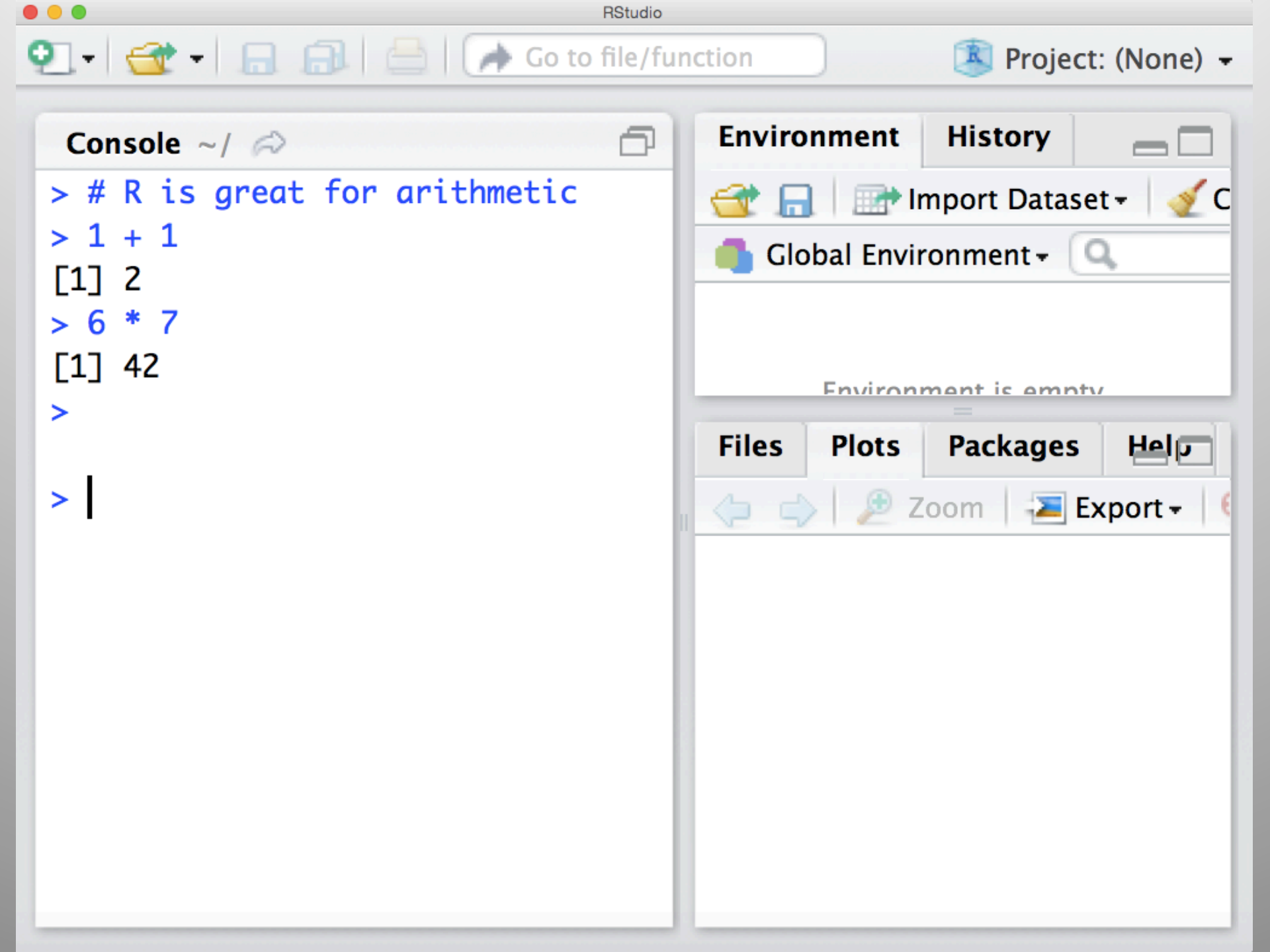
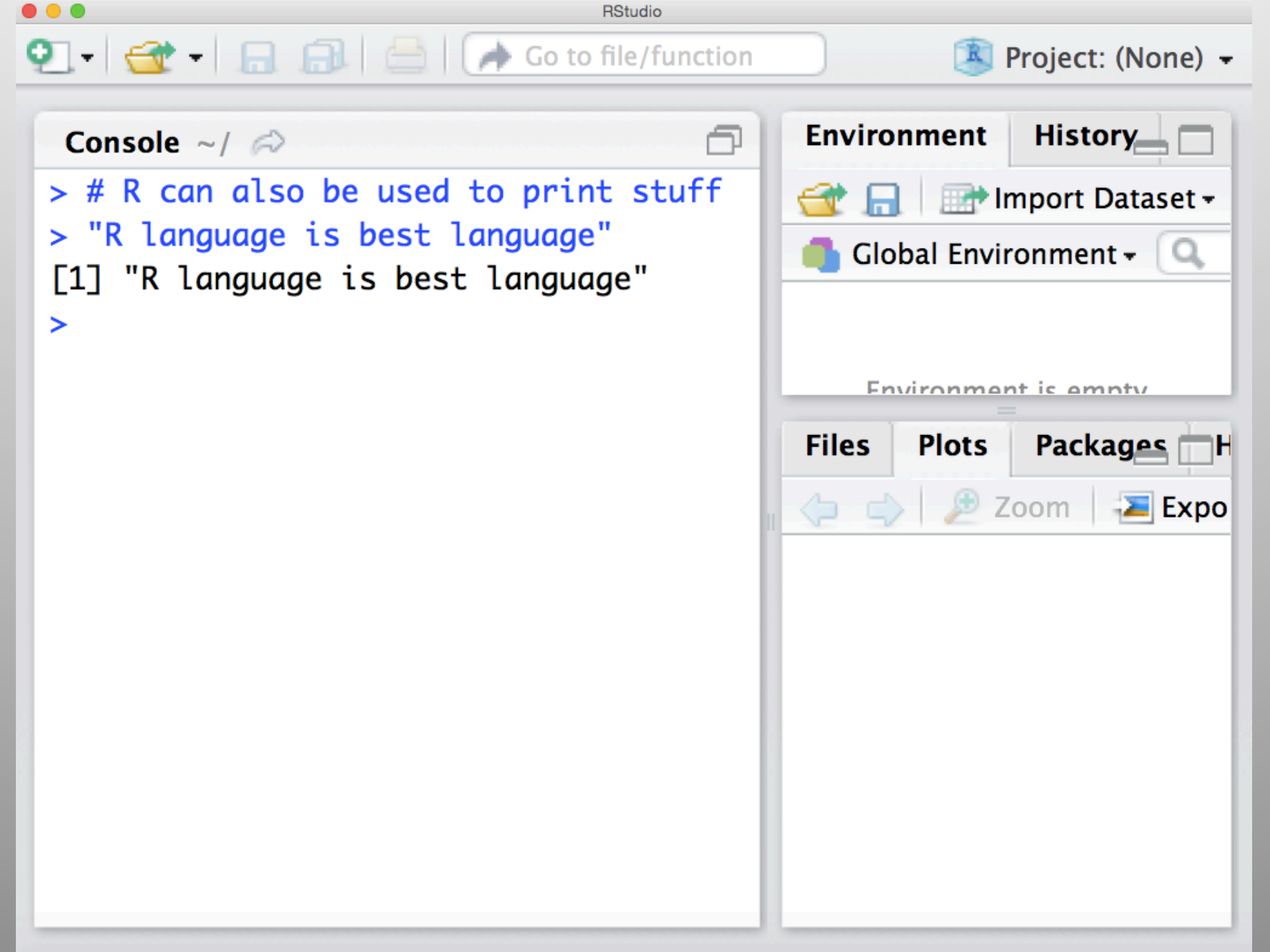
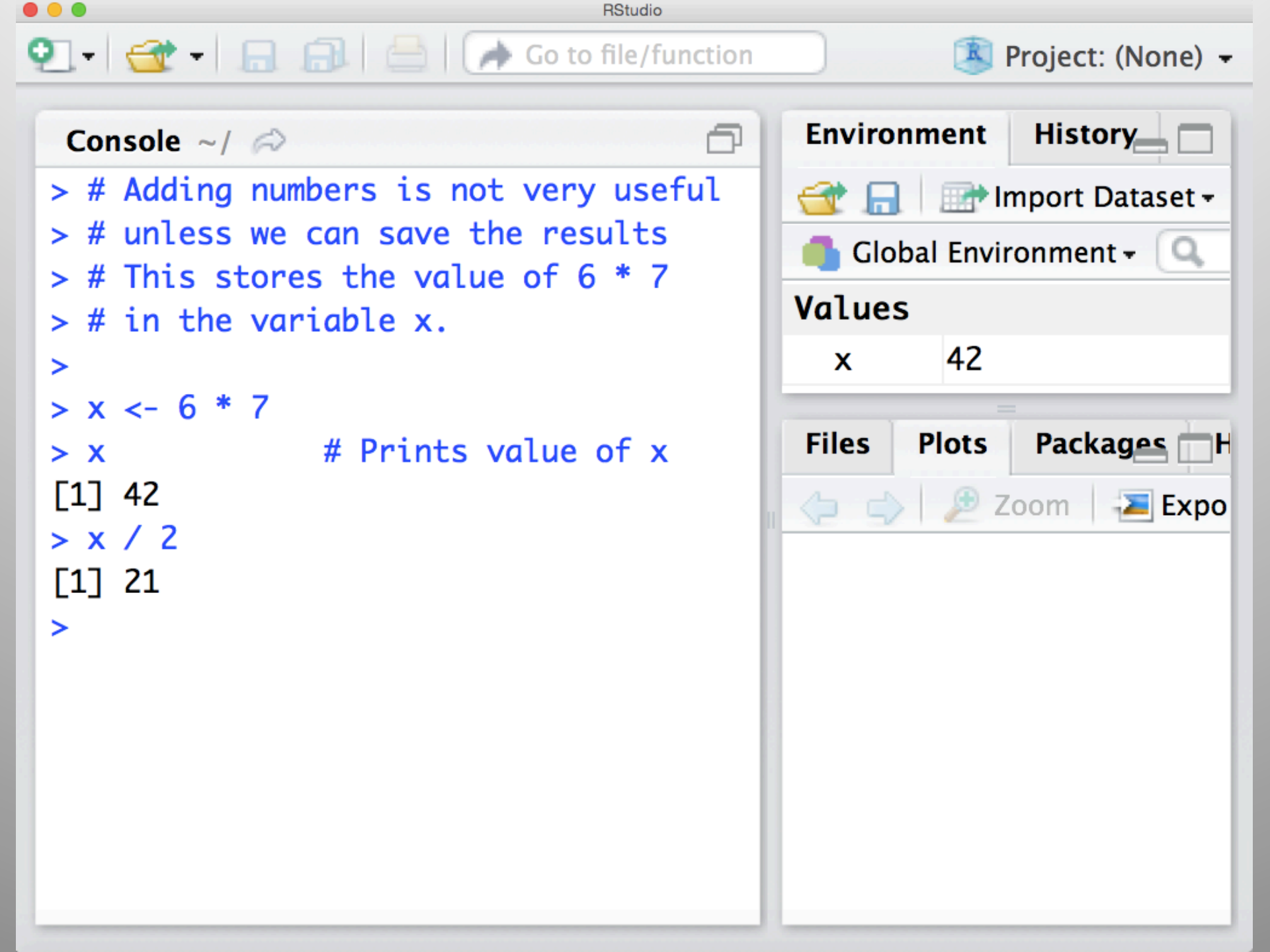
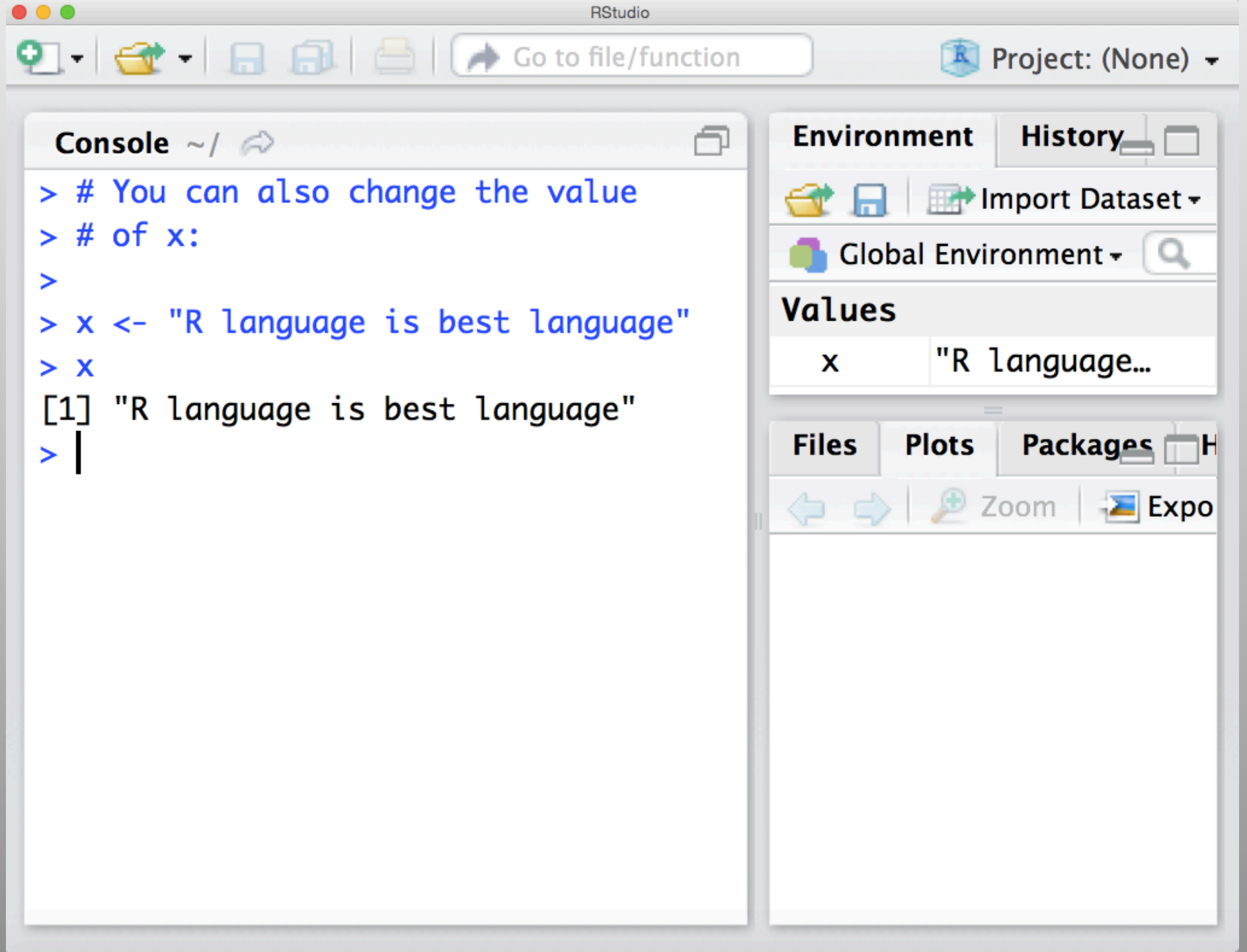
A taste of R

presented by the
Data Science Drop-In

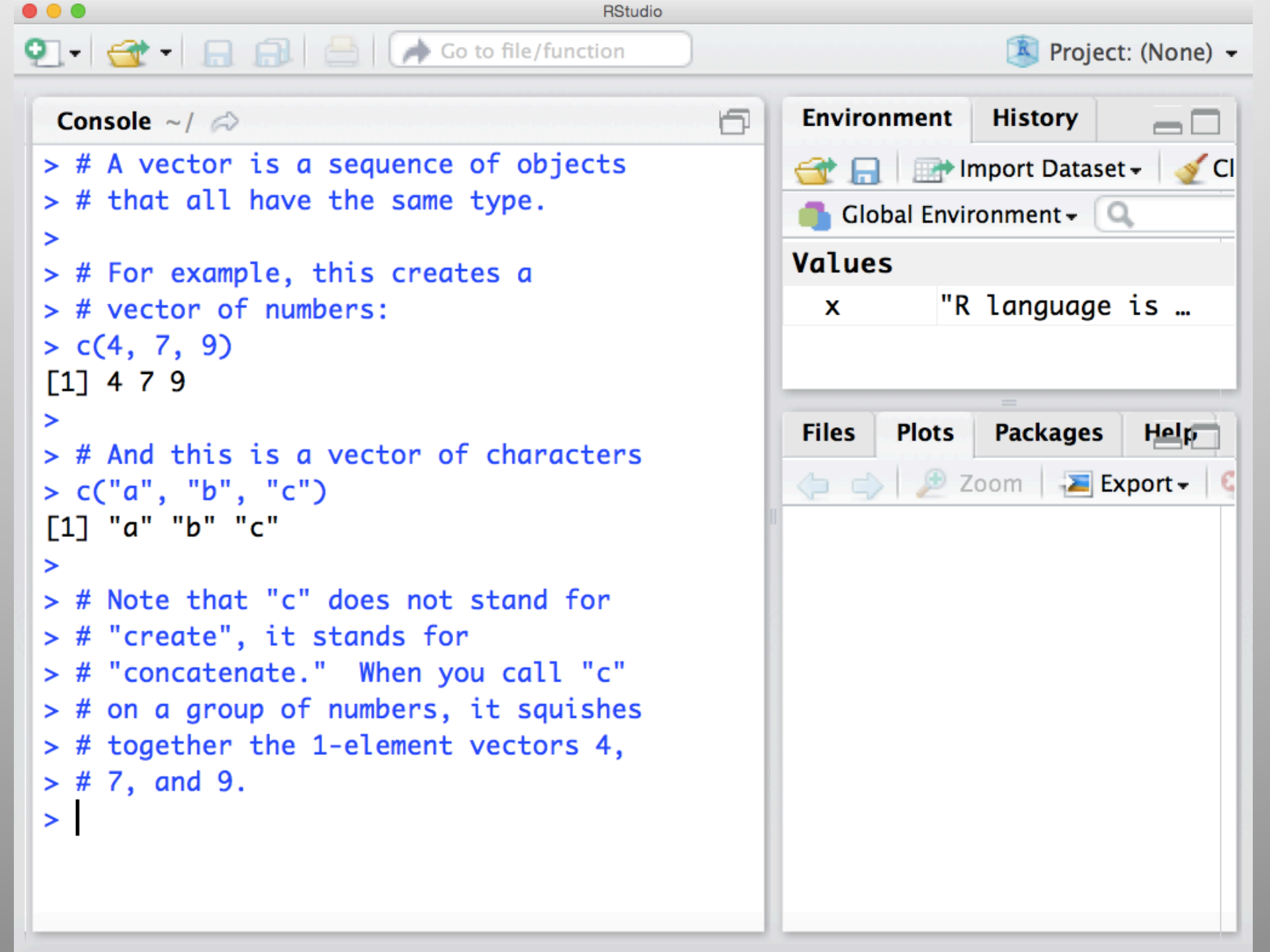

RStudio
Go to file/function
Project: (None)
Console ~/
> # R is great for arithmetic
> 1 + 1
[1] 2
> 6 * 7
[1] 42
>
>
Environment
History
Import Dataset
Global Environment
Environment is empty
Files
Plots
Packages
Help
Zoom
Export

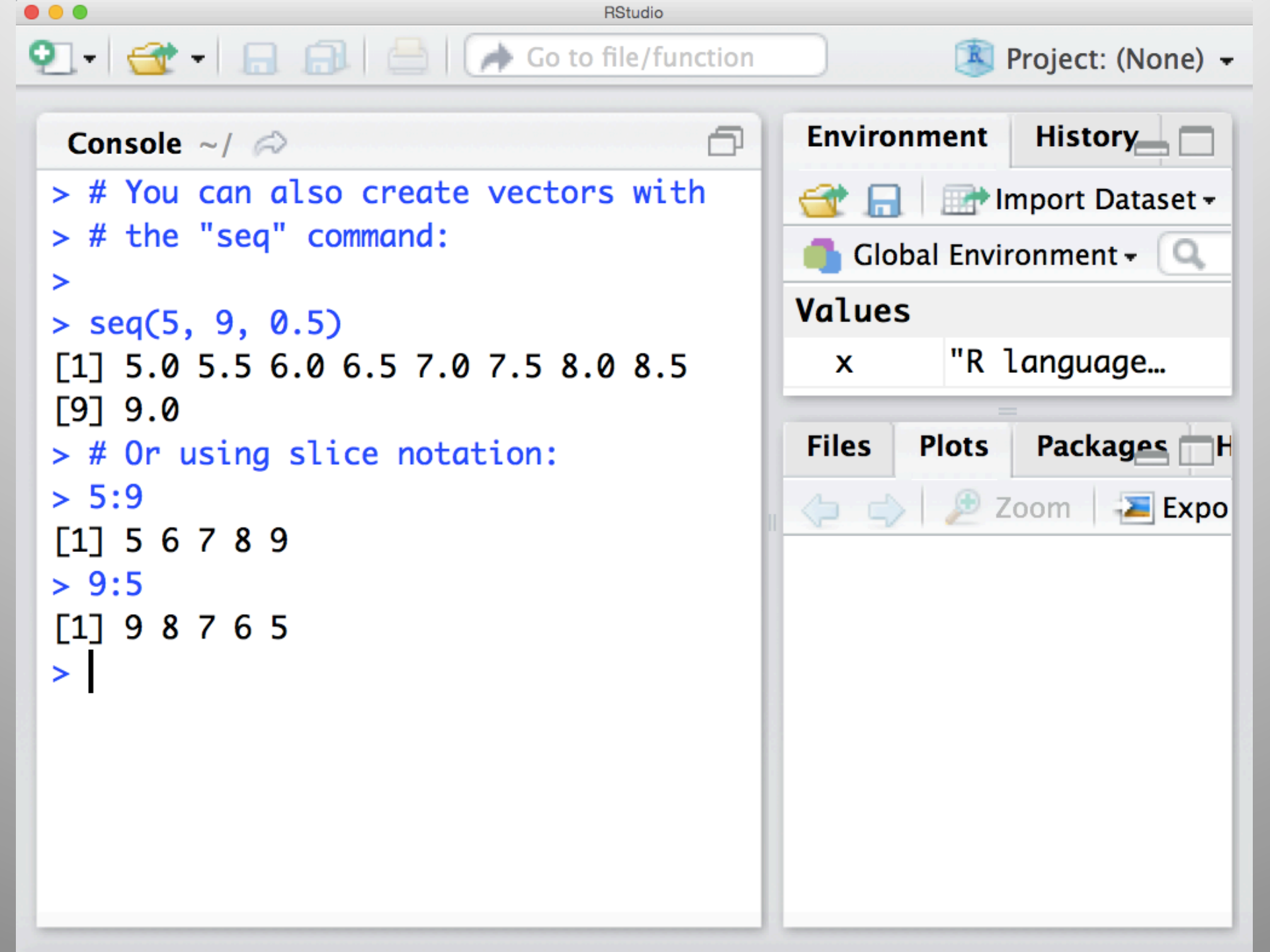

RStudio
Go to file/function
Project: (None)
Console ~/
> # R can also be used to print stuff
> "R language is best language"
[1] "R language is best language"
>
Environment
History
Import Dataset
Global Environment
Environment is empty
Files
Plots
Packages
Zoom
Expo

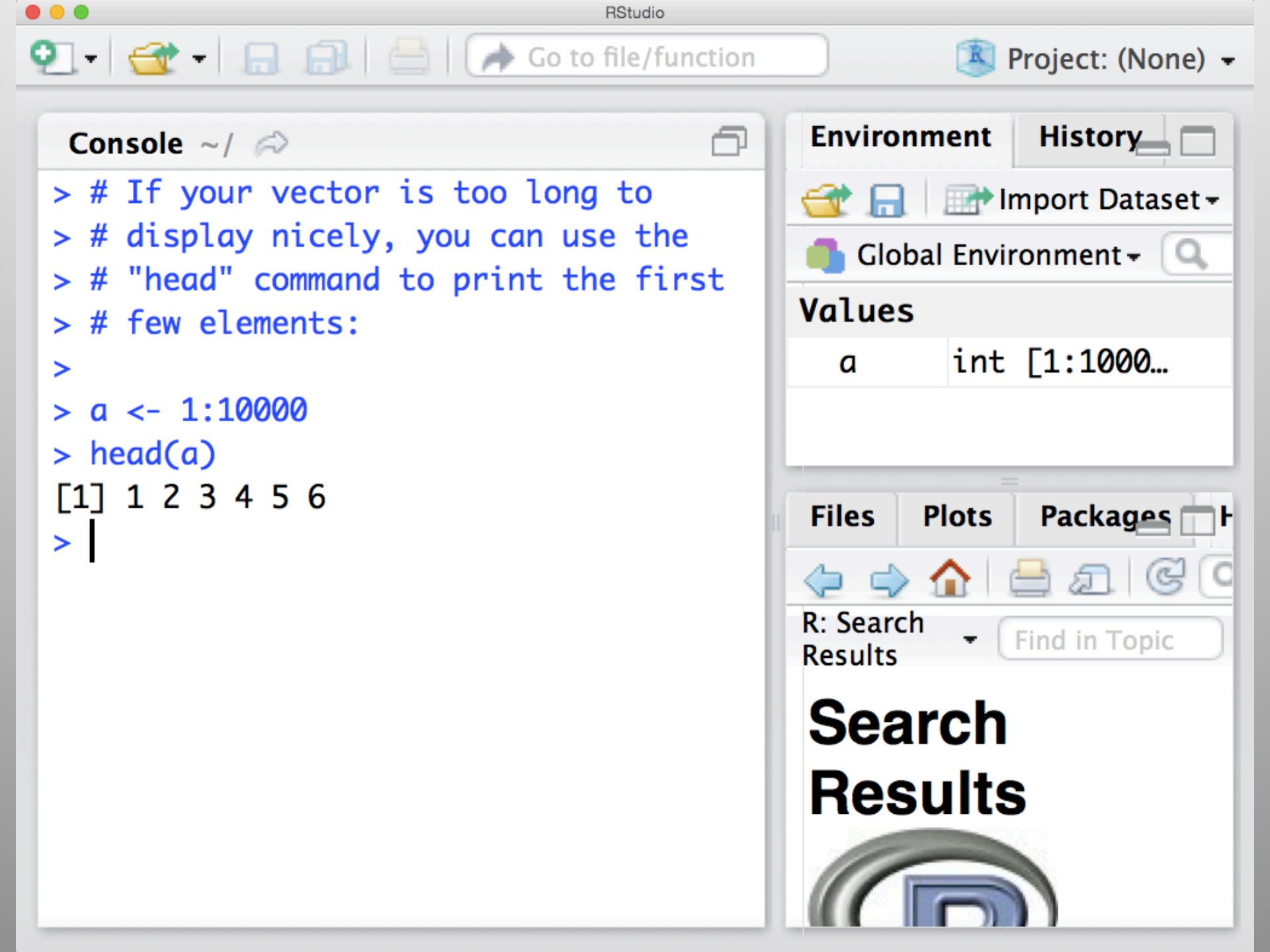

RStudio
Go to file/function
Project: (None)
Console ~/
> # Adding numbers is not very useful
> # unless we can save the results
> # This stores the value of 6 * 7
> # in the variable x.
>
> x <- 6 * 7
> x # Prints value of x
[1] 42
> x / 2
[1] 21
>
Environment
History
Import Dataset
Global Environment
Values
x 42
Files
Plots
Packages
Zoom
Expo

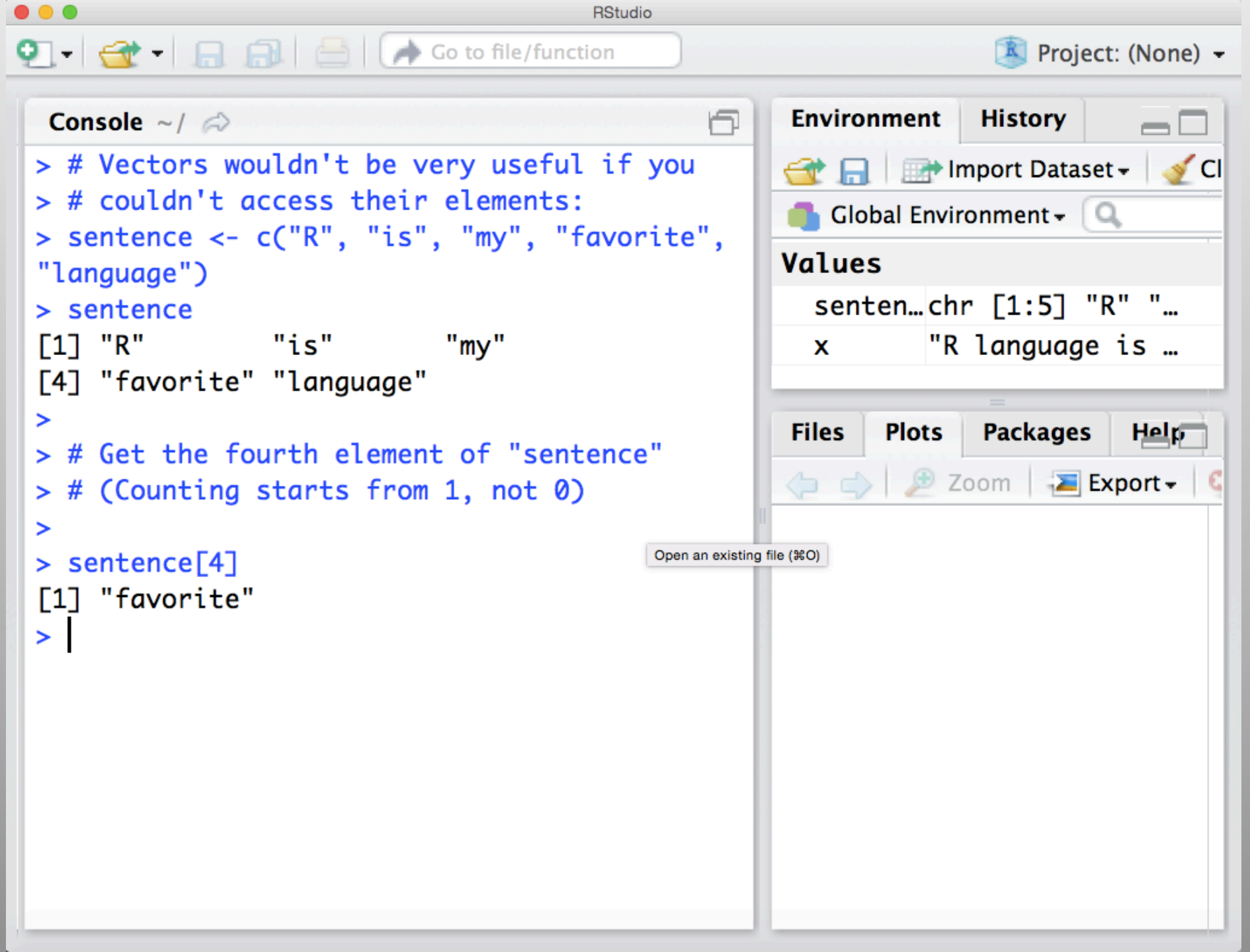

RStudio
Go to file/function
Project: (None)
Console ~/
> # You can also change the value
> # of x:
>
> x <- "R language is best language"
> x
[1] "R language is best language"
>
Environment
History
Import Dataset
Global Environment
Values
x
"R language...
Files
Plots
Packages
Zoom
Expo

