

Introduction to R

LONDON
SCHOOL of
HYGIENE
& TROPICAL
MEDICINE

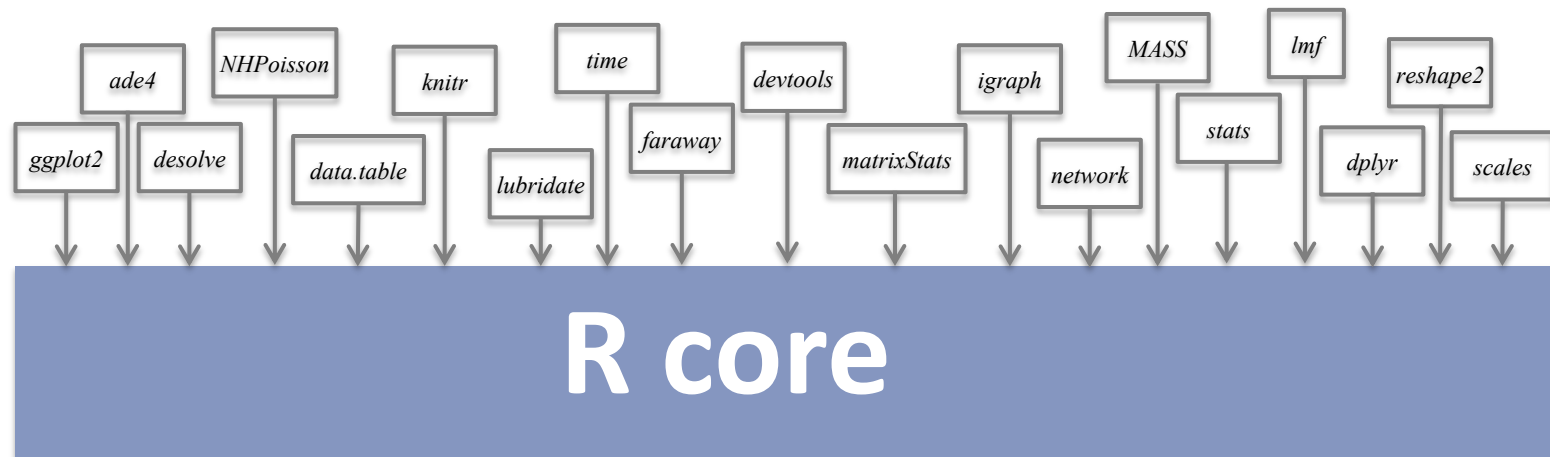


What is R?



Programming language

- Free, open source core, maintained and regularly updated
- Statistics and data-focussed
- Add-on packages made by users



Why use R?

Free

Active development

No limit on what you can do:

- Statistics
- Data cleaning/processing
- Interacting with websites
- ...

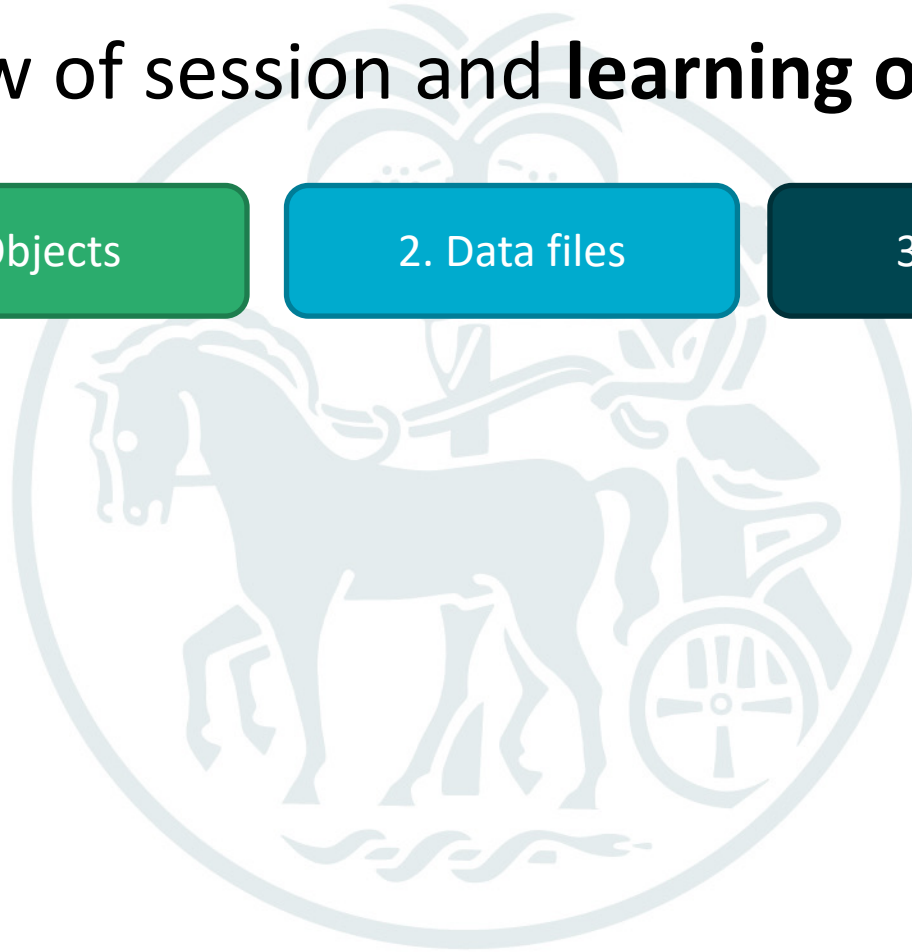


Overview of session and **learning objectives**

1. Objects

2. Data files

3. Plotting



Overview of session and learning objectives

1. Objects

Familiarise yourself with:

- RStudio
- working directory

Execute statements for:

- basic R functionality
- built-in R functions
- ask R for help with a function

Consider:

