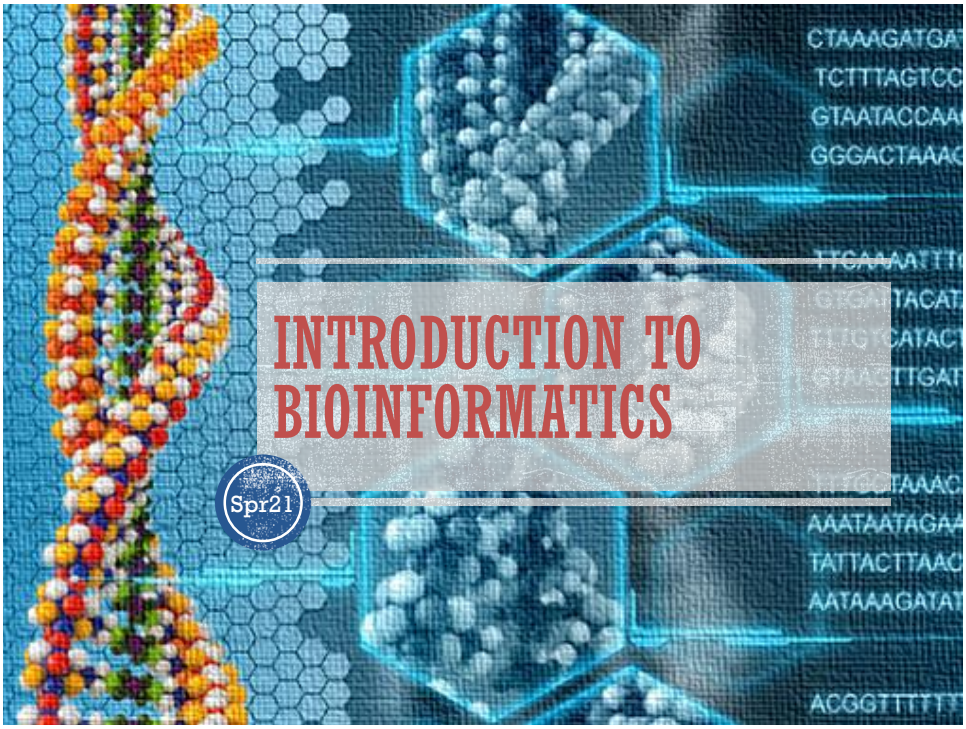


Introduction to R

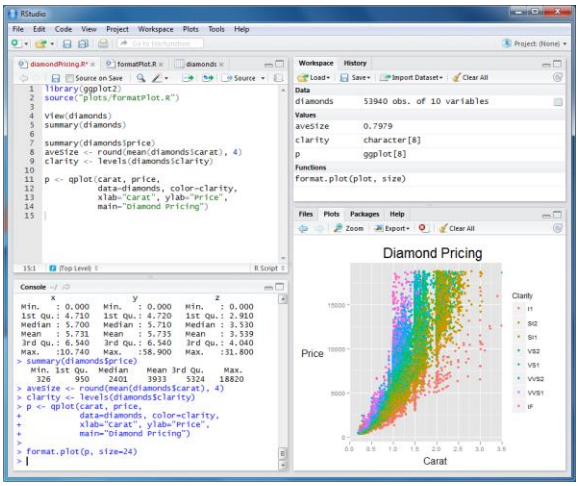
- What is R?
- Getting started
  - Commands
  - Objects
  - Data structures
  - Functions
  - packages
- Data exploration and visualization

What is R?

- R is:
  - an object-oriented programming language.
    - Everything in R is an object: e.g. datasets, functions, models, plots, etc.
  - an open source language
    - anyone can develop and distribute code to run on the R platform
    - Extensible – CRAN, Bioconductor, github, …
  - focused on manipulating and analyzing data
    - one of the most popular languages used by statisticians and data scientists
    - Providing robust computational statistics tool and environment for bioinformatics
  - Coherent and extensively documented
    - a massive community of contributers

- RStudio is an integrated development environment (IDE) for R.
  - It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.
- RStudio is a free environment for R
  - http://www.rstudio.com

IDE elements



Entering commands in R

- Interactive and interpreted – convenient and forgiving

1. via the console (bottom-left panel of RStudio)
   - the traditional way
   - The history panel automatically keep track of the entered commands
2. *R script:*
   - can be used to keep a record of the commands you used.
   - The R code can be run from inside the document and the results are displayed directly underneath
     - Click on the line and press CTRL and ENTER

use R as a calculator to compute a simple mathematical expression

Save the file you are working on!

