Tutorial and Laboratories

11520G Data Capture and Preparations

Week 4

Introduction

In this tutorial will continue practising basic operations in R. In particular, you will learn about how to read external txt and CSV files, and how read specific lines within the text. In addition, you will have another go with conditionals. You will also learn about packages in R.

Skills Covered in this tutorial include:

- Practise conditionals in R
- Packages in R
- Read lines of txt files
- Read CSV files

1. Conditionals in R

Last week we introduce conditionals in R. These are expressions that perform different computations or actions depending on whether a predefined Boolean condition is TRUE or FALSE.

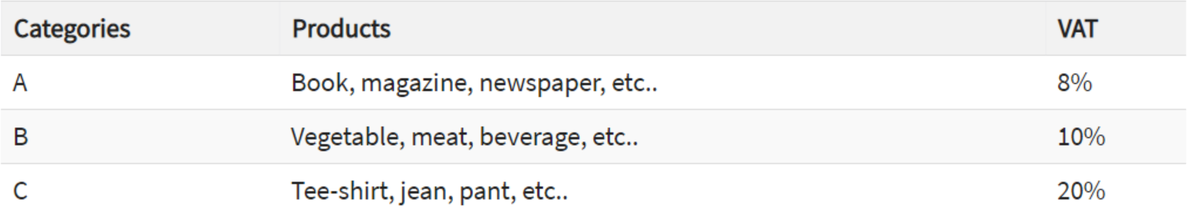
Exercise 1. We are interested to know if we sold quantities between 20 and 30. If we do, then we print “Average day”. If quantity is > 30 we print “What a great day!”, otherwise “Not enough for today”.

Hints:

- create a variable called *quantity* to evaluate each condition, e.g., quantity <- 10
- use the following syntax: if (condition1) {expr1} else if (condition2) {expr2} else {expr3}
- print “Not enough for today” if *quantity* is less than 20
- print “Average day” if *quantity* is more than 20 but less than or equal than 30
- finally, print “What a great day!” if *quantity* is more than 30

What is the output with quantity = 10, 30, 40?

Exercise 2. A value-added tax (VAT) has different rates according to the product purchased. Imagine we have three different kind of products with different VAT applied:



Categories	Products	VAT
A	Book, magazine, newspaper, etc..	8%
B	Vegetable, meat, beverage, etc..	10%
C	Tee-shirt, jean, pant, etc..	20%

We need to write a script that applies the correct VAT rate to the product a customer bought.

Hints:

- Define a variable called *category* that can accept the categories A, B, C. e.g., category <- ‘A’
- Define another variable called *price* to store the price of the product. e.g., price <- 30
- Use the following flow of code: if (condition1) {expr1} else if (condition2) {expr2} else {expr3}
- Print the VAT rate applied to the product and also the total price.
- Use the cat() function to concatenate and print e.g.,
  cat('A VAT rate of 8% is applied.', ' The total price is: ', price * 1.08)

2. How to install R Packages?

R comes with extensive capabilities right out of the box. But some of its most exciting features are available as optional modules that you can download and install. To install a package, simply type the following command in the console area:

```
install.packages("package name")
```

Also, you can do this by using “install” button in the top left corner of the packages window, for example, type in the console window:

```
install.packages("gclus")
```

You only need to install a package once. But like any software, packages are often updated by their authors. Use the command *update.packages()* to update any packages that you’ve installed. To see details on your packages, you can use the *installed.packages()* command. It lists the packages you have, along with their version numbers, dependencies, and other information.

To use a package in an R session, you need to load the package using the *library()* command. For example, to use the package *gclus*, issue the command *library(gclus).*

2.1. How can I check if a specific package is installed?

Sometimes, you might want to know if you have already installed a specific package. Let’s say we want to check if we have installed the package "boot". Instead of checking the entire list of installed packages, we can do the following.

```
a<-installed.packages()          # store the full table
packages<-a[,1]                  # get only the names
is.element("boot", packages) # check if the package is in the list
```

2.2. How can I add or delete packages?

Any package that does not appear in the installed packages matrix must be installed and loaded before its functions can be used. A package can be added using install.packages("package name"). A package can be removed using remove.packages("package name").

