Business Analytics with Power BI

Module 3 – Predictive Analytics with Power BI and R

Student Lab Manual – Lab 1 – Introduction to R

Version 1.0

Conditions and Terms of Use

Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

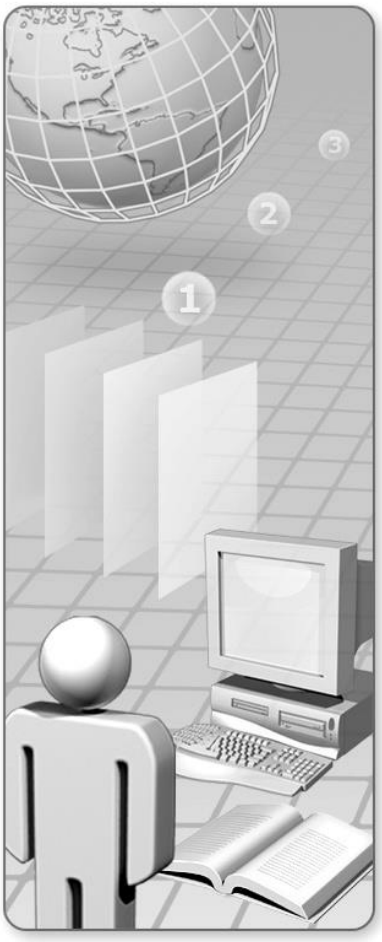
The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Module 3

Predictive Analytics with Power BI and R

Lab 1 – Introduction to R



Lab 1: Introduction to R

Introduction

In this lab, you will learn the basics of the R language. There are two exercises in this lab: R programming, which will make you comfortable with the integrated development environment (IDE) and the language itself. The lab also includes an introduction on displaying results graphically and with greater flexibility.

Objectives

After completing this lab, you will be able to:

- Create basic R scripts.
- Plot data by using R libraries.

Estimated time to complete this lab

60 minutes

Resources

Virtual machine (VM) Name	**Business Analytics with Power BI - Module 1**
Domain	**POWERBI-WIN10**
User	**POWERBI-WIN10\LabUser**
Password	**P@ssw0rd1!**
Lab Files	**E:\Labs**
Asset Files	**E:\Assets**

Exercise 1: R Programming

Introduction

In this exercise, you will use R IDEs to code in R. You will learn how to use R language to create variables, vectors, matrices, and data frames.

Objectives

After completing this exercise, you will be able to:

- Use basic R features.
- Manipulate variables and workspaces.
- Create data structures in R.

Creating an R project by Using R Tools for Visual Studio

In this task, you will create an R project by using Microsoft Visual Studio.

1. To open the Power BI Desktop, on the taskbar, click the **Visual Studio 2015** shortcut.

2. Click the **File** tab, click select **New**, and then click **Project**.

3. In the **New Project** window, on the left side, click **Templates** and then click **R**.

4. Select **R Project**.

5. In the **Name** box, type **MyFirstRProject**, and in the **Location** box, browse to **E:\Labs\M3Lab1**.

6. In the **Solution name** box, type **MyFirstRProject**.

7. Leave **Create directory for solution** check box selected.

8. Click **OK**.

9. Notice that Solution Explorer contains a file called **script.R**. Double-click the file name to open the file. Notice that it is empty. You can leave this file open.

10. Make sure you save your progress as you progress through the lab. Do not close Visual Studio, but minimize it.

Using RStudio

In this task, you will use **RStudio** to run your first R script. We will use Visual Studio for other labs, but you can use **RStudio** if you like.

1. On the taskbar, click the **RStudio** shortcut.

2. On the left, you can see the **Console** window. You can run a command directly there or open a file to save your script.

3. Write **5 + 10** in the console, and then press Enter. You should see the result right below the expression.

```
Default CRAN mirror snapshot taken on 2016-07-0
.microsoft.com/.
> 5+10
[1] 15
>
```

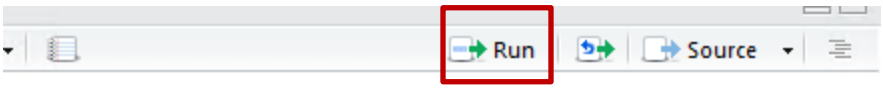
4. Click the **File** menu, click **New File**, and then click **R Script**. Alternatively, you can press Ctrl+Shift+N.

