

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

Debug mode in RStudio (from the IDE cheat sheet)

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

Click next to line number to add/remove a breakpoint.

Highlighted line shows where execution has paused

Run commands in environment where execution has paused

Examine variables in executing environment

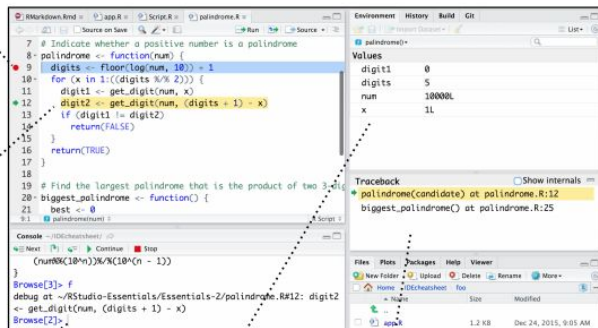
Select function in traceback to debug

Step through code one line at a time

Step into and out of functions to run

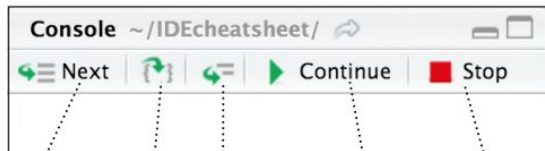
Resume execution mode

Quit debug



Launch debugger mode from origin of error

Open traceback to examine the functions that R called before the error occurred



Resources

Debugging with RStudio

<https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio>

Debugging Shiny applications

<https://shiny.rstudio.com/articles/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>

