

Contents

[Lab Prerequisites 3](#_Toc527456595)

[Document Structure 4](#_Toc527456596)

[Overview 5](#_Toc527456597)

[Introduction 5](#_Toc527456598)

[Data Set 5](#_Toc527456599)

[What Is R? 6](#_Toc527456600)

[What Is R? – Installing R 6](#_Toc527456601)

[What Is R? – Installing RStudio 7](#_Toc527456602)

[What Is R? – Basic Syntax 8](#_Toc527456603)

[What Is R? – Data Frames 10](#_Toc527456604)

[What Is R? – Creating Visauls in R (Part 1) 12](#_Toc527456605)

[What Is R? – Creating Visuals in R (Part 2) 15](#_Toc527456606)

[Utilizing R in PowerBI 19](#_Toc527456607)

[Utilizing R in PowerBI – Using What we Learned in R Studio. 19](#_Toc527456608)

[Conclusion: 24](#_Toc527456609)

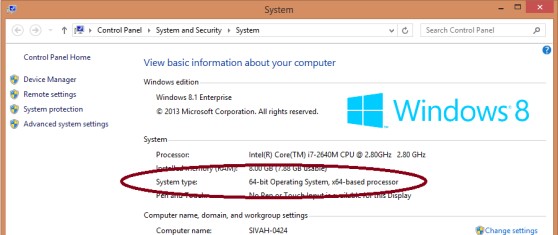
[Conclusion: Questions 24](#_Toc527456610)

# Lab Prerequisites

Following prerequisites and setup must be complete for successful completion of the exercise:

* You must have an internet connection.
* **Signup for Power BI:** Go to <http://aka.ms/pbidiadtraining>and sign up for Power BI with a business email address. If you cannot sign up for Power BI, let the instructor know.
* If you have an existing account, please go to [http://app.powerbi.com](http://app.powerbi.com/) and **Sign in** using your **Power BI Account.**
* At minimum, a computer with 2-cores and 4GB RAM running one of the following version of Windows: Windows 10, Windows 7, Windows 8 (64-bit preferred), Windows 8.1, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2.
* Microsoft Power BI Desktop requires Internet Explorer 9 or greater.
* Verify if you have 32-bit or 64-bit operating system to decide if you need to install the 32-bit or 64- bit applications.
  + Search for computer on your PC, right click properties for your computer.
* You will be able to identify if your operating system is 64 or 32 bit based on “system type”

as shown below.



* Download all our session content here: <https://github.com/FreshBI/R-in-PowerBI-Training-Pack>
* **Download and install Power BI Desktop** using any one of the options listed below:
  + If you have Windows 10, use Microsoft App Store to download and install Power BI Desktop app.
  + Download and install Microsoft Power BI Desktop from [http://www.microsoft.com/en- us/download/details.aspx?id=45331.](http://www.microsoft.com/en-us/download/details.aspx?id=45331)

## Document Structure

This document has two main sections:

* **What is R?**: This section is an introduction to R: It goes through some examples of basic syntax, and building visuals in R. This in no way is a comprehensive course in R, but it will provide you with the mindset needed to go out and explore on your own.
* **Utilizing R Within Power BI** : This section covers how to use R in PowerBI: This includes key tips and tricks, and general philosophy of how to develop R Scripts in PowerBI. In the process here we go through the process of implementing our visual we created in RStudio within PowerBI, and then utilize R to add a bit more value to our visual.

Throughout this document you will find all steps in a table on the left side. On the right will be screenshots or images to aid in following along.

# Overview

## Introduction

Today we will be learning and exploring the power that R adds to PowerBI. The challenge with this topic is that, while R can be very simple and intuitive, there is so much that it can do that it can be very overwhelming. Additionally, many of the basics that R provides can be done solely with PowerBI. This causes a tricky dichotomy in which the easy stuff is too simple, but the useful stuff is too complicated. The goal of this work book is to introduce to you the basics of R and examine some relatively simple tools that R provides that are not easily accomplished in PowerBI.

## Data Set

For the first portion of this workshop we will be working with sets and arrays of our creation.

The latter portion here will use a dataset from the public data catalog of the government of Canada. Our source was download from [here:](https://catalogue.data.gov.bc.ca/dataset/bc-surgical-wait-times)

After downloading, I trimmed out some of larger hospitals to have a more consistent distribution of data.

