

Interactive debugging with Environment Browser

Leon Kim | www.betweentwotests.com

Overview of `base::browser()`

- ▶ This is a brief overview of `base::browser()` and how to use it for debugging in R.
- ▶ `browser()` is default function found in R.
- ▶ Useful tool to “hack into” R functions (that you or other people wrote)

Typical day in lab

```
head(iris)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	5.1	3.5	1.4	0.2	setosa
## 2	4.9	3.0	1.4	0.2	setosa
## 3	4.7	3.2	1.3	0.2	setosa
## 4	4.6	3.1	1.5	0.2	setosa
## 5	5.0	3.6	1.4	0.2	setosa
## 6	5.4	3.9	1.7	0.4	setosa

- Task: Researcher's ruler was off by 5 inches. Let's adjust it!

Oops

```
# Loop through each column and add 5
df <- iris
for(j in 1:ncol(df)){
  df[,j] <- df[,j] + 5
}
```

```
## Warning in Ops.factor(df[, j], 5): '+' not meaningful for factors
```

```
unlist(lapply(iris, class))
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##      "numeric"      "numeric"      "numeric"      "numeric"      "factor"
```

- Forgot that Species column is not numeric!

Quick and dirty solution

Loop through each column EXCEPT the last one

```
df <- iris
for(j in 1:(ncol(df)-1)){
  df[,j] <- df[,j] + 5
}
head(df)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	10.1	8.5	6.4	5.2	setosa
## 2	9.9	8.0	6.4	5.2	setosa
## 3	9.7	8.2	6.3	5.2	setosa
## 4	9.6	8.1	6.5	5.2	setosa
## 5	10.0	8.6	6.4	5.2	setosa
## 6	10.4	8.9	6.7	5.4	setosa

We got lucky

- ▶ Not all error messages are obvious
- ▶ Not all bugs come from such easy code
- ▶ Not all bugs are this easy to fix

More likely scenario

```
foo_1 <- function(x) { ... }  
foo_2 <- function(x) { ... }  
# Code that produces some error  
for(j in 1:ncol(df)){  
  foo_1(df[,j]); foo_2(df[,j])  
}
```

```
foos <- function(x) {  
  foo_1(foo_2(x))  
}  
super_foo <- function(df) {  
  # Code that produces some error  
  for(j in 1:ncol(df)){  
    foos(df[,j])  
  }  
}  
super_foo(iris)
```

browser() to the rescue

```
for(j in 1:ncol(df)){  
  browser() <-----  
  foo_1(df[,j])  
  foo_2(df[,j])  
}
```


Example

```
add_five <- function(x) {x + 5}
df <- iris
outside_var <- "I'm outside the loop"
for(j in 1:ncol(df)){
  print(paste("start of the iteration:", j))
  inside_var <- "I'm inside the loop"
  browser()
  df[,j] <- add_five(df[,j])
  print(paste("end of the interation:", j))
}
```

Trigger browser() mode

► Run your code

```
> add_five <- function(x) {x + 5}
> df <- iris
> outside_var <- "I'm outside the loop"
> for(j in 1:ncol(df)){
+   print(paste("start of the iteration:", j))
+   inside_var <- "I'm inside the loop"
+   browser()
+   df[,j] <- add_five(df[,j])
+   print(paste("end of the iteration:", j))
+ }
[1] "start of the iteration: 1"
Called from: top level
Browse[1]> |
```

► Run expressions within browser() mode

```
Browse[1]> outside_var; j; inside_var
[1] "I'm outside the loop"
[1] 1
[1] "I'm inside the loop"
Browse[1]> j+1
[1] 2
Browse[1]> head(df)
  Sepal.Length Sepal.width Petal.Length Petal.width Species
1          5.1         3.5         1.4         0.2  setosa
2          4.9         3.0         1.4         0.2  setosa
3          4.7         3.2         1.3         0.2  setosa
4          4.6         3.1         1.5         0.2  setosa
5          5.0         3.6         1.4         0.2  setosa
6          5.4         3.9         1.7         0.4  setosa
Browse[1]>
```

What can I do in browser() mode?

