INTRODUCTION TO R AND RSTUDIO

Shane Galvin

Setting up your environment and Getting started

Download R

Go to:

http://ftp.heanet.ie/mirrors/cran.r-project.org/

Select operating system (Windows, MAC)

Linux/Ubuntu/Debian → Sudo install from a ppa

Download latest version (3.5.0)

Install using the binary file

Download Rstudio

Go to:

https://www.rstudio.com/products/rstudio/download/#download

Select operating system (Windows, MAC)

Linux/Ubuntu/Debian → Sudo install from a ppa

Select latest version (1.1.453)

Install using the binary file

R and Rstudio Download

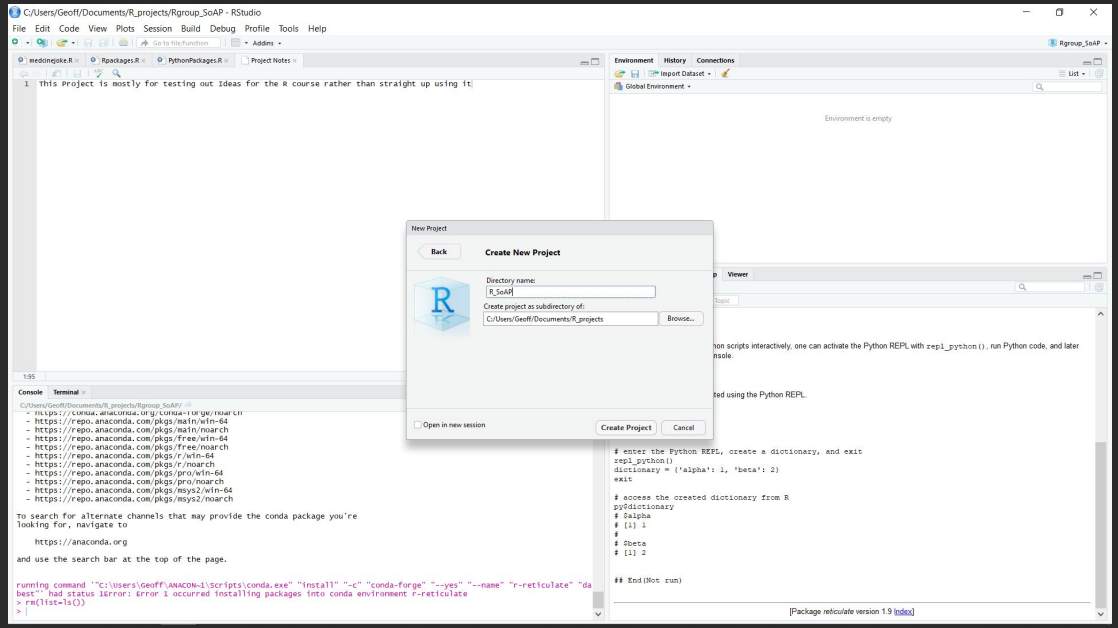
1) In Windows File Viewer

- Create a new folder in My Documents named R_Projects

Set up the project 1

2) Open Rstudio

- Click File > New Project >
- Select R_Projects using the Browse button
- Name the Project R_SoAP



Set up the project 2

3) Open the R_SoAP folder in Windows File Viewer

- Copy the contents of RGroupSoAP folder into the R_SoAP folder

Set up the project 3

About Me.

Interests

- Psychometric modelling
- Response Behaviour
- Research Methods
- Statistics

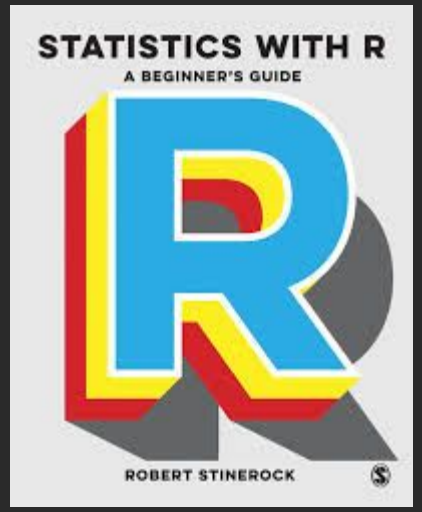
Education

BA, PhD Student
School of Applied Psychology UCC

Contact

Twitter: @S_B_Galvin
Email: shane.galvin@ucc.ie
Github: @SB_Galvin

Recommended Reading and Resources I



Stinerock, R. (2018). *Statistics with R: A Beginner's Guide*. SAGE.

Bonus, no mention of cats...

Recommended Reading and Resources II

Fundamentals of Data Visualization
Claus O. Wilke
http://serialmentor.com/dataviz/

ggplot2: Elegant Graphics for Data Analysis
Hadley Wickham
http://moderngraphics11.pbworks.com/f/ggplot2-Book09hWickham.pdf

R for Data Science
Garrett Grolemund, Hadley Wickham
http://r4ds.had.co.nz/

Advanced R
Hadley Wickham
https://adv-r.hadley.nz/

ANOVA: A Short Intro Using R
Lukas Meier
http://stat.ethz.ch/~meier/teaching/anova/

