: venue argument (default is "gh" for GitHub)

```
## render to markdown tuned to Stack Overflow (vs  
## GitHub or Discourse)
```

```
reprex(  
  mean(rnorm(100)),  
  venue = "so"  
)
```

```
## render to a commented R script  
## great for email or Slack
```

```
reprex(  
  mean(rnorm(100)),  
  venue = "r"  
)
```

```
## render to RTF to paste into Keynote or PowerPoint
```

```
reprex(  
  mean(rnorm(100)),  
  venue = "rtf"  
)
```

demo: advertise, si, style arguments

```
## suppress the "advertisement" (toggle it!)
reprex(
  mean(rnorm(100)),
  advertise = TRUE
)

## include session info (toggle it!)
reprex(
  mean(rnorm(100)),
  si = TRUE
)

## re-style the code (toggle it!)
reprex(
  input = c(
    'if (TRUE) "true branch" else {',
    '"else branch"',
    '}'
  ),
  style = TRUE
)
```

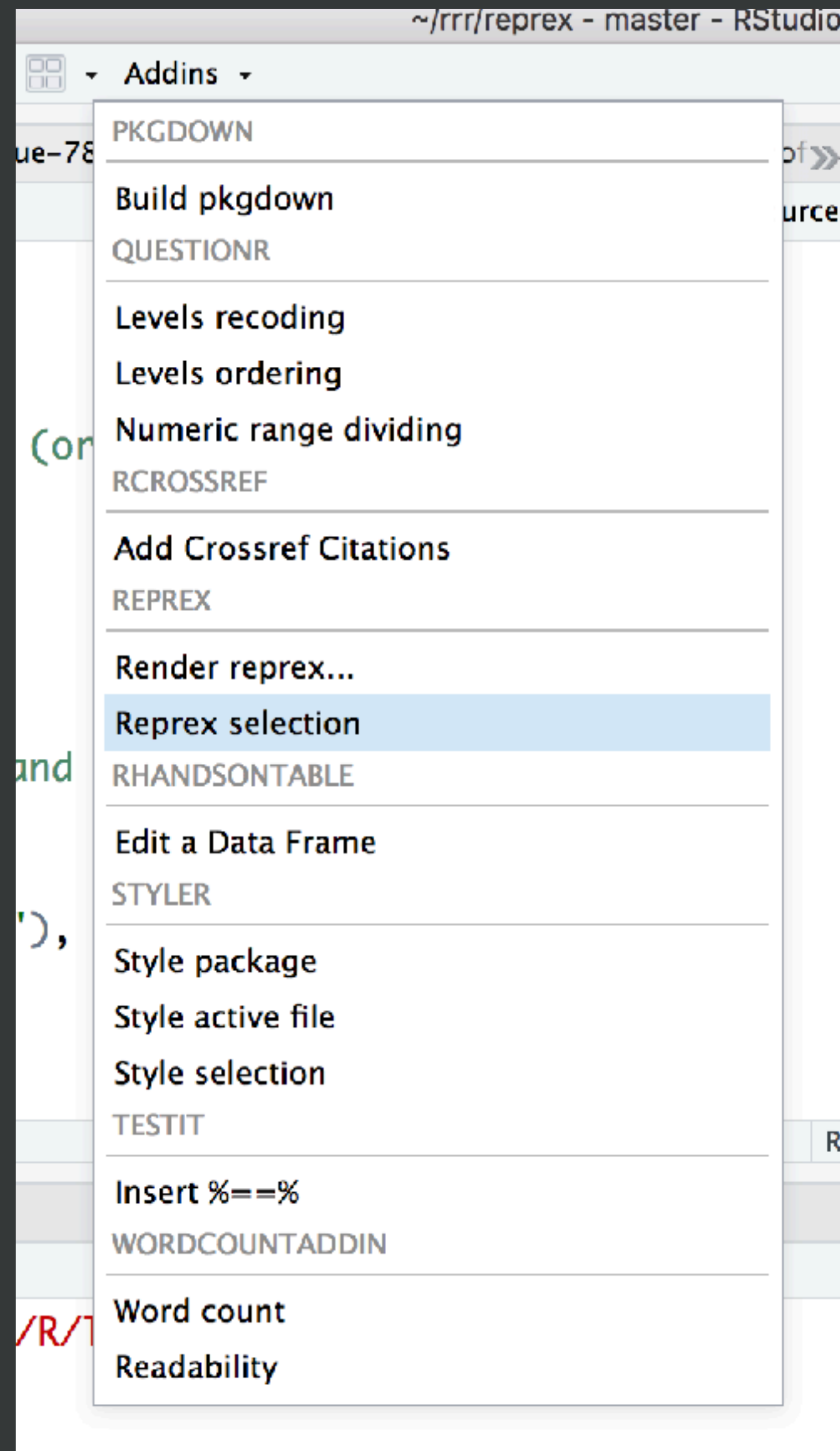
demo: std_out_err argument

```
## include output from standard output and standard error
remove.packages("bench")
reprex(
  devtools::install_github("r-lib/bench"),
  std_out_err = TRUE
)
```

