Working with R Scripts

R is one of the most powerful computer languages used for data manipulation and visualization, so we need to understand how we can use it in the Power BI Desktop to import data from where we want it and use it to create some cool visualizations.

Power BI Desktop is another powerful tool that helps you create reports. These reports can easily be saved to the Power BI Report Server or published to the Power BI Service. The fact that Power BI Desktop is integrated with the R language is one of the most valuable features; we can import data and transform it using R; we can use it to create data visualizations, and there are also several pre-built visualizations based on the R language that you can import, without having to write any R or interact with it.

In this chapter, we're going to look at how we can work with the R language in the Power BI Desktop. Our focus will be on performing tasks specific to R rather than delving into the actual language. It is for this reason that I will only be using simple script examples; these are purely to demonstrate how R can be used in the Power BI Desktop. When you grasp these basic concepts, you can move on and use the extensive capabilities offered by R for the analysis, transformation, and visualization of your data. If you have no knowledge or R, now would be a good time to go and learn the basics.

To work with R in Power BI, you must ensure the language is installed with Power BI on the same computer. R can be downloaded and installed for free from several places, depending on the distribution you want. For example, you can install the CRAN distribution from the R Project website, https://cran.r-project.org/bin/windows/base/, or, if you want the Microsoft R Open distribution, head to

https://mran.revolutionanalytics.com/download. I will be using CRAN v 3.5.0.

It is also best if you install a separate R IDE (Integrated Development Environment). Doing this means your R scripts can be verified before you run them in Power BI Desktop. Plus, if you are familiar with other computing

languages, you will know that using an IDE means you can easily install the packages you need to use R within the Desktop environment. I am using RStudio (the free version), which you can get from https://www.rstudio.com/products/rstudio/download/.

When you have installed R and set it up, the Power BI Desktop will look for the installation. To make sure it has picked up the right version, click on File>Options and Settings>Options. It should detect the right installation version and your IDE if you installed one. If the information isn't right, provide the details manually.

You only need to associate an IDE with the Desktop when you want R visualizations in the Reports view. When an IDE is associated, it can be launched from the Desktop, and you can work directly on your visualization script.

Importing Data Using R Scripts

When you import data using an R script, that script has got to return one or more data frames; these will be the basis for the table you are importing. If you get several data frames returned, you can pick the ones you want included in the import, and Power BI will then create one table for each data frame. I do need to warn you that, if one of your data frames has a column that was configured using the vector or complex type, the values in the column will be replaced with errors by Power BI Desktop.

If you want to import data using an R script, go to the Desktop Home ribbon and click on Get Data. A dialog box opens; go to the Other category and click on R Script. Click the Connect button, and Power BI Desktop will open the dialog box for R Script. Here, you can type in your script or paste it in. We're going to get some data from the iris dataset, which is in the CRAN distribution; the following script is used, assigning the data to the variable called iris_raw:

```
iris_raw <- iris
```

In the Power BI Desktop, the dataset has to be assigned to a variable, whether you make any modifications to the data frame or not. If you only typed in the dataset name, you wouldn't find any data frames to import into the Desktop.

Before you put your script into the dialog box, run it through the IDE; this will allow you to check it is running okay, and you get the results you want. If you don’t do this and you get an error from it in Power BI, it won't be easy to understand, and you will need to start again with the import.

When you are happy that the script is ready, type or paste it into the Script dialog box. Click OK, and Desktop will process it, opening the Navigator dialog box. From here, you can choose the data frames you want to import, and you can see some sample data for each one. In our case, we will only get the iris_raw data frame.

Where you have multiple data frames, they are listed in the Navigator box under R[1]>Display Options. Make sure you check the box next to each data frame you want imported and then click on Load. Once the data has been loaded, click on the Data view to see the dataset.

Sometimes, your data may be in a file and not in one of the datasets built-in to the distribution. Let’s say, for example, that you took the data from the iris dataset and copied it to a CSV file. You saved it to C:\DataFiles on your computer; that data can be pulled into an R data frame with the statement below:

```
iris_csv <- read.csv(file="C:/DataFiles/iris.csv", header=TRUE, sep=",")
```

This is using a function called read.csv. We set the argument for the header as TRUE, indicating that headers should be created from the first-row values. Then we used the sep argument to indicate that the data values in the file should be comma-separated. After that, the data is imported from the iris_csv data frame using the same process as above.

