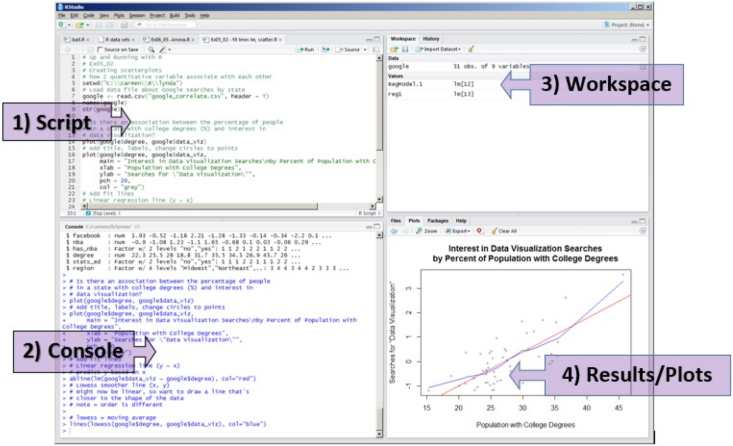
Introduction to R

February 9, 2018

R and RStudio



```
# Up and Running with R
# Ex05_02
# Creating scatterplots
# how 2 quantitative variable associate with each other
setwd("C:\\carmen\\R\\lynda")
# Load data file about Google searches by state
google <- read.csv("google_correlate.csv", header = T)
names(google)
str(google)

# Is there an association between the percentage of people
# in a state with college degrees (%) and interest in
# data visualization?
plot(google$degree, google$data_viz)
# Add title, labels, change circles to points
plot(google$degree, google$data_viz,
     main = "Interest in Data Visualization Searches\nby Percent of Population with C
     xlab = "Population with College Degrees",
     ylab = "Searches for \"Data Visualization\"",
     pch = 20,
     col = "grey")
# Add fit lines
# Linear regression line (y ~ x)
```

```
 $ facebook  : num  1.93 -0.52 -1.18 2.21 -1.28 -1.33 -0.14 -0.34 -2.2 0.1 ...
 $ nba       : num  -0.9 -1.08 1.23 -1.1 1.65 -0.68 0.1 0.03 -0.06 0.29 ...
 $ has_nba   : Factor w/ 2 levels "no","yes": 1 1 2 1 2 2 1 1 2 2 ...
 $ degree    : num  22.3 25.5 28 18.8 31.7 35.5 34.5 26.9 45.7 26 ...
 $ stats_ed  : Factor w/ 2 levels "no","yes": 1 1 1 2 2 1 1 1 2 2 ...
 $ region    : Factor w/ 4 levels "Midwest","Northeast",..: 3 4 4 3 4 4 2 3 3 3 ...
>
> # Is there an association between the percentage of people
> # in a state with college degrees (%) and interest in
> # data visualization?
> plot(google$degree, google$data_viz)
> # Add title, labels, change circles to points
> plot(google$degree, google$data_viz,
+      main = "Interest in Data Visualization Searches\nby Percent of Population with
College Degrees",
+      xlab = "Population with College Degrees",
+      ylab = "Searches for \"Data Visualization\"",
+      pch = 20,
+      col = "grey")
> # Add fit lines
> # Linear regression line (y ~ x)
> # predict y based on x
> abline(lm(google$data_viz ~ google$degree), col="red")
> # Lowess smoother line (x, y)
> # might now be linear, so want to draw a line that's
> # closer to the shape of the data
> # note = order is different
>
> # lowess = moving average
> lines(lowess(google$degree, google$data_viz), col="blue")
> 
```

```
google       51 obs. of 9 variables
RegModel.1   lm[12]
reg1         lm[13]
```

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. So far we have seen:
  - Autocomplete suggestion (all alone or press tab)
  - Ctrl-Enter to send command to R
  - str() and View() objects in Environment

1. The mean
2. Data-frames
3. Visualisation
4. 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

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

[1] 2 3 5 1
```

Calculating a mean: using vectors

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

[1] 2 3 5 1
```

```r
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>:1:14:  unexpected ','
1:  mydata <- (2,
                ^
```

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>:1:14: unexpected ','
1: mydata <- (2,
                ^
```

Why bother with vectors?

```
  mydata[2] <- 4
  mydata

[1] 2 4 5 1
```

Calculating a mean: using functions

How to use a function?

```
?mean
```

Calculating a mean: using functions

How to use a function?

