Learning R

Session 1: Introduction to the R programming language

DR EDMUND RYAN

Overview of this R Course

1. Introduction to R (session 1)
2. Creating graphical plots with R (session 2)

Task 1

3. Correlation and Regression with R (session 3)
4. Creating a model with R using For loops and if-else statements (session 4)

Task 2

5. Hypothesis testing with R (session 5)
6. Using R packages to solve problems (session 6)

Task 3

Overview of this R Course

1. Introduction to R (session 1)
2. Creating graphical plots with R (session 2)

Task 1

3. Correlation and Regression with R (session 3)
4. Creating a model with R using For loops and if-else statements (session 4)

Task 2

5. Hypothesis testing with R (session 5)
6. Using R packages to solve problems (session 6)

Task 3

Why use R?

- Simple to use!
- R is an interpretable language, which means that the user gets an immediate response, i.e. no compiler.
- R is open source and used extensively in academia and industry
- R interfaces with most programming languages (e.g. C++, …)
- It's free!!!

What you will learn today

- How to install R and R studio
- R as a calculator
- Saving your work in a script file
- Importing / Exporting data
- Data types
- Matrices and data frames
- Vectors and lists
- Converting between matrices, data frames, vectors and lists.
- Subsetting and missing values
- Getting help and resources

Installing R and R studio

- Installing R: https://cran.r-project.org/
- R studio is a useful user interface for running R:
  - You can run R without R studio, but R on its own has a very basic interface. R studio has a lot more functionality.
  - Like R it is open source and so free to install.
  - If you want to use R studio you have to install R first. Remember it is only an interface.
- Installing R studio ("R studio desktop"): https://rstudio.com/products/rstudio/download/

R as a calculator

After opening R Studio, type the following **bits in red** pressing 'Enter' after each row. The **bits in brown** is what answer you should be given:

```
> 6 – (9 + 10) * pi^3
[1]  -583.1193

> 6 – 9 + 10 * pi^3
[1]  307.0628

> exp(log(5))
[1]  5

> log(1000, 10)
[1]  3
```

Saving your work as a script file

- Go to: File → New File → R script
- Copy the R commands from the previous slide into your new script file, i.e.

```
6 – (9 + 10) * pi^3
6 – 9 + 10 * pi^3
etc…
```

- Save your script file (e.g. Rcourse_session1.R)
- To run a section of the script, highlight that section and press the 'Run' button (a shortcut is Ctrl+Enter).
- To add comments to a line use # at start of the line.

Importing / Exporting data (csv, txt, etc..)

- Set your working directory (where your data file is stored). For example:
  `setwd('C:/Work/Rcourse/Session1')`
- Import data. Some common examples:
  `my.data=read.table("filename", header=TRUE, sep=",")`
  (for any file type - txt, csv, etc..)
  `my.data=read.csv("filename", header=TRUE)`
  (for csv files only)
- Exporting data (x=name of data structure in R):
  `write.table(x,"filename")` *(for any file type - txt, csv, etc..)*
  `write.csv(x,"filename")` *(for csv files only)*

Importing / Exporting data (csv, txt, etc..)

- Try this out by downloading some data from: http://doi.org/10.5281/zenodo.3873765
- This is model output of gross primary production (amount of carbon taken up by a plant via photosynthesis) from research I did in a previous job (PHACE experiment).
- It contains four columns of gross primary production model output corresponding to four treatment groups: (i) control, (ii) warming, (iii) elevated CO_2, (iv) warming & elevated CO_2.
- Type 'PHACE experiment' into google for more information.
- Save this csv file to a relevant folder on your computer.

Importing / Exporting data (SQL server)

- Reading in data from a table within a SQL server database.

```
dbhandle = odbcDriverConnect('driver={SQL Server}; server=ServerName;
database=DatabaseName;trusted_connection=true')
my.data <- sqlQuery(dbhandle,
                    paste('
                    SELECT Variable1, Variable2
                    FROM TableName;
                    ',sep=''));
odbcClose(dbhandle)
```

Importing / Exporting data (R data)

- R lets you easily save all your files

```
a = 1
b = 2
c = 3
save(a, b, file = "stuff.RData")
```

- *Stuff.RData* will appear in the working directory. When you next open R you can load it by:

```
load("stuff.RData")
```

- R automatically saves your commands from previous session:

```
loadhistory(".Rhistory")
history()
```