- how to search for help online

Write code to:

- explore basic R functionality

2. Data files

Execute statements for:

- reading files
- viewing data
- manipulating data

Consider:

- different ways to access the same information
- how to trouble-shoot file read in issues

Write code:

- to correct an issue

3. Plotting

Execute statements for:

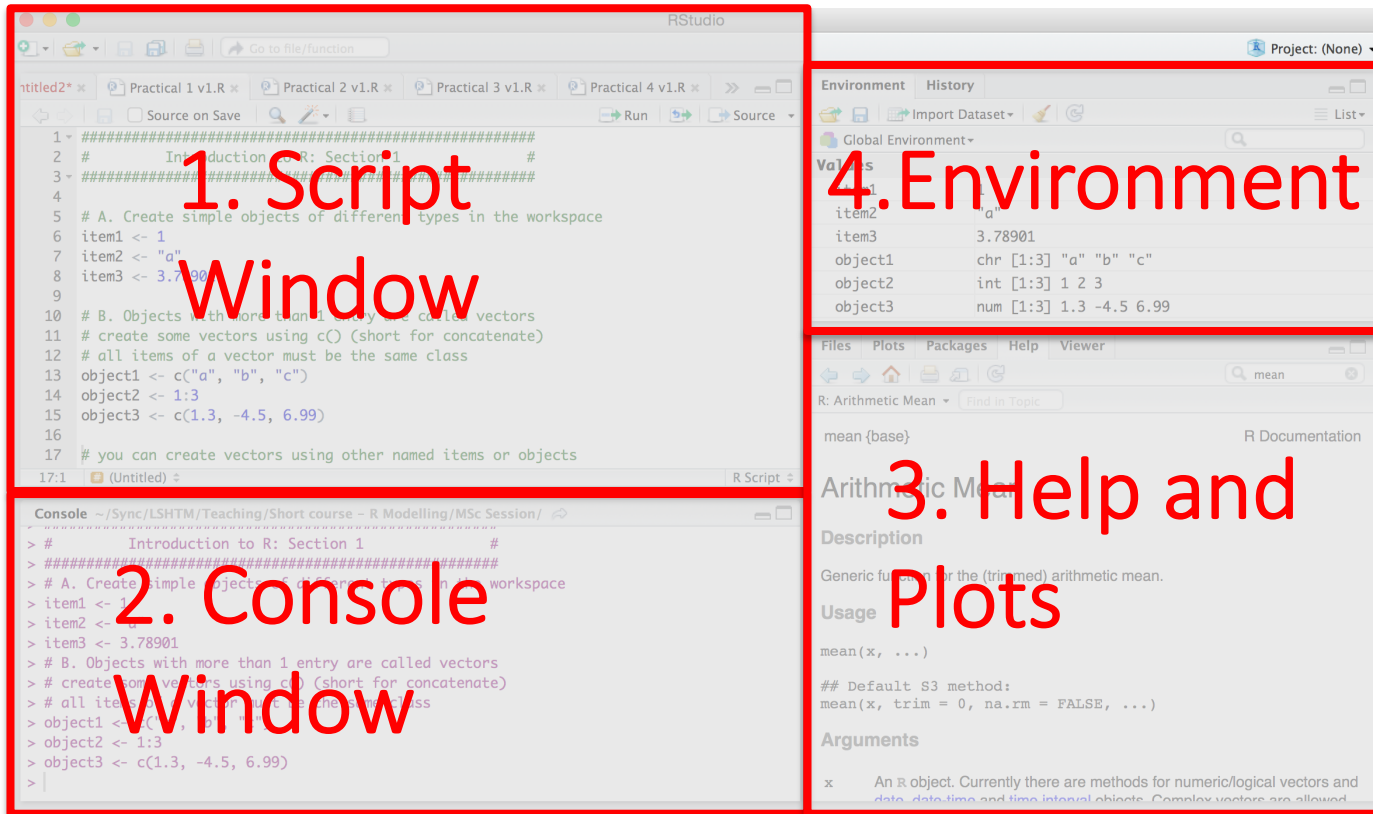
- displaying data on a plot
- personalising your plot e.g. colour, axis labels, title

Consider:

- different ways to plot data

Session 1: Set working directory

- Open RStudio
- Set your working directory to wherever you saved the folder of practicals:
 - Session → Set working directory → Choose directory
 - Then navigate to the right folder



1. Script Window

```
1- #####  
2- # Introduction to R: Section 1 #  
3- #####  
4-  
5- # A. Create simple objects of different types in the workspace  
6- item1 <- 1  
7- item2 <- "a"  
8- item3 <- 3.78901  
9-  
10- # B. Objects with more than 1 entry are called vectors  
11- # create some vectors using c() (short for concatenate)  
12- # all items of a vector must be the same class  
13- object1 <- c("a", "b", "c")  
14- object2 <- 1:3  
15- object3 <- c(1.3, -4.5, 6.99)  
16-  
17- # you can create vectors using other named items or objects  
17:1 (Untitled) R Script
```

2. Console Window

```
> # Introduction to R: Section 1 #  
> #####  
> # A. Create simple objects of different types in the workspace  
> item1 <- 1  
> item2 <- "a"  
> item3 <- 3.78901  
> # B. Objects with more than 1 entry are called vectors  
> # create some vectors using c() (short for concatenate)  
> # all items of a vector must be the same class  
> object1 <- c("a", "b", "c")  
> object2 <- 1:3  
> object3 <- c(1.3, -4.5, 6.99)  
>
```

