R FOR ARTS

MARCH 10TH 2017

ABOUT THIS COURSE

R: A LOVE-HATE RELATIONSHIP

> *Frustration is natural when you start programming in R, because it is such a stickler for punctuation, and even one character out of place will cause it to complain.*
>
> Hadley Wickam, Chief Scientist at RStudio

So, we'll do our very best to keep things light-hearted. But...

we will also get you to learn R the hard way.

MEET THE TUTORS

EWA WANAT	See me for any administrative issues.	
ROBERT LENNON	I'm here to help with R.	
VIJAY SOLANKI	I'm here to answer any and all of your questions about R.	

COURSE STRUCTURE

Three day course, where each day covers a specific set of goals:

DAY 1 - FRIDAY 10TH MARCH

Introduces R, demonstrates basic skills and outlines proper coding and script management practices.

DAY 2 - FRIDAY 17TH MARCH

Begins to consider "real" data and how to manage, interpret and evaluate it in R.

DAY 3 - FRIDAY 24TH MARCH

Focuses on communication of findings from analyses performed in R.

COURSE STRUCTURE

Each day will roughly follow the same format:

- Begin at 2pm, finish at 6pm
- Start with a 1hr presentation
- Followed by 3hrs of guided practice

WHERE TO FIND COURSE CONTENT

www.ewanat1.wixsite.com/rforarts

WHAT IS R?

- According to the official R website:

  > *"R is a free software environment for statistical computing and graphics."*

is a language

WHY USE R?

- Because it’s FREE
- Because it is fit for purpose
  - It is designed for data analysis
  - It has a LARGE repository of packages
  - It is being actively updated and maintained
  - It makes sharing analyses easy
  - It makes it easy to communicate your results
- Because you are not alone!
  - Most R users are not programmers
  - It has an active community

SAY HELLO TO

```
#R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
#Copyright (C) 2016 The R Foundation for Statistical Computing
#Platform: x86_64-pc-linux-gnu (64-bit)
#
#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 publicati
#
```

```
#!/usr/bin/Rscript
###
## R Script for pre-processing of inquiry reports
## Written by : Vijay Solanki
## Date: 14/12/16
###

## Pre-processing warning
n = 0
while(n <1 ){
          cat("WARNING: Before beginning the pre-processing, I need to know a few things.\n \n(1)
          selector <- readline( "Do you have this information to hand? (y/n)\n" )

          if(selector == "n" | selector == "no" | selector == "No" | selector == "NO"){
                    #cat("Okay, please collect that information and then restart this script. \n"
                    stop("Okay, please collect that information and then restart this script. \n"
```

UNDERSTANDING THE ENVIRONMENT

- 2 core elements:
  - Command Window

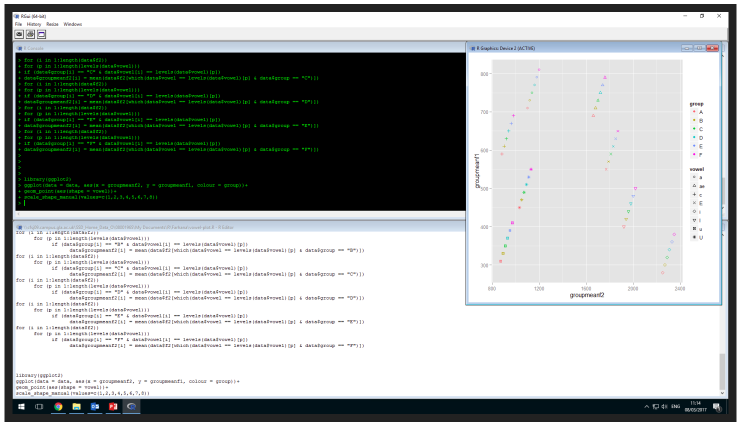
    This is where you tell R to do stuff (and where R will tell you why you've done it wrong)
  - Script Window

    This is where you should be doing all of your work!
- 1 additional element:
  - Plot window

    This is where you will be able to 'see' your data

The aesthetics of these windows differ between operating systems.

