

**Tutorial Week2**  
**STAT2008**

**Yunxi Hu (Lucy)**

u4957066@anu.edu.au

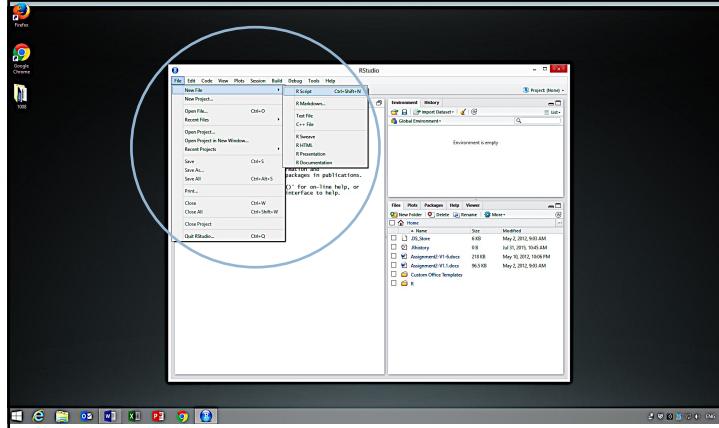
Research School of Finance, Actuarial Studies and Applied Statistics  
College of Business and Economics  
The Australian National University

### Install R Studio

- You should download both R and R studio and install R before installing your R studio
  - For R (Console):
    - ◆ <https://cran.r-project.org/>
  - For R studio:
    - ◆ <https://www.rstudio.com/products/rstudio/download/>
- For more information about installing:
  - Log on to Wattle - STAT2008 – Statistical Computing – Introduction to R

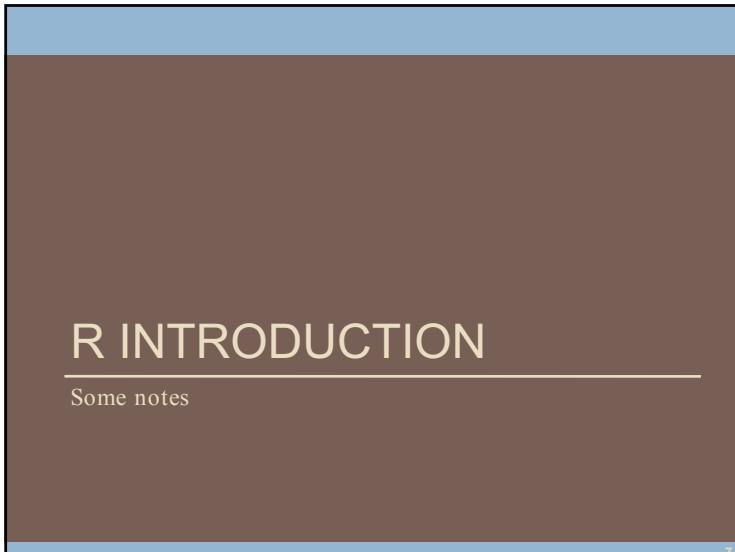
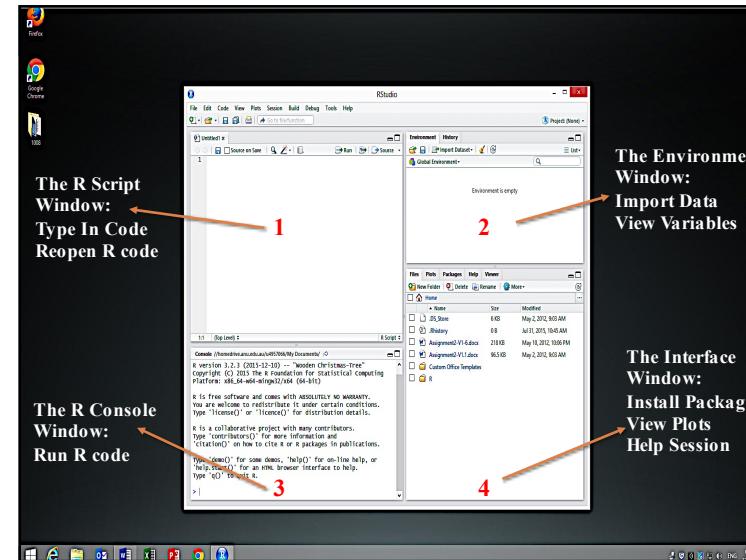
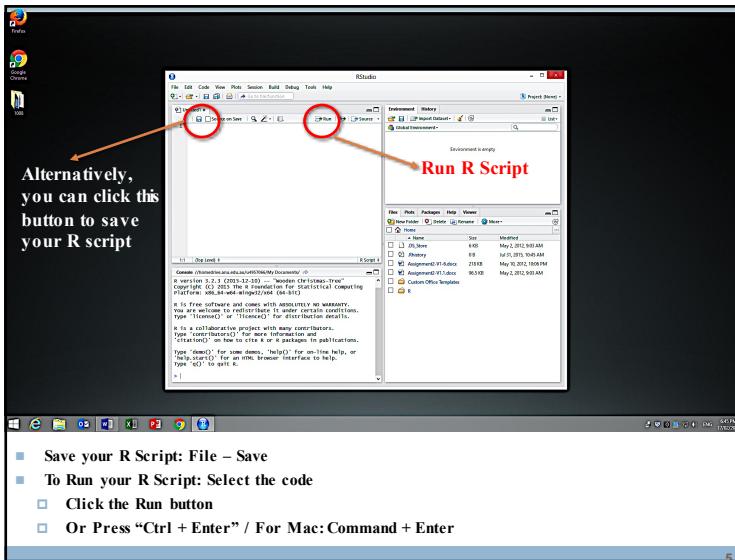
### Open R Studio

