Introduction to

Introduction to Statistics and R

May 25,26,29,30, 2017

Sarah Bonnin – Bioinformatics facility

Estefania Mancini – lab Juan Valcárcel

CRG Centre for Genomic Regulation

prbb Barcelona Biomedical Research Park

Introduction to Statistics and R

- **Module 0: Introduction to R**
  May 25th, 26th, 29th and 30th

- **Module I: Descriptive Statistics & Intro to Probability**
  June 6th

- **Module II: Statistical Inference**
  June 8th

- **Module III: Statistical modeling & Regression**
  June 9th

Bioinformatics room
10:00-13:00

Lectures:
Ramon y Cajal
10:00-13:00

Practicums:
Bioinformatics room
14:00-17:00

Introduction to Statistics and R – Course page

https://biocore.crg.eu/wiki/CRG_Introduction_to_Statistics_and_R_2017

Outline

day 1 – May 25th

- R Studio
- R basics
- Data types
- Data structures: Vectors
- Input/Output

Outline

day 2 – May 26th

- Data structures:
  - Matrix
  - Data Frames
- More on Input and Output

Outline

day 3 – May 29th

- Lists
- Library / packages
  - CRAN
  - Bioconductor
- (Writing functions in R)

Outline

day 4 – May 30th

- Graphing in R:
  - basic graphing
  - introduction ggplot2 package

What is R?

- Used for **data manipulation**, **calculation** and **graphical display.**

- **Open source !**

https://www.r-project.org/

- **Interactive, flexible**

- **Very active community of developers and users!**

R Studio

R studio?

- **Free and open source** Integrated Development Environment for R

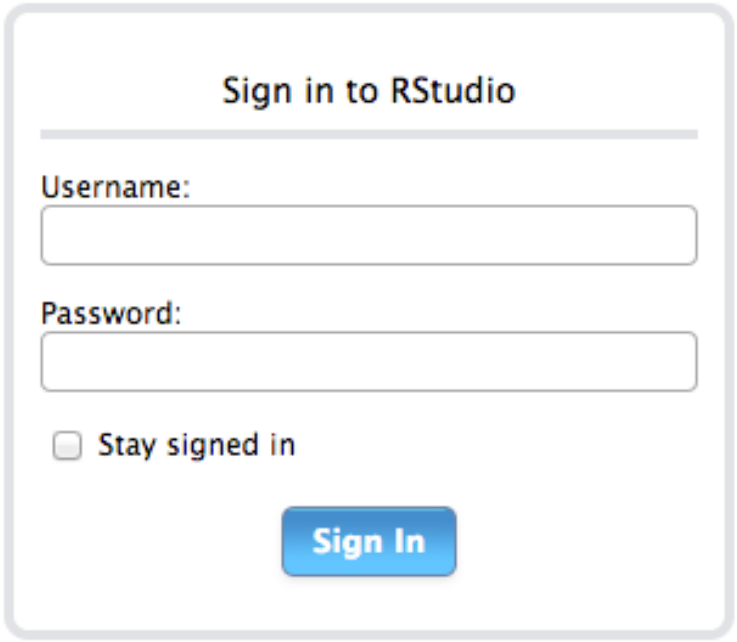
- Available for Windows, Mac OS and Linux