Windows

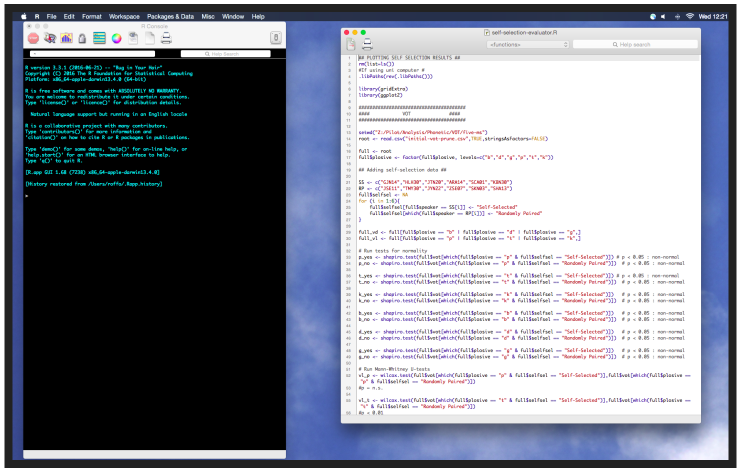


```
> for (i in 1:length(data$f2))
+ for (p in 1:length(levels(data$vowel)))
+ if (data$group[i] == "C" & data$vowel[i] == levels(data$vowel)[p])
+ data$groupmeanf2[i] = mean(data$f2[which(data$vowel == levels(data$vowel)[p] & data$group == "C")])
> for (i in 1:length(data$f2))
+ for (p in 1:length(levels(data$vowel)))
+ if (data$group[i] == "D" & data$vowel[i] == levels(data$vowel)[p])
+ data$groupmeanf2[i] = mean(data$f2[which(data$vowel == levels(data$vowel)[p] & data$group == "D")])
> for (i in 1:length(data$f2))
+ for (p in 1:length(levels(data$vowel)))
+ if (data$group[i] == "E" & data$vowel[i] == levels(data$vowel)[p])
+ data$groupmeanf2[i] = mean(data$f2[which(data$vowel == levels(data$vowel)[p] & data$group == "E")])
> for (i in 1:length(data$f2))
+ for (p in 1:length(levels(data$vowel)))
+ if (data$group[i] == "F" & data$vowel[i] == levels(data$vowel)[p])
+ data$groupmeanf2[i] = mean(data$f2[which(data$vowel == levels(data$vowel)[p] & data$group == "F")])
>
>
>
>
> library(ggplot2)
> ggplot(data = data, aes(x = groupmeanf2, y = groupmeanf1, colour = group))+
+ geom_point(aes(shape = vowel))+
+ scale_shape_manual(values=c(1,2,3,4,5,6,7,8))
> 
```

\\cfsj09.campus.gla.ac.uk\SSD_Home_Data_O\0800196S\My Documents\R\Farhana\vowel-plot.R - R Editor

```
for (i in 1:length(data$f2))
      for (p in 1:length(levels(data$vowel)))
            if (data$group[i] == "B" & data$vowel[i] == levels(data$vowel)[p])
                  data$groupmeanf2[i] = mean(data$f2[which(data$vowel == levels(data$vowel)[p] & data$group == "B")])
for (i in 1:length(data$f2))
      for (p in 1:length(levels(data$vowel)))
            if (data$group[i] == "C" & data$vowel[i] == levels(data$vowel)[p])
                  data$groupmeanf2[i] = mean(data$f2[which(data$vowel == levels(data$vowel)[p] & data$group == "C")])
for (i in 1:length(data$f2))
      for (p in 1:length(levels(data$vowel)))
            if (data$group[i] == "D" & data$vowel[i] == levels(data$vowel)[p])
                  data$groupmeanf2[i] = mean(data$f2[which(data$vowel == levels(data$vowel)[p] & data$group == "D")])
for (i in 1:length(data$f2))
      for (p in 1:length(levels(data$vowel)))
            if (data$group[i] == "E" & data$vowel[i] == levels(data$vowel)[p])
                  data$groupmeanf2[i] = mean(data$f2[which(data$vowel == levels(data$vowel)[p] & data$group == "E")])
for (i in 1:length(data$f2))
      for (p in 1:length(levels(data$vowel)))
            if (data$group[i] == "F" & data$vowel[i] == levels(data$vowel)[p])
                  data$groupmeanf2[i] = mean(data$f2[which(data$vowel == levels(data$vowel)[p] & data$group == "F")])



library(ggplot2)
ggplot(data = data, aes(x = groupmeanf2, y = groupmeanf1, colour = group))+
geom_point(aes(shape = vowel))+
scale_shape_manual(values=c(1,2,3,4,5,6,7,8))
```