Browse the computer and try to locate it

Exercise

Working Directory

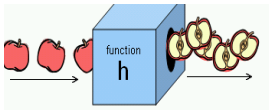
- It's always good to check and see where R will be saving your files–that includes data from your current session and any objects that you export from R
- To find the current working directory →`getwd()`.
- To change the default directory:
  - Run `setwd()`
    - include the path you want.
    - use of forward slashes ("/") in the path.
  - Use the browse command in the files window
    - set As Working Directory

Labeled key	Ctrl-key combination	Effect
Up arrow	Ctrl-P	Recall previous command by moving backward through the history of commands.
Down arrow	Ctrl-N	Move forward through the history of commands.
Backspace	Ctrl-H	Delete the character to the left of cursor.
Delete (Del)	Ctrl-D	Delete the character to the right of cursor.
Home	Ctrl-A	Move cursor to the start of the line.
End	Ctrl-E	Move cursor to the end of the line.
Right arrow	Ctrl-F	Move cursor right (forward) one character.
Left arrow	Ctrl-B	Move cursor left (back) one character.
	Ctrl-K	Delete everything from the cursor position to the end of the line.
	Ctrl-U	Clear the whole darn line and start over.
Tab		Name completion (on some platforms).

Keystrokes for command-line editing

Functions

- Functions transform inputs (arguments) to outputs, perhaps with side effects.
  - Arguments are always contained in parentheses – curved brackets, **()** – separated by commas.



- Examples:

```
print("Hello World")
```

```
print("The zero occurs at", 2*pi, "radians.")
```

```
sin (pi/2)
```

Functions

- Multiple-argument matching is done first by name, then by position

```
seq(from = 3, to = 20, by = 4)
```

```
seq(3, 20, 4)
```

- Functions may define (some) arguments to have default values
  - meaning we do not need to specify values for these in order to run the function.

```
rnorm(n=10)
```


Getting Help on a Function

- All arguments to a function and their default values are listed in the help page
  - (?) shortcut for the help command

```
? Function_name
```

  - to display the documentation for the function:

```
help(function_name)
```

  - for a quick reminder of the function arguments:

```
args(function_name)
```

  - to see examples of using the function:

```
example(function_name)
```

Objects/variables

- An object is an abstraction of a memory address used to store value(s).
- Object names should begin with a letter and can contain letters, numbers, underscores, or periods
  - remember that case matters!
- use the assignment 'operator', `<-` to create a variable and store some value in it.

```
myNumber <- 25
```

- Use `rm` function to permanently remove one or more objects from the workspace
- Ex:
  - perform arithmetic on variables using functions
  - Include them into expressions
  - Reassign their values

Define 3 objects to store numbers

Calculate the average of the three values

Use the mean function

Print the results

Exercise

Vectors

- The basic data structure in R is a vector
  - an ordered collection of values.
  - elements are of the same type.
  - R treats even single values as 1-element vectors
- To access :
  - individual elements → use the [] notation
  - Slices (range) → use the colon operator (`n:m`)
- The function `c` combines its arguments into a vector:
  - *I* specify it's arguments in curved brackets(...)

```
x <- c(1,2,3,4,5)
x[1]
```

```
firstNames <- c("Shinji", "Aska", "Rey", "Misato")
```


Vectors

- When applying all standard arithmetic/logical operations to vectors, application is element-wise.
  - R supports *vectorized* operations.

```
x <- seq(from=1, to=5, by=2)
y <- x*2
X == y
```

- Logical values are useful when to create subsets of data.
  - use *comparison* operators; `==`, `>`, `<`, `!=` to check values

```
x <- c("A","A", "B","B","C")
x == "A"
```

Define a vector of 5 numbers

Extract slices of the vector

Double the values in the slice

Try the `mean()`, `length()`, `sum()` functions on vector/slice

Exercise

Lists

- A list is a sequence of elements of <u>different</u> types.

```
lst <- list(0.5, "amal", 0.977, c(1,1,2,3))
```

- We can combine three vectors, each of a different type, into a single list

```
x <- c(1,2,3,4,5)
firstNames <- c("Shinji", "Aska", "Rey", "Misato")
gender <- c("f", "f", "f", "m", "m", "m", "m")
myList <- list(x=x, firstNames=firstNames, gender=gender)
```

Selecting List Elements

- To access specific elements within the list:
  - By position : [[list index]]
  - By name :
    - using $ followed by name of the element
    - [[name of the element]]
      - Don't forget to enclose name in ' '

Matrix

- Find about this data structure in R?

```
print(matrix(c(1,2,3,4), 2, 2))
```

Dataframes

- Data frames are two dimensional objects; think rows and columns.
  - Basically, tables of data.

```
dfrm <- data.frame(v1, v2, v3, f1, f2)
```