Customize your defaults in .Rprofile

```
options(  
  rerex.advertise = FALSE,  
  rerex.si = TRUE,  
  rerex.style = TRUE,  
  rerex.comment = "#;-)",  
  rerex.tidyverse_quiet = FALSE  
)
```

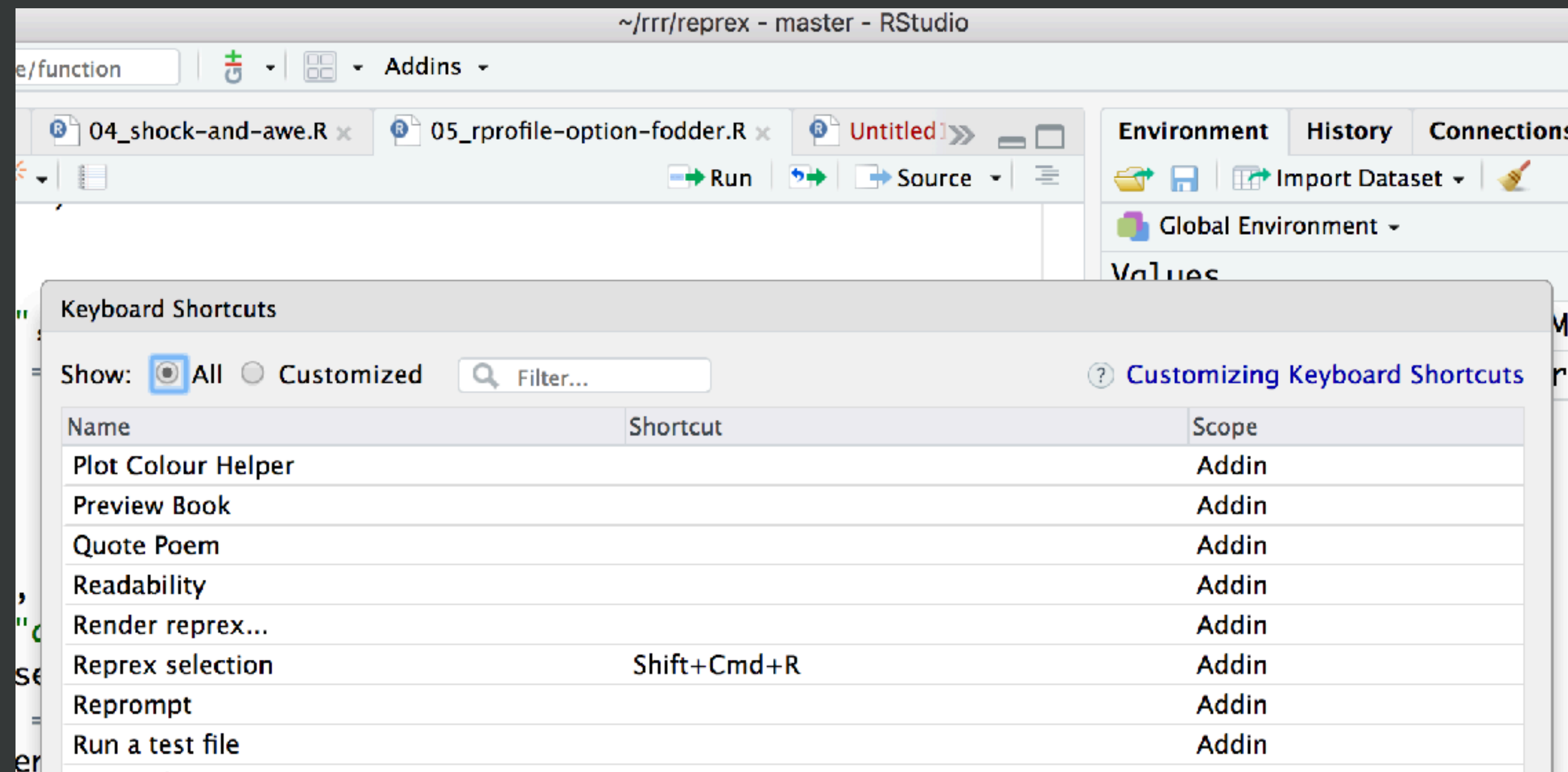

Two RStudio addins



Render reпреx... launches
a gadget, i.e. a GUI

Reprex selection is
conceived for use with a
keyboard shortcut

Tools > Modify Keyboard Shortcuts...



I bind **Reprex selection** to Shift + Cmd + R.
Hadley binds to Alt + Cmd + R.



reprex

www.rstudio.com

the
human
side



WARNING

Tough love!

Hyperbole!

Real talk!

With all the love in the world 🤗 ...

if your theory about what's wrong was so great?

we probably wouldn't be having this conversation.

Show us the code.

Have you ever helped a relative with their computer problem over the phone?

That's how it feels to answer a programming question based on a prose narrative.

Show us the code.

Assume everyone is acting in good faith.

(If not, they are irrelevant.)

True story: experts are afraid to offer a solution if they can't prove to themselves that it works.

Show us the code.

"Making a good repress is a lot of work!"

Yes, it is!

You're asking others to experience your pain.

This is how you meet them halfway.

Let's get selfish.

Making good reflexes?

Reproducing other people's problems?

Eventually ... solving them?

This is a great way to get better at programming.

Let's stay selfish.

Pleasant surprise: making a good replex often leads to solving your own problem. In private.

replex() helps you organize your attack. It forces you to strip your problem down to basics.



reprex

www.rstudio.com

what
actually
happens


```
{{{yaml}}}
```

```
{{{so_syntax_highlighting}}}
```

```
#+ reprex-setup, include = FALSE
```

```
options(tidyverse.quiet = {{{tidyverse_quiet}}})
```

```
knitr::opts_chunk$set(collapse = TRUE, comment = "{{{comment}}}", error = TRUE)
```

```
knitr::opts_knit$set(upload.fun = {{{upload_fun}}})
```

```
#+ reprex-body
```

```
{{{body}}}
```

```
{{{std_file_stub}}}
```

```
{{{ad}}}
```

```
{{{si}}}
```