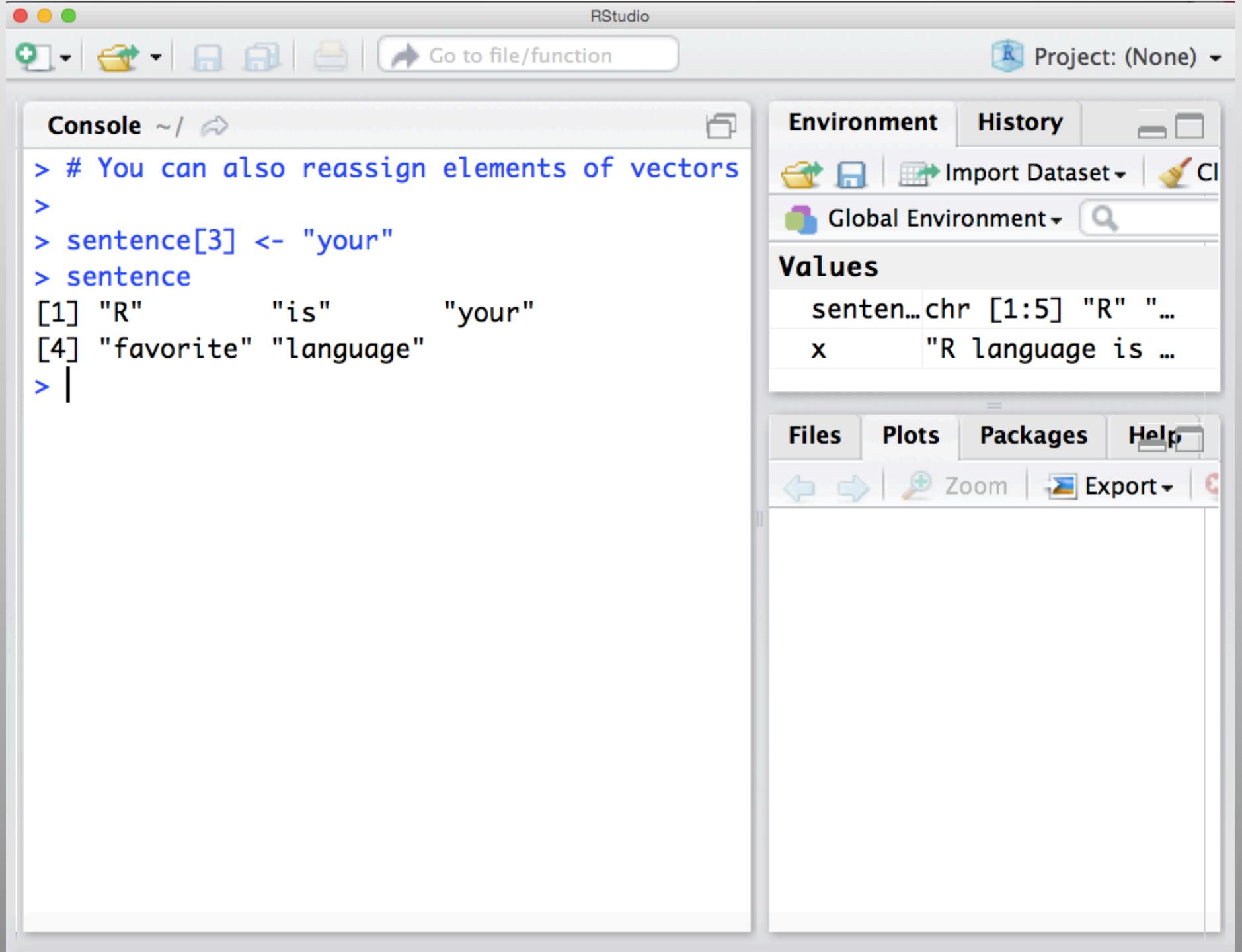
Vectors

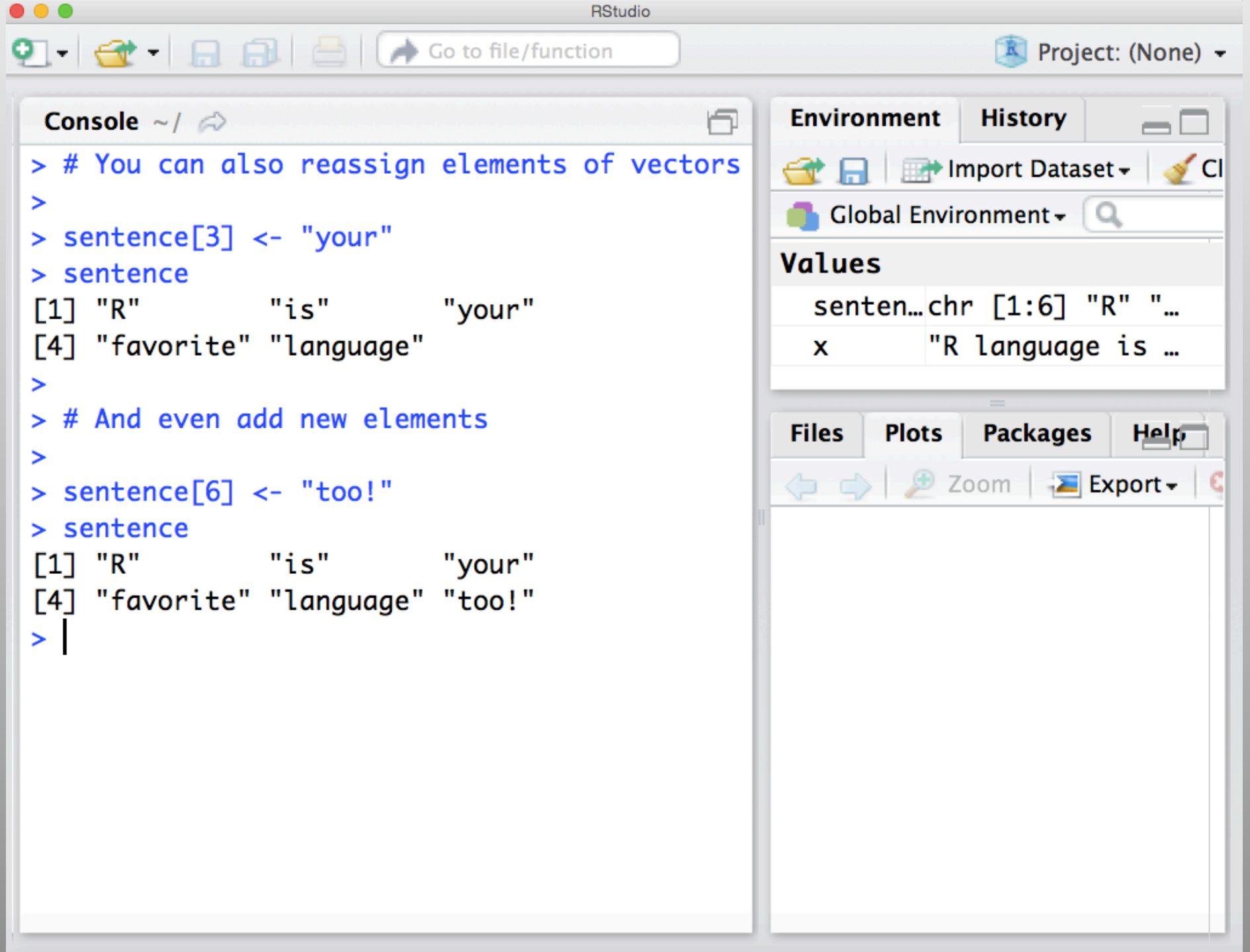

RStudio
Go to file/function
Project: (None)
Console ~/
> # A vector is a sequence of objects
> # that all have the same type.
>
> # For example, this creates a
> # vector of numbers:
> c(4, 7, 9)
[1] 4 7 9
>
> # And this is a vector of characters
> c("a", "b", "c")
[1] "a" "b" "c"
>
> # Note that "c" does not stand for
> # "create", it stands for
> # "concatenate." When you call "c"
> # on a group of numbers, it squishes
> # together the 1-element vectors 4,
> # 7, and 9.
>
Environment
History
Import Dataset
Global Environment
Values
x
"R language is …
Files
Plots
Packages
Help
Zoom
Export

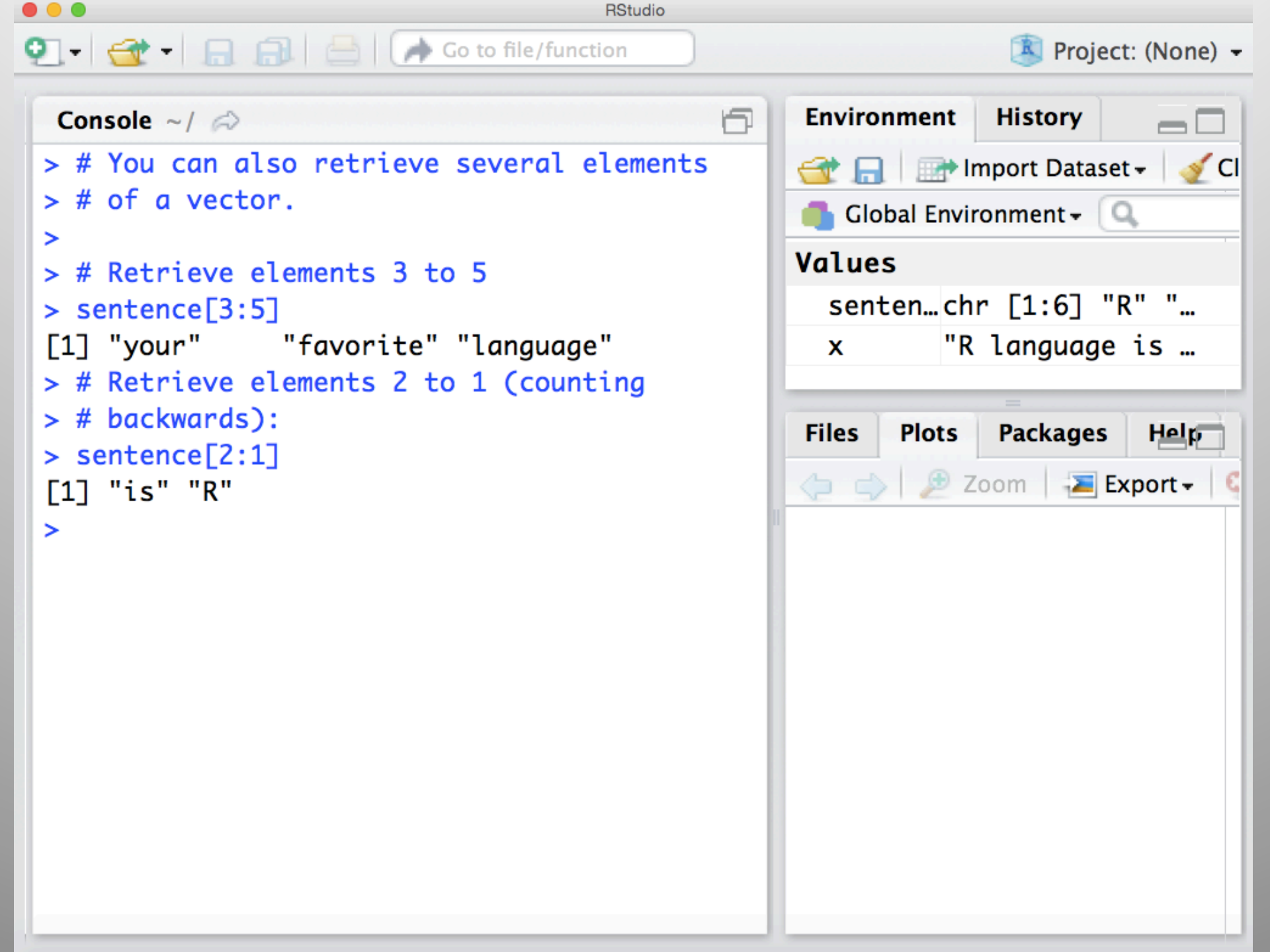

RStudio
Go to file/function
Project: (None)
Console ~/
> # You can also create vectors with
> # the "seq" command:
>
> seq(5, 9, 0.5)
[1] 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5
[9] 9.0
> # Or using slice notation:
> 5:9
[1] 5 6 7 8 9
> 9:5
[1] 9 8 7 6 5
>
Environment
History
Import Dataset
Global Environment
Values
x
"R language...
Files
Plots
Packages
Zoom
Expo

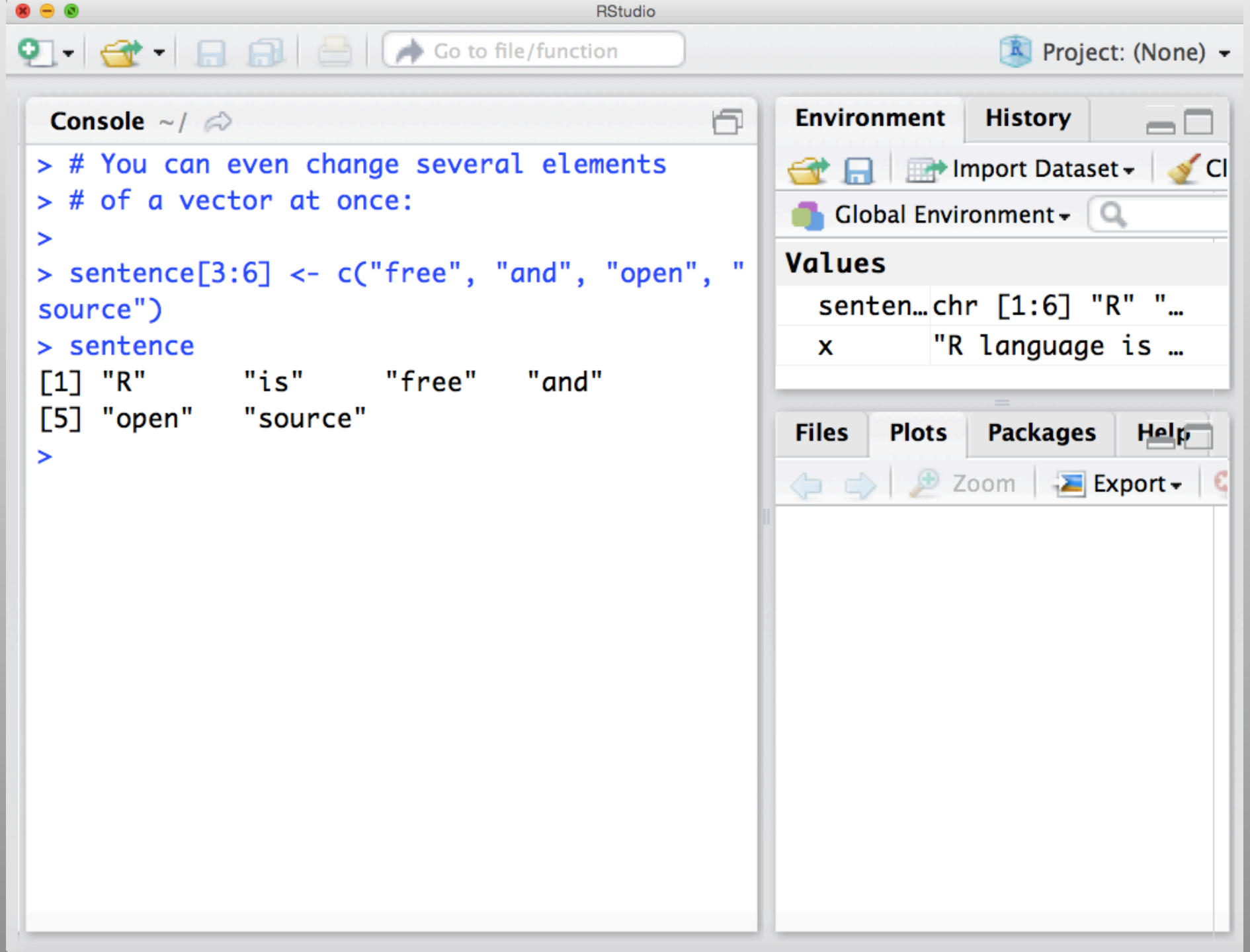

RStudio
Go to file/function
Project: (None)
Console ~/
> # If your vector is too long to
> # display nicely, you can use the
> # "head" command to print the first
> # few elements:
>
> a <- 1:10000
> head(a)
[1] 1 2 3 4 5 6
>
Environment
History
Import Dataset
Global Environment
Values
a
int [1:1000…
Files
Plots
Packages
R: Search Results
Find in Topic
Search Results


RStudio
Go to file/function
Project: (None)
Console ~/
> # Vectors wouldn't be very useful if you
> # couldn't access their elements:
> sentence <- c("R", "is", "my", "favorite",
"language")
> sentence
[1] "R"   "is"   "my"
[4] "favorite" "language"
>
> # Get the fourth element of "sentence"
> # (Counting starts from 1, not 0)
>
> sentence[4]
[1] "favorite"
>
Environment
History
Import Dataset
Global Environment
Values
senten… chr [1:5] "R" "…
x "R language is …
Files
Plots
Packages
Help
Zoom
Export
Open an existing file (⌘O)

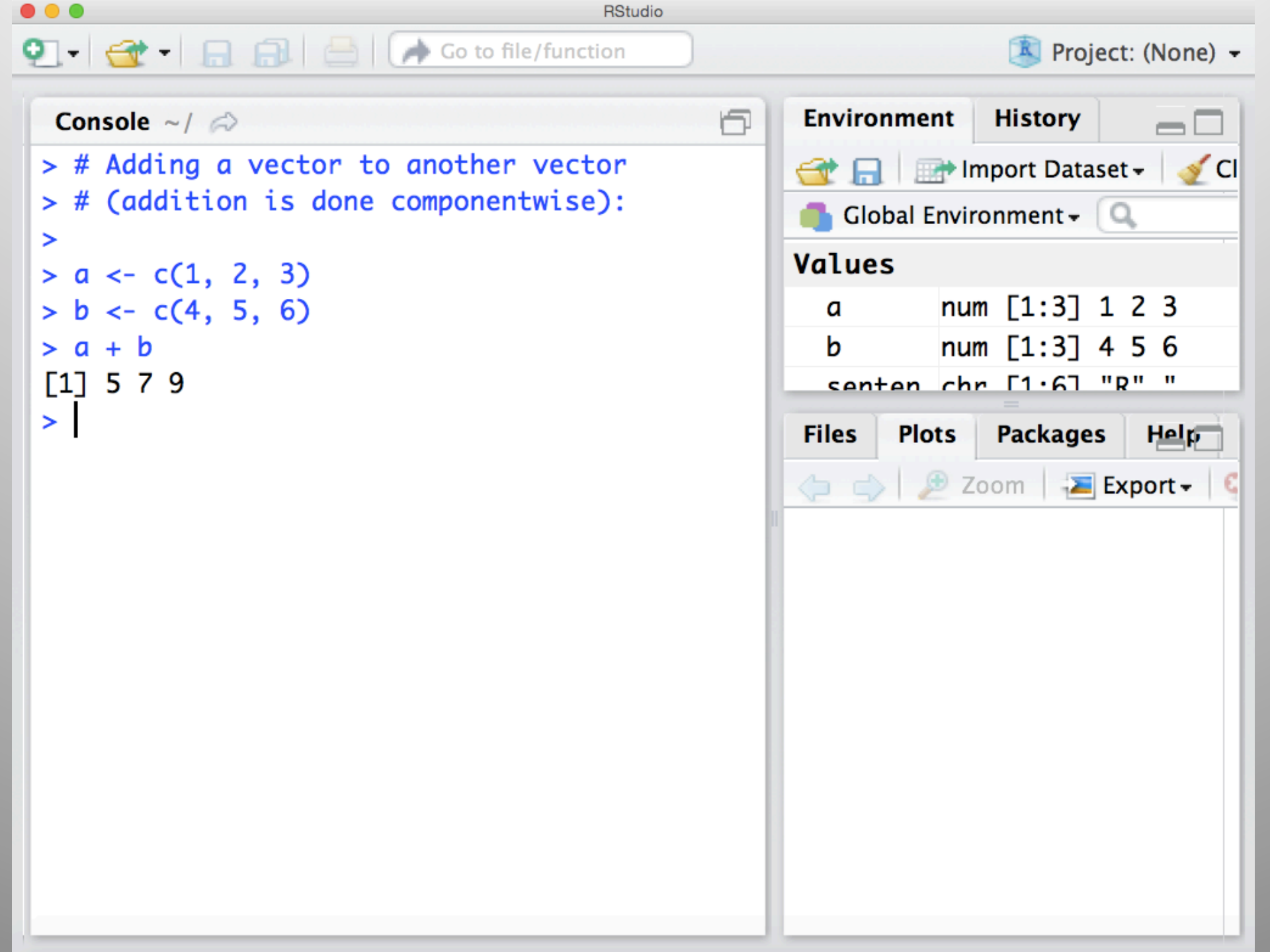

RStudio
Go to file/function
Project: (None)
Console ~/
> # You can also reassign elements of vectors
>
> sentence[3] <- "your"
> sentence
[1] "R"   "is"   "your"
[4] "favorite" "language"
>
Environment
History
Import Dataset
Global Environment
Values
senten… chr [1:5] "R" "…
x "R language is …
Files
Plots
Packages
Help
Zoom
Export

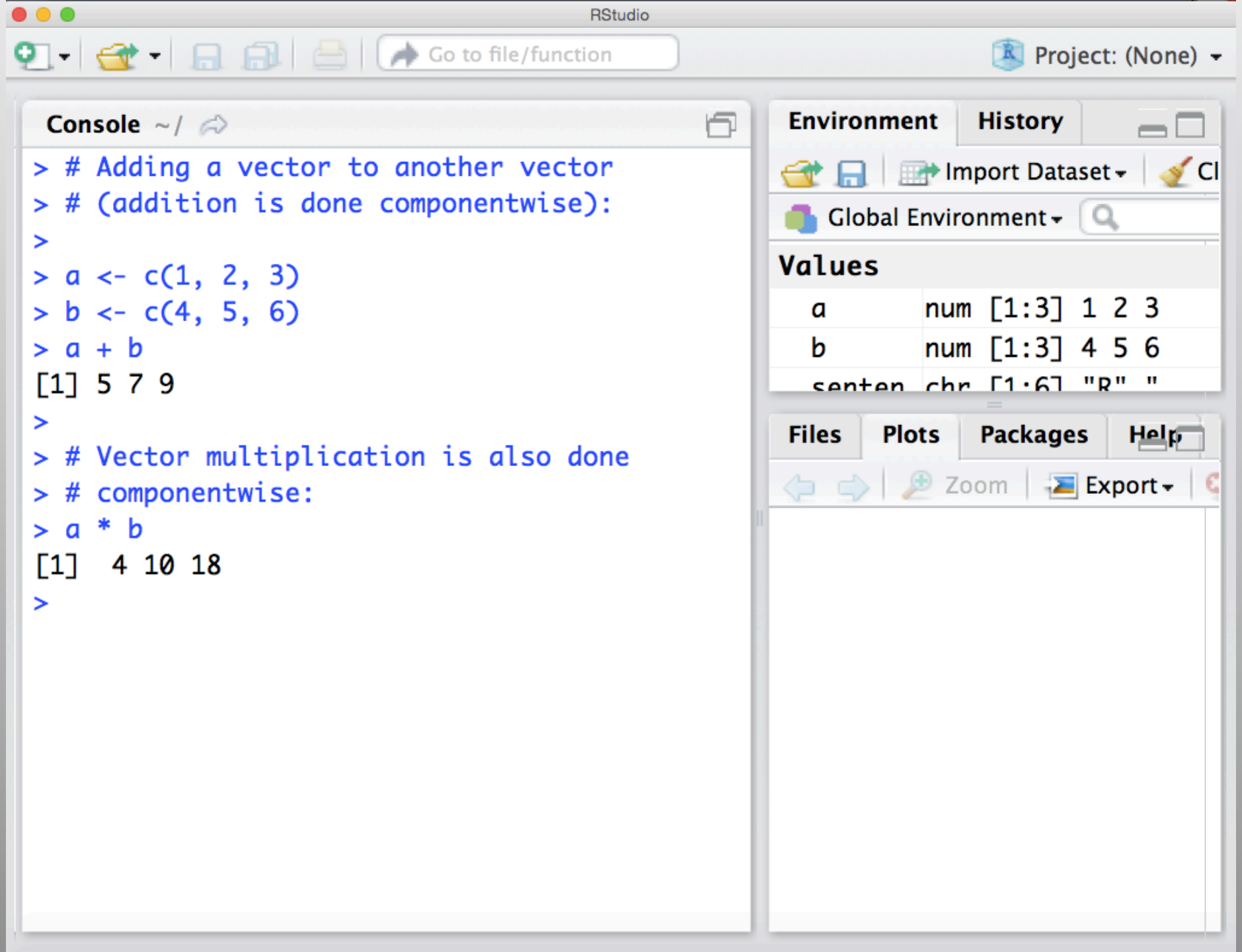

RStudio
Go to file/function
Project: (None)
Console ~/
> # You can also reassign elements of vectors
>
> sentence[3] <- "your"
> sentence
[1] "R"   "is"   "your"
[4] "favorite" "language"
>
> # And even add new elements
>
> sentence[6] <- "too!"
> sentence
[1] "R"   "is"   "your"
[4] "favorite" "language" "too!"
>
Environment
History
Import Dataset
Global Environment
Values
senten… chr [1:6] "R" "…
x "R language is …
Files
Plots
Packages
Help
Zoom
Export

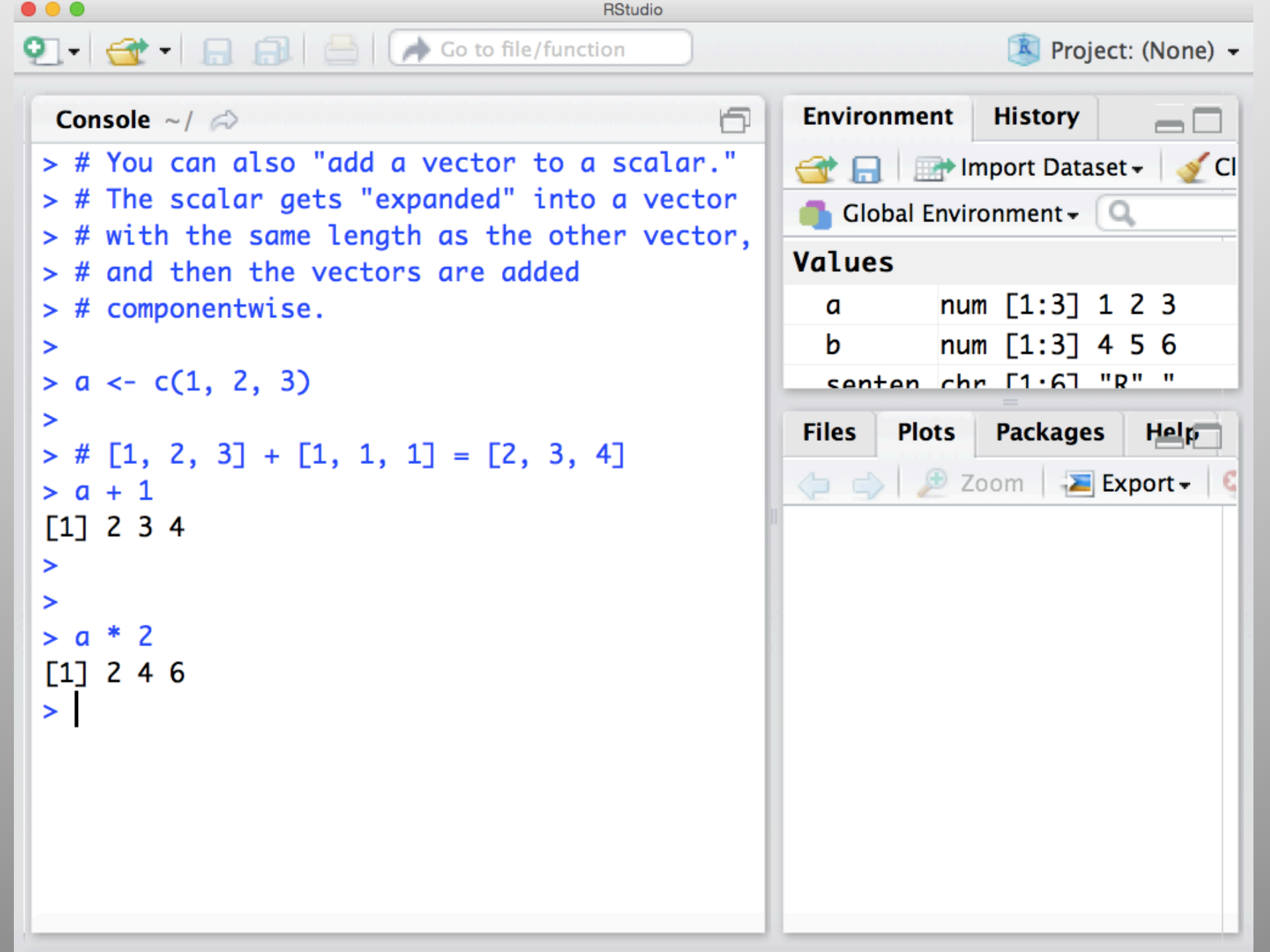

RStudio
Go to file/function
Project: (None)
Console ~/
> # You can also retrieve several elements
> # of a vector.
>
> # Retrieve elements 3 to 5
> sentence[3:5]
[1] "your" "favorite" "language"
> # Retrieve elements 2 to 1 (counting
> # backwards):
> sentence[2:1]
[1] "is" "R"
>
Environment
History
Import Dataset
Global Environment
Values
senten… chr [1:6] "R" "…
x "R language is …
Files
Plots
Packages
Help
Zoom
Export