- Opening Computer
  - Log on to your computer with your Uni ID and your Password
- Download data from Wattle
  - Log on to Wattle (<https://wattlecourses.anu.edu.au>)
  - STAT2008 – Statistical Computing – Tutorial0- Download “worksheet2\_women” (DO NOT OPEN)
    - ◆ For Mac Users, you may find it's hard to download data by using “Safari”, you can use browsers such as “Firefox” and “Google Chrome” to download your data.
- Open R Studio [The following is for Windows 8 ONLY ]
  - Click  (bottom-left)
  - Top-Right => Search => “RStudio”
  - Or, bottom-left => arrow down  => RStudio



**Create R Script**

■ RStudio – File – New File – R Script



## Some Properties for R

- R Code:**
  - Case Sensitive
  - Separate by new line (Enter/Return) Or **Semicolon ;**
- R Code (Command)** are either
  - Expressions:**
    - a sequence of symbols interpreted by R
    - Eg:
      - > **2+2**
      - > **e+c**
  - Assignments** (of a value to a name)
    - Evaluate an expression
    - pass the value to a variable but the result is not automatically printed
    - Eg:
      - > **matrices**
      - > **value**
      - > **Happy <- 3+2**

[Happy = a name; 3+2= value; <- means to assign 3 to this name]
  - Object:** the entities that R creates and manipulates
    - Eg: variables, vectors, matrices, data sets, results, lists
    - If display the objects currently stored in R:
      - > **objects()**

## Some Properties for R

- <-**
  - Assign
- c(...)**
  - Combine arguments into a vector
- seq(x)**
  - Generate a sequence from 1 to x
- seq(from,to,by)**
  - Generate sequence with increment by
- from:to**
  - Generate sequence from...to...
- rep(x,times)**
  - Replicate x

## Data Structure of R

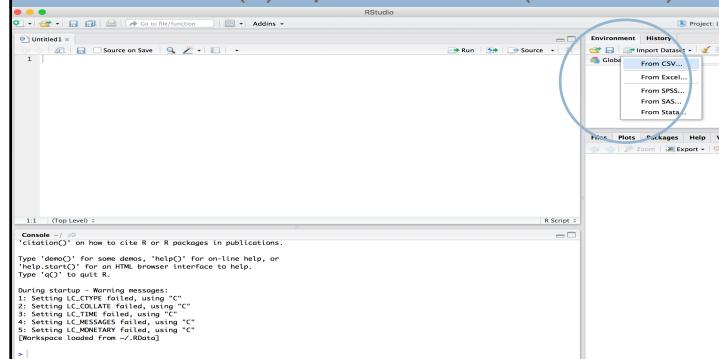
- Vector**
  - 1 single entity consisting of an ordered collection of numbers
- Matrices / Arrays**
  - Multi-dimensional vectors
- Lists**
  - Vector in which various elements need not to be of the same type
- Data frames**
  - Matrix-like structure
  - Columns can be of different type

(Source: ANU STA12008 Lecture Notes 2013 Semester 1)

## EXERCISE 2

Part (f) – How to open a CSV file

### Exercise 2 Part (e): Open a CSV file (Method 1)



#### Open the CSV file

- RStudio – Environment – Import Dataset – From CSV – Choose “[worksheet2\\_women](#)”  
(If R asks you whether you want to install this package now, choose yes)

**Exercise 2 Part (e): Open a CSV file (Method 1 continue)**

Open the CSV file

- Click “Browse” – Select the data “[worksheet2\\_women.csv](#)” you downloaded from Watt previously – Open – **Unclick** “First Row as Names”, **Rename it as** “ws2.women” - Import

13

**Exercise 2 Part (e): Open a CSV file (Method 1 continue)**

**Unclick** “First Row as Names”. **Rename it as** “ws2.women” - Import

14

**Exercise 2 Part (e): Open a CSV file (Method 1 continue)**

- Now you will see your worksheet (csv file) in your R studio!
- The whole process for Method 1 is equivalent to the process of typing the following code in Console (Run automatically) in R Script (but you need to click the Run button)
  - ws2.women <- read.csv("the location for your data/worksheet2\_women.csv", header=FALSE)**
  - NB: for me, it is:
    - ws2.women <- read.csv("~/Documents/ANU-Master/2017S1/STAT2008-2017S1/Tutorial/Tutorial10/worksheet2\_women.csv", header=FALSE)**
    - Since the first row in our worksheet is NOT the variable, we choose FALSE for the header
    - Use all “/” instead of “\” for your directory
    - The directory should be used by applying “ ” NOT “ ”
  - View(ws2.women)**
  - [NB: here we can rename the worksheet to ws2.women]

