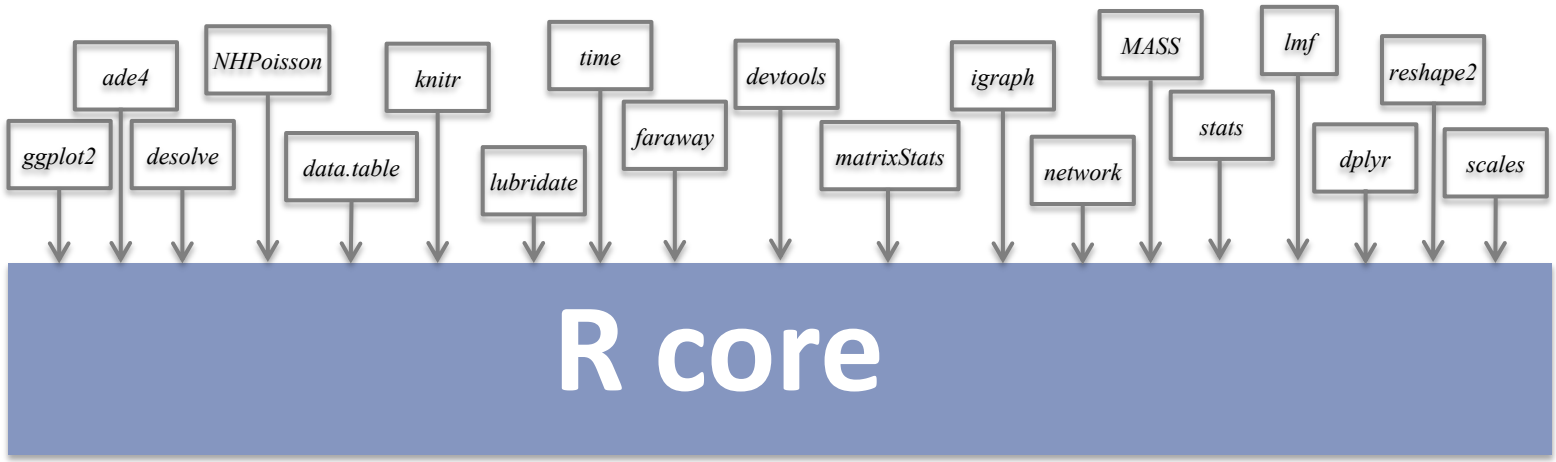
Introduction to R

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

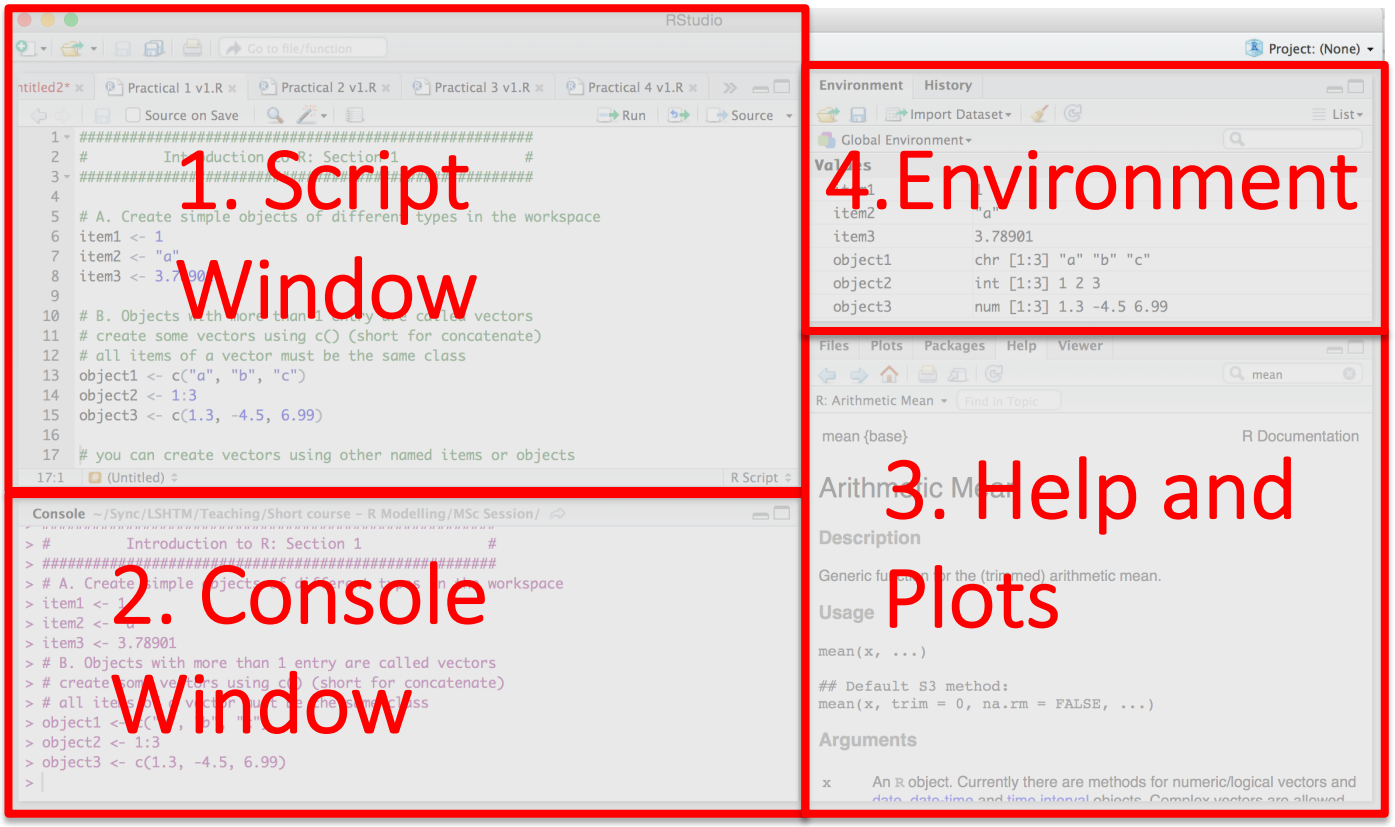
Execute statements for:
- displaying data on a plot
- personalising your plot e.g. colour, axis labels, title