Assigning values to variables

- Use of the equal sign:

```
a = 8
a
 [1] 8
```

- Assigning values

```
a <- 8
a
 [1] 8
8 -> a
a
 [1] 8
```

- When assigning values make sure the arrow is in correction direction:

```
a -> 8
Error in a -> 8 :
invalid (do_set) left-
hand side to assignment
```

Data types

- Logical

```
x = 5; y = 3
x > y
 [1] TRUE
```

- Numeric / integer

```
a=5; b=sqrt(2)
a; b
 [1] 5
 [2] 1.41214
```

- Character

```
a="1"; b=1
a; b
 [1] "1"
 [2] 1

a = "hello"; a
 [1] "hello"

c = a; c
 [1] "hello"
```

Matrices and data frames

- Matrix
  - Rectangular table of numerical data.
  - For example, average temperatures for every month of a year, across five UK cities.
  - Matrices are useful if you need to do matrix algebra.
- Data frame
  - Similar to a matrix but we can store categorical data.
  - For example rather than using a number for age we could make it categorical, e.g. “0-10”, “11-20”, etc…
  - In data frames we can specify column and row names.

Matrices

```
m = matrix(1:6,nrow=2, ncol=3, byrow=T);
m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
t(m)    #Transpose of matrix
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
dim(m)   #Dimension of matrix
[1] 2 3
```

Data frames

```
L3 = c("a","b","c")
L3rep = rep(L3, 2)        #rep means repeat L3 twice.
d = data.frame(x = 1, y = 1:6, lets = L3rep)
d
  x y lets
1 1 1    a
2 1 2    b
3 1 3    c
4 1 4    a
5 1 5    b
6 1 6    c
```

Vectors and lists

- A vector is a one dimension matrix.
- We can express it as a list (c means 'combine'):

```
x = c(3.1, 6.7, 7.5)
log(x)
[1] 1.131402 1.902108 2.014903

a = 1:4
b = seq(1,1.75, by=0.25) #seq means 'sequence'.
a + b
[1] 2.00 3.25 4.50 5.75

length(a)  #this gives the number of elements in a.
[1] 4
```

Converting between matrices, vectors and data frames

- Certain aspects of using R requires the data to be in a particular structure (e.g. when you write data to a file it usually needs to be in the form of a data frame).
- R will let you know this by an error message.
- Use:
  - as.matrix() to convert to a matrix.
  - as.data.frame() to convert to a data frame.
  - as.vector() to convert to a vector.
  - as.list() to convert to a list (I rarely use this).
  - as.factor() to convert to a factor (I only used this when doing regression).

Subsetting

- We may need to extract a subset of our data.
- R offers a couple of ways to do that:

```
x = c("a", "b", "c", "d", "e", "b")
x[2]
[1] "b"

x[3:5]
[1] "c" "d" "e"

y=c(2:7)    #This is the same as y=c(2,3,4,5,6,7)
y[x=="b"]
[1] 3 7
```

Missing value

- R can handle data that are “not available” (NA):

```
x = c(3, 7, NA)
x
[1]  3  7 NA
```

- You can still apply mathematical operations:

```
x + 5
[1]  7 12 NA
```

- Use is.na() function to identify or remove NAs:

```
x[is.na(x)==FALSE]
[1] 3 7
```

Putting it altogether (example code)

```
#Set working directory:
setwd('C:/Work/Rcourse/Session1')

#Import data:
my.data=read.csv('PHACE_photosynthesis_data.csv',header=TRUE)

#list the variable names of my.data and display the first few rows of data.
names(my.data)
head(my.data)

#Calculate the number of rows in my.data. If I'd used [2] → no. of columns:
N=dim(my.data)[1]

#Create a list of numbers which will be the row index (and call it 'index'):
index = c(1:N)
```

Putting it altogether (example code)

```
#Select the 'Control' column of my.data:
Control = my.data$Control

#Select the row numbers where there are NAs in the 'Control' column:
index.NA = index[is.na(Control)==TRUE]; index.NA

#Select the row numbers where there are no NAs in the 'Control' column:
index.noNA = index[is.na(Control)==FALSE]
length(ind); length(index.noNA)

#Choose the first three columns of my.data and select rows with no NAs.
my.data.noNA = my.data[index.noNA,1:3]

#Write the data frame
write.csv(my.data.noNA,"PHACE_photosynthesis_data_Control_noNA.csv",
row.names = FALSE)
```

Getting help

- Search on google.
- Getting help about a specific command:

```
> ? apply
```

- Finding functions related to a key word:

```
> help.search("boxplot")
```

- Starting the R installation help pages:

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