- Type `help` for list of available commands

```
Browse[1]> help
n          next
s          step into
f          finish
c or cont  continue
Q          quit
where      show stack
help       show help
<expr>     evaluate expression
Browse[1]> c
```

'c': execute all of the function

- Executes everything left (until the next browser() call that is)

```
Browse[1]> c  
[1] "end of the iteration: 1"  
[1] "start of the iteration: 2"  
Called from: top level  
Browse[1]>
```

- Sanity check: has the first column of df been changed?

```
Browse[1]> j; df[1:5, 1:2]; head(iris[1:5, 1:2])  
[1] 2  
  Sepal.Length Sepal.width  
1      10.1      3.5  
2       9.9      3.0  
3       9.7      3.2  
4       9.6      3.1  
5      10.0      3.6  
  Sepal.Length Sepal.width  
1       5.1      3.5  
2       4.9      3.0  
3       4.7      3.2  
4       4.6      3.1  
5       5.0      3.6  
Browse[1]>
```

'n': Go to the next line

- Moves to beginning of the next line

```
Browse[1]> n
debug at #5: df[, j] <- add_five(df[, j])
Browse[1]> j; df[1:5, 1:2]; head(iris[1:5, 1:2])
[1] 2
  Sepal.Length Sepal.width
1         10.1         3.5
2          9.9         3.0
3          9.7         3.2
4          9.6         3.1
5         10.0         3.6
  Sepal.Length Sepal.width
1          5.1         3.5
2          4.9         3.0
3          4.7         3.2
4          4.6         3.1
5          5.0         3.6
```

- Also executes your current line (before moving to next line)

```
Browse[1]> n
debug at #6: print(paste("end of the iteration:", j))
Browse[1]> j; df[1:5, 1:2]; head(iris[1:5, 1:2])
[1] 2
  Sepal.Length Sepal.width
1         10.1         8.5
2          9.9         8.0
3          9.7         8.2
4          9.6         8.1
5         10.0         8.6
  Sepal.Length Sepal.width
1          5.1         3.5
2          4.9         3.0
3          4.7         3.2
4          4.6         3.1
5          5.0         3.6
Browse[1]> |
```

's': R-reception

► Go into the function call

```
Browse[1]> c  
[1] "end of the iteration: 2"  
[1] "start of the iteration: 3"  
Called from: top level  
Browse[1]> n  
debug at #5: df[, j] <- add_five(df[, j])  
Browse[1]> s  
debugging in: add_five(df[, j])  
debug at #1: {  
  x + 5  
}  
Browse[2]> |
```

In RStudio

- GUI supports (for function calls)

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains a function definition `add_five(x) {x + 5}`. A yellow warning bar indicates "Debug location is approximate because the source is not available." The function is currently selected.
- Console:** Shows the execution of `add_five(df[, j])`. It displays the function's output, a data frame with columns `Sepal.Length` and `Sepal.width`, and the execution path through the function's body.
- Environment:** Shows the current environment with variables `x` and `df[, j]`.
- Traceback:** Shows the call stack, highlighting the function `add_five(df[, j])`.
- Environment Browser:** Displays the current environment, showing the base environment and the current environment.

The console output shows the function's execution path, including the function definition and the execution of the function body. The output is a data frame with columns `Sepal.Length` and `Sepal.width`.

```
3      9.7      8.2
4      9.6      8.1
5     10.0      8.6
Sepal.Length Sepal.width
1      5.1      3.5
2      4.9      3.0
3      4.7      3.2
4      4.6      3.1
5      5.0      3.6
Browse[1]> c
[1] "end of the iteration: 2"
[1] "start of the iteration: 3"
Called from: top level
Browse[1]> n
debug at #5: df[, j] <- add_five(df[, j])
Browse[1]> s
debugging in: add_five(df[, j])
debug at #1: {
  x + 5
}
Browse[2]> |
```

