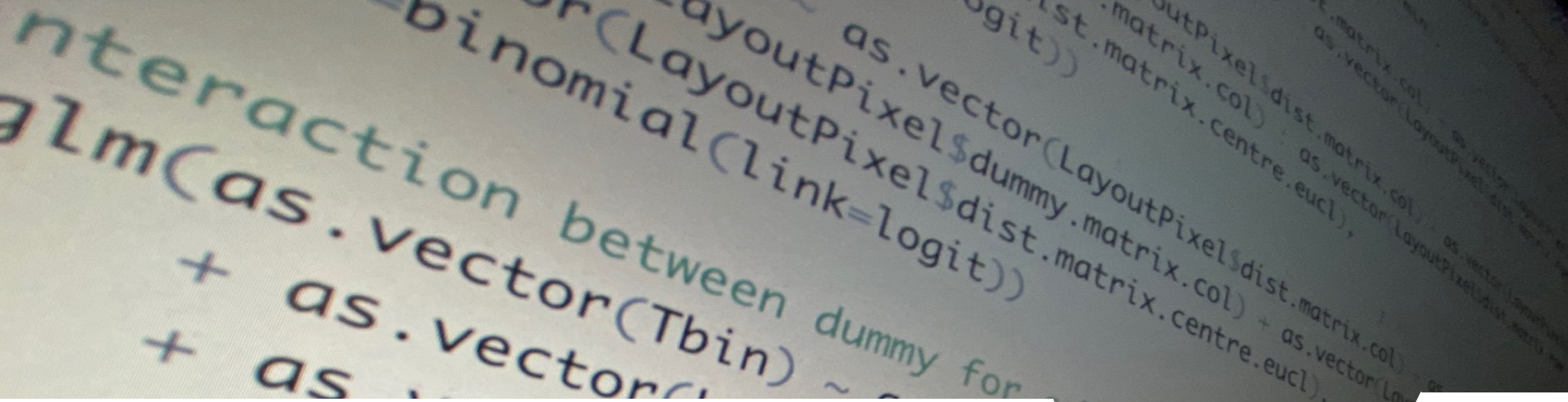


ST 117

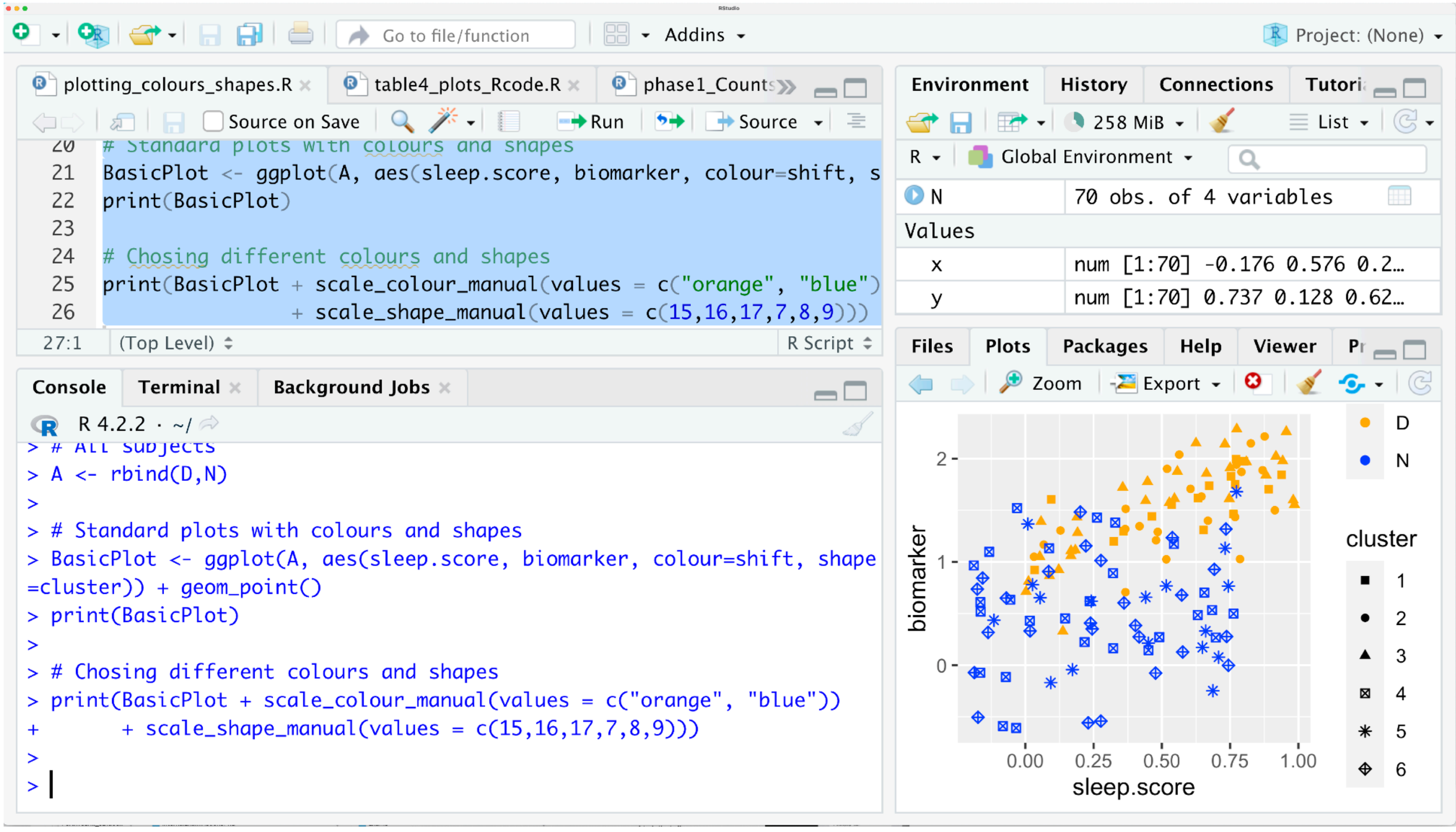
2. Basic R

WARWICK

Working environment
Getting help
Small exercises

Integrated development environment: RStudio

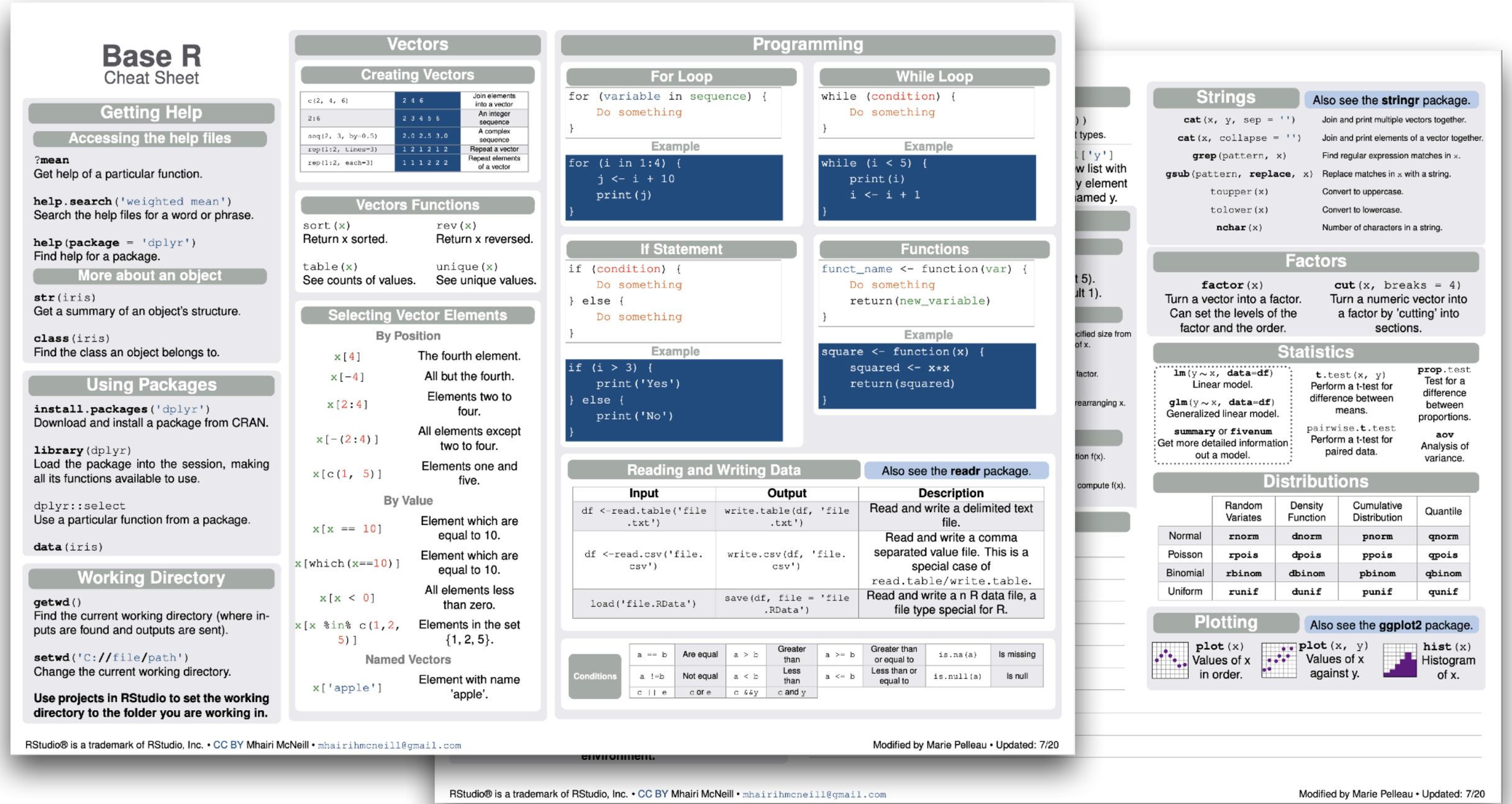
R console integrated with editor, var list, output, help etc



R language basics

Rough summary of topics we will cover in the next four lectures

https://github.com/rstudio/cheatsheets/blob/main/base-r.pdf



Base R
Cheat Sheet

Getting Help

Accessing the help files

`?mean`
Get help of a particular function.

`help.search('weighted mean')`
Search the help files for a word or phrase.

