

Tutorial Pack 1

(To be completed during LW1 tutorial)

(90 minutes)

LEARNING OBJECTIVES

- **To build up an understanding of the R Statistical Programming Language**

LEARNING OUTCOMES

- **By the end of this tutorial students will have;**
 - **Familiarised themselves with the R programming environment.**
 - **Used different assignment, mathematical and logical operators.**
 - **Practiced using some basic commands and functions in R.**
 - **Developed experience of using matrices**

RESOURCES AND TOOLS REQUIRED

- **R or RStudio**

In today's session we are going to go through the basics of using the R programming language.

R is a very powerful open-source statistical package that can be used to analyse data from a variety of sources and in different formats. We will use R in future tutorials to analyse web traffic data and conduct hypothesis testing.

You can install R for free and use it with no restrictions. Both R and RStudio are available through appsanywhere.westminster.ac.uk.

IMPORTANT:

The pack is designed for you to go at your own speed and gets progressively more difficult. At the end of each section there is a series of practice questions – you should attempt to answer **all** questions.

Any questions you do not complete today should be completed before your next tutorial.

Study Notes

Upon opening R you are greeted with the R console. The cursor indicates where commands can be entered. You will need to press enter after entering a command to submit it for processing.

```
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.  
  
Natural language support but running in an English locale  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
> |
```

Before we go any further, it's a good idea to try a few basic R commands. Let's begin by using the `sqrt` (or square root function). Enter the command `sqrt(x)` into the R console, where `x` is the value you want to square root. When you press enter R will print the result back to the console.

```
> sqrt(16)  
[1] 4  
> |
```

We can also do simple **mathematical operations**, like add, subtract, multiply, divide and raise a number to a particular power. In the screenshot below I've used `^` to raise a number to the power of another, in this case 2^2 or $2 \times 2 = 4$. Other operators include the modulus operator `%%`.

```
> sqrt(16)  
[1] 4  
> 4-5+8*2  
[1] 15  
> 2^2  
[1] 4  
> 10/2  
[1] 5  
> |
```

Questions

Using R, enter the appropriate commands to answer the following questions

Qnum	Question	ANSWER
1	Calculate 155.2 multiplied by 16	
2	18 divided by 12	
3	11.2 subtracted from 186	
4	The square root of 2	
5	The remainder of 18 when divided into 100	
6	R distinguishes between input commands and the corresponding output using colours. In the screenshots above what colour is used to represent the program output?	

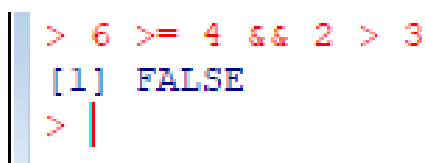
Relational operators allow us to compare values or variables. In other programming systems they are also known as comparison operators. The result of a command using relational operators is either true or false.

The relational operators in R include less than (<), greater than (>), less than equal to (<=) , greater than equal to (>=), equal to (==) and not equal to (!=). In the example below I show how to use a greater than equal to relational operator to compare 6 with 4. In this case the result is TRUE.



```
>
> 6 >= 4
[1] TRUE
> |
```

The **logical operators** && (AND) and | | (OR) can be used to chain multiple relational operators together. In this way we can determine whether a set of comparisons are true or at least one of them is. In the example below I am testing whether the following two conditions are true.



```
> 6 >= 4 && 2 > 3
[1] FALSE
> |
```

Questions

Using R, enter the appropriate commands to answer the following questions and verify the result

Qnum	Question	ANSWER
7	Verify whether 14 is greater than 4	
8	Verify whether 9 is less than 8	
9	Verify whether 2 is equal to 7	
10	Verify whether 3.15 is less than or equal to 5	
11	Verify whether 11 is greater than or equal to 10.2	
12	Verify 9 is greater than 4 and at the same time 37 is less than 40	
13	Verify 81 is equal to 4 or 56 is less than 47	

In the context of programming, **variables** can be thought of as data containers or references to items of data. It is good programming practice to give names to variables that clearly and concisely indicate what the data they reference represents.

Suppose we wanted to create a variable to store the number of hits to a specific webpage. In this case we could create a variable called “hitsPerPage” or perhaps “hits_per_page”

For a variable to be valid, its name should follow the following rules.

- It should contain letters, numbers, and only dot (.) or underscore characters (_).
- It should **not** start with a number (e.g:- 1stPage)
- It should **not** start with a dot followed by a number (eg:- .PageHits)
- It should **not** start with an underscore (eg:- _PageId)
- It should **not** be a reserved keyword (see below for examples)
 - if else repeat while function for in next break
 - TRUE FALSE NULL Inf NaN NA
 - NA_integer_ NA_real_ NA_complex_ NA_character_

What are reserved keywords?