The original data was a little too varied to provide useful insights it was. I additionally filtered down to a few surgical procedures, again to eliminate some clutter.

Workshop Outline

1. What is R?
2. Utilizing R in PowerBI
3. Questions

# What Is R?

R is an open source language and environment for statistical computing and graphics. It provides a wide variety of statistical and graphical techniques and is highly customizable. This section examines some of the basics of how to use R and what it is capable of.

## What Is R? – Installing R

Installing R is as easy as 1. 2. 3. We need to have it installed to work with it today.

### What is R? – Installing R

|  |  |
| --- | --- |
| We must begin with installing R. This is very simple. We’ll navigate to R’s page and download R from there. If you already have R installed on your computer you can skip this step.   1. Navigate to the following link <https://cran.r-project.org/bin/windows/base/> 2. Click “Download R 3.5.1 for Window” 3. Open the downloaded file and proceed as normal to install.   It really is that easy. |  |

## What Is R? – Installing RStudio

The default R GUI is not very friendly to work in. We are going to install RStudio which is way better to work and experiment with R in. It will be much easier to see what we are doing. This step is again going to be extremely simple.

### What is R? – Installing RStudio

|  |  |
| --- | --- |
| This will again be as easy as last step. If you already have R Studio you can skip this.   1. Navigate to the following in your web browser <https://www.rstudio.com/products/rstudio/download/> 2. Click the download button for “RStudio Desktop”, this is the free version of RStudio 3. Once your download finishes, go through the normal process of installing.   I promise things get more exciting after this… |  |

## What Is R? – Basic Syntax

One of the reasons R is so widely used is because it is very simple to get started. While you can get very detailed, you can also just explore the easily accessible packages that R has available. This section will start to go through some basics and allow us to build a foundation for what we can do once we are in PowerBI.

### What is R? – Basic Syntax

|  |  |
| --- | --- |
| Now we are going to review some basics in R syntax. We are going to talk about variables, operations and arrays.   1. Open RStudio. You will be presented with a simple workspace for us to begin developing in. 2. Now we are going to assign some variables. In the console window, type “a = 6” and hit return. Observe the new item under values in our environment. 3. Type “b <- 12” and hit return. 4. Type “42 -> c” and hit return. 5. Again, have a look at our environment and see these are just three ways of assigning new variables. 6. Type “b <- FALSE” hit return. 7. Our variable b which was an integer is now overwritten as a Boolean value. Note the Boolean values TRUE and FALSE must be represented in all capitals as shown. 8. We can also create character variables. Type “txt = “Hello””. 9. Let’s make one more value, type “true = TRUE”. Now we are going to have a look at operations within R. |  |
| So far, our console should look something like the screen shot to the right. Now we are going to use these variables in some very simple cases.   1. In the console type “a + c”. Notice the result it returns. Also notice that no variables have changed. This is a ‘one time’ result from R. 2. Now type “result = a + c - 10”. This time, R did not give you the result. But rather the result was stored as a variable. To see the result in the console type “result” and hit return. 3. Type a^2 and hit return. R executed 6 squared and returned the result. 4. We can also use Boolean expressions using the symbols & for and, and | for or. Type “b & true”, then “b | true”. Observe. 5. Now we are going to create some arrays. In the console type “arr1 = c(1,2,3,4,5)”. We have created an array with the values 1 through 5, entered manually. 6. Type “arr2 = 6:10”.This is a quick way of creating an array with numbers 6 through 10. 7. Type arr1 and hit enter, then arr2 and hit enter to see the contents of both arrays. 8. Type “arr1\*arr2” ad observe the dot product of the two arrays. 9. Type arr2[3], this yields the 3rd element of arr2. You can experiment with elements of arrays. Tyr arr2[2:4].   Arrays are the first step in a very important concept for us. |  |

## What Is R? – Data Frames

Now that we have spent some time exploring syntax and operations in R we can talk about data frames. They are very

simple to understand, but they are very important for working with R in PowerBI. You will see why later today.