15

**Exercise 2 Part (e): Open a CSV file (Method 2 – setwd – openfile directly)**

Alternatively, we can set the working directory to the location that we stored our data “[worksheet2\\_women.csv](#)” previously by typing: **setwd("the location for your data")** in the console

- For me, it's **setwd("~/Documents/ANU-Master/2017S1/STAT2008-2017S1/Tutorial/Tutorial10 - Wk2 - Q2 + Intro/R")**
- Or, choose Session – Set working directory – Choose Directory [The location that you stored the worksheet “[worksheet2\\_women.csv](#)”]

16

**Exercise 2 Part (e): Open CSV file (Method 2 – `setwd` – open file directly)**

```

ws2.women <- read.csv("worksheet2_women.csv", header=FALSE)
View(ws2.women)
  
```

After setting the working directly, we can now type the following command in Console (Run automatically) in R Script (but need to click the Run button) to open our worksheet **"worksheet2\_women.csv"** [NB: here we can rename the worksheet]

- `ws2.women <- read.csv("worksheet2_women.csv", header=FALSE)`
- `View(ws2.women)`

NB: by using "setwd" command, there is no need for you to type the whole directory of your csv file as you have already set up the working directory!

**Exercise 2 Part (e): Open CSV file (Method 2 – `setwd` – open file directly)**

```

ws2.women <- read.csv("worksheet2_women.csv", header=FALSE)
View(ws2.women)
  
```

getwd()

- ◆ To know the current working directory

setwd("")

- ◆ To reset the working directory

View(ws2.women)

- ◆ To view ourworksheet in R Studio

# EXERCISE 2

Part (a)

**Copy and Paste R Script**

- Cope R Script from Wattle
- Log on to Wattle (<https://wattlecourses.anu.edu.au>)
- STAT2008 – Statistical Computing – Tutorial0- Open "Tutorial0ex2.R" File
- Copy and Paste this R Script to your local R Studio R Script Window

```

ws2.women <- read.csv("worksheet2_women.csv", header=FALSE)
View(ws2.women)
  
```

## Exercise 2 – Part (f): Run your R Script

- ❑ Select R code for **Part (a)**
  - ❑ Click the “Run” button
  - ❑ Or,
    - ❑ For Windows: Ctrl + Enter
    - ❑ For Mac: Command + Enter

21

## Exercise 2 – Part (a): Getting Help with R

- ❑ To run the following is equivalent to type the “topic” in the R interface window
    - ❑ **help(topic)**
  - ❑ Eg: Run **help(vector)** in your R Script, the interface window will then show you how **vector** function works.
  - ❑ This is equivalent to type “vector” in the R interface window

The screenshot shows the RStudio interface with the following details:

- Environment:** Shows objects like `v`, `vector`, `length`, `double`, `scalar`, `list`, `matrix`, `array`, `character`, `logical`, `complex`, `function`, `environment`, `vector`, `double`, `matrix`, `array`, `character`, `logical`, `complex`, `function`, `environment`.
- Data:** Shows a dataset named `worksheet2_women` with 38 observations and 2 variables.
- Code Editor:** The file `vector.R` contains the following code:
 

```

1 v <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23)
2 
3 # length(v)
4 
5 search()
6 
7 ## [1] "vector"
8 
```
- Console:** Displays the output of the code execution.
- Help:** A tooltip for the `length()` function is shown, pointing to the [RDocumentation](#) page for `length`.
- Documentation:** The [RDocumentation](#) page for `length` is open, showing the function's purpose, arguments, and usage examples.

1

## Exercise 2 – Part (a): Function

The screenshot shows the RStudio interface. The top menu bar includes 'File', 'Edit', 'File History', 'Source', 'Run', 'Help', and 'Project'. The left sidebar has sections for 'Environment', 'History', 'Global Environment', 'Data', 'Files', 'Plots', 'Packages', 'Help', 'View', and 'Documentation'. The main area has tabs for 'Console' (active), 'Script', and 'Repl'. The 'Console' tab shows R code and its output. The 'Script' tab shows an R script with several lines of code. A blue circle highlights the 'Console' tab and the output area.

- R Code:
    - Case Sensitive
    - When type `#` at the beginning of each line and run this line, this line will be treated as a comment, and R Console will no longer run this line's code (that's why you see the green fonts in your R Script since it is a comment ONLY rather than a function)

NB: This rules works for all cases even if you put `#` in the middle of one R code line (and all the words after `#` in the middle will be treated as a comment).

1

## Exercise 2 – Part (a): Getting Help with R

- ❑ The `help()` function and `?` help operator in R provide access to the documentation pages for R functions, data sets, and other objects, both for packages in the standard R distribution and for contributed packages. To access documentation for the standard `lm` (linear model) function, for example, enter the command `help(lm)` or `help("lm")`, or `?lm` or `?"lm"` (i.e., the quotes are optional).