Mac

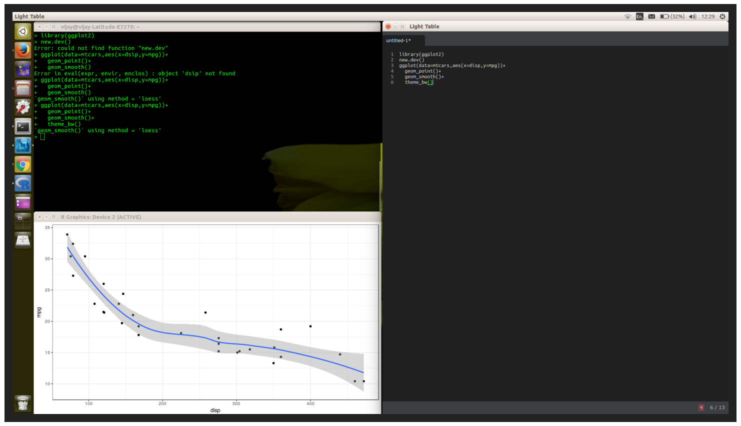


R File Edit Format Workspace Packages & Data Misc Window Help
Wed 12:21
R Console
Help Search
self-selection-evaluator.R
<functions>
Help search

```
R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

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.

[R.app GUI 1.68 (7238) x86_64-apple-darwin13.4.0]

[History restored from /Users/roffo/.Rapp.history]

>
```

```
## PLOTTING SELF SELECTION RESULTS ##
rm(list=ls())
#If using uni computer #
.libPaths(rev(.libPaths()))

library(gridExtra)
library(ggplot2)

#######################################
####            VOT              ####
#######################################

setwd("Z:/Pilot/Analysis/Phonetic/VOT/five-ms")
root <- read.csv("initial-vot-prune.csv",TRUE,stringsAsFactors=FALSE)

full <- root
full$plosive <- factor(full$plosive, levels=c("b","d","g","p","t","k"))

## Adding self-selection data ##

SS <- c("GJN14","HLH30","JTN20","ARA14","SCA01","KBN30")
RP <- c("JSE11","TMY30","JYN22","ZSE07","SKN03","SHA13")
full$selfsel <- NA
for (i in 1:6){
    full$selfsel[full$speaker == SS[i]] <- "Self-Selected"
    full$selfsel[which(full$speaker == RP[i])] <- "Randomly Paired"
}

full_vd <- full[full$plosive == "b" | full$plosive == "d" | full$plosive == "g",]
full_vl <- full[full$plosive == "p" | full$plosive == "t" | full$plosive == "k",]

# Run tests for normality
p_yes <- shapiro.test(full$vot[which(full$plosive == "p" & full$selfsel == "Self-Selected")]) # p < 0.05 : non-normal
p_no <- shapiro.test(full$vot[which(full$plosive == "p" & full$selfsel == "Randomly Paired")])  # p < 0.05 : non-normal

t_yes <- shapiro.test(full$vot[which(full$plosive == "t" & full$selfsel == "Self-Selected")]) # p < 0.05 : non-normal
t_no <- shapiro.test(full$vot[which(full$plosive == "t" & full$selfsel == "Randomly Paired")])  # p < 0.05 : non-normal

k_yes <- shapiro.test(full$vot[which(full$plosive == "k" & full$selfsel == "Self-Selected")])   # p < 0.05 : non-normal
k_no <- shapiro.test(full$vot[which(full$plosive == "k" & full$selfsel == "Randomly Paired")])  # p < 0.05 : non-normal

b_yes <- shapiro.test(full$vot[which(full$plosive == "b" & full$selfsel == "Self-Selected")])   # p < 0.05 : non-normal
b_no <- shapiro.test(full$vot[which(full$plosive == "b" & full$selfsel == "Randomly Paired")])  # p < 0.05 : non-normal

d_yes <- shapiro.test(full$vot[which(full$plosive == "d" & full$selfsel == "Self-Selected")])   # p < 0.05 : non-normal
d_no <- shapiro.test(full$vot[which(full$plosive == "d" & full$selfsel == "Randomly Paired")])  # p < 0.05 : non-normal

g_yes <- shapiro.test(full$vot[which(full$plosive == "g" & full$selfsel == "Self-Selected")])   # p < 0.05 : non-normal
g_no <- shapiro.test(full$vot[which(full$plosive == "g" & full$selfsel == "Randomly Paired")])  # p < 0.05 : non-normal

# Run Mann-Whitney U-tests
vl_p <- wilcox.test(full$vot[which(full$plosive == "p" & full$selfsel == "Self-Selected")],full$vot[which(full$plosive ==
 "p" & full$selfsel == "Randomly Paired")])
#p = n.s.

vl_t <- wilcox.test(full$vot[which(full$plosive == "t" & full$selfsel == "Self-Selected")],full$vot[which(full$plosive ==
 "t" & full$selfsel == "Randomly Paired")])
#p < 0.01
```