3. Help and Plots

R: Arithmetic Mean

mean {base}

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)
```

Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)

Arguments

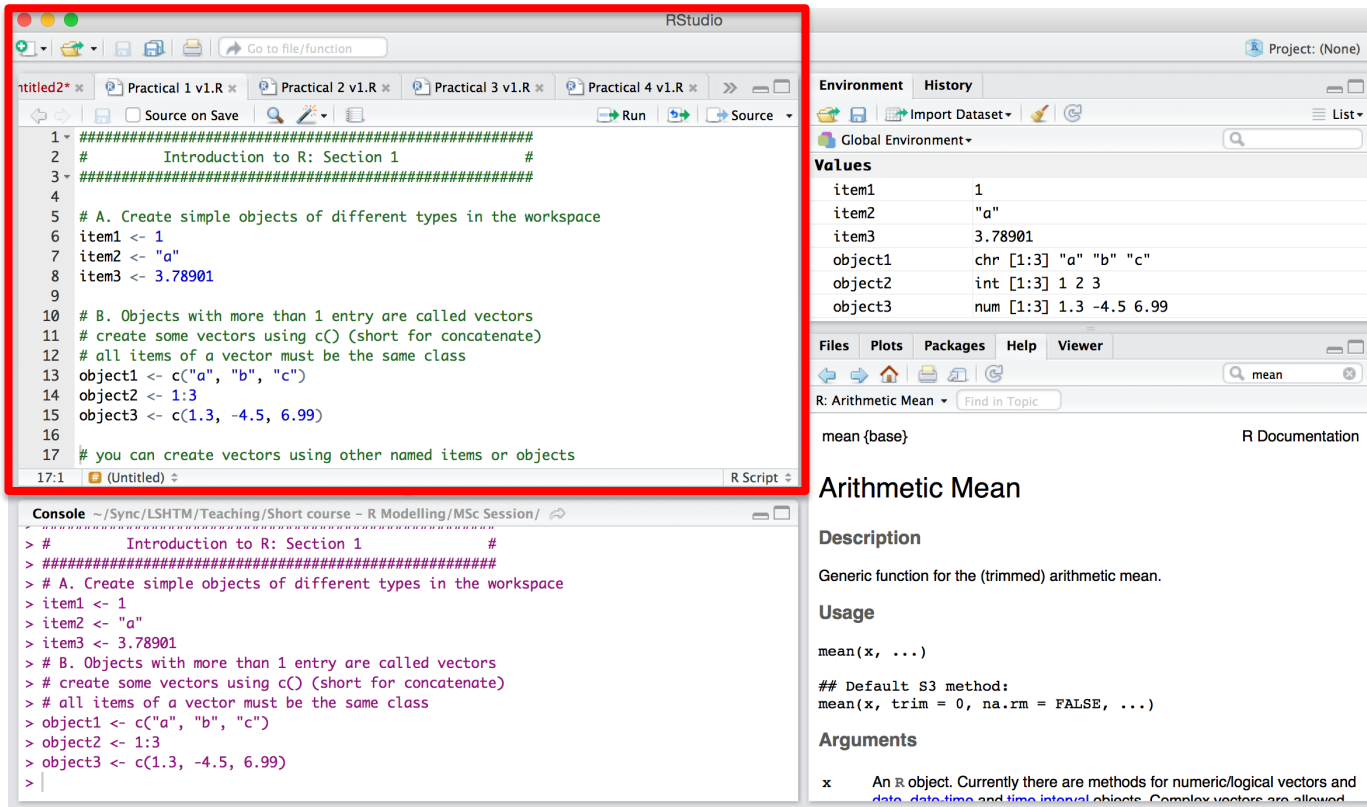
x An R object. Currently there are methods for numeric/logical vectors and data, date, time, and time interval objects. Complex vectors are allowed.

4. Environment

Variable	Value
item1	1
item2	"a"
item3	3.78901
object1	chr [1:3] "a" "b" "c"
object2	int [1:3] 1 2 3
object3	num [1:3] 1.3 -4.5 6.99

Script window

Can have multiple R files open in tabs



The screenshot displays the RStudio interface with three main panels:

- Script Window (left):** Contains R code for creating objects and vectors. The code is as follows:


```

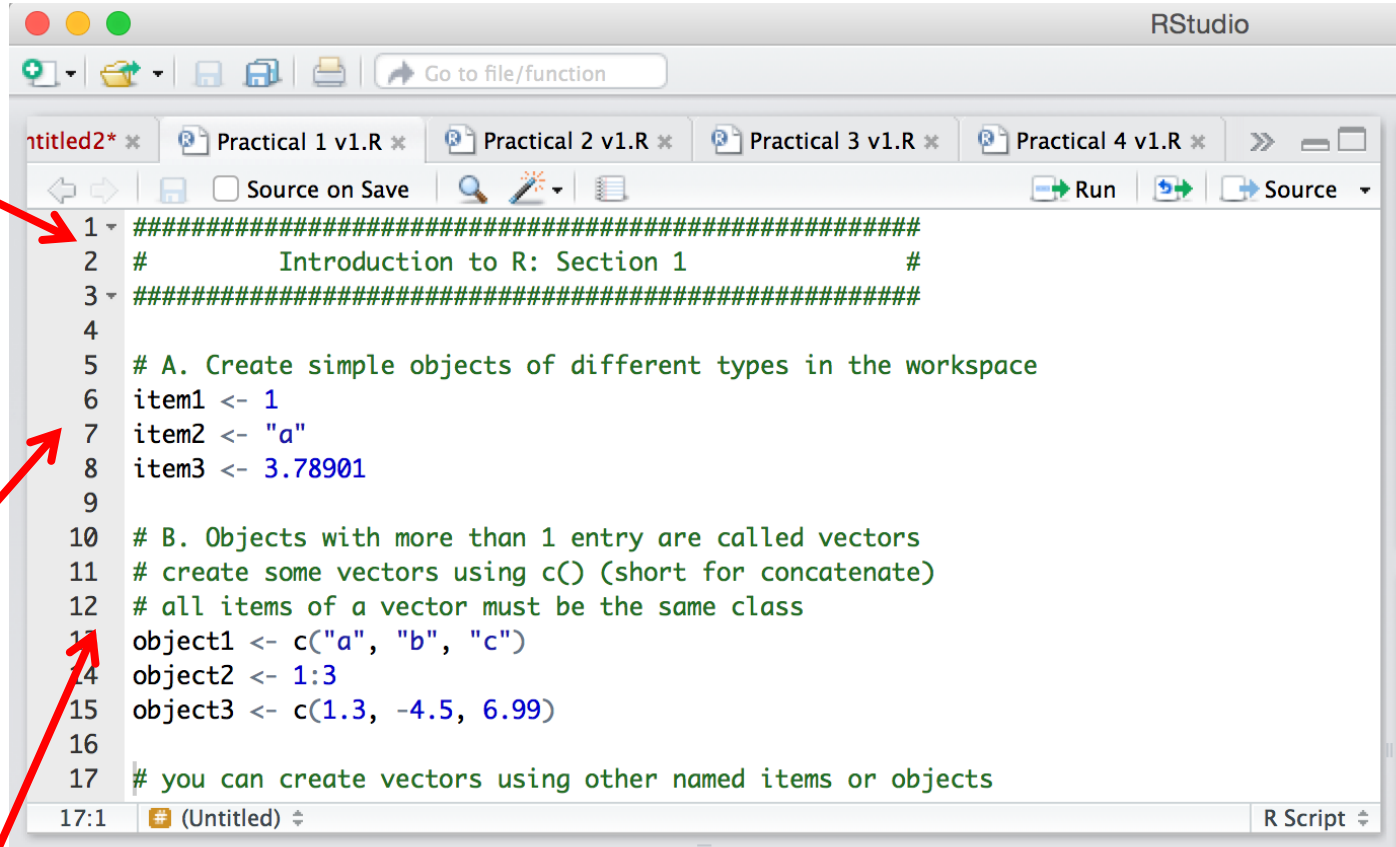
1- #####
2- # Introduction to R: Section 1 #
3- #####
4-
5- # A. Create simple objects of different types in the workspace
6- item1 <- 1
7- item2 <- "a"
8- item3 <- 3.78901
9-
10- # B. Objects with more than 1 entry are called vectors
11- # create some vectors using c() (short for concatenate)
12- # all items of a vector must be the same class
13- object1 <- c("a", "b", "c")
14- object2 <- 1:3
15- object3 <- c(1.3, -4.5, 6.99)
16-
17- # you can create vectors using other named items or objects
17:1 (Untitled)
      
```
- Console Window (bottom left):** Shows the output of the R code:


```

> # Introduction to R: Section 1 #
> #####
> # A. Create simple objects of different types in the workspace
> item1 <- 1
> item2 <- "a"
> item3 <- 3.78901
> # B. Objects with more than 1 entry are called vectors
> # create some vectors using c() (short for concatenate)
> # all items of a vector must be the same class
> object1 <- c("a", "b", "c")
> object2 <- 1:3
> object3 <- c(1.3, -4.5, 6.99)
>
      
```
- Viewer Window (right):** Displays the R documentation for the `mean` function. The title is "Arithmetic Mean". The description states: "Generic function for the (trimmed) arithmetic mean." The usage is `mean(x, ...)`. The default S3 method is `mean(x, trim = 0, na.rm = FALSE, ...)`. The arguments section indicates that `x` is an R object, currently with methods for numeric/logical vectors and `date`, `date.time`, and `time.interval` objects. Complex vectors are allowed.

marks a commented line
When sent to console, does not get run
RStudio shows comments in a different colour

“Syntax highlighting”
RStudio shows R code in different colours according to what it is