(Source: <https://www.r-project.org/help.html> )

- Now, try to run the following codes



```
# Exercise 2 of the Introductory R Worksheet Tutorial 0
# Part (c)
# 
# T
# factor
# matrix()
# rep()
# rep()
# constant
# diag()
# 
# length()
# rep(C)
# rep(1)
# rep(vector())
# rep(0)
# 
# P
# Purge()
# 
# R Script
```

vector produces a vector of the given length and mode.  
as.vector, a generic, attempts to coerce its argument into a vector of mode mode. The default method for atomic vectors and lists is to return the result.  
is.vector returns TRUE if x is a vector of the specified mode having no attributes other than names. It returns FALSE otherwise.

Usage

```
vector(mode = "logical", length = 0)  
as.vector(x, mode = "any")  
is.vector(x, mode = "any")
```

Arguments

1

## Exercise 2 – Part (a): Functions

- c()**
    - Combine values into a vector or list
  - t()**
    - Matrix transpose
  - vector()**
    - A vector of the given length and mode
  - matrix()**
    - Creates a matrix from the given set of values

2

## EXERCISE 2

**Part (b)**

1

## Exercise 2 – Part (b)

The screenshot shows the RStudio interface with the following details:

- Code Editor:** The main pane displays R code from a file named "worksheet2\_women.R". The code includes various search operations and a definition of a variable "constant1".
- Environment Tab:** Shows the global environment with 108 objects and 2 variables.
- Console Tab:** Displays the command history and the output of the executed code, including the definition of "constant1".
- Plots Tab:** No plots are currently displayed.
- Packages Tab:** No packages are currently displayed.
- Help Tab:** No help topics are currently displayed.
- Viewer Tab:** No files are currently displayed.

27

## Exercise 2 – Part (b)

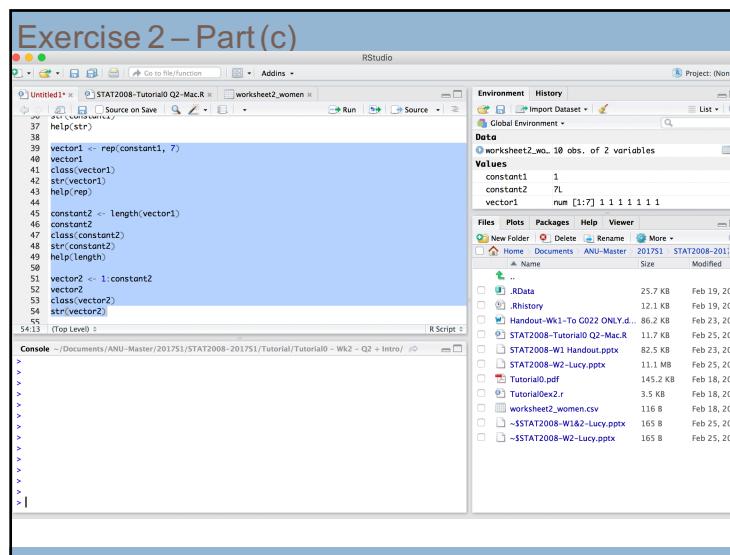
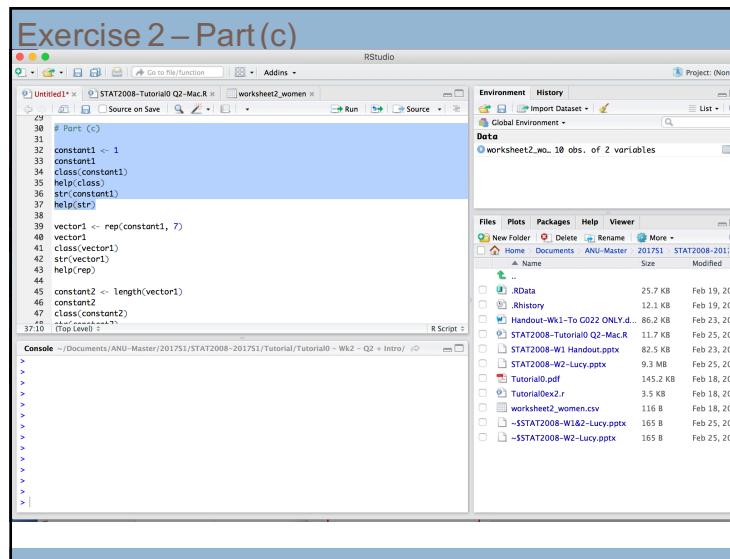
- search()**
    - # to see which packages are already included in your search path
  - ls()**
    - # list the contents of the packages in your search path
    - # to see all the objects defined
  - ls(pos=1)**
    - # By default, ls() lists the contents of position 1
  - ls(pos="package:datasets")**
    - # show you the sample datasets that come with standard R

1

## EXERCISE 2

### Part (c)

- constant1 <- 1**
    - # assign 1 to the object name “constant1”
    - # <- equals to =
  - constant1**
    - # assign 1 to the object name “constant1”
    - # <- equals to =
  - class(constant1)**
    - # class: check data type
  - help(class)**
  - str(constant1)**
    - # str: display the internal structure of an R object => number/integer..+ value
  - help(str)**