Consider:
- different ways to plot data

Session 1: Set working directory

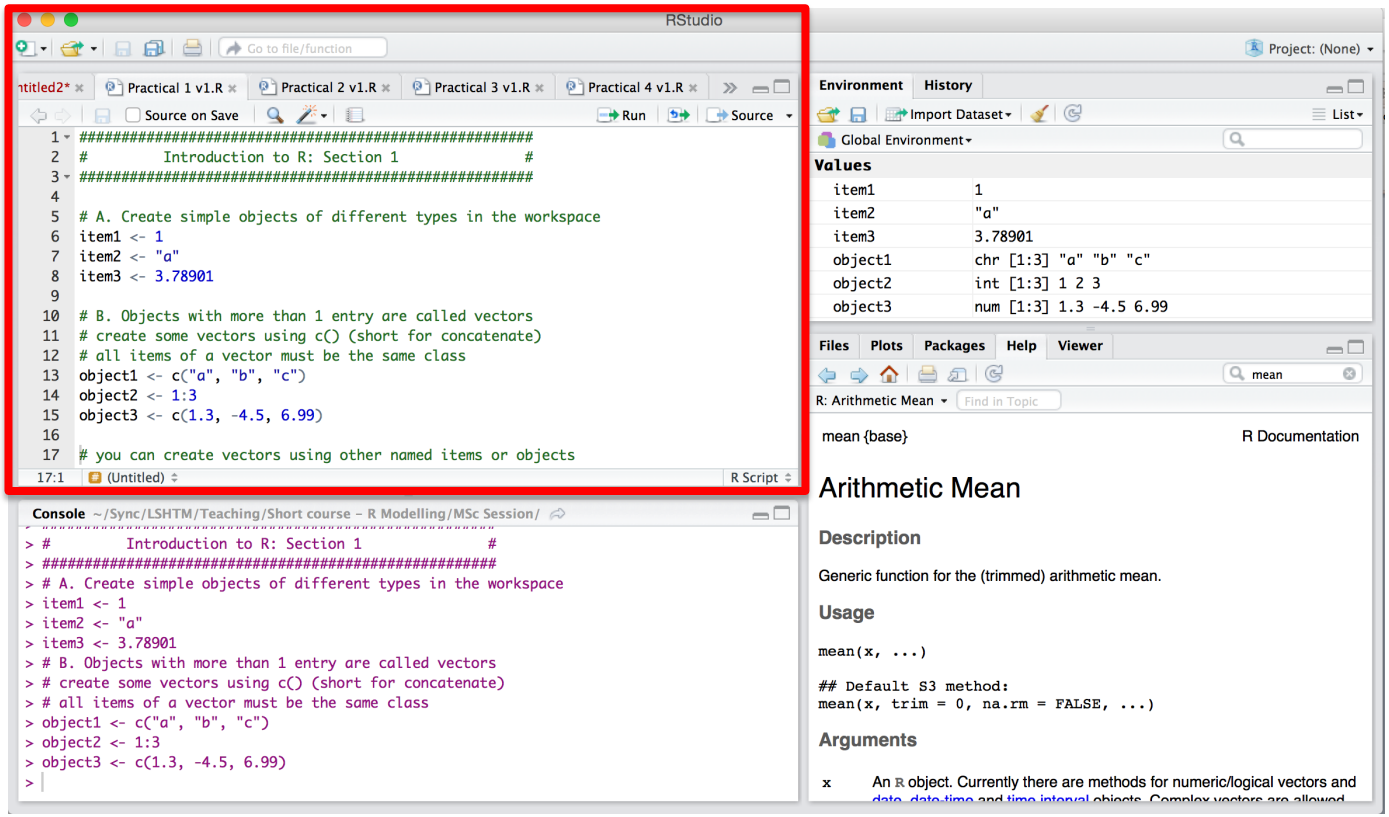
- Open RStudio
- Set your working directory to wherever you saved the folder of practicals:
  - Session → Set working directory → Choose directory
  - Then navigate to the right folder