Tidyverse Website
https://www.tidyverse.org/

R DOCUMENTATION *****

Rbloggers *

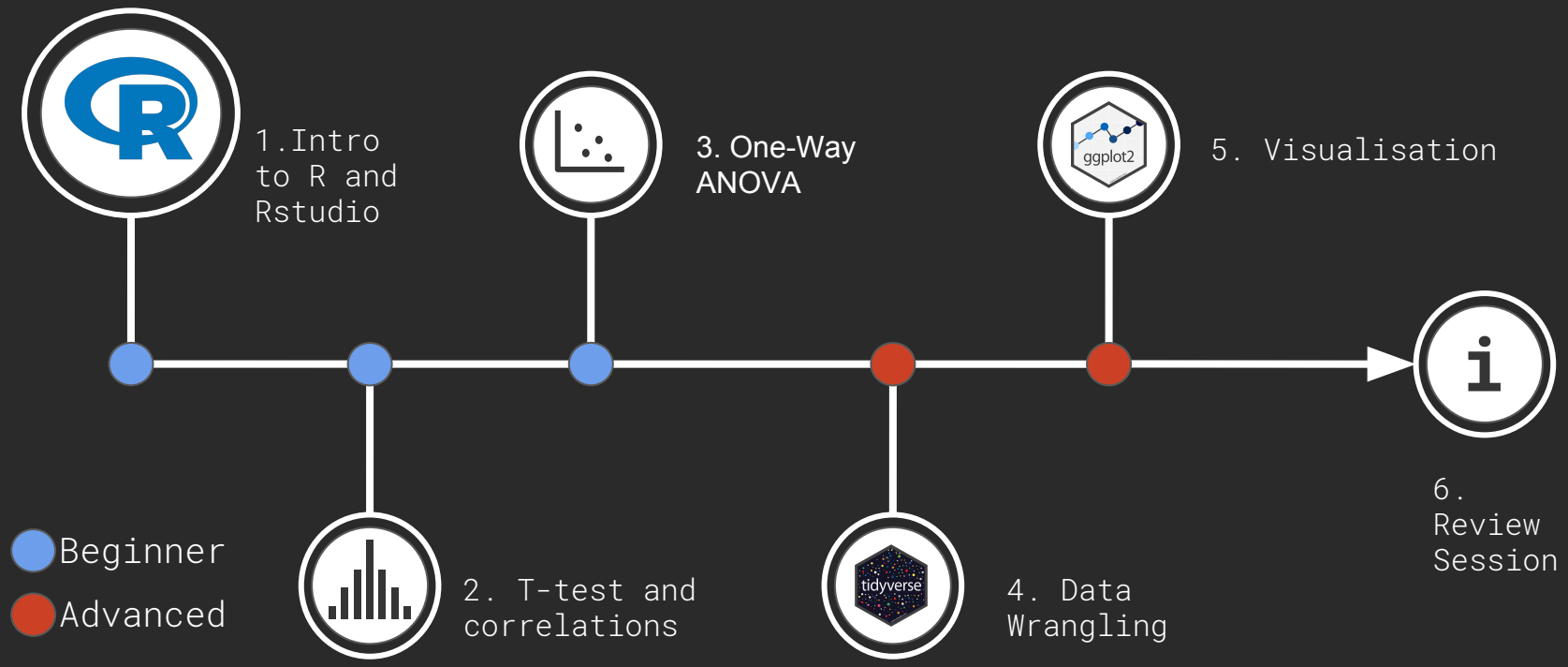
Stack Exchange -Psych *

Recommended Reading and Resources III

Swirl R package
https://swirlstats.com/
Tutorial package for stats in R, run inside R console
Very detailed, focused on teaching stats

Install and run inside R:

```
# Rcode
install.packages("swirl")
library(swirl)
swirl() # this runs the swirl routine
```



Workshop layout

Familiarity with R, Rstudio and “lingo”