Reserved keywords are words or phrases that have a predefined meaning within a given programming language. Consequently, they cannot be used by the programmer to name variables.

Questions

Using the rules set out above, answer the following questions.

Qnum	Question	ANSWER
14	Choose a variable name to represent the average bounce rate across all pages on a website	
15	Choose a variable name to store the total number of clicks on the add to cart button	
16	Why might the variable name “_total_pageviews” not be appropriate to use in R	
17	Why might the variable name “x” not be appropriate to store the average amount of time spent browsing a page	

Variables in R can be created and modified using one of the **assignment operators**. The assignment operators within R include, “=”, “<-” and “<<-”.

In the screenshot below I show an example of how to use each of these assignment operators to create and modify the variable I have labelled “x” by setting its value to 5. This technique is known as **left-hand assignment (LHA)**, since the variable we are interested in assigning a value appears on the left-hand side.

```
> x = 5
> x <- 5
> x <<- 5
>
```

In contrast to many other programming languages, R also allows variables to be assigned **using right-hand assignment (RHA)** when using the “->” or “->>” operators. In this case the variable to be assigned a value appears on the right-hand side of the operator. Generally, you should favour using LH assignment because not all R functions support RH assignment, but it is good to be aware of other approaches.

```
> 5 -> x
> 5 ->> x
```

Whenever we are unsure about the current value of a variable, we can enter its name on a line of its own and press enter. This will cause R to print out its current value.

```
> x
[1] 5
>
```

Questions

Using your knowledge of variable assignment, answer the following questions.

Qnum	Question	ANSWER
18	Create a variable to store the daily page views of a website and assign it the value 16,219 using LH assignment	
19	Create a variable to store the average visit duration of a website and assign it the value 18.82 seconds using RH assignment	
20	Create a new variable and assign it the value 2 using RH assignment. Print out the current value of the variable and then reassign it the value of 4 using LH assignment. Verify that the variable's value has been updated.	

In addition to its name, each variable created in R is associated with an **object type**. The object types available in R include: the numeric type, the integer type, the factor type, and the character (string) type. The type associated with the variable can have an impact on how it can be manipulated in the remainder of the R program.

In the majority of cases R will infer the object type that should be associated with a variable when we assign it a value. So far, we have only created variables storing numeric measurements hence they will have been assigned the **numeric** type. We can verify this by using the `class()` function as shown below.

```
> class(x)
[1] "numeric"
> |
```

Integers on the other hand can be created by appending "L" to a variable's value or by using a built-in function (`as.integer`) that converts a literal value into an integer type.

```
> x <- as.integer(5)
> class(x)
[1] "integer"
> y <- 5L
> class(y)
[1] "integer"
> |
```

The **factor** object type can be used when we want to store ordinal or categorical/nominal data. For example, if we were recording the names of pages that a user had viewed then the list of possible pages could be represented by a factor type.

Factor variables are created using the `factor()` function.

```
> site_pages <- factor(c("home", "checkout", "search", "news"))
> site_pages
[1] home      checkout search    news
Levels: checkout home news search
```

```
> class(site_pages)
[1] "factor"
```

In the example above you will also notice I have used the `c()` or concatenate function to create my factor variable “site_pages”. This is because unlike the other object types, factors contain multiple values and hence we use the concatenate (or join) function to group all the possible values we want to use for our factors into a single vector before we can use the factor function.

Characters (non-numeric data or data with a non-numeric interpretation) on the other hand can be assigned to a variable by using quotation marks. For example;

```
> msg <- "A piece of text"
> class(msg)
[1] "character"
> |
```

Questions

Using your knowledge of object types in R, answer the following questions.

Qnum	Question	ANSWER
21	Create a new variable to store visits to a given page. Ensure that this variable is assigned the integer value 372	
22	The referrer field within the web log file contains the name and version of a user’s browser. What R object type do you think would be most appropriate to store this type of data?	
23	The cookie field within the web log file contains a unique session id assigned to each user that access the site. It is made up of a mixture of letters and numbers. Which object type do you think is most appropriate to store this kind of data in R.	
24	Suppose we operate a website that attracts visitors from several countries. Create a variable to store the continent associated with each user of our site using an appropriate R object type. Verify using the <code>class()</code> function that the correct object type has been used.	

Recall that **vectors**, in the context of computer programming, represent a collection of values. Often you will see them referred to as one-dimensional arrays or lists. Previously we created a vector to create our `site_pages` variable as a factor object type. To do this we used the `c()` or concatenate function but note there are several other ways to create vectors.