We can obtain the full list of available packages, the list of available R packages is constantly growing. The actual list can be obtained using *available.packages()*. This returns a matrix with a row for each package. For example, try the following:

```
p <- available.packages()
dim(p)
```

How many packages did you find?

3. Input external documents into R

As a data scientist/analyst, you are typically faced with data that comes from a variety of sources and in a variety of formats. Your task is to import the data into your tools, analyse the data, and report the results. R provides a wide range of tools for importing data.

Exercise 3. Read Text from structure files using *read.table()* function

read.table() allows us to read a file containing structured (table-like format) text into a data frame. The file can be comma delimited, tab, or any other delimiter specified by parameter "sep=". If the parameter "header = TRUE", then the first row will be used as the row names.

The "sep=" argument can be used to specify different separators, some of the most common separators are: tab (\t), space (\s), single backslash (\\), comma (,), and blank space ( ).

For this exercise, please copy the data below, paste it in notepad, and save it as tab.txt in the working directory. You can also download this file from canvas and save it in your working directory (recommended option).

	t1	t2	t3	t4	t5	t6	t7	t8
r1	1	0	1	0	0	1	0	2
r2	1	2	2	1	2	1	2	1
r3	0	0	0	2	1	1	0	1
r4	0	0	1	1	2	0	0	0
r5	0	2	1	1	1	0	0	0
r6	2	2	0	1	1	1	0	0
r7	2	2	0	1	1	1	0	1
r8	0	2	1	0	1	1	2	0
r9	1	0	1	2	0	1	0	1
r10	1	0	2	1	2	2	1	0
r11	1	0	0	0	1	2	1	2
r12	1	2	0	0	0	1	2	1
r13	2	0	0	1	0	2	1	0
r14	0	2	0	2	1	2	0	2
r15	0	0	0	2	0	2	2	1
r16	0	0	0	1	2	0	1	0
r17	2	1	0	1	2	0	1	0
r18	1	1	0	0	1	0	1	2
r19	0	1	1	1	1	0	0	1
r20	0	0	2	1	1	0	0	1

Now, lets use *read.table()* to get the data from the txt file.

```
# read data
data_tab <- read.table("tab.txt",sep="\t")
# check if it is a data frame
is.data.frame(data_tab)
```

How many columns and rows did you obtained?
What is the name of each column?
How can you read correctly the names of each column and rows?

Exercise 4. Read Lines of txt File via *readLines()* function

When you have to do text mining / text analysis of larger texts, you will typically be provided with relatively unstructured .txt files. The *readLines()* function is perfect for such text files, since it reads the text **line by line** and creates character objects for each of the lines.

We should create a simple txt file that we can use for the application *of readLines()*. So, let's first store the directory, where we want to store and load our example data. Remember to choose your working directory by means of Session → Set Working Directory → Choose Directory, in the main menu of RStudio.

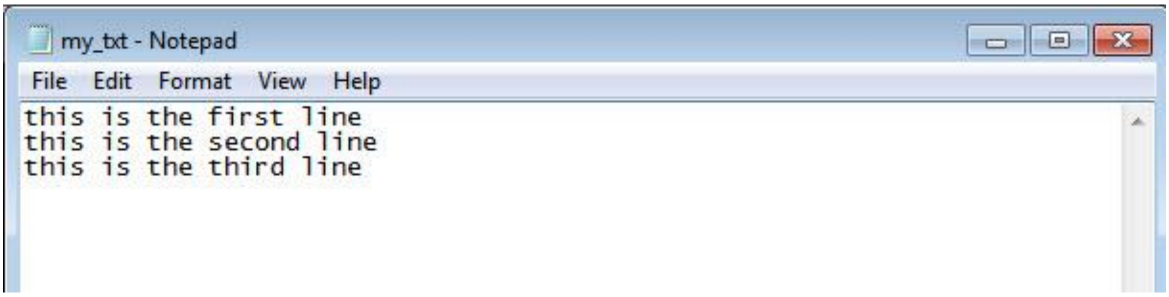
```
# Store currently used directory
path <- getwd()
```