Session 1: RStudio



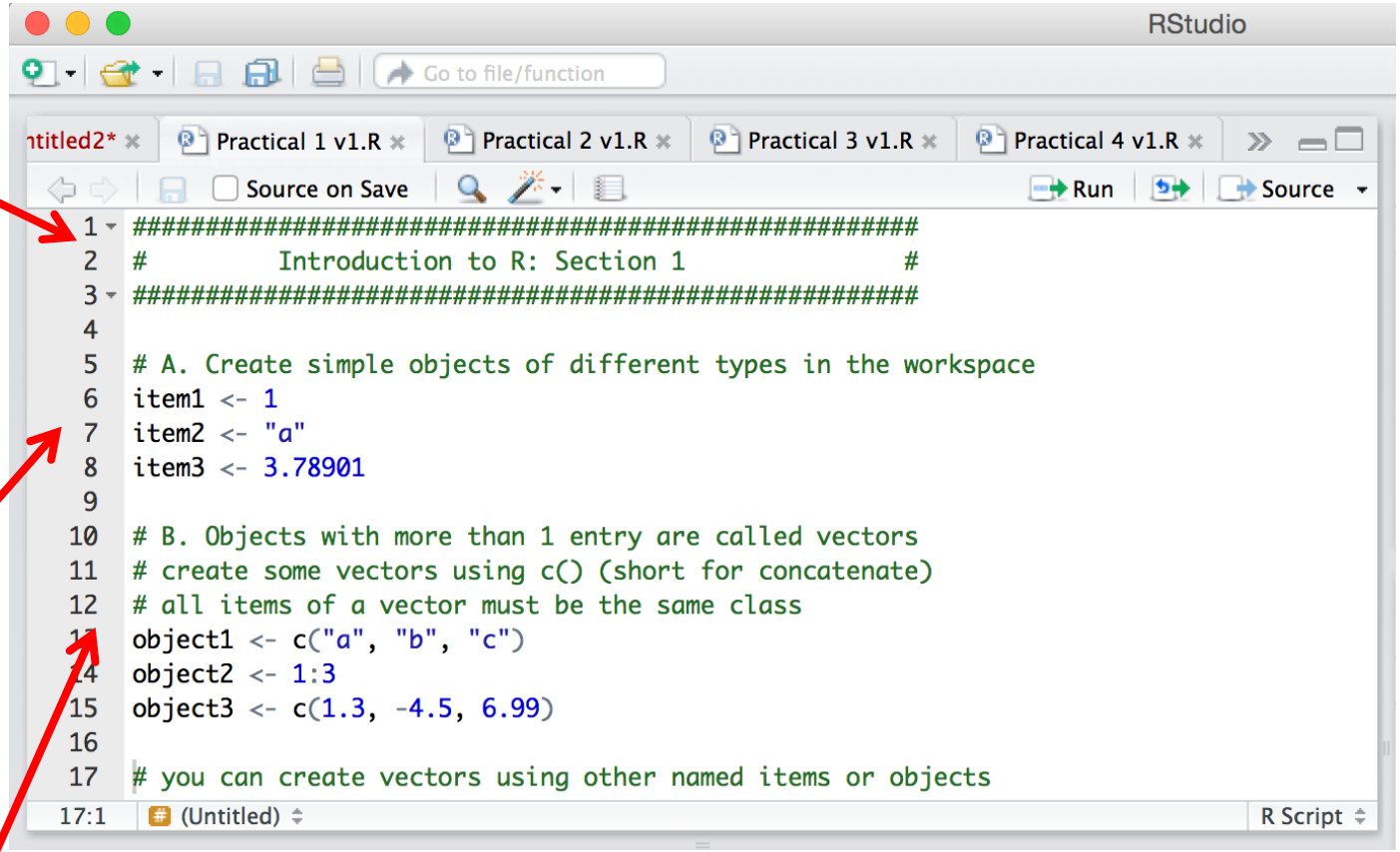
Script window

Can have multiple R files open in tabs