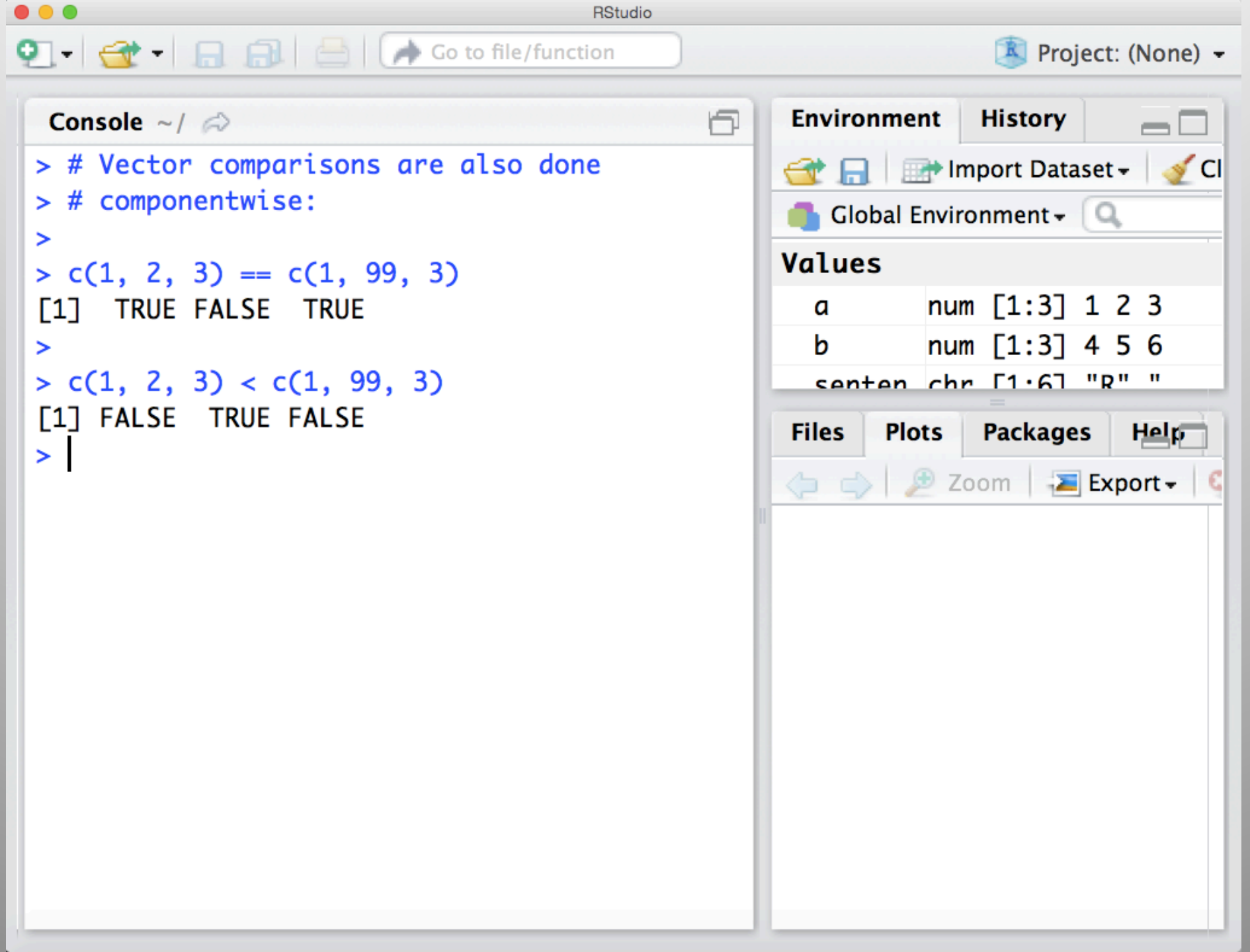

RStudio
Go to file/function
Project: (None)
Console ~/
> # You can even change several elements
> # of a vector at once:
>
> sentence[3:6] <- c("free", "and", "open", "
source")
> sentence
[1] "R"   "is"   "free"   "and"
[5] "open"   "source"
>
Environment
History
Import Dataset
Global Environment
Values
senten… chr [1:6] "R" "…
x "R language is …
Files
Plots
Packages
Help
Zoom
Export

Vector math

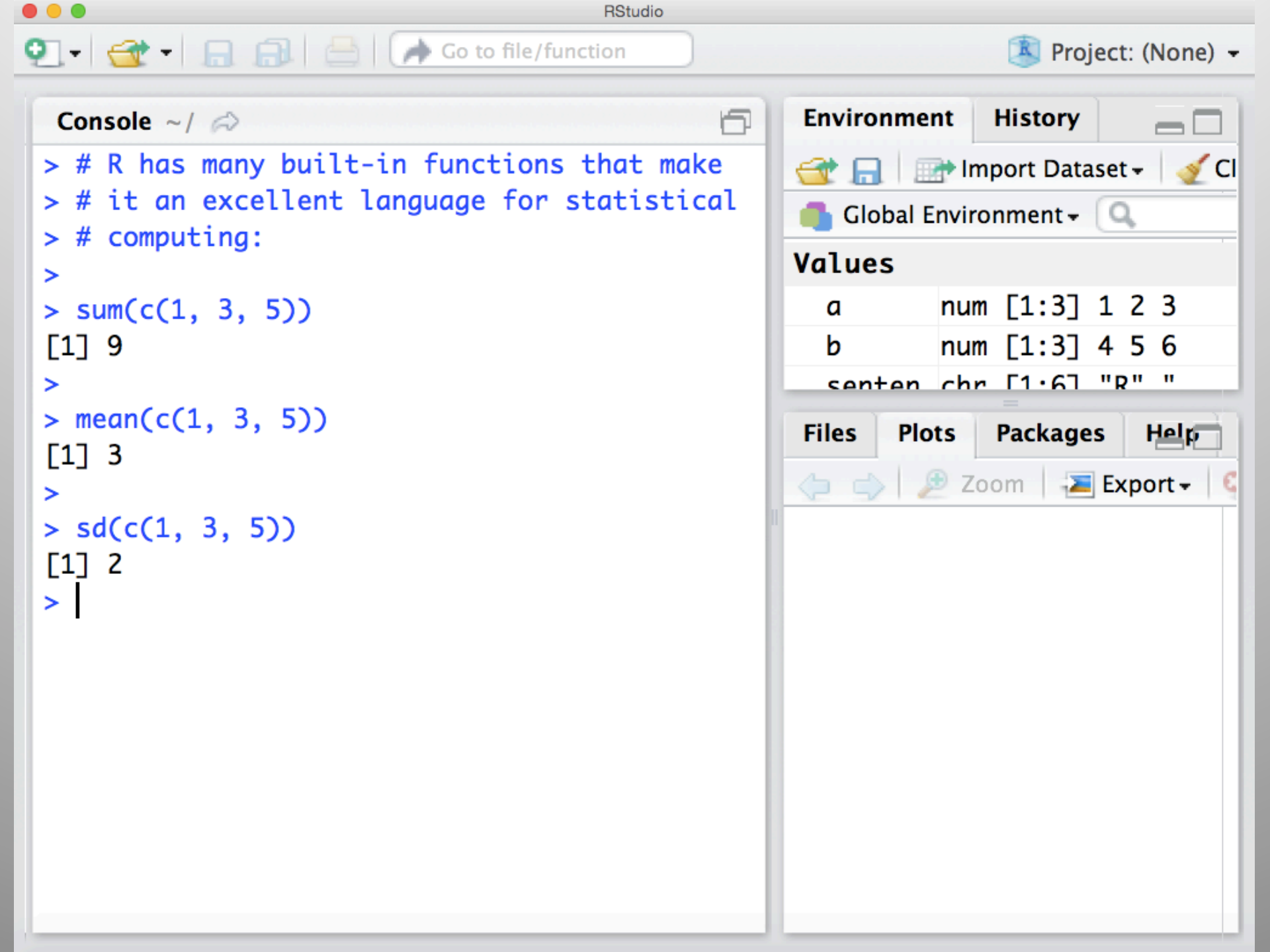

RStudio
Go to file/function
Project: (None)
Console ~/
> # Adding a vector to another vector
> # (addition is done componentwise):
>
> a <- c(1, 2, 3)
> b <- c(4, 5, 6)
> a + b
[1] 5 7 9
>
Environment
History
Import Dataset
Global Environment
Values
a num [1:3] 1 2 3
b num [1:3] 4 5 6
Files
Plots
Packages
Help
Zoom
Export

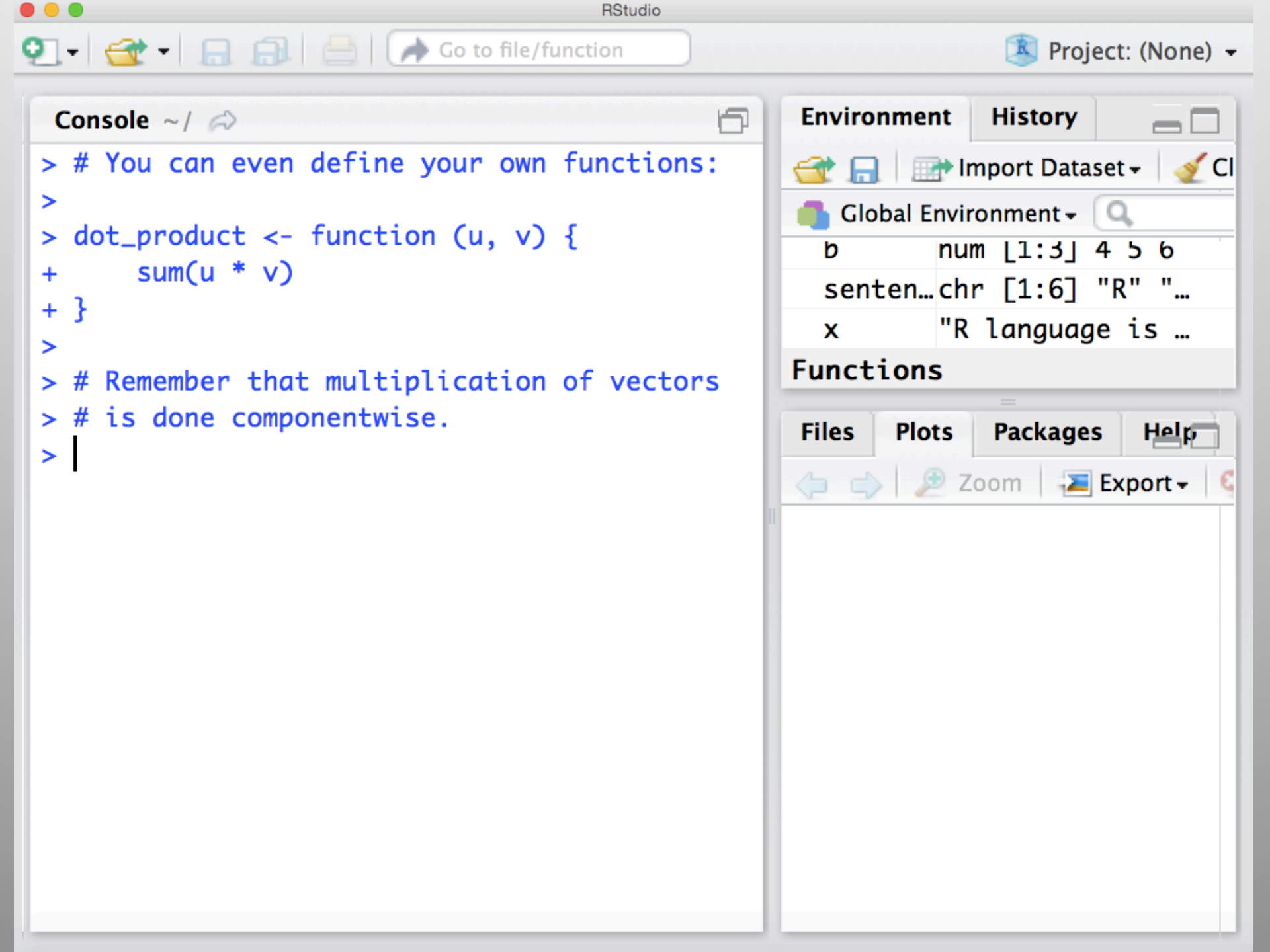

RStudio
Go to file/function
Project: (None)
Console ~/
> # Adding a vector to another vector
> # (addition is done componentwise):
>
> a <- c(1, 2, 3)
> b <- c(4, 5, 6)
> a + b
[1] 5 7 9
>
> # Vector multiplication is also done
> # componentwise:
> a * b
[1] 4 10 18
>
Environment
History
Import Dataset
Global Environment
Values
a num [1:3] 1 2 3
b num [1:3] 4 5 6
Files
Plots
Packages
Help
Zoom
Export

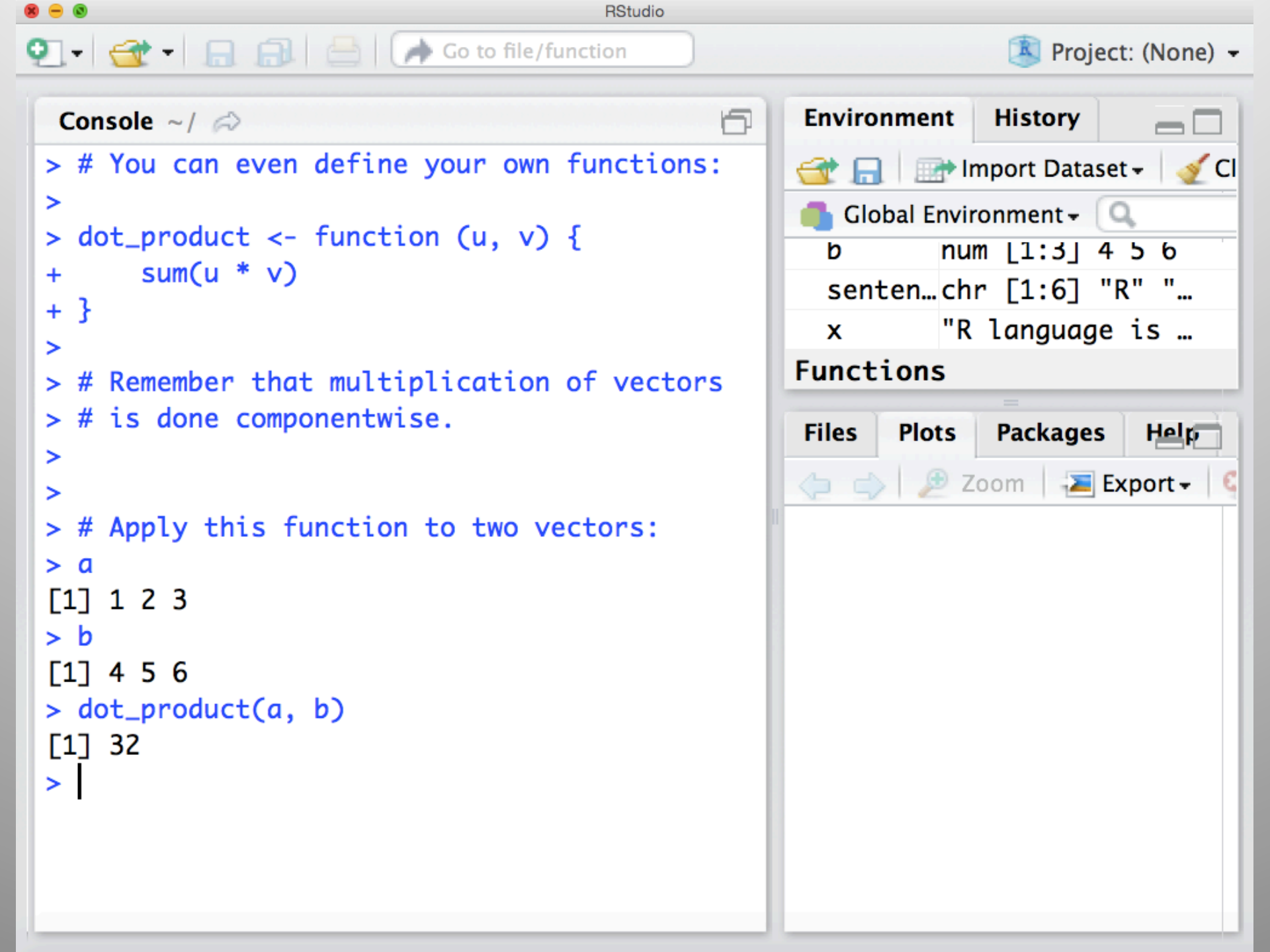

RStudio
Go to file/function
Project: (None)
Console ~/
> # You can also "add a vector to a scalar."
> # The scalar gets "expanded" into a vector
> # with the same length as the other vector,
> # and then the vectors are added
> # componentwise.
>
> a <- c(1, 2, 3)
>
> # [1, 2, 3] + [1, 1, 1] = [2, 3, 4]
> a + 1
[1] 2 3 4
>
>
> a * 2
[1] 2 4 6
>
Environment
History
Import Dataset
Global Environment
Values
a num [1:3] 1 2 3
b num [1:3] 4 5 6
Files
Plots
Packages
Help
Zoom
Export


RStudio
Go to file/function
Project: (None)
Console ~/
> # Vector comparisons are also done
> # componentwise:
>
> c(1, 2, 3) == c(1, 99, 3)
[1] TRUE FALSE TRUE
>
> c(1, 2, 3) < c(1, 99, 3)
[1] FALSE TRUE FALSE
>
Environment
History
Import Dataset
Global Environment
Values
a num [1:3] 1 2 3
b num [1:3] 4 5 6
Files
Plots
Packages
Help
Zoom
Export

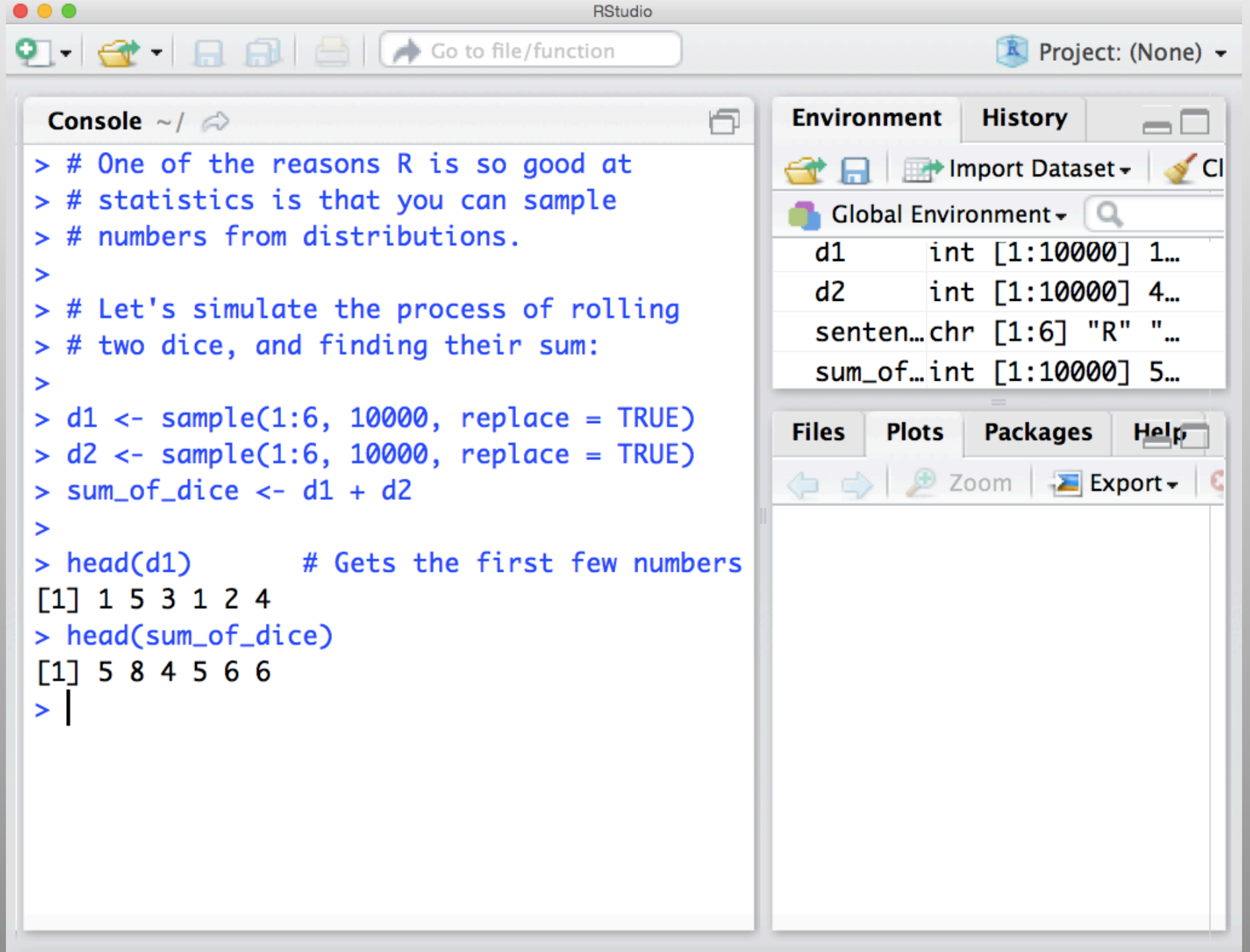
Functions

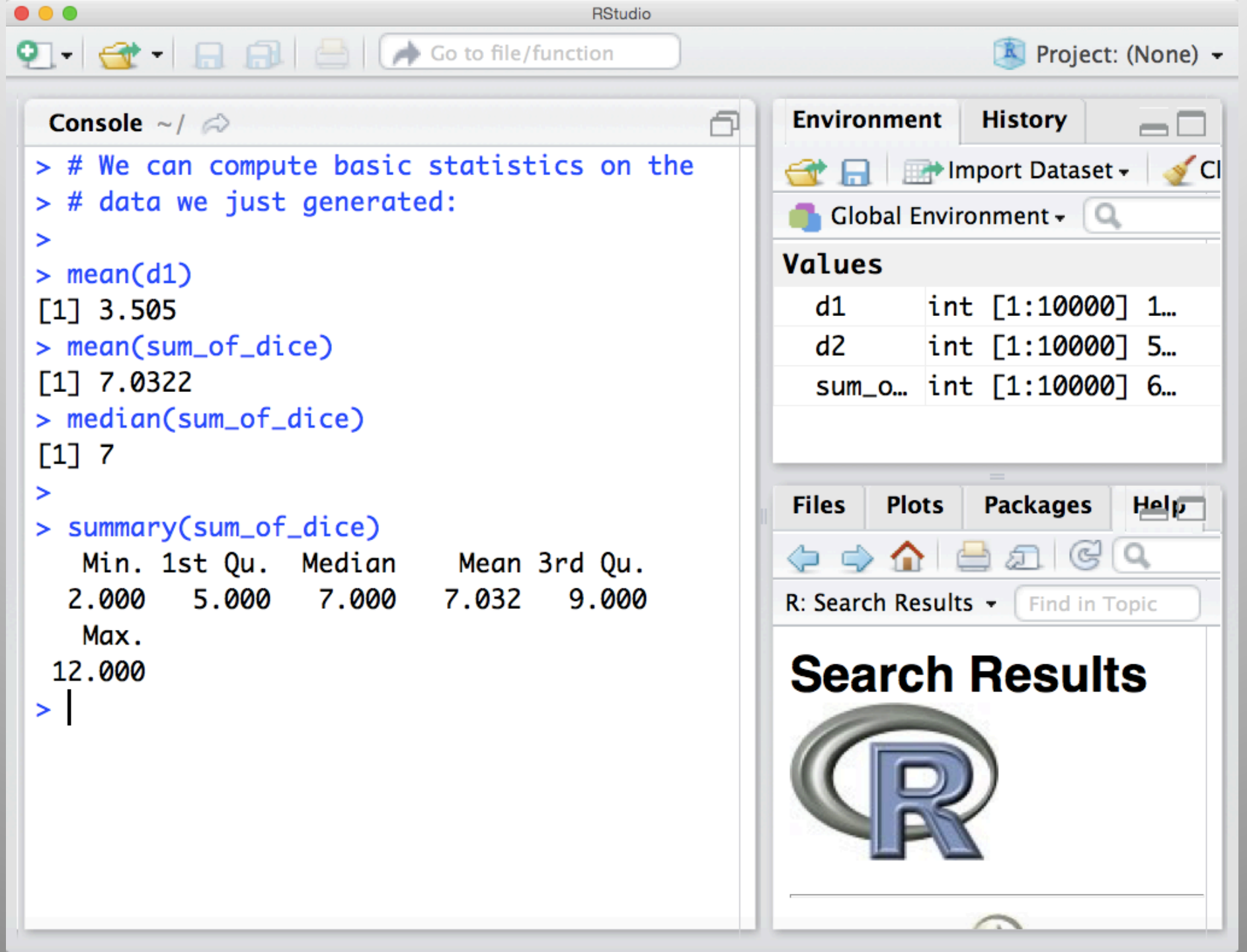

RStudio
Go to file/function
Project: (None)
Console ~/
> # R has many built-in functions that make
> # it an excellent language for statistical
> # computing:
>
> sum(c(1, 3, 5))
[1] 9
>
> mean(c(1, 3, 5))
[1] 3
>
> sd(c(1, 3, 5))
[1] 2
>
Environment
History
Import Dataset
Global Environment
Values
a num [1:3] 1 2 3
b num [1:3] 4 5 6
Files
Plots
Packages
Help
Zoom
Export


RStudio
Go to file/function
Project: (None)
Console ~/
> # You can even define your own functions:
>
> dot_product <- function (u, v) {
+ sum(u * v)
+ }
>
> # Remember that multiplication of vectors
> # is done componentwise.
>
Environment
History
Import Dataset
Global Environment
b num [1:3] 4 5 6
senten… chr [1:6] "R" "…
x "R language is …
Functions
Files
Plots
Packages
Help
Zoom
Export


RStudio
Go to file/function
Project: (None)
Console ~/
> # You can even define your own functions:
>
> dot_product <- function (u, v) {
+ sum(u * v)
+ }
>
> # Remember that multiplication of vectors
> # is done componentwise.
>
>
> # Apply this function to two vectors:
> a
[1] 1 2 3
> b
[1] 4 5 6
> dot_product(a, b)
[1] 32
>
Environment
History
Import Dataset
Global Environment
b num [1:3] 4 5 6
senten… chr [1:6] "R" "…
x "R language is …
Functions
Files
Plots
Packages
Help
Zoom
Export

Statistics


RStudio
Go to file/function
Project: (None)
Console ~/
> # One of the reasons R is so good at
> # statistics is that you can sample
> # numbers from distributions.
>
> # Let's simulate the process of rolling
> # two dice, and finding their sum:
>
> d1 <- sample(1:6, 10000, replace = TRUE)
> d2 <- sample(1:6, 10000, replace = TRUE)
> sum_of_dice <- d1 + d2
>
> head(d1) # Gets the first few numbers
[1] 1 5 3 1 2 4
> head(sum_of_dice)
[1] 5 8 4 5 6 6
>
Environment
History
Import Dataset
Global Environment
d1 int [1:10000] 1…
d2 int [1:10000] 4…
senten… chr [1:6] "R" "…
sum_of… int [1:10000] 5…
Files
Plots
Packages
Help
Zoom
Export


RStudio
Go to file/function
Project: (None)
Console ~/
> # We can compute basic statistics on the
> # data we just generated:
>
> mean(d1)
[1] 3.505
> mean(sum_of_dice)
[1] 7.0322
> median(sum_of_dice)
[1] 7
>
> summary(sum_of_dice)
Min. 1st Qu. Median Mean 3rd Qu.
2.000 5.000 7.000 7.032 9.000
Max.
12.000
>
Environment
History
Import Dataset
Global Environment
Values
d1 int [1:10000] 1...
d2 int [1:10000] 5...
sum_o... int [1:10000] 6...
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results

RStudio

Go to file/function

Project: (None)

Console ~/

```
> # Let's plot a histogram of the
> # data:
>
> hist(sum_of_dice)
>
>
> # This histogram is okay-
> # looking, but we'll talk about
> # nicer graphics packages later
> # in the talk.
>
```

