Lab

Data Frames and R Markdown

Aim

- To learn to create and manipulate data frames.
- To introduce R markdown. This will enable you to:
    - Test your code, check your results and add explanations as you develop your R program.
    - Produce documents where your code, results and explanations appear together.

Note: before you attempt this lab you should look at the introductory lab from module CMM535.

If you are new to R, concentrate on learning about data frames. You will use R markdown for your labs and will need to learn it but you do not need to know much to be able to run code.

Data Frames

Data frames are used to hold data in tabular form. Normally, the columns hold attributes (features) and each row represents an instance (observation). In some modules you will be using data frames to hold your datasets.
To obtain a dataset you can

- Create your own in R (from vectors or matrices)
- Load from a file (usually in csv format)
- Use a predefined dataset already in R

Creating a data frame from vectors

You can create a data frame from vectors. E.g.

```
caseID <- c("C1", "C2", "C3", "C4")
problem <- c("theft", "robbery", "assault", "robbery")
solved <- c(TRUE, TRUE, FALSE, TRUE)
courtDays <- c(3,2,1,14)
courtCases <-
       data.frame(caseID, problem, solved, courtDays, stringsAsFactors=T)
```

The option `stringsAsFactors=T` ensures that all strings are treated as Factors. If this option is not set, the data will be of type character.

Note: operator "<-" is equivalent to "=".

To inspect the data frame use `View()` *[note that it is uppercase 'V']*

```
View(courtCases)
```

Creating a data from from a matrix (if you don't know matrices, skip this section)

Given the following matrix of 18 numbers 1 to 18 with 6 rows and 3 columns

```
mynums <- matrix(data= c(1:18), nrow=6,ncol=3)
mynums
     [,1] [,2] [,3]
[1,]    1    7   13
[2,]    2    8   14
[3,]    3    9   15
[4,]    4   10   16
[5,]    5   11   17
[6,]    6   12   18
```

It can be converted to a data frame using `as.data.frame()`

```
mynumsDF <- as.data.frame(mynums)
```

Note: the option `StringsAsFactors=T` can be used with data of type string.

Adding rows

Given the following 2 dataframes `my1stNumsDF` and `my2ndNums`

```
my1stNums <- matrix(data= c(1:18), nrow=6,ncol=3)
my1stNumsDF <- as.data.frame(my1stNums, stringsAsFactors=T)
my2ndNums <- matrix(data= c(24:47), nrow=8,ncol=3)
my2ndNumsDF <- as.data.frame(my2ndNums, stringsAsFactors=T)
```

These 2 data frames can be put together by binding rows with `rbind()`

```
allNumsDF <- rbind(my1stNumsDF,my2ndNumsDF)
```

You can check that the binding has occurred using `View()` or the name of the new data frame.

```
View(allNumsDF)
```

Adding columns

Given the following dataset

```
someNums <- data.frame(c(1:10), c(101:110))
names(someNums) <- c("firstNum","secondNum")
```

To create a final column called addition with the sum of the 2 columns

```
someNums$addition <- someNums$firstNum + someNums$secondNum
```

To create an extra column stating if the number in the 1st column is > 4

```
someNums$greater4 <- someNums$firstNum >4
```

Check the results using the View function.

Creating a data frame from a file

Download the dataset.zip file (containing *consumptionDM.csv and labor.csv)* from CampusMoodle. Save it on your H drive and unzip it.

We want to obtain a data frame from the data in consumptionDMData. First, inspect this dataset using excel (or a similar program). You will see that the dataset has headers (i.e. the first row contains the names of the columns. Once you have inspected the dataset, close excel.

On the console window in R Studio, type the following

```
consumptionDMData <-
      read.csv("consumptionDM.csv", header=T, stringsAsFactors=T)
```

or, if your file is not in the same working directory as the one you are in (within RStudio) you should specify the path to the file, i.e.

```
consumptionDMData <-
      read.csv("<pathTofile>/consumptionDM.csv", header=T,
                  stringsAsFactors=T)
```

This reads the consumptionDM dataset in comma separated form and assigns it to variable *consumptionDMData*. This data is of type data.frame.

