ADVANCED ANALYTICS WITH POWER BI AND R

RADACAD

WRITTEN BY
DR LEILA ETAATI

PUBLISHED BY
RADACAD Systems Limited
http://radacad.com
89A Fancourt street, Meadowbank,
Auckland 1072 New Zealand

Cover: Freda Fung

About the book; Quick Intro from Author

In 2016, after brining the capability of writing R codes inside Power BI, I've been encouraged to publish an online book through a set of blog posts. The main reason to publish this book online, was that there is no integrated and comprehensive book on how to use R inside Power BI. From that time till now, I've been writing blog posts (or sections) of this book almost weekly in RADACAD blog. So far, I have more than 20 sections wrote in this book. This book covers most aspects of R inside Power BI; from creating R visual inside Power BI, how to run Machine Learning algorithm and how to create R custom visual. This book explains the main concepts of machine learning, R from novice to professional level. You can start reading this book with no prerequisite. I recommend to follow the book structure rather than read each section by itself. However, there are some sections, you don't need to follow specific order. After six months of writing online, I decided to release this book as a PDF version as well, for two reasons; First to help community members who are more comfortable with PDF books, or printed version of materials. Second as a giveaway in my Advance Analytics training courses. Feel free to print this book and keep it in your library, and enjoy. This book is FREE! This book will be updated with updated editions (hopefully every month), so you can download the latest version anytime from my blog post here :http://www.radacad.com I will do my best to update any changes in next few editions. Just to keep you informed, the publish date of each section is mentioned at the beginning of each section under the header.

About Author

Leila Etaati is invited speaker in world's best and biggest SQL Server and BI conferences such as Microsoft Data Insight Summit, PASS Summits, PASS24H, SQL Nexus, PASS Rallys, SQLBits, TechEds, Ignites, SQL Nexus, SQL Days, SQL Saturdays and so on. She obtained her PhD in Information System from University of Auckland. She has more than 10 years experience in Microsoft technologies. More than 5 years of her experience focused on training and consulting in Machine Learning Concepts and BI Technologies. She is Microsoft Data Platform MVP (Most Valuable Professional) focused on BI and Data Analysis, She has been awarded MVP from Microsoft because of his dedication and expertise in Microsoft BI technologies from 2016 till now. These days Leila runs Advance Analytics training, consulting, and mentoring in many cities and countries around the world (USA, Canada, Europe, Asia, Australia, and New Zealand). She trained more than 100 students in just last few months for Microsoft Advance Analytics training. Leila lives in Auckland, New Zealand, but you will probably see her speaking in conferences, or teaching courses near your city or country from time to time. If you are interested to be in touch with Leila, or learn about her upcoming courses, visit RADACAD events page. http://radacad.com/events

Upcoming Training Courses

Leila runs Advance Analytics with R, Power BI, Azure Machine Learning and SQL Server training courses both online and in-person. RADACAD also runs a course by Reza Rad On Power BI both online, and in-person in major cities and countries around the world. Check schedule of upcoming courses here:

http://radacad.com/events
http://radacad.com/power-bi-training
http://radacad.com/advanced-analytics-training
http://radacad.com/analytics-with-power-bi-and-r

some of upcoming events in next few months:

31st of July 2017- Live Instructure Led 3-days course for USA East and Europ
16th of Aug 2017-Advanced Analytics-Bangalore
11th of September 2017-Live Instructor Led 3-days Course, Asia and Australia
25th of September 2017- Live Instructor Led 3-days Course, US West
9th of October 2017- Live Instructor Led 3-days Course, Australia East
16th of October 2017-Advanced Analytics, Auckland, New Zealand
20th of November 2017- Advanced Analytics, Wellington, New Zealand

Who Is This Book For?

This book is designed for BI Developers, Consultants, Data scientists who wants to know how to develop machine learning solutions inside Power BI. BI Architects and Decision Makers who wants to make their decision about using or not using R visuals or Machine Learning inside Power BI in their BI applications. Business Analysts who want to get better insight on data and learn tricks of how to apply machine learning on specific data. The book titled "Advance Analytics with Power BI and R", and that means it will cover wide range of readers. I'll start by writing 100 level and we will go deep into 400 level at some stage. So, if you don't know what Power BI is, or If you are familiar with R but want to learn how to use Power BI, this book able to show you the main process.