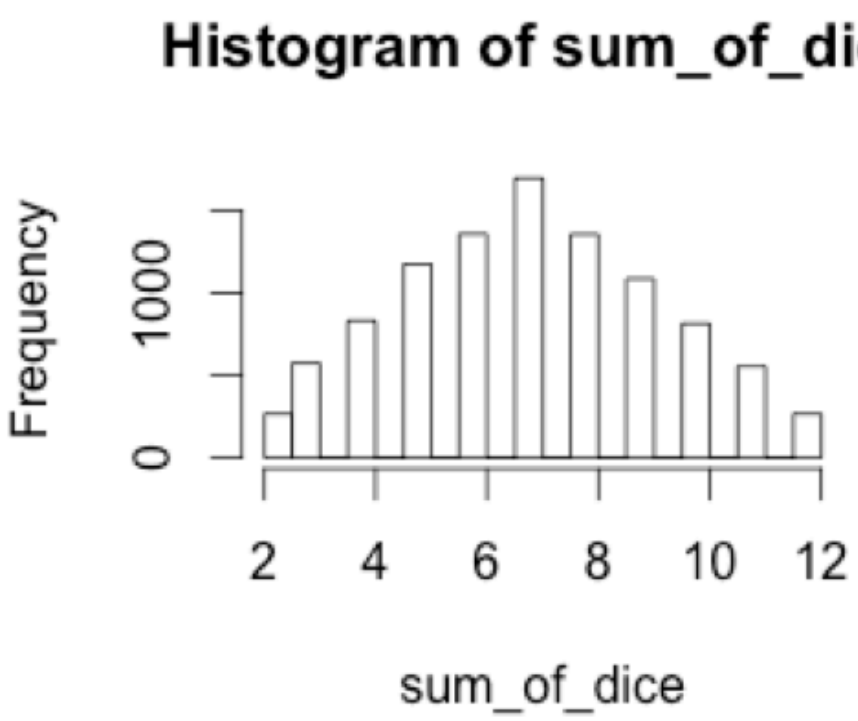
Environment | History

Import Dataset | Clear | List

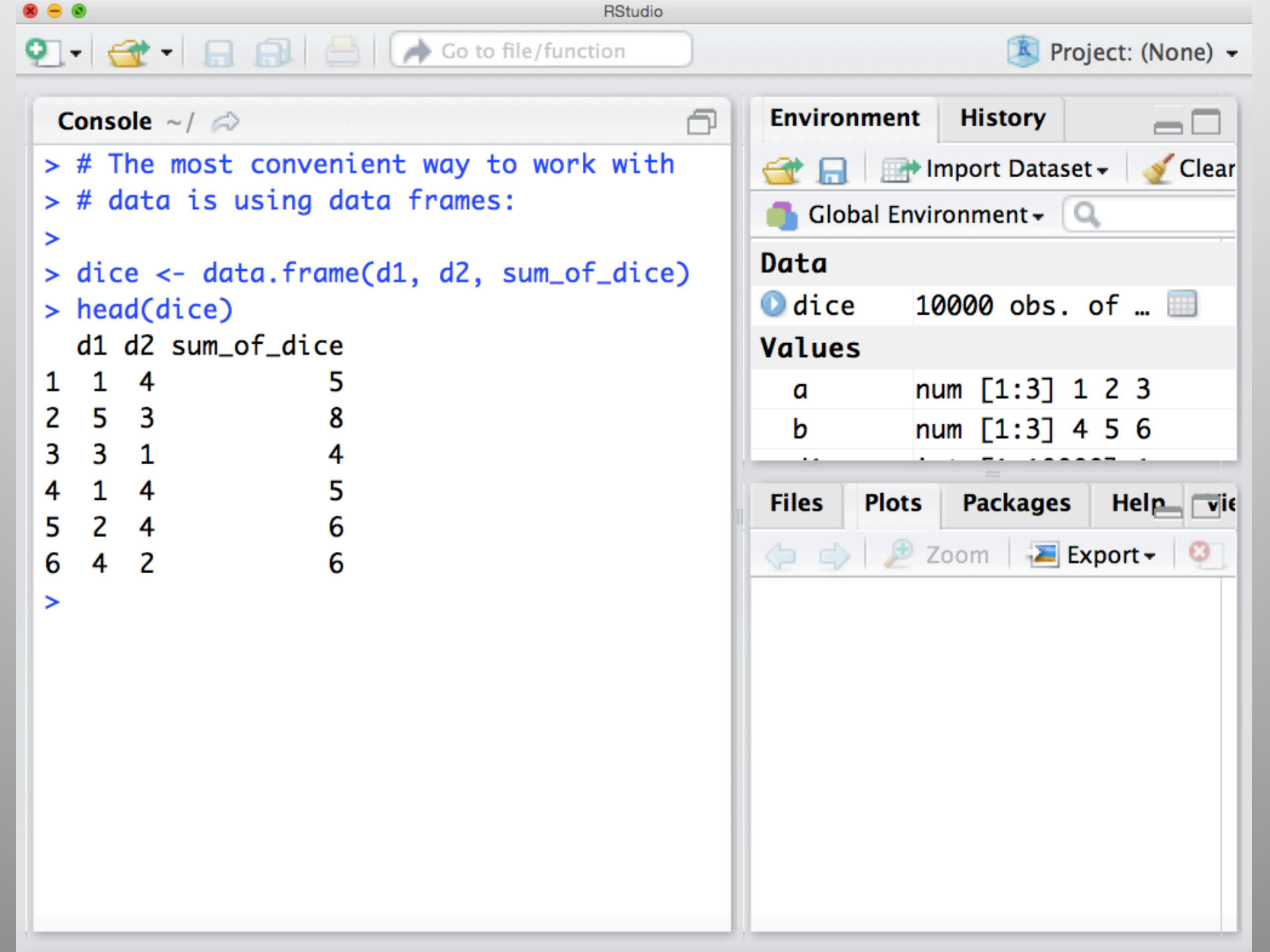
Global Environment

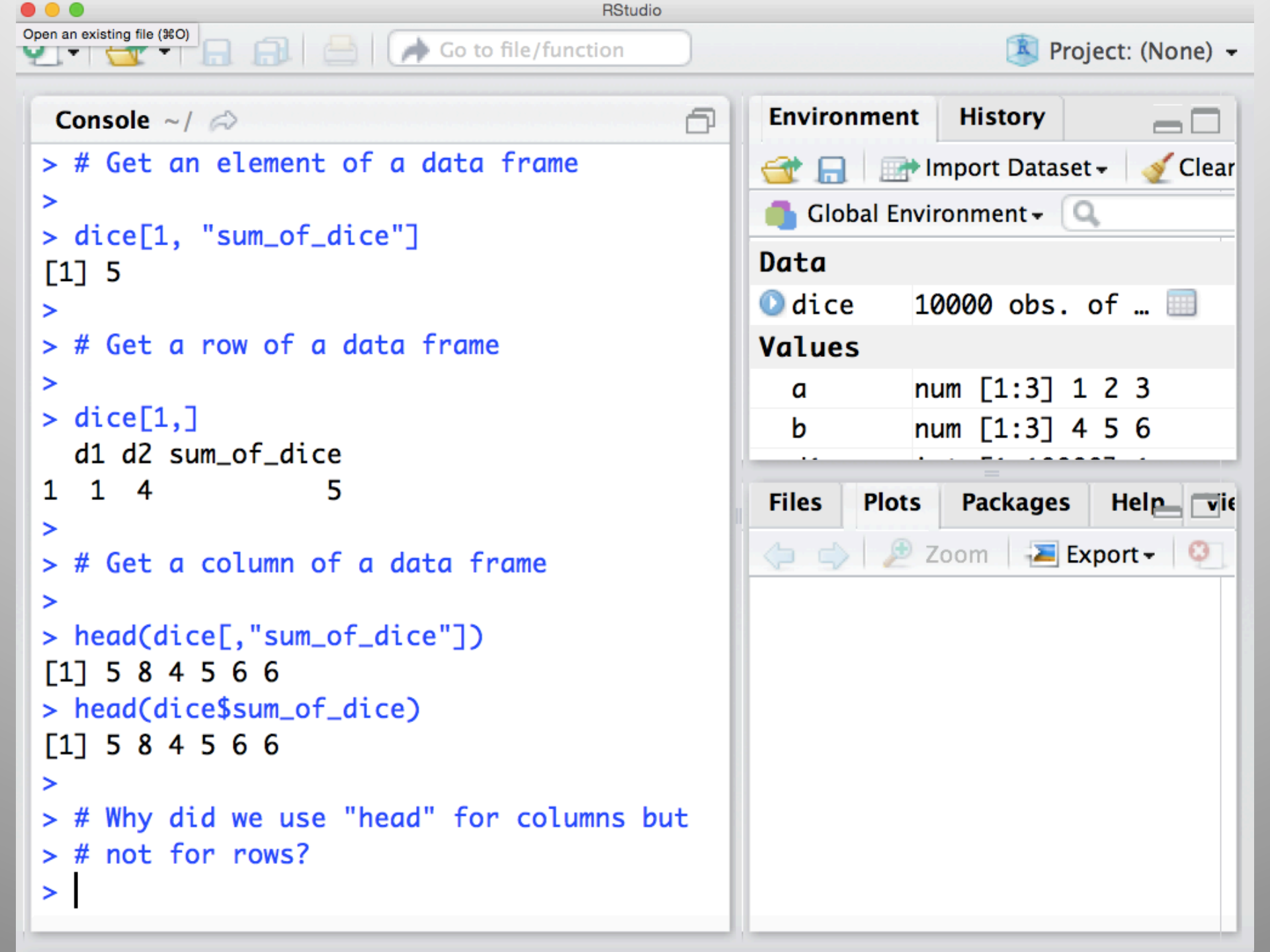
Files | Plots | Packages | Help | Viewer

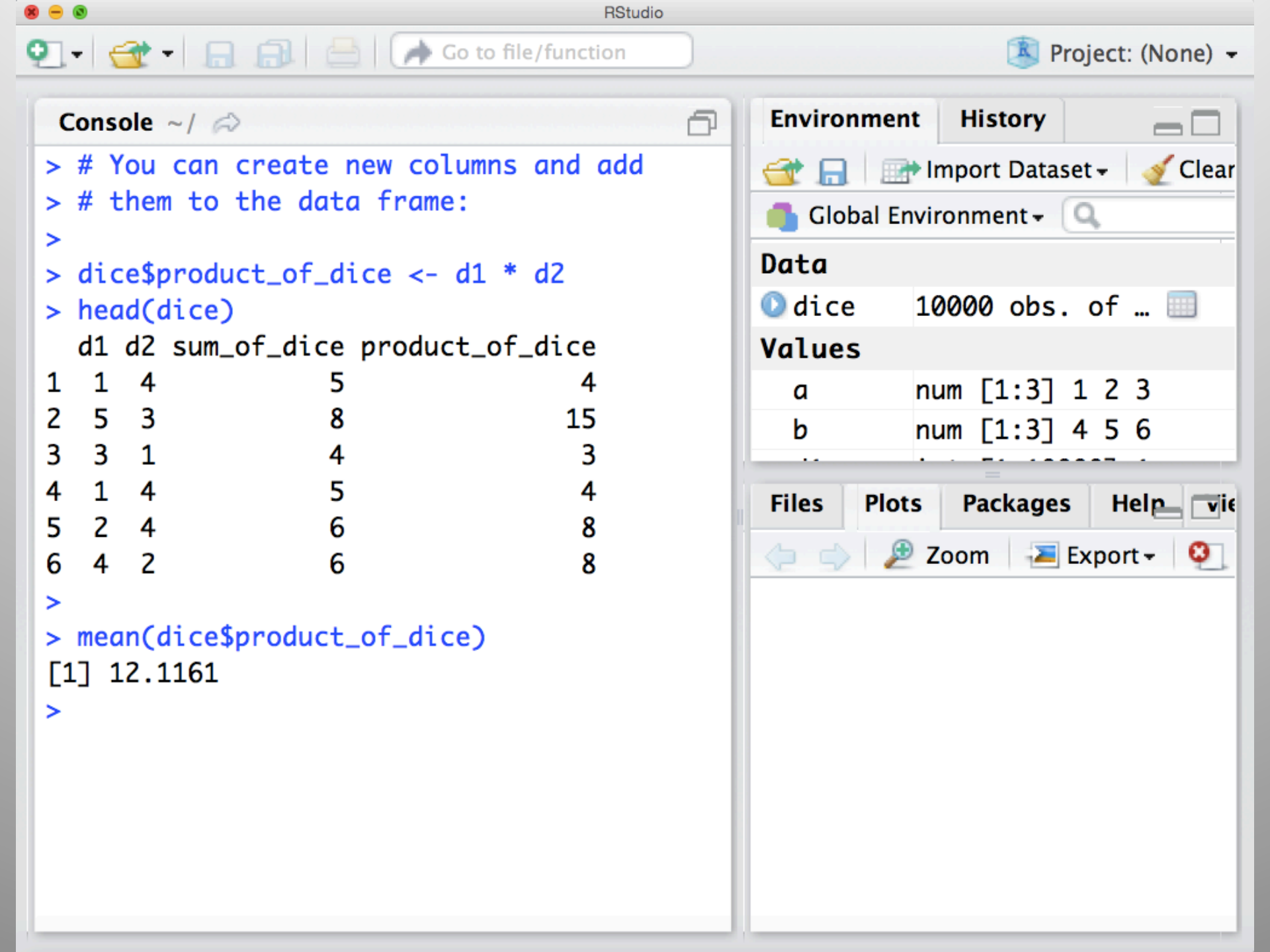
Zoom | Export | Clear All



Data frames


RStudio
Go to file/function
Project: (None)
Console ~/
> # The most convenient way to work with
> # data is using data frames:
>
> dice <- data.frame(d1, d2, sum_of_dice)
> head(dice)
  d1 d2 sum_of_dice
1 1 4   5
2 5 3   8
3 3 1   4
4 1 4   5
5 2 4   6
6 4 2   6
>
Environment
History
Import Dataset
Clear
Global Environment
Data
dice
10000 obs. of …
Values
a
num [1:3] 1 2 3
b
num [1:3] 4 5 6
Files
Plots
Packages
Help
Zoom
Export


RStudio
Open an existing file (⌘O)
Go to file/function
Project: (None)
Console ~/
> # Get an element of a data frame
>
> dice[1, "sum_of_dice"]
[1] 5
>
> # Get a row of a data frame
>
> dice[1,]
d1 d2 sum_of_dice
1 1 4 5
>
> # Get a column of a data frame
>
> head(dice[,"sum_of_dice"])
[1] 5 8 4 5 6 6
> head(dice$sum_of_dice)
[1] 5 8 4 5 6 6
>
> # Why did we use "head" for columns but
> # not for rows?
>
Environment
History
Import Dataset
Clear
Global Environment
Data
dice
10000 obs. of …
Values
a
num [1:3] 1 2 3
b
num [1:3] 4 5 6
Files
Plots
Packages
Help
Zoom
Export


RStudio
Go to file/function
Project: (None)
Console ~/
> # You can create new columns and add
> # them to the data frame:
>
> dice$product_of_dice <- d1 * d2
> head(dice)
  d1 d2 sum_of_dice product_of_dice
1 1 4   5   4
2 5 3   8   15
3 3 1   4   3
4 1 4   5   4
5 2 4   6   8
6 4 2   6   8
>
> mean(dice$product_of_dice)
[1] 12.1161
>
Environment
History
Import Dataset
Clear
Global Environment
Data
dice
10000 obs. of …
Values
a
num [1:3] 1 2 3
b
num [1:3] 4 5 6
Files
Plots
Packages
Help
Zoom
Export

RStudio

Go to file/function

Project: (None)

Console ~/

```
> # Working with data wouldn't be very
> # useful if you couldn't output it to
> # a file:
>
> my_file <- "/Users/jessica/Dropbox/drop-in/tutorials/dice.tsv"
> write.table(dice, file = my_file, quote = FALSE, sep = "\t", row.names = FALSE, col.names = TRUE)
>
```

1. less

d1	d2	sum_of_dice	product_of_dice
1	4	5	4
5	3	8	15
3	1	4	3
1	4	5	4
2	4	6	8
4	2	6	8
4	4	8	16
6	5	11	30
2	6	8	12
6	4	10	24
5	5	10	25
1	3	4	3
2	5	7	10
5	5	10	25
4	5	9	20
2	2	4	4
6	4	10	24
2	1	3	2
3	3	6	9
6	3	9	18
4	6	10	24
6	5	11	30
6	5	11	30

dice.tsv

Environment | History

Import Dataset | Clear

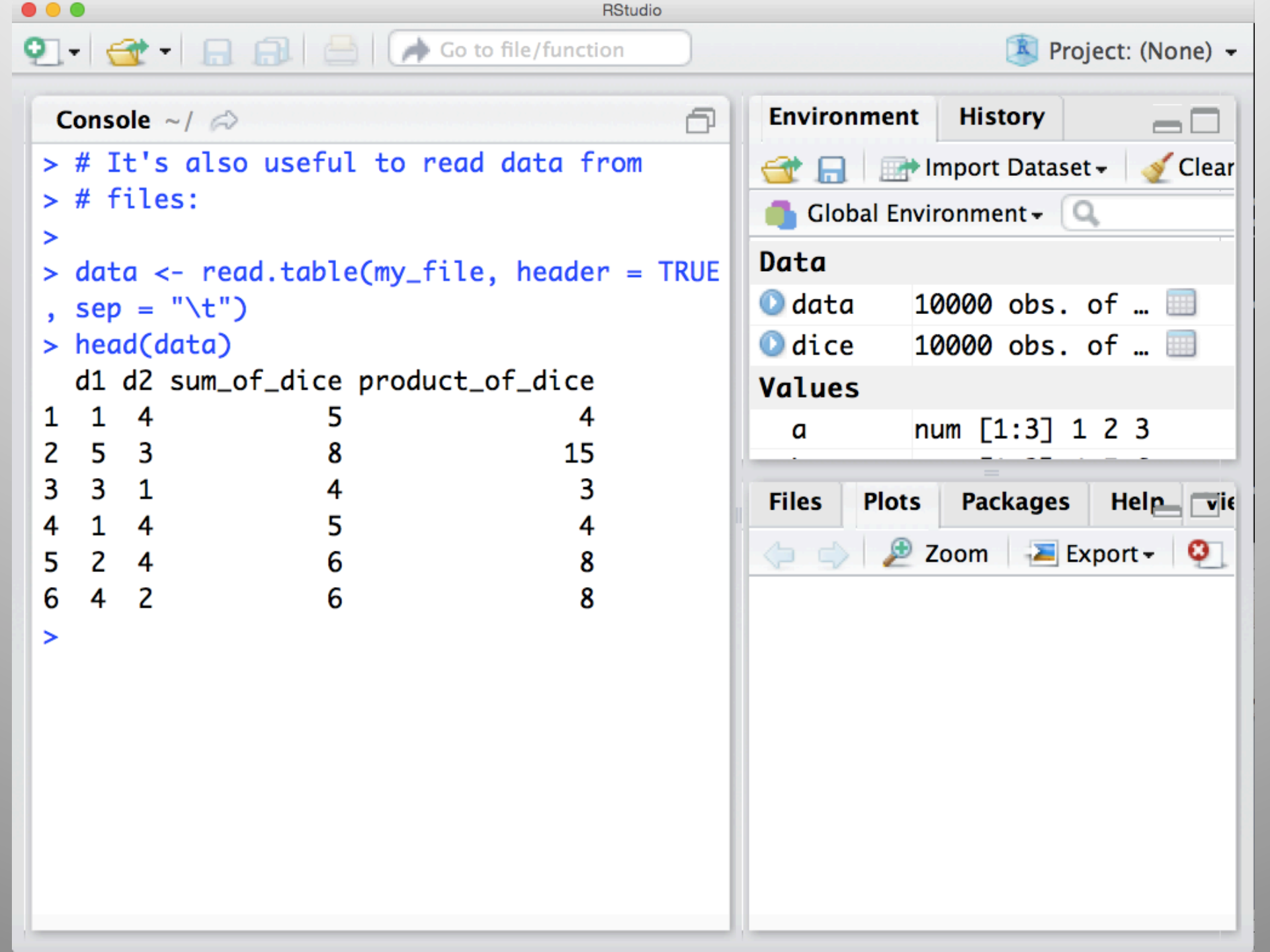
Global Environment

Values

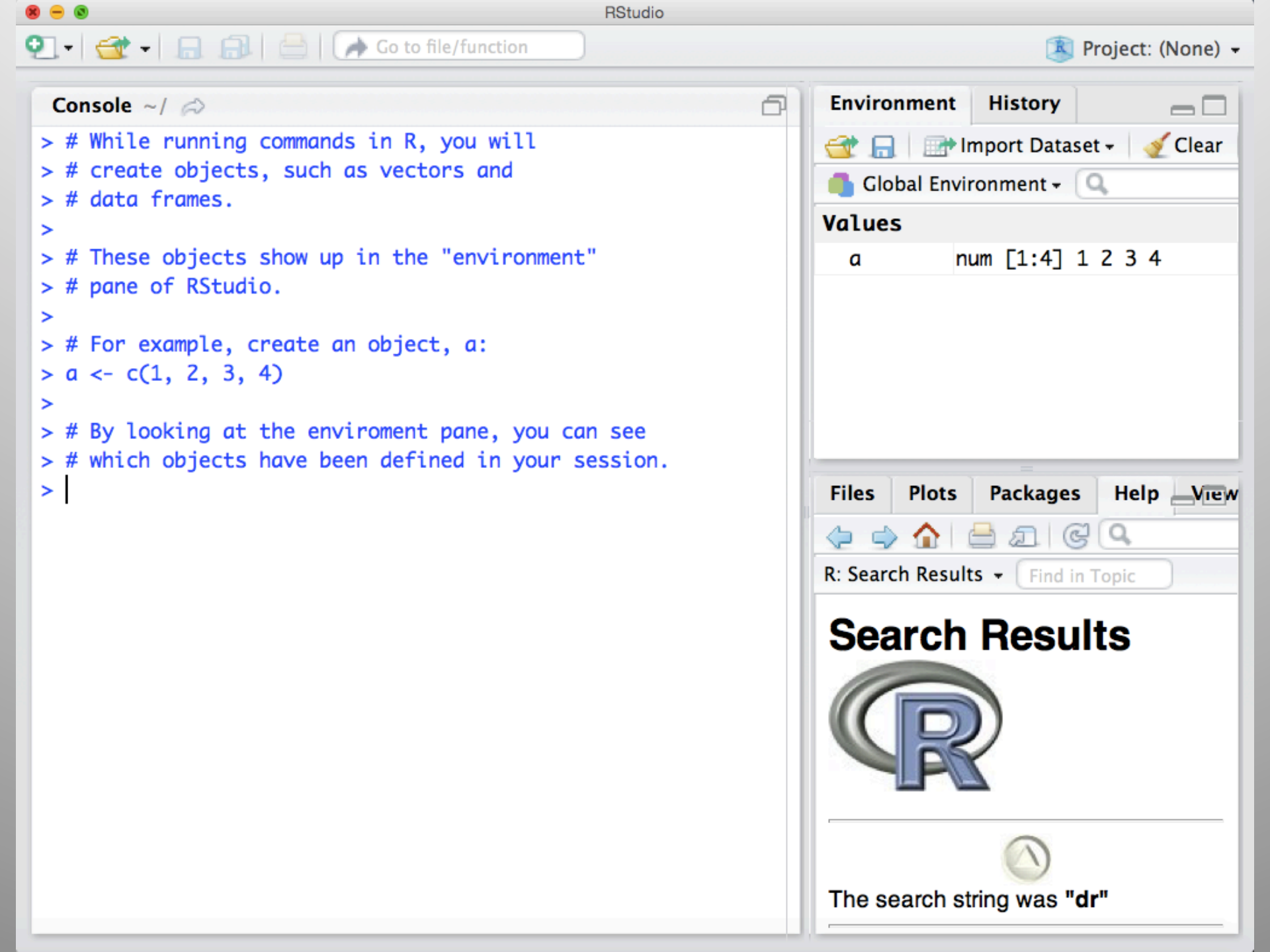
a	num [1:3] 1 2 3
b	num [1:3] 4 5 6
d1	int [1:10000] 1 …
d2	int [1:10000] 4 …
my_file	"/Users/jessica/…

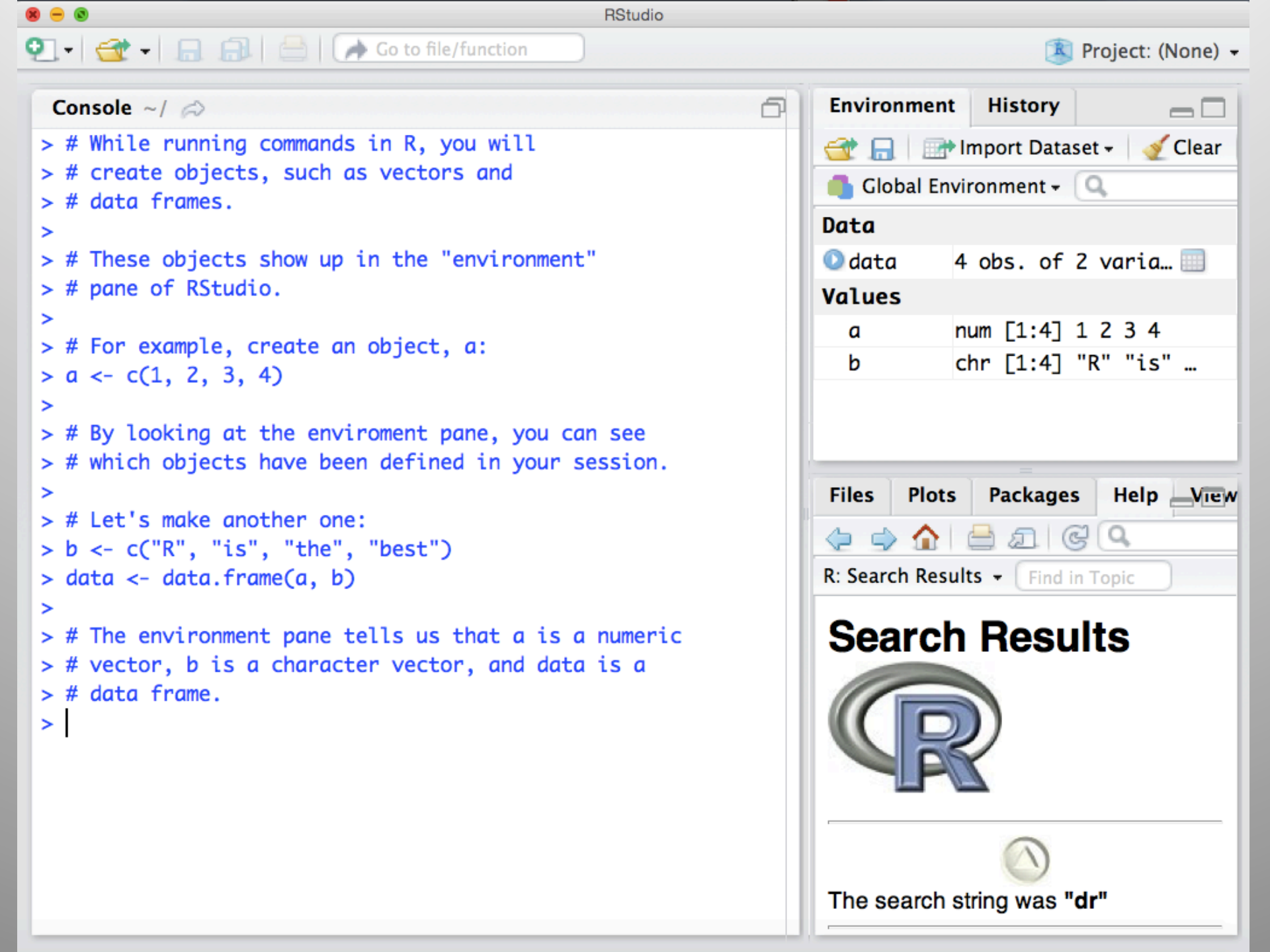
Files | Plots | Packages | Help | Viewer

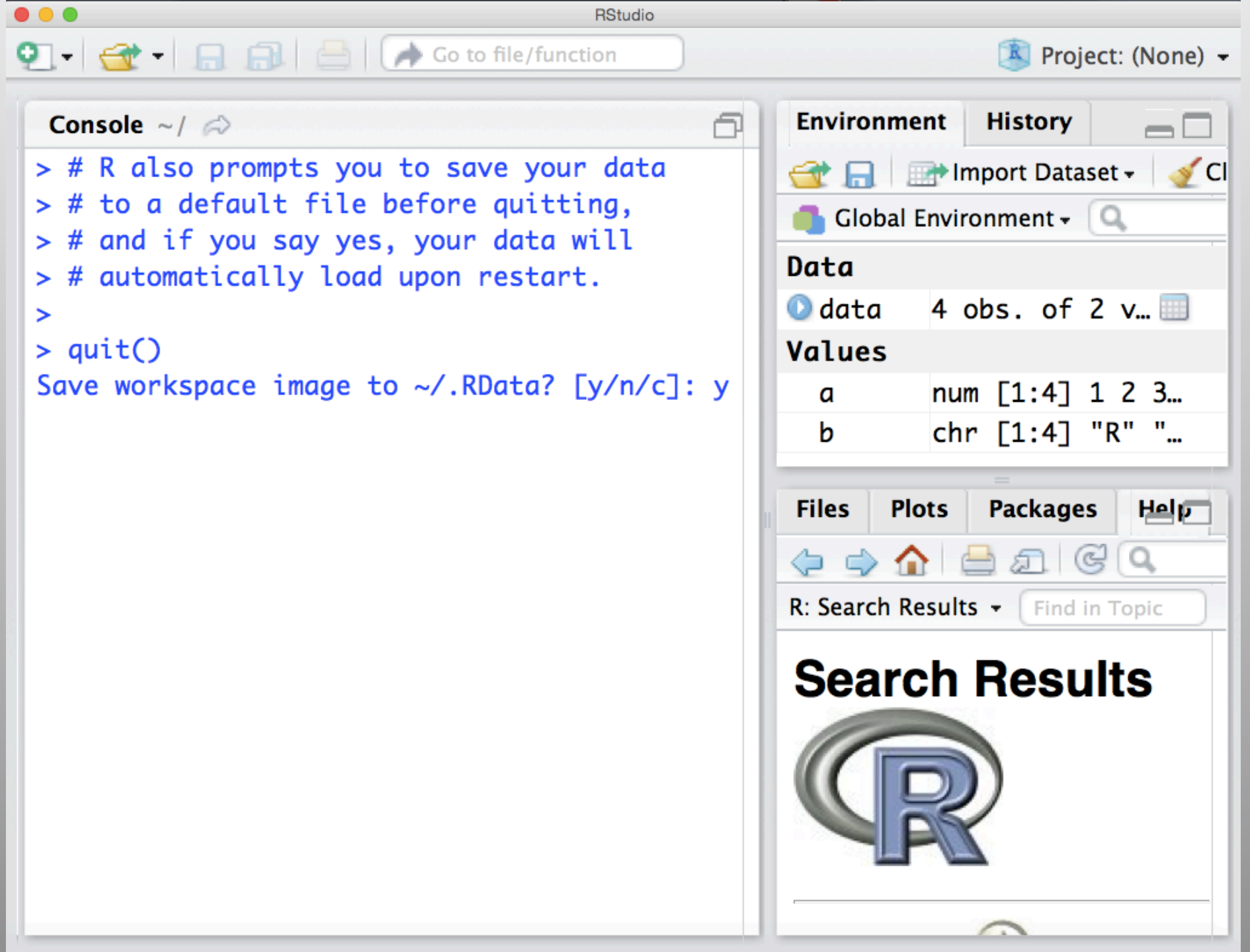
Zoom | Export

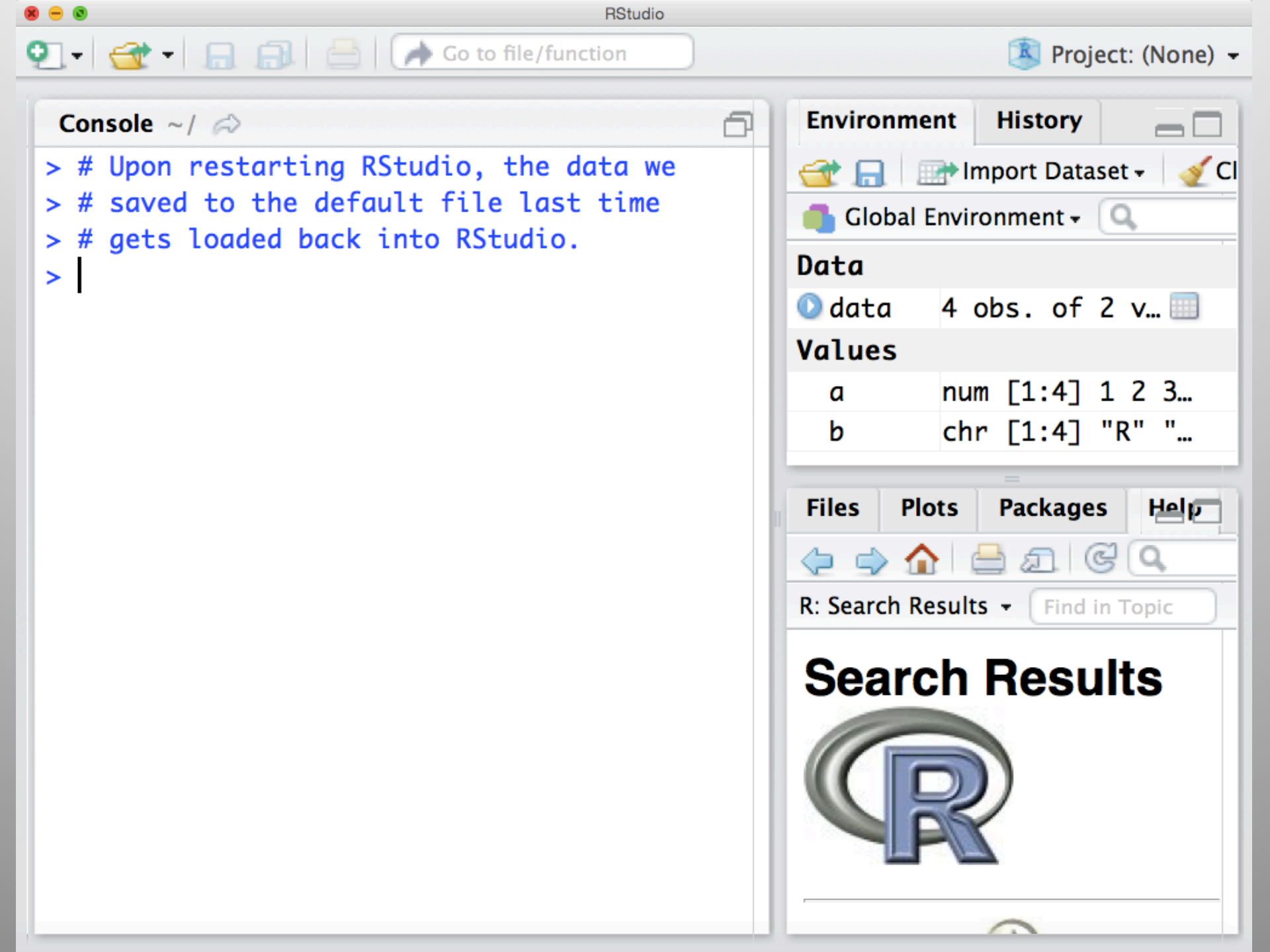

RStudio
Go to file/function
Project: (None)
Console ~/
> # It's also useful to read data from
> # files:
>
> data <- read.table(my_file, header = TRUE
, sep = "\t")
> head(data)
  d1 d2 sum_of_dice product_of_dice
1 1 4   5   4
2 5 3   8   15
3 3 1   4   3
4 1 4   5   4
5 2 4   6   8
6 4 2   6   8
>
Environment
History
Import Dataset
Clear
Global Environment
Data
data 10000 obs. of …
dice 10000 obs. of …
Values
a num [1:3] 1 2 3
Files
Plots
Packages
Help
Zoom
Export

