R:
Preparing for your data analysis

Helen Lockstone
hel23@well.ox.ac.uk

iT Centre Learning

iT services

UNIVERSITY OF OXFORD

Schedule

09:30 – 10:15	Introduction and overview presentation
10:15 – 11:00	Guided tutorials
11:00 – 11:15	Tea/coffee break
11.15 – 12:30	Guided tutorials
12.30 – 13:30	Lunch break
13.30 – 15:00	Guided Tutorials
15:00 – 15:15	Tea/coffee break
15:00 – 16:30	Guided Tutorials

Please feel free to ask questions at any time and talk to us during the day about your work and reasons for taking this course

You can leave a little earlier than 16:30 if you need to – the course notes guide you through the material and it can be continued at home

Housekeeping

- Be prepared to exit if the fire alarm sounds
- Water and hot drinks are available
- Toilets are located along the corridor
- The seats are adjustable
- The monitors are adjustable: height, tilt & brightness
- If you have any concern or problem during the day please let me know

About the tutors

- Bioinformaticians at Wellcome Centre for Human Genetics (Helen Lockstone) and Big Data Institute (Punam Amratia)
- Many years experience using R for data analysis (I've used it almost daily for the past 15 years!)
- Have taught introductory R courses to DPhil students and post-doctoral researchers in Medical Sciences Division for a few years now – together with my colleague Ben Wright who has contributed substantially to the course material we are using today

We are grateful to Dave Baker, Gabriele Pani and the administrative staff at IT Services for their support and assistance.

A series of R and Statistics courses are offered by IT Services – currently involved in their coordination and teaching are Andre Python, Punam Amratia, Rohan Arambepola from the Big Data Institute, Helen Lockstone and Ben Wright from the Wellcome Centre for Human Genetics. They were previously taught by Samantha Curle, who recently moved on from Oxford. We are grateful to all of them for discussions, contributions and ideas.

Introductory Remarks

iT Centre Learning

iT services

UNIVERSITY OF OXFORD

What is R?

R is a powerful software package for statistical analysis and also a high-level programming language

Originally written in the 1990s for teaching statistics by Ross Ihaka and Robert Gentleman at the Department of Statistics of the University of Auckland

It's open-source, available for free for Win, Mac, Linux and regularly updated (maintained by R Core development team)
http://www.r-project.org/index.html

Has become a mainstream research tool - users have contributed hundreds of add-on packages (CRAN) and the Bioconductor resource is invaluable for genomic data (https://www.bioconductor.org/)

The downsides to R

R is not an intuitive language to learn, even for those who are familiar with programming concepts

It can take many months to start to feel comfortable writing your own R code, so be prepared for some investment of time, hard work and a steep learning curve

R’s extensive functionality and versatility are its main attributes but also the reason it can be hard to know where to begin…

Tips for Successful Programming

Logical thinking and problem solving skills:

- Decide what steps needed to solve a task
- Troubleshooting error messages
- Test code to ensure it does what it should

Consistency, accuracy and attention to detail:

- Easy to make unintentional mistakes (which R may well execute with no warning message)
- Check what code is doing at every step
- Mentally predict what should happen, so you can spot potential errors

Accept that it can be a frustrating process!

Getting started with R

Some aspects of R are analogous to tasks you might routinely perform in Excel:

- sorting/filtering data
- finding a particular occurrence of text
- simple data summary statistics and tests
- plotting graphs

R provides far more complex functionality besides, particularly for statistical modelling/analysis and data visualisation. It can also perform small useful tasks with a simple line or two of code (where there may not be a quick way to do the equivalent in Excel) e.g. finding the overlap of two lists of genes

Getting started with R

For those without a programming background, the biggest challenge might be getting used to how the R language is structured, and working with objects to store and manipulate data.

Imagine that instead of using Excel, we are giving R step-by-step instructions of what we want to do (e.g. filter a column for all values <0.05):

- Our data could be created in Excel and read into R (like a table)
- It can be stored in an appropriate object (named by the user)
- We'd need to specify the column of interest somehow (R has more than one way to do this) and set the condition 'values less than 0.05'
- Finally we'd have to decide whether to display the output or save it in a new variable

Although some aspects of R can be quite unintuitive and take some time to become familiar with, they provide the flexibility that makes the software so powerful.

Getting started with R

R is extremely extensive and versatile, quite possibly no two people use it in the same way. So how best to get started?'

Hopefully this course will help, and R itself it very well documented with a large and active user community (mailing lists, online forums like stack overflow etc) – usually googling any R problem will find relevant threads.

Learn by doing – follow examples in the manuals or start by running scripts written by others to understand what the code does before moving on to modifying code or writing your own scripts from scratch.

It's impossible to remember the details of all R functions (or even know about all of them!) – make use of the help pages to check syntax, arguments, examples of usage etc

Some general advice

A computer will do exactly what you tell it to do and you need to tell it every little step, in the right order

- Imagine writing instructions that someone can follow to produce a particular result, including every tiny detail they would need
- Akin to instructions for an experimental protocol - important to be very precise

Comment your code as much as possible – if you need to revisit it at a later date, it will make much more sense!