In the previous example, `header = T` (or header = TRUE)
This means that the first line in the csv file contains the names of the columns for our data.
If the file contained just data, you would set `header = F` (or header = FALSE).

Creating a data frame with no headings

Inspect the labor.csv dataset using excel to check that it has no headers. Once you've completed your inspection close excel and go back to RStudio. Try

```
labor <- read.csv("labor.csv", header = FALSE, stringsAsFactors=T)
```

This will load the dataset. You will see that the column names are V1, V2, V3, etc
You can change these using the names function. For example

```
names(labor) <- c('duration' , 'wage-increase-first-year',
      'wage-increase-second-year', 'wage-increase-third-year' ,
      'cost-of-living-adjustment', 'working-hours' , 'pension',
      'standby-pay', 'shift-differential' , 'education-allowance' ,
      'statutory-holiday', 'vacation', 'longterm-disability-assistance' ,
       'contribution-to-dental-plan' ,'bereavement-assistance' ,
      'contribution-to-health-plan', 'class')
```

Inspect the labor dataset using

```
View(labor)
```

You will see there are a number of `NA` values
This means there is no value for that specific attribute (column) and instance (row).

Using R's pre-installed datasets

R has pre-installed datasets. Also, some packages have datasets. To check the list of datasets use

```
data()
```

One existing dataset is BOD. To load the BOD (biochemical oxygen demand) dataset use

```
data(BOD)
```

To see what this dataset contains use

```
view(BOD)
```

To check its type, i.e. that it is a data.frame

```
class(BOD)
```

Another dataset is the co2 dataset. To inspect the first 3 rows of the co2 dataset

```
head(CO2, n = 3)
```

To inspect the last 3 rows of the co2 dataset

```
tail(CO2, n = 3)
```

To check the number of columns of the co2 dataset

```
ncol(CO2)
```

To check the number of rows of the co2 dataset

```
nrow(CO2)
```

To check both rows and columns use

```
dim(CO2)
```

Extracting information from a data frame

To get the column names

```
names(CO2)
```

To get column and row names

```
attributes(CO2)
```

To get a good idea of the characteristics of a dataset use

```
summary(CO2)
```

To get all the values in a column use the name of the data frame followed by `$` and the name of the column.

For example, to get all the values for the `Treatment` column in the `CO2` dataset

```
CO2$Treatment
```

To get a dataset like CO2 but with only the 4th ("conc") column

```
 CO2fourth <- CO2[,4]
```

To get a dataset like CO2 but with only the 3rd and 4th columns

```
CO2thirdFourth <- CO2[,c(3,4)]
```

To get a dataset like CO2 but with only rows 10 to 20

```
CO2Rows10to20 <- CO2[c(10:20),]
```

Combining both

```
CO2extract <- CO2[c(10:20),c(3,4)]
```

Excluding specific rows and columns - building a dataset from specific columns and rows

To get a dataset like CO2 but with no Treatment column (the 3rd one)

```
CO2NoTreat <- CO2[,-3]
```

To get a dataset like CO2 but excluding rows 4 to 9

```
CO2NoRows4to9 <- CO2[-c(4:9),]
```

Combining both

```
CO2NoTreatNoRows4to9 <- CO2[-c(4:9),3]
```

Instance (observation) selection

Assume that we want instances where the Treatment is "chilled".

```
CO2chilled <- subset(CO2, Treatment =="chilled")
```

Column Treatment is useless (all values are the same) so can be deleted.

```
CO2chilled$Treatment <- NULL
```

We want to use CO2 data, but only where the treatment is chilled and the uptake is at least 30.

```
CO2selectedVals <- subset(CO2, Treatment =="chilled" & uptake >=30)
```