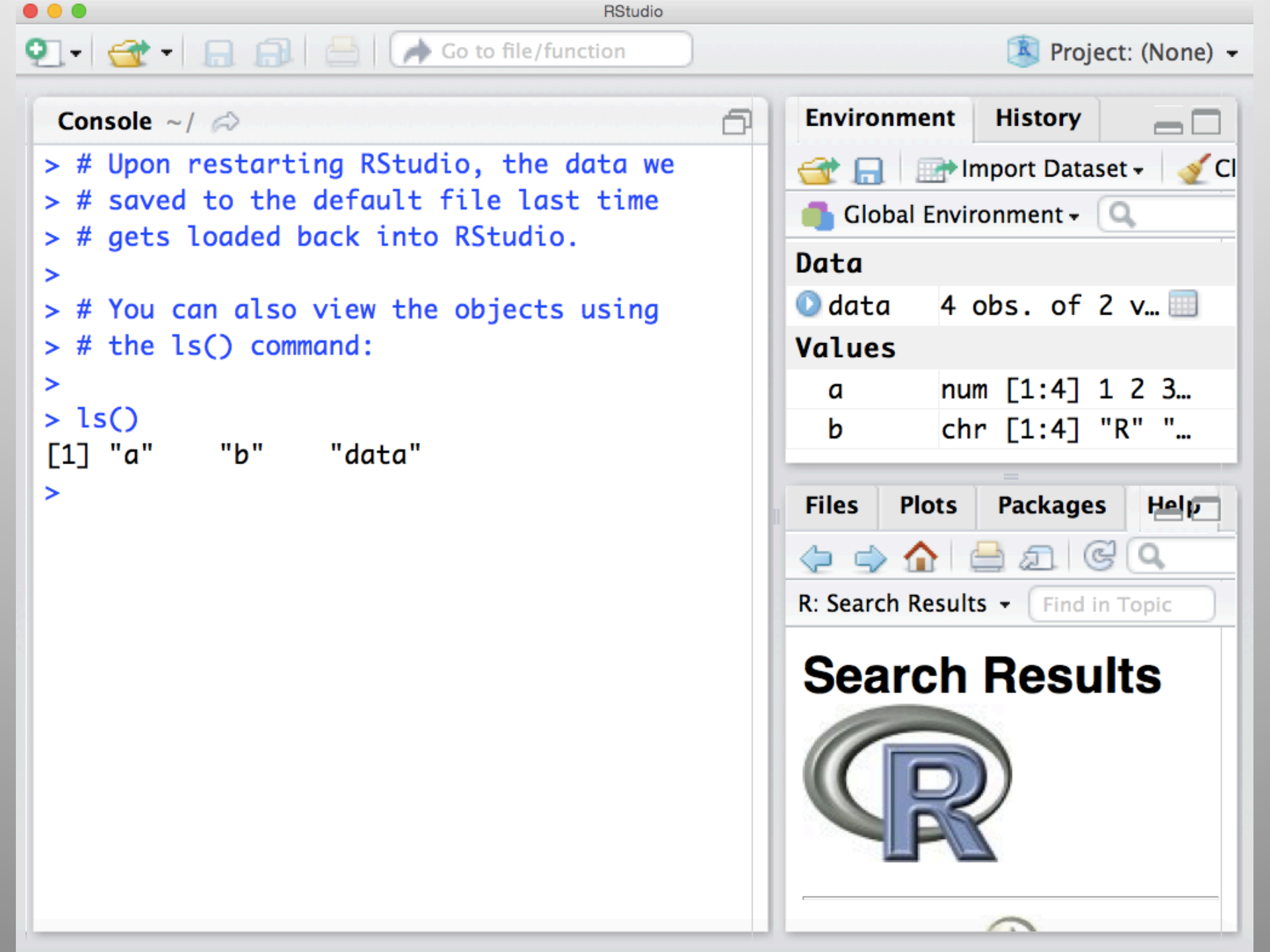
RStudio

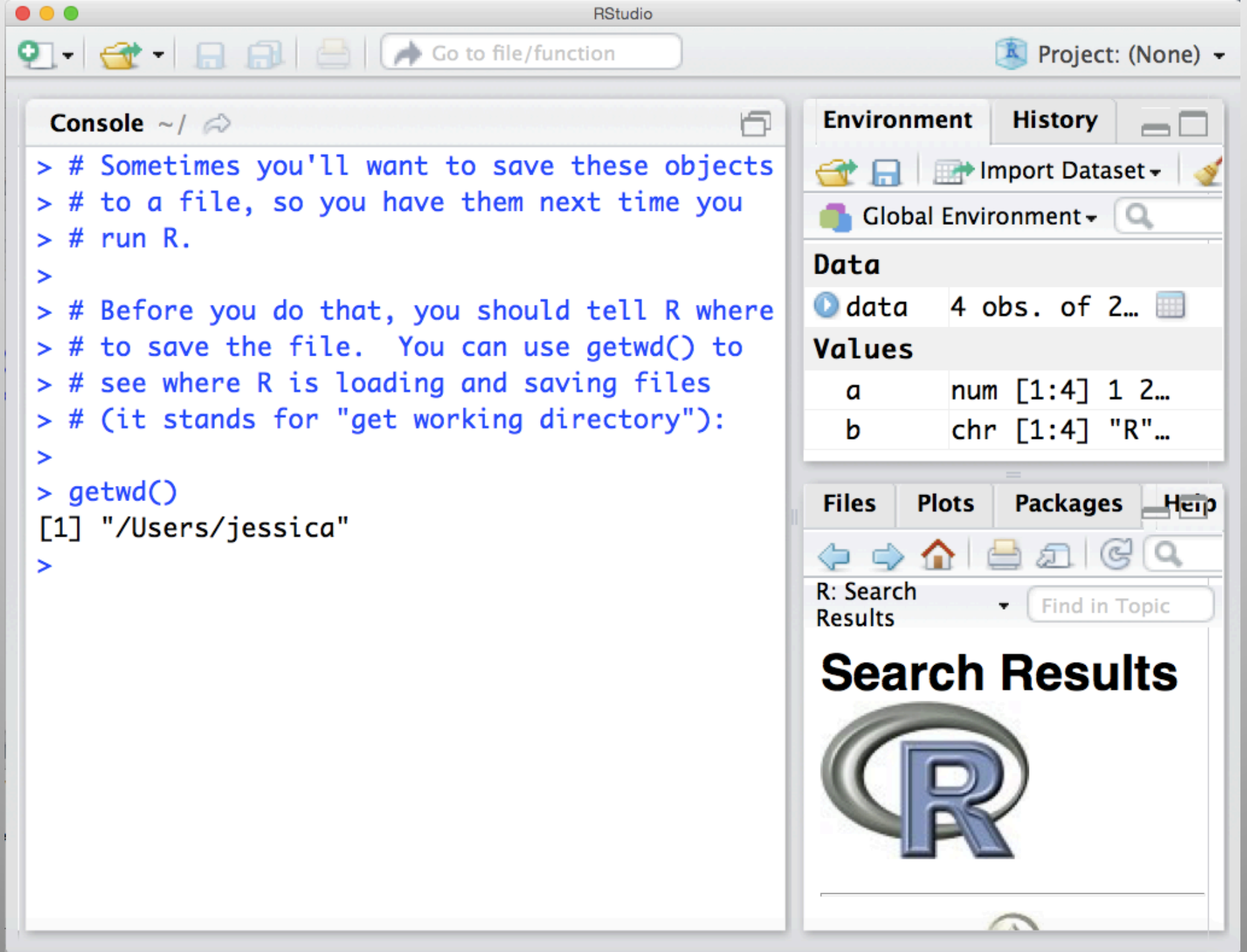

RStudio
Go to file/function
Project: (None)
Console ~/
> # While running commands in R, you will
> # create objects, such as vectors and
> # data frames.
>
> # These objects show up in the "environment"
> # pane of RStudio.
>
> # For example, create an object, a:
> a <- c(1, 2, 3, 4)
>
> # By looking at the enviroment pane, you can see
> # which objects have been defined in your session.
>
Environment
History
Import Dataset
Clear
Global Environment
Values
a
num [1:4] 1 2 3 4
Files
Plots
Packages
Help
View
R: Search Results
Find in Topic
Search Results
The search string was "dr"

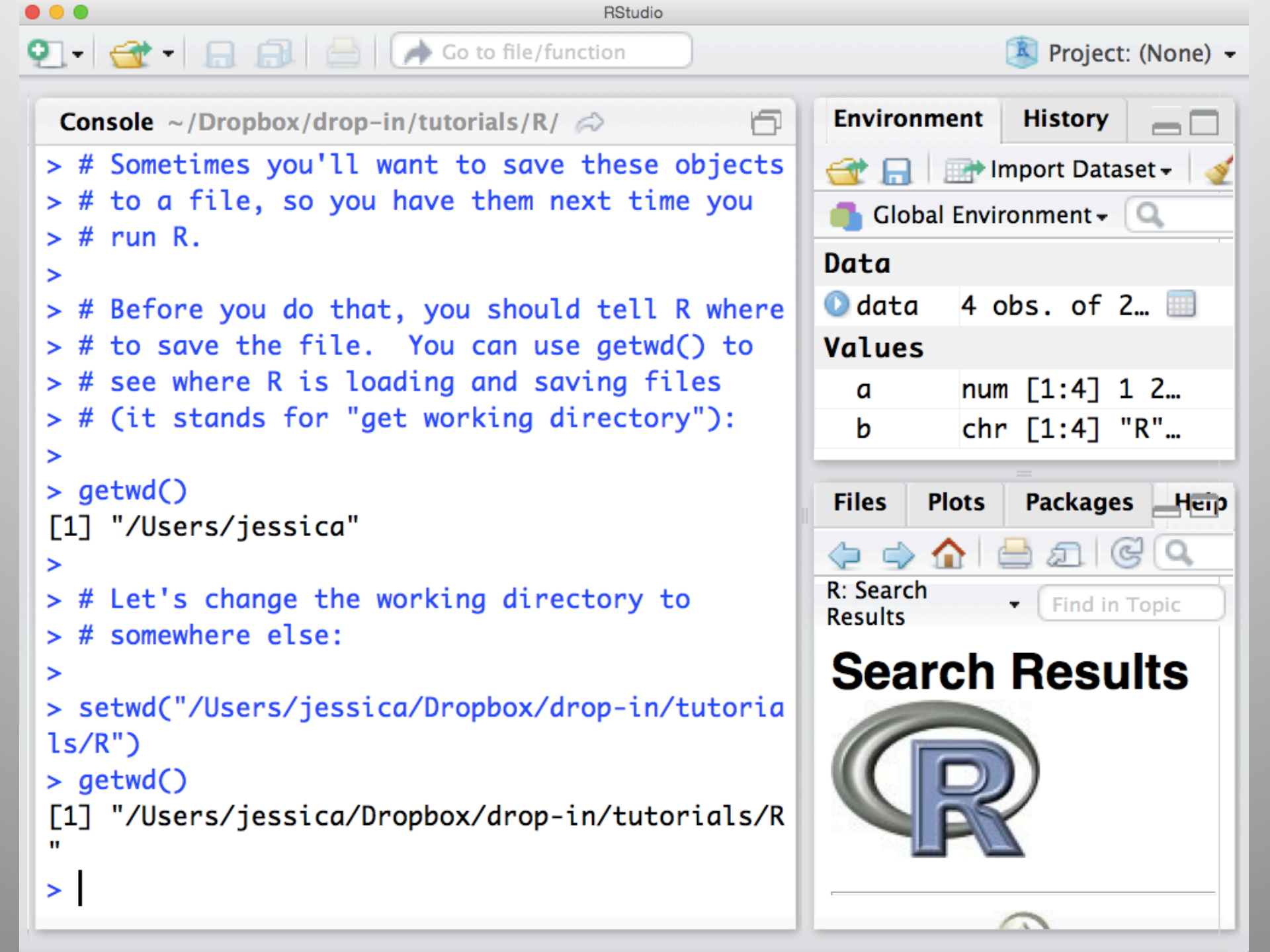

RStudio
Go to file/function
Project: (None)
Console ~/
> # While running commands in R, you will
> # create objects, such as vectors and
> # data frames.
>
> # These objects show up in the "environment"
> # pane of RStudio.
>
> # For example, create an object, a:
> a <- c(1, 2, 3, 4)
>
> # By looking at the enviroment pane, you can see
> # which objects have been defined in your session.
>
> # Let's make another one:
> b <- c("R", "is", "the", "best")
> data <- data.frame(a, b)
>
> # The environment pane tells us that a is a numeric
> # vector, b is a character vector, and data is a
> # data frame.
>
Environment
History
Import Dataset
Clear
Global Environment
Data
data
4 obs. of 2 varia…
Values
a
num [1:4] 1 2 3 4
b
chr [1:4] "R" "is" …
Files
Plots
Packages
Help
View
R: Search Results
Find in Topic
Search Results
The search string was "dr"

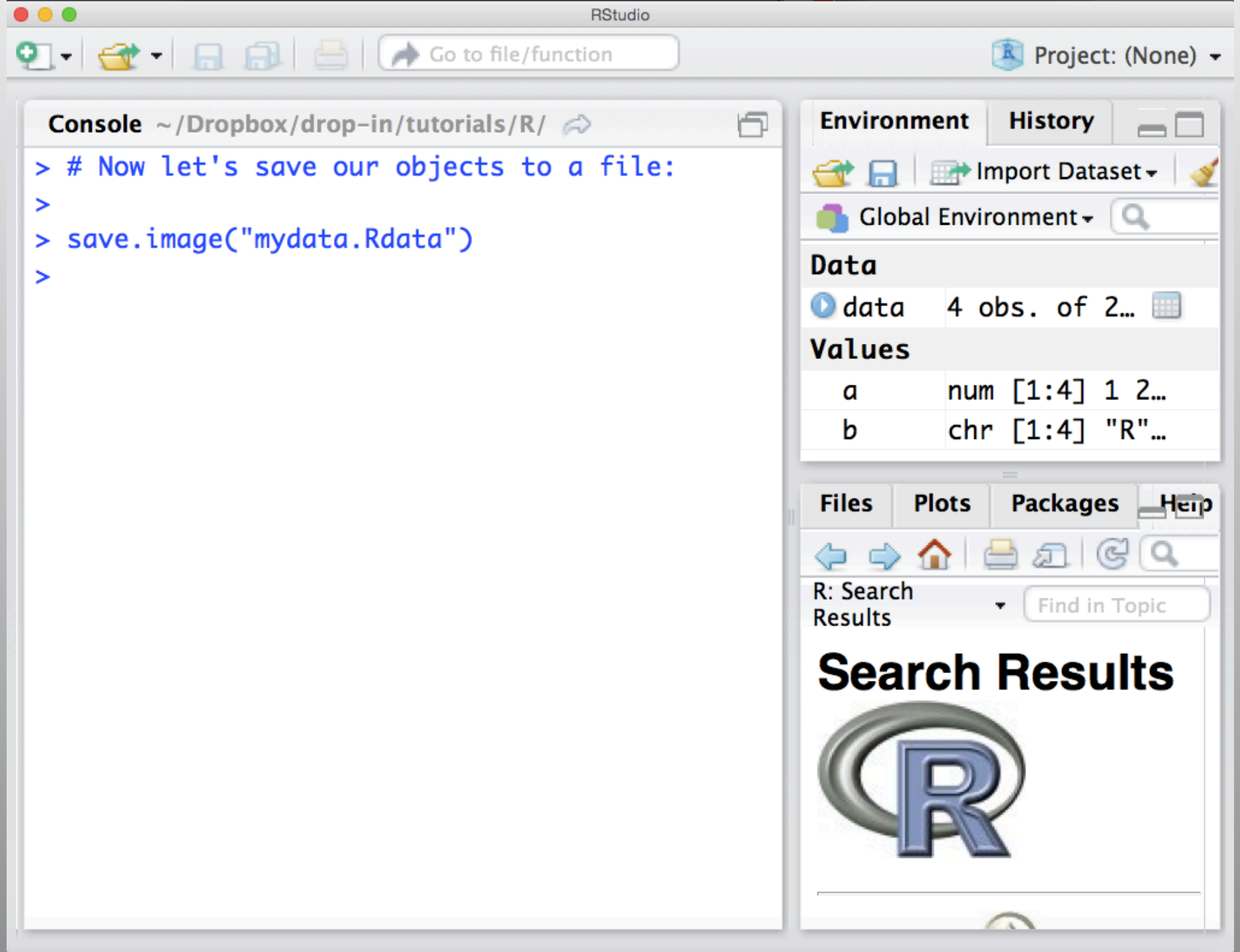

RStudio
Go to file/function
Project: (None)
Console ~/
> # R also prompts you to save your data
> # to a default file before quitting,
> # and if you say yes, your data will
> # automatically load upon restart.
>
> quit()
Save workspace image to ~/.RData? [y/n/c]: y
Environment
History
Import Dataset
Global Environment
Data
data
4 obs. of 2 v...
Values
a
num [1:4] 1 2 3...
b
chr [1:4] "R" "...
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results

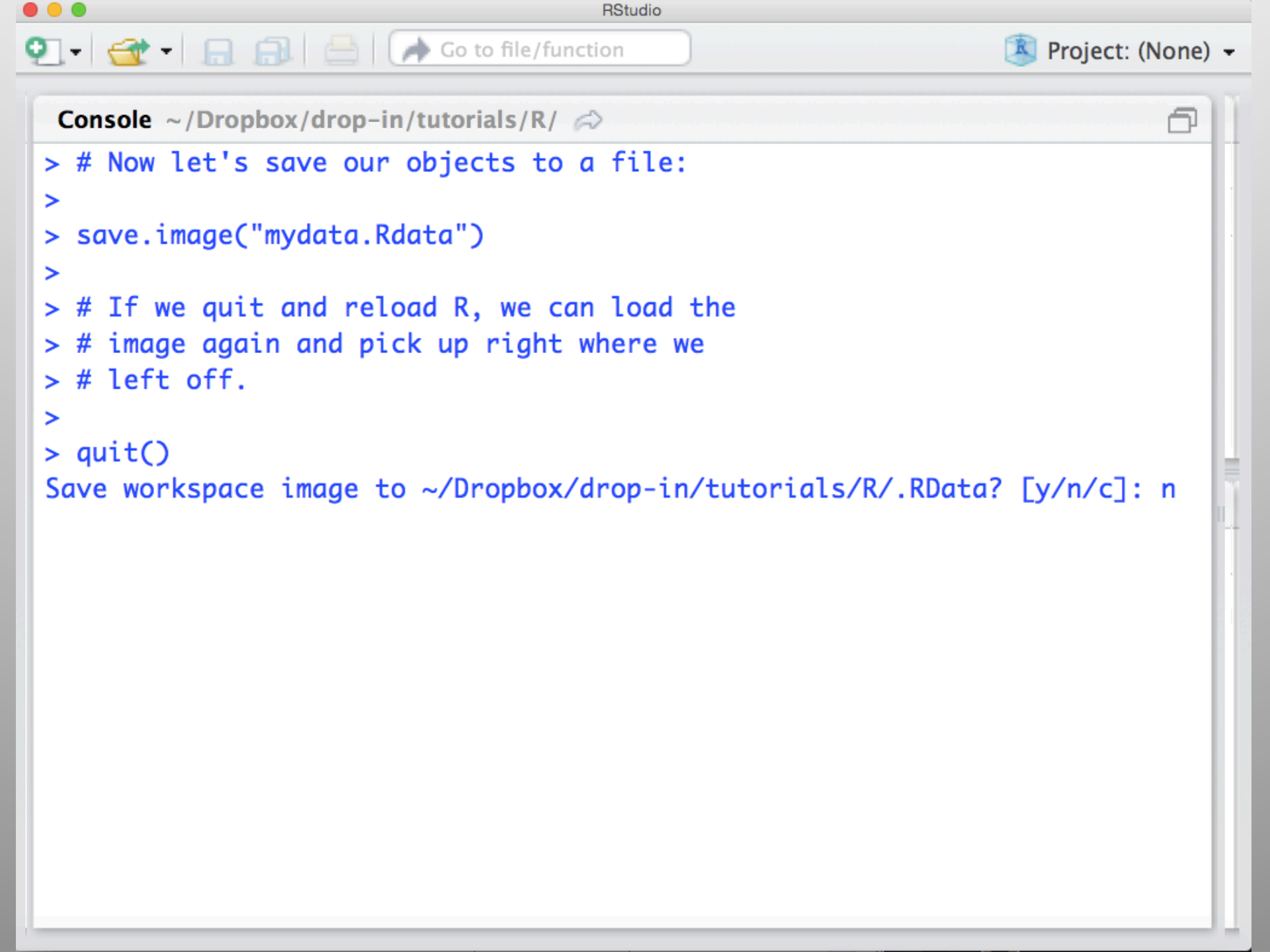

RStudio
Go to file/function
Project: (None)
Console ~/
> # Upon restarting RStudio, the data we
> # saved to the default file last time
> # gets loaded back into RStudio.
>
Environment
History
Import Dataset
Global Environment
Data
data
4 obs. of 2 v...
Values
a
num [1:4] 1 2 3...
b
chr [1:4] "R" "...
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results

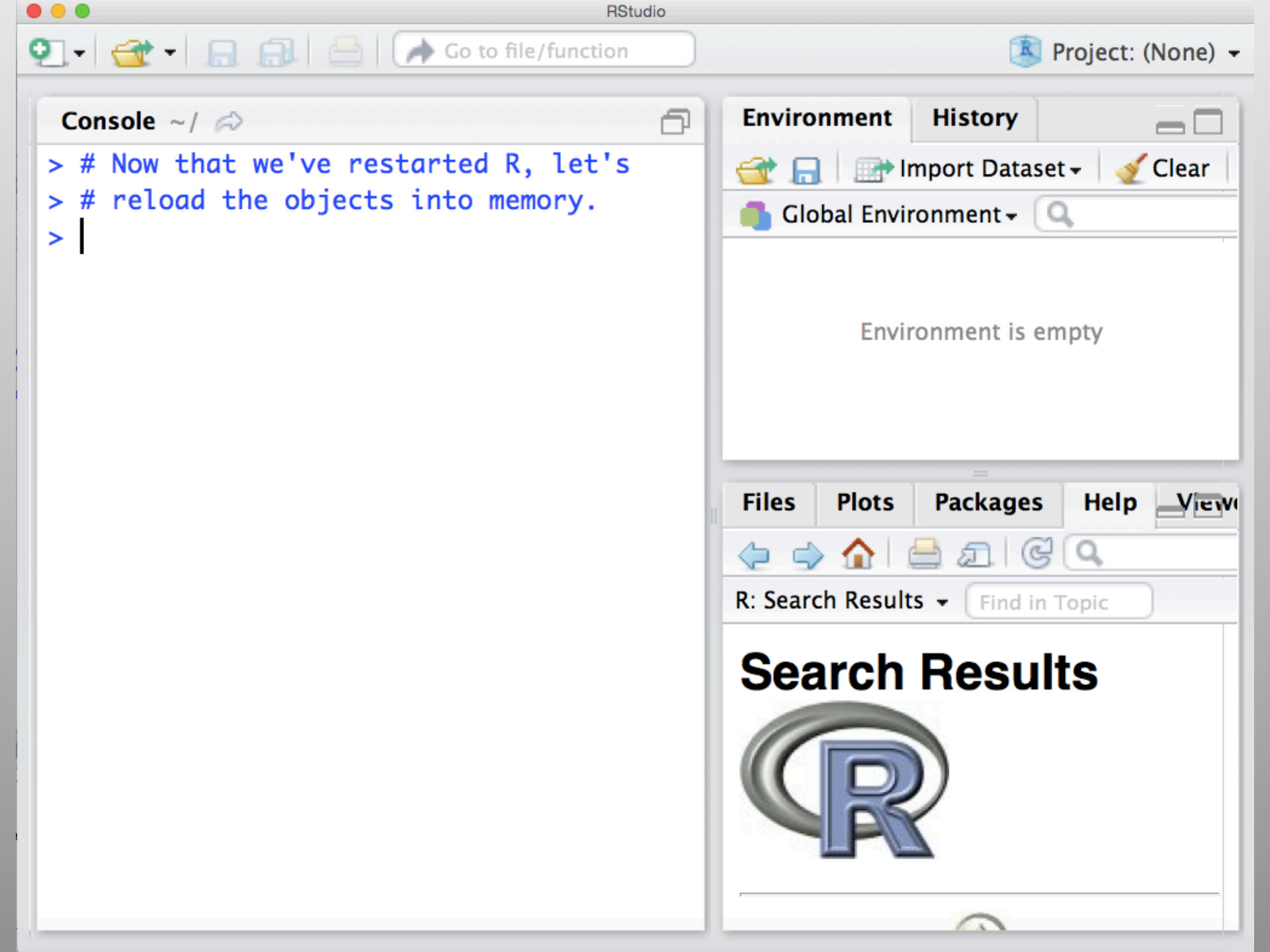

RStudio
Go to file/function
Project: (None)
Console ~/
> # Upon restarting RStudio, the data we
> # saved to the default file last time
> # gets loaded back into RStudio.
>
> # You can also view the objects using
> # the ls() command:
>
> ls()
[1] "a"   "b"   "data"
>
Environment
History
Import Dataset
Global Environment
Data
data
4 obs. of 2 v...
Values
a
num [1:4] 1 2 3...
b
chr [1:4] "R" "...
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results

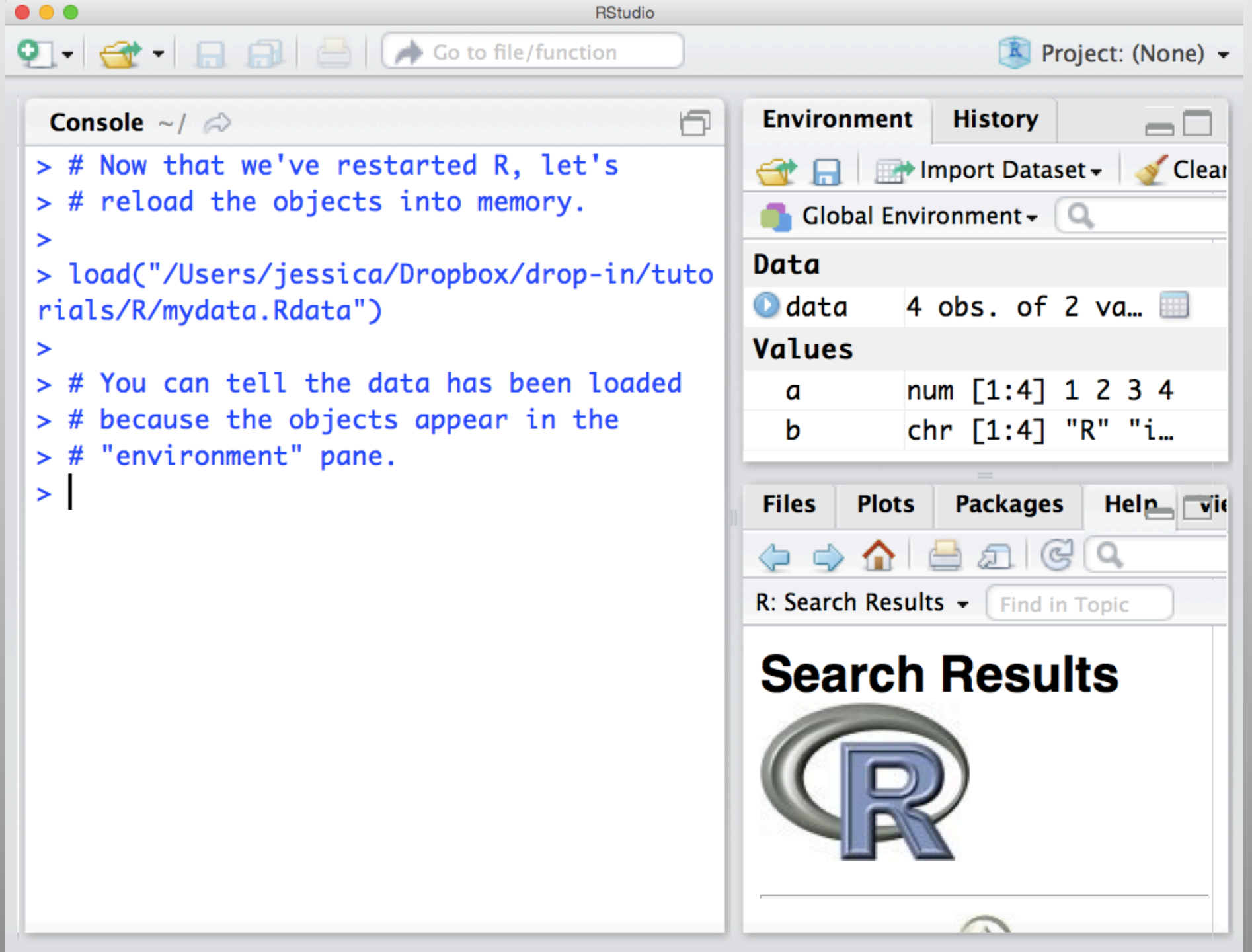

RStudio
Go to file/function
Project: (None)
Console ~/
> # Sometimes you'll want to save these objects
> # to a file, so you have them next time you
> # run R.
>
> # Before you do that, you should tell R where
> # to save the file. You can use getwd() to
> # see where R is loading and saving files
> # (it stands for "get working directory"):
>
> getwd()
[1] "/Users/jessica"
>
Environment
History
Import Dataset
Global Environment
Data
data
4 obs. of 2…
Values
a
num [1:4] 1 2…
b
chr [1:4] "R"…
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results

