

Introduction to R

November 22, 2018

R and RStudio

The image shows the RStudio interface with four key components highlighted by numbered callouts:

- 1) Script:** Points to the editor window containing R code for loading data, creating a scatterplot, and fitting a linear regression model.
- 2) Console:** Points to the terminal window showing the execution output of the R script, including variable types and the fitted regression line.
- 3) Workspace:** Points to the environment pane showing the objects loaded into memory, including the 'google' data frame and the 'reg1' regression object.
- 4) Results/Plots:** Points to the plot window displaying a scatterplot titled 'Interest in Data Visualization Searches by Percent of Population with College Degrees' with a fitted linear regression line.

Script Code:

```
1 # up and running with R
2 # ex01.R
3 # creating scatterplots
4 # how 2 quantitative variable associate with each other
5 # load data file about google searches by state
6 # load data file about google searches by state
7 google = read.csv("google.correlate.csv", header = T)
8 # create google
9 str(google)
10 # there is an association between the percentage of people
11 # with college degrees (%) and interest in
12 # data visualization?
13 # data visualization
14 # add title, labels, change circles to points
15 plot(google$degree, google$data_viz)
16 main = "Interest in Data Visualization Searches/ny Percent of Population with C"
17 xlab = "Population with college degrees",
18 ylab = "Searches for 'data visualization'",
19 pch = 20,
20 col = "grey")
21 # Add Fit Lines
22 # linear regression line (y ~ x)
23 reg1 = lm(data ~ degree, data = google)
24 #
```

Console Output:

```
$facebook : num 1.93 -0.52 -1.18 2.21 -1.28 -1.33 -0.14 -0.34 -2.2 0.1 ...
$haa_rba : factor w/ 2 levels "no","yes": 1 1 2 1 2 1 2 2 ...
$degree : num 22.3 25.3 28.8 31.7 35.5 36.5 26.9 35.7 26 ...
$statu_ad : factor w/ 2 levels "no","yes": 1 1 2 2 1 1 2 2 ...
$region : factor w/ 4 levels "Midwest","Northeast"...: 3 4 4 3 4 4 2 3 3 ...

# Is there an association between the percentage of people
# in a state with college degrees (%) and interest in
# data visualization?
# data visualization
# add title, labels, change circles to points
# plot(google$degree, google$data_viz)
# main = "Interest in Data Visualization Searches/ny Percent of Population with
# college degrees"
# ylab = "Searches for 'data visualization'",
# # linear regression line (y ~ x)
# # lowest smoother line (x, y)
# abline(lm(google$data_viz ~ google$degree), col="red")
# # lowest smoother line (x, y)
# # now how be linear, so want to draw a line that's
# # closer to the shape of the data
# # note = order is different
# # lowest = moving average
# lines(lowess(google$degree, google$data_viz), col="blue")
#
```

Workspace:

```
google 31 obs. of 9 variables
Values
regmodel.1 1e[12]
reg1 1e[13]
```

Plot:

Interest in Data Visualization Searches
by Percent of Population with College Degrees

Searches for "Data Visualization"

Population with College Degrees

What R can do

What R can do

Everything.^{1,2}

1 Except think about your science

2 Occasionally in a non efficient way

What R can do

Everything.^{1,2}

1 Except think about your science

2 Occasionally in a non efficient way

What about RStudio?

- Makes your life easier
- Many handy tricks
 - ▶ Autocomplete suggestion
 - ▶ Ctrl-Enter to send command to R
 - ▶ `str()` and `View()` objects in Environment
 - ▶ Files, packages, help selectors
 - ▶ Version control. . .

1 About R-studio projects

2 The mean

3 Data-frames

4 For-loop

5 While-loop

6 If-else statements

7 Visualisation

8 T-test

About R-studio projects

Use them

- 1 About R-studio projects
- 2 The mean**
- 3 Data-frames
- 4 For-loop
- 5 While-loop
- 6 If-else statements
- 7 Visualisation
- 8 T-test

Calculating a mean: Arithmetic and assignment

```
(2 + 3 + 5 + 1) / 4
```

```
[1] 2.75
```

Calculating a mean: Arithmetic and assignment

```
(2 + 3 + 5 + 1) / 4
```

```
[1] 2.75
```

```
a <- 2
```

```
b <- 3
```

```
c <- 5
```

```
d <- 1
```

```
(a + b + c + d) / 4
```

```
[1] 2.75
```