`help(package = 'dplyr')`
Find help for a package.

More about an object

`str(iris)`
Get a summary of an object's structure.

`class(iris)`
Find the class an object belongs to.

Using Packages

`install.packages('dplyr')`
Download and install a package from CRAN.

`library(dplyr)`
Load the package into the session, making all its functions available to use.

`dplyr::select`
Use a particular function from a package.

`data(iris)`

Working Directory

`getwd()`
Find the current working directory (where inputs are found and outputs are sent).

`setwd('C://file/path')`
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

`c(2, 4, 6)`	2 4 6	Join elements into a vector
`2:6`	2 3 4 5 6	An integer sequence
`seq(2, 3, by=0.5)`	2.0 2.5 3.0	A complex sequence
`rep(1:2, times=3)`	1 2 1 2 1 2	Repeat a vector
`rep(1:2, each=3)`	1 1 1 2 2 2	Repeat elements of a vector

Vectors Functions

`sort(x)` Return x sorted.
`rev(x)` Return x reversed.
`table(x)` See counts of values.
`unique(x)` See unique values.

Selecting Vector Elements

By Position

`x[4]`	The fourth element.
`x[-4]`	All but the fourth.
`x[2:4]`	Elements two to four.
`x[-(2:4)]`	All elements except two to four.
`x[c(1, 5)]`	Elements one and five.