Heading Table of Content

1-R Data Structures for Machine Learning

Published Date : January 9, 2017

Every programming language has specific data structure. R language also has some predefined data structures that each serves specific purpose. For doing machine learning in R, we normally use data structure such as Vector, List, Data Frame, Factors, Arrays and Matrix. In this post I will explain some of them briefly.

Vector – C()

Vector stores the order set of values. Each value belongs to a data type. Vector can hold data types like **Integer** (numbers without decimals), **Double** (numbers with decimals), **Character** (text data), and **Logical** (TRUE or FALSE values).

Integer	Integer	Integer	Integer	Integer	Integer	Integer	Integer	Integer	Integer

We use Function **C ()** to define a vector to store people name.

```
subject_name <- c("Jane", "Mike", "Petter")
```

Subject_name is a Vector that contains Character value (People name).

We can use the **Typeof ()** to determine the type of Vector.

```
typeof(subject_name)
```

The output will be:

```
> typeof(subject_name)
[1] "character"
```

Now we are going to have another vector that stores the people age.

```
Age<- c(20,56,34)
```

The Age vector stores Integer value. We create another vector to store a Boolean information about whether people married or single:

```
Married_status<-c(FALSE, TRUE,FALSE)
```

Using the **Typeof ()** Function to see the Vector type:

```
> typeof(Age)
[1] "double"
> typeof(Married_status)
[1] "logical"
```

We can select specific elements of the each vector, for example to extract the second name in Subject_Name vector, we write below code:

```
subject_name[2]
```

which the output will be:

```
> subject_name[2]
[1] "Mike"
```

Moreover, there is a possibility to get the range of value in a Vector. For example, we want to fetch the age of second and third person we stored in Age vector, the code should be look like below:

```
Age[2:3]
```

The out put will be like:

```
> Age[2:3]
[1] 56 34
```

Factor – Factor()

Factor is specific type of Vector that stores the categorical or ordinal variables, for instance, instead of storing the female and male type in a vector, computer stores 1,2 that takes less space in storage, for defining a Factor for storing gender we first should have a vector of gender as below:

C(“Female”, “Male”)

then we use commend Factor() as below

```
> gender <- factor(c("MALE", "FEMALE", "MALE"))
> gender
[1] MALE   FEMALE MALE
Levels: FEMALE MALE
```

as you can see in above output, when I called the “gender” , it shows genders of people that we stored in Vector plus a value called “Level”, Level show the possible value in gender vector.

for instance, currently we just have BA and Master students . However, in the future there is a possibility that we have PhD or Diploma students. So we create a factor as below that can support future types as well:

```
> student_level<- factor(c("BA","Master"), levels = c("BA","Master", "PhD","Diploma"), ordered = TRUE)
> student_level
[1] BA     Master
Levels: BA < Master < PhD < Diploma
```

we should specify the “**Levels**” like this: levels = c(“BA”,”Master”, “PhD”,”Diploma”)

Lists-list()

List is similar to vector but is able to have a combination of data types whilst in Vector we just can have one data type.

Integer	String	Double	character	Logical

For instance: To store students’ information we can use list as below:

```
students<-list(StudentName=subject_name[1],StudentAge=Age[1],StudentLevel=Student_level[1])
```

the out put of calling students list will be look like:

```
> students
$StudentName
[1] "Jane"

$StudentAge
[1] 20

$StudentLevel
[1] BA
Levels: BA < Master < PhD < Diploma
```

List helps us to have combination of data types.

Data frames- data.frame()

The most important data structure in machine learning process is Data Frames. Similar to Table, it has both columns and rows.

Integer	String	Double	character	Logical
Integer	String	Double	character	Logical
Integer	String	Double	character	Logical
Integer	String	Double	character	Logical
Integer	String	Double	character	Logical

To define a Frame we use data.frame syntax as below:

```
StudentData <- data.frame(subject_name, Age, gender, Student_level, stringsAsFactors = FALSE)
```

studentData is a data frame that contains some vectors including subject_name, Age, Gender and Student_Level.