Install R Studio locally

- **https://www.rstudio.com/products/rstudio/download2/**

Connect to R Studio server on the CRG cluster via a web browser

- **http://rstudio.linux.crg.es/**



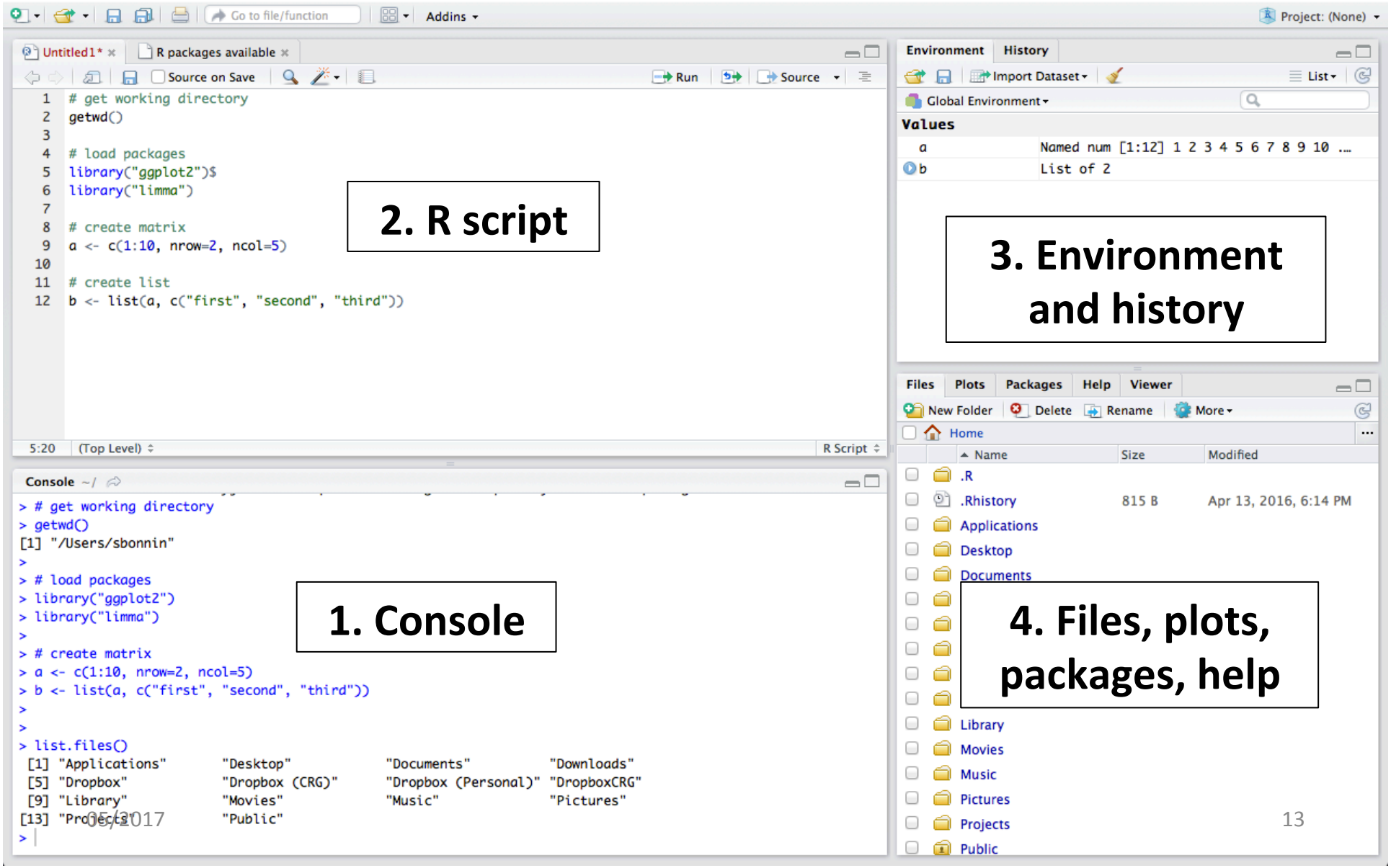
Connect to R Studio on the CRG server via a web browser