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



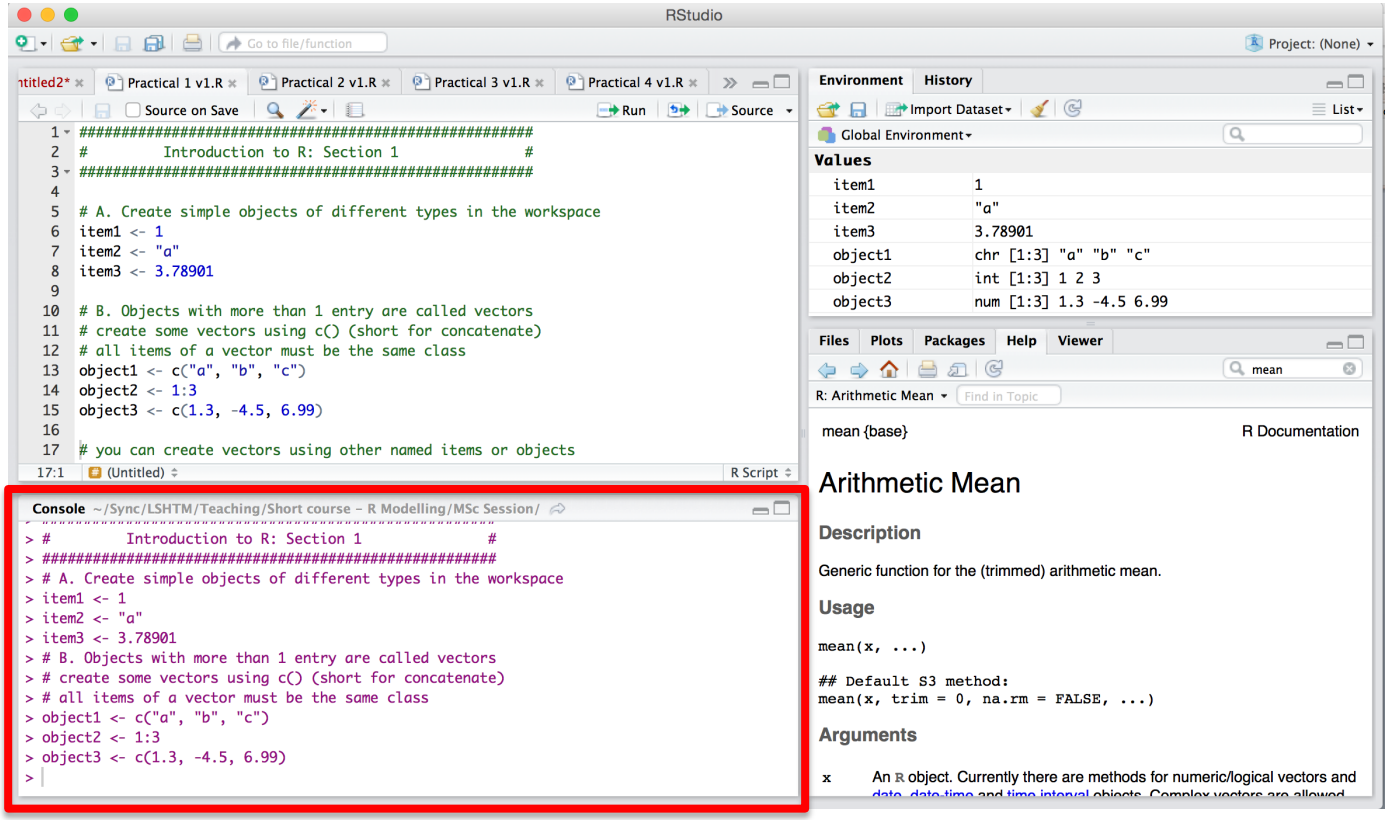
Keep your script tidy:
- Space out sections
- Write useful comments