```

1 #####
2 # Introduction to R: Section 1 #
3 #####
4
5 # A. Create simple objects of different types in the workspace
6 item1 <- 1
7 item2 <- "a"
8 item3 <- 3.78901
9
10 # B. Objects with more than 1 entry are called vectors
11 # create some vectors using c() (short for concatenate)
12 # all items of a vector must be the same class
13 object1 <- c("a", "b", "c")
14 object2 <- 1:3
15 object3 <- c(1.3, -4.5, 6.99)
16
17 # you can create vectors using other named items or objects
  
```

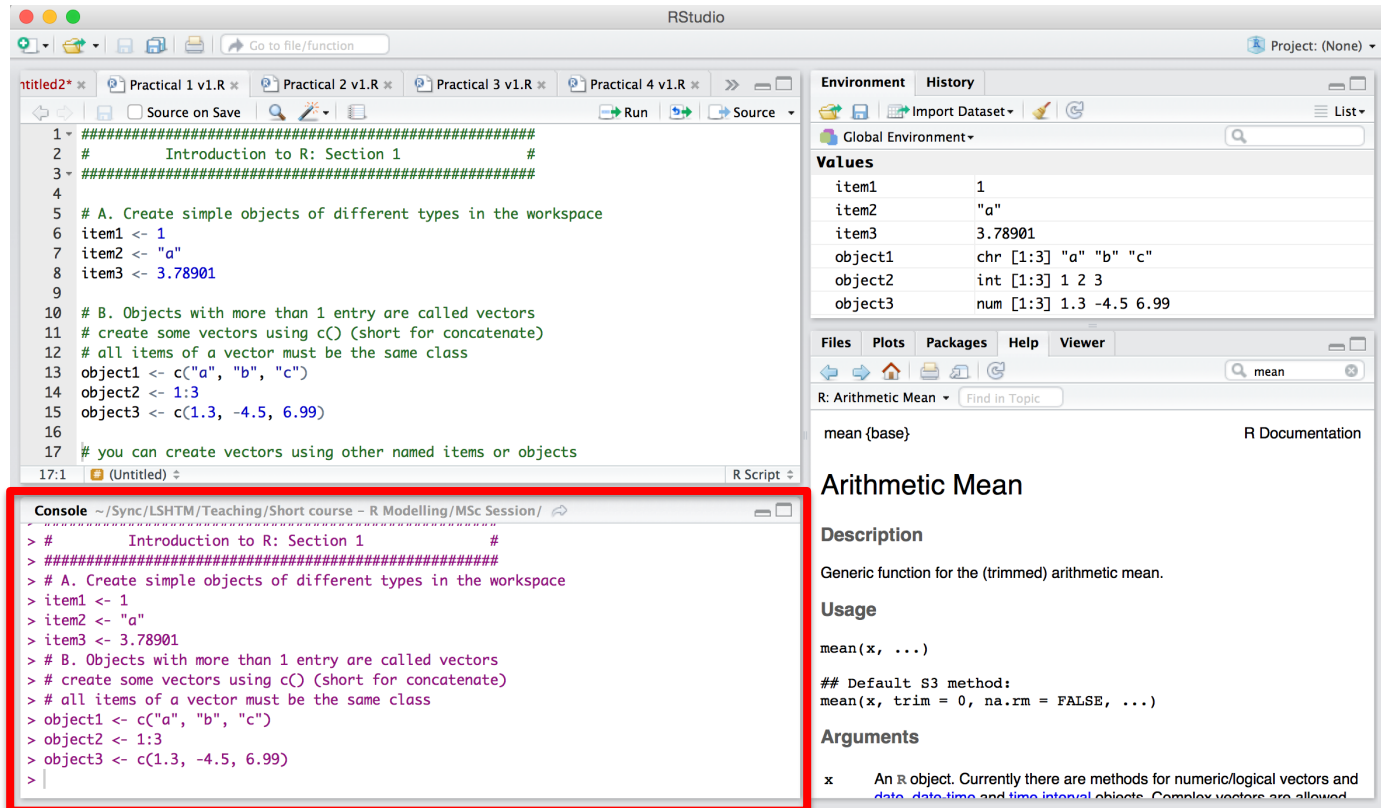
Keep your script tidy:
- Space out sections
- Write useful comments

Console window

Where R code from the script window is run

Commands and results in same window

Ctrl+enter
(Windows)



The screenshot shows the RStudio interface. The script window contains R code for creating objects and vectors. The console window shows the execution of this code. The help window displays the documentation for the `mean` function.

```

1- #####
2- # Introduction to R: Section 1 #
3- #####
4-
5- # A. Create simple objects of different types in the workspace
6- item1 <- 1
7- item2 <- "a"
8- item3 <- 3.78901
9-
10- # B. Objects with more than 1 entry are called vectors
11- # create some vectors using c() (short for concatenate)
12- # all items of a vector must be the same class
13- object1 <- c("a", "b", "c")
14- object2 <- 1:3
15- object3 <- c(1.3, -4.5, 6.99)
16-
17- # you can create vectors using other named items or objects
17:1 (Untitled)
  
```

```

> # Introduction to R: Section 1 #
> #####
> # A. Create simple objects of different types in the workspace
> item1 <- 1
> item2 <- "a"
> item3 <- 3.78901
> # B. Objects with more than 1 entry are called vectors
> # create some vectors using c() (short for concatenate)
> # all items of a vector must be the same class
> object1 <- c("a", "b", "c")
> object2 <- 1:3
> object3 <- c(1.3, -4.5, 6.99)
>
  
```

Environment

Values	
item1	1
item2	"a"
item3	3.78901
object1	chr [1:3] "a" "b" "c"
object2	int [1:3] 1 2 3
object3	num [1:3] 1.3 -4.5 6.99

Files **Plots** **Packages** **Help** **Viewer**

R: Arithmetic Mean

mean {base} R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

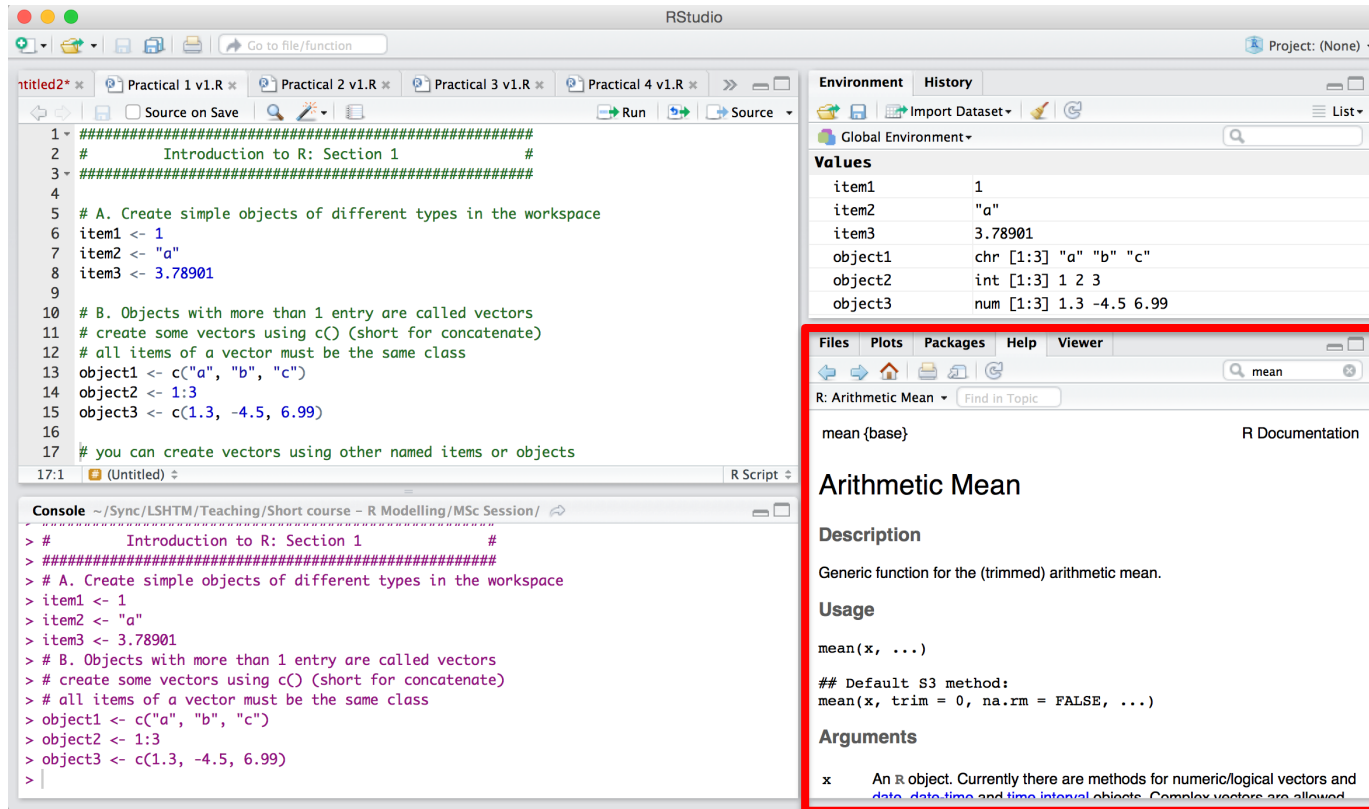
```
mean(x, ...)
```

Default S3 method:

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

x An R object. Currently there are methods for numeric/logical vectors and `date`, `date-time` and `time-interval` objects. Complex vectors are allowed.



The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for creating objects and vectors.