R is an incredibly flexible language because it has the capability of installing packages that offer extra functionality. There are a few of these that help you with reports and analysis. For example, data.table and dplyr give you some powerful functions to use with data frames, and ggplot2 is great for visualizing data. These must be installed, and you can do this through the IDE. We used R Studio – open it and run the following commands:

```
install.packages("dplyr")

install.packages("data.table")
```

```
install.packages("ggplot2")
```

Once they have been installed, the library function in the R script can be used to call the package you want when you import the data. By doing this, you can use all the functions in the imported package.

A significant benefit of using R for data importation is that the data can be manipulated during the import process. The script below, for example, uses two functions from dplyr – group_by and summarize – to group the data and then aggregate it is imported:

```
library(dplyr)

iris_mean <- summarize(group_by(iris, Species),

  slength = mean(Sepal.Length), swidth = mean(Sepal.Width),

  plength = mean(Petal.Length), pwidth = mean(Petal.Width))
```

You use the group_by function to ensure data is prepared for a different function, in our case, the summarize function. In the example above, summarize is used together with a function called mean, so that the mean can be found for the four measures. The values listed in the species column are used to group these.

Going back to the R statement we use, the data that was aggregated was saved into the variable called iris_mean, and that is the name given to the dataset when it is imported to the Desktop.

This is just a simple script, and you can write scripts as complex as you want them to be. All I have done here is given you an idea of the simplicity of using R for importing data to the Desktop. The more experience you have with R, the more powerful this is.

Transforming Data Using R Scripts

Sometimes, you might want to manipulate a dataset that has already been imported into the Desktop, and you can do this using Query Editor. With this, an R script can be applied to the dataset for transforming the data. Using our iris dataset, before the data is modified, look at how the dataset looks in Query Editor. In the section called Applied Steps, there are two steps:

1. **Source –** in the Source step, the main pane in Query Editor shows a small table. This table is representing the operation we did to import the dataset, and there would be one row for each data frame returned. In our case, because we only have one data frame, the table only has one row. In the column titled Value, the Table value is representing the data that goes with the data frame and, when you choose that value, step two kicks in.
2. **Navigation –** This is the actual data that we imported.

When an R script is used for importing data, both of these steps are added by the Power BI Desktop.

Now let's look at running an R script against our iris_raw dataset. We'll keep it simple and use the exact same aggregation logic we used to import iris_mean. There is one main difference – rather than iris being specified to reference the dataset, the dataset variable is used instead, as you can see in this script:

```
library(dplyr)

iris_mean <- summarize(group_by(dataset, Species),

  slength = mean(Sepal.Length), swidth = mean(Sepal.Width),

  plength = mean(Petal.Length), pwidth = mean(Petal.Width))
```

To run scripts in Query Editor, go to the Transform ribbon and click on Run R Script. The relevant dialog box opens, and you can input your script to the textbox. Power BI Desktop will add a comment that tells you the input data is held in the dataset variable. That input data is the dataset active in Query Editor; in our case, that is iris_raw.

Click on OK after you add your script and, in the Applied Steps, Query Editor will add in two steps that work much like Source and Navigation did:

1. **Run R Script –** this step is reflecting the data frames that the script returned
2. **"iris_mean" –** this is the data frame that was selected

The ability to run R scripts against datasets gives you another powerful tool

when you want to work with data you imported, be it from a text file, an online service, or a database system. Once the data has been imported into the Power BI Desktop, it is there to do what you want with, regardless of where it came from.

Creating R Visualizations Using R Scripts

Power BI Desktop also allows you to create visualizations using R scripts, and this is done through the Report view. Overall, this is as easy to do as using R scripts in other ways, but there is one important thing to remember – data is automatically grouped and summarized by R whether you want it or not, and there is no way of getting around the behavior. It's something of a mystery as to why Microsoft did this, and it can provide results that you really didn't want.

As with most things, though, we can get around this. How? Simply by adding a column in the dataset to identify each row uniquely. If you are familiar with the SQL Server table (more about that later), this is much like the IDENTITY column found in those.

If you are importing your data using an R script, the column can be added at the same time. Take the following script, for example; this gives our iris dataset an identifier column based on the row names or index of the dataset:

```
library(data.table)

iris_id <- iris

iris_id <- setDT(iris_id, keep.rownames=TRUE)[]

setnames(iris_id, 1, "id")

iris_id$id <- as.integer(iris_id$id)
```

We start our script by calling a package names data.table. This packaged gives us some useful tools to work with objects in data frames. Next, we use a function called setDT with an argument of keep.rownames, to use the index values to create our new column. Note that the iris dataset has to be assigned to the iris_id variable before you can use setFT. The reason for this is that he setDT function directly changes the dataset, and this cannot be done with any

built-in dataset.