Check and double check (and check again) that your code is doing what you ***think*** it is doing – R has some default behaviours that may not always be realised, and can lead to problems if not spotted

- Inspect the objects created, their length, contents and so on.
- Manually check a few elements of the output to be sure they are correct

Problem solving

You will inevitably run into problems when you are programming - try solving them by:

- reading the error message
- reading the help file
- make a list of what could possibly be wrong and check these possibilities one by one
- break down a complex step into smaller, simpler steps (this is also a useful way to build up a more complex piece of code)

If you really cannot solve a problem, remember there are usually alternative ways in R to achieve the same goal (using a related function for example)
Google the problem – many helpful forums/lists
If all else fails, post a question on the relevant help list (but be sure to read the guidance first!)

R documentation and help

```
> help.start()
```

- Most useful if you don't know which command to use
- Starts a Search Engine in your favourite browser
- Can get this also from the R GUI Help Menu

If you do know which command to use but need to check details

```
> ?plot
```

or equivalently

```
> help(plot) # help("plot") in RStudio
```

Brings up the help page on the function 'plot'

From the R GUI Help menu or the R website, you can get Manuals (in PDF)

An Introduction to R

R Reference manual

R documentation and help

Further reading and resources:

- Michael J Crawley (2005) Statistics: An Introduction using R. Wiley: Chichester, England
  - This is an excellent book, and well worth working through
  - Also has an associated web page with datasets, exercises etc

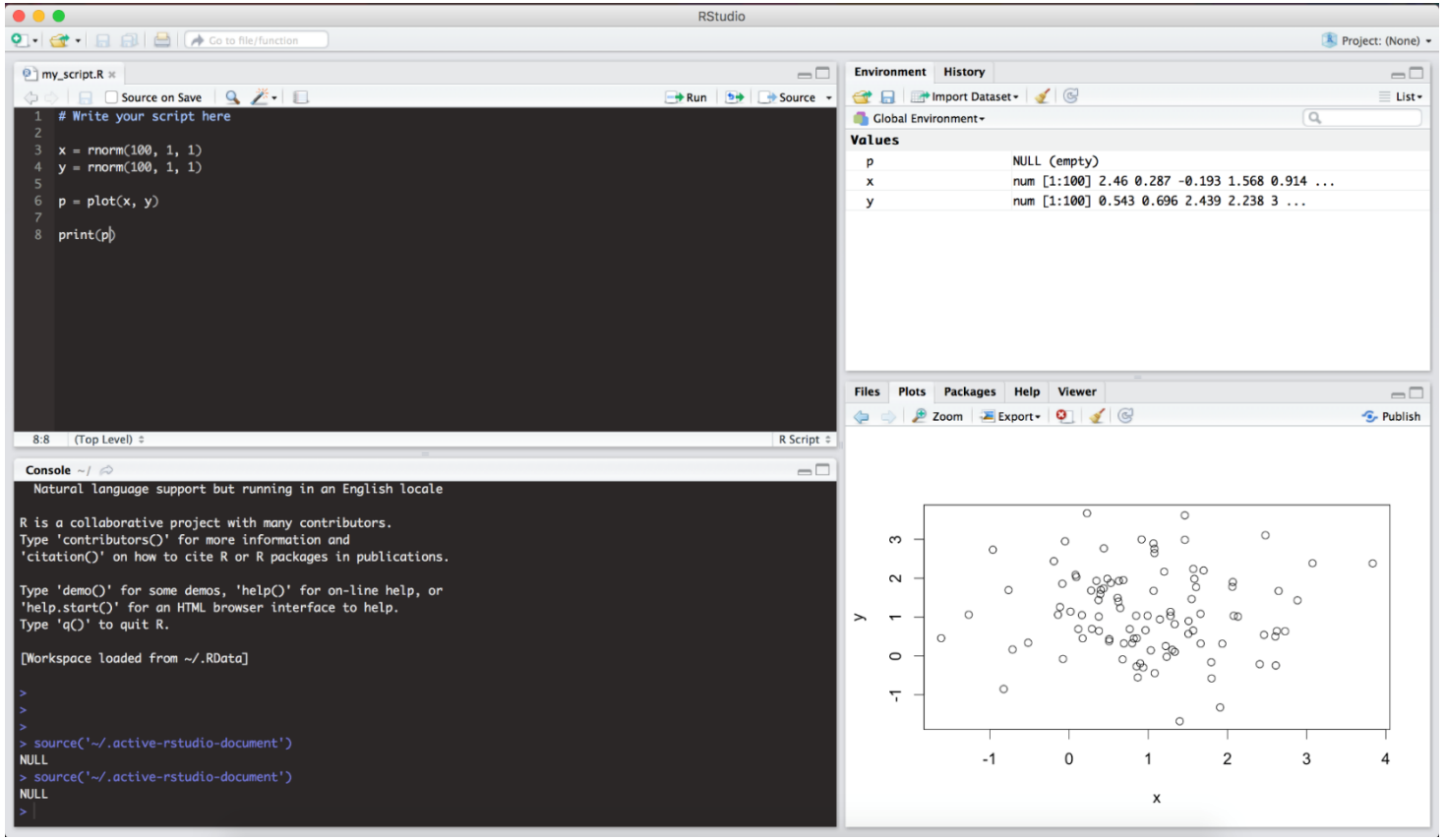
  http://www.imperial.ac.uk/bio/research/crawley/statistics/
- A selection of recommended online resources, tutorials, local courses and textbooks are collated at https://help.it.ox.ac.uk/courses/R
- R-help and Bioconductor mailing lists
- Stack Overflow