- **http://rstudio.linux.crg.es/**

- **Connect with your CRG mail/cluster credentials**

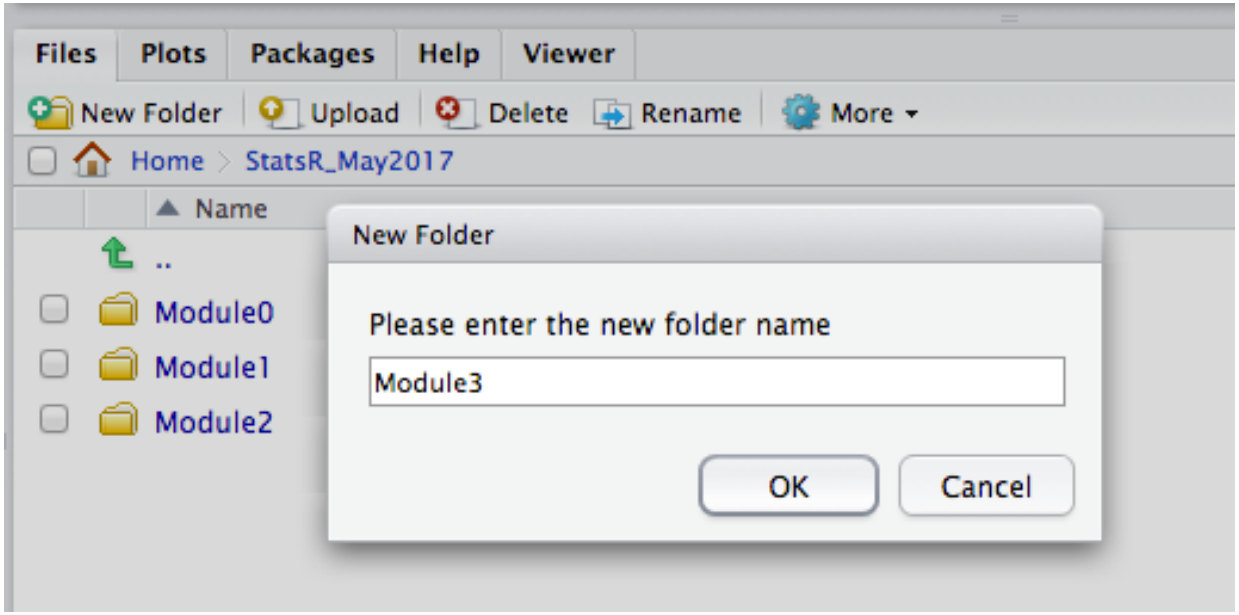
R Studio

Screen: 4 windows



Create a directory and subdirectories for the course

StatsR_May2017/
|-- Module0
|-- Module1
|-- Module2
|-- Module3



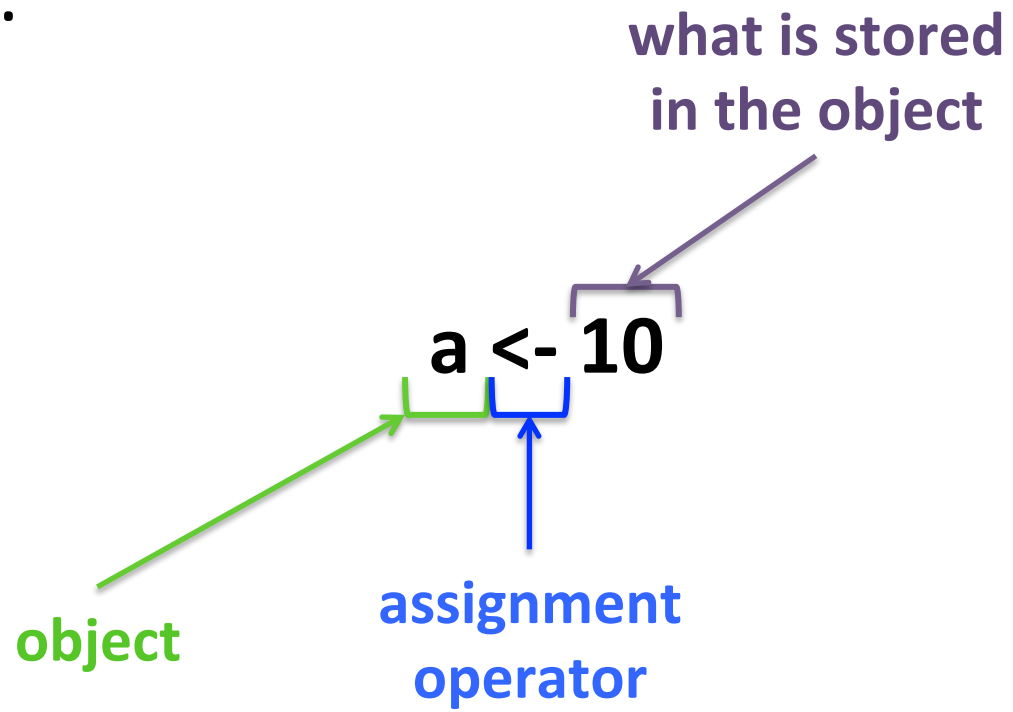
R Basics

Elementary arithmetic operators

addition	**+**
subtraction	**-**
division	**/**
multiplication	*****
exponentiation	**^** or **