- Packages, Repository etc

Understand basics of programming in R

- Variable assignment, Loops, Functions

Familiarity with data types

- Characters, Strings, Factors, Numerical

Installing and Loading Packages

Be able to read in different types of data files

Be able to summarise data

Learning Outcomes

How to use R

Theory

- Introduction
- Rstudio IDE
- Programming basics
- R: Good Code Bad Code

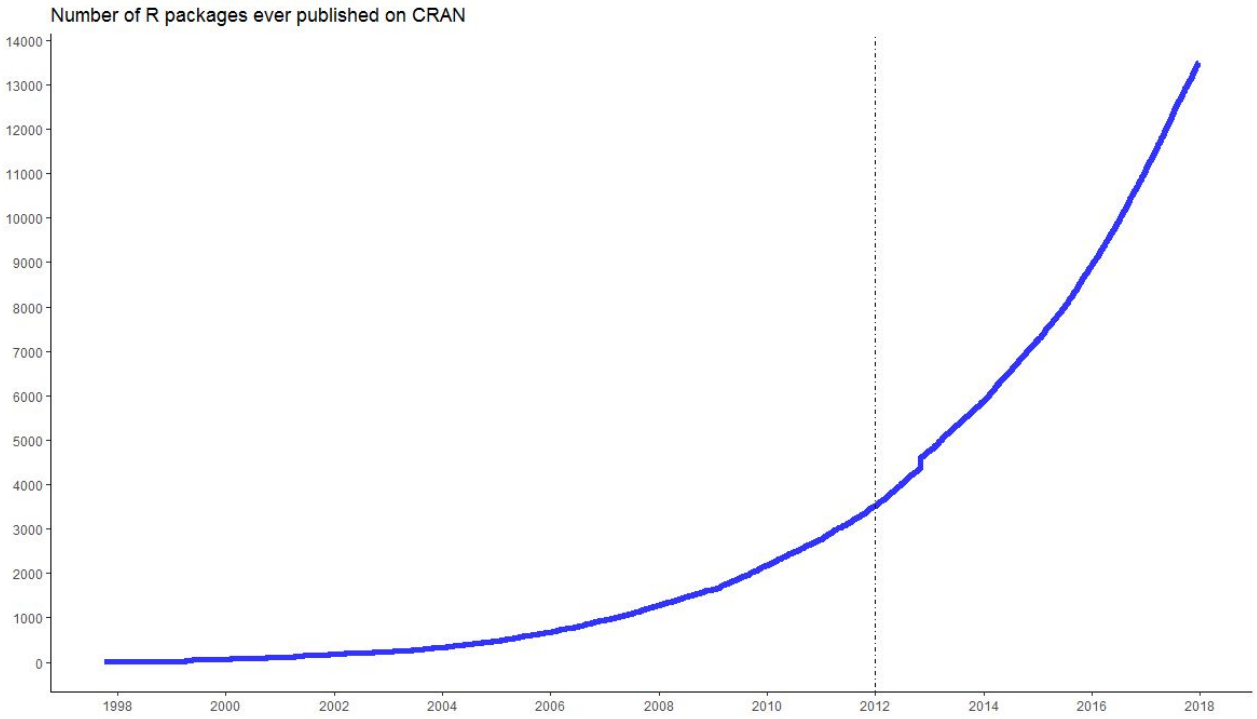
Session 1 Exercises

- Section 0: Programming
- Section 1: Good Code/ Bad Code
- Section 2: Loops and Functions
- Section 3: DataFrames
- Section 4: Reading in Data
- Section 5: Summarising Data
  - Notebook, psych package

Session 1 Practical

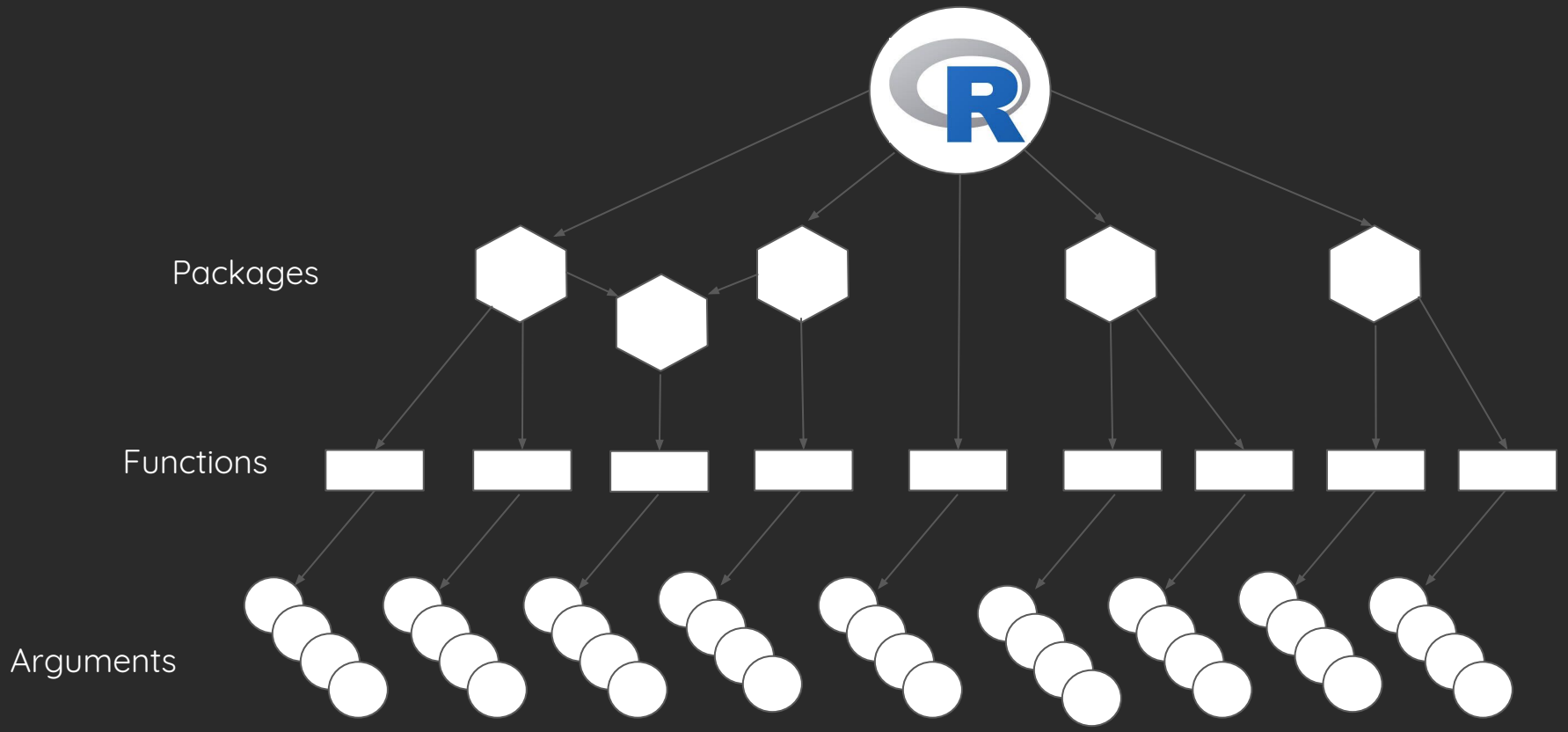
- Please copy the Link above into a separate window in a web browser
- (Phone or Laptop)
- Post questions there
- Questions will be addressed at the end of each section
- At the end of your question please type the slide number to which it relates
- You can up or down vote questions