```
?mean
```

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

[1] 3

mean(mydata)

[1] 3

mean(x = mydata)

[1] 3
```

1. The mean
2. Data-frames
3. Visualisation
4. T-test

Loading data

```
data("trees")
```

Loading data

```
  data("trees")
```

```
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: num  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]

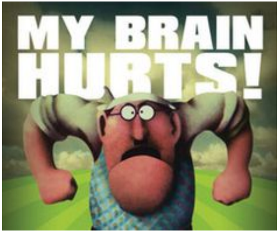
```
trees[,1]
trees[1:8,]
trees[c(2,1,2), 3]
trees[, "Height"]
```

Dollar-syntax

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

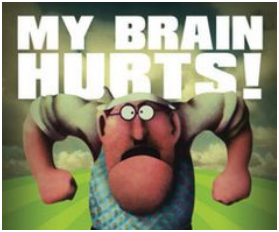
```
  trees$Height
```

Finally 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[...,])
```

:

How to get the row means? For-loops

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

How to get the row means? For-loops

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

```
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.

```
data("rock")
```

Use a for loop to obtain column averages

Solution

Load rock data.

```r
data("rock")
```

Use a for loop to obtain column averages

```r
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)
```

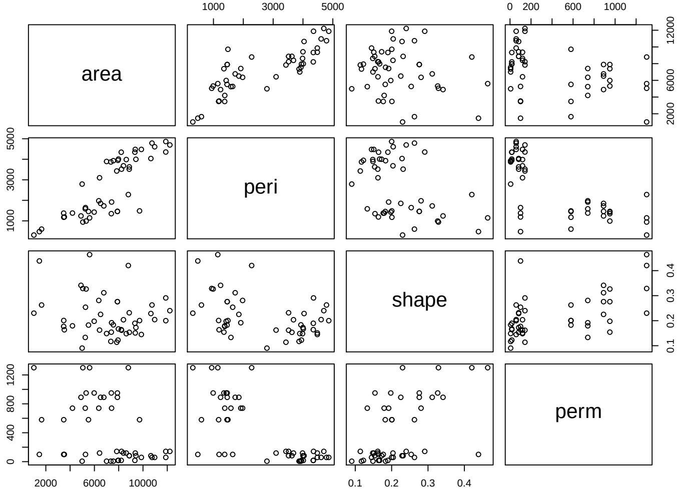
Trade-off concision / flexibility

- 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)

1. The mean
2. Data-frames
3. **Visualisation**
4. T-test

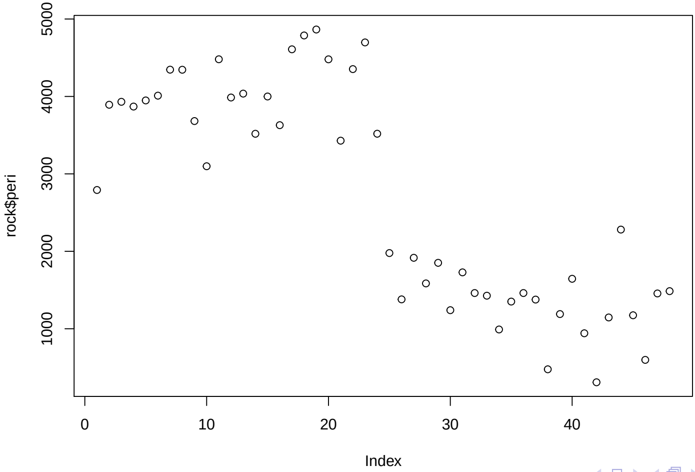
plot function