By Value

`x[x == 10]`	Element which are equal to 10.
`x[which(x==10)]`	Element which are equal to 10.
`x[x < 0]`	All elements less than zero.
`x[x %in% c(1,2,5)]`	Elements in the set {1, 2, 5}.

Named Vectors

`x['apple']`	Element with name 'apple'.

Programming

For Loop

```
for (variable in sequence) {
    Do something
}
```

Example

```
for (i in 1:4) {
    j <- i + 10
    print(j)
}
```

While Loop

```
while (condition) {
    Do something
}
```

Example

```
while (i < 5) {
    print(i)
    i <- i + 1
}
```

If Statement

```
if (condition) {
    Do something
} else {
    Do something
}
```

Example

```
if (i > 3) {
    print('Yes')
} else {
    print('No')
}
```

Functions

```
funct_name <- function(var) {
    Do something
    return(new_variable)
}
```

Example

```
square <- function(x) {
    squared <- x*x
    return(squared)
}
```

Reading and Writing Data

Also see the **readr** package.

Input	Output	Description
`df <-read.table('file.txt')`	`write.table(df, 'file.txt')`	Read and write a delimited text file.
`df <-read.csv('file.csv')`	`write.csv(df, 'file.csv')`	Read and write a comma separated value file. This is a special case of `read.table/write.table`.
`load('file.RData')`	`save(df, file = 'file.RData')`	Read and write a n R data file, a file type special for R.

Conditions

`a == b`	Are equal	`a > b`	Greater than	`a >= b`	Greater than or equal to	`is.na(a)`	Is missing
`a != b`	Not equal	`a < b`	Less than	`a <= b`	Less than or equal to	`is.null(a)`	Is null
`c | e`	c or e	`c && y`	c and y				

Strings

Also see the **stringr** package.

`cat(x, y, sep = '')`	Join and print multiple vectors together.
`cat(x, collapse = '')`	Join and print elements of a vector together.
`grep(pattern, x)`	Find regular expression matches in x.
`gsub(pattern, replace, x)`	Replace matches in x with a string.
`toupper(x)`	Convert to uppercase.
`tolower(x)`	Convert to lowercase.
`nchar(x)`	Number of characters in a string.

Factors

`factor(x)` Turn a vector into a factor. Can set the levels of the factor and the order.
`cut(x, breaks = 4)` Turn a numeric vector into a factor by 'cutting' into sections.

Statistics

`lm(y ~ x, data=df)` Linear model.
`glm(y ~ x, data=df)` Generalized linear model.
`summary` or `fivenum` Get more detailed information out a model.
`t.test(x, y)` Perform a t-test for difference between means.
`pairwise.t.test` Perform a t-test for paired data.
`prop.test` Test for a difference between proportions.
`aov` Analysis of variance.

Distributions

	Random Variates	Density Function	Cumulative Distribution	Quantile
Normal	`rnorm`	`dnorm`	`pnorm`	`qnorm`
Poisson	`rpois`	`dpois`	`ppois`	`qpois`
Binomial	`rbinom`	`dbinom`	`pbinom`	`qbinom`
Uniform	`runif`	`dunif`	`punif`	`qunif`

Plotting

Also see the **ggplot2** package.

`plot(x)` Values of x in order.
`plot(x, y)` Values of x against y.
`hist(x)` Histogram of x.

RStudio® is a trademark of RStudio, Inc. • CC BY Mhairi McNeill • mhairihmcneill@gmail.com

Modified by Marie Pelleau • Updated: 7/20

RStudio® is a trademark of RStudio, Inc. • CC BY Mhairi McNeill • mhairihmcneill@gmail.com

Modified by Marie Pelleau • Updated: 7/20

Warwick Statistics

First Year R Course

WARWICK STATISTICS SOCIETY

Overview of topics:

- Basics of R (R studio, operations in R and logical operators)
- Data Types & Data Structures
- Control Structures
- Custom Functions
- Brief insight into Applications of R

Access to slides (with theory, code demos and questions), demonstrations in R, practice questions and more…

When & Where?