### What is R? – Data Frames

|  |  |
| --- | --- |
| Let’s create some arrays to create our data frame with.   1. Type “id = 100:104” and hit return. This gives us an array with 5 entries. 2. Type “name = c(‘Chris’,’Craig’,’Michael’,’Micah’,’Dmitry’)”, now we have a array with 5 names. 3. Now type “frame = data.frame(id,name)”. Have a look in our environment now. Rather than a data frame being a value it is now in a different section labeled “Data”. It states that frame has 5 objects of 2 variables. What exactly have we created? 4. Type “frame” and hit enter to see what we have created. Once we’ve done this it becomes very clear that a data frame is just a fancy word for a table! The 5 objects refer to the number of rows, and the two variables refer to the number of columns.   Now that we have seen what a data frame is, we have some logical restrictions. If we wish to create a data frame, all of the arrays we want to combine must have the same number of elements.   1. We can experiment by creating a new id array and adding a sixth element. We can do this by typing “id2 = id”, then “id2[6] = 105”. 2. Next, we can attempt to create a new data frame with id2 and name and see what happens. Typing “frame2 = data.frame(id2,name)” yields the following.   What can we do with a Data Frame? Well the answer is a lot. But first we will just go over some basic operations.   1. Type names(frame) and notice this lists the names of the columns in our data frame. 2. Type “ncol(frame)”, this returns the number of columns our data frame has. 3. Type “nrow(frame)”, this (can you guess?) returns the number of rows in our data frame. 4. “frame[2:4,]” returns the 2nd through the 4th rows. 5. “frame[2:4,1]” returns the values from the first column for rows 2 through 4. 6. Lastly “frame$id” and “frame$name” return their respective column.   Now we’ve seen what data frames are and some basic operations we can do with them. Our next step is to explore R visuals and what we have access too. |  |

## What Is R? – Creating Visauls in R (Part 1)

Now that we’ve covered basics of syntax and data frames, we can start to go over visuals in R. This section is first going to go through some basic default visuals in R, then we will have a look at some customization and more advanced visual packages. It is key to remember that once we have a data frame ready, most visuals are easy to implement. The hardest part is finding the right package and process for what you are trying to do.

### What is R? – Creating Visuals in R

|  |  |
| --- | --- |
| We are going to go through some simple examples first. For our first two visuals we will just be creating some simple data to work with. After this we will import some data to work with. Remember the idea behind these exercises is to get our feet wet in some of the simple analysis that R can enable.   1. First, we need to create some new data. You can copy and paste this into your console   numbers = c(6, 6, 6, 5, 5, 8, 9, 3, 4, 2, 3, 4, 5, 8, 9, 0, 5, 2, 3, 4, 5, 1, 4, 6, 3)   1. Now type “hist(numbers)”. Observe how a histogram was created in the bottom right of RSutdio. This displays the number of occurrences of each number in our array. Notice how in groups up numbers in pairs of two. We can customize this. 2. Type “hist(numbers, breaks = 10)”. Now our histogram has a column for each digit. Let’s try a little more customization. 3. Now we are going to build a normal curve of best fit. Type   xfit = seq(min(numbers),max(numbers),length=40)  and  yfit=dnorm(xfit,mean=mean(numbers),sd=sd(numbers))  and  h = hist(numbers, breaks = 10)  and lastly  yfit=yfit\*diff(h$mids[1:2])\*length(numbers)  This process created the normal distribution of best fit and used this to approximate a line of best fit.   1. Type “lines(xfit,yfit, col = “blue”)”.   Every visual in R has many parameters for customization. We cannot go through all of them, however, they will always be just a Google search away.  Next, we are going to work with a scatter chart. Again, as another example of how we can extract information from data using some ‘basic’ formulas.   1. Enter the following arrays into R.   x = c(1,3,4,7,8,13,14,18,22,21,22,24,25,28,30,35)  y = c(1,171, 16, 20, 124, 299, 135, 450, 244, 591, 424, 726, 595, 624, 1170, 1225)   1. Now type “plot(x,y)”. Observe the scatter plot. Now we want to see what we can determine about this plot. We are going to experiment with creating lines of best fit. 2. Type “ lm = lm(y~x) “. In this statement lm stands for linear model. We now have an object lm which represents the line of best fit for this data. Type ‘lm’ and hit return to see the details. Recall a line in the plane is represented as y = ax + b. a is the slope, and b is the intercept. 3. Type “abline(lm(y~x))”. Have a look at our line of best fit! This is linear regression. It looks like we can do a bit better though. 4. Our next step is to create a quadratic model. Recall a quadratic is in the form y = ax^2 + bx + c. We need a new variable to represent x^2. Type “x2 = x^2”. 5. Now we are going to create our model.   Type “q = lm(formula = y~x+x2)” This gives us a line of best fit in two variables.   1. Now we need to use this model to create a new line. Enter the following into R and have a look!   xmodel = seq(0,35,0.5)  ymodel = predict(q, list(x=xmodel,x2=xmodel^2))  lines(xmodel,ymodel)  This is quadratic regression!   1. You can also use this model to predict a specific value. Let’s predict for x = 50. Type the following.   predict(q,list(x=50,x2=50^2))   1. Before we finish up with this example, I want to do one more thing.   Type “cor(x,y)”. This gives you the ‘correlation coefficient’ between x and y. This number shows you how positively or negatively the data points are associated.  These are the kinds of operations and results that R offer that are not so easy to obtain within PowerBI. Linear and quadratic regression are very simple examples of machine learning and prediction. R can do far more than just those!  The purpose of these exercises has been to see the process in which one can explore data within R. The most challenging part of building visuals within PowerBI is testing. It is very difficult to test much before the final product  This is where RStudio comes in. |  |