## Exercise 2 – Part (c)

- ❑ `vector1 <- rep(constant1, 7)`
  - ❑ # rep: repeat value of constant1 (which is 1) for 7 times => 1 vector contains 7 "1" ["1" is ONLY a number]
- ❑ `constant2 <- length(vector1)`
  - ❑ # length: calculate the length of vector 1
  - ❑ # which is 7
- ❑ `vector2 <- 1:constant2`
  - ❑ # : means to define the whole number from 1 to constant2 (which is 7)

33

## Exercise 2 – Part (c)

The screenshot shows the RStudio interface. The workspace pane displays variables: constant1 (1), constant2 (7L), and vector1 (a numeric vector of length 7 with all elements equal to 1). The file browser pane shows various files related to the tutorial, including R scripts and presentation files.

```

constant1 <- 1
constant2 <- 7L
vector1 <- c(1, 1, 1, 1, 1, 1, 1)

```

34

## Exercise 2 – Part (c)

- ❑ `vector3 <- c(0, 2^(1/3), sqrt(2), 2, 2^2, 2^3, 2^4)`
  - ❑ # c: Combine Values into a Vector or List
- ❑ `tm3 <- mean(vector3, trim=0.25)`
  - ❑ # take the average of vector3 when get rid of the LB and UB @ area equals to 0.25
  - ❑ LB,UB = lower bound, upper bound
  - ❑ # another example
    - ❑ `a<-c(1,4,8,60,80,90,100)`
      - ❑ # l= 1st obs, 100=7th obs
    - ❑ `mean(a, trim=0.2)`
      - ❑ # For LB: n=7, LHS area=0.2 => position=(7+1)\* 2/100= 1.6 => LB = 1.6th observation = 1+(4-1)\*0.6=2.8
      - ❑ # For UB: n=7, RHS area=0.2 => position = (7+1)\* 2/100= 1.6 => 7+1-1.6=6.4th observation => UB= 90+(100-90)\*0.4=94
      - ❑ # average: (2.8+4+8+60+80+90+94)/7
  - ❑ `vector4 <- seq(28, 1, -3.5)`
    - ❑ # seq: generate a sequence from 28 to 1, the distance between intervals is -3.5
    - ❑ # Rule: seq(from,to,by)
  - ❑ `seq(x)`
    - ❑ #generate sequence from 1 to x

35

## Exercise 2 – Part (c)

The screenshot shows the RStudio interface. The workspace pane displays variables: constant1 (1), constant2 (7L), and tm3 (3.348269245359). The help documentation pane shows the details for the seq() function, including its usage and examples.

```

constant1 <- 1
constant2 <- 7L
tm3 <- 3.348269245359

```

**seq**(...)

Generate regular sequences. `seq` is a standard generic with a default method. `seq`.`int` is a primitive which is much faster but has a few restrictions. `seq_along` and `seq_len` are very fast primitives for two common cases.

**Usage**

```

seq(...)

# Default S3 method:
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL, along.with = NULL, ...)

seq.int(from, to, by, length.out, along.with, ...)

seq_along(along.with)
seq_len(length.out)

```

36

## Exercise 2 – Part (c)

- vector4 <- vector4[1:7]**
  - # take out the first 7 elements from "vector4"
  - # **from:to** means to generate sequence from...to
- matrix1 <-cbind(vector1, vector2, vector3, vector4)**
  - # Create a matrix
  - # cbind = different columns bind [c=column + bind]
  - # cbind (c(Col1Row1,Col1Row2,Col1Row3),c(Col2Row1,Col2Row2,Col2Row3))
- constant3 <- ncol(matrix1)**
  - #ncol = number of rows
  - #nrow = number of columns

37

## Exercise 2 – Part (c)

```

RStudio
Project: (None)

Environment History
matrix1 num [1:7, 1:4] 1 1 1 1 1 1 1 2 3 ...
constant1 1
constant2 7L
constant3 4L

Values
nrow(x) return the number of rows or columns present in x. xcols and
ncol do the same treating a vector as 1-column matrix.
nrow(base)
R Documentation

The Number of Rows/Columns of an Array
Description
nrow and ncol return the number of rows or columns present in x. xcols and
ncol do the same treating a vector as 1-column matrix.

Usage
nrow(x)
ncol(x)
xcols(x)
ncol(x)

Arguments
x a vector, array or data frame

Value

```

R Script : /Documents/ANU-Master/201751/STAT2008-201751/Tutorial/Tutorial0 - Wk2 - Q2 + Intro.R

```

93 matrix2 <- rbind(vector1, vector2, vector3, vector4)
94
95 class(matrix2)
96 str(matrix2)
97
98 matrix3 <- matrix1 - t(matrix2)
99
100 class(matrix3)
101 str(matrix3)
102
103 matrix4 <- matrix1 %*% matrix2
104 matrix4
105 class(matrix4)
106 str(matrix4)
107
108 vector4d <- dim(matrix4)
109 vector4d
110 class(vector4d)
111 str(vector4d)
112 help(dim)
113
114 (Top Level) :