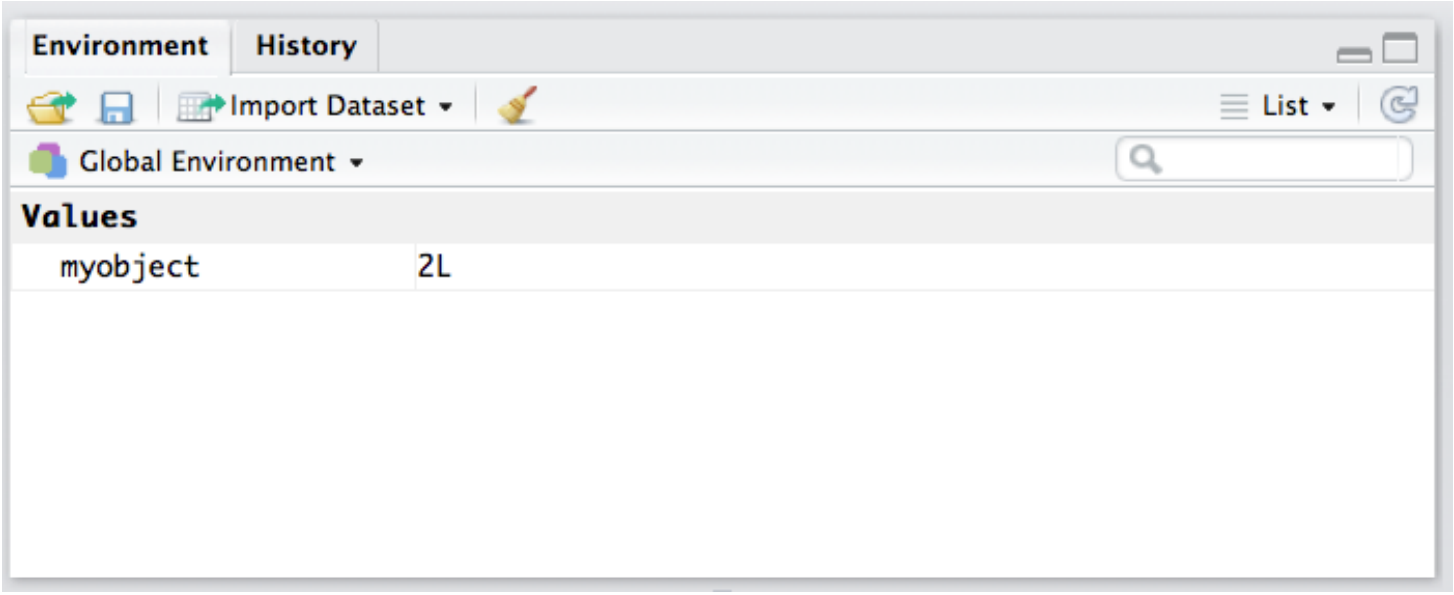
R base syntax

example:



Objects in R Studio

The object can be found in the "Environment" tab of R Studio.

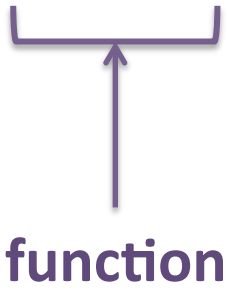


R functions

function()

example:

getwd()

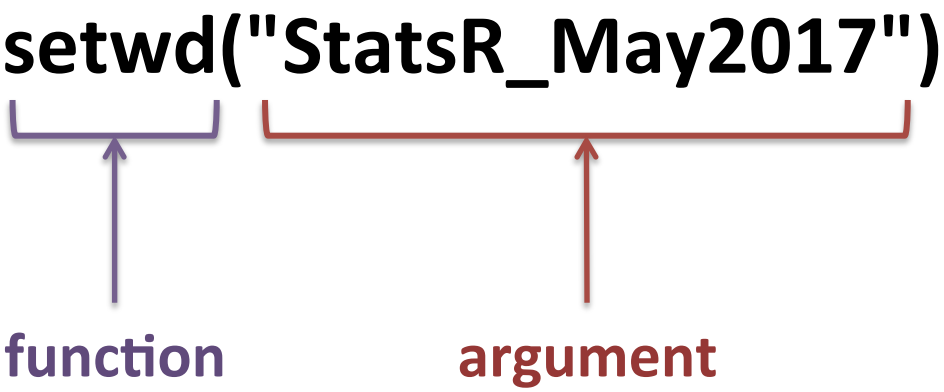


function

R functions

function(argument)

