# Introduction to R

Course organiser: Eilis Hannon UEMS

E.J.Hannon@exeter.ac.uk

Please:

Collect two different coloured post it notes from the demonstrators.

Course notes and data are available here: **https://biomedicalhub.github.io/R-intro/.** Download the data files for today's session onto your desktop and unzip the data.zip folder.

# Guide to today's workshop

Designed to be very hands on – uses a lot of content from data carpentry/software carpentry.

(http://www.datacarpentry.org/)

(https://software-carpentry.org/)

Many of the techniques are applicable to many other programming languages.

Mixture of live coding and exercises.

Sticky/Post-it notes are for you to indicate when you have finished a task (green) or need help from a demonstrator (red).

Course notes and ppt are available online.

# Reasons to use R

- it's free, well-documented, and runs almost everywhere
- it has a large (and growing) user base among scientists
- it has a large library of external packages available for performing diverse tasks
- it is a gateway into other programming languages

What benefit do you think R will have for your research?

# Exercise

After running these commands:

```
mass <- 47.5
age <- 122
mass <- mass * 2.0
age <- age – 20
```

What is the value of

1. mass? 95
2. age? 102

# Recap

**Command** – a line of code you pass to the R console

e.g. x<-10

**Function** – performs a specific task and has a name

e.g. read.csv()

NB commands are based on functions

**Variable** – a value* with a name or reference that can be changed. * can be a number, string, matrix, list…

**Argument** – a value that is passed to a function to specify what the function should work on or how it should work

# Exercise

We can take slices of character vectors as well:

What would the following return?

```
animal <- c("m", "o", "n", "k", "e", "y")
animal[1:3]
animal[4:6]
```

# Exercise

Suppose you want to determine the maximum inflammation for patient 5 across days three to seven.

Which of the following lines of R code gives the correct answer?

1. `max(dat[5, ])`
2. `max(dat[3:7, 5])`
3. `max(dat[5, 3:7])`
4. `max(dat[5, 3, 7])`

Hint: To do this you would extract the relevant slice from the data frame and calculate the maximum value.

# Structure of a for loop

```
for (variable in collection){

    do things with variable

}
```

# Exercise

What would be the output of this for loop?

```
for (i in 1:10){

    print(i*2)

}
```