```

Console : /Documents/ANU-Master/201751/STAT2008-201751/Tutorial/Tutorial0 - Wk2 - Q2 + Intro.R

```

... .5 : NULL
... .6 : <environment: 0x0> "vector1" "vector2" "vector3" "vector4"
> ncol(matrix1)
> constant3 <- ncol(matrix1) #ncol = number of rows; nrow = number of columns
> constant3
[1] 4
[2] <class(constant3)>
[3] <integer>
> str(constant3)
int [1:1]
> help(ncol)
> |

```

38

## Exercise 2 – Part (c)

- matrix2 <- rbind(vector1, vector2, vector3, vector4)**
  - # rbind = different rows bind [r=row + bind]
  - # rbind (c(Col1Row1,Col2Row1,Col3Row1),c(Col1Row2,Col2Row2,Col3Row2))
- matrix3 <- matrix1 - t(matrix2)**
  - # t = transpose of 1 matrix
- matrix4 <- matrix1 %\*% matrix2**
  - # "A %\*% B" means want to use A MATRIX \* B MATRIX;
  - # if we only use "A\*B" - use EACH element in matrix A to multiply EACH element in matrix B
- vector4d <- dim(matrix4)**
  - # dim = dimension

39

## Exercise 2 – Part (c)

```

RStudio
Project: (None)

Environment History
matrix1 num [1:7, 1:4] 1 1 1 1 1 1 1 2 3 ...
matrix2 num [1:7, 1:4] 0 0 0 0 0 0 0 0 0 ...
matrix3 num [1:7, 1:7] 786.689 592.495 398.728 ...
matrix4 num [1:4, 1:4] 7 28 32.7 122.5 28 ...
worksheet2.women10 obs. of 2 variables
Values
nrow(x) return the number of rows or columns present in x. xcols and
ncol do the same treating a vector as 1-column matrix.
nrow(base)
R Documentation

The Number of Rows/Columns of an Array
Description
nrow and ncol return the number of rows or columns present in x. xcols and
ncol do the same treating a vector as 1-column matrix.

Usage
nrow(x)
ncol(x)
xcols(x)
ncol(x)

Arguments
x a vector, array or data frame

Value

```

R Script : /Documents/ANU-Master/201751/STAT2008-201751/Tutorial/Tutorial0 - Wk2 - Q2 + Intro.R

```

114 matrix5 <- matrix2 %*% matrix1
115 matrix5
116 class(matrix5)
117 str(matrix5)
118
119 vector5d <- dim(matrix5)
120 vector5d
121 class(vector5d)
122 str(vector5d)
123
124 # Matrix multiplication is not transitive; A %*% B does not equal B %*% A
125
126 vector6 <- apply(matrix1, 2, mean)
127 vector6
128 class(vector6)
129 str(vector6)
130
131 help(apply)
132 # apply() has been used to calculate the column means of matrix1
133
134 matrix6 <- sweep(matrix1, 2, apply(matrix1, 2, mean))
135 matrix6
136 class(matrix6)
137 str(matrix6)
138 help(sweep)
139 # sweep() and apply() have been used to "mean-correct" the columns of matrix1
140
141 matrix6[3:5,3] <- 0
142 matrix6
143 class(matrix6)
144 str(matrix6)
145 # this sets the 3rd, 4th and 5th rows of the 3rd column of matrix6 to 0
146
147 (Top Level) :

```

Console : /Documents/ANU-Master/201751/STAT2008-201751/Tutorial/Tutorial0 - Wk2 - Q2 + Intro.R

```

... > matrix5 <- matrix2 %*% matrix1
> vector5d <- dim(matrix5)

```

40

## Exercise 2 – Part (c)

- ❑ **vector6 <- apply(matrix1, 2, mean)**
    - ❑ # apply() has been used to calculate the column means of matrix 1
    - ❑ # apply (where to apply, 1=apply to row and 2=apply to column, what function to apply) => to calculate the mean of each column for matrix 1
  - ❑ **matrix6 <- sweep(matrix1, 2, apply(matrix1, 2, mean))**
    - ❑ # sweep() and apply() have been used to "mean-correct" the columns of matrix 1
    - ❑ # sweep (where to apply, 1=apply to row and 2=apply to column, what function to apply)
    - ❑ # here we use "apply(matrix1, 2, mean)" to calculate the average of each column, then use each column element of "matrix 1" to minus this column's average

4

## Exercise 2 – Part (c)

1

## Exercise 2 – Part (c)

- matrix6[3:5,3] <- 0**
    - # assign 0 to [3:5,3] over row 3 to 5, and column 3
  - prod(vector3)**
    - # get the product of all element from vector 3
  - order(vector4)**
    - # get the order of vector 4 based on an ascending order
  - sort(vector4)**
    - # get the ascending order of vector 4
  - rm(hello)**
    - # define "hello" as the a object before, and now remove it
  - ls()**
    - # to see all the objects defined
  - objects()**
    - # to see all the objects defined