```

1- #####
2 # Introduction to R: Section 1 #
3- #####
4
5 # A. Create simple objects of different types in the workspace
6 item1 <- 1
7 item2 <- "a"
8 item3 <- 3.78901
9
10 # B. Objects with more than 1 entry are called vectors
11 # create some vectors using c() (short for concatenate)
12 # all items of a vector must be the same class
13 object1 <- c("a", "b", "c")
14 object2 <- 1:3
15 object3 <- c(1.3, -4.5, 6.99)
16
17 # you can create vectors using other named items or objects

```
- Environment:** Shows the current workspace with variables:

Variable	Value
item1	1
item2	"a"
item3	3.78901
object1	chr [1:3] "a" "b" "c"
object2	int [1:3] 1 2 3
object3	num [1:3] 1.3 -4.5 6.99
- Console:** Shows the execution output of the code in the source editor.
- Viewer:** Displays the R documentation for the `mean` function, which is highlighted with a red box.

Arithmetic Mean

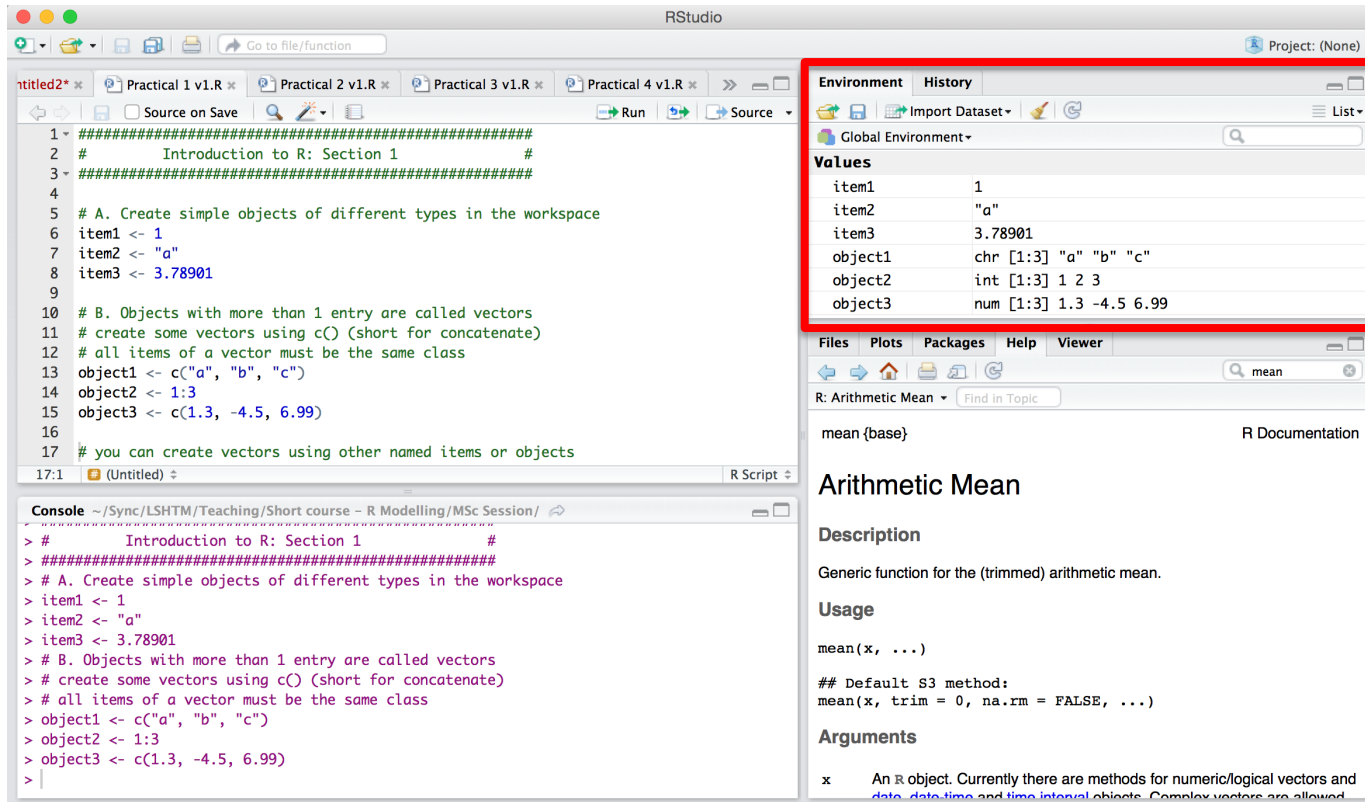
Description
Generic function for the (trimmed) arithmetic mean.

Usage
`mean(x, ...)`

Arguments
`x` An R object. Currently there are methods for numeric/logical vectors and `data`, `date.time` and `time.interval` objects. Complex vectors are allowed.

Help, Plotting,
packages
Use different
tabs depending
on what you
need

Objects in your environment, and history of commands run. Use different tabs depending on what you need



The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for creating objects and vectors.