For Q & A


Number of R packages ever published on CRAN
14000
13000
12000
11000
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018

	SAS	SPSS	R	Python
Advantages	1. High adoption rate in major industries 2. Flow based interface with drag and drop 3. Official support 4. Handling large datasets 5. 'PROC SQL'	1. Used a lot in universities 2. Good user interface with extensive documentation 3. Click & Play functionality 4. Writing code made easy using the ´paste´ button. 5. Official support	1. Big community who creates libraries 2. Free 3. Early adopter in explanatory and predictive modeling. 4. Easy to connect to data sources, including NoSQL and webscraping.	1. Scalability 2. General purpose language 3. Easy to learn 4. Good in machine learning 5. Big community 6. Free
Disadvantages	1. Relatively high cost 2. For not-standard options not in interface, you'll need to write the code 3. Slow adapting to new techniques 4. Different programs for visualization or Data Mining	1. Relatively high cost 2. different licenses for different functionalities. 3. Syntax limited 4. Slow adapting to new techniques 5. Slow in handling large datasets	1. Can be slow with big datasets 2. Steep learning curve 3. No official support 4. No user interface	1. Not as strong in explanatory modeling 2. Choice of version: 2.7 or 3.5? 3. No user interface 4. No official support

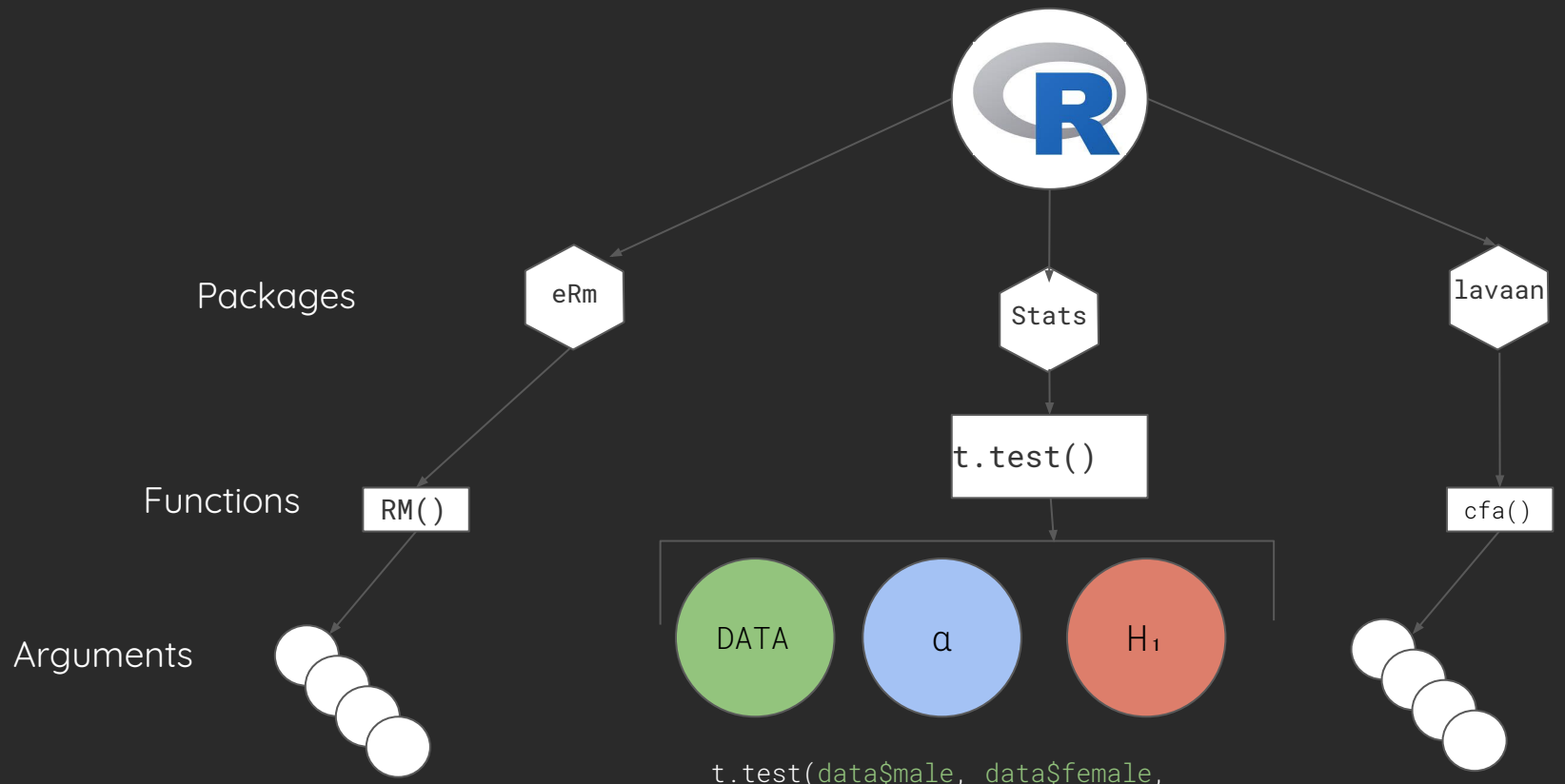
structure


Packages
Functions
Arguments

Package <- A collection of functions

Function <- A type of procedure or routine that performs a specific task
E.g. mean() finds the mean

Argument <- A value that you pass into a function
E.g. passing an IQ variable as an argument to the mean function → mean(IQ)
Other arguments could be an alpha level