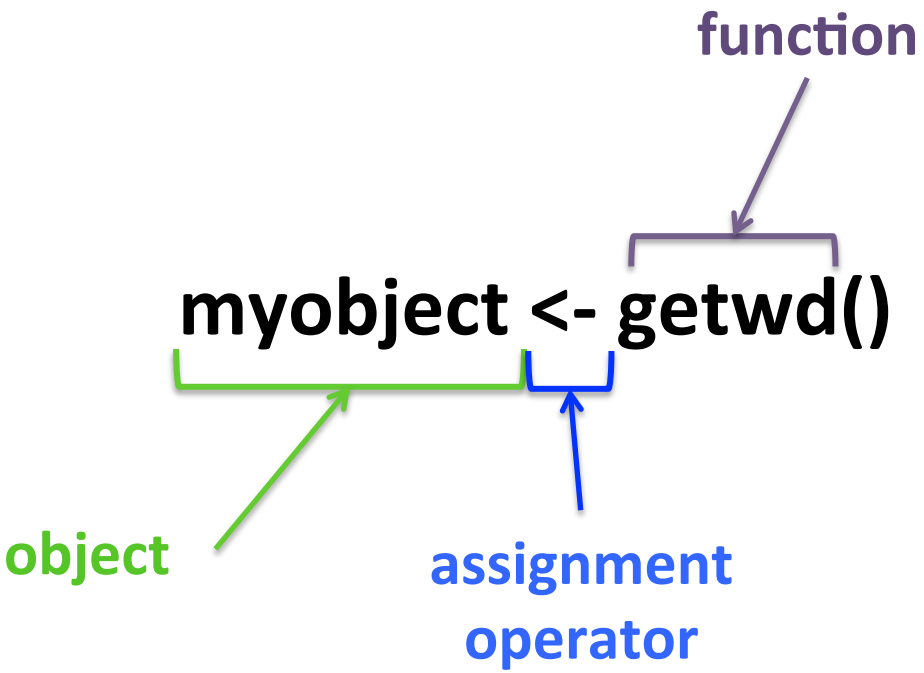
example:



R functions

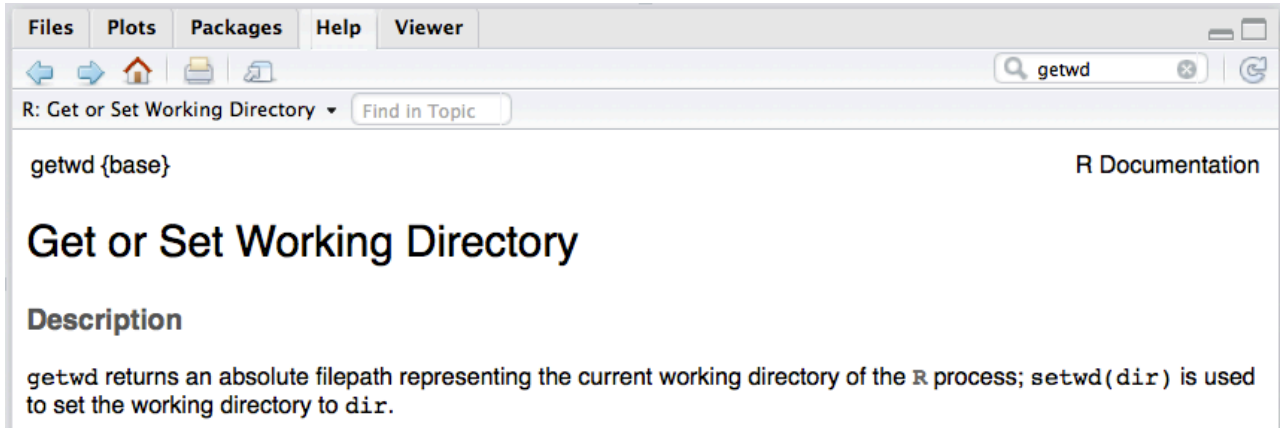
a <- function()

example:



Getting help on a function

- help(**getwd**)
- ?**getwd**
- Tab "help" in R Studio



Example of functions

- example(**mean**)

```
mean> x <- c(0:10, 50)
mean> xm <- mean(x)
```

R script

- Store commands in a ".R" script
- Execute commands or blocks of commands from R studio

R syntax

- Case sensitive
- Comment lines start with **#**
- Commands separated by **;** or a new line

Data Types

Numeric

Character

Logical

Data Types

- Every object has a **data type** that tells what sort of value it is:
  - Numeric (Numbers)
  - Character (Text)
  - Logical (True / False)

Data Types

- a <- 10
  - str(a)

  num 10

- b <- "word"
  - str(b)

  chr "word"

