Background

- This assignment is worth 15% of your course grade. You have to complete it yourself.
- In answering the questions properly, you will need to at-the-least consult the lecture videos, Chapter 2 in *R_introduction.pdf* (and any additional sections mentioned in question hints below) and the R help files (using `?`).
- **Deadline: April 11, any time until 7am the next day (provisional - to be confirmed).**
- Unless alternative MyLO instructions are given in class, email me your answer script directly (ian.hunt@utas.edu.au); in the subject line put "KLA assignment 1" and then your full name.

Instructions

- The assignment is to be answered in a single R script: please put all your code into `.R` file with "A1_" and then your student ID as the name.
- When I ask "Set $somevariable =$ ... " I mean create an R variable in a script and assign it the value described by "...". For example the correct answer to "Set $z =$ five times ten" is to write in the script "`z<-5*10`".
- **Ensure that your variable names are EXACTLY the same as requested (including upper or lower CaSe). Any answer with a variable typo will score zero.**
- The instructions that directly follow "**Required:** " must be followed in order to receive full marks.
- There are three sections. Each question is worth the same. Please begin each section with a separate line starting with a double hash and name of the section e.g. "`## Section 1`". Within each section please label each individual answer with a line beginning with single comment e.g. "`# 1.2`".
- You (and me) should be able to run your *entire* script in one go, without any errors. If you have questions that you cannot answer which continue to generate errors, then comment them out (using `#`).
- Make sure that your answers are your own. In other words, if you get help from someone then do not blindly copy their answer or make cosmetic changes: make sure that your code works! If in doubt you can ask me.

Required: Generate all the required data variables in your R script and leave them there. Your script should work when you close R and RStudio, then open it again and run everything.

Required: For the avoidance of doubt: your answers should be "dynamic code" rather than "hard-coded numbers". For example, if there is a variable holding some data and your answer uses that data, then the code in your answer should refer directly to the variable holding the data. See the example below.

```
# assume the question is: "set zmu = the average of z, where z<-c(1,3)".

# first make z
z<-c(1,3)

# then base zmu on z (dynamic because z can change and your code still works)
zmu<-mean(z)    # correct answer (full marks)

# this would be incorrect (right value but would be static with respect to z)
zmu<-2          #  wrong answer (zero marks)
```

Hint: You should check your answers by typing the variable into the R console. I am happy for you to include this in your R script (as long as the code that creates the variable is in the script too).

```
# assume the question is: "set zmu = the average of z, where z<-c(1,3)".
z<-c(1,3)
zmu<-mean(z)    # correct answer (full marks)
zmu             # run this to check your answer (you can leave this if you want)
```

1 Basic Vectors

This section tests basic numerical skills in R, which are required for all applications.

1.1

Set $x1 =$ a vector with the integers from 2 to 55 in it.

Required: One line of code only. Do not type in *all* the numbers — generate a sequence of numbers automatically.

```
x1<-2:55
```

1.2

Set $x2sum =$ a vector with two elements: the mean and sd of $x2$, where $x2$ is as follows:

```
x2<-c(12.3275,14.082,12.074)
```

Hint: Make $x2$ first.

```
x2<-c(12.3275,14.082,12.074)
x2sum<-c(mean(x2),sd(x2))
x2sum
```

1.3

Set $x3sum =$ a vector with two elements: the mean and sd of the numbers in $x3$ that are not missing values, where $x3$ is as follows:

```
x3<-c(12.3275,14.082,12.074,NA,8.463333333,7.557333333,7.197333333,NA)
```

Required: Generate $x3$ first and then calculate the data summaries directly from $x3$.
Hint: Check the summary functions in the help file for how to leave out missing values.

```
x3<-c(12.3275,14.082,12.074,NA,8.463333333,7.557333333,7.197333333,NA)
x3sum<-c(mean(x3,na.rm=TRUE),sd(x3,na.rm=TRUE))
x3sum
```

1.4

Using the same $x3$ as above, set $x3miss$ = a vector with two elements: the number of non-missing values in $x3$ and the number of missing values in $x3$.