R automatically converts every character vector to a factor, to avoid that we normally use StringAsfactor as parameter that specify character data type should not be considered as factor.

the output of calling Studentdata will be look like:

```
  subject_name Age gender student_level
1         Jane  20 FeMALE            BA
2         Mike  56   MALE        Master
3       Petter  34   MALE        Master
```

As data frame is like a table we can access the cells, rows and columns separately

for instance, to fetch a specific column like age we use below code:

```
> StudentData$Age
[1] 20 56 34
```

only the Age column as a Vector has been shown.

Moreover, we just want to see age and gender of students so we employ below code:

```
> StudentData[c("Age","gender")]
  Age gender
1  20 FeMALE
2  56   MALE
3  34   MALE
```

we can extract all the rows of the first column:

```
> StudentData[,1]
[1] "Jane"   "Mike"   "Petter"
```

or extract all columns data of specific students using below code

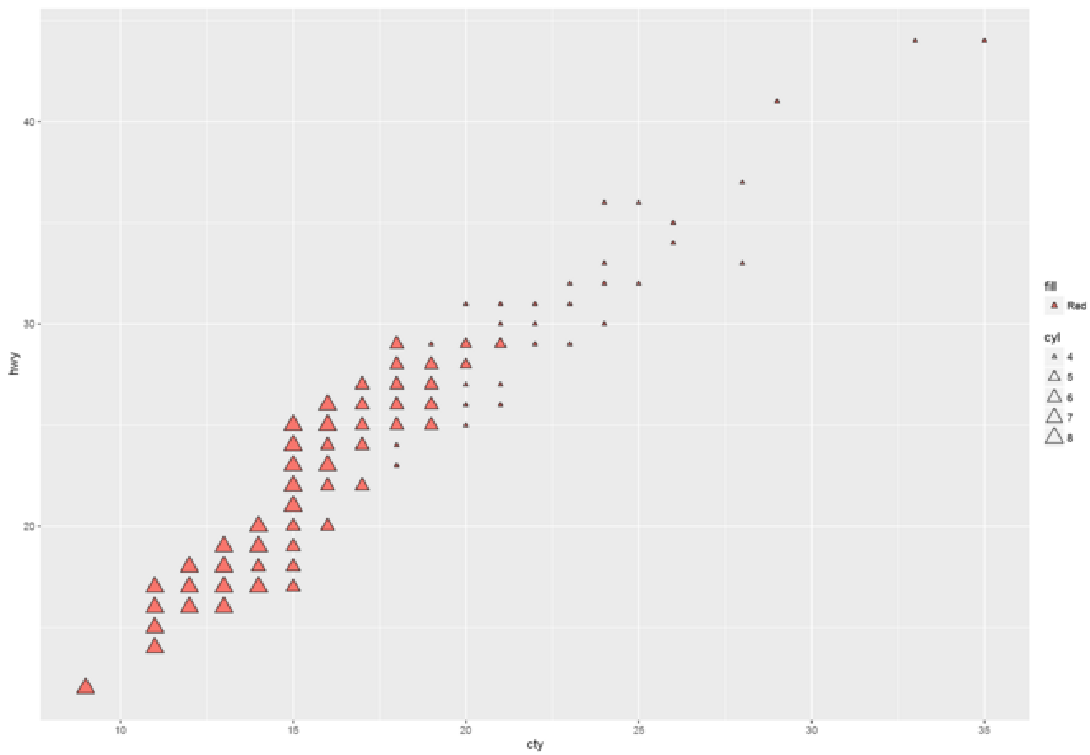
```
> StudentData[1,]
  subject_name Age gender student_level
1         Jane  20 FeMALE            BA
```

in next post I will show how we can get data from different resources and how to visualize the data inside R.

Reference:L. Brents. Machine Learning with R, Pack Publishing, 2015

2-Have More Charts by writing R codes inside Power BI: Part 1

Published Date : April 7, 2017



Power BI recently enable users to embed R graphs in Power BI. There are some R visuals that would be very nice to have them in Power BI.