The RStudio interface

RStudio®

An interactive and easy-to-use interface with many features that make working with R easier:

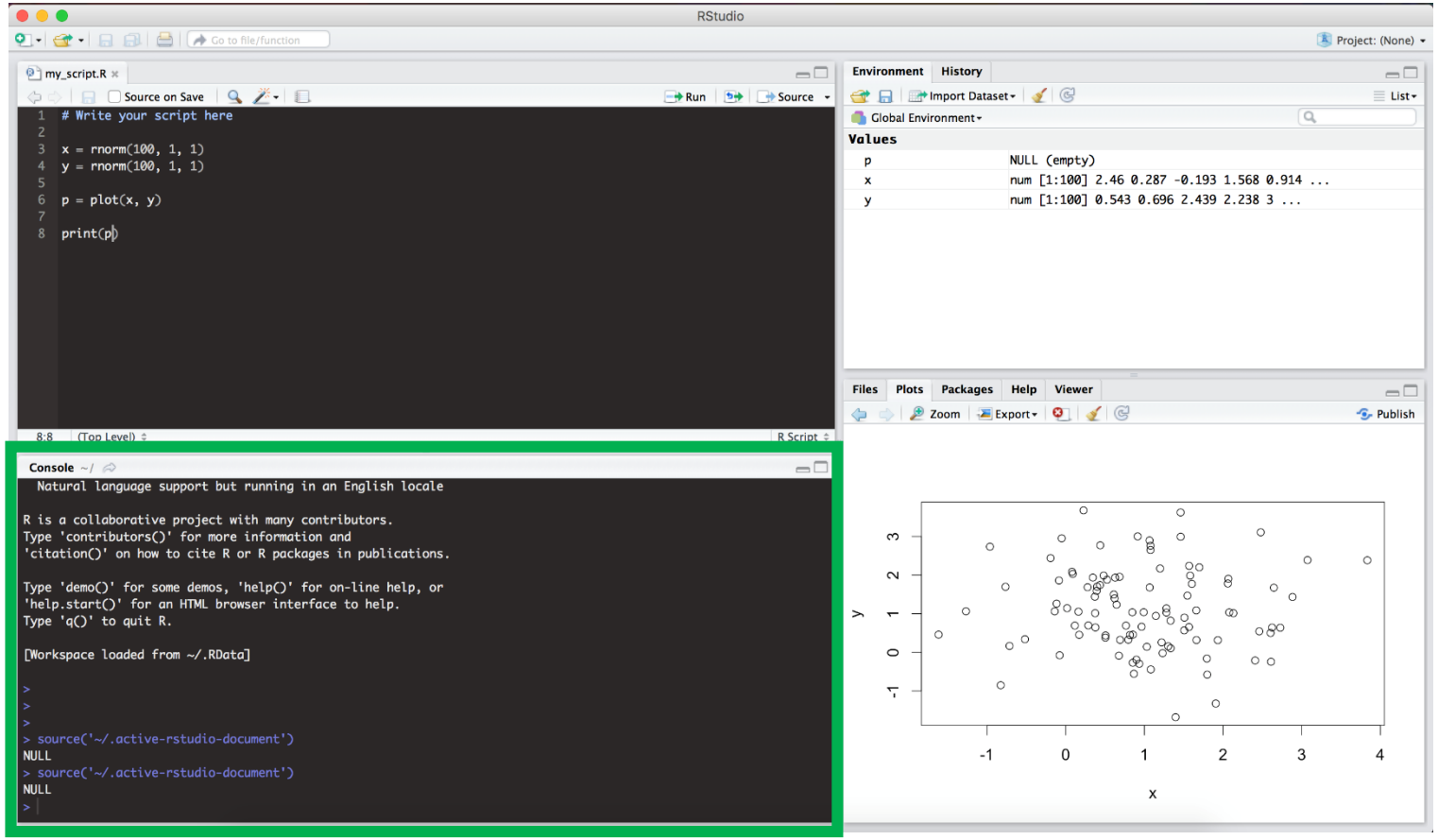
(https://www.rstudio.com)