Data Structures

Vectors

Matrices

Data frames

List

Data Structures

Vectors

Matrices

Data frames

List

Vectors

- Sequence of data elements of the same type

- Assignment of values to vector using the

  c command (combining elements)

```
a <- c(1, 9, 4, 8, 0, 11, 7)
```

Numeric vectors

- Create a sequence of consecutive numbers

```
a <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

```
a <- 1:10
```

Character vectors

- each element is between (double) quotes

```
b <- c("test", "ok”)
```

Vector manipulation

- Fetch elements of a vector **a**:

values	**1**	**9**	**4**	**8**	**0**	**11**	**7**

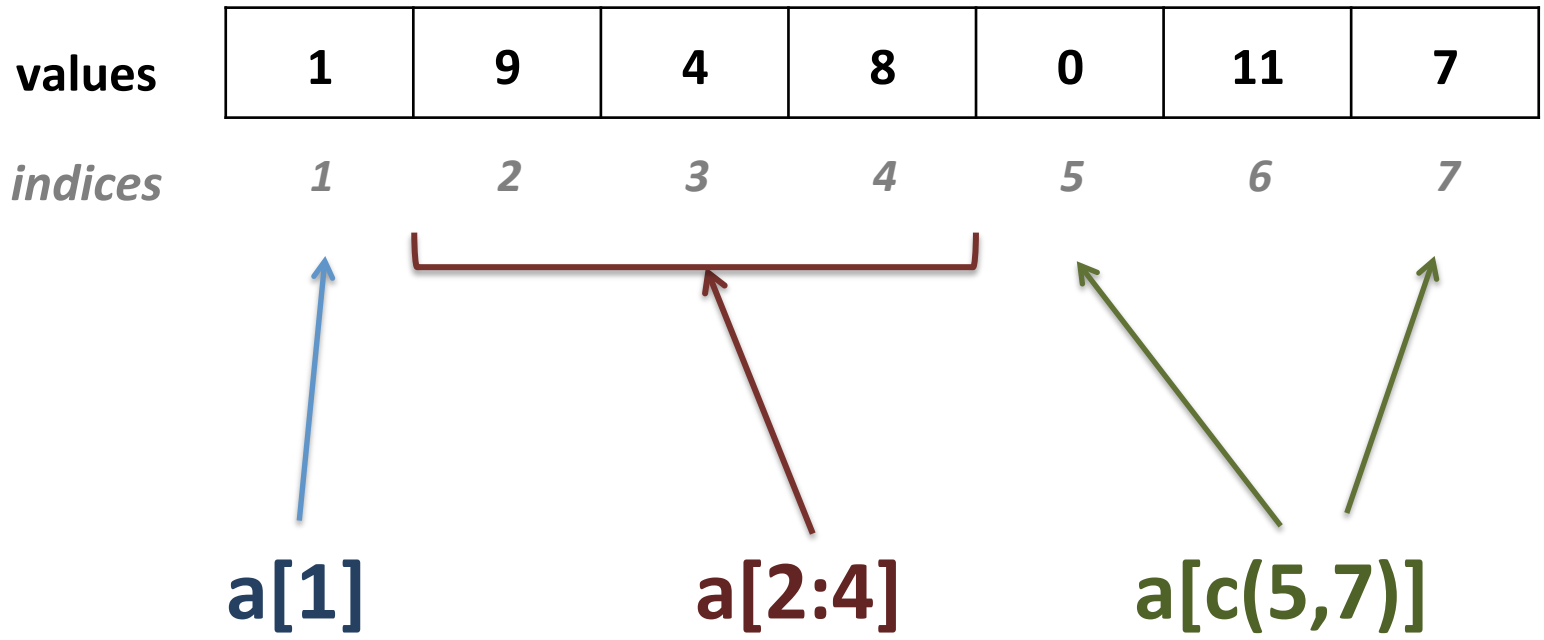
Vector manipulation

- Fetch elements of a vector **a**:

values	**1**	**9**	**4**	**8**	**0**	**11**	**7**
indices	*1*	*2*	*3*	*4*	*5*	*6*	*7*

Vector manipulation

- Fetch elements of a vector **a**:



Vector manipulation

- Get length of a vector

```
a <- 1:5
[1] 5
length(a)
```

Vector manipulation

- Replacing a vector's value:

```
a <- 1:5
[1] 1 2 3 4 5

a[2] <- 3
[1] 1 3 3 4 5
```

Numeric vector manipulation

- Add 2 to each element of a numeric vector

```
a <- 1:5
a + 2
[1] 3 4 5 6 7
```

Numeric vector manipulation

- a <- 1:5
- Is **2** present in **a**?