Linux



```
> library(ggplot2)
> new.dev()
Error: could not find function "new.dev"
> ggplot(data=mtcars,aes(x=dsip,y=mpg))+
+   geom_point()+
+   geom_smooth()
Error in eval(expr, envir, enclos) : object 'dsip' not found
> ggplot(data=mtcars,aes(x=disp,y=mpg))+
+   geom_point()+
+   geom_smooth()
`geom_smooth()` using method = 'loess'
> ggplot(data=mtcars,aes(x=disp,y=mpg))+
+   geom_point()+
+   geom_smooth()+
+   theme_bw()
`geom_smooth()` using method = 'loess'
> 
```

```
library(ggplot2)
new.dev()
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point()+
  geom_smooth()+
  theme_bw()
```

WORKING WITH R

As stated above, do ALL of your work in the script window. This will:

- Create a record of your work
- Allow you to comment your work for others (and your future self!)
- Make you work reproducible

Working in the command window is fine but should only be used for quick tasks that aren't important to your overall goal.

CODING PRACTICE (CP)

Throughout this course, we will encourage some sensible general coding practices.

CP1: # EVERYTHING, BECAUSE #YOLO

Possibly, one of the more important pieces of advice.

In R, the # symbol will 'comment out' everything that follows it on that line.

This allows you to make comments about your work and is SUPER IMPORTANT!

PRELIMINARIES

Some brief notes about the code you will see in this course:

- It should be possible to copy any code presented in this course directly into R.
- The # indicates that the code in that line is a comment and will not be interpreted by R. This is true for all R code, not just the code in this course.
- All output printed to the command window will be commented out to ensure that code can be run without issue, eg:

```
1 + 1
#[1] 2
```

PRELIMINARIES

WORKING DIRECTORIES

You will need to become familiar with the notion of "working directories". This simply indicates the folder (directory) that you are currently working out of. This is important because it dictates where your work will be saved!

In R you can find your current working directory by typing the following into the command window

```
getwd()
```

There are some differences between how Windows systems and Unix based systems (Mac/Linux) manage their directory structures.

Windows	C:\Users\vijay\Documents
Unix	/home/vijay/

The \ character has a special meaning for R. Windows users can either use \\ when writing directory paths or can use Unix style / characters.

PRELIMINARIES

WORKING DIRECTORIES

You can change your current working directory by using the following function

```
setwd()
```

You will need to provide the function with the address of the directory you wish to change to, eg:

```
setwd('/home/vijay/R-for-Arts')
```

Note that the directory address MUST be enclosed in quotation marks

CP2: LEAVE YOURSELF SPACE!

Code is hard to read at the best of times. Make life easy for yourself and put spaces between code elements.

Bad:

```
1+2+3+4+5+6+7+8+9+10
```

Good:

```
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
```

BASIC OPERATIONS

In some ways R is just a glorified calculator. It can be used to perform simple arithmetic for:

Addition

```
1 + 1
#[1] 2
```

Subtraction

```
2 - 1
#[1] 1
```

Division

```
4 / 2
#[1] 2
```

Multiplication

```
2 * 2
#[1] 4
```

However, R is so much more than a simple calculator.

For instance, R is able to handle text:

```
'R for Arts'
#[1] "R for Arts"
```

It can also evaluate logical statements:

```
2 > 1
#[1] TRUE
2 + 1 == 5
#[1] FALSE
```

Note that in the above examples, all output values are printed to the screen, eg:

```
1 + 1
#[1] 2
```

This applies only to commands issued to the command window that are not assigned to an object.

OBJECTS

- Assigning a value to an object stores it in R's memory.
- This is done using the assignment character

  `<-`

  or

  `=`

- It is recommended that `<-` is used to assign value to objects, this will save confusion in code later on.

```r
a <- 1 + 1
a
# [1] 2
```

CP3: SENSIBLE NAMES

Providing your R objects with sensible names will save you a lot of pain in the future.