Once the column has been created, the function called setnames is used to change the first column's name from rn, which is the default, to id. Lastly, the data type for the column is changed to integer.

Once you have the iris_id dataset, you can use an R script to create your visualization. In Report View, go to Visualizations and then click on R. When you do this for the first time, you are asked to enable script visualizations; just click on Enable and away you go.

On the Visualizations pane, after you click R, a graphic placeholder is added to the report by Power BI Desktop, and the R script editor pane opens. Before you can start writing your script, you must identify which columns from the dataset are being used in your visualization. The easy way to do this is simply to go to Fields (in Visualization) and drag your preferred columns into the Values section. For our example, I want you to do this with the columns title id, species, Petal_Width, and Petal_Length.

When the columns are added, you will see that Desktop adds some comments in the script editor box. Comments one and two are indicating that we created a data frame called a dataset, and this was based on our chosen columns. That data frame name, dataset, has to be used in your R script as a way of referencing the source data.

The next pair of comments, together with a warning message you will see at the top of the pane, are indicating that some duplicate rows have been eliminated from the dataset – this is why that identifier column was required.

Underneath those comments goes your script. The script below will produce a basic scatter plot:

```
library(ggplot2)

ggplot(data=dataset, aes(x=Petal.Width, y=Petal.Length)) +

  geom_point(aes(color=Species), size=3) +

  ggtitle("Petal Widths and Lengths") +

  labs(x="Petal Width", y="Petal Length") +
```

```
  theme_bw() +
  theme(title=element_text(size=15, color="blue3"))
```

The ggplot function from the ggplot2 package is used for creating the visualization, using the labels and the colors specified in the script. Note that, for the X-axis, we used the Petal_Width column and, for the Y-axis, we used the Petal_Length column. The plot colors are served by the column named Species.

Once your R script has been defined, go to the top of the editor pane and click on Run Script. The script is them processed by Power BI Desktop, and the visualization is placed where the placeholder was inserted earlier on.

Visualizations created using R can be updated whenever you want; all you need to do is modify your script and then click on Run Script to update it.

If you wanted to use your IDE to edit your code, go to the top of the editor pane and click on Edit Script in External R IDE. The IDE will open, showing the R script with the right code for connecting to your chosen visualization data source. The script also has the code added to your visualization script in Desktop. You will still need to copy most of the code and paste it in the Power BI Desktop if any modifications are made, but this does get you out of needing to set the data source to test the script in the IDE.

When you add visualizations based on R to a report, you need to keep in mind that there are licensing restrictions with Power BI and, unless you hold a Power BI Pro license, these R visualizations cannot be used.

Importing Custom Visuals Using R

Power BI Desktop also lets you import predefined visualizations that are based on R to the workspace. You can find these in the Microsoft AppSource gallery, and this is directly accessible via the Power BI Desktop. There is no need to have an understanding of R, and you don’t need to build scripts or run them.

To get one of these visualizations, go to Report view and click on Visualizations. From there, click the ellipsis (…) and click on Import from Marketplace. The Power BI Visuals window opens, and you can browse

what’s there.

When you have found the one you want, click on Add. If the R Packages Required dialog box loads, you will know that you need other packages. The box will list the packages you need, and you have a choice – click on Install and Power BI Desktop will do it all for you, or click on Cancel and manually install them.

To continue our example, choose the Spline Chart. If you can’t find it, go to the search box and type in Spline. When you have added it, the required packages box will open; just click on Install and let Power BI do the work for you.

When custom visualizations are used, you will see a new button in the Power BI Visualizations pane; it will relate to the specific visualization. That visualization can then be included in your report and configured the same as you would the prebuilt ones. For ours, you can specify three columns from our iris_id dataset – Species, Sepal_Width, and Sepal_Length.

The Microsoft AppSource gallery has several interesting, free visualizations and is certainly worth a look as it can help you to create even more engaging reports than the prebuilt visualizations allow. And Microsoft has gone the extra mile and made those visualizations dead simple to get into Power BI Desktop.

Getting the Best Out of R in Power BI

The fact that R has been integrated makes Power BI even more powerful when it comes to the transformation and presentation of business intelligence data. R is one of the most comprehensive of the statistical graphics and computing languages, implemented extensively, and with one of the largest user communities. Used in Power BI, it can help you import data, modify it, and come up with many visualizations that produce deep insights into your data.