What is R ? Based on Wikipedia, R is an open source programming language and software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls, surveys of data miners, and studies of scholarly literature databases show that R's popularity has increased substantially in recent years.

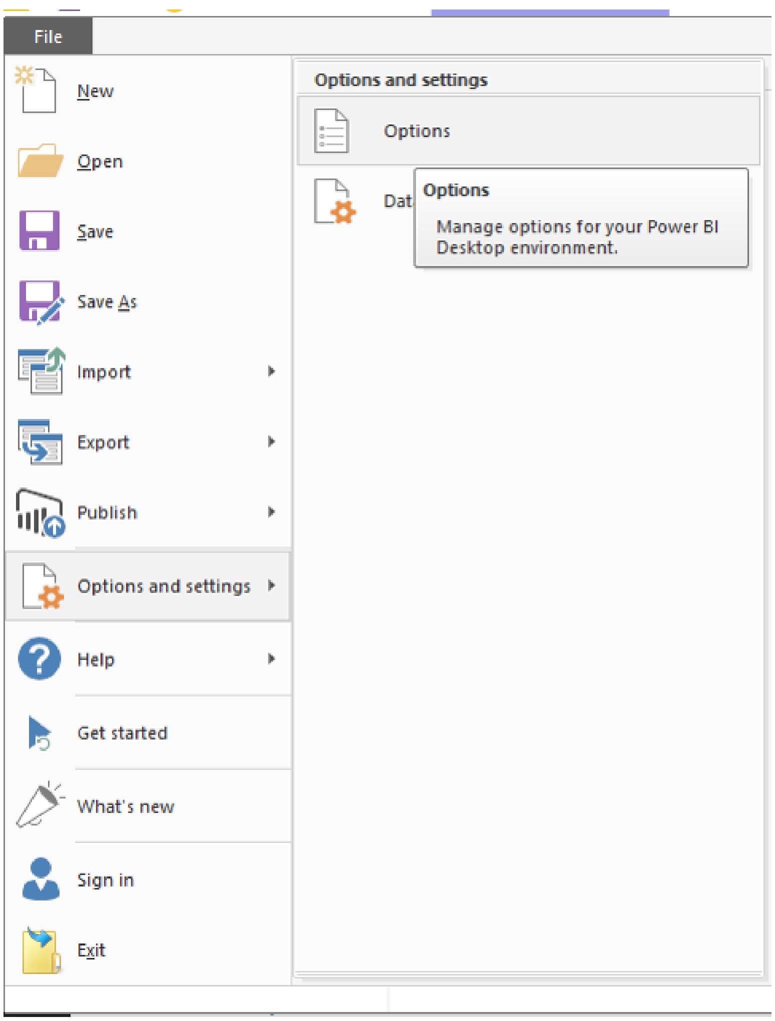
R has more than 1000 packages to perform different tasks. "ggploy2" is the main package for drawing visuals which contains various functions to draw different type of charts.