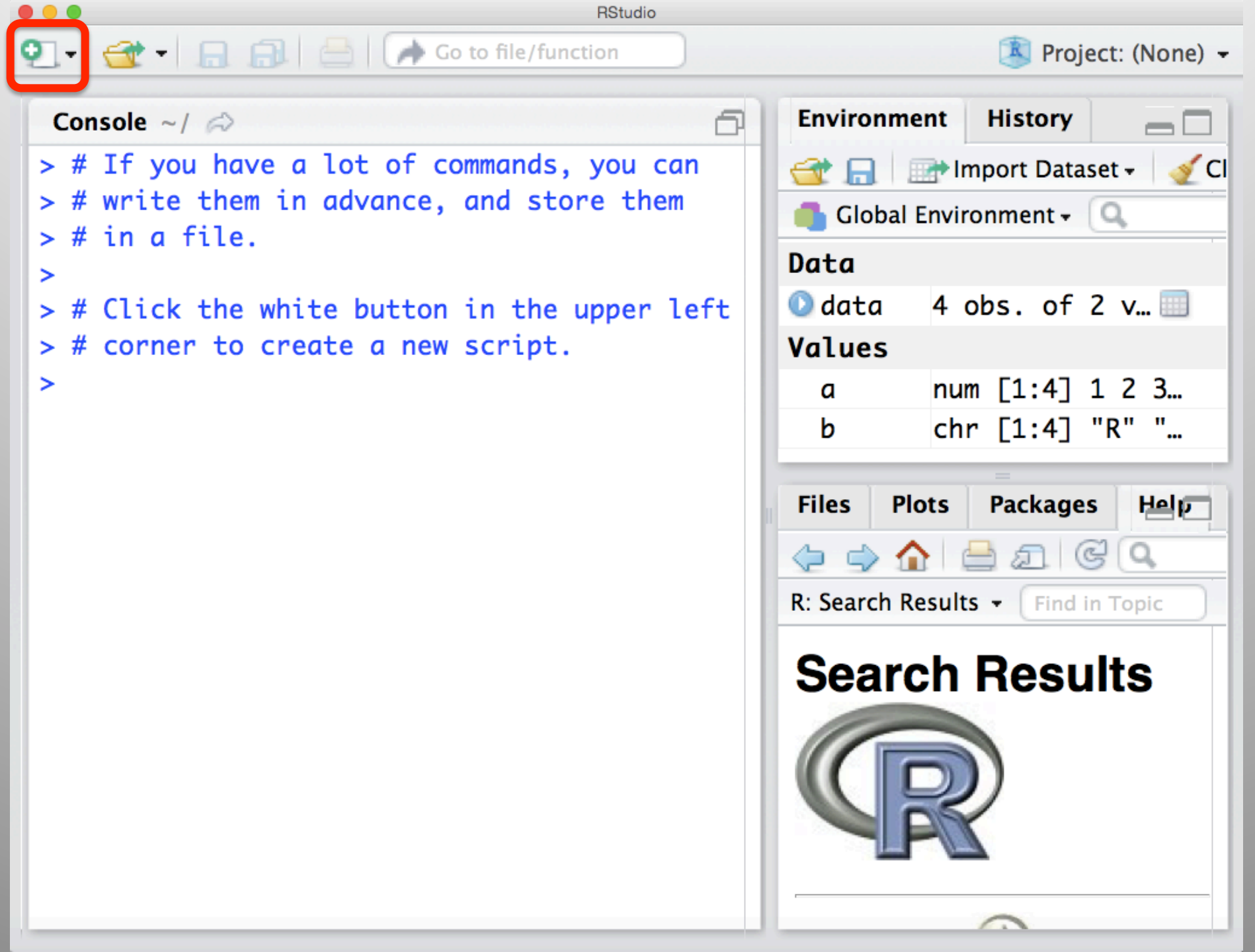

RStudio
Go to file/function
Project: (None)
Console ~/Dropbox/drop-in/tutorials/R/
> # Sometimes you'll want to save these objects
> # to a file, so you have them next time you
> # run R.
>
> # Before you do that, you should tell R where
> # to save the file. You can use getwd() to
> # see where R is loading and saving files
> # (it stands for "get working directory"):
>
> getwd()
[1] "/Users/jessica"
>
> # Let's change the working directory to
> # somewhere else:
>
> setwd("/Users/jessica/Dropbox/drop-in/tutorials/R")
> getwd()
[1] "/Users/jessica/Dropbox/drop-in/tutorials/R"
>
Environment
History
Import Dataset
Global Environment
Data
data
4 obs. of 2…
Values
a
num [1:4] 1 2…
b
chr [1:4] "R"…
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results

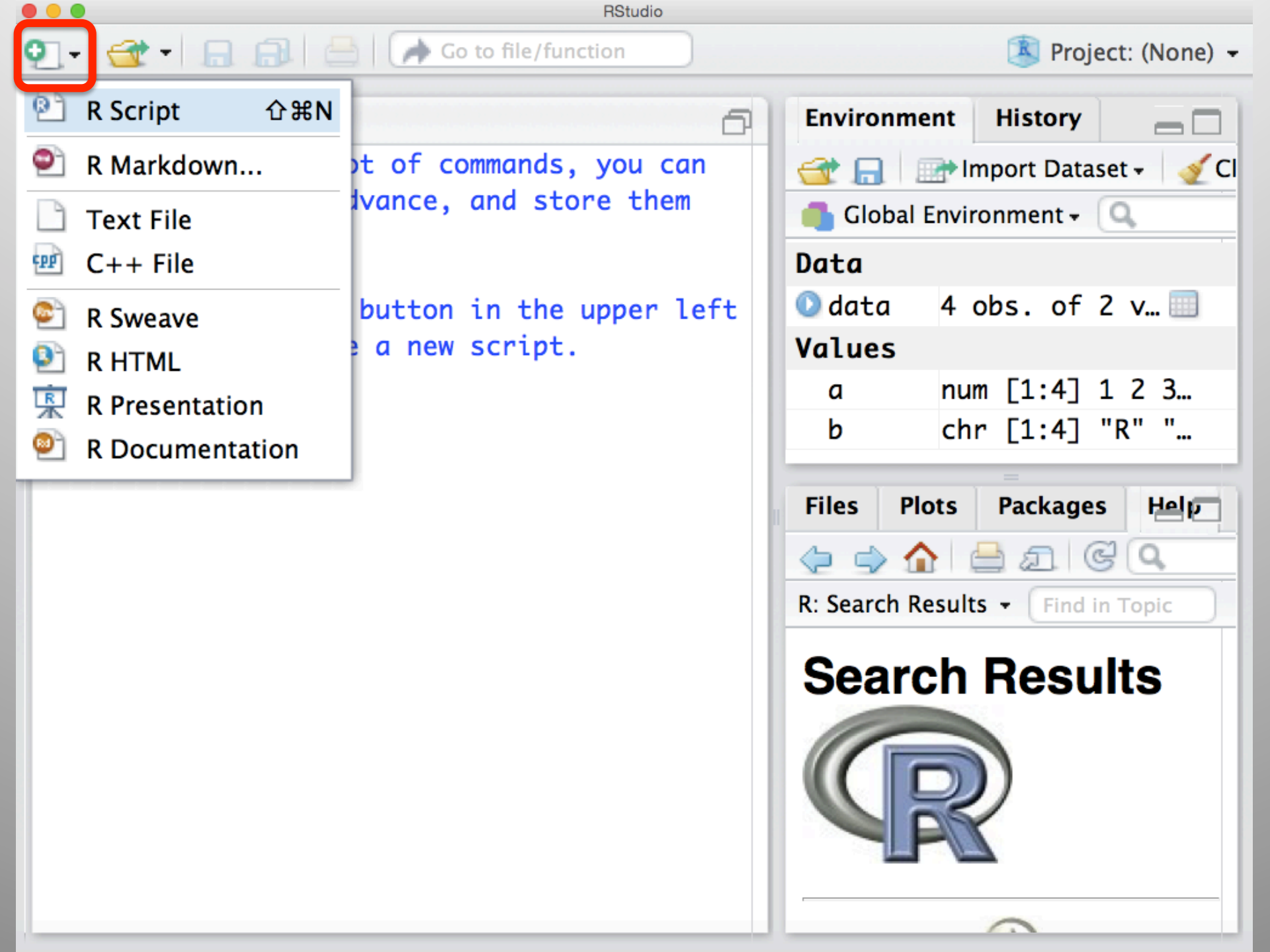

RStudio
Go to file/function
Project: (None)
Console ~/Dropbox/drop-in/tutorials/R/
> # Now let's save our objects to a file:
>
> save.image("mydata.Rdata")
>
Environment
History
Import Dataset
Global Environment
Data
data
4 obs. of 2…
Values
a
num [1:4] 1 2…
b
chr [1:4] "R"…
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results

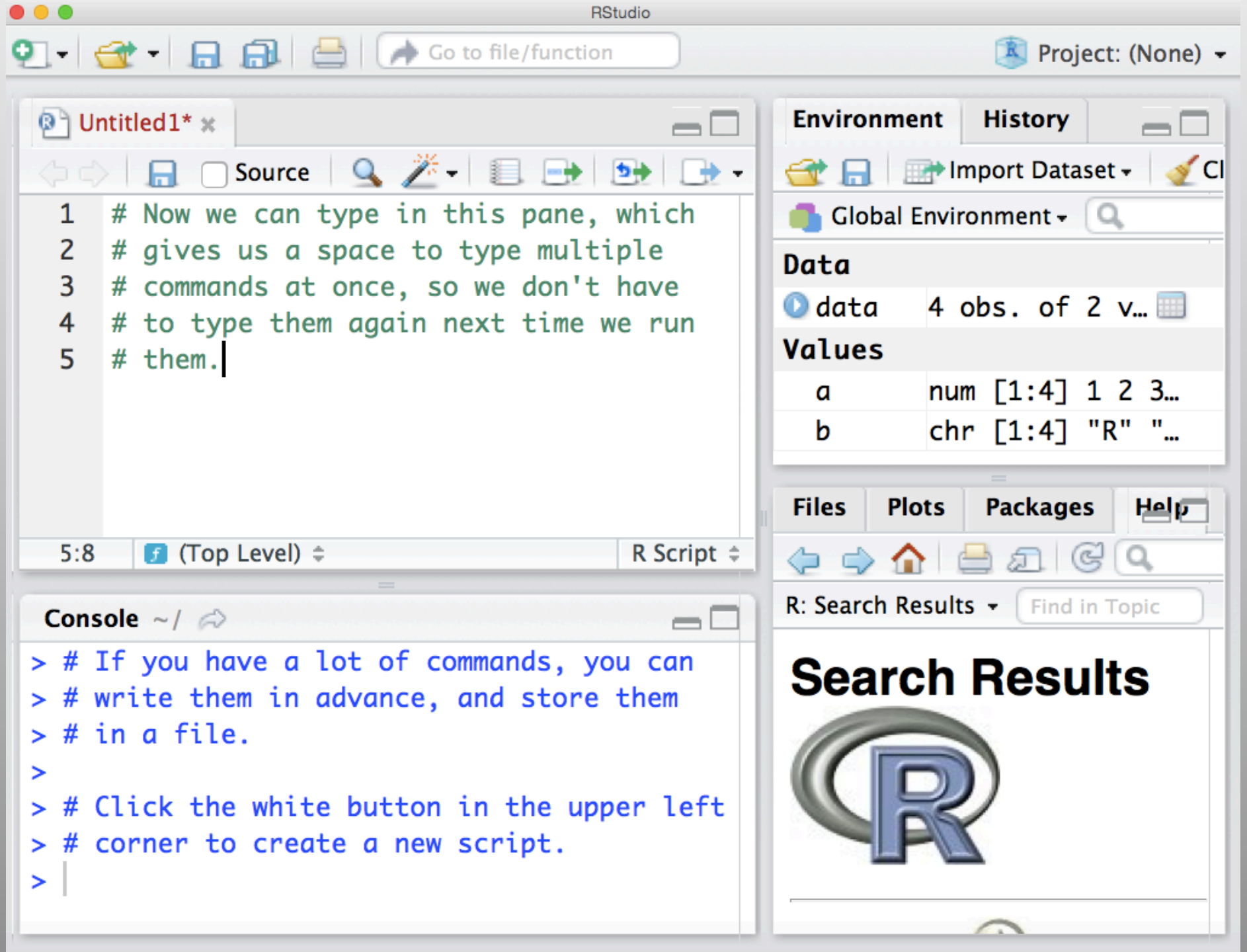

RStudio
Go to file/function
Project: (None)
Console ~/Dropbox/drop-in/tutorials/R/
> # Now let's save our objects to a file:
>
> save.image("mydata.Rdata")
>
> # If we quit and reload R, we can load the
> # image again and pick up right where we
> # left off.
>
> quit()
Save workspace image to ~/Dropbox/drop-in/tutorials/R/.RData? [y/n/c]: n

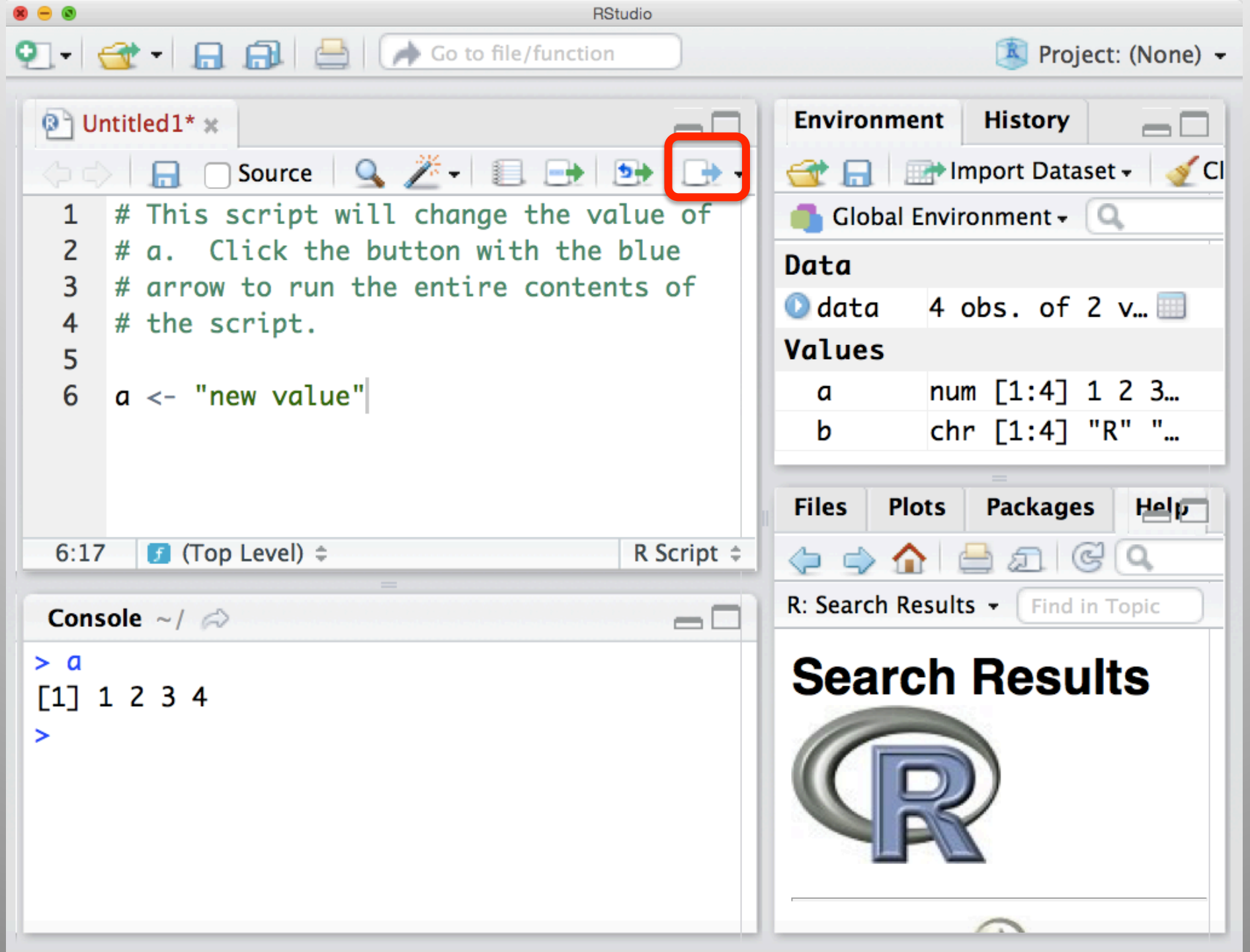

RStudio
Go to file/function
Project: (None)
Console ~/
> # Now that we've restarted R, let's
> # reload the objects into memory.
>
Environment
History
Import Dataset
Clear
Global Environment
Environment is empty
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results

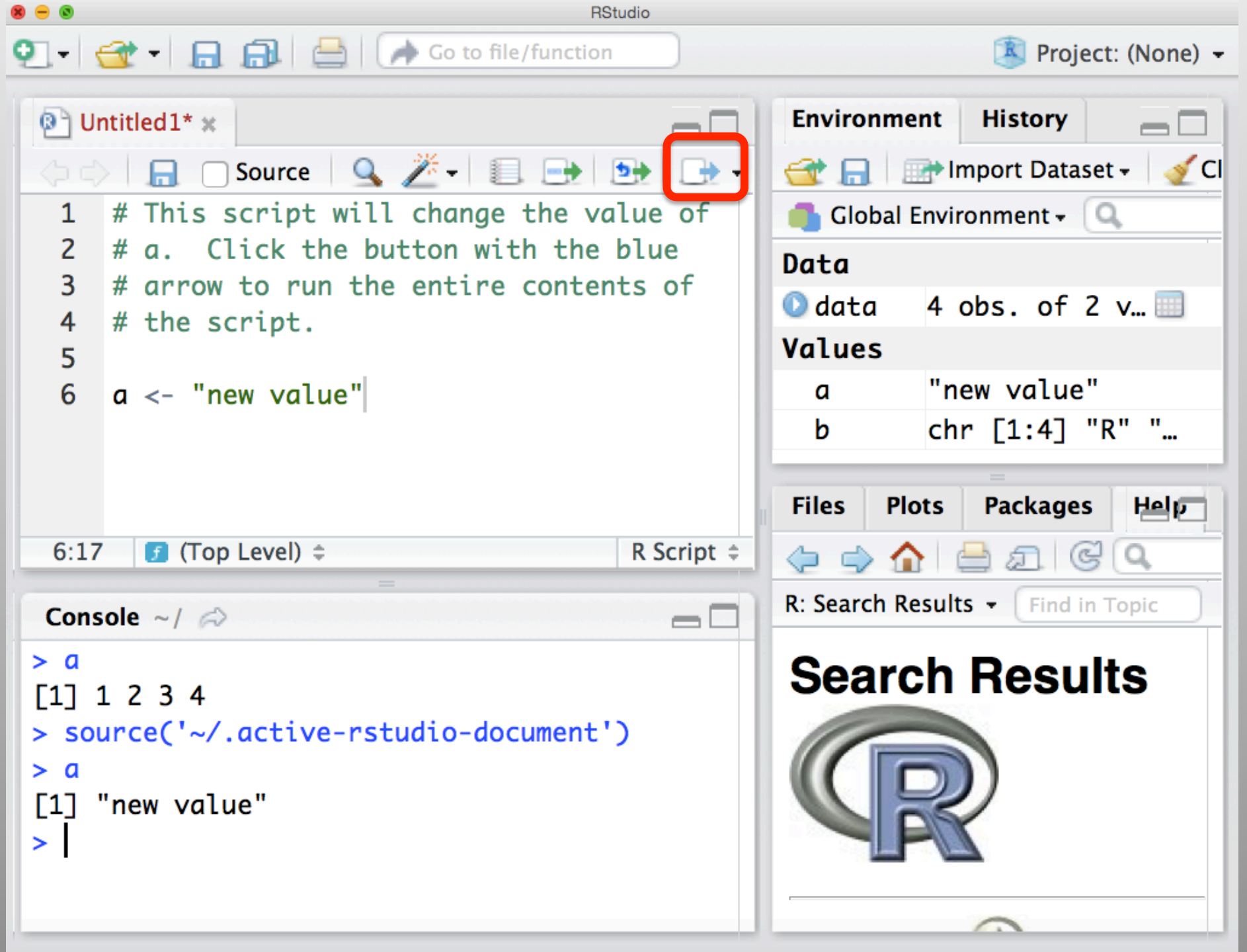

RStudio
Go to file/function
Project: (None)
Console ~/
> # Now that we've restarted R, let's
> # reload the objects into memory.
>
> load("/Users/jessica/Dropbox/drop-in/tuto
rials/R/mydata.Rdata")
>
> # You can tell the data has been loaded
> # because the objects appear in the
> # "environment" pane.
>
Environment
History
Import Dataset
Clear
Global Environment
Data
data
4 obs. of 2 va…
Values
a
num [1:4] 1 2 3 4
b
chr [1:4] "R" "i…
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results


RStudio
Go to file/function
Project: (None)
Console ~/
> # If you have a lot of commands, you can
> # write them in advance, and store them
> # in a file.
>
> # Click the white button in the upper left
> # corner to create a new script.
>
Environment
History
Import Dataset
Global Environment
Data
data
4 obs. of 2 v...
Values
a
num [1:4] 1 2 3...
b
chr [1:4] "R" "...
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results

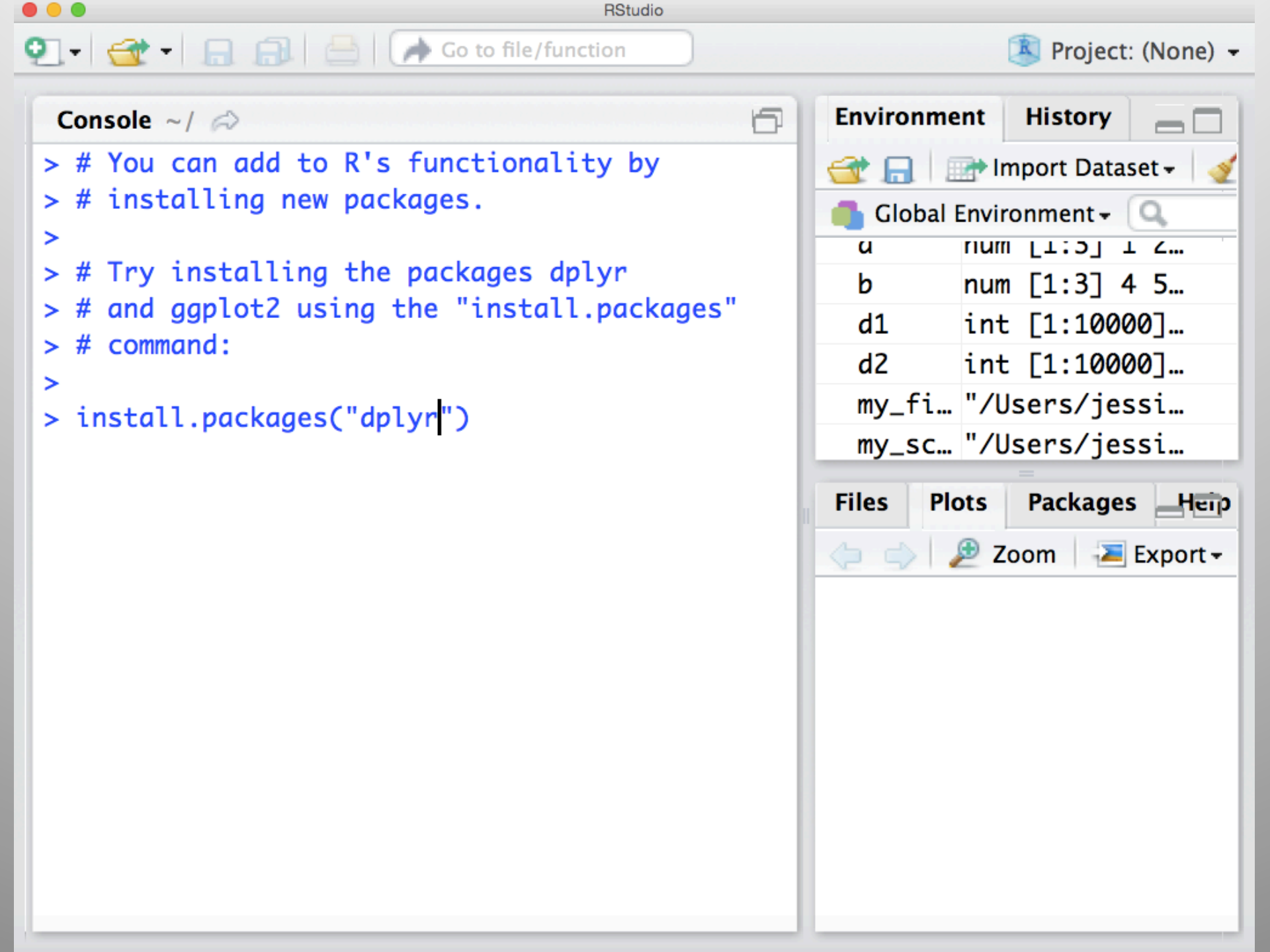

RStudio
Go to file/function
Project: (None)
R Script ⇧⌘N
R Markdown...
Text File
C++ File
R Sweave
R HTML
R Presentation
R Documentation
of commands, you can
dvance, and store them
button in the upper left
a new script.
Environment
History
Import Dataset
Global Environment
Data
data
4 obs. of 2 v...
Values
a
num [1:4] 1 2 3...
b
chr [1:4] "R" "...
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results


RStudio
Go to file/function
Project: (None)
Untitled1*
Source
1 # Now we can type in this pane, which
2 # gives us a space to type multiple
3 # commands at once, so we don't have
4 # to type them again next time we run
5 # them.
5:8
(Top Level)
R Script
Console ~/
> # If you have a lot of commands, you can
> # write them in advance, and store them
> # in a file.
>
> # Click the white button in the upper left
> # corner to create a new script.
>
Environment
History
Import Dataset
Global Environment
Data
data 4 obs. of 2 v...
Values
a num [1:4] 1 2 3...
b chr [1:4] "R" "...
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results


RStudio
Go to file/function
Project: (None)
Untitled1*
Source
1 # This script will change the value of
2 # a. Click the button with the blue
3 # arrow to run the entire contents of
4 # the script.
5
6 a <- "new value"
6:17
(Top Level)
R Script
Console ~/
> a
[1] 1 2 3 4
>
Environment
History
Import Dataset
Global Environment
Data
data 4 obs. of 2 v...
Values
a num [1:4] 1 2 3...
b chr [1:4] "R" "...
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results


RStudio
Go to file/function
Project: (None)
Untitled1*
Source
1 # This script will change the value of
2 # a. Click the button with the blue
3 # arrow to run the entire contents of
4 # the script.
5
6 a <- "new value"
6:17
(Top Level)
R Script
Console ~/
> a
[1] 1 2 3 4
> source('~/.active-rstudio-document')
> a
[1] "new value"
>
Environment
History
Import Dataset
Global Environment
Data
data 4 obs. of 2 v…
Values
a "new value"
b chr [1:4] "R" "…
Files
Plots
Packages
Help
R: Search Results
Find in Topic
Search Results

Installing packages


RStudio
Go to file/function
Project: (None)
Console ~/
> # You can add to R's functionality by
> # installing new packages.
>
> # Try installing the packages dplyr
> # and ggplot2 using the "install.packages"
> # command:
>
> install.packages("dplyr")
Environment
History
Import Dataset
Global Environment
b num [1:3] 4 5…
d1 int [1:10000]…
d2 int [1:10000]…
my_fi… "/Users/jessi…
my_sc… "/Users/jessi…
Files
Plots
Packages
Help
Zoom
Export

Homework

- **Install the packages "dplyr" and "ggplot2."**
- Simulate an experiment with 10000 trials.
  - Flip 3 coins in each trial
  - For each trial, compute the number of coins that come up "heads."
  - Write this data to a file, then read it back into R.
  - Find the fraction of trials where at least 1 coin comes up heads.
  - How close is this to your theoretical prediction?

RStudio

Go to file/function

Project: (None)

Console ~/

```
> c1 <- sample(0:1, 10000, replace = TRUE)
> c2 <- sample(0:1, 10000, replace = TRUE)
> c3 <- sample(0:1, 10000, replace = TRUE)
> num_heads <- c1 + c2 + c3
>
> did_we_get_a_head <- num_heads >= 1
> head(did_we_get_a_head)
[1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
>
> sum(did_we_get_a_head) / 10000
[1] 0.8703
> 7 / 8
[1] 0.875
>
> data <- data.frame(c1, c2, c3, num_heads, did_we_get_a_head)
> filename = "/Users/jessica/Dropbox/drop-in/tutorials/coins.tsv"
> write.table(data, file = filename, quote = FALSE, sep = "\t", row.
names = FALSE, col.names = TRUE)
>
> new_data <- read.table(filename, header = TRUE, sep = "\t")
> head(new_data)
  c1 c2 c3 num_heads did_we_get_a_head
1  0  1  0         1              TRUE
2  0  1  1         2              TRUE
3  0  1  1         2              TRUE
4  0  1  0         1              TRUE
5  1  0  0         1              TRUE
6  0  0  0         0             FALSE
```

Environment | History

Import Dataset

Global Environment

data	10000 obs. o...
new_da...	10000 obs. o...

Values

c1	int [1:10000] ...
c2	int [1:10000] ...
c3	int [1:10000] ...
did_we...	logi [1:10000]...
filena...	"/Users/jessic...