Calculating a mean: Arithmetic and assignment

```
(2 + 3 + 5 + 1) / 4
```

```
[1] 2.75
```

```
a <- 2
```

```
b <- 3
```

```
c <- 5
```

```
d <- 1
```

```
(a + b + c + d) / 4
```

```
[1] 2.75
```

```
a <- 45
```

```
(a + b + c + d) / 4
```

```
[1] 13.5
```

Calculating a mean: using vectors

```
c(2,3,5,1) # c is for concatenate
```

```
[1] 2 3 5 1
```

Calculating a mean: using vectors

```
c(2,3,5,1) # c is for concatenate
```

```
[1] 2 3 5 1
```

```
mydata <- c(2,3,5,1) # save the vector
```

Calculating a mean: using vectors

```
c(2,3,5,1) # c is for concatenate
```

```
[1] 2 3 5 1
```

```
mydata <- c(2,3,5,1) # save the vector
```

```
mydata <- (2,3,5,1) # c is missing => error!
```

```
Error: <text>:1:15: unexpected ',',
```

```
1: mydata <- (2,
```

```
^
```

Why bother with vectors?

Substitution:

```
mydata[2] <- 4  
mydata
```

```
[1] 2 4 5 1
```

Vectorized operations:

```
mydata*5 + 2
```

```
[1] 12 22 27 7
```

Exercise

`-5:20`

```
[1] -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15  
[24] 18 19 20
```


Calculating a mean: using functions

How to use a function?

```
?mean
```

Or use **tab**

Calculating a mean: using functions

How to use a function?

```
?mean
```

Or use **tab**

```
mean(c(2,4,5,1))
```

```
[1] 3
```

```
mean(mydata)
```

```
[1] 3
```

```
mean(x = mydata)
```

```
[1] 3
```

- 1 About R-studio projects
- 2 The mean
- 3 Data-frames**
- 4 For-loop
- 5 While-loop
- 6 If-else statements
- 7 Visualisation
- 8 T-test

Loading data

```
trees <- read.csv("trees.csv")
```

Loading data

```
trees <- read.csv("trees.csv")
```

```
str(trees)
```

```
'data.frame': 31 obs. of 3 variables:
```

```
$ Girth : num 8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
```

```
$ Height: int 70 65 63 72 81 83 66 75 80 75 ...
```

```
$ Volume: num 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

Try also `summary`, `class`, `head`, `tail`

Access

Bracket-syntax

- Row: `dataframe[row,]`
- Column: `dataframe[, column]`
- Element: `dataframe[row, column]`

Access

Bracket-syntax

- Row: `dataframe[row,]`
- Column: `dataframe[, column]`
- Element: `dataframe[row, column]`

```
trees[,1]  
trees[1:8,]  
trees[c(2,1,2), 3]  
trees[, "Height"]
```

Dollar-syntax

- Column `dataframe$column_name`
- Element `dataframe$column_name[row]`

```
trees$Height
```

Time to think a tiny bit!

Calculate the mean for all three variables in trees, excluding the last (31st) record.

Solution for one column

Calculate the mean for all three variables in trees, excluding the last (31st) record.

```
mean(trees$Girth[1:30])  
mean(trees[1:30, "Girth"])  
mean(trees$Girth[-31])  
mean(trees[-31, "Girth"])
```

How to get the row means?

```
mean(trees[1,])  
mean(trees[2,])  
mean(trees[...])
```

How to get the row means?

```
mean(trees[1,])  
mean(trees[2,])  
mean(trees[...])
```



- 1 About R-studio projects
- 2 The mean
- 3 Data-frames
- 4 For-loop**
- 5 While-loop
- 6 If-else statements
- 7 Visualisation
- 8 T-test

How to get the row means? For-loops

```
for (i in 1:N)
{
  something as a function of i
}#end of the loop
```

How to get the row means? For-loops

```
for (i in 1:N)
{
  something as a function of i
}#end of the loop
```

```
for (i in 1:31)
{
  print(i)
}
```

How to get the row means? For-loops

Start by buiding the code for 1 iteration (1 "i" value, e.g., 22):

```
mean(as.numeric(trees[22,]))
```

How to get the row means? For-loops

Start by buiding the code for 1 iteration (1 "i" value, e.g., 22):

```
mean(as.numeric(trees[22,]))
```

We will want to store the result somewhere:

```
ResultMean <- vector() # we will store the results there  
ResultMean[22] <- mean(as.numeric(trees[22,]))
```


How to get the row means? For-loops

Start by buiding the code for 1 iteration (1 "i" value, e.g., 22):

```
mean(as.numeric(trees[22,]))
```

We will want to store the result somewhere:

```
ResultMean <- vector() # we will store the results there  
ResultMean[22] <- mean(as.numeric(trees[22,]))
```

Now change 22 to "i" and write a loop around:

```
ResultMean <- vector() # we will store the results there  
for (i in 1:31)  
{  
  ResultMean[i] <- mean(as.numeric(trees[i,]))  
}
```

For-loops: your turn!

Load rock data

```
rock <- read.csv("rock.csv")
```

Use a for loop to obtain column averages

Solution

Load rock data.

```
rock <- read.csv("rock.csv")
```

Use a for loop to obtain column averages

```
storage <- vector(length = ncol(rock))  
for (i in 1:ncol(rock))  
{  
  storage[i] <- mean(rock[,i])  
}
```

More concise alternative: apply functions

```
apply(X = dataframe, MARGIN = 1 (row) or 2 (col), FUN = function)
```

More concise alternative: apply functions

```
apply(X = dataframe, MARGIN = 1 (row) or 2 (col), FUN = function)
```

```
apply(X = rock, MARGIN = 1, FUN = mean) #by row (not meaningful)  
apply(X = rock, MARGIN = 2, FUN = mean) #by column
```

Even better (worse)...

```
colMeans(rock)  
rowMeans(rock)
```

Even better (worse)...

```
colMeans(rock)
rowMeans(rock)
```

Trade-off concision / flexibility

- colMeans shortest, but does only means
- apply very flexible, but does only array/matrix/data-frame
- for-loop looks complex, but infinitely flexible
- (NB: your computer does a for-loop whether you see it or not)

The last value

Sometimes a function takes very long to run

```
rowMeans(rock[rep(1:nrow(rock),100000),])
```


The last value

Sometimes a function takes very long to run

```
rowMeans(rock[rep(1:nrow(rock),100000),])
```

What if you forgot to save the output to an object??

The last value

Sometimes a function takes very long to run

```
rowMeans(rock[rep(1:nrow(rock),100000),])
```

What if you forgot to save the output to an object??

```
ourlasthope <- .Last.value
```

- 1 About R-studio projects
- 2 The mean
- 3 Data-frames
- 4 For-loop
- 5 While-loop**
- 6 If-else statements
- 7 Visualisation
- 8 T-test

While-loop: idea

Less common than for-loops

Stop the loop after a condition is met

```
while(condition TRUE)
{
  something
}
```

What is the smallest reproductive rate necessary to obtain a growing population?

```
library(popbio)
mat <- matrix(c(0,0.8,1,0), nrow = 2)
lambda(mat)

[1] 0.8944272

cond <- 1
while(lambda(mat) < 1 )
{
  mat[1,2] <- mat[1,2]+0.001
}
mat[1,2]

[1] 1.251
```

But think twice before running a while loop...

What happens if you run:

```
x <- 1
while(x>0)
{
  x <- x + 1
}
```

Looking for a rare event

The function `sample()` takes 5 number between 1 and 6 (like 5 dice!):

```
x <- sample(x = 1:6, size = 5, replace = TRUE)
```

Are all die equal?

```
all(x == x[1])
```

```
[1] FALSE
```

Are they ever going to be equal? Write a while loop to find a case

- 1 About R-studio projects
- 2 The mean
- 3 Data-frames
- 4 For-loop
- 5 While-loop
- 6 If-else statements**
- 7 Visualisation
- 8 T-test

If-else statements

```
if(condition)
{
  do something
}
```

```
if(condition)
{
  do something
}else{
  do something else
}
```

If-else statements

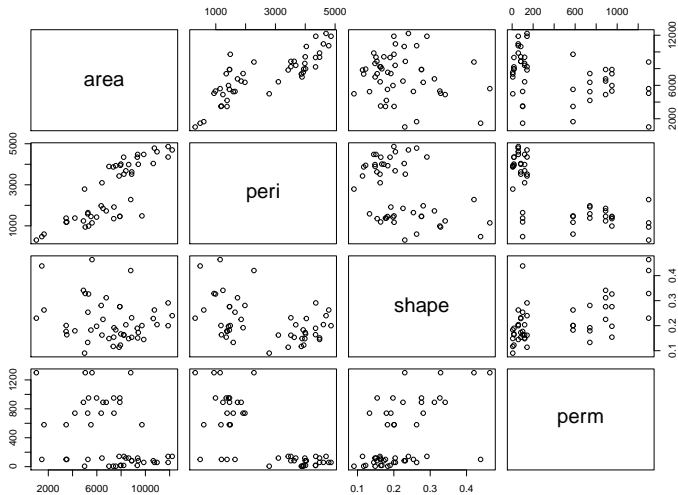
For instance:

```
for (i in 1:10)
{
  if(i < 6)
  {
    print("tofu")
  }else{
    print("bacon")
  }
}
```

- 1 About R-studio projects
- 2 The mean
- 3 Data-frames
- 4 For-loop
- 5 While-loop
- 6 If-else statements
- 7 Visualisation**
- 8 T-test

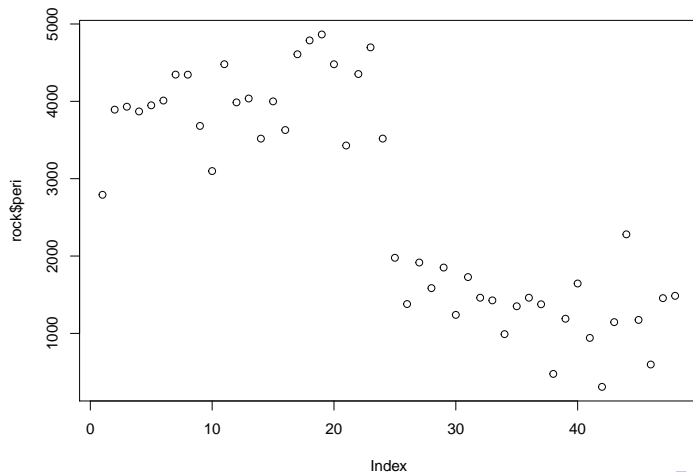
plot function

```
plot(rock)
```



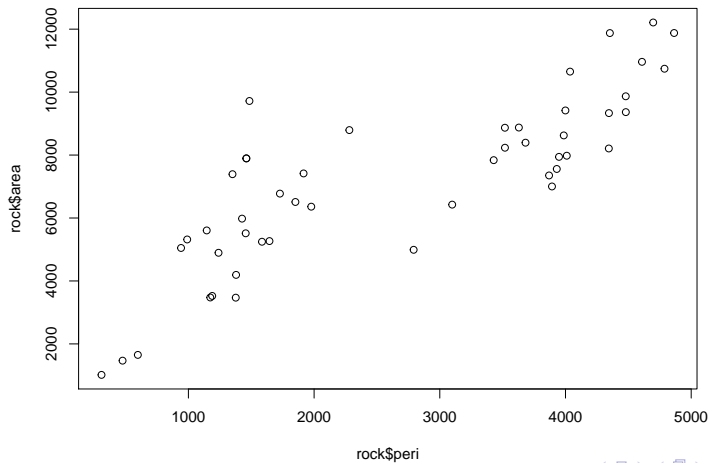
plot function

```
plot(rock$peri)
```



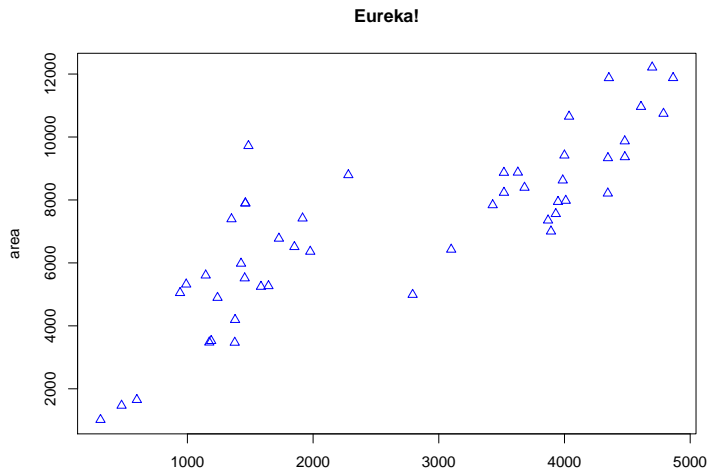
plot function