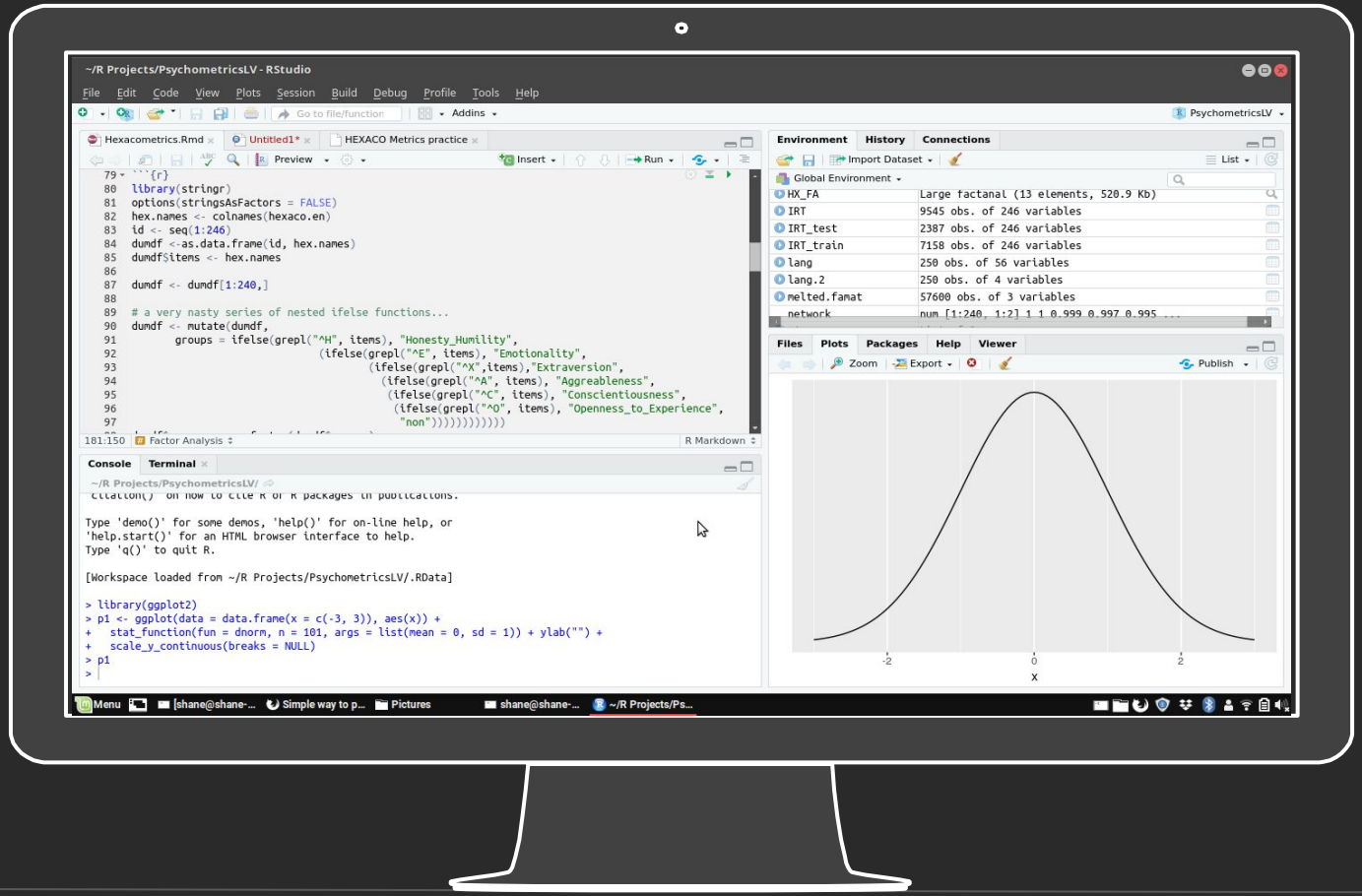



```
t.test(data$male, data$female,
       p.value=0.05, alternate='greater')
```

RStudio IDE

Scripting

Environment



Console

Plotting
Help
File Explorer

Rstudio IDE

1 Titles

2 Text and Notes

3 Code Chunks

5 Output

6 This allows to jump between sections

R Chunk Label

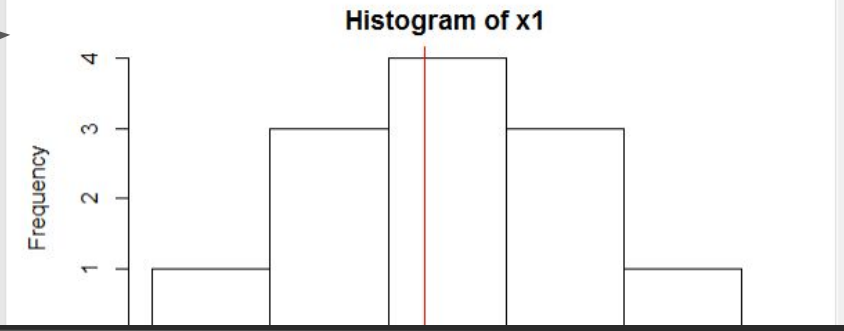
4 To run a code chunk

```
# Section 0
## Programming
This section is for an introduction to programming section

### The octothorpe
slide number
```{r Commenting}
x1 <- c(1.1,
        2.6, 2.5, 2.1,
        3.2, 3.6, 3.2, 3.4,
        4.1, 4.5, 4.2,
        5.2)
# mean(x1) # this will not run
median(x1) # this will run
```

[1] 3.3