Object names require the following:

- They MUST begin with a letter
- They MUST NOT contain spaces
- They ARE case sensitive
- They can ONLY contain letters, numbers, _ and .

Be consistent in your naming convention, some example conventions are:

- this_is_snake_case
- ThisIsCamelCase
- this.uses.full.stops

Objects can contain most output that you generate from your calls to R.

```r
a <- 1 + 1
b <- 'R for Arts'
c <- 2 > 1
```

To see what is in an object, simply type its name

```r
a
#[1] 2
b
#[1] "R for Arts"
c
#[1] TRUE
```

These examples show the three of the four R data classes that are of interest to us.

DATA CLASSES

R has a number of different data classes, those of interest to us are:

- Numeric

```
a <- 1
class(a)
#[1] "numeric"
```

- Character

```
b <- 'R for Arts'
class(b)
#[1] "character"
```

- Logical

```
c <- TRUE
class(c)
#[1] "logical"
```

- Factor

```
d <- factor(c("heads", "tails", "tails", "heads", "tails"))
class(d)
#[1] "factor"
```

The `class()` function can be used to determine the class of an object (we'll get to functions shortly).

COMBINING DATA

Numeric, Character and Logical data classes can be combined with other data of the same class using the `c()` function, eg:

```
a <- c( 1 , 2 , 3 , 4 , 5 )
a
#[1] 1 2 3 4 5

b <- c( 'a' , 'b' , 'c' , 'd' , 'e')
b
#[1] "a" "b" "c" "d" "e"

c <- c( T , F , T , T , F )
c
#[1] TRUE FALSE TRUE TRUE FALSE
```

Factors cannot. This is because they are made up of smaller elements such as numerics, char acters or logicals.

Factors are a special type of dat a class.

FACTORS

Factors are best used to represent categorical data. They can be thought of as a vector where each element is associated with a category, eg.

['Brian' ,	'Garfield' ,	'Claus' ,	'Mog' ,	'Nemo']
|	|	|	|	|
Dog	Cat	Fish	Cat	Fish

Factors can be ordered or unordered.

FACTORS

This is an example of an unordered factor:

```
d <- factor(c("heads", "tails", "tails", "heads", "tails"))
d
#[1] heads tails tails heads tails
#Levels: heads tails
```

This is an example of an ordered factor:

```
d <- factor(c("low", "middle", "high", "high", "low"),levels=c
d
#[1] low middle high high low
#Levels: low < middle < high
```

Notice how the levels for the ordered factor have symbols indicating which element is higher.

FACTORS

Factors help in identifying the possible values that a variable might take even if they don't all appear in your data.

```
coin_toss_character <- c( 'heads' , 'heads' , 'heads' )
table(coin_toss_character)
#heads
#3

coin_toss_factor <- factor(coin_toss_character , levels = c( '
table(coin_toss_factor)
#heads   tails
#    3       0
```

DATA STRUCTURES

DO YOU SPEAK VECTOR?

R is a vector based language. Meaning that it performs operations on colections of values at the same time, rather than individually.

A vector can be thought of simply as a collection of values.

[1 2 3 4 5 6 7 8 9 10]

or

['a' 'b' 'c' 'd' 'e' 'f' 'g']

DO YOU SPEAK VECTOR?

By using vectors, R makes data analysis easier.

$$\begin{array}{ccccc} 1 & & 2 & & 2 \\ 2 & & 2 & & 4 \\ 3 & X & 2 & = & 6 \\ 4 & & 2 & & 8 \\ 5 & & 2 & & 10 \end{array}$$

(ATOMIC) VECTORS

Vectors only have 1-dimension and can contain any data class as long as they are all the same.

```
numeric_vector <- c( 1 , 2 , 3 , 4 , 5 )
#[1] 1 2 3 4 5

character_vector <- c( 'a' , 'b' , 'c' , 'd' , 'e' )
#[1] "a" "b" "c" "d" "e"

logical_vector <- c( T , F , F , T , T )
#[1] TRUE FALSE FALSE TRUE TRUE
```