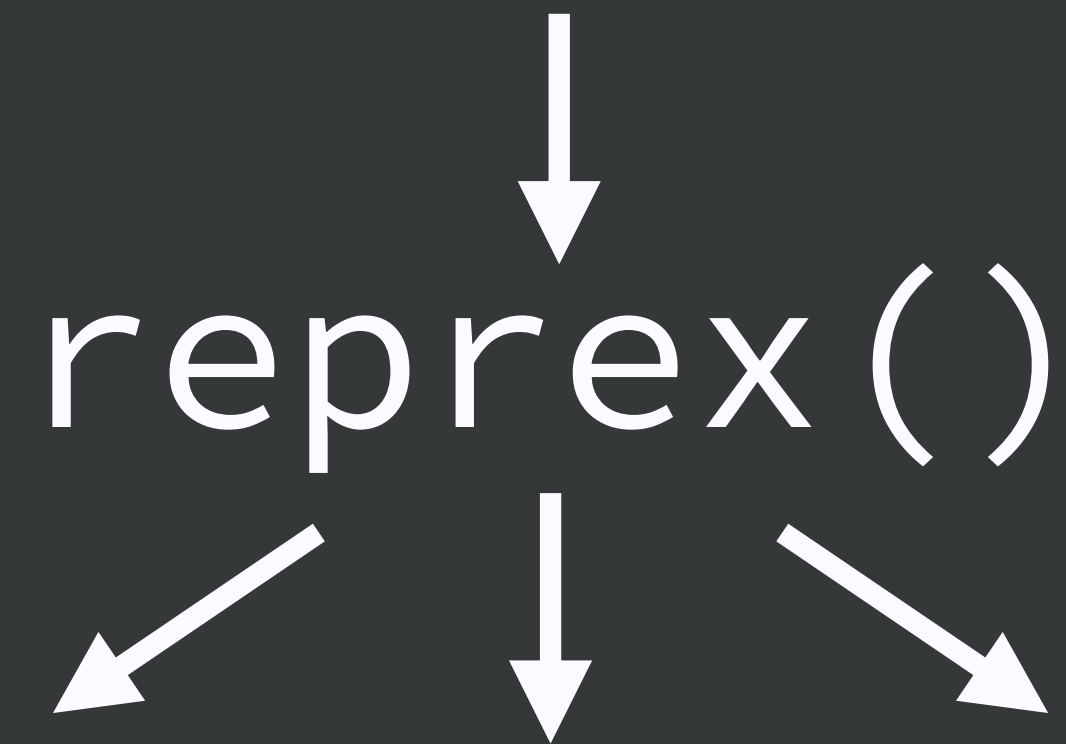
your code goes here

then .R → .md → .html

↙
?.R .rtf?

bit o' code

```
(y <- 1:4)
mean(y)
```



html preview in RStudio

```
(y <- 1:4)
#> [1] 1 2 3 4
mean(y)
#> [1] 2.5
```

Created on 2018-02-06 by the [reprex package](#) (v0.2.0).

gfm on 

```
` `` `r
(y <- 1:4)
#> [1] 1 2 3 4
...

```

as seen on GitHub

```
(y <- 1:4)
#> [1] 1 2 3 4
mean(y)
#> [1] 2.5
```

Created on 2018-02-06 by the [reprex package](#) (v0.2.0).

bit o' code


```
(y <- 1:4)
mean(y)
```

↓

```
reprex(venue = "so")
```

↙ ↓ ↘

html preview in RStudio



The screenshot shows the RStudio interface with the 'Viewer' pane displaying the HTML output of the R code. The code is:

```
(y <- 1:4)
#> [1] 1 2 3 4
mean(y)
#> [1] 2.5
```

 The output is rendered as a code block with syntax highlighting. At the bottom, it says 'Created on 2018-02-06 by the reprex package (v0.2.0).'

SO md on 

```
<!-- language-all: lang-r -->
```

```
(y <- 1:4)
#> [1] 1 2 3 4
```

...

as seen on StackOverflow



The screenshot shows a snippet of R code and its output as it might appear on StackOverflow. The code is:

```
(y <- 1:4)
#> [1] 1 2 3 4
mean(y)
#> [1] 2.5
```

 Below the code, it says 'Created on 2018-02-06 by the reprex package (v0.2.0).'