## What Is R? – Creating Visuals in R (Part 2)

Now we are going to develop a visual using a dataset that we will eventually port into PowerBI. This one is a going to be more advanced and use a much larger dataset. It will utilize the package ggplot2 which has an extensive visual library for R.

Once we have a visual that we are happy with here, we are going into PowerBI. We will import the same dataset and use R in PowerBI to recreate our visual and create some interactive slicers to go along side it.

### What is R? – Creating Visuals in R

|  |  |
| --- | --- |
| We have covered some of the basic visuals available in R. Now we are going to explore some options in ggplot2 to try and find a visual that we would want in our PowerBI report.   1. Our first step in this exploration is to import some data. So far, we have only worked with small arrays of our creation. This time we are going to import our project dataset into R. To do this click File -> Import Dataset -> From Text (base). From here you will navigate to our content work book and choose Surgeries.csv. From here click Import. 2. Have a look this created a new data frame for us. Type “head(Surgeries)” to get the first 6 items of each column. 3. Now we want to start utilizing some of this data to create a visual using ggplot2. We first must install ggplot2. Type ‘install.packages(“ggplot2”)’ and hit return. 4. Now we need to load the package into our library. Type “library(ggplot2)”. Any time you want to use a package in R, you simply must follow those two steps to load it up. 5. Now we are going to start to build our visual. ggplot is an incredible visual package in R. It is one of the most used visual packages. It can be a bit strange to work with at first, but it is built to be flexible and reasonable to learn. There are many resources to learn how to use ggplot but [here is one](https://ggplot2.tidyverse.org/), and [another](http://r-statistics.co/ggplot2-Tutorial-With-R.html), that I recommend for you if you wish to try more outside of this tutorial. 6. To summarize quickly, a plot in ggplot consists of layers built upon a canvas. The first step we will take, is to create the canvas. To do this type (or copy & paste) the following:   gg = ggplot(Surgeries , aes(x = Surgeries$FISCAL\_YEAR, y= Surgeries$WAITING))  This creates a canvas with ‘aesthetic’ defining the x and y axis. Type “gg” and hit return and we should see a blank canvas.   1. Now we are going to start adding (literaly) some ‘geoms’, or geometric representations, to this canvas. There are many different geoms that work in different aesthetic structures. Type:   gg2 = gg + geom\_point()  Now type “gg2” and you should see a canvas with our points represented as dots.  Now notice our x axis is very static. All our values are on integers. This is fine, but I think we can make a more interesting viewing experience by “shifting” them all a little bit.   1. We are going to recreate this canvas with a different geom.     gg2 = gg + geom\_jitter()  Now type “gg2” and hit enter to see this effect. This time we used a different geom which is like plotting points but avoids overlap when it can by shuffling the x axis.  This is interesting, but I think we can add some more to this!   1. Type “gg2 = gg + geom\_jitter(aes(color = Surgeries$HEALTH\_AUTHORITY))” and type gg2 to see the result! Now we adjusted the aesthetic to include HEALTH\_AUTHORITY as the color of our points. 2. Now let’s add lines which represent movement of the average as well. 3. Type “gg2 = gg2 + geom\_smooth(aes(color= Surgeries$HEALTH\_AUTHORITY))”   Type “gg2” to see the result! Notice here how we added the lines directly to our plot with the points already included. We can iterate over our plot like this as much as we wish!   1. Next, just for the sake of it, let’s add the line for the linear model of our entire data set combined to this chart as well. Type this to create our linear model:   lm=lm(Surgeries$WAITING~Surgeries$FISCAL\_YEAR)  Now unfortunately we can’t quite plug this model into a chart like before. This time we must extract our coefficients from the model (remember y = ax + b). This is kind of crazy, but it turns out that our model is an object pretty similar to a data frame. We can access different “columns” which each have different data (type summary.lm(lm) if you are interested.)  To extract the slope and intercept type:  slope = summary.lm(lm)$coefficients[2,1]  And,  int = summary.lm(lm)$coefficients[1,1]       1. Now we get to add this line to our plot. 2. Type the following:   gg2 = gg2 + geom\_abline(intercept = int, slope = slope , color = “black”)  And now type gg2 to see the result.  Here we have it. We’ve created a visual using ggplot. While it isn’t anything too fancy, it demonstrates the power of ggplot2.  As one final note I’d like to mention that you can change the scale in which you are plotting **extremely easy**. To the right is an example of plotting this exact same visual except with our y axis as sqrt(WAITING). Our original canvas is modified as such:  gg = ggplot(Surgeries, aes(x= Surgeries$FISCAL\_YEAR, y=sqrt(Surgeries$WAITING)))  While not technically a rigorous way of viewing this data, this shows us the utility that this sort of plotting can provide. Note our linear model is not on this chart because we would have to recreate our model in the square root space for it to fit in this frame. |  |