LISTS

Lists only have 1-dimension and can contain any data class.

```
mixed_list <- list( 1 , 'a' , TRUE , 63 , FALSE )
#[[1]]
#[1] 1

#[[2]]
#[1] "a"

#[[3]]
#[1] TRUE

#[[4]]
#[1] 63

#[[5]]
#[1] FALSE
```

MATRICES

Matrices have 2-dimensions and can contain any class of data as long as they are all the same.

```
numeric_matrix <- matrix(0,2,2)
#       [,1] [,2]
#[1,]     0    0
#[2,]     0    0

character_matrix <- matrix('R',2,2)
#       [,1] [,2]
#[1,]   "R"  "R"
#[2,]   "R"  "R"

logical_matrix <- matrix(T,2,2)
#       [,1] [,2]
#[1,]   TRUE TRUE
#[2,]   TRUE TRUE
```

DATA FRAMES

Data frames have 2-dimensions and can contain any class of data.

```
mixed_data_frame <- data.frame(c(1,2,3),c('a','b','c'),c(T,F,F
#  c.1..2..3. c..a....b....c.. c.T..F..F.
#1          1                a         TRUE
#2          2                b        FALSE
#3          3                c        FALSE
```

The data frame is probably the most common data structure that you will work with.

Notice that the character vector does not have quotation marks.

This is a built in process of data frames, character vectors are automatically converted to factors

SUMMARISING DATA STRUCTURES

	Homogeneous	Heterogeneous
1d	Atomic Vector	List
2d	Matrix	Data Frame
nd	Array	

Table from Advanced R

FUNCTIONS

Functions can be thought of as series of commands that are collected into a single object.

For example, R has the function `mean()` which is used to determine the mean of a collection of data, eg:

```
a <- c( 1 , 2 , 3 , 4 , 5)
mean(a)
#[1] 3
```

The same result could also be achieved by performing the arithmetic:

```
a <- (1 + 2 + 3 + 4 + 5) / 5
a
#[1] 3
```

But functions are cleaner (and often faster!).

SOME HANDY FUNCTIONS:

	Object Management		Object Management (cont.)
Removes objects	`rm()`	Get object class	`class()`
Lists objects	`ls()`	Is object a numeric	`is.numeric()`
Gets object data class	`class()`	Is object a character	`is.character()`
Gets object structure	`str()`	Is object a logical	`is.logical()`
Gets object length	`length()`	Is object a factor	`is.factor()`
Gets object dimensions	`dim()`	Remove all objects	`rm( list = ls() )`

SOME (MORE) HANDY FUNCTIONS:

	Window Management		File Management
Closes R	`q()`	List files in wd	`dir()`
Opens plot window	`dev.new()`	Lists files in wd	`list.files()`
Closes plot window	`dev.off()`	Gets wd	`getwd()`
'See' your data	`View()`	Sets wd	`setwd()`
		Runs saved script	`source()`

UNDERSTAND BY DOING

As with many things, the best way to understand how something works is to build it yourself.

One of the powerful things about R is that if you need a function and it doesn't exist, you can write it yourself!

The general form of a function is:

```
function_name <- function( ){
                    stuff what the function goes and does
                    }
```

Notice that functions make use of curly brackets {}, more on types of bracket next week.

UNDERSTAND BY DOING

If you type most functions into R's command window without their accompanying brackets, ie. ( ), you will see the code for that function, eg:

```
View
#function (x, title)
#{
#    check <- Sys.getenv("_R_CHECK_SCREEN_DEVICE_", "")
#    msg <- "View() should not be used in examples etc"
#    if (identical(check, "stop"))
#        stop(msg, domain = NA)
#    else if (identical(check, "warn"))
#        warning(msg, immediate. = TRUE, noBreaks. = TRUE, dom
#    if (missing(title))
#        title <- paste("Data:", deparse(substitute(x))[1])
#    as.num.or.char <- function(x) {
#        if (is.character(x))
#            x
#        else if (is.numeric(x)) {
```

UNDERSTAND BY DOING

So, lets say that we want to write a function that calcuates the mean of a numeric vector. We need to know 2 things:

1. The number of elements in the vector
2. The sum of the elements in the vector

If we know both of these things, we can write the function!

UNDERSTAND BY DOING

```
# My function to calculate the mean of a numeric vector
my_mean <- function( x ){                        # x is the input numer
                element_no <- length( x ) # Here I find out th
                vector_sum <- sum( x )    # Here I sum the ele
                input_mean <- vector_sum / element_no # Here I
                return( input_mean )      # Here I return the
                }

a <- c( 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 )

my_mean( a ) == mean( a )
#[1] TRUE
```

Functions are a bit like Las Vegas.