…and then let's create a txt file in this directory:

```
# Write example text to currently used directory
write.table(x = print("This is the first line\nThis is the second line\nThis is the third line"),
        file = paste(path, "/my_txt.txt", sep = ""),
        row.names = FALSE, col.names = FALSE, quote = FALSE)
```

Note: Paste function in R is used to concatenate Vectors by converting them into character.

The txt file looks as follows:



```
this is the first line
this is the second line
this is the third line
```

Figure 1. Text file for the application of readLines().

Now, we can apply the R *readLines()* command to this text file:

```
# Apply readLines function to txt file
my_txt <- readLines("my_txt.txt")
my_txt

[1] "this is the first line" "this is the second line" "this is the third line"
```

The output of the function is a vector that contains 3 character strings, i.e. *this is the first line*, *this is the second line*, and *this is the third line*.

Exercise 5: Read CSV files

Now, we are required to read into R information from a CSV (comma-separated values) file. Let’s first create an example file in our currently used directory.

For this exercise, please copy the data below and paste it in notepad, and save it as employee_sales.csv

```
Employee_ID,FirstName,LastName,Education,Occupation,YearlyIncome,Sales
1,"John","Yang","Bachelors","Professional",90000,3578.27
2,"Rob","Johnson","Bachelors","Management",80000,3399.9899999999998
3,"Ruben","Torres","Partial College","Skilled Manual",50000,699.09820000000002
4,"Christy","Zhu","Bachelors","Professional",80000,3078.27
5,"Rob","Huang","High School","Skilled Manual",60000,2319.9899999999998
6,"John","Ruiz","Bachelors","Professional",70000,539.99000000000001
7,"John","Miller","Masters Degree","Management",80000,2320.4899999999998
8,"Christy","Mehta","Partial High School","Clerical",50000,24.989999999999998
9,"Rob","Verhoff","Partial High School","Clerical",45000,24.989999999999998
10,"Christy","Carlson","Graduate Degree","Management",70000,2234.9899999999998
11,"Gail","Erickson","Education","Professional",90000,4319.9899999999998
12,"Barry","Johnson","Education","Management",80000,4968.5900000000001
13,"Peter","Krebs","Graduate Degree","Clerical",50000,59.530000000000001
14,"Greg","Alderson","Partial High School","Clerical",45000,23.5
```

In this example, we use the read.csv() function to read data from a CSV file that is present in the current working directory.

```
# Read CSV File from Current Working Directory
employees <- read.csv("employee_sales.csv", TRUE, sep = ",", quote="")
print(employees)
```

Now remove the quote="" from the above instruction and observe the result.

Exercise 6. Functions useful in .txt and .csv files.

Although we are working with text files, we can use the following functions to obtain valuable information from the data:

- max: This method returns the maximum value within the column.
- min: This method returns the minimum value within the column.
- mean: It calculates the Mean value.
- median: It calculates the median value of the specified column.
- subset(data, condition): This method returns the subset of data, and the data depends on the condition.

Use the employee data frame to try these functions.