Column deletion

Column Treatment is useless (all values are the same) so can be deleted.

```
CO2selectedVals$Treatment <- NULL
```

Exercises with data frames

Write R code to do the following:

1. Create a data frame *courses* containing the following information:

course	module1	credits1	module2	credits2
Data Science	analytics	15	project	45
Cyber Security	ethical hacking	15	forensics	15
IT	databases	20	Java programming	20
Digital media	3D animation	30	perception	12

2. Extend *courses* with a row with the following information

course	module1	credits1	module2	credits2
Networks	CISCO CCNA	25	Operating systems	40

3. Extend *courses* with a column *total* which contains the total number of credits for a course.
4. Extend *courses* with a column pgcert which is true if the *total* is at least 60 credits
5. Create a data frame with details of all the courses where at least one module is 20 credits or more
6. Create a data frame with details of all the courses which do NOT have databases.
7. Download and load file contactLenses.csv into variable lenses
8. Show the first 4 rows of the lenses data frame.
9. Show the last 3 rows of the lenses data frame.
10. Check the types of data lenses contains. Does it have any missing values?
11. Assign the 3rd instance of lenses age column value *pre-presbyopic*.
12. Remove column *tearproductionRate*
13. Rename column *contactLenses* to *recommendation.*
14. Produce a version of lenses where astigmatism is numeric (1 if it is a yes and 0 if it has value no)

If you are new to R, this may be enough for today. The rest of the lab can be done later on, but no later than week 3.

R Markdown (optional in 1st week but complete by 3rd week)

There are excellent RMarkdown resources (https://rmarkdown.rstudio.com/) including a tutorial (https://rmarkdown.rstudio.com/lesson-1.html).

The cheatsheet is available from RStudio by going to

Help -> Cheat Sheets -> R Markdown Cheat Sheet. You can also get it from this link (https://raw.githubusercontent.com/rstudio/cheatsheets/master/rstudio-ide.pdf).

Getting started

Create a new Rmd (rmarkdown) file by selecting the following

File→ New file → R Markdown

You will be asked for a title, author and default output. Put "My first R Markdown" as the title, your name as the author and HTML as the output.

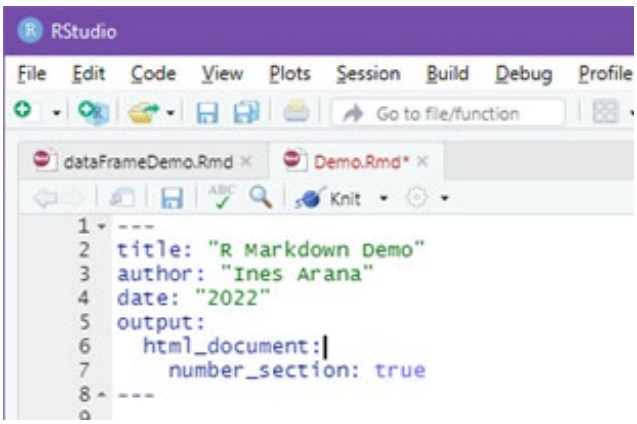
Save your file using

File → Save as

Ensure that you save your file to a suitable folder in your H: drive and you give your file a suitable name. The file will have extension ".Rmd" This is important for knitting the file later on (this creates a document with code, output and explanations).

R Markdown document elements

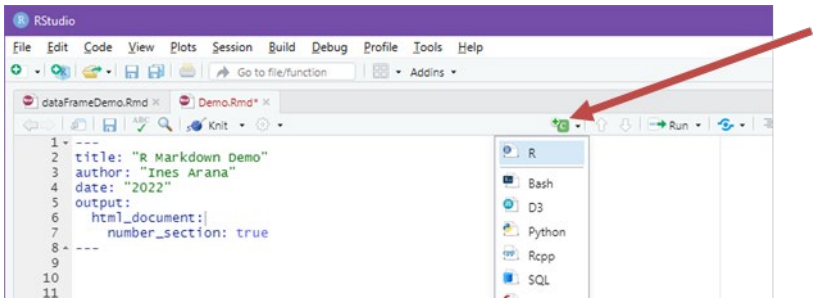
- A YAML header which includes title, author, etc. This header appears at the top of the document between ---. For example



- Code chunks. These are delimited between

```
```{r}
```
```

These code chunks are where the R code goes. You can type them in or use the the R block option.