5. Enter the simple expression that you used in the console (5+10), and then press Enter.

Does the code run?

You can run a line of code from a script by pressing Ctrl + Enter.

6. Select the line with the expression **5+10,** and then click **Run**. You can also press Ctrl + Enter. After that, notice that the expression is executed in the **Console**.



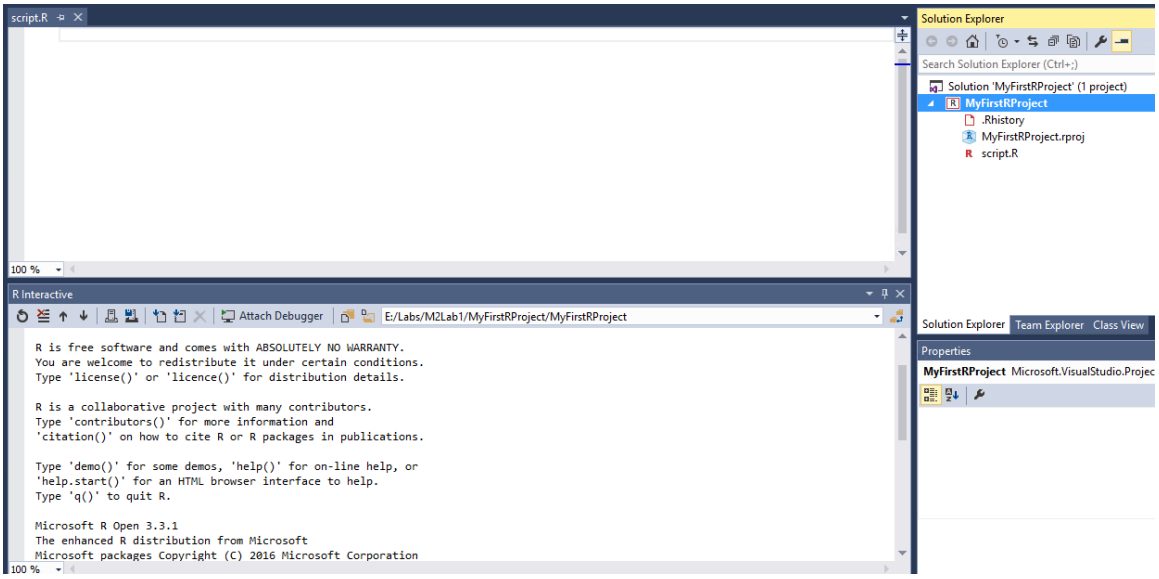
If you want reproducibility, you should always use script files to store the code.

Dealing with Workspaces

In this task, you will create variables in R and manipulate the workspace by using the **ls()** and **rm()** functions.

1. You probably have Visual Studio already open. If not, open it from the taskbar and then open the project you previously created (**File** >> **Open** >> **Project/Solution**) called **MyFirstRScript**.

2. Open the **script.R** file (in the **Solution Explorer** window, double-click the **script.R** file. Your screen should be similar to the following screenshot:

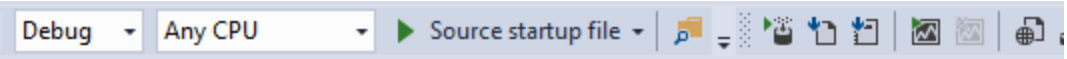


3. The **R Interactive** window is very similar to the Console window from RStudio. Try the same expression (**5+10**), and then press Enter.

4. Type the following code in the **script.R** window.

```
# x <- 1
y <- 2
z = 3
```

5. Place the cursor on the first line, and then click the **Execute on Interactive** icon three times. You could also press Ctrl + Enter.