Access everything inside the function environment

► Access variables as the function sees it

```
debug at #5: df[, j] <- add_five(df[, j])
```

```
Browse[1]> s
```

```
debugging in: add_five(df[, j])
```

```
debug at #1: {
```

```
  x + 5
```

```
}
```

```
Browse[2]> x
```

```
[1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1 1.2 1.5 1.3 1.4 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5  
[29] 1.4 1.6 1.6 1.5 1.5 1.4 1.5 1.2 1.3 1.4 1.3 1.5 1.3 1.3 1.3 1.6 1.9 1.4 1.6 1.4 1.5 1.4 4.7 4.5 4.9 4.0 4.6 4.5  
[57] 4.7 3.3 4.6 3.9 3.5 4.2 4.0 4.7 3.6 4.4 4.5 4.1 4.5 3.9 4.8 4.0 4.9 4.7 4.3 4.4 4.8 5.0 4.5 3.5 3.8 3.7 3.9 5.1  
[85] 4.5 4.5 4.7 4.4 4.1 4.0 4.4 4.6 4.0 3.3 4.2 4.2 4.2 4.3 3.0 4.1 6.0 5.1 5.9 5.6 5.8 6.6 4.5 6.3 5.8 6.1 5.1 5.3  
[113] 5.5 5.0 5.1 5.3 5.5 6.7 6.9 5.0 5.7 4.9 6.7 4.9 5.7 6.0 4.8 4.9 5.6 5.8 6.1 6.4 5.6 5.1 5.6 6.1 5.6 5.5 4.8 5.4  
[141] 5.6 5.1 5.1 5.9 5.7 5.2 5.0 5.2 5.4 5.1
```

```
Browse[2]> df[,3]
```

```
[1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1 1.2 1.5 1.3 1.4 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5  
[29] 1.4 1.6 1.6 1.5 1.5 1.4 1.5 1.2 1.3 1.4 1.3 1.5 1.3 1.3 1.3 1.6 1.9 1.4 1.6 1.4 1.5 1.4 4.7 4.5 4.9 4.0 4.6 4.5  
[57] 4.7 3.3 4.6 3.9 3.5 4.2 4.0 4.7 3.6 4.4 4.5 4.1 4.5 3.9 4.8 4.0 4.9 4.7 4.3 4.4 4.8 5.0 4.5 3.5 3.8 3.7 3.9 5.1  
[85] 4.5 4.5 4.7 4.4 4.1 4.0 4.4 4.6 4.0 3.3 4.2 4.2 4.2 4.3 3.0 4.1 6.0 5.1 5.9 5.6 5.8 6.6 4.5 6.3 5.8 6.1 5.1 5.3  
[113] 5.5 5.0 5.1 5.3 5.5 6.7 6.9 5.0 5.7 4.9 6.7 4.9 5.7 6.0 4.8 4.9 5.6 5.8 6.1 6.4 5.6 5.1 5.6 6.1 5.6 5.5 4.8 5.4  
[141] 5.6 5.1 5.1 5.9 5.7 5.2 5.0 5.2 5.4 5.1
```

```
Browse[2]>
```


Effectively use browser() : #1

```
add_five <- function(x) {x + 5}  
df <- iris  
for(j in 1:ncol(df)){  
  if(j == 5) {  
    browser()  
  }  
  df[,j] <- add_five(df[,j])  
}
```

Effectively use browser() : #2

```
library(dplyr)
df <- iris
df %>%
  group_by(Species) %>%
  mutate(avg_sepal_length =
    list(Species = Species,
         Sepal.Length = Sepal.Length) %>%
    browser()
  )
```

Look at individual group

► Pass data to browser()

```
> library(dplyr)
> df <- iris
> df %>%
+   group_by(Species) %>%
+   mutate(avg_sepal_length = list(Species = Species, Sepal.Length = Sepal.Length)) %>% browser()
called from: function_list[[k]](value)
Browse[1]> .
$Species
 [1] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[15] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[29] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[43] setosa setosa setosa setosa setosa setosa setosa setosa
Levels: setosa versicolor virginica

$Sepal.Length
 [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8
[26] 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0

Browse[1]> n
called from: function_list[[k]](value)
Browse[1]> .
$Species
 [1] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[10] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[19] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[28] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[37] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[46] versicolor versicolor versicolor versicolor versicolor
Levels: setosa versicolor virginica

$Sepal.Length
 [1] 7.0 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1 6.3 6.1 6.4
[26] 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7
```

- ▶ `browser()` allows you to interactively code in any environment scope during R execution