Slide courtesy of Quentin Ferry

The RStudio interface

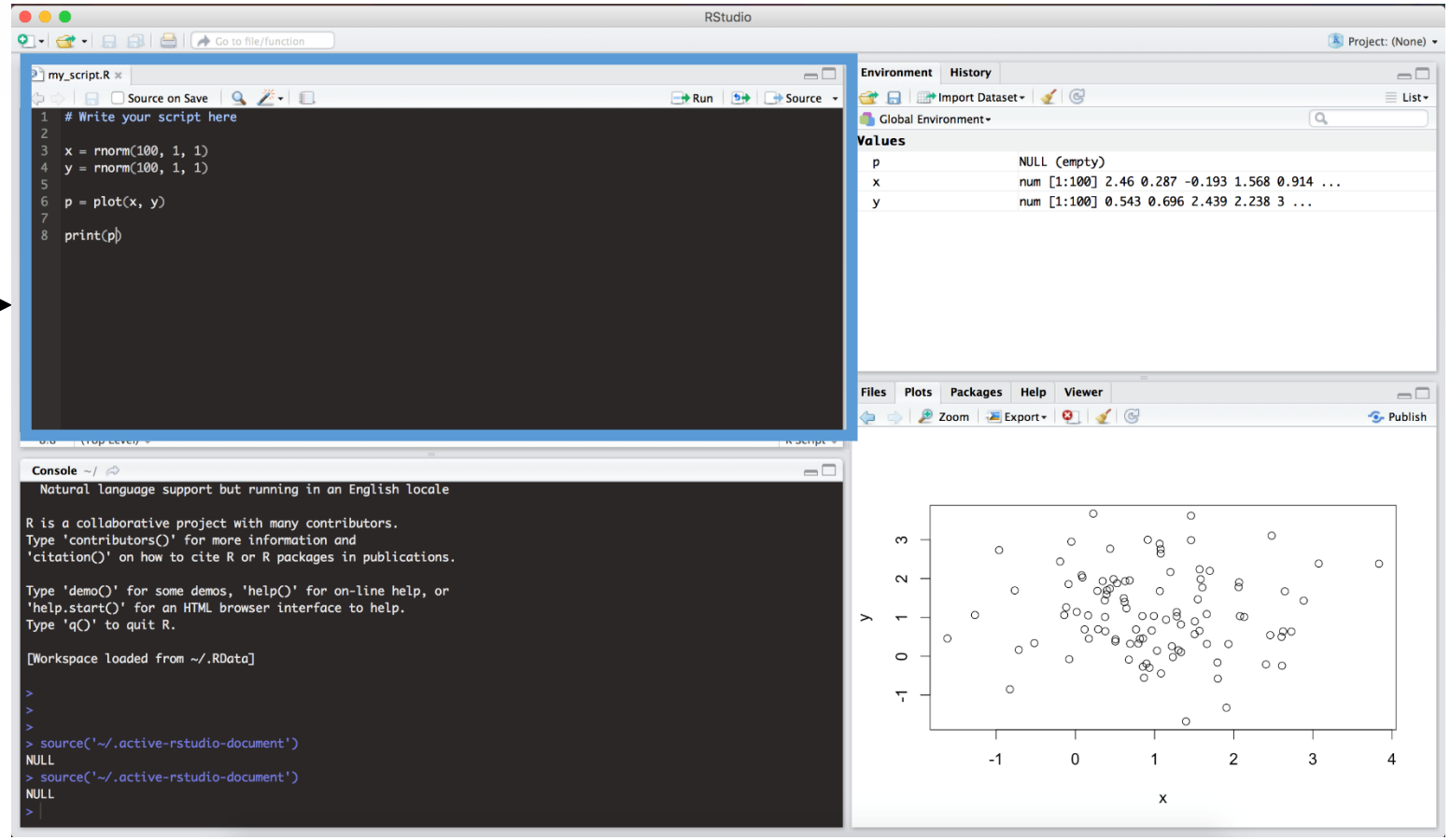
Console for entering
R commands



Slide courtesy of Quentin Ferry

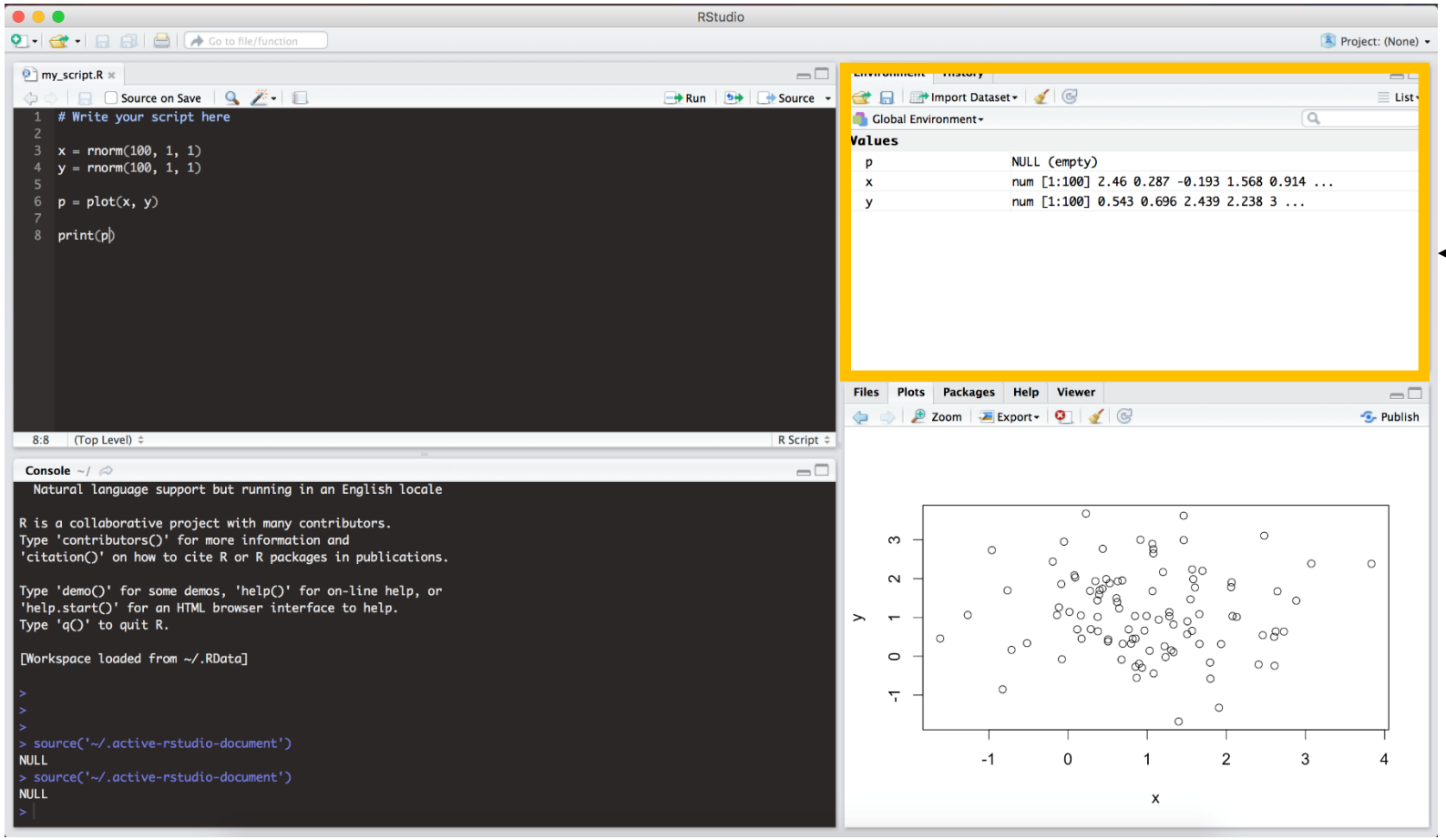
The RStudio interface

Script, data tables etc



Slide courtesy of Quentin Ferry

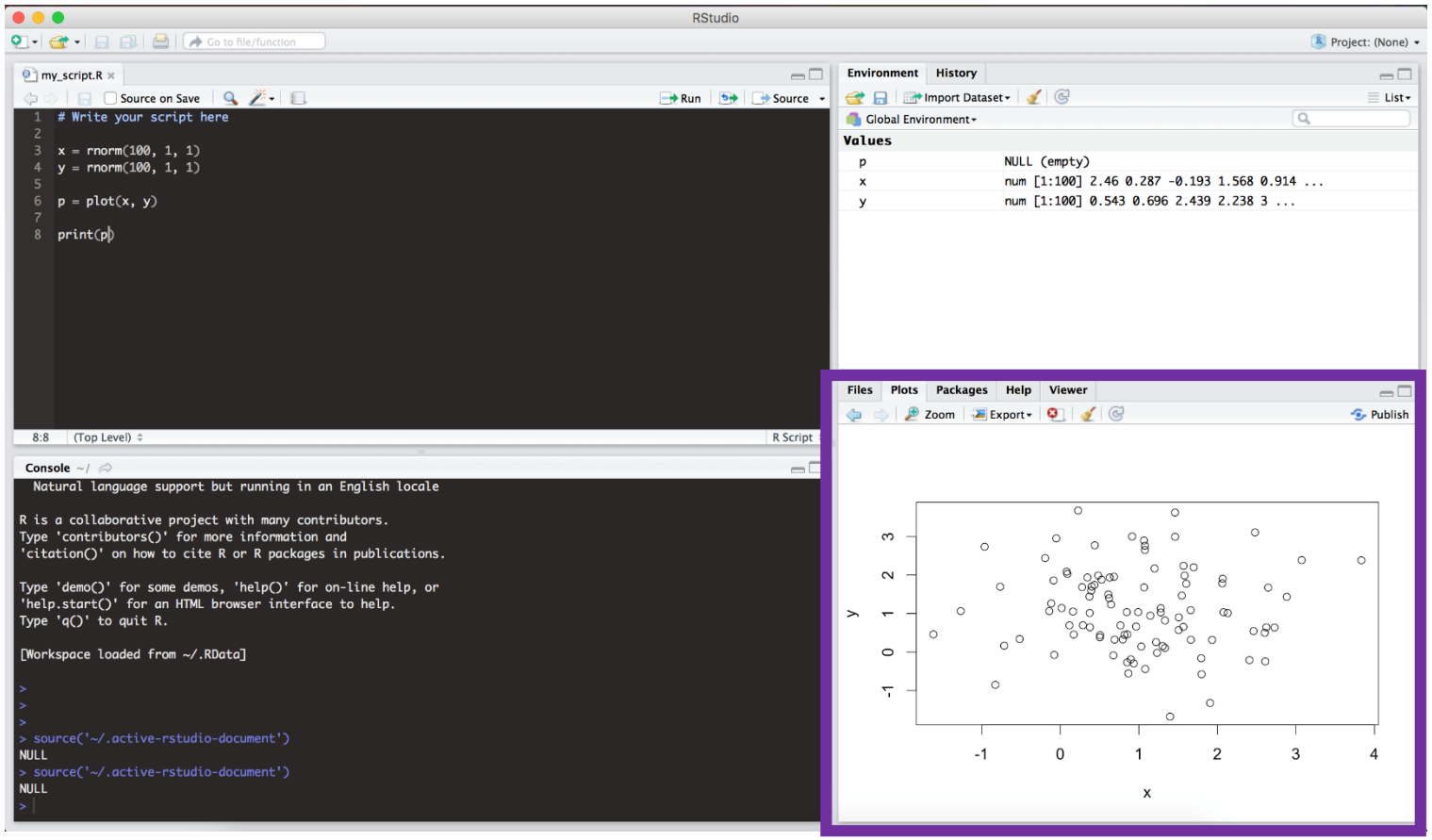
The RStudio interface



Clickable list of objects in memory

Slide courtesy of Quentin Ferry

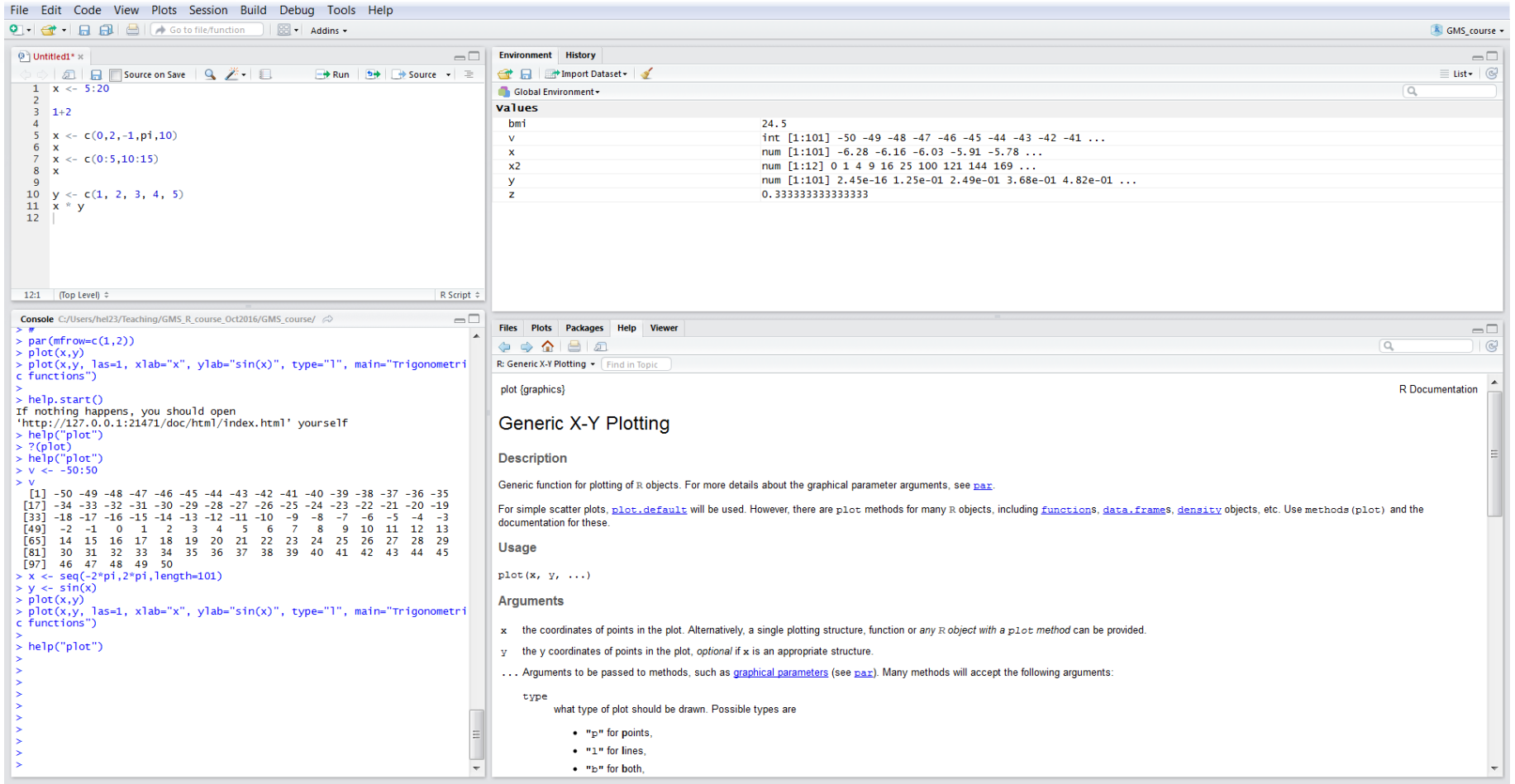
The RStudio interface



Plots, files, packages, help etc…

Slide courtesy of Quentin Ferry

RStudio on Windows

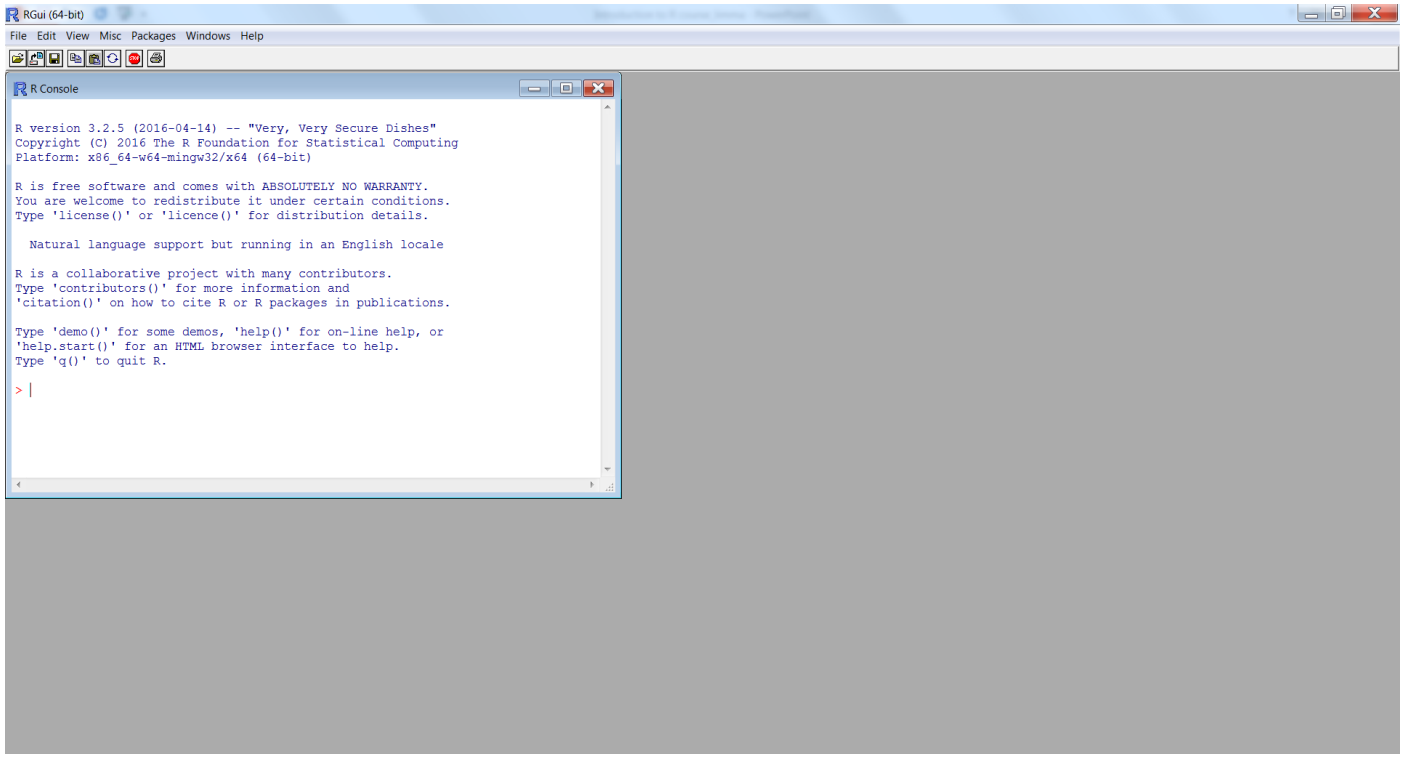


The R Interface on Windows