47

## EXERCISE 2

---

**Part (d)**

## Exercise 2 – Part (d)

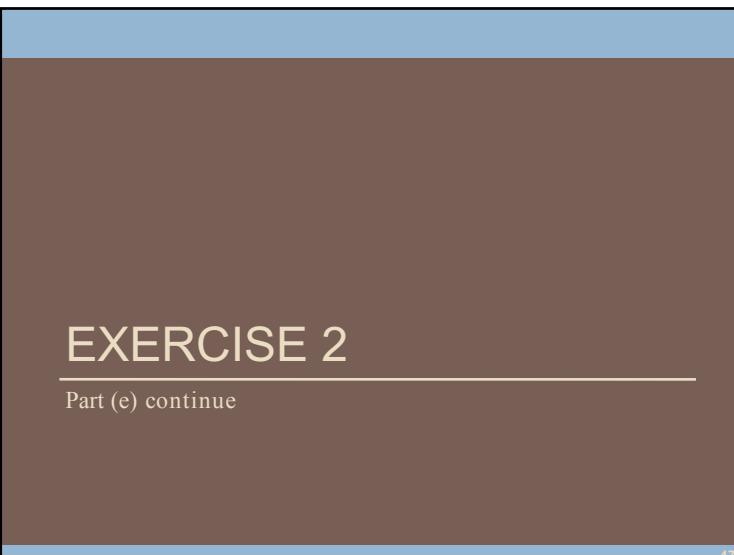
The screenshot shows the RStudio interface with the following details:

- Top Bar:** Contains tabs for "Untitled1", "STAT2008-Tutorial0 Q2-Mac.R", "worksheet2\_women", "Run", "Source", "Environment", "History", and "Project (None)".
- Code Editor:** Displays R code for generating matrices and vectors.
- Global Environment:** Shows a table of variables:

Variable	Type	Value
matrix1	matrix	[3x7]
matrix2	matrix	[3x4]
matrix3	matrix	[3x7]
matrix4	matrix	[3x7]
matrix5	matrix	[3x4]
matrix6	matrix	[3x7]
- Files Tab:** Lists "R Scripts", "Plots", "Packages", "Help", "Viewer", and "R Documentation".
- Environment Tab:** Shows sorting and ordering vectors.
- Help Bar:** Includes "R: Sorting or Ordering Vectors" and "sort (base)".
- Bottom Bar:** Shows the R version as "R 3.6.1" and the operating system as "Windows 10".

## Exercise 2 – Part (d)

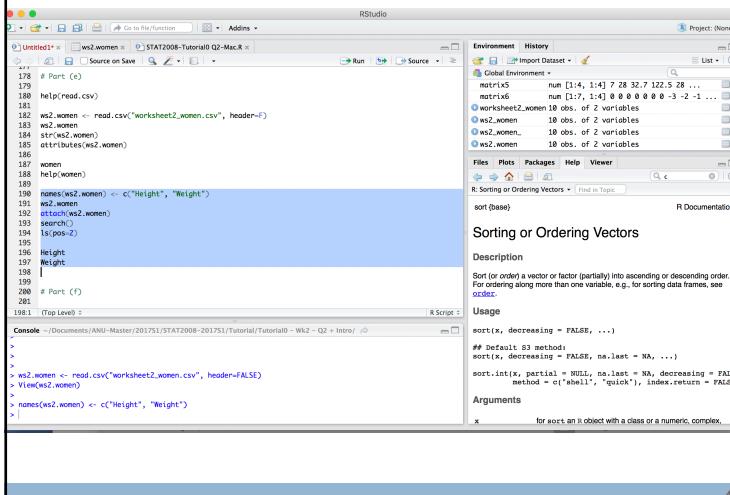
- ❑ # here, "function()" means for this function, there is an empty input ["()"] means the input
  - ❑ In general, we can use
    - ❑ **function (input) {output}**
    - ❑ for example:
      - ❑ # **happy=function(x) {x+10}**
      - ❑ # x is the input, and x+10 is the output
      - ❑ # **happy(2)**
      - ❑ # () means to make input (x) equal to 2, and then run the function of "x+10", and get the output :2+10=12
  - ❑ NB
    - ❑ # "**happy**" only shows the definition of happy, which is **function(x) {x+10}**
    - ❑ # "**happy ()**" runs this x+10 function



## Exercise 2 – Part (e)

- The codes below is equivalent to what I have mentioned earlier, the 2<sup>nd</sup> method of opening a csv file

## Exercise 2 – Part (e)



```

178 # Port (c)
179
180 help(read.csv)
181
182 ws2.women <- read.csv("worksheet2_women.csv", header=FALSE)
183 ws2.women
184 str(ws2.women)
185 attributes(ws2.women)
186
187 women
188 head(women)
189
190 names(ws2.women) <- c("Height", "Weight")
191 ws2.women
192 attach(ws2.women)
193 search()
194 ls(pos=2)
195
196 Height
197 Weight
198
199 |
200 # Port (f)
201
202
188.1 (Top Level) R Script

```

Console output:

```

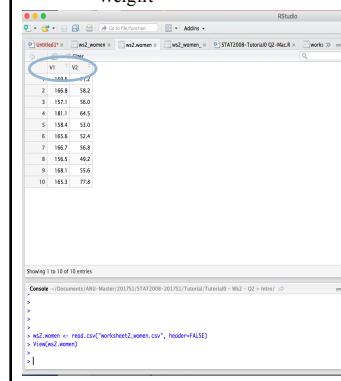
> 
> 
> 
> ws2.women <- read.csv("worksheet2_women.csv", header=FALSE)
> str(ws2.women)
> attributes(ws2.women)
> names(ws2.women) <- c("Height", "Weight")
> 

Arguments
x           for a sort an R object with a class or a numeric, complex