```
a == 2
```

```
[1] FALSE  TRUE FALSE FALSE FALSE
```

- Which element of **a** are superior to **2**

```
a > 2
```

```
[1] FALSE FALSE  TRUE  TRUE  TRUE
```

Numeric vector manipulation

- a <- 1:5
- Select elements superior or equal to 2

a >= 2

1	2	3	4	5

[1] FALSE TRUE TRUE TRUE TRUE

a[a >= 2]

2	3	4	5

Vectors comparison
the **%in%** operator

- a <- 2:6
- b <- 4:20

- What elements of **a** are in **b**?

a %in% b

```
[1] FALSE FALSE  TRUE  TRUE  TRUE
```

- Get the actual values of **a** that are in **b**

a[a %in% b]

```
[1] 4 5 6
```

Logical operators

inferior	**<**
inferior or equal	**<=**
superior	**>**
superior or equal	**>=**
equality	**==**
inequality	**!=**
not x	**!x**
x OR y	**x | y**
x AND y	**x & y**

Obtaining summary statistics

average/mean	**mean(x)**
median	**median(x)**
minimum	**min(x)**
maximum	**max(x)**
variance	**var(x)**
correlation	**cor(x)**

Exercise 1:
Numeric vector manipulation

Character vector manipulation

- b <- c("ok", "yes", "no", "ok")
- Select all the "ok" elements

b == "ok"

ok	yes	no	ok

[1] TRUE FALSE FALSE TRUE

b[b == "ok"]

ok	ok

Character vector manipulation

- b <- c("ok", "yes", "no", "ok")
- Select all the elements that are **not** "ok"

b != "ok"

ok	yes	no	ok

[1] FALSE TRUE TRUE FALSE

b[b != "ok"]

yes	no

Exercise 2:
Character vector manipulation

Exercise 1

Vector manipulation

- Create vector **x** of 1000 random numbers.

  rnorm()

- Compute the mean, median, minimum and maximum of this vector.

  mean(), median(), min(), max().

- Create **y**, vector containing the first 100 elements of **x**.

- Compute summary statistics of **y**

  summary()

Input / Output

File/directory path

- The path of a file/directory is its **location** in the file system.
- Your home directory is the one that hosts your personal folder:

/nfs/users/[yourgroup]/[yourlogin]

- Shortcut to your home directory**:** ~

File/directory path

- Path of the current directory obtained with: **getwd**()

[1] "/nfs/users/bi/sbonnin"

- Move one directory up with: **setwd**("..")

You are now in: "/nfs/users/bi/"

- Go to another directory:

setwd("/nfs/users/bi/sbonnin/test")

Read in a file into a vector

- **scan**("file.txt")

By default, scans "double" elements.

→can fail if the input contains characters

- **scan**("file.txt", what="character")

→ specify the type of data to scan!

- **scan**("/users/bi/sbonnin/Dir/file.txt",)

→ If file is not in the current directory:

Write a vector into a file

- **write**(ourfile, file="ourfile.txt")
- **write**(ourfile, file="ourfile.txt", ncolumns=1)

→organizes the data in one column

Exercise 3:
R script

R script

Create an empty file **ex2.R**

Within that file:

- Create a directory "exercise2" – **dir.create()**
- Go to directory "exercise2" – **setwd()**
- Read file **TO ADD** into tt object – **scan()**
- Add 10 to each element of the vector
- Write result into a file – **write()**
- Goes back to home directory – **setwd("~")**

Execute script **ex2.R** – source in RStudio

Factors

- Factors are **categorical vectors**

- Important for:
  - plotting
  - statistical modeling

Factors

- Like vectors but with **levels** that represent each category

```
e <- factor(c("high", "low", "medium", "low"))

e
[1] high   low    medium low
Levels: high low medium
```

Factors

- Levels can be given **ordered**

```
e <- factor(e, levels=c("low", "medium", "high"),
ordered=TRUE)

e
[1] high   low    medium low
Levels: low < medium < high
```

Factors

- Example of a character vector versus a factor

```
e <- factor(c("high", "low", "medium", "low"))
e2 <- c("high", "low", "medium", "low")

str(e)
 Factor w/ 3 levels "high","low","medium": 1 2 3 2
str(e)
 chr [1:4] "high" "low" "medium" "low"
```