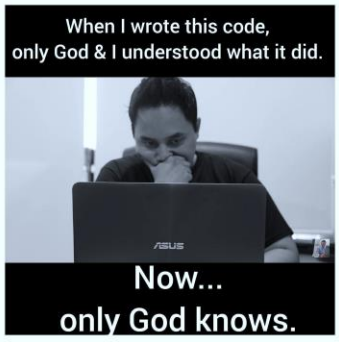
- You can manually create data frames by combining two vectors with the `data.frame` function

```
franchise <- c("Mets", "Nationals", "Marlins", "Phillies",
"Braves")
city <- c("New York", "Washington, DC", "Miami", "Philadelphia",
"Atlanta")
teams <- data.frame(franchise, city)
names(teams)
```


Comments

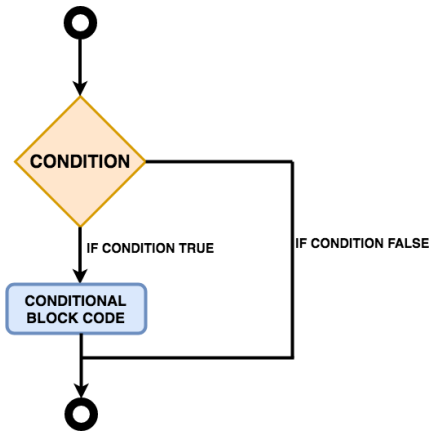
- Any text preceded by a # will be treated as a comment by R. That is, R will not try to execute it as code.

```
# This is a comment
# please use often!!
```



Selection statements

```
if (test) {
## code if TEST == TRUE }
else {
## code if TEST == FALSE }
```



loops

- loop constructs are for, while and repeat, with the additional clauses break and next.

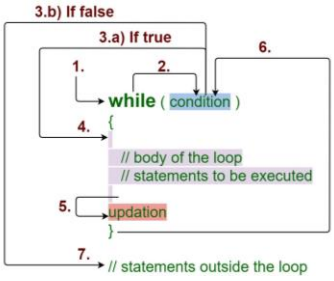
```
for (val in sequence) { statement }
```

```
while (test_expression) { statement }
```

```
repeat { statement }
```

```
if (test_expression) { break }
```

```
if (test_condition) { next }
```



User-defined functions

- Functions are used to logically break our code into simpler parts which become easy to maintain and understand.
- Syntax:

```
func_name <- function (argument) {
 statement
 return(expression)
}
```

- eg. a function for calculating the mean of a vector

Define a vector of 5 numbers

Define a function tht computes the mean of a vector.

Call the function using your vector

Compare your result eith the built-in mean() function

Exercise

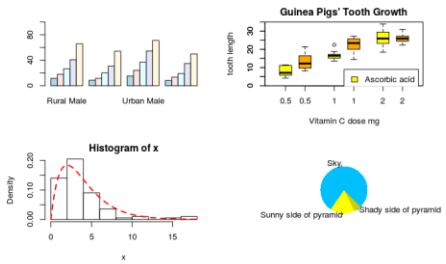
Q: how to define default Values for Arguments?

Packages in R

- The open-source nature of R encourages others to write their own functions for their particular data-type or analyses.
  - Check the Packages tab in the bottom-right panel of RStudio to see a list of all packages that you currently have installed.
- Packages are distributed through repositories.
  - The most-common ones are CRAN and Bioconductor.
- To install packages within R:
  - use `install.packages`
    - If your package is part of the main CRAN repository
- To load a package and make it's functions / data available in your current R session:
  - Use the `library` function.
    - You need to do this every time you load a new RStudio session.

Plotting with ggplot2

- The ggplot2 package provides a graphics paradigm, which is called the Grammar of Graphics.
- It uses a highly modular approach to graphics
  - lets you construct and customize your plots more easily.

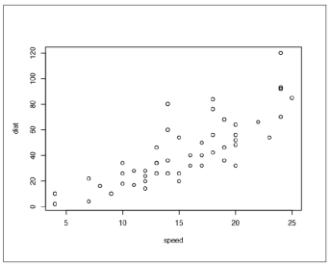


Creating a Scatter Plot

- You have paired observations: (*x*1, *y*1), (*x*2, *y*2), ..., (*xn*, *yn*). You want to create a scatter plot of the pairs.

```
plot(x, y)
```

- When calling plot:
  - Use the main argument for a title.
  - Use the xlab argument for an x-axis label.
  - Use the ylab argument for a y-axis label.



Questions?

More info?

https://www.datamentor.io/r-programming/