The `seq()` or sequence function creates a vector of values between an upper and lower range. By default, the `seq()` function will produce values in 1 unit increments but it is also possible to specify the increments manually using the “by” parameter (as show below)

```
> seq(1,10)
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1,10,by=0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0
> |
```

The `rep()` or repeat function creates a vector of values that are repeated a set number of times. In the example below the vector produced contains the letter “A” five times.

```
> rep("A",5)
[1] "A" "A" "A" "A" "A"
```

Vectors can also be combined to form larger vectors, in this case we use the `c()` or concatenate function to join two or more vectors together.

```
> c(rep("A",5),rep("B",4))
[1] "A" "A" "A" "A" "A" "B" "B" "B" "B"
```

Sometimes it can be useful to access a specific element in a vector. To do this we can use indexing. In this case elements are numbered from 1...N. In the example below I have printed out the 6th element in the vector created by joining two vectors together.

```
> my_vector <- c(rep("A",5),rep("B",4))
> my_vector
[1] "A" "A" "A" "A" "A" "B" "B" "B" "B"
> my_vector[6]
[1] "B"
```

Questions

Using your knowledge of vectors, answer the following questions.

Qnum	Question	ANSWER
25	Create a vector containing the whole numbers 15 through 25	
26	Create a vector containing numbers between 1 and 5 in 0.25 increments.	
27	Create a vector containing the following data “PageA”, “PageB”, “PageC” and “PageD”	
28	Create a vector that contains the whole numbers 2 through 8 and 12 through 18	

29	Create a vector that contains the number 9 repeated 6 times	
30	Create a vector that contains the letter "A" repeated 5 times and then modify your vector so that the last value is changed to "B"	

Many **mathematical functions** within R, such as min(), max(), mean(), median(), sd(), quantile() and IQR(), accept vectors as parameters. In the screenshot below I demonstrate how to create a vector containing five numbers before calculating the minimum value and the 10th percentile.

```
> x <- c(1,2,3,4,5)
> min(x)
[1] 1
> quantile(x, 0.1)
10%
1.4
```

Questions

Suppose the amount of time each user spent waiting for their credit card details to be processed by a website was randomly recorded and consisted of the following data.

waiting time in seconds = 1.3, 10, 3, 11.2, 18.2, 5.1, 7.4, 7.2, 41.2, 90.8

Use your knowledge of vectors and mathematical functions to answer the following.

Qnum	Question	ANSWER
31	Calculate the average time users spend waiting for their credit card to be processed	
32	Calculate the median time users spend waiting for their credit card to be processed	
33	Calculate the maximum waiting time that 90% of users experience	
34	Assuming that outliers can be detected using the 1.5 Interquartile range rule (IQR), determine if there are any outliers in the above dataset.	
35	Suppose the waiting time of the 10 th user was entered incorrectly. Amend your dataset so that the waiting time for the 10 th user is changed to 9.8 seconds.	
36	As a result of carrying out Qnum35, what is the new maximum waiting time experienced by users?	
37	The owner of the website will not tolerate poor performance and will likely switch to a different payment processor if the variability in payment processing times is greater than 10 seconds. Determine whether, based on this sample data, the owner should switch to a different payment processor.	

A **matrix** in R represents a two-dimensional structure containing collections of columns and rows. Each row in a matrix might represent an observation, whilst the columns represent attributes associated with a given row.

The `matrix()` function can be used to create a new matrix. The matrix function requires a series of parameters to be specified, including the name of the vector that contains the input data, the number of rows and columns to use and a list containing the row and column names that are to be used. An example of how to create a simple 2x2 matrix is shown below.

```
> rows <- c(1, 2, 3, 4)
> colnames <- c("columnone", "columntwo")
> rownames <- c("rowone", "rowtwo")
> matrix(rows, nrow=2, ncol=2, dimname=list(rownames, colnames))
      columnone columntwo
rowone         1         3
rowtwo         2         4
> |
```

What is a list?

A list is a special type of vector used in R that can contain mixed object types. Lists can also contain elements that refer to other lists, in this way it is possible to create lists of lists. Lists are created using the `list()` function.

```
> class(list(rownames, colnames))
[1] "list"
> |
```

Questions

Using your knowledge of matrices in R, answer the following questions.

Qnum	Question	ANSWER
38	Modify the example above so that the row names are changed to "Home Page" and "Checkout Page"	
39	Modify the example above so that the column names are changed to "UK Visitors" and "CN Visitors"	
40	Modify the example above so that the following data is stored in the matrix. UK Visitors to Home Page = 10, UK Visitors to Checkout Page = 20, CN Visitors to Home Page = 25, CN Visitors to Checkout Page = 50.	