# Utilizing R in PowerBI

Alright, so far, we have explored some basic syntax in R, used it to build some basic visuals, and then took a step forward and built a visual using the package ggplot2. Our next goal is to rebuild that visual using R Script in PowerBI.

The key difference between working in RStudio and R Script is that we can’t test each line individually. When PowerBI runs the script it goes through everything we write and if there are any errors it will spit back an error. Henceforth, we must always work carefully, ensuring that we can always recognize where our errors are coming from.

## Utilizing R in PowerBI – Using What we Learned in R Studio.

Let’s get started.

### Utilizing R in PowerBI – Using What we Learned in R Studio.

|  |  |
| --- | --- |
| Okay, so now we need to get into PowerBI. Open a blank work book. Our first step is going to be to import our surgeries data.   1. Click the drop down by get data, select text/csv. Then find and select our Surgeries.csv folder in our content pack. Click load. 2. Next, we are going create an R visual. Under the visualizations tab click the R visual icon. You should see a visual appear along with a new text box down below.   Let’s resize this visual so it’s easier to see what we are doing.   1. Now we need to drag our data that we are going to need into our R visual. Recall we only used WAITING, FISCAL\_YEAR, and HEALTH\_AUTHORITY. These names in the boxes under values are how we are going to address these items in our script.   **Now it is very import to set our values to ‘Don’t Summarize’ in our values. Do this now by clicking the drop down beside the title.**   1. Now have a look at our Script. Notice the first few lines are comments explaining our situation. We have a data frame named dataset. We will need to address our data using this data frame name. 2. First thing is simple. We need to include ggplot2 in our Script. Type “library(ggplot2)” into our script. 3. Now we need to create our canvas, type:   gg = ggplot(dataset, aes(x=FISCAL\_YEAR,y=WAITING))  Notice how we used ‘dataset’ this time rather than ‘Surgeries’. That is because our data is aliased differently this time.   1. Let’s test our script. Add gg to the end of our script and the click the ‘run’ button at the top right of the script window. 2. Now lets add our jitter and test this out. Type:   gg = gg + geom\_jitter(aes(color = dataset$HEALTH\_AUTHORITY))  Make sure this occurs before the final ‘gg’ call, then hit run and check that this works. Our script should look like the image to the right.   1. Now to add our average lines:   gg = gg + geom\_smooth(aes(color = dataset$HEALTH\_AUTHORITY))  Again, run this to make sure it works. This should be inserted between our last geom, and our final call of gg. Have a look at the result!   1. Lastly, we get to create our regression line. Insert our steps at the beginning of our script.     model = lm( dataset$WAITING ~  dataset$FISCAL\_Year)  int = summary.lm(model)$coefficients[1,1]  slope = summary.lm(model)$coefficients[2,1]  and lastly insert our last geom onto our canvas.  gg = gg + geom\_abline(slope = slope, intercept =  int, colour = “black”)  Our final script should look something like the one to the right.   1. Our last step is to run this and see how it looks.   Now the cool stuff comes in the interaction! Let’s add some slicers and see what we can see.   1. I prefer the ‘chiclet’ slicer from the visual market place. So, let’s go ahead and include that.   Click on the three ellipses in the visualizations selection pane. Then select ‘Import From Marketplace’. From here use the search bar to find Chiclet Slicer and load it into your report by clicking ‘Add’.  