From here we can start working in R and do thinks like:

- Check what packages are already installed
- Install new packages
- Check the current working directory or change to a new one
- Access the help options

As always, there are several ways to do things



Getting Started with R

iT Centre Learning

iT services

UNIVERSITY OF OXFORD

R basics: notation

Throughout these slides, text in red denotes commands that you can type in your R session (it is good to type them directly rather than copy and paste to get used to syntax of commands and expressions, at least to begin with)
Note that commands are shown including the prompt sign > but ignore this when entering the command. For example:

```
> x <- 5
```

should be entered at the command line as:

```
x <- 5
```

An extra > as well as the in-built R prompt will give an error:

```
> > x <- 5
Error: unexpected '>' in ">"
```

Commands are shown in red, and output in blue (this is the convention in the standard Windows R GUI but differs in other interfaces)

R basics: command line

Getting used to the command line (in red):

Prompt [Whitespace] Command [Whitespace] Return

```
>                    1+2
[1] 3
```

Results are output underneath (in blue). Elements of the output are indexed by the square brackets

- The prompt tells you that R is waiting for you to enter a command – type a valid command and press return or enter
- Some output may be printed to the console (as above) or a new variable may be created for example
- The prompt symbol will be displayed again afterwards
- Sometimes a + appears instead of the usual > prompt symbol. This means the previous line was incomplete while a command that is invalid will generate an error message

R basics: command syntax

```
> x <- c(1:5,8,9)
```
this creates a vector named 'x' containing some numeric values

If we forget the closing bracket, a + sign indicates the command is incomplete:

```
> x <- c(1:5,8,9
+ )
>
```