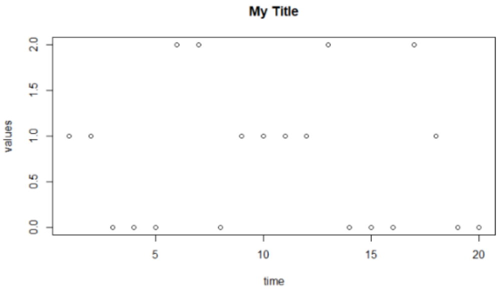
```
# It returns the Maximum Value present in the Yearly Income Column
max.salary <- max(employees$YearlyIncome)
print(max.salary)

# It returns the Minimum Value present in the Sales Column
min.sales <- min(employees$Sales)
print(min.sales)

# It calculates and returns the Median of Sales Column
median.sales <- median(employees$Sales)
print(median.sales)

# It calculates and returns the Mean value of Sales Column
mean.sales <- mean(employees$Sales)
print(mean.sales)

# It returns all the records, whose Education is equal to Bachelors
data1 <- subset(employees, Education == "Bachelors")
print(data1)

# It returns all the records, whose Education is equal to Bachelors and Yearly Income > 70000
data <- subset(employees, Education == "Bachelors" & YearlyIncome > 70000)
print(data)
```

What are the names of the employees with the highest and lowest sales?

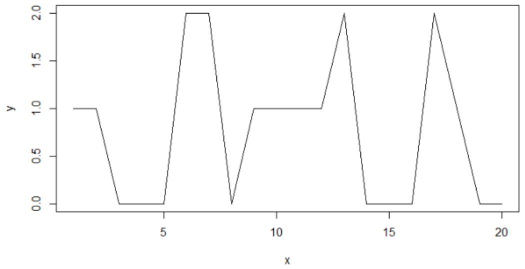
<u>Exercise 7.</u> Plotting data

In this example, we will plot the data use in exercise 3.

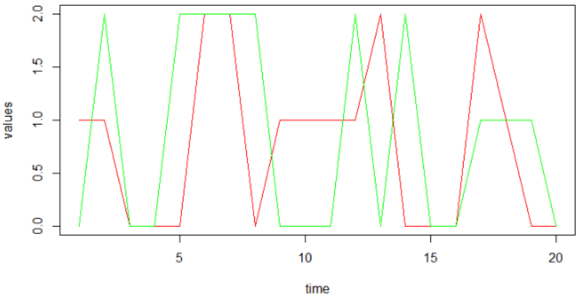
```
# pass the header=TRUE
data_tab <- read.table("tab.txt",header=TRUE,sep="\t",)

x <- 1:length(data_tab$t1)
y <- data_tab$t1

# scatter plot
plot(x,y, xlab = "time", ylab = "values",
        main = "My Title")
```



```
# time series plot
plot(x,y,type = "l")
```



```
# multiple time series
y1 <- y
y2 <- data_tab$t2
plot(x, y1, type = "l", col = "red")
#add new line in the same plot
lines(x, y2, type = "l", col = "green")
```



Solution to Exercise 1:

```
# Create variable quantity
quantity <-  10
# Create multiple condition statement
if (quantity <20) {
      print('Not enough for today')
} else if (quantity > 20  & quantity <= 30) {
     print('Average day')
} else {
      print('What a great day!')
}
```

Solution to Exercise 2:

```
category <- 'A'
price <- 10
if (category =='A'){
  cat('A vat rate of 8% is applied.','The total price is',price *1.08)
} else if (category =='B'){
  cat('A vat rate of 10% is applied.','The total price is',price *1.10)
} else {
  cat('A vat rate of 20% is applied.','The total price is',price *1.20)
}
```

Solution to Exercise 3:

```
ncol(data_tab)
nrow(data_tab)
colnames(data_tab)
data_tab <- read.table("tab.txt", header=TRUE, sep="\t", row.names="X")
colnames(data_tab)
rownames(data_tab)
```

Solution to Exercise 6:

```
#What are the names of the employees with the highest and lowest sales? One possible solution is as follows:

max.sales <- max(employees$Sales)
min.sales <- min(employees$Sales)

name_max <- subset(employees, Sales == max.sales)
name_min <- subset(employees, Sales == min.sales)

paste(name_max$FirstName,name_max$LastName)
paste(name_min$FirstName,name_min$LastName)
```