```
plot(rock)
```



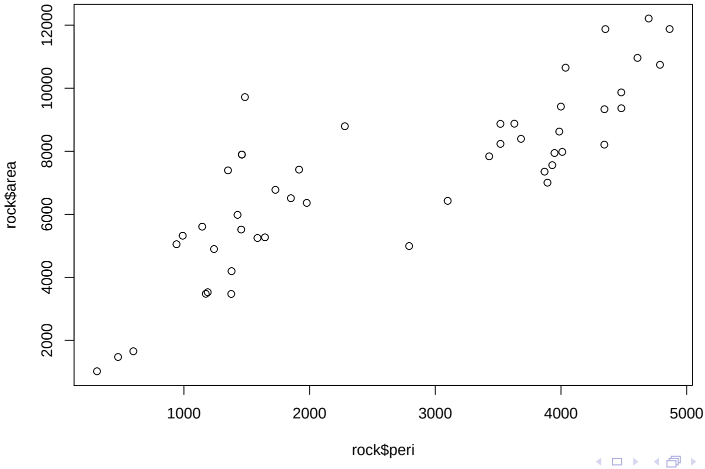
plot function

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



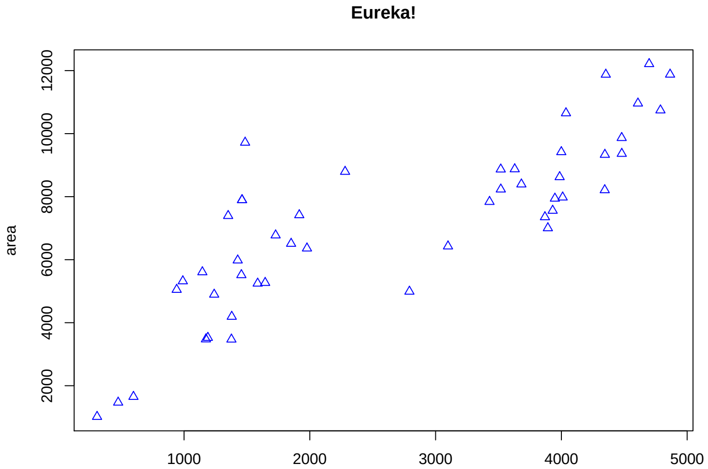
plot function

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



plot function

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

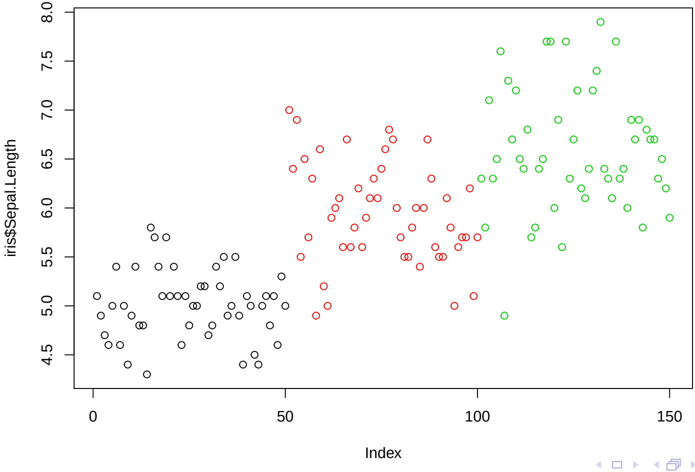


plot function: back to the mean

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

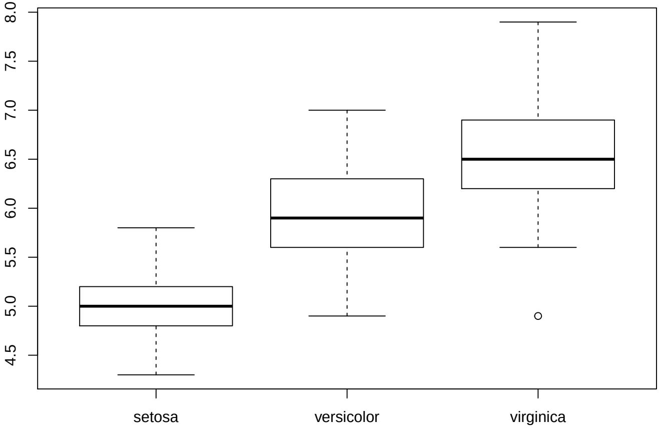
plot function: back to the mean

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



boxplots

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



1. The mean

2. Data-frames

3. Visualisation

4. T-test

Student's T.test introduction

```
?t.test
```

Student's T.test introduction

```r
?t.test
```

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

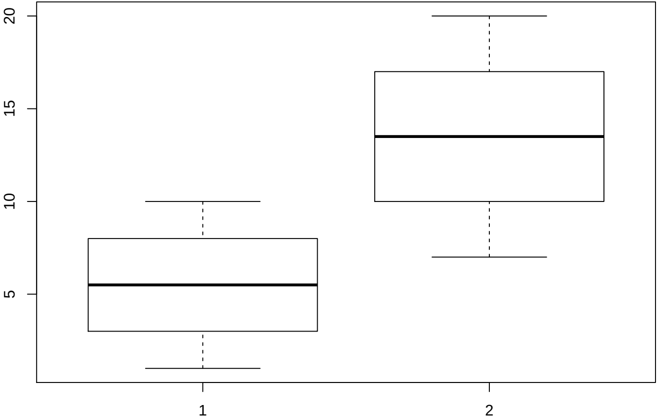
```

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"])
```