6. Now, in the **R Interactive** window, you should see the code from the script.

7. In the **R Interactive** window, write **y** and then press Enter.

8. Write **z** and press Enter.

9. Write **x** and press Enter.

10. Notice that you received an error. This is because all the code after a # sign is ignored by R (commentary).

11. Write **`x <- 1`** and press Enter. Write x and press Enter again.

12. Now you have three variables defined. Use the **ls()** function to see all variables in your workspace.

```
> ls()
[1] "x" "y" "z"
```

13. Write the following code in the **script.R** window and execute it (Ctrl + Enter).

```
result < - x + y + z
```

This will store the sum of the three values.

14. It is a good practice to remove from your workspace all the variables that you will not use anymore. Use the following code to remove **z** and **y** variables from your workspace. Type the following in your script file (script.R) and then execute all three lines.

```
rm(y)
rm(z)
ls()
```

15. Notice the result. Variables **y** and z are no longer available.

```
> result <- x + y + z
> rm(y)
> rm(z)
> ls()
[1] "result" "x"
```

16. Add the following script into your script file:

```
varBool1 <- TRUE
varBool2 <- FALSE
varbool1
```

```
varBool1
class(varBool2)
varChar <- "Text"
class(varChar)
```

17. Execute the newly added lines. You get an error while executing **varbool1**. This is because R is case-sensitive. Always pay attention to this to avoid errors.

```
> varBool1 <- TRUE
> varBool2 <- FALSE
> varbool1
Error: object 'varbool1' not found
> varBool1
[1] TRUE
> class(varBool2)
[1] "logical"
> varChar <- "Text"
> class(varChar)
[1] "character"
>
```

18. Leave Visual Studio open. Remember to regularly save your script file.

Getting Help

In this task, you will see how to retrieve information from documentation. The help commands will help you to find information about the functions an structures that you need to use.

1. In your open script, type the following line and then run it:

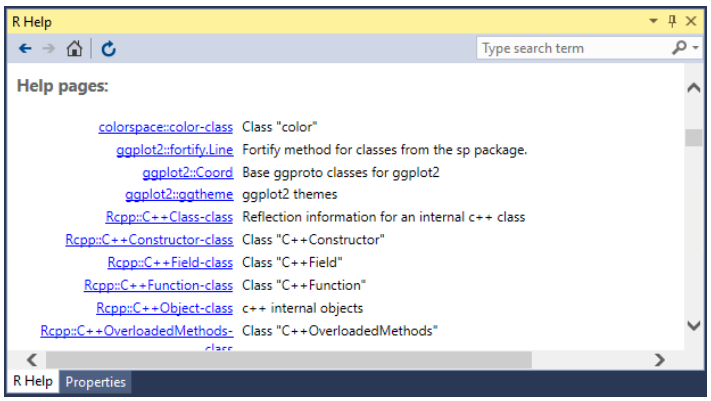
```
help(class)
```

2. In your open script, type the following line and then run it. This is a shorter version of the **help()** function.

```
?class
```

3. In your open script, type the following line and then run it. Notice that it will search for the term **class** in all the content from the documentation.