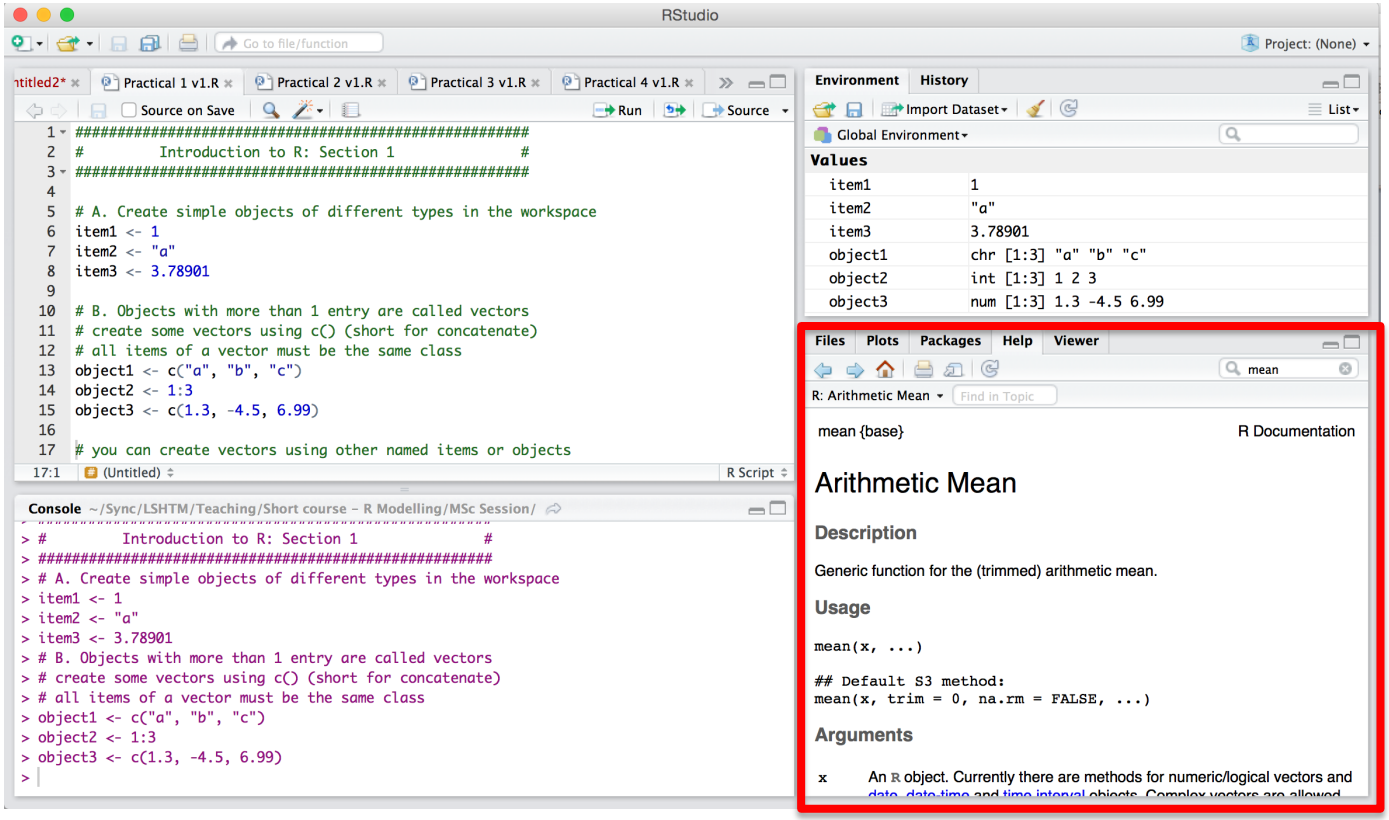
Console window

Where R code from the script window is run

Commands and results in same window

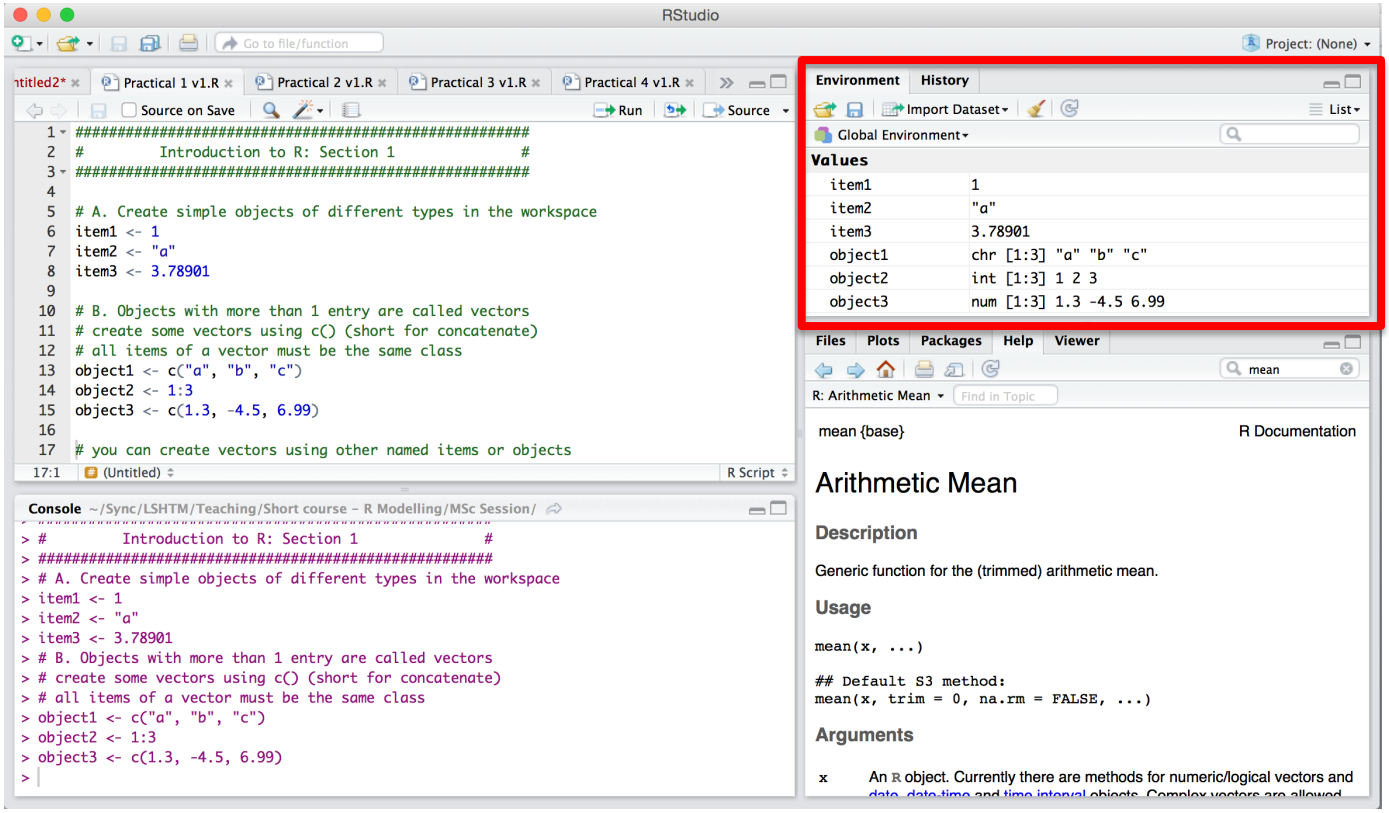
Ctrl+enter (Windows)





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



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]
```

Session 1: Creating objects

- 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

Session 1: Classes of objects

Objects are of certain types, called classes:

- Character: `“a”, “dog”, “orange”`
- Integer: `1, 2, 658, -32`
- Numeric: `1.2, 3.141592653, -2.1x10`$^{-4}$
- Logical: `TRUE/FALSE`
- Date: `“2015-07-25”`

`NAs are missing values and can be of any type`

Session 1: Operations

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

Session 2: Data files

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
…	…	…	…

Session 2: Data files

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
…	…	…	…

Session 2: Data files

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'`

Session 2: Data files

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'`

Session 2: Data files

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'`

Session 2: Data files

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
```

Session 2: Data files

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
```

Session 2: Data files

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`

Session 2: Data files

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

Summary

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