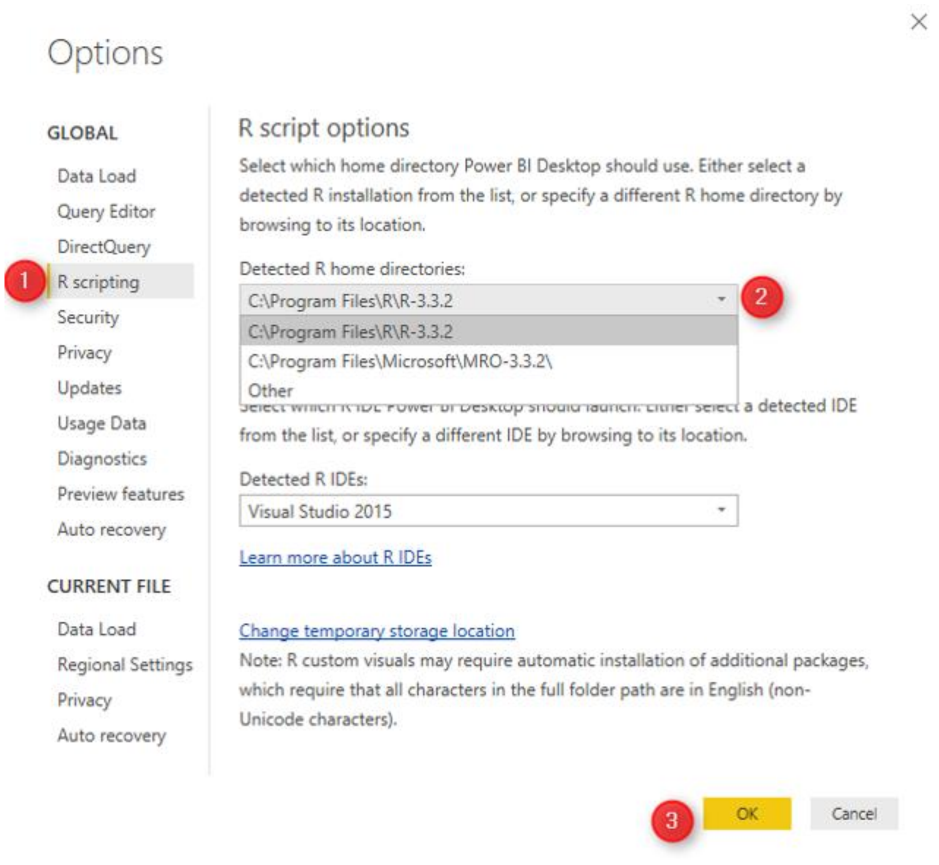
How to start?

First download one version of R in your machine, I have downloaded “Microsoft R open” in my machine from here.

then open power bi desktop or download it from here

in power bi click on the File menu, then click on the “Options and Settings” then on ” Options”. under the “Global” option click n the “R Scripting” specify the R version.





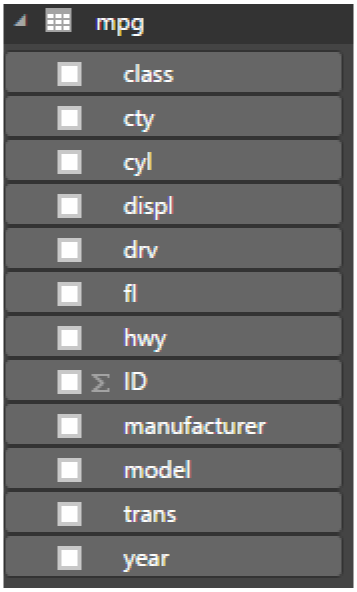
For the first time working in R, it is required to install the packages you need. in this example I am going to use "ggplot2" in power bi so first I have to open Microsoft R open and type

install.packages("ggplot2")