To run individual code, you can use the run icon, which offers you several options or you can click on the green arrow on the top right of the R block.

- Text area. This is any area except the YAML header and code chunks.

Code chunk options

A code chunk can have a number of different options, depending on whether:

- the code and/or the results are wanted in the final document.
- the code is to be run or not.
- messages are to appear or not.
- warnings are included in the final document or not.
- errors in the code stop the production of the document or whether these are displayed and the processing of the rest of the file continued.

The default display includes code and output.

The options are included in the line which indicates the start of an R block, after the r.

The include option

The include option indicates whether the code chunk and results are to be included in the final document. The code is run.

For example, the code below can be used to clear everything and set the working directory to the folder where this file (Demo.Rmd) is located. if we do not wish this code and results to appear, we add `include = F`, i.e. our start of code chunk becomes

```
```{r include = F}
```

For example

```
```{r include = F}
x <- 10
y <- 20
z <- 10 * 20
z
```

```
```

Note that you will not see the above chunk in the final document.

By default include is set to TRUE, so the code below (which does not have the include option set) will be included as well as any output.

```{r}
p <- 1000
y <- p / 55
```

The echo option

The example below prints code and results.

```{r}
x = 4
y = x * 6
y
```

Use `echo = FALSE` to exclude the R code from the document, displaying only the results of running the code.

```{r echo=FALSE}
x = 4
y = x * 6
y
```

The message option

This option can be used suppress messages from appearing when set to FALSE. These may be, for example, warning messages or error messages. There are other options to supress specific types of messages (see below).

For example, below we try to multiply 3 vectors of different lengths. Note that the length of the 1st vector (the longest) is not a multiple of the length of the 2nd vector. Thus, multiplying these 2 vector s will give a warning.

```{r}
a <- c(10,20,30)
b <- c(7,8)
a * b
```

But if we set message to false, the warning will not appear.

```{r  message = FALSE}
a <- c(10,20,30)
b <- c(7,8)
a * b
```

The eval option

Use eval = FALSE to display the code, but not run it.

````
```{r  eval=FALSE}
z = x * y
z
```
````

The warning option

There are times when the code runs, but issues warnings which you do not wish to include in your final document. The warning option can be used to exclude warnings.

For example, the following will issue a warning.

````
```{r}
p <- c(1:10)
q <- c(25:27)
r <- p*q
r
```
````

Use `warning = FALSE` to suppress warnings.

Compare the output of this code with the one for the previous block

````
```{r warning=FALSE}
p <- c(1:10)
q <- c(25:27)
r <- p*q
r
```
````

The error option

To "ignore" errors and allow the code to run with errors use error = TRUE setting, this tries to run the code, fails when an error occurs, displays the error in the markdown output file but then carries on to process the remainder of the markdown file:

````
```{r  error = TRUE}
a <- "this is text"
b <-  10
c <-  a / b  # this produces an error as varible a contains text.
```
````

Any later code that relies on something that has failed to run will also have errors.

Text options

Sections

One or more has characters at the beginning of a line indicate that the text following them is a section heading. The more hashes, the lower the level of the section. For example, one hash indicates section, two hashes indicates subsection, three hashes indicates subsubsection, etc.

Sections must be preceded by a blank line (or an R block).

```
# This is a section
```

```
## This is a sub-section
```

So the actual heading is smaller than for a section.

```
### This is a sub-sub-section
```

So the heading is even smaller.