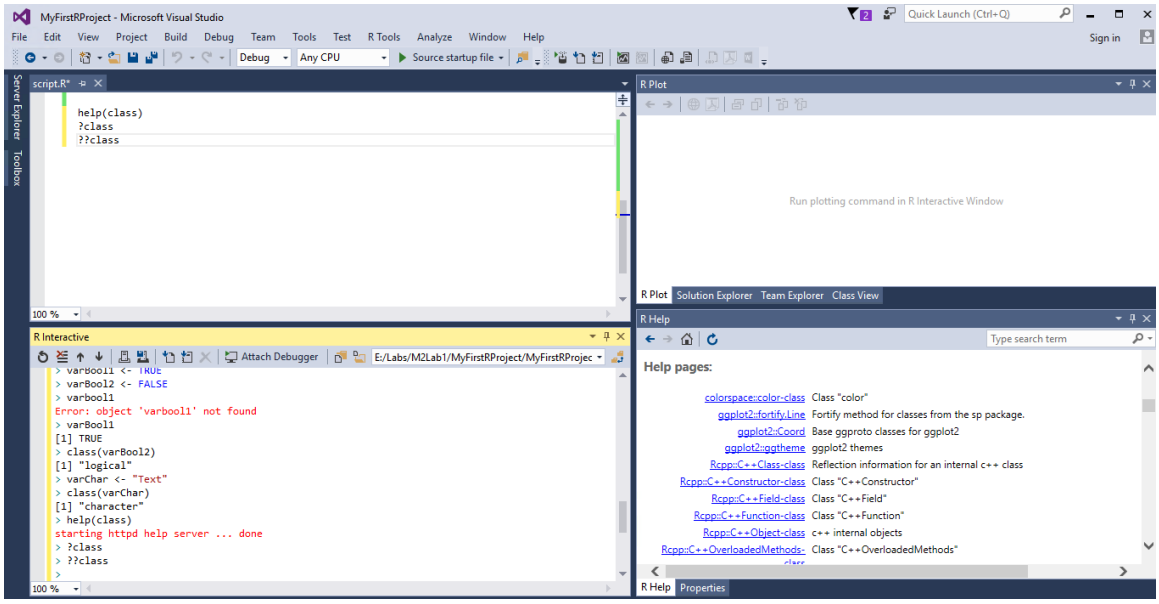
```
??class
```



Working with Vectors, Matrices, and Data Frames

In this task, you will work with basic data structures in R. You will create and work with vectors, matrices, and data frames.

1. You should have the script open in your screen. If not, open it (script.R).



2. Add the following code to your script, and then run it. Notice that the previous value for the **x** variable is overwritten. The function **c()** creates a new vector.

```
x <- c(1,2,3)
y <- c(4,5,6)
x
y
```


3. Add the following code to your script, and then run it. Notice that if you sum a 1x3 vector with another 1 x 3 vector, you will have a 1 x 3 vector.

```
x + y
```

4. You should see **5, 7, 9** as the result.

```
> x <- c(1, 2, 3)
+ y <- c(4, 5, 6)
+ x
+ y
[1] 1 2 3
[1] 4 5 6
> x + y
[1] 5 7 9
```

5. Now, let us run an operation with a vector and a single element. Notice that the operation is the same, and that it is done by the element.

```
x * 3
y + 1
```

```
> x * 3
[1] 3 6 9
> y + 1
[1] 5 6 7
```

6. Now try to create a vector with more than one type. Add the following code to your script, and then run it. Notice the type conversion

```
z <- c(TRUE, 1)
z2 <- c(1, 2, 3, "ABC")
z
z2
```

Note: See that all elements were converted to the most basic data type in the context. Vectors (and matrices) can hold elements from the same data type only.

```
> z <- c(TRUE, 1)
+ z2 <- c(1, 2, 3, "ABC")
+ z
+ z2
[1] 1 1
[1] "1"   "2"   "3"   "ABC"
```

7. Matrices and vectors exhibit similar behavior. Add the following code and execute it row by row to see the results. Notice that the "**:**" operator creates a range from all values between the specified values (1 and 10).

```
m <- matrix(1:10)
m
m * 2
m + 100
m1 <- matrix(1:10, ncol = 2)
m1
matrix(1:12, nrow = 5, byrow = TRUE)
```

```
Warning message:
In matrix(1:12, nrow = 5, byrow = TRUE) :
data length [12] is not a sub-multiple or multiple of the number of rows
[5]
```

Note: See the warning that is thrown when you run the last line. This is due to the size of the matrix. It will continue to fill with the same values again.

8. To access values, you should use "**[**" and "**]**". You can use names or indexes. Add the following script, **run it row by row** and see how to retrieve values from a matrix. (This is very similar for other data structures, such as vectors).

```
m2 <- matrix(c(4, 3, 2, 17, 18, 19), nrow = 2, ncol = 3, byrow =
TRUE,
    dimnames = list(c("row1", "row2"), c("C1", "C2", "C3")))
m2
m2[1, 1]
m2["row1", "C2"]
m2[, 1]
m2[1,]
```

```
> m2 <- matrix(c(4, 3, 2, 17, 18, 19), nrow = 2, ncol = 3, byrow = TRUE,
+     dimnames = list(c("row1", "row2"), c("C1", "C2", "C3")))
+ m2
     C1 C2 C3
row1  4  3  2
row2 17 18 19
> m2[1, 1]
[1] 4
> m2["row1", "C2"]
[1] 3
> m2[, 1]
row1 row2
   4   17
> m2[1,]
C1 C2 C3
 4  3  2
```