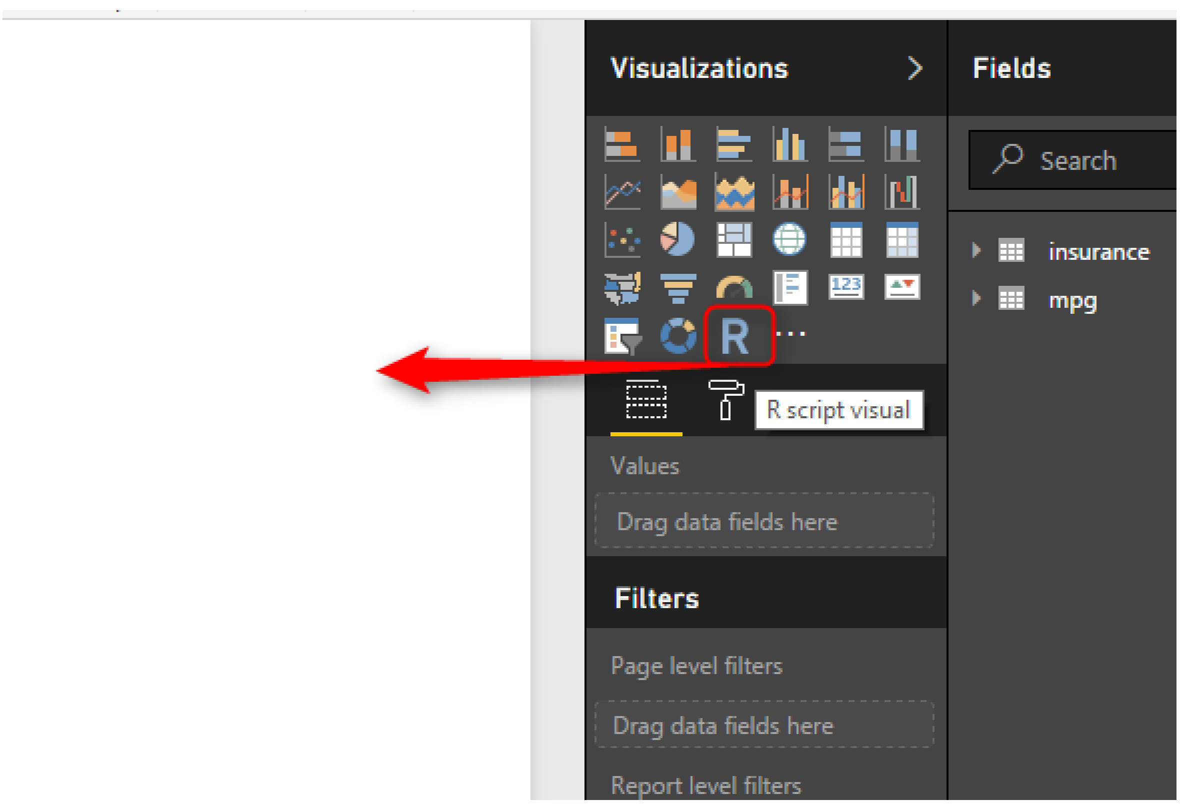
ggplot2 will install some other packages itself.

now I can start with power BI.

In power BI I have a dataset, that show specifications of cars such as : speed in city and highway, cylinder and so forth. if you interested to download this dataset, it is free name "mpg" from [here.](#)

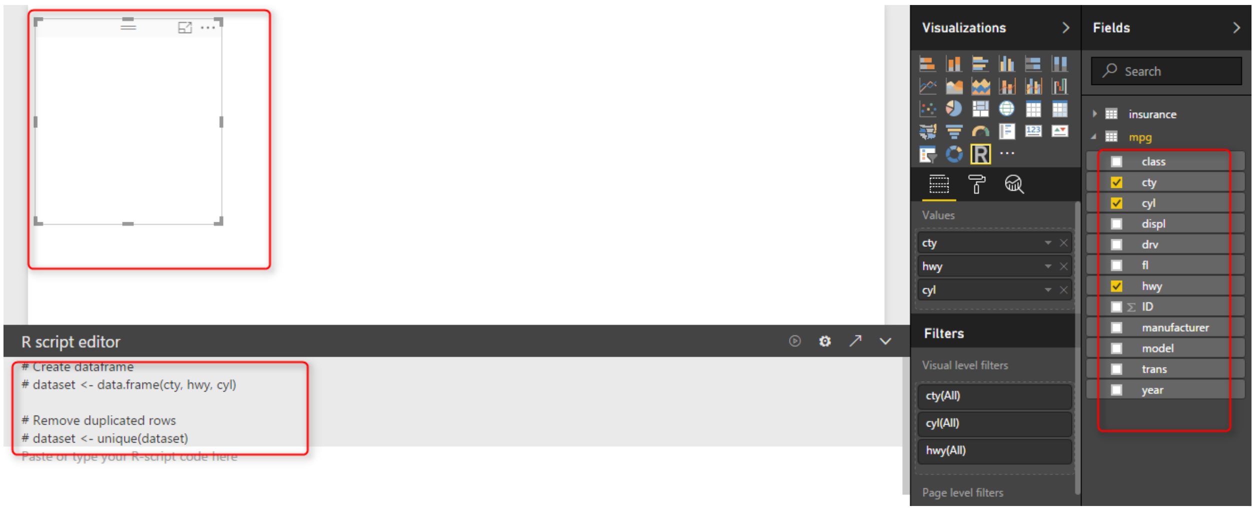


then just click on the R visual in Power BI "R" visual and put in the white space as below.