You can use even more hashes for sub-sub-...sections.

```
###### This is not effective, but see how it is numbered.
```

If the number_section in the YAML area is set to true, your sections (and subsections, etc) will be numbered. If the number_section is not set or it is set to false, sections will not be numbered. With the number_section set to true, you can ensure specific sections are not numbered by adding `{-}` after their section name (title).

For example

```
### This is an un-numbered subsubsection when others are numbered {-}
```

The hashes used to denote a section must be followed by a blank space.

```
###This is not a subsubsection
```

The above is not a subsubsection because there is no space between the three hashes and the section title.

Basic text formatting

- Text between single stars `*is in italics*`
- Text between double stars `**is in bold**`
- Text between triple-stars `***is in bold italics***`
- Superscripts are indicated between `^` symbols. For example
  - x square is `x^2^` and y to the power of 12 is `y^12^`.
  - Subscripts are indicated between ~ symbols. For example, results subscript test1 is created using `results~test1~`.

- Strikethrough text is indicated between double ~ symbols. For example `~~crossed out rubish text~~`.

Bullets, numbers and indentations

Bullets

A single start preceding text produces a bulleted paragraph.

Numbers

A number ending with a full stop at the beginning of a line indicates a numbered paragraph. Number 1 can be used repeatedly and the numbering will be "put right".

```
1. So this section which was started with "1." will be numbered 1.
8. This starts with an 8, but will be numbered 2.
```

Indentations

Indentations are indicated by exactly 4 blank spaces preceding the paragraph. This can also be used to indent numbering.

```
1. this is a numbered paragragh
    1. this is a numbered paragraph inside a numbered paragraph.
1. this is a second numbered paragraph
```

Horizontal lines

Three starts on a new line gives a horizontal line.

```
***
```

Tables

You can display a basic table (from a data frame), e.g.

````
```{r}
data(iris)
iris[1:4,]  # first 4 rows, all columns

iris[1:3, 2:5] # first 3 rows, columns 2 to 5
```
````

```
##   Sepal.Width Petal.Length Petal.Width Species
## 1         3.5          1.4         0.2  setosa
## 2         3.0          1.4         0.2  setosa
## 3         3.2          1.3         0.2  setosa
```

The knitr library has a kable function that can generate nicer tables (see output below)

````
```{r}
library(knitr)
````
````

```
kable(iris [1: 6,], caption = "Table 1: The first 6 instances of the iris dataset")

```
```

Table 1: The first 6 instances of the iris dataset

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

Links

Links are indicated with the text inside[] immediately followed by the link inside ().

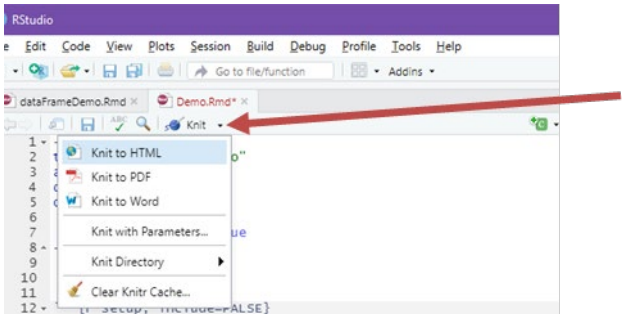
For example,

```
1. There are excellent [RMarkdown resources](https://rmarkdown.rstudio.com/) including a [tutorial](https://rmarkdown.rstudio.com/lesson-1.html).


1. The cheatsheet is available from RStudio by going to
Help -> CheatSheets -> R Markdown Cheat Sheet.  You can also get it from [this link](https://raw.githubusercontent.com/rstudio/cheatsheets/master/rstudio-ide.pdf).
```

Knitting your file

To generate a file containing the code, explanations and outputs use knit. You may knit to HTML, pdf or word. For the practical examination you will need to knit to HTML so use this option, which can be found by clicking on the knit option.



You will obtain a nicely formatted file with your work.
```