Required: Generate $x3$ first and then calculate the data summaries directly from $x3$.
Hint: The sum of the false values in `c(TRUE,FALSE)` is `sum(!c(TRUE,FALSE))`.
Hint: See Section 7.4 for a function that checks for `NA` values in vectors and Section 7.3 for the R code for "NOT" which turns `TRUE` into `FALSE`. Run some toy examples and then return to your answer for this question.

```
x3<-c(12.3275,14.082,12.074,NA,8.463333333,7.557333333,7.197333333,NA)
x3miss<-c(sum(!is.na(x3)),sum(is.na(x3)))
x3miss
```

1.5

Let $x4$ be

```
x4<-c(12,15.55,2.00)
```

Then set the names property of $x4$ to have three elements — the first should be "n", the second "mean" and the third "sd".

```
x4<-c(12,15.55,2.00)
names(x4)<-c("n","mean","sd")
x4
```

Hint: Check that your name allocation has worked by looking at $x4$ after you change the names.
Hint: See Section 4.22 in the R notes.

2 Basic Data Frames and Lists

This section tests basic skills with data frames and lists.

For this section first include the following code in your script and run it.

```
set.seed(99)
xlabs<- sample(c("high","low"),92, replace=TRUE)
xmeasurement<-rpois(92,10)
xplant<-data.frame(plantID=1:length(xlabs), fert=xlabs,height=xmeasurement)
```

Hint: Be sure to check, as you go, what is in your data frames and lists using `str` and `View`.

2.1

Set $y =$ the data frame called $xplant$. Then change the name of the second column (fert) of y to be called “treatment”.

Required: Don’t change $xplant$.
Hint: The column name vector for a data frame (or matrix) can be found with the function `colnames`; just like with `names` for vectors, you can replace or update any element of `colnames` (or all of them at once) as you wish.

```
y<-xplant
colnames(y) # find which element is "treatment" and then change manually
colnames(y)[2]<-"treatment"
colnames(y)
```

2.2

Add a new column to $xplant$ called “xlow” which identifies (using logical values) which plants (each row is a plant) have a height lower than or equal to median of the overall heights.

Hint: See Section 4.24 of the R notes for adding columns to data frames.
Hint: Rs logical values are `TRUE` or `FALSE`.

```
xplant$xlow<-xplant$height<=median(xplant$height)
str(xplant)
```

2.3

Set $xplantlist =$ a list that has two elements, both based on $xplant$. Let one element of $xplantlist$ hold all the data in $xplant$ for which the fert column is ‘high’; and let the other element have the data for which the fert column is ‘low’.

Required: Use only one line of R code.
Hint: Your list should hold two separate data frames. You do not need to use `==` or direct comparisons to do this.

```
xplantlist<-split(xplant,xplant$fert)
length(xplantlist) # check for 2
xplantlist # take a look
```

2.4

Set $xplantsum =$ a data frame that reports both the sample size (i.e. the count of the numbers), average height and maximum height for both levels of fert (‘high’ and ‘low’).

Required: Use the function `aggregate`.
Hint: You can create a summary statistic function inside `aggregate` itself, or make a custom function that you run before the line of code with `aggregate`.

```
fsum<-function(y){
out<-c(length(y),mean(y),max(y))
names(out)<-c("n","mean","max")
out
}
xplantsum<-aggregate(height~fert,data=xplant,FUN=function(i){fsum(i)})
xplantsum
```

2.5

Set $xhigh20 =$ the average height of the plants that have a plantID no greater than 20 (i.e. the first 20 plants) and a fert level of ‘high’.