After bringing R visual in the report area. and by selecting "cty" (speed in city), "hwy" (speed in highway), and "cyl" (cylinder"). then in the "R script editor" you will see R codes there!

"#" is a symbol for comments in R language which you can see in R scripts Editor. Power BI automatically puts the selected fields in a variable name "dataset" so all fields (cty,hwy, and cyl) will store in a dataset variable by "<-" sign. also it automatically remove the duplicated rows. all of these has been explain in R script editor area.



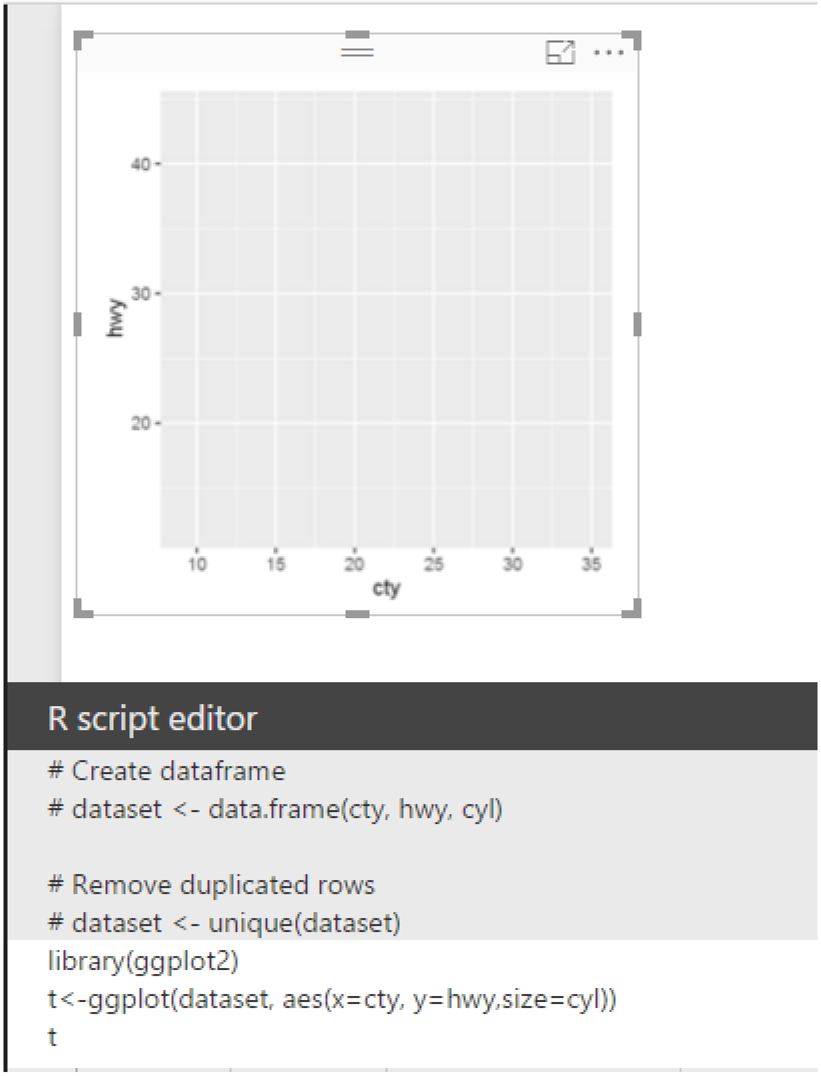
next we are going to put our R code for drawing a two dimensional graph in power BI. In power BI, to use any R scripts, after installing in R version that we have, we have to call the packages using "library" as below

library(ggplot2)

So always, whatever library you use in power bi, call it by library function first. There are some cases that you have to install some other packages to make them work, based on my experience and I think this part is a bit challenging!.