Every **Wednesday 3-4pm**
(After Stats Café)

MB0.07 (W2-4,6-7,9-10)
MB0.08 (W5,8)

Mathematical Sciences (Stats) Building

Finding help

- If you know what command you need, typing ?command brings up a help page, e.g. ?sqrt
- If you know approximately what command, or a keyword, ??word searches for help pages.
- Lots of free online help is available.
  - Short intro: http://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf
  - Short reference card: http://cran.r-project.org/doc/contrib/Short-refcard.pdf
  - Long reference: http://web.udl.es/Biomath/Bioestadistica/R/Manuals/r_in_a_nutshell.pdf
  - Online course (if you like watching video lectures): https://www.coursera.org/course/rprog
- Lots of non-free books are now available
  - General books, e.g., Venables and Ripley, Modern Applied Statistics with S-Plus; Crawley, The R Book and Statistics: An introduction using R.
  - Specific books, with titles like Statistical Method X with R, often published by CRC Press.

Sorting or Ordering Vectors

Description

Sort (or *order*) a vector or factor (partially) into ascending or descending order. For ordering along more than one variable, e.g., for sorting data frames, see `order`.

Usage

```
sort(x, decreasing = FALSE, ...)

## Default S3 method:
sort(x, decreasing = FALSE, na.last = NA, ...)

sort.int(x, partial = NULL, na.last = NA, decreasing = FALSE,
         method = c("auto", "shell", "quick", "radix"), index.return = FALSE)
```

Arguments

`x`	for `sort` an `R` object with a class or a numeric, complex, character or logical vector. For `sort.int`, a numeric, complex, character or logical vector, or a factor.
`decreasing`	logical. Should the sort be increasing or decreasing? Not available for partial sorting.
`...`	arguments to be passed to or from methods or (for the default methods and objects without a class) to `sort.int`.
`na.last`	for controlling the treatment of `NA`s. If `TRUE`, missing values in the data are put last; if `FALSE`, they are put first; if `NA`, they are removed.
`partial`	`NULL` or a vector of indices for partial sorting.

Interacting with R

- Console: Type here for immediate action. Text output comes here.
- R script: Write programs, longer work that you want to save, modify, etc.
- Graphics windows are generated by R commands.
- Important shortcuts:
  - Up-arrow repeats previous line. Repeating goes back through your history.
  - Ctrl-Enter (Cmd-Enter on Mac) after selecting text in an R-script window runs the selected commands in the console.
- Integrated development environments (IDEs) such as RStudio (available free) help to organise the windows, scripts, variables, etc.

“Translate” into R:

$(-17)^2$

$x = 23,\ y = (x - 89)^{-2}$

$0^0 + 1^1 + 2^2 + 3^3 + 4^4 + 5^5 + 6^6 + 7^7$

The formula for the solution of quadratic equations $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ (either case) and calculate the value for a few cases of values a, b, c. Try to use some values such that the discriminant is negative to see what happens then.

An expression that returns TRUE or FALSE depending on whether the discriminant in the expression above.

Find out which class these expression are:

```
“Hello world”
‘Hello world’
`Hello world’
```

Note: One of these create a problem that is worth to have encountered, so you will remember how to get unstuck. You will get unstuck by pressing control C.

```
7
“7”
as.integer(7)
as.integer(“7”)
as.integer(7)^2
floor(8.9)
7==8
7==7
```

Generating sequences:

Generate a sequence of numbers divisible by 3 between 1 and 31.

Find an alternative expression that returns the same result.

Create a sequence of even numbers between 1 and 6 that show each number half of the times as its value. Do the same but showing them inn reverse order.

Show all numbers between 99 and 100 in steps of 0.01.

Create a sequence that shows ABC three times. Do this in another way, too.