```
plot(x = rock$peri, y = rock$area)
```



plot function

```
plot(x = rock$peri, y = rock$area, main = "Eureka!",  
     xlab = "Perimeter", ylab = "area", col="blue", pch=2)
```

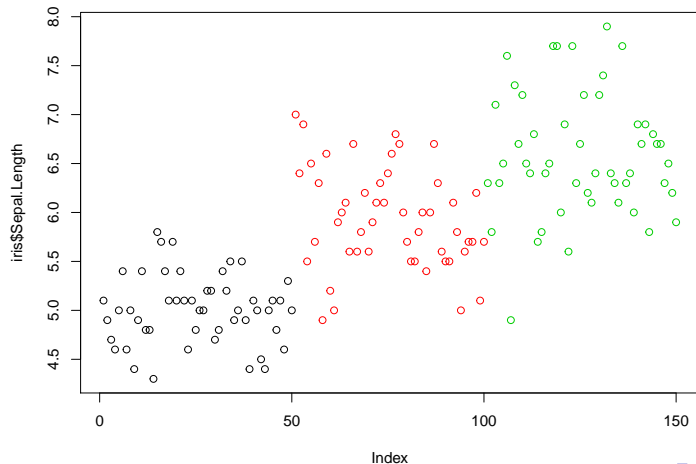


plot function: back to the mean

```
data("iris")
```

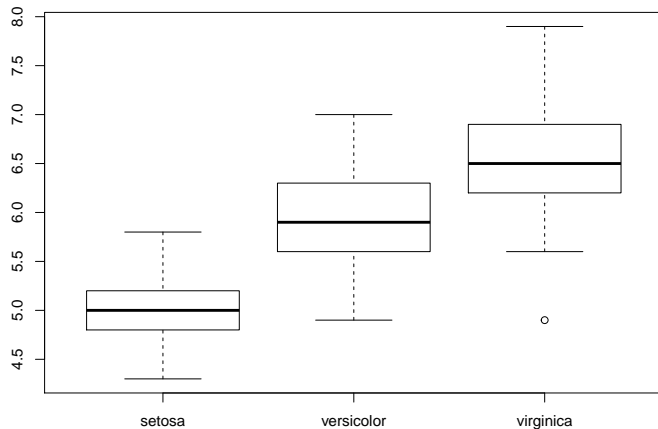

plot function: back to the mean

```
plot(iris$Sepal.Length, col=iris$Species)
```



boxplots

```
boxplot(iris$Sepal.Length ~ iris$Species)
```



- 1 About R-studio projects
- 2 The mean
- 3 Data-frames
- 4 For-loop
- 5 While-loop
- 6 If-else statements
- 7 Visualisation
- 8 T-test**

Student's T.test introduction

```
?t.test
```

Student's T.test introduction

```
?t.test
```

```
t.test(1:10, y = c(7:20))
```

Welch Two Sample t-test

data: 1:10 and c(7:20)

t = -5.4349, df = 21.982, p-value = 1.855e-05

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

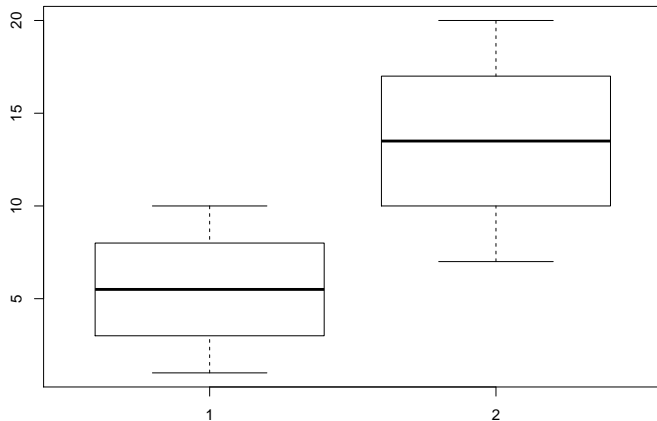
-11.052802 -4.947198

sample estimates:

mean of x	mean of y
5.5	13.5

T.test introduction

```
boxplot(c(1:10, 7:20) ~ c(rep(1,10), rep(2, 14)))
```



Are irises different?

Use t-tests to compare species in the iris dataset



Are irises different? Solution

Use t-tests to compare species in the iris dataset

Sorry, I was mean and forgot to tell about subsetting, which you needed here.

Subset to the species *setosa*:

```
iris[iris$Species == "setosa", ]
```

One t-test for sepal length between *setosa* and *versicolor*:

```
t.test(x = iris$Sepal.Length[iris$Species == "setosa"],  
       y = iris$Sepal.Length[iris$Species == "versicolor"])
```