- Some text editor programs highlight different types of syntax in different colours or automatically close brackets and quotation marks to help eliminate typing mistakes
- If we can't simply continue our command, use Esc or control-C to return to the prompt and start again
- There is also a useful command recall option – you can use the up/down arrows to scroll through previously entered commands, which can be edited or re-run to save typing again

R basics: assignment

The following command assigns the value 1 to a new variable we create and name ‘x’. The assignment operator is <-

```
> x <- 1

## Inspect the contents of the new variable named x
> x
[1] 1
## we can also use = as the assignment operator and write the command as:
> x = 1
```

Note that the contents of any variable will be overwritten if later assigned something else:

```
> x <- 4
> x
[1] 4
```

R basics: naming variables

R is case sensitive so x and X are different variables:

```
> X
Error: object "X" not found
```

Variable names are chosen by the programmer and should start with a letter followed by letters and numbers – informative names are helpful for several reasons.
You can use . or _ to separate parts of a variable name but not whitespace since this is used to detect the end of a token ('word'). A variable named data.norm or d.norm or dataNorm is fine, but trying to assign a value to a variable named 'data norm' will give an error:

```
Error: unexpected symbol in "data norm"
```

Elsewhere R ignores whitespace so the following commands are equivalent:

```
> x<-4+3
> x <- 4 + 3
```

Data Types and Structures in R

Common data structures (object types)

To do anything useful in R, we need to use **objects** to hold data or information and perform various operations on them. The terms ‘object’, ‘variable’ and ‘data structure’ can all refer generally to objects created in R.

There are several different data structures in R including:

- Vectors
- Factors
- Matrices
- Dataframes
- Lists

Common data structures (object types)

Objects can be created in many different ways and hold different kinds of information. Unlike other programming languages, there is no need to initialise a variable or object in R (define it before first use) – it can simply be created and used directly

```
> x <- 1:10
> x*2
[1]  2  4  6  8 10 12 14 16 18 20
```

We'll work through some examples of each type and look at ways to access or manipulate the data contained within an object. Be aware that the type (class) of an object and data it contains (numeric, character etc) can affect how it is treated by R.

R basics: vectors

One-dimensional objects with length attribute:

```
> x <- 1:10
```

1	2	3	4	5	6	7	8	9	10

```
> x <- c(1:5, 10:14)
```

1	2	3	4	5	10	11	12	13	14

```
> length(x)
[1] 10
> x[3]; x[7]; x[1:5]
[1] 3
[1] 11
[1] 1 2 3 4 5
```

R basics: vectors

Elements automatically added to create vector of appropriate length

```
> x <- 1:15
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Can contain numeric or character data

```
> class(x)
[1] "integer"
```

```
> x <- c("a", "b", "c")
```

a	b	c

```
> class(x)
[1] "character"
```

R basics: factors

Factors are useful for handling categorical data

```
> groups <- rep(c("WT", "MU"), each=3)
> groups
> class(groups) # character vector
> groups <- as.factor(groups) # coerce to factor
> class(groups) # factor
> groups
[1] WT WT WT MU MU MU
Levels: MU WT

> table(groups) # useful summary function giving the number
of entries for each level of the factor

> groups <- factor(groups, levels=c("WT", "MU")) # re-order
the levels (default alphabetical)
```

R basics: matrices

Two-dimensional object with row and column number attributes

```
> x <- 1:20
> y <- matrix(x, ncol=2, nrow=10)
> dim(y) # get the dimensions
[1] 10  2
> head(y) # shows the first 6 rows
> colnames(y) <- c("Col_1", "Col_2")
> y
```

All columns of a matrix must contain data of the same type. A dataframe can hold a mixture of types – very useful for experimental data with associated descriptive information.

Accessing elements of an object

Square brackets are used to refer to specific elements or subsets of a vector, factor, matrix or dataframe

```
> x <- 1:20 # vector again
> x[5] # print 5th element of x to screen
[1] 5
```

For a 2 dimensional object need two indices separated by a comma [row, column]

```
> y[2,1] # specific element – second row, column 1
> y[1,] # all of first row
> y[1:5,1] # first 5 entries in column 1
```

Setting the working directory

There are many useful functions in R for manipulating data stored in vectors, matrices or dataframes - best introduced through practical exercises of the online tutorial

One important concept before we get started is the **working directory**. If we want to read in data from existing files or create new ones to save any plots or analysis results, R needs to know where to find/save them.

```
> getwd() # tells us the current working directory
> setwd() # allows us to set a different working directory
```

For the latter, a path to the directory needs to be provided in quotes within the brackets. The path can be full or relative:

```
> setwd("C:/Users/jbloggs/data")
> setwd("./data") # assuming the current working directory is
C:/Users/jbloggs (a level above the 'data' directory)
```

Setting the working directory

The working directory can also be set and changed via the menu bar:

- Using the R console you find ‘Change dir…’ under the File menu
- In RStudio, the Session menu has an option ‘Set working directory’.

From these you can navigate to the desired directory and select OK to set it as the working directory. Note that this happens behind the scenes and nothing will appear to happen on your screen - you can use getwd() to confirm if you have set it as intended.

Now we are ready to start using R!