I’m going to create two slicers next to my visual, one for hospital name and the other for procedure.     1. With the white space selected, click the Chiclet slicer icon in the Visualizations pane. Click and drag HOSPITAL\_NAME into this one. Repeat this process with PROCEDURE\_GROUP. 2. Now I am going to do some simple analysis. Lets create a Card with the average waiting for 2016.   Create a new measure by right clicking on our table in the fields section, and then selecting ‘New measure’. Enter the following:  2016 avg = CALCULATE(AVERAGE(Surgeries[WAITING]),FILTER(Surgeries,Surgeries[FISCAL\_YEAR] = CALCULATE(MAX(Surgeries[FISCAL\_YEAR]))))  Now drag this measure onto our canvas and then with the visual selected click the card icon in the Visualizations pane.   1. Now I want to predict the average waiting for surgeries in 2020 using our regression line.   There is a problem however; we cannot pass a value out of our R Script. So, we are going to have to get creative.  In our script lets add the following.  predict = slope\*2020 + int  max = max(dataset$WAITING,na.rm = TRUE)  text = paste("2020 Predicted Average Waiting is",  round(predict,digits = 2))  Each of these three lines does something important for us.  First, we get our predicted value,  second, we find the max waiting value,  third we need to create a text to display.  Now we are going to display this in a text box in our visual.   1. We are going to add this text to our canvas again using the following.   gg = gg + geom\_text(x=2011,y=max\*0.95,label= text, size = 5)  Our final script should look like the one to the right. When we run this, we get the following.  Copied below is our final script as well.  library(ggplot2)  model = lm(dataset$WAITING~dataset$FISCAL\_YEAR)  slope = summary.lm(model)$coefficients[2,1]  int = summary.lm(model)$coefficients[1,1]  predict = slope\*2020+int  max = max(dataset$WAITING,na.rm = TRUE)  label = paste("2020 Predicted Average Waiting is", round(predict,digits = 2))  gg = ggplot(dataset, aes(x=FISCAL\_YEAR,y=WAITING))  gg = gg + geom\_jitter(aes(color = dataset$HEALTH\_AUTHORITY))  gg = gg + geom\_smooth(aes(color = dataset$HEALTH\_AUTHORITY))  gg = gg + geom\_abline(slope = slope, intercept = int, colour = "black")  gg = gg + geom\_text(x=2011,y=max\*0.95,label= label,size = 5)  gg |  |

# Conclusion:

And this concludes our R in PowerBI training manual. I hope you’ve found this a helpful resource in practicing and learning the foundations of R and how it can be used in PowerBI. The applications and techniques available to you now in R now extend vastly in machine learning, visualizations and all sorts of other mathematical and statistical constructs. It is now up to you to explore and figure out what you can do with it.

## Conclusion: Limitations

There are a few limitations with using R in PowerBI. There are a few key ones to be aware of that I will mention here. The rest can be found [here.](https://docs.microsoft.com/en-us/power-bi/desktop-r-visuals)

* Data size limitations – data used by the R visual for plotting is limited to 150,000 rows. If more than 150,000 rows are selected, only the top 150,000 rows are used and a message is displayed on the image.
* Calculation time limitation – if an R visual calculation exceeds five minutes the execution times out, resulting in an error.

## Conclusion: Questions

Now is the time for your questions and requests. Let me know if you have any projects you wish to work with while I am available. Or, let me know if you have any higher-level questions about machine learning in general and where to start with that.