```

1 #####
2 # Introduction to R: Section 1 #
3 #####
4
5 # A. Create simple objects of different types in the workspace
6 item1 <- 1
7 item2 <- "a"
8 item3 <- 3.78901
9
10 # B. Objects with more than 1 entry are called vectors
11 # create some vectors using c() (short for concatenate)
12 # all items of a vector must be the same class
13 object1 <- c("a", "b", "c")
14 object2 <- 1:3
15 object3 <- c(1.3, -4.5, 6.99)
16
17 # you can create vectors using other named items or objects
      
```
- Console:** Shows the execution of the code above.


```

> # Introduction to R: Section 1 #
> #####
> # A. Create simple objects of different types in the workspace
> item1 <- 1
> item2 <- "a"
> item3 <- 3.78901
> # B. Objects with more than 1 entry are called vectors
> # create some vectors using c() (short for concatenate)
> # all items of a vector must be the same class
> object1 <- c("a", "b", "c")
> object2 <- 1:3
> object3 <- c(1.3, -4.5, 6.99)
>
      
```
- Environment/History Panel (highlighted in red):** Shows the current state of the workspace.

Object Name	Value
item1	1
item2	"a"
item3	3.78901
object1	chr [1:3] "a" "b" "c"
object2	int [1:3] 1 2 3
object3	num [1:3] 1.3 -4.5 6.99
- Viewer Panel:** Shows the R documentation for the `mean` function.

Arithmetic Mean

Description
Generic function for the (trimmed) arithmetic mean.

Usage
`mean(x, ...)`

Arguments
x An R object. Currently there are methods for numeric/logical vectors and `date`, `date-time` and `time-interval` objects. Complex vectors are allowed.

Session 1: Running / executing

Running lines of code from script window

- Buttons – “Run” at the top of script window
- Shortcut – Ctrl + Enter (Windows); Cmd + Enter (Mac)

Run single lines or multiple lines

Session 1: Creating objects

Create objects in the workspace, or read them in from files using the “assignment operator”: `<-`

- `my.object <- “orange”`

- `this.object <- c(1, 2, 3)`

Then these objects exist in the workspace.

Check for them by looking in the environment tab

Or running the command: `objects()`

Session 1: Creating objects

Objects:

- Single items (atomic)
- Vectors
 - Access elements of vectors

```
object1 <- 546.32  
object1 <- "orange"
```

```
object1 <- c(1.15, 2.33, 3.84)  
object1[2]
```

Session 1: Creating objects

Objects:

- Single items (atomic)
- Vectors
 - Access elements of vectors
- Data frames
 - Access elements of data frames by row and column

Use this function to define a data frame

Set the first column to be a vector of numbers