Files | Plots | Packages | Help

R: Search Results

Find in Topic

Search Results

The search string was **"dr"**

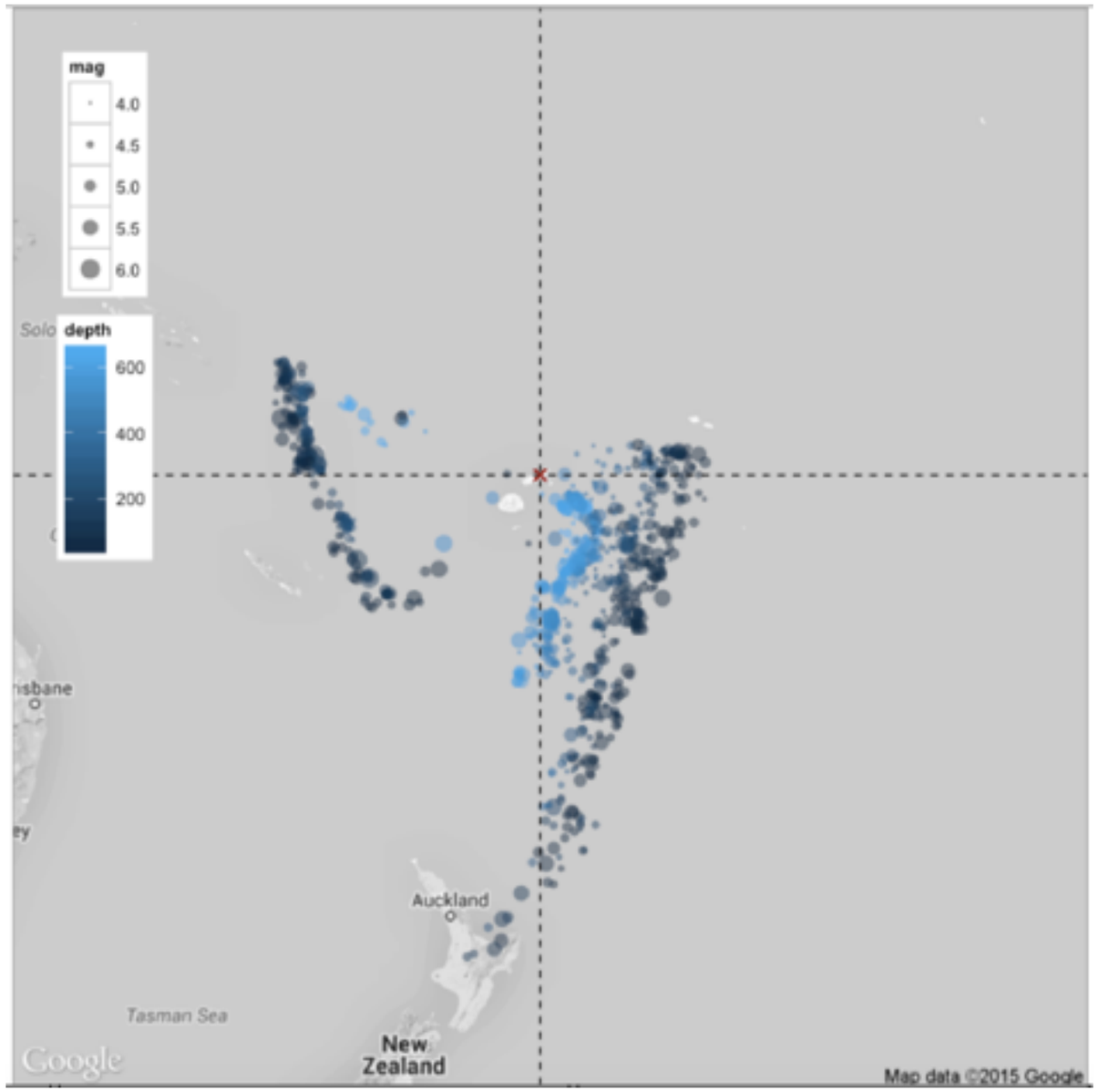
ggplot2 library

Generate complex plots simply!

part of **Data Science Drop-in**, by Houshmand Shirani-Mehr

Some of figures based on Josef Fruehwald's tutorial on ggplot2

Generate elegant plots

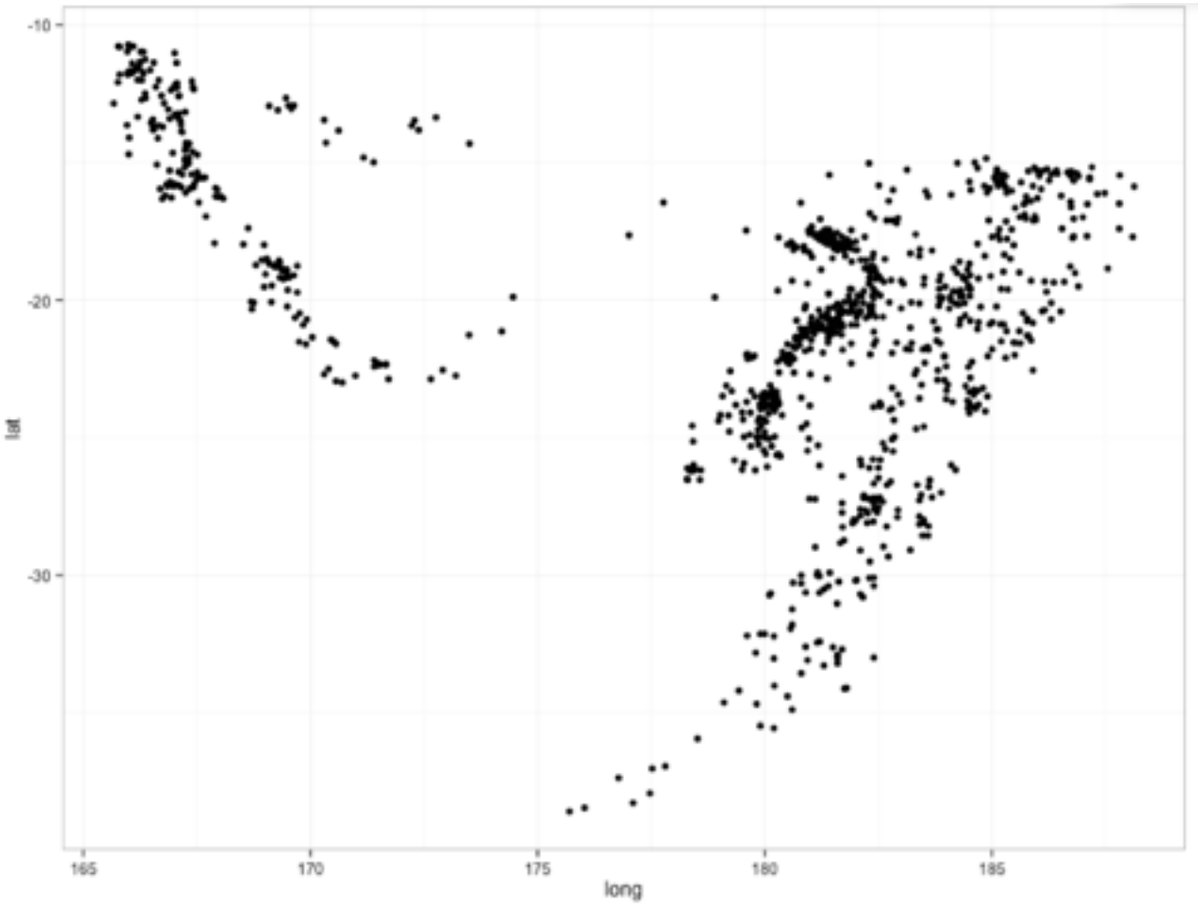


Locations of Earthquakes off Fiji

First try

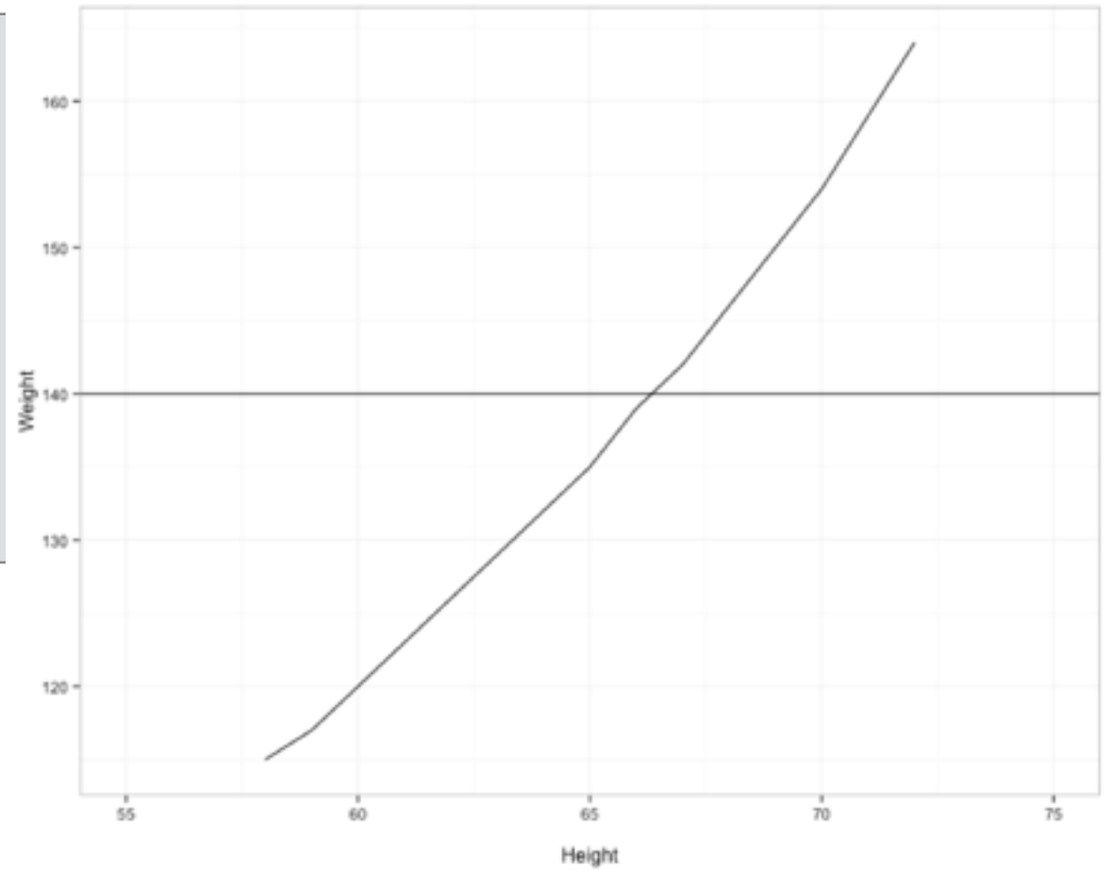
```
ggplot(quakes,aes(x=long,y=lat)) + geom_point()
```
[R]

To generate the plot in the previous slide, we need to know how exactly ggplot() works!



Built by Layers

```R
ggplot(women, aes(x=height, y=weight) ) +
  geom_line() +
  geom_hline(yintercept = 140) +
  scale_x_continuous('Height', limits=c(55,75)) +
  scale_y_continuous('Weight')
```



Every component of a graph (underlaying data, coordinate system, labels, …) is a layer you can manipulate.

Main Layer Types

- Data Layer
- Geometries layer
- Axis scales and labels

```R
ggplot(women, aes(x=height, y=weight) ) +
  geom_point() +
  geom_hline(yintercept = 140) +
  scale_x_continuous('Height', limits=c(55,75)) +
  scale_y_continuous('Weight')
```

Data Layer

```
ggplot(women , aes(x=height, y=weight) )
```

Data Frame

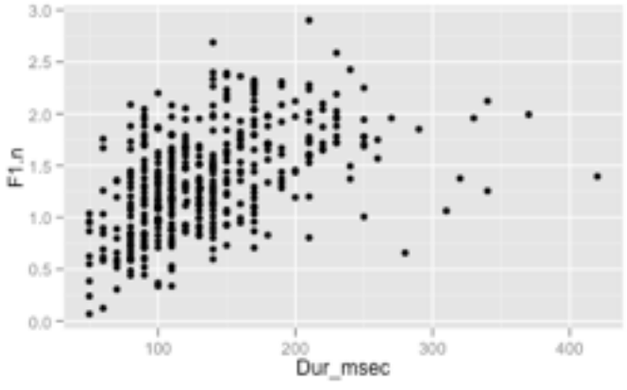
aesthetic: mapping of data frame variables to components of the plot

`aes(x=…, y=…, color=…,fill=…)`

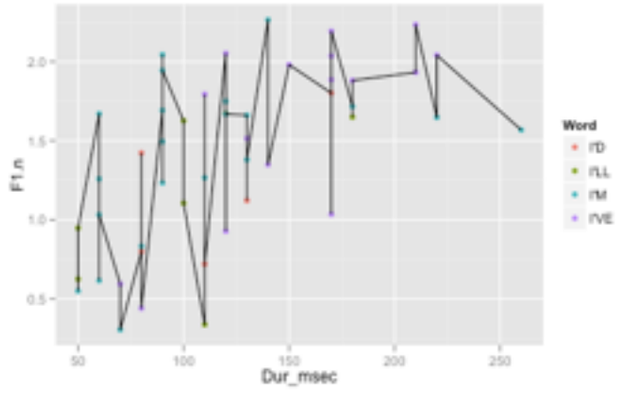
If you execute this command, no plot is generated.

Need to add geometries!

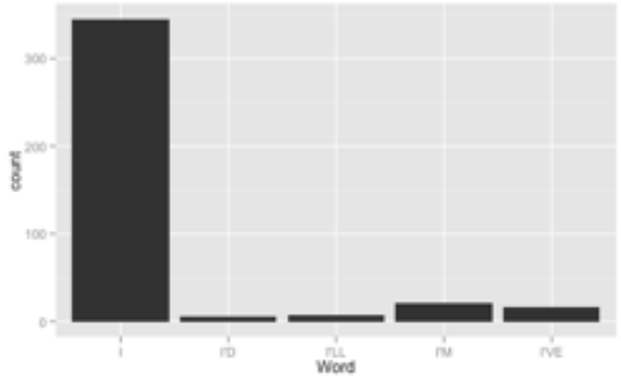
Geometries Layer



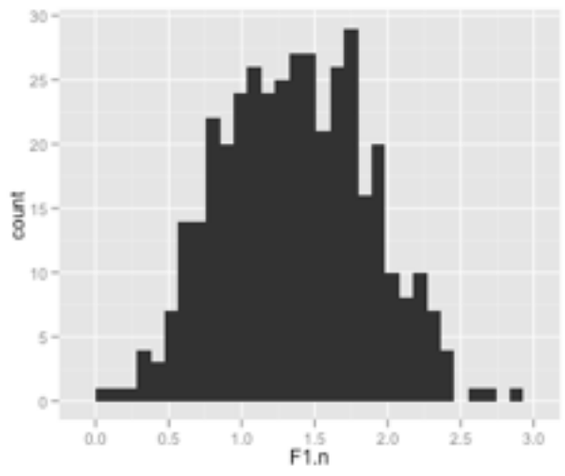
geom_point()



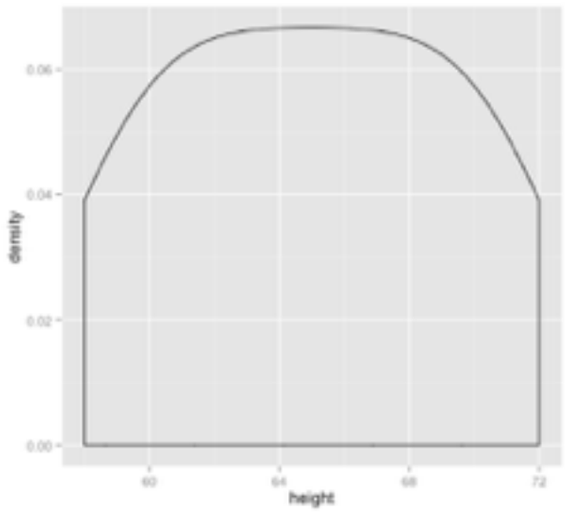
geom_line()



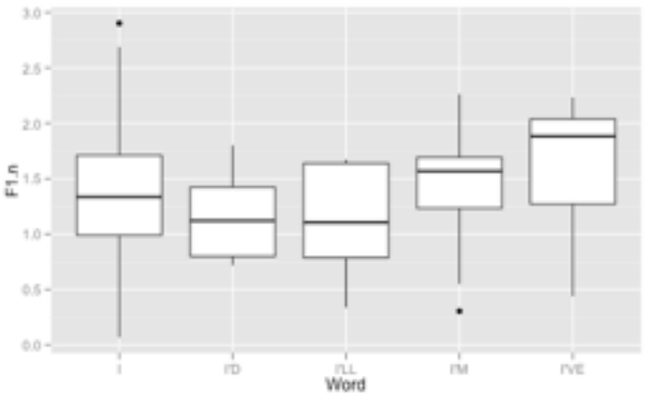
geom_bar()



geom_histogram()

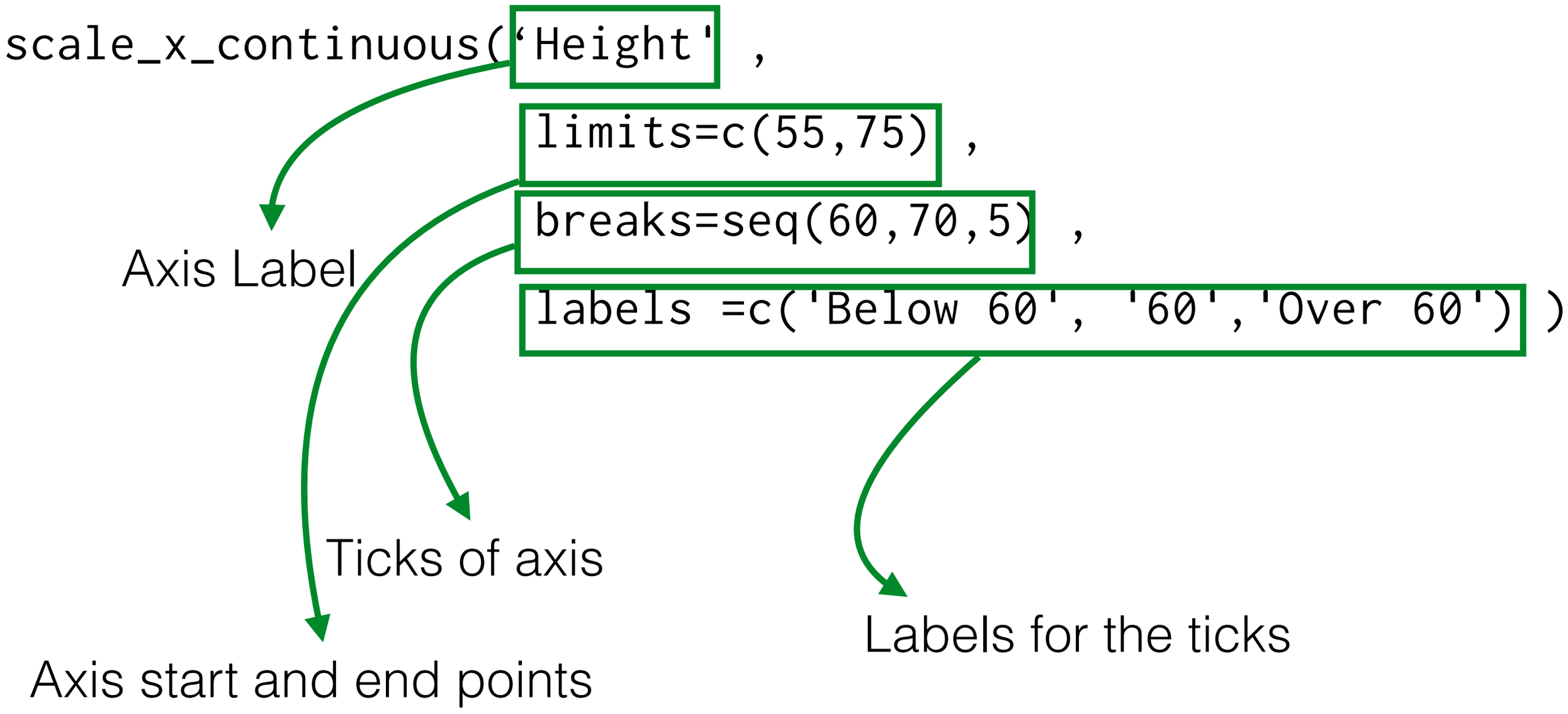


geom_density()



geom_boxplot()

Axis Manipulation



ggplot() in action

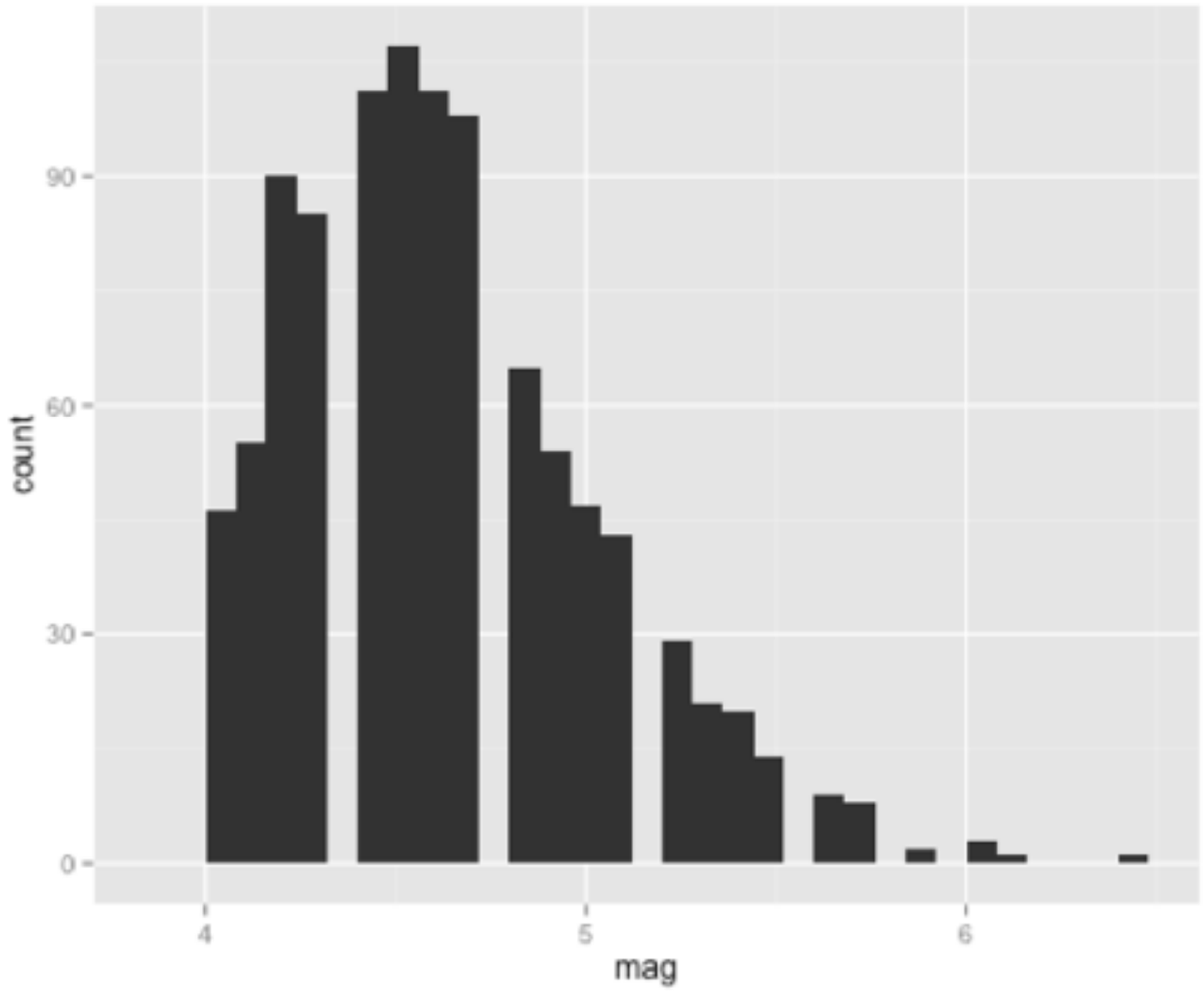
Today's **GOAL**

Draw the histogram for magnitudes in *quakes* dataset

First try

```
p <- ggplot(quakes, aes(x=mag) )
p <- p + geom_histogram()
p
```

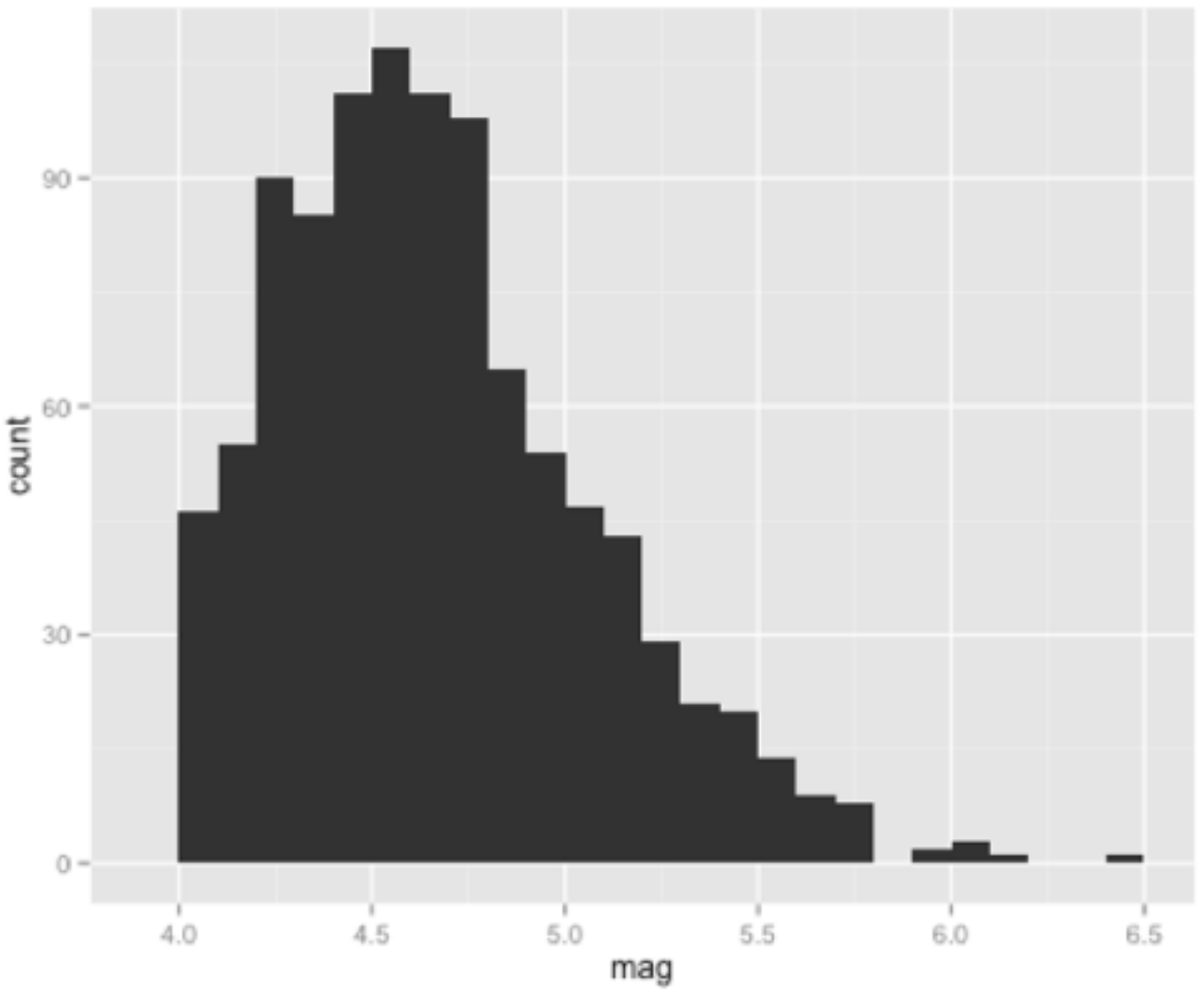
[R]



Edit bin size

```
p <- ggplot(quakes, aes(x=mag) )
p <- p + geom_histogram(binwidth=0.1)
p
```

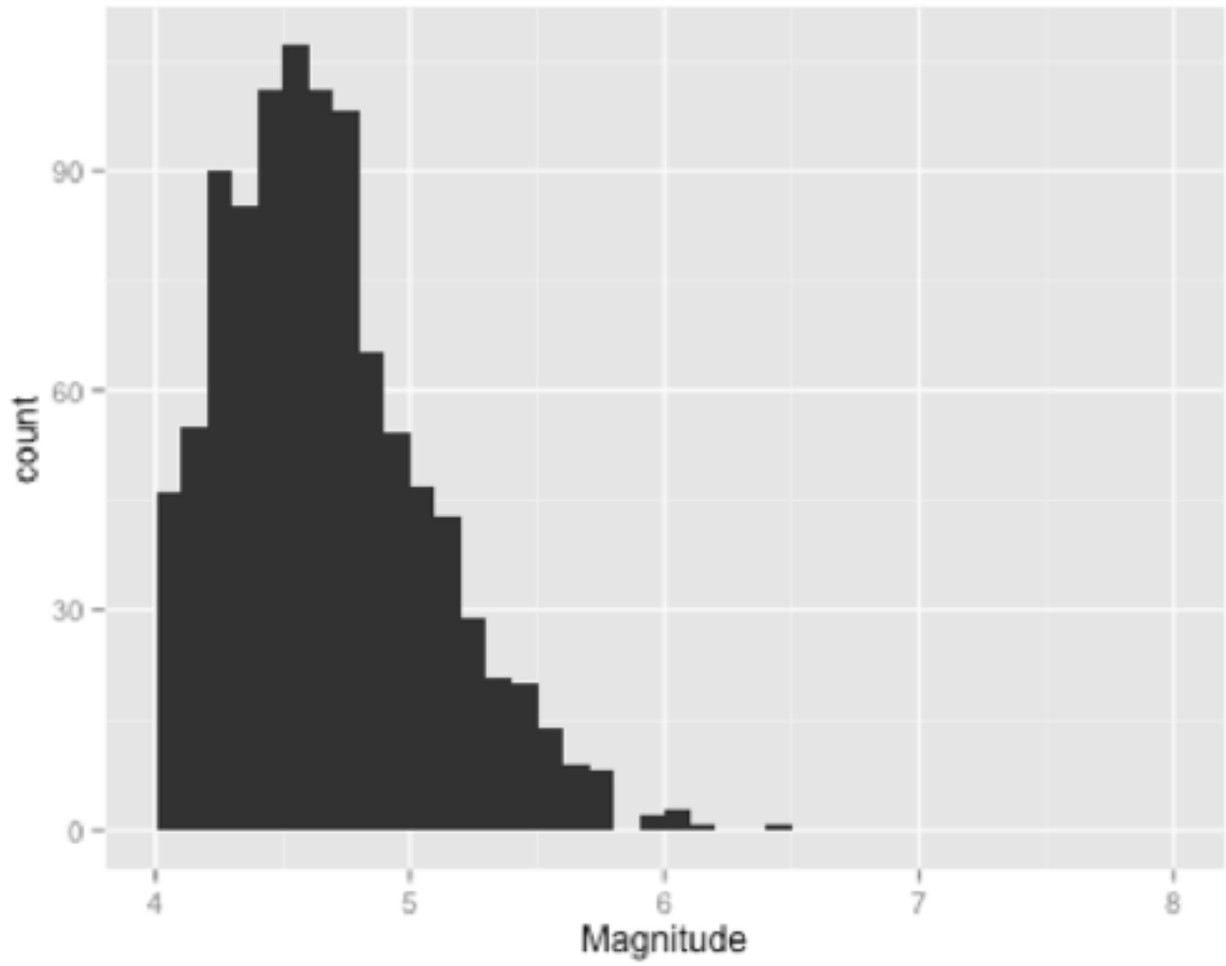
[R]



Edit axis

```
p <- ggplot(quakes, aes(x=mag) )
p <- p + geom_histogram(binwidth=0.1)
p <- p + scale_x_continuous("Magnitude",
limits=c(4,8), breaks=seq(4,8,1))
p
```

[R]