bit o' code

```
(y <- 1:4)
mean(y)
```

↓

```
reprex(venue = "r")
```

↙ ↓ ↘

html preview in RStudio



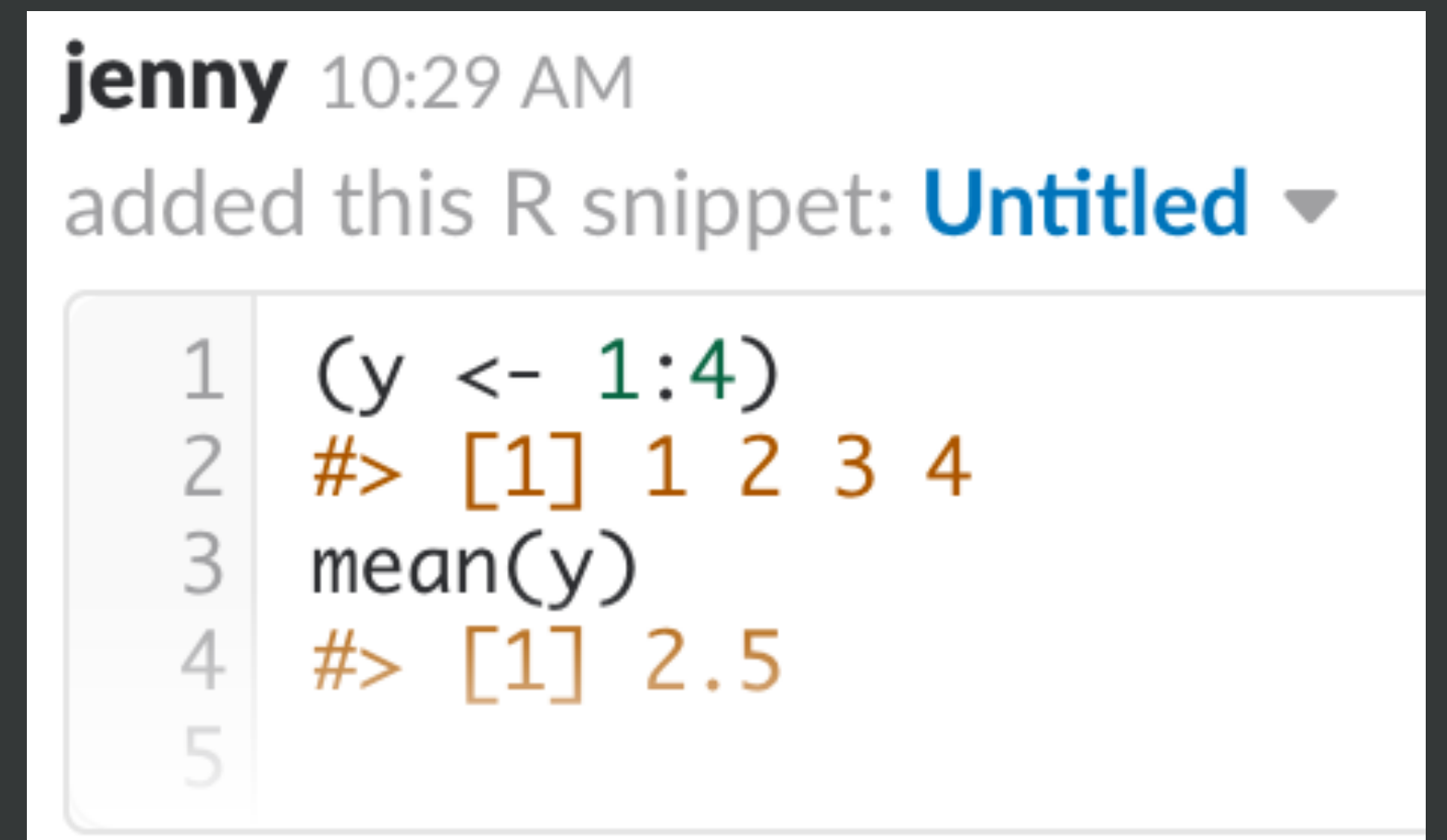
```
(y <- 1:4)
#> [1] 1 2 3 4
mean(y)
#> [1] 2.5
```

Created on 2018-02-06 by the [reprex package](#) (v0.2.0).

commented R on 

```
(y <- 1:4)
#> [1] 1 2 3 4
mean(y)
#> [1] 2.5
...
```

as Slack R snippet



jenny 10:29 AM
added this R snippet: **Untitled** ▼

```
1 (y <- 1:4)
2 #> [1] 1 2 3 4
3 mean(y)
4 #> [1] 2.5
5
```

Huge 🙏 to Yihui Xie and all those who bring us
rmarkdown and Pandoc

reprer is "just" a wrapper around those things 😊

All reprer co-authors, contributors, users



reprex

www.rstudio.com

engage in Q & A
report bugs
request features
be a chatty R nerd!