Generating sequences:

Generate a sequence of numbers divisible by 3 between 1 and 31.

Find an alternative expression that returns the same result.

Create a sequence of even numbers between 1 and 6 that show each number half of the times as its value. Do the same but showing them inn reverse order.

Show all numbers between 99 and 100 in steps of 0.01.

Create a sequence that shows ABC three times. Do this in another way, too.

Create vectors:

- Create a vector with 10 Poisson(3) distributed values.
- Create a vector `prime` containing all one digit prime numbers starting from 2.
- Without typing it into R, say what will > `prime[order(-prime)]` show?
- Without typing it into R, say what will the following commands show?
  `> c(1:10)[prime]  > c(-10:5[prime])  > c(-5:9)[order(-prime)]`

- Create a vector `u` containing all odd numbers smaller than 26 starting from 1. Do so without listing all these numbers.
- Try what happens if you type `LETTER[1]`
- Print every other capital letter starting with “A”. Make the command as short as possible. You may use `u.`
- Print every other letter starting with “B”. Make the command as short as possible. You may use `u.`
- Print every other letter starting with “Z”going backwards. Make the command as short as possible. You may use `u.`

Operating with vectors and lists

- Create the vector (1,2,3,4,5) in R, call it `v`, and divide each component by -1. Can you do this with a command that is at most 5 characters long?
- Divide the components of `v` by 0.1,0.2,0.3,0.4,0.5, respectively. Do so with a command that is at most 6 characters long.
- Without typing it into R, what is the output of the following command?
  `> (v-3)^2 != 0`
- First generate a vector `w` of length 20 with entries of your choice. Then remove the components in the even positions. Use no more than 15 characters to create this expression. (You don't have to create any variable for the vector you output.)
- You type `> seq(0,1,.1)=0.5` and get the error message below. Explain what it actually means and what the correct command should be.
  `target of assignment expands to non-language object`
- Create three vectors of lengths three. The first one contains three items, the second one has the colours, and the third one gives their weight. Write some commands that select objects based on their colour or their weight. Also try to select them based on a combination of conditions about colour and weight.
- Do the previous exercise again, but use the data structure list.

Create matrices and arrays:

- Create a 4 by 5 array using 20 values sampled from a normal distribution. Display the last 3 columns. Display the array without the first row.
- Create the same but using matrix syntax.
- Create the array using only three normally distributed values by recycling them.
- Create a 3 by 4 array using the first 12 letters of the alphabet. Display the first two rows.
- Create a 3-dimensional array of dimensions 2,3,4 and display it by showing two subarrays.

Create your own questions and tasks!

- In the last lectures I suggested short practical exercise for each topic we studied. These questions were created to help you reviewing, understanding, and remembering the material. Going forward, can generate such questions and tasks yourself? This will make the material more memorisable and lead to a deeper understanding.
- Some ways of *generating* questions and tasks:
  - For which data types does this command work/not work? Why/why not?
  - Can I do something like it was done in a different dimension? Or involving different functions?
  - Pick a few lines of R code from the slides at random and copy them onto a different page. Without looking at the original context, can you say what their output would be and what they do?
  - For data visualisations: Why is this an effective way to communicate the message in the data? Why not? How could you improve it?
  - Try to “break” some R code from the lecture slides. What can you change about the commands or the arguments so it results in an error message?

Exercise

Task

Write an `R` function which:

1. Takes as argument a vector `x`.
2. Returns `x` if all the elements of `x` are non-negative. Else, it returns a message saying that "Some elements in the input vector are negative".

For example, suppose your input vector is `x=(1,0,1,5,100)`. Then the function should return `x`. If your input vector is `x=(1,-1,1)`, then the function should return "Some elements in the given vector are negative".

Exercise

Task

Write an `R` function `multiple.return` which takes a vector `x` of length 4, and returns:

- the sum of the elements of `x`
- the product of the elements of `x`
- a 2×2 matrix composed of the elements in `x`