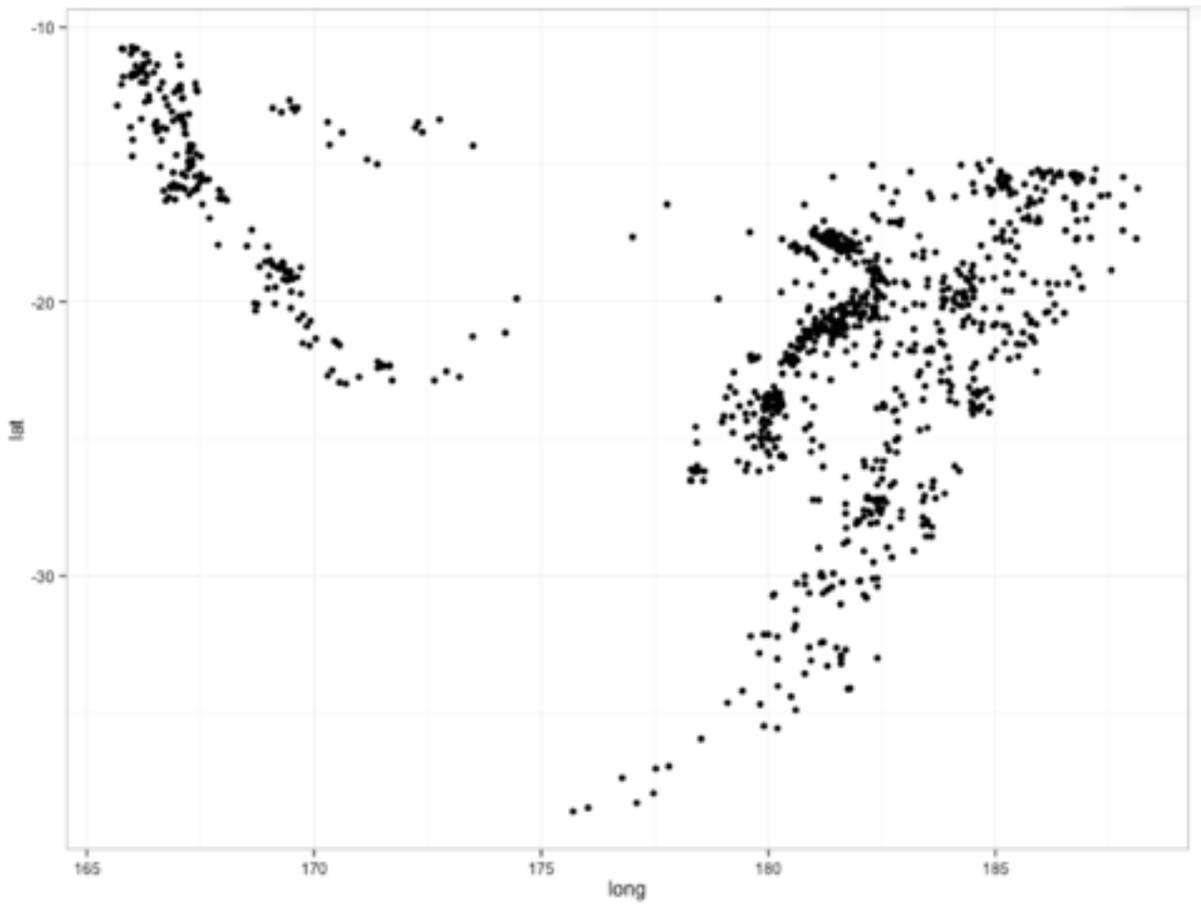
ggplot() in action

Today's **GOAL**

Draw the scatterplot for locations in *quakes* dataset

First try

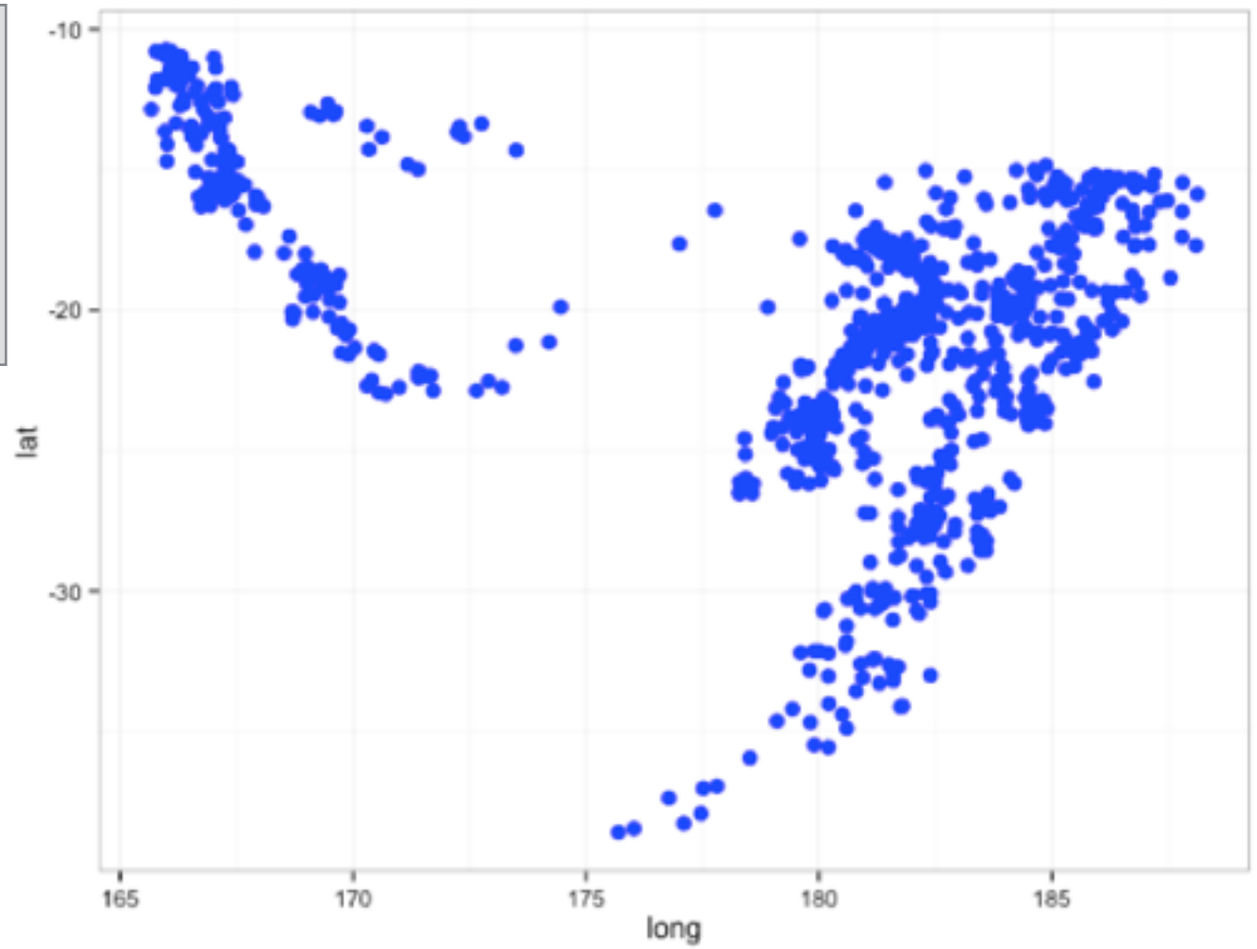
```R
p <- ggplot(quakes, aes(x=long, y=lat) )
p <- p + geom_point()
p
```



Edit points

```
p <- ggplot(quakes, aes(x=long, y=lat) )
p <- p + geom_point(size = 3, color = "blue")
p
```

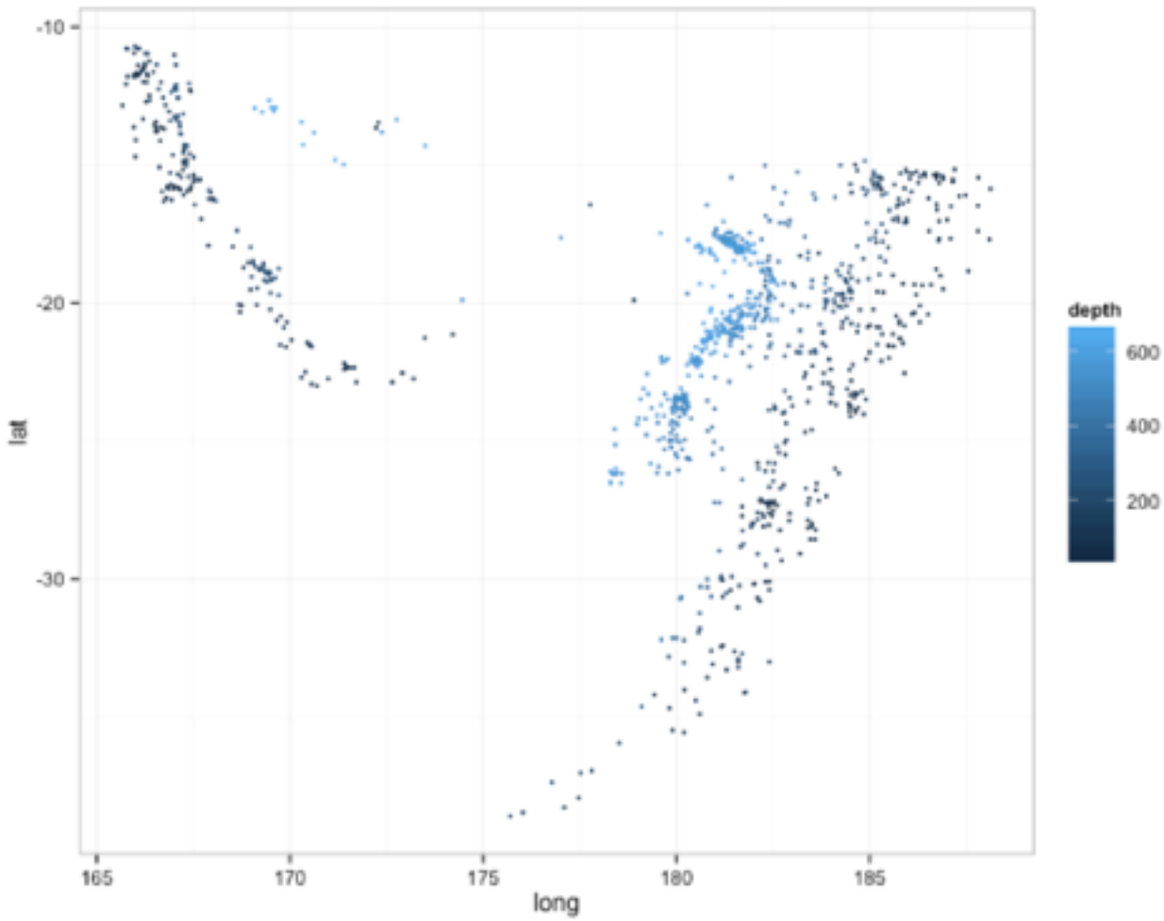
[R]



Use aes() with geometries

```
p <- ggplot(quakes, aes(x=long, y=lat) )
p <- p + geom_point(size = 1, aes(color = depth))
p
```

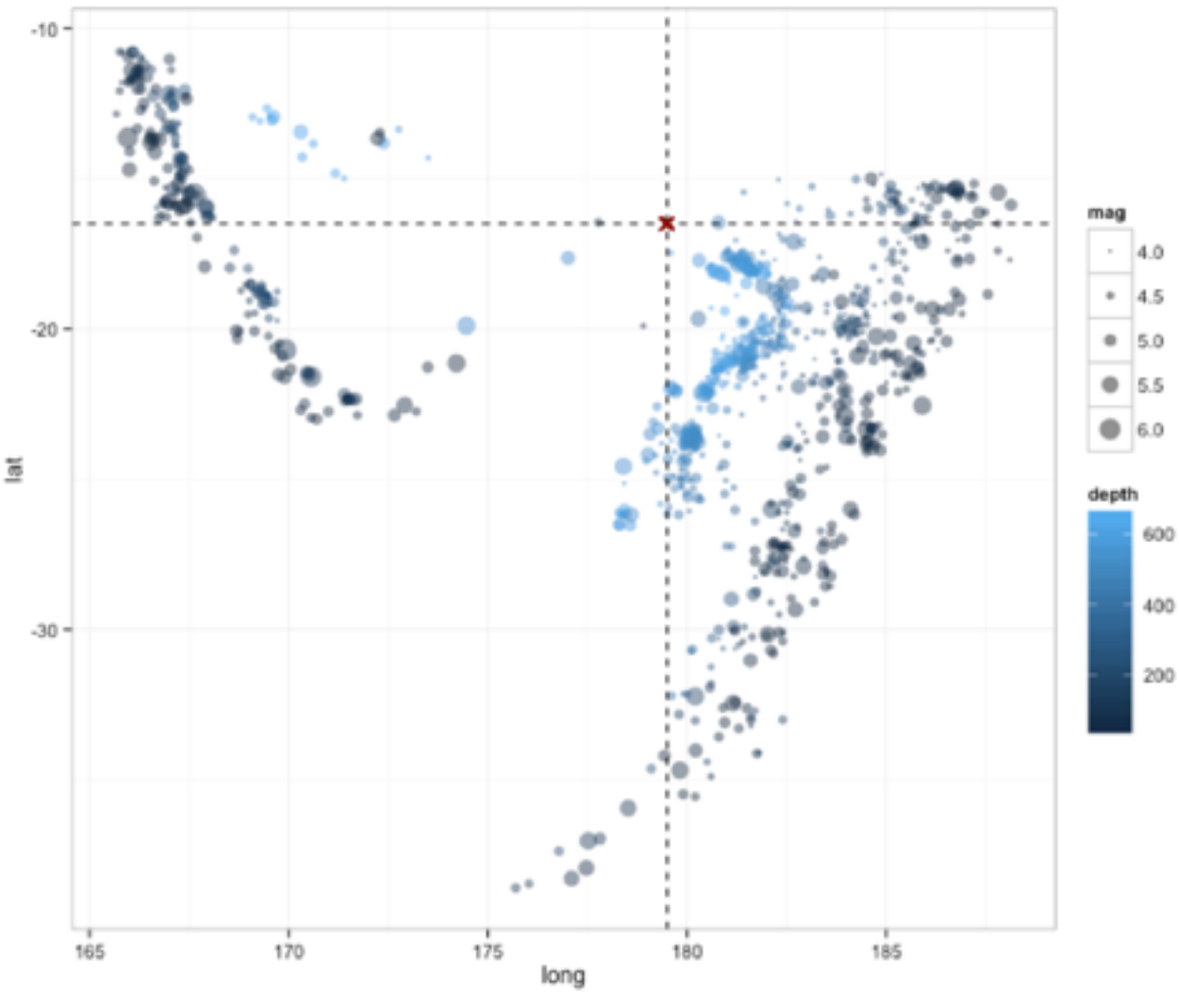
[R]



Add layers

```
p <- ggplot(quakes, aes(x=long, y=lat) )
p <- p + geom_point(aes(color = depth,
size=mag), alpha = 0.5)
p <- p + geom_hline(yintercept=-16.5, linetype=2)
p <- p + geom_vline(xintercept=179.5, linetype=2)
p <- p + geom_point(x=179.5, y=-16.5, shape=4,
color="darkred", size=3)
p
```

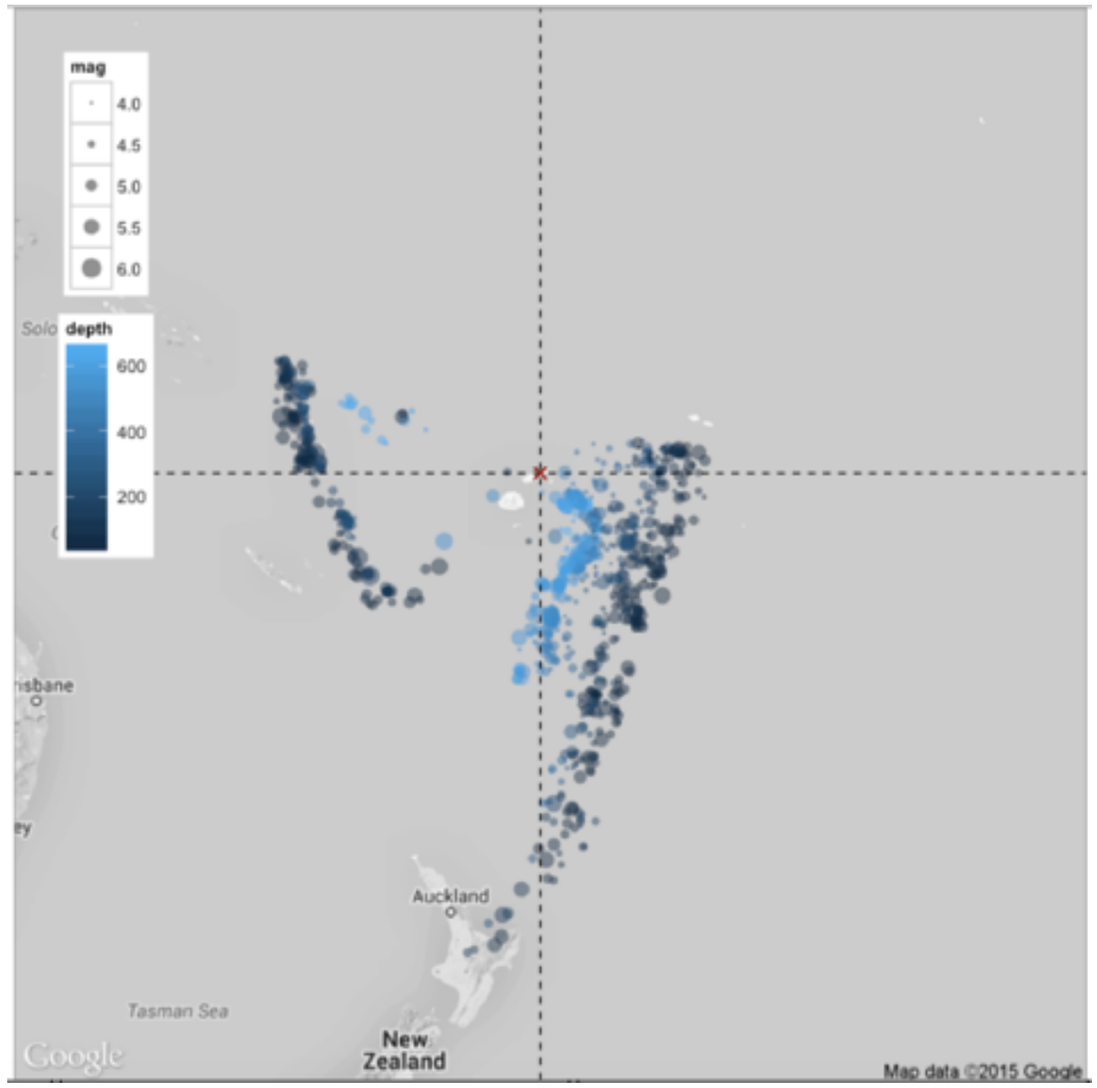
[R]



Add map layer*

```
p <- qmap(location=c(lon=180,lat=-20), zoom = 4,
color = "bw", legend = "topleft")
p <- p + geom_point(data=quakes, aes(x=long,
y=lat, color=depth, size=mag), alpha=0.5)
p <- p + geom_hline(yintercept=-16.5,linetype=2)
p <- p +  geom_vline(xintercept=179.5,linetype=2)
p <- p + geom_point(x=179.5, y=-16.5, shape=4,
color="darkred", size=3)
p
```

[R]



Intro to `R` - `dplyr`

a **Data Science Drop-in** Tutorial

by **Jongbin Jung**
(jongbin@stanford.edu)

Dependencies

- Install **`dplyr`** package and sample data

```
install.packages(c("dplyr", "nycflights13"))
```

- Load them to your workspace

```
library("dplyr")
library("nycflights13")
```

The **`nycflights13 data.frame (flights)`** contains all 336,776 flights that departed from New York City in 2013. The data comes from the US Bureau of Transporation Statistics, and is documented in **`?nycflights13`**.

`nycflights13` data

- take a look at the `flights data.frame`

```
  year month day dep_time dep_delay arr_time arr_delay carrier tailnum flight origin dest air_time distance hour minute
1 2013     1   1      517         2      830        11      UA  N14228   1545    EWR  IAH      227     1400    5     17
2 2013     1   1      533         4      850        20      UA  N24211   1714    LGA  IAH      227     1416    5     33
3 2013     1   1      542         2      923        33      AA  N619AA   1141    JFK  MIA      160     1089    5     42
   ...   ... ...      ...       ...      ...       ...     ...     ...    ...    ...  ...      ...      ...  ...    ...
```

- what questions could you ask with this data?
  - how many flights were there each day?
  - what's the mean departure delay for flights every month / day
  - what else?

verb

- A verb in **R** is a function that takes a **data.frame** as its first argument, for example, try

- The key concept of **dplyr**:
  most of your data manipulation needs can be satisfied with 5 basic **verbs**
  (4 verbs, depending on how you categorize them)

5 basic `verb`s

verb	action
`filter()`	select a subset of **rows** by conditions
`select()`	select a subset of **columns** from the data
`mutate()`	create a **new column** (usually based on existing columns)
`arrange()`	reorder (sort) **rows**
`summarise()`	aggregate values and reduce to single value

selecting rows - `filter()`

- select a subset of **rows**
- multiple conditions can be used
- use & to specify an AND operation

```
filter(flights, tailnum == "N14228" & arr_delay > 10)
```

- use | to specify an OR operation

```
filter(flights, tailnum == "N14228" | tailnum == "N24211")
```

- mix AND/OR operations (default behavior is AND)

```
filter(flights, tailnum == "N14228" | tailnum == "N24211", arr_delay > 10)
```

selecting rows - `slice()`

- similarly, select a subset of **rows** by position using `slice()`
- for example, to select the first 10 rows

```
slice(flights, 1:10)
```

- or to select the last 10 rows

```
slice(flights, (n()-9):n())
```

- use `n()` inside a `dplyr` verb to use the *number of rows* in the data

selecting columns - `select()`

- select a subset of **columns**
- either specify the columns that you want to select

```
select(flights, c(carrier, tailnum))
```

- or specify the columns you don't want to select

```
select(flights, -c(year, month, day))
```

- also works without the `c()`

```
select(flights, carrier, tailnum)
select(flights, -year, -month, -day)
```

selecting columns - `select()`

- use helper functions such as `starts_with()`, `ends_with()`, `matches()` and `contains()`

```
select(flights, starts_with("dep"))
select(flights, matches("_"))
select(flights, contains("delay"))
```

- assign new column names with `select()`

```
select(flights, tail_num = tailnum)
```

- to keep the rest of the data, use `rename()`

```
rename(flights, tail_num = tailnum)
```

create columns - `mutate()`

- create new **columns**, usually as a function of old columns

```
mutate(flights, gain = arr_delay - dep_delay,
                speed = distance / air_time * 60)
```

- you can also refer to columns that you just created

```
mutate(flights, gain = arr_delay - dep_delay,
                gain_per_hour = gain / (air_time / 60))
```

create columns - `mutate()`

- if you just want to keep the new columns, use `transmute()` instead

```
transmute(flights, gain = arr_delay - dep_delay,
                   gain_per_hour = gain / (air_time / 60))
```

sorting rows - `arrange()`

- reorder (sort) the data by specified **rows**
- multiple conditions are arranged from left-to-right

```
arrange(flights, year, month, day)
```

- use `desc()` to arrange in descending order

```
arrange(flights, year, desc(month), day)
arrange(flights, year, month, desc(day))
arrange(flights, year, desc(month), desc(day))
```

summarise()

- aggregate/collapse data into single row

```
summarise(flights, delay = mean(dep_delay, na.rm = TRUE))
```

- more useful with grouped operations (see next)

group operations

- indicate a grouping variable with group_by()

```
flights_by_day <- group_by(flights, day)
```

- some verbs have specific behavior with groups

verb	group specific actions
`arrange()`	orders first by grouping variable
`slice()`	extract rows within each group
`summarise()`	aggregate values for each group, and reduce to single value

group `slice()`

- retrieve the first 2 rows of each day

```
slice(flights_by_day, 1:2)
```

group `summarise()`

- `summarise()` makes much more sense when used with grouped data
- retrieve (1) number of flights, (2) average distance, and (3) average arrival delay **<u>for each day</u>** (i.e., **<u>for flights grouped by days)</u>**

```
summarise(flights_by_day,
  count = n(),
  dist = mean(distance, na.rm = TRUE),
  delay = mean(arr_delay, na.rm = TRUE))
```

multiple (chained) operations

"find days when the mean arrival delay
OR departure delay was greater than 30"

group data by date `(year, month, day)`

aggregate each group by mean arrival/departure delay

filter aggregated result `(mean arr_delay > 30 | mean dep_delay > 30)`

- **`dplyr`** verbs won't affect your original data (this is generally a good/safe thing, but potentially makes it difficult to do multiple operations on a single **`data.frame`**)
- there are two (acceptable) ways of doing this, and one is probably better than the other

multiple (chained) operations

"find days when the mean arrival delay
OR departure delay was greater than 30"

group data by date `(year, month, day)`

aggregate each group by mean arrival/departure delay

filter aggregated result `(mean arr_delay > 30 | mean dep_delay > 30)`

```
flights_by_date <- group_by(flights, year, month, day)
summary_by_date <- summarise(flights_by_date,
  arr = mean(arr_delay, na.rm = TRUE),
  dep = mean(dep_delay, na.rm = TRUE))
big_delay_days <- filter(summary_by_date, arr > 30 | dep > 30)
```

this isn't too bad. but it's not very nice.

multiple (chained) operations

"find days when the mean arrival delay
OR departure delay was greater than 30"

a better way to do this with **`dplyr`**

the pipe operator

%>%

```
some_function() %>% another_function()
```

use the result from this...

... as the first argument for this

multiple (chained) operations

"find days when the mean arrival delay
OR departure delay was greater than 30"

group data by date `(year, month, day)`

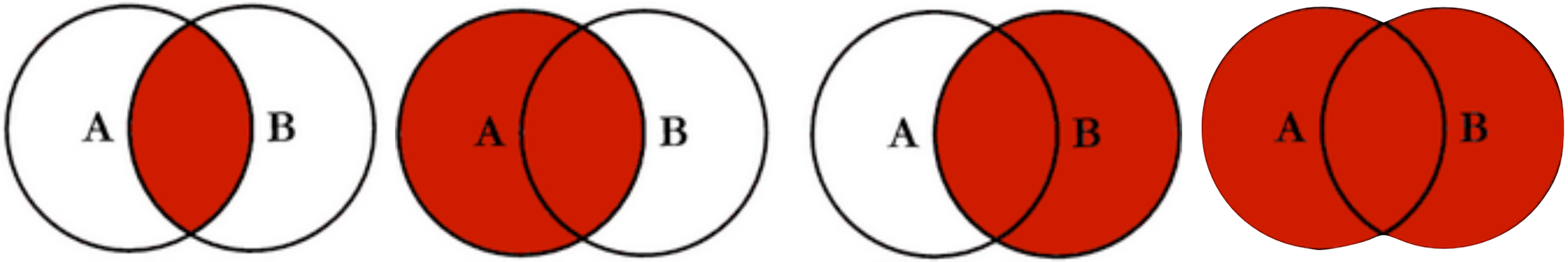
aggregate each group by mean arrival/departure delay

filter aggregated result `(mean arr_delay > 30 | mean dep_delay > 30)`

```
flights %>%
  group_by(year, month, day) %>%
  summarise(arr = mean(arr_delay, na.rm = TRUE),
            dep = mean(dep_delay, na.rm = TRUE)) %>%
  filter(arr > 30 | dep > 30)
```

this is easier to read. no need to save intermediate results.

joins (merge)

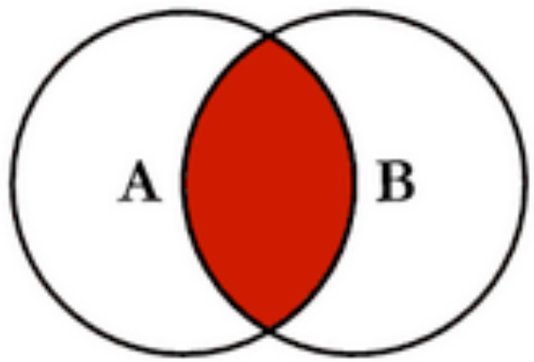


inner left right outer

- **merge(x, y, ...)**
  merge two data frames by common columns or row names,
  or do other versions of database *join* operations.

- Not really a **`dplyr`** function, but works well

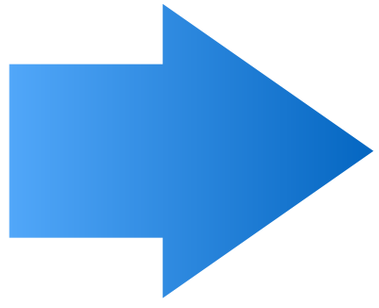
joins (inner)



- **merge(x, y,** all.x = FALSE, all.y = FALSE**)**
  default behavior is that only rows with data from both x and y are included in the output. this can also be specified with the arguments `all.x = FALSE` and `all.y = FALSE`

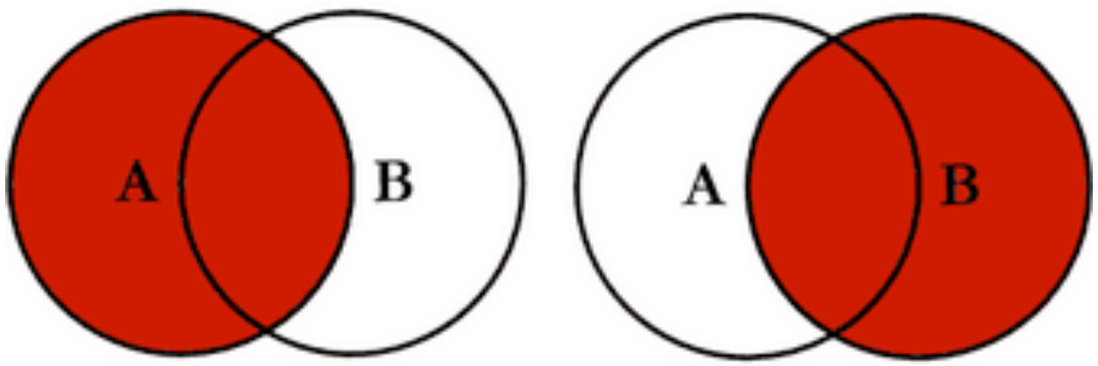
ID	sex
1	M
2	F
3	M
4	F

ID	age
2	20
3	18
6	23



ID	sex	age
2	F	20
3	M	18

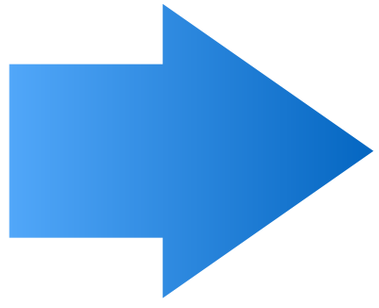
joins (left / right)



- **merge(x, y, all.x = TRUE,** all.y = FALSE**)**
  when `all.x = TRUE`, extra rows will be added to the output, one for each row in x that has no matching row in y. These rows will have NAs in those columns that are usually filled with values from y (right joins are achieved with `all.y = TRUE`)

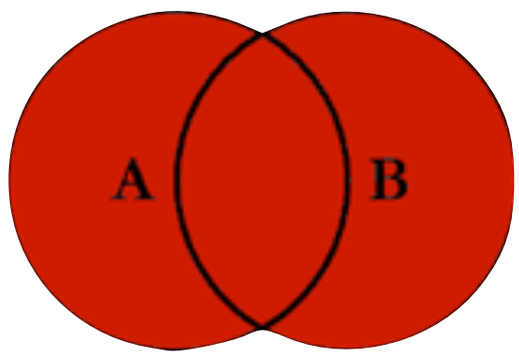
ID	sex
1	M
2	F
3	M
4	F

ID	age
2	20
3	18
6	23



ID	sex	age
1	M	*NA*
2	F	20
3	M	18
4	F	*NA*

joins (outer)



- **merge(x, y, all = TRUE)**
  `all = TRUE` is shorthand for `all.x = TRUE` and `all.y =TRUE`. If `all = TRUE` , then all the rows in both x and y are included in the output.

ID	sex
1	M
2	F
3	M
4	F

ID	age
2	20
3	18
6	23

ID	sex	age
1	M	*NA*
2	F	20
3	M	18
4	F	*NA*
6	*NA*	23

Today's **CHALLENGE**s

Find the average speed (`distance / air_time * 60`) by each carrier (ignore any `NA`s), and sort the data in descending order of average speed

Find the number of flights longer than 10 hours by each carrier in April