```{r}
hist(x1)
abline(v=mean(x1),col='red')
```
```



Rmd Notebook

Section 0: Programming Concepts

Assignment Operators

- <- left assignment operator (also =, <<-)
- ->> right assignment operator (also =, ->)

Arithmetic operators

- + [add]
- - [subtract]
- / [divide]
- * [multiply]
- ^ [raise to the power]

Logical operators

- < [less than]
- > [greater than]
- & [and]
- | [Logical OR operator]
- == Equivalence

https://www.tutorialspoint.com/r/r_operators.htm

Operators

#comment

Using the # octothorpe in (most) programming languages prevents that commented line from running

This is handy for annotating code with comments

Also for testing your code for problems

Remove the octothorpe in line 55 to find the mean of x1

```
38 ```{r}
39 x1  <-  c(1, 2, 3, 4, 5)
40 x1
41 ```
```

```
[1] 1 2 3 4 5
```

```
42
43 This code will run
44 ```{r}
45 median(x1) # this will run
46 ```
```

```
[1] 3
```

```
47
48 This Code will not
49 ```{r}
50 # mean(x1) # this will not run
51 ```
52
53 Now remove the `#` at the start of line 52
54 ```{r}
55 #mean(x1)
56 ```
57
```

Commenting

We can perform arithmetic easily in R

C:/Users/Shane/Desktop/ShaneHome/R Projects/R_Group_SoAP - RStudio Source...

Session1_Exercises.Rmd

Knit Insert Run

```
### Arithmetic Operators

So we know that R can function as a calculator:
```{r Arithmetic 1}
3 + 4
```

Subtraction:
```{r Arithmetic 1}
1372 - 21
```


Multiplication:
```{r Arithmetic 2}
# Multiplication
5 * 5
```

Division:
```{r Arithmetic 3}
# Division
21 / 3
```

Raise to the power
```{r Arithmetic 4}
# Power
2 ^ 3
```


square root function divided by 2:
```{r Arithmetic 5}
# Square Root
sqrt(7056)/2
```

`pi` is already contained as a named value
```{r Arithmetic 6}
# pi
sqrt(pi)
```

```

135:19 Chunk 11: Arithmetic 8 R Markdown

# Arithmetic operators

We can assign a datum to an object

To Run a line of code, click the mouse cursor on the line and press “ctrl + ENTER”

The output will print to the console

What is g ?

```
# Rcode
a <- 1
a
b <- 2
b
c <- 3
c
a + b
# assigning new value to e
e <- c - a
e

# f <- b * e
g <- f + e
```

Variables

Using the concatenate **c()** function we can concatenate a values into a list named my_list.

If we just wanted to pull up the value for b we could use indexes or the position number of b

Interestingly we can also multiply all of the my_list values

Now multiply my_list * 7 and save this as a new object my_list2, print mylist2 to the console

```
# Rcode
my_list <- c(a, b, c)
my_list

# Index to find b
my_list[2]

# Multiply by 7
my_list * 7

# my_list2
```

Vectors

Indexes numbers are location identifiers for values in an object

We can use indexes to retrieve values by placing an index value inside square brackets beside the name of an object e

g OBJECTNAME[1] will return the first value in the OBJECTNAME object.

R indexes start from 1

In R we can index to select specific values from a list

```
Mp  <- c(9, 2, 7, 4, 5)

Mp[1]

## Returns 9

Mp[2]

## Returns 2

Mp[1:4]

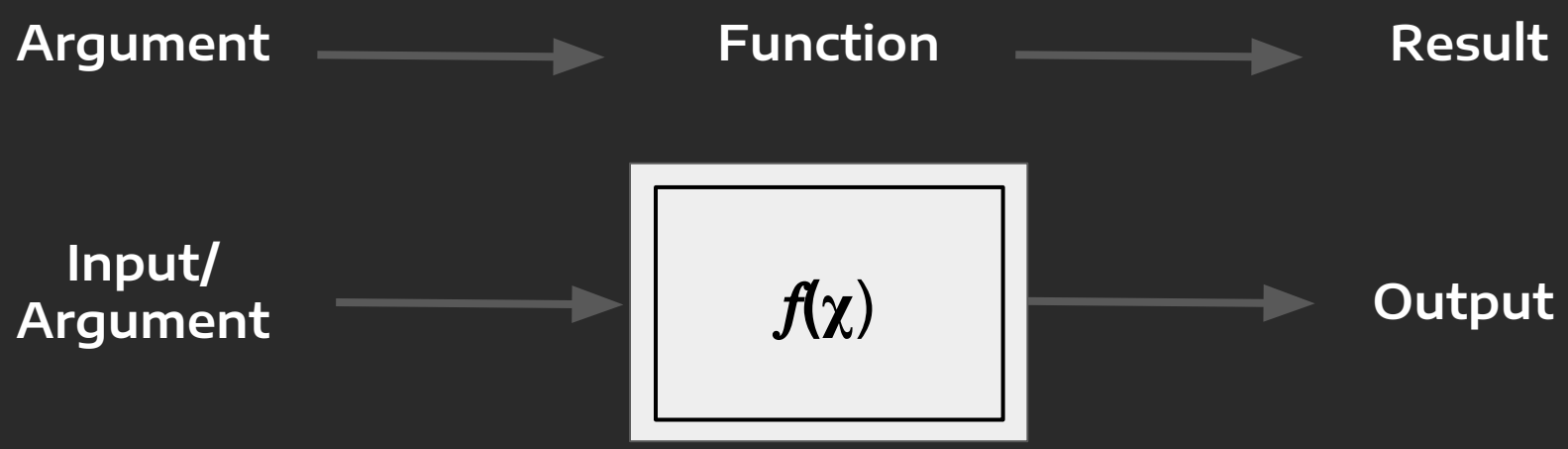
## Returns 9, 4
```