```

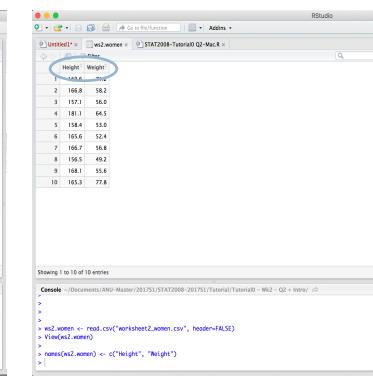
## Exercise 2 – Part (e)

- ❑ **names(ws2.women) <- c("Height", "Weight")**
- ❑ # rename each variable (each column)
- ❑ # previous we see V1 and V2 as our variable names, now they change to Height and Weight



Global Environment pane showing two variables:

V1	V2
2	164.8
3	157.1
4	181.1
5	158.4
6	163.8
7	164.7
8	156.5
9	168.1
10	165.3



Global Environment pane showing two variables:

Height	Weight
2	164.8
3	157.1
4	181.1
5	158.4
6	163.8
7	164.7
8	156.5
9	168.1
10	165.3

## Exercise 2 – Part (e)

- ❑ **attach(ws2.women)**
- ❑ # we can directly apply the variable names from our worksheet, i.e. "Height, Weight" in R Studio
- ❑ # When there is no such code, if we want to apply the "Height" column variable from our worksheet, we have to type
  - ❑ **ws2.women\$Height**
- ❑ NB: \$ means to apply the variable name from the worksheet

# EXERCISE 2

Part (f) continue

## Exercise 2 – Part (f)

The screenshot shows the RStudio interface with the following details:

- Script Editor:** Displays R code for generating summary statistics, histograms, boxplots, and scatter plots for the 'ws2.women' dataset.
- Console:** Shows the execution of the R code, including the creation of a linear model 'lm(Weight ~ Height)' and its summary.
- Environment:** Shows the global environment with objects like 'heights', 'ws2.women', and 'women.lm'.
- Help:** A tooltip for the 'lsnames' function is visible, explaining its arguments: 'name' (the environment to search), 'pos' (an alternative argument to 'envir'), 'envir' (an alternative argument to 'env'), and 'pattern' (an optional regular expression).

## Exercise 2 – Part (f)

- ❑ **summary(ws2.women)**
- ❑ # statistic summary for each variable in our worksheet
- ❑ **hist(Height)**
- ❑ # histogram for Height
- ❑ **boxplot(Height, Weight)**
- ❑ # boxplot for Height and Weight
- ❑ **plot(ws2.women)**
- ❑ # scatter plot for Height and Weight
- ❑ **women.lm <- lm(Weight ~ Height)**
- ❑ # fit a Simple linear regression, Y=Weight, X=Height
- ❑ **attributes(women.lm)**
- ❑ # check all the properties (names) for this regression
- ❑ **plot(women.lm)**
- ❑ # residual plots for Y versus X
- ❑ **anova(women.lm)**
- ❑ # anova for Y versus X
- ❑ **summary(women.lm)**
- ❑ # summary for Y versus X
- ❑ **plot(Height, Weight, xlab="Height (in cm)", ylab="Weight (in kg)")**
- ❑ # scatter plot for Y versus X
- ❑ **title("Sample of 10 women")**
- ❑ # add title
- ❑ **abline(women.lm\$coef)**
- ❑ # add the regression function line with its coefficients (slope and intercept) to the scatter plot

## Extra Notes

- ❑ **reg.y.on.x <- lsfit(Height, Weight)**
- ❑ # LSE (least squares regression)
  - ❑ # include
    - ❑ A vector of the parameter estimates (bo,b1)
    - ❑ **reg.y.on.x\$coef**
  - ❑ Residuals
    - ❑ **reg.y.on.x\$resid**

# EXERCISE 1

Extra Notes

## Some Notes for Exercise 1

- **`sqrt(4)`**
  - # square root of 4
- **`abs(4)`**
  - # absolute value of 4
- **`log(4)`**
  - # ln(4) since the default for the `log()` function is to compute logarithms to the base e
  - # for common logarithms (logs to the base 10), you can use `log(0)` or `log(number, base=10)`
  - # for binary logarithms use `log2(number, base=2)`
- **`4 > 2`**
  - # whether "4>2" is true?
- **`4 < 2 || 4 > 2`**
  - # whether "4<2" OR "4>2" is true?
  - # & and && indicate logical AND
  - # | and || indicate logical OR.
- **`rm()`**
  - # to remove the unwanted object defined previously
  - Eg: if we defined `hello` previously, then type `rm(hello)` means to remove the object `hello`
- **`q()`**
  - # to quit R, which equals to choose File - Quit R studio
- **`ls()`**
  - List objects in workspace
- **`rm(list = ls())`**
  - Remove all objects from workspace
- **`saveimages()`**
  - Saves workspace
- **`#`**
  - comments

57