Useful commands in base R

```
debug(), undebug()
debugonce()
browser()
traceback()
options (error = browser), options (error = NULL)
```

Debugging Shiny applications

Automatic traceback() in error output, in RStudio and application log Can set a breakpoint in the server function

Use browser () everywhere else (ui, sourced file, etc.)

Open traceback to examine

the functions that R called

before the error occurred

options(shiny.error = browser)

Tracing: cat(file=stderr(),...), options(shiny.reactlog=TRUE)

Debug mode in RStudio (from the IDE cheat sheet)

Open with debug(), browse(), or a breakpoint. RStudio will open the debugger mode when it encounters a breakpoint while executing code.

Click next to ... line number to digit1 for (x in 1:((digits %/% 2))) 8 digits add/remove a digit2 <- get_digit(num, (digits + 1) - x) breakpoint. if (digit1 != digit2) return(TRUE) Highlighted 20 - biggest palindrome <- function() { 21 best <- 8 line shows (num%(18^n))%/%(18^(n - 1)) execution has debug at ~/RStudio-Essentials/Essentials-2/palindr

Run commands in environment where execution has paused

where

paused

Examine variables Select function in executing

in traceback to debug

Step through code one line at a time

Step into and to run

Console ~/IDEcheatsheet/ @

Launch debugger

mode from origin

Console ~/IDEcheatsheet/ <

of error

> foo()

Error!

Resume Quit debug out of functions execution mode

Rerun with Debug

Stop

Resources

Debugging with RStudio https://support.rstudio.com/hc/e n-us/articles/205612627-Debug ging-with-RStudio

Debugging Shiny applications https://shiny.rstudio.com/article s/debugging.html

"Debugging, condition handling, and defensive programming" in Advanced R

http://adv-r.had.co.nz/Exception s-Debugging.html



https://github.com/rstudio/cheatsheets/raw/master/rstudio-ide.pdf

environment