Indexing and Sequences

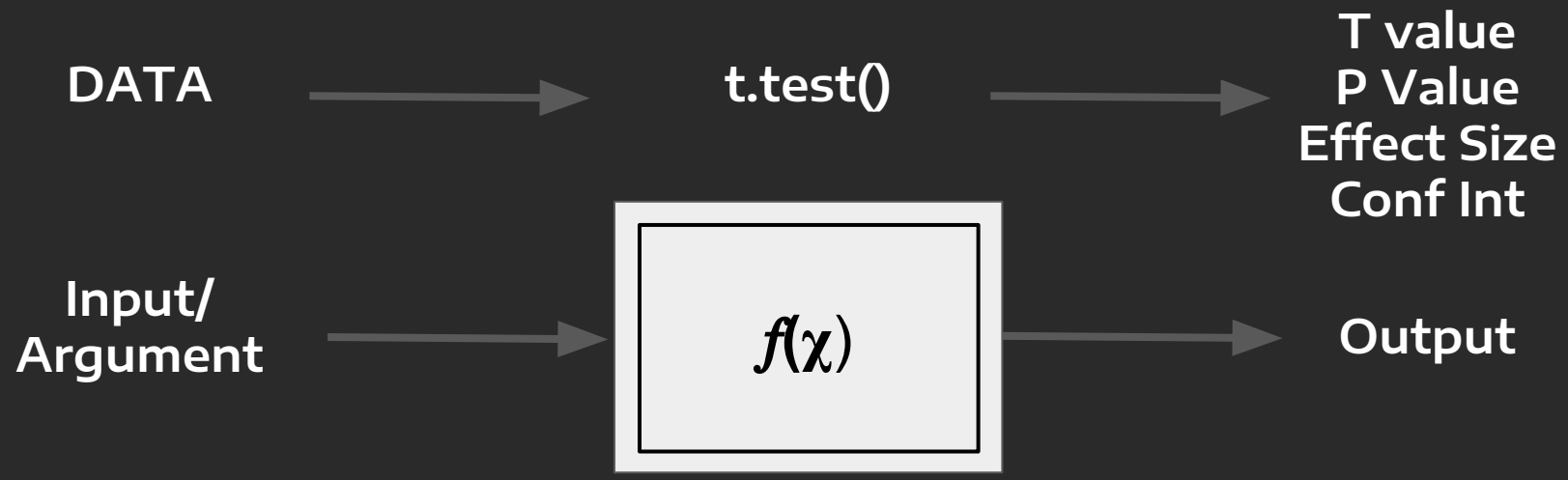
- **Characters**
  - Individual text symbols enclosed in quotes "A"
- **Strings**
  - A string of characters enclosed in quotes "SuperCalifragilisticExpialidocious"
- **Integer**
  - Whole numbers eg. 1
- **Numeric**
  - Numeric values (Continuous) eg 23.456321
- **Factors** (Categorical variables with levels)
  - Sex <- c("Male", "Female")
- Matrices → A two-dimensional table
- Array → A n-dimensional table
- Vectors → An object containing a list of values
- Data frames → A data table with headers and different data types