Required: Use the `&` symbol in R to combine two logical conditions (see Section 7.3 and Section 7.8 of the R notes).
Hint: You might first want to create an auxiliary variable holding the subset of data to summarize, and then take the mean of this variable.
Hint: Looking up R help for symbols requires using rabbit ears e.g. `?"&"`.

```
xplantsub<-xplant$height[xplant$plantID<=20&xplant$fert=="high"]
xhigh20<-mean(xplantsub)
xhigh20
```

3 Files and folders

This section tests basic skills with file paths and folders.

For this section first include the following code in your script and run it (ensure your computer has access to the internet at the time).

```
# read in raw file from GITHUB, formatted with columns set by a comma
xurl<-"https://raw.githubusercontent.com/STATShunt/tia/"
xurl<-paste0(xurl,"master/data/cannabis/data_buds.csv")
xbud<-read.csv(file=xurl,check.names = FALSE,stringsAsFactors =TRUE)
```

Hint: Be sure to check, as you go, what is in your data frames with `str` and `View`.

3.1

Set $mydir$ = your working directory. Then add the output from this variable (from your machine) in a commented line of code.

Hint: Check Section 3.2 of the R notes.
Hint: On my machine, I would add the following comment to my answer:

`# "C:/Users/hunti/OneDrive - University of Tasmania/Documents/"`

Hint: On your machine the default working directory (which is set when you installed R) will be different. And if you are using a computer with fruit on the lid, then you will not have drive letters (like "C:") in your file paths.

```
mydir<-getwd()
mydir
# "C:/Users/hunti/OneDrive - University of Tasmania/Documents/"  in my case ...
```

3.2

Write out into your working directory the data in $xbud$, calling the file "my-bud.csv".
Required: Include the R code to do this in your script (do not send me the csv file).
Required: Ensure any column names remain in the new file.
Hint: Open your csv file in Excel to check it.
Hint: See Section 3.8 and Section 3.9 in the R notes for reading and writing data files, but you can just use `write.csv`.

```
write.csv(xbud,"mybud.csv")
#or
write.csv(xbud,"mybud.csv",rownames=FALSE) #stops the first column of index numbers
```

3.3

Outside of RStudio, find the data file that you saved in your working directory and open it with Microsoft Excel. Manually delete all the columns except “Sample”, “treatment”, “weight” and “CBDA”. Save this as a new file called “xbudmini.csv” in your R working directory. Back in RStudio and your answer script, set $xbudmini =$ the data from this new csv file.

Required: Make use of `read.csv` in your answer.
Required: Include the R code to do this in your script (do not send me the csv file).
Hint: If you were unable to answer question 3.2, then for this question you can download the original data file straight from the github site (find data_buds.csv within the zip folder: `https://github.com/STATShunt/tia/blob/master/data/cannabis/all_data_cannabis.zip`)

```
# first create the new data file in excel
# column of index numbers may need deleting
# save it as csv format, with a new name
# then upload back to R with the following code
xbudmini<-read.csv("xbudmini.csv")
```

3.4

Produce a scatter plot of “CBDA” (on the x-axis) versus “weight” (on the y-axis) from $xbud$. Save the chart in a “.png” format to your working directory. Call the chart “cbdaweight.png”

Hint: Check out Section 9.9 (both pages) in the R notes for saving plots. Be sure to check that the plot appears in your working directory.

```
png("cbdaweight.png", height=900,width=900) # pixel units for height/width
plot(xbud$CBDA,xbud$weight)
dev.off()
```

3.5

Make a new folder somewhere on your computer (not your current R working directory). Then in Rstudio save a “new project” in that folder, close RStudio, go to the new directory and the open your newly created Rproject file. Report the new R working directory when RStudio opens this way.

Required: No R code is needed in your script; just include the file path of the new working directory (from your machine) in a commented line of code (using #) in your answer script.
Hint: This question is just an exercise in setting up a convenient space on your hard drive to use RStudio — you can store your scripts and data files in their own folders and simply work on them by opening up an Rproject file in the same folder.
Hint: `https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects`.
Hint: Check Section 3.2 of the R notes and question 3.1 above.

```
# i will just have to trust you!
```