What happens in a function, st ays in a function

Unless you ask it to spill the beans, notice the `return()` function.

GETTING HELP

ASK R

R has a number of in-built functions for getting help. These examples use the mean function as a test case.

If you know the function name use

```
help(mean)
```

or

```
?mean
```

If you just want to perform a general search use

```
help.search('mean')
```

or

```
??mean
```

LISTEN TO R

If R gets upset with you and throws an error, the very first thing to do is look at the error message.

Although R is very picky about how you give it instructions, it is also very vocal about why it is upset with you. Reading the error message often points to the solution to the problem.

IF ALL ELSE FAILS, ASK GOOGLE

The beauty of R being open source is that someone else is likely to have already encountered your error (or at least something similar).

Some useful websites include:

- stackoverflow
- Quick-R
- R-bloggers

EXERCISES

UNDERSTANDING THE ENVIRONMENT

1. Identify the command window and describe its use.
    a. Would you use this window for recording your analyses?
    b. How would you use this window to close R?
    c. Would you use the # character in this window?

2. Identify the script window and describe its use.
    a. Would you use this window for recording your analyses?
    b. What purpose does the # character serve in this window?
    c. Should you be using the # character regularly in this window? Why?

UNDERSTANDING THE ENVIRONMENT

3. Use the command window to bring up a plot window.
4. Use the command window to close the plot window
5. Explain what the "workspace image" is.
    a. Why is it important to not rely on the workspace image?
    b. Would you save your workspace image at the end of an R session? Why?

DEALING WITH WORKING DIRECTORIES

1. Use R to determine your current working directory. What is it?
2. Use R to change your current working directory.
    a. What is the core difference between how windows and Unix systems write file paths?
    b. Is there a way to write file paths that is consistent across operating systems?
3. There are two ways to list the files in your new working directory. What are they?
    a. Use both to determine the contents of your working directory and assign the output to an object.

BASIC OPERATIONS

1. Produce a numeric vector of length 1
    a. How do you know it has a length of 1? Use an appropriate function to find its length
2. Produce a character vector of length 1
    a. How do you know it has a length of 1? Use an appropriate function to find its length
3. Produce a logical vector of length 1
    a. How do you know it has a length of 1? Use an appropriate function to find its length

BASIC OPERATIONS

4. Without running any of this code, try to determine if the code will work. Give reasons.

```
a <- c( 1 , 2 , 3 , 4 , 5 )
```

```
a <- c( 1 , 'a' , 3 , 'b' , 5 )
```

```
a <- list( 1 , 2 , 3 , 4 , 5 )
```

```
a <- list( 1 , 'a' , 3 , 'b' , 5 )
```

```
a <- c( 1 , 2 , 3 , 4 , 5 )
b <- c( 'a' , 'b' , 'c' , 'd' , 'e')
a - b
```

```
a <- c( 1 , 2 , 3 , 4 , 5 , 6 )
b <- c( 1 , 2 , 3 )
c <- a + b
```

```
a <- c( 1 , 2 , 3 , 4 , 5 )
b <- c( 1 , 2 , 3 )
c <- a + b
```

BASIC OPERATIONS

5. Now run the code from exercise 4 to see if your predictions were correct. Were there are results that surprised you? If so, discuss why the code performed in the way that it did.

BASIC OPERATIONS

6. If velocity = distance / time, use these two numeric vectors to calculate velocity and assign it to an object

   ```
   distance <- sample(5:100)
   time <- sample(200:295)
   ```

   a. Do you answers match to your neighbours' answers? If not, why might this be?

BASIC OPERATIONS

7. How would you change a vector into an unordered factor? Write the code to do this.
8. How would you change a vector into an ordered factor? Write the code to do this.

DATA STRUCTURES

1. Why does the use of vectors help to make R good for data analysis?
2. How would you describe a vector?
3. How many dimensions does a vector have?
4. Can a factor be a vector? How would you check?

DATA STRUCTURES

5. What types of element can be contained in a list?
6. How would you describe a list?
7. How many dimensions does a list have?
8. Can a factor be a list? How would you check?

DATA STRUCTURES

9. What types of element can be contained in a matrix?
10. How would you describe a matrix?
11. How many dimensions does a matrix have?
12. Can a factor contained in a matrix? How would you check?