Data Types

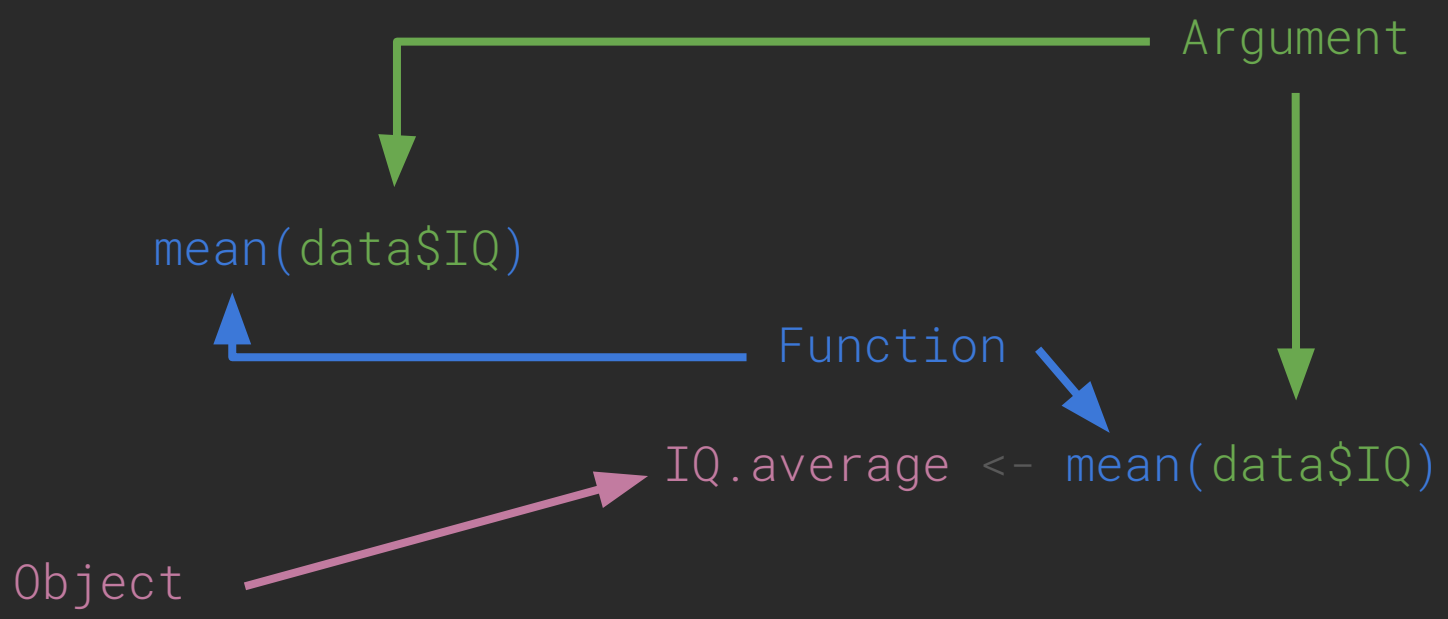
Section 1: Functions



Functions


DATA
t.test()
T value
P Value
Effect Size
Conf Int
Input/
Argument
f(χ)
Output

Functions


Argument
mean(data$IQ)
Function
IQ.average <- mean(data$IQ)
Object

Functions

```
install.packages("") # install r packages
require(); library() # call r packages
length() # length
mean() # average
sum() # sum
prod() # product
names() # column names
head() # first 6 rows
tail() # last 6 rows
View() # view data
class() # data structure
str() # structure
apply # vectorisation
rm() # remove R object
summary() # summary
plot() # plot
```

```
c() #concatenate
cbind() #column combine
rbind() #row combine
as.factor(), as.matrix() ; as.numeric();
as.character() #data structure
data.frame() # data frame object
list() # list object
ifelse() #if-else statement
stop("") #kill message
message("") #write message
write.csv(); # write csv file
read.csv("") # read csv files
attach(); detach() # work with one
dataset
rm(list = ls()) # Clears all in
environment
capture.output() # Captures output as
object
```

Pressing F1 during the function autocomplete in a script will bring up the help documentation for a function

Frequently used functions

A for loop is a way to iteratively apply function

FOR each value *i* in a sequence X, do this..

```
X <- c(1, 2, 3)

for (i in X) {
  print(i * 2)
}
#Output:
2
3
6
```

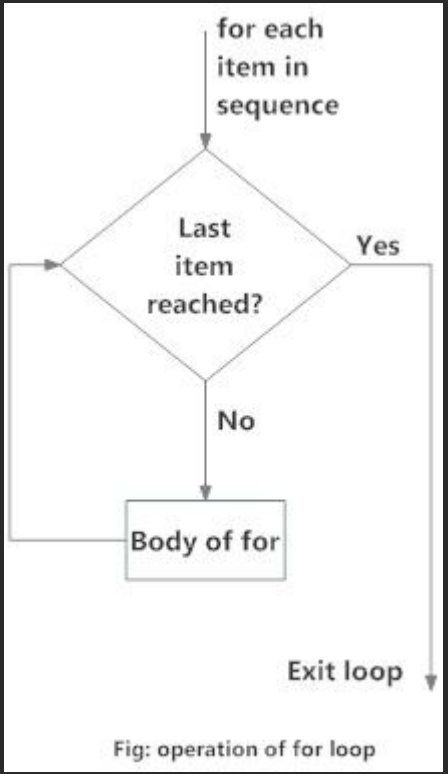


Fig: operation of for loop

For loops

If else statements are conditional statements;

IF THING meets *this CONDITION*, then do *THIS*

ELSE, do THIS..

```
If (df$Sex == “Male”){

        (df$Score - 1)

    }Else{df$Score = df$Score
```

Ifelse Statements

A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean (True/False) condition.

The while loop can be thought of as a repeating if statement.

WHILE this CONDITION is true; then do THIS, when this CONDITION is FALSE, then **stop**.

While Loops

There is a function for creating functions called function(){}

```r
# Rcode
DoubleNum <- function(x){
              for (i in x):{
                  i * 2
                  }
               }
DoubleNum(my_list2)
```

User Defined Functions

For loops, while loops and ifelse functions in can be quite slow.

We can vectorise our functions using apply() to speed the process up

If we have a ton of data, or have a lot of conditions to satisfy we can use C++ (C plus plus) code in R using the Rcpp package to process data faster

this is largely unnecessary for experimental data, but useful for large data (2gb +) from social media sites, or for neuroscience data, fmri, eye-tracking data sets etc.

Section 1: Good Code/ Bad Code

Messy code is not useful to anyone

ALWAYS LABEL YOUR CODE

Use the octothorpe # to insert comments to explain what your code is doing

This way someone else can read your code, and you won't have to remember

Google's R style guide: https://google.github.io/styleguide/Rguide.xml

```
# Crap Rcode
xX= rnorm(10000,100,15)
Xx =rnorm(10000, 37, 1.6)

# Good Rcode
# Generates 10k,random values
iq.sim  <-  rnorm(10000, 100, 15)

open.b5 <-  rnorm(10000, 37, 1.6)

hist(iq.sim)
```

Good Code/ Bad Code

Open Session_1Exercise.Rmd

Coffee Break

https://github.com/SBGalvin/Introduction-to-R-and-Rstudio

Basics of R:

Data Types

Functions

Loops

Vectorisation

Reading in Data

Writing Data

Rstudio

Notebooks

Scripts

Review

Questions:

Answers:

Thank you!