9. Add the following code to script.R, and then run it. **Mtcars** is a built-in dataframe in R.

```
head(mtcars) #first lines
str(mtcars) #structure
summary(mtcars) #statistics about dataframe
```

10. Add the following code to script.R, and then run it.

```
df <- read.csv("E:\\Assets\\M3 - Lab 1\\sample_csv.csv", fill =
TRUE, header = TRUE)
df
str(df)
df[, "C2"]
df[1:2,]
```

Note: Data frames can hold elements with different data types (each column can have a different data type).

Exercise 2: Plotting Data

Introduction

In this exercise, you will use R to create different types of visualizations.

Objectives

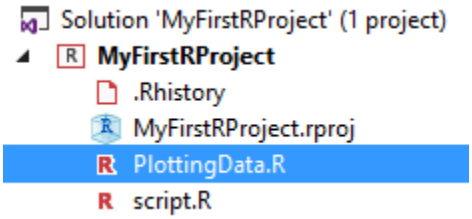
After completing this exercise, you will be able to:

- Install new packages in R.
- Create charts and visualizations by using R

Installing Packages in R

In this task, you will install the **ggplot2** and **corrplot** libraries. These libraries are extensively used by R developers to generate several types of visualizations.

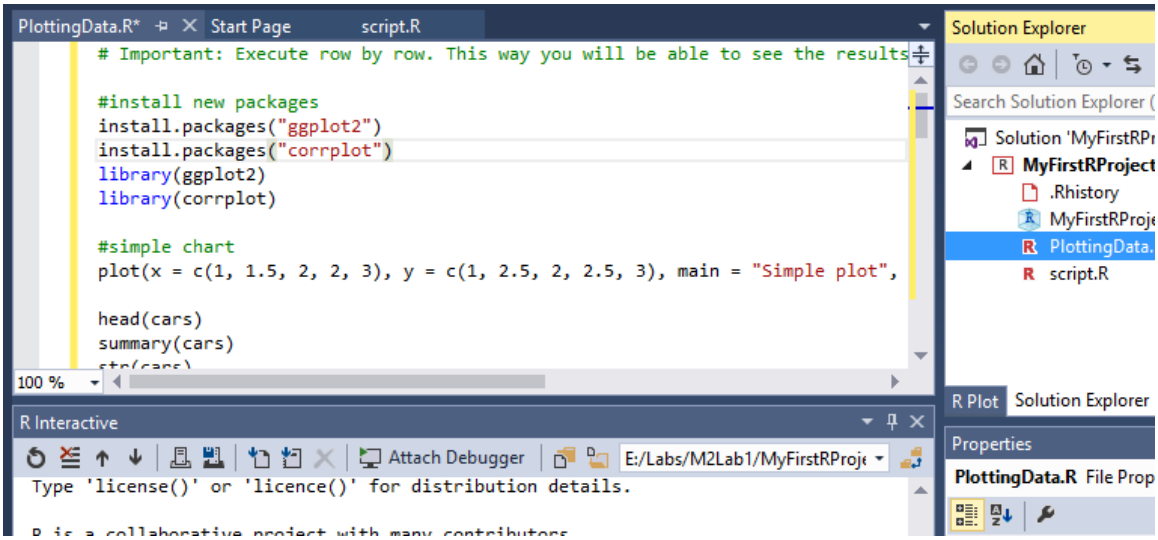
1. Go to the **Solution Explorer** pane, right-click **MyFirstRProject** (project name), and then click **Add R Script**. Rename the script file to **PlottingData.R** (right-click the file name, and then click **Rename**).



2. Open the R script located at **E:\Assets\M3 – Lab 1\PlottingData.txt**.

3. Copy the entire content to the recently created file in your project (PlottingData.R). Save your project, but do not close it.