DATA STRUCTURES

13. What types of element can be contained in a data frame?
14. How would you describe a data frame?
15. How many dimensions does a data frame have?
16. Can a factor contained in a data frame? How would you check?

DATA STRUCTURES

This exercise requires input from everyone in your group. If everyone isn't at this point yet, either take a wee break or have a look at some of the other e xercises.

17. Go around your group collecting the following information, assigning the information to an appropriate vector:
    a. Name
    b. Age
    c. Coin toss result
    d. University
    e. Subject studied
18. In your group, discuss how to link this data together. What data structure would you use and how ould you do it?

FUNCTIONS

1. Go back to the slides that present some 'handy functions' and make sure that you know how to use all of them.
2. After having familiarised yourself with those functions, think about what the following code would do but don't run it.

```
view()
```

```
q( 'no' )
```

```
setwd( /home/vijay )
```

```
rm( list <- ls() )
```

```
class( F )
```

```
str( length )
```

FUNCTIONS

3. Run the code from exercise 2, did it do what you expected? If not, why?
4. Write a function that reverses a numeric vector and shows the result in a new window (not the command window).
5. The Pythagorean theorem states that, for a right-angle triangle, the square of the hypotenuse (the side opposite the right angle) is equal to the sum of the squares of the other two sides. Given that the formula for finding the square of the hypotenuse of a right-angle triangle is $a^2 + b^2 = c^2$ write a function that will return the hypotenuse given a and b.

GETTING HELP

1. Find the help page for the `c()` function using `?`
2. Find the help page for the `str()` function using the `help()` function
3. Find a function that will `transpose` a matrix. Use `??` to do this
   a. How do you use the function that you found?
4. Find a function that will calculate the `standard deviation` of a vector. Use the `help.search()` function to do this
   a. How do you use the function that you found?
5. Find a function that will save your script for you and use it to save your work so far.

PROJECT 1: CODING A SIMPLE CALCULATOR

To complete this project you will need to look up the usage of the following functions:

```
as.integer()
```

```
readline()
```

```
prompt()
```

```
switch()
```

```
print()
```

```
paste()
```

First find the help pages for these functions and learn how they work.

You will also need to refer to the section in this presentation about functions.

See the next slide for further details...

PROJECT 1: CODING A SIMPLE CALCULATOR

The code below is missing some vital elements, make adjustments so that the my_calc function takes two numbers from the user and returns the result of the requested calculation.

```
my_calc <- function(,){
select = as.integer(readline(prompt="Choose operator number[1(
operator <- switch(,"","","","")
result <- switch( ,  ,   ,   ,   )
print(paste(, operator, , "=", ))
}
```

PROJECT 2: CODING A MAD LIB GENERATOR

Adapt the code provided below to produce a mad lib generator.

```
# Hint
# a = adjective
# n = noun
# v = verb
    mad_lib <- function(){
story <- c('I walk through the', n1 ,'jungle. I take out my', a1, 'canteen. The
     a2, 'parrot with a', a3, n2, 'in its mouth right in front of me in th
     a4, 'trees! I gaze at its', a5, n3, '. A sudden sound awakes me from
     v1, 'in front of my head! I', v2, 'its ', a6, 'breath. I remember I h
     n4, 'that makes it go into a deep slumber! I ', v3, 'it away in front
     '. Yes it is eating it! I ', v4, 'away through the', a7, 'jungle.',
     'Phew; It has been an exciting day in the jungle.)'
cat(story)
    }
```

Once you've accomplished that, re-write the story with a tale of your own making.

Are there any functions you don't recognise? If there are, find out what they do?

PROJECT 3: HOW MUCH TO TIP?

Imagine that you're having a meal with some friends at a restaurant. You've come to the end of the meal and it is time to pay. Use R to calculate the following:

1. The total amount to tip when the agreed tip is 10% of the total bill
2. The total amount to tip per person when the agreed tip is 15% of the total bill
3. The amount that each person should tip based on what they spent individually

PROJECT 3: HOW MUCH TO TIP?

Think about what you need to know in order to calculate the tip for questions 1, 2 and 3.

4. Use that information to build a tip calculator that can perform the calculations in questions 1, 2 and 3.

NEXT WEEK

DAY 2 - FRIDAY 17TH MARCH

Begins to consider "real" data and how to manage, interpret and evaluate it in R.