```
df1 <- data.frame(col1=c(1,2,3),  
                  col2=c("cat", "dog", "bear"))
```

Second column
a vector of
names

Column
names

col1	col2
1	"cat"
2	"dog"
3	"bear"

Row

Column

df1[3, 2]

- Objects:
 - Single items (atomic)
 - Vectors
 - Access elements of vectors
 - Data frames
 - Access elements of data frames by row and column
 - Matrices
 - Similar to data frames
 - (key difference in practical!)
 - Access elements of matrices in the same way
 - Lists

Objects are of certain types, called classes:

- Character: “a”, “dog”, “orange”
- Integer: 1, 2, 658, -32
- Numeric: 1.2, 3.141592653, -2.1×10^{-4}
- Logical: TRUE/FALSE
- Date: “2015-07-25”

NAs are missing values and can be of any type

Operations:

- +, -, *, /

In built statistical functions

- Mean, median, log, etc

Manipulating data sets

- Extracting certain columns
- Subsetting by value
- Replacing values

Session 1. Finding help

Finding help

- From the console:
 - “?” if you know the name of the function you’re looking for: `?mean`
 - “??” to search for something: `??mean`
- In R Studio:
 - In the “Help” tab
 - Search box on the upper right
- Online:
 - Stackoverflow is a question-and-answer website
 - Lots of error messages explained

Session 1: Practical

- Use the R script called `Practical_P01_1.R`
- Work through it, instructions are in comments:
 - `# this is a comment`
- Enter answers, where it says “Answer:” as a comment
- Save these scripts so you can refer to them later

Session 1: Summary

We've:

- Created objects of different types
- Performed operations on those objects
- Extracted elements of data frames and matrices
- Created new objects by subsetting

Common data file types: .txt .csv (NB: need 'packages' to read in .dta .xls etc.)

Common delimiters: tab (“\t”), space (“ ”), comma (“,”)

Common read in functions: `read.table(...)`, `read.csv(...)`

Some function options:

- delimiter e.g. `sep=" "`
- column names e.g. `header=TRUE`

Example: `mydata <- read.table("mydatafile.txt", header=FALSE, sep="\t")`

Outputting data to check examples:

- All data: `mydata`
- First few lines of data: `head(mydata)`
- Column names: `colnames(mydata)`
- Dimensions of data: `dim(mydata)`

Session 2: Data files

Data read in as data.frame

Access columns via

- number e.g. `important.column <- mydata[,3] # access third column`

Country	Year	Incidence	Deaths
UK	1970	1	0
UK	1980	6	0
UK	1990	34	3
France	1970	32	4
France	1980	17	2
France	1990	12	0
Belgium	1970	5	0
...

Data read in as data.frame

Access columns via

- number e.g. `important.column <- mydata[,3]` # access third column
- names e.g. `incidence <- mydata$Incidence` # access incidence column
`incidence <- mydata[, "Incidence"]`

Country	Year	Incidence	Deaths
UK	1970	1	0
UK	1980	6	0
UK	1990	34	3
France	1970	32	4
France	1980	17	2
France	1990	12	0
Belgium	1970	5	0
...

More on indexing / slicing / subsetting:

Country	Year	Incidence	Deaths
UK	1970	1	0
UK	1980	6	0
UK	1990	34	3
France	1970	32	4
France	1980	17	2
France	1990	12	0
Belgium	1970	5	0
...

1) If you know row and/or column *location*

e.g. `mydata[1,4]` # 1st row 4th column

`mydata$Incidence[5]` # 5th row of 'Incidence'

More on indexing / slicing / subsetting:

Country	Year	Incidence	Deaths
UK	1970	1	0
UK	1980	6	0
UK	1990	34	3
France	1970	32	4
France	1980	17	2
France	1990	12	0
Belgium	1970	5	0
...

1) If you know row and/or column *location*

e.g. `mydata[1,4]` # 1st row 4th column

`mydata$Incidence[5]` # 5th row of 'Incidence'

More on indexing / slicing / subsetting:

Country	Year	Incidence	Deaths
UK	1970	1	0
UK	1980	6	0
UK	1990	34	3
France	1970	32	4
France	1980	17	2
France	1990	12	0
Belgium	1970	5	0
...

1) If you know row and/or column *location*

e.g. `mydata[1,4]` # 1st row 4th column

`mydata$Incidence[5]` # 5th row of 'Incidence'

More on indexing / slicing / subsetting:

Country	Year	Incidence	Deaths
UK	1970	1	0
UK	1980	6	0
UK	1990	34	3
France	1970	32	4
France	1980	17	2
France	1990	12	0
Belgium	1970	5	0
...

1) If you know row and/or column *location*

e.g. `mydata[1,4]` # 1st row 4th column

`mydata$Incidence[5]` # 5th row of 'Incidence'

2) If you know *value* in row and/or column but not location

e.g. `mydata[mydata$Country=="UK",]`
rows with column is equal to 'UK'

`mydata[(mydata$Country=="UK" & (mydata$Year<1985)),]`
UK data prior to 1985

More on indexing / slicing / subsetting:

Country	Year	Incidence	Deaths
UK	1970	1	0
UK	1980	6	0
UK	1990	34	3
France	1970	32	4
France	1980	17	2
France	1990	12	0
Belgium	1970	5	0
...

1) If you know row and/or column *location*

e.g. `mydata[1,4]` # 1st row 4th column

`mydata$Incidence[5]` # 5th row of 'Incidence'

2) If you know *value* in row and/or column but not location

e.g. `mydata[mydata$Country=="UK",]`

rows with column is equal to 'UK'

`mydata[(mydata$Country=="UK" & (mydata$Year<1985)),]`

UK data prior to 1985

More on indexing / slicing / subsetting:

Country	Year	Incidence	Deaths
UK	1970	1	0
UK	1980	6	0
UK	1990	34	3
France	1970	32	4
France	1980	17	2
France	1990	12	0
Belgium	1970	5	0
...

1) If you know row and/or column *location*

e.g. `mydata[1,4]` # 1st row 4th column

`mydata$Incidence[5]` # 5th row of 'Incidence'

2) If you know *value* in row and/or column but not location

e.g. `mydata[mydata$Country=="UK",]`

rows with column is equal to 'UK'

`mydata[(mydata$Country=="UK" & (mydata$Year<1985)),]`
UK data prior to 1985

Over to you ...

Open `Practical_P01_2.R` in Rstudio

GUIDANCE:

Write your answers next to `#Answer:`

Error fixing can depend on you doing two things

Fixing the code **OR** Fixing the data file itself

Session 2: Summary

What we've done:

- read in data files and saved them as data.frames
- stored additional columns in the data.frame
- accessed data from data.frame by *location* and *by value*
- used logical expressions (to find equality and to search data.frame by value)
- encountered (and solved) several common issues with data file reading

Session 3: Plots

Plot an x-y chart: `plot(x=... y=...)`

Add a title by adding an option to the plot: `main=...`

Add labels to the x and y axes: `xlab=...`, `ylab=...`

You find these options (and many more) by checking the help file for the plot you want

You can customise almost any aspect

Session 3: Histogram

Histograms are made using: `hist(...)`

Often the options are the same for different plot types,

- e.g. title: `main=...`

- E.g. labels to the x and y axes: `xlab=...`, `ylab=...`

You find these options (and many more) by checking the help file

Session 3: Multiple plots

Plot multiple charts of any type

Use: `par(mfrow=c(1,2))`

- Multi-Figure ROW-wise
- First number = number of rows
- Second = number of columns

Then run your plotting code

Plot window remains in this layout until you change it

Session 3: Practical

- Practical! (`Practical_P01_3.R`)
- There are some advanced exercises at the end if you get there

Session 3: Summary

We've:

- Made some basic plots
- Learned how to change options to add titles and labels
- Learned how to customise colours
- Practiced adding more than 1 chart
- Exported a figure

Take-away messages from this introduction:

- Comment* everything
- Check *each line of code* is doing the right thing before writing more
- Name variables *sensible* things
e.g. `IncidenceRate <- mydata[, 2]` is better than `A <- mydata[, 2]`
- If you are having a problem, other people will have had it too ... Google / StackOverflow etc. are your friends