4. Your screen should be similar to the following screenshot:



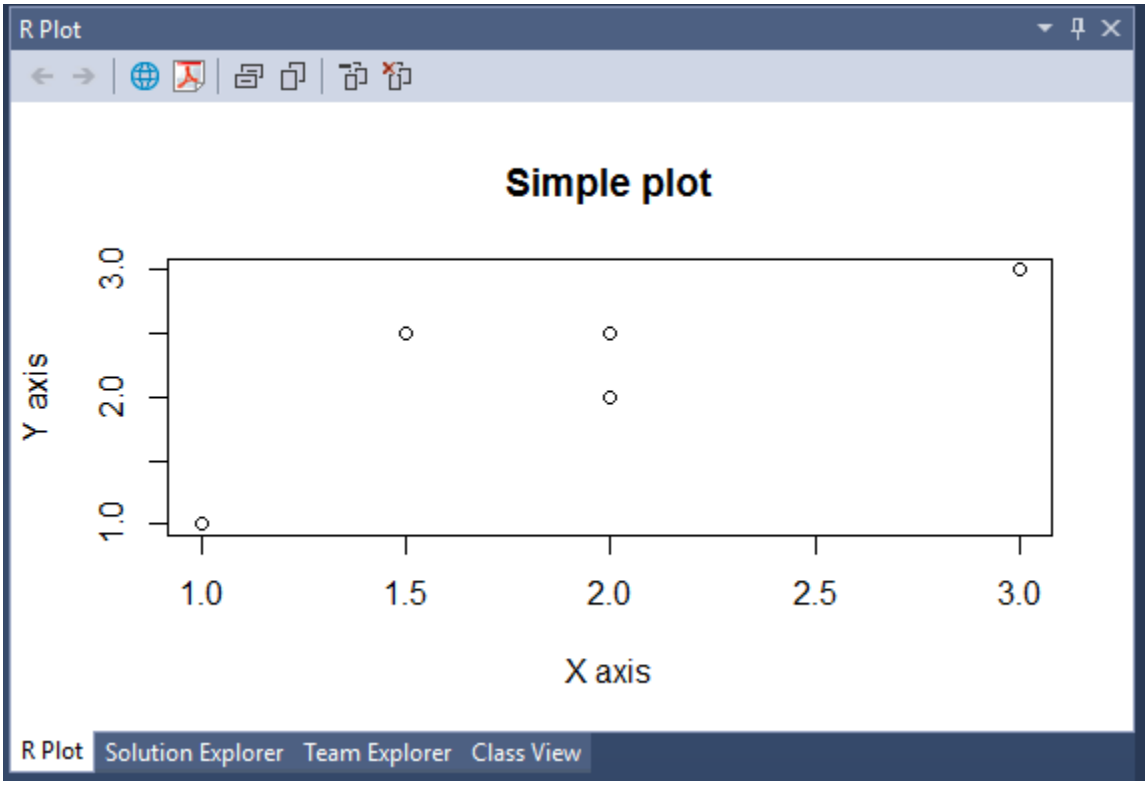
5. Execute the first four rows. This can take a few minutes to finish. Notice that the packages will be installed along with its dependencies. The library() function will tell R that we want to use the functions inside these libraries.

Plotting Data in R

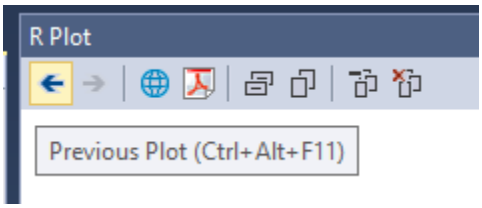
In this task, you will run commands in R that will create several visualizations by using basic R, **ggplot2**, and **corrplot** functions.

1. Now that the libraries are installed, execute the other rows of script (row–by-row basis). Notice all the plots that can be created. This is just a small subset of all that you can do with R. Increase the size of the R Plot pane, if it is too small.

2. After you complete running all the code, you can leave your project open to use in the next lab.



You can use the arrows to move between the plots.



Note: You can always use help functions to know more about a specific command or function.