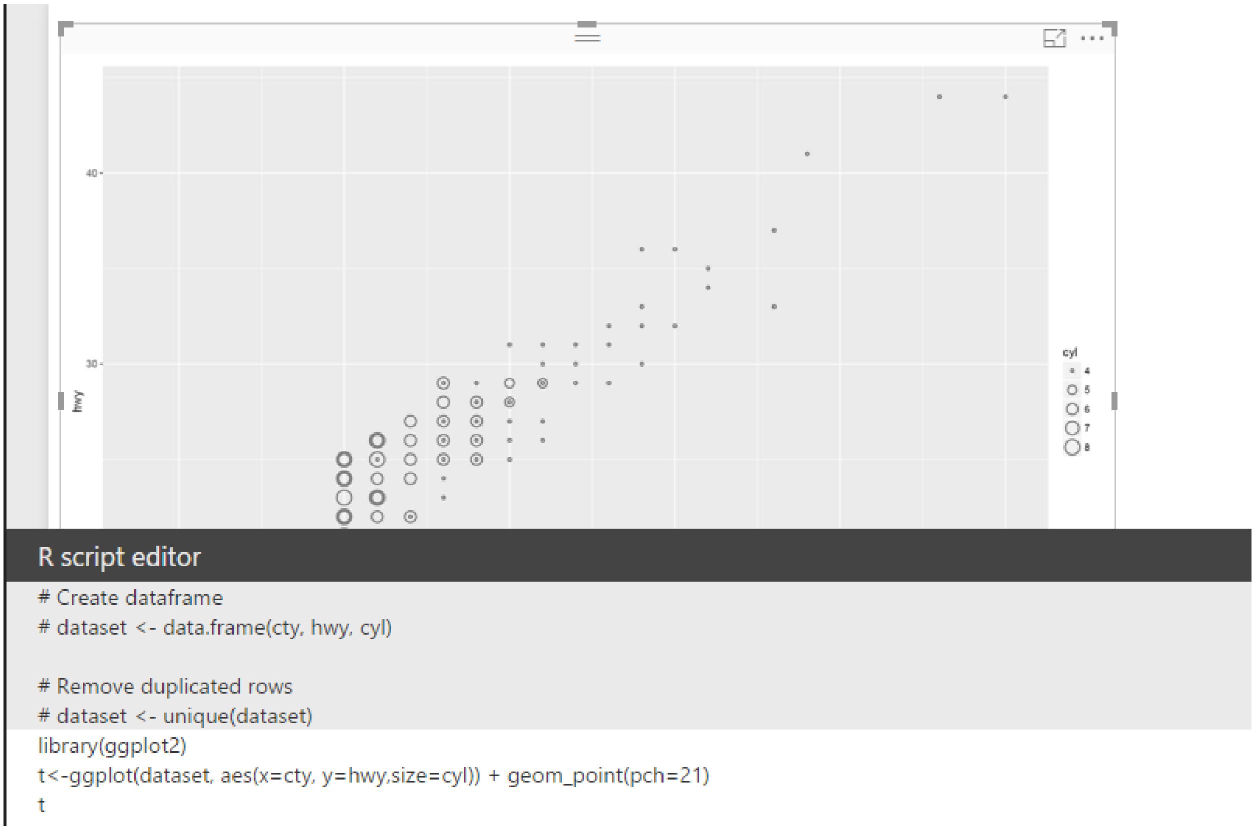
To draw a chart I first use "ggplot" function to draw a two dimensional chart. The first argument is "dataset" which holds our three fields. Then we have another function inside the ggplot, named "aes" that identify which filed should be in x axis or in y axis. finally I also interested to shows the car cylinder in chart. This can be done by adding another layer in aes function as "Size". so bigger cylinder cars will have bigger dots in picture.

t<-ggplot(dataset, aes(x=cty, y=hwy,size=cyl))



However, this just show the graphs with out any things! we need a dot chart here to create that we need to add other layer with a function name

geom_point enables to draw a scatter charts. This function has a value as pch=21 which the shape of the dot in chart, for instance if I put this value as 20 it become a filled cycle or 23 become a diamond shape.



In the above picture we can see that we have 3 different fields shown in the chart: highway and city speed in y and x axis accordingly. While the car's cylinder variable has been shown as different cycle size. However may be you need a bigger cycle to differentiate cylinder with 8 to 4 so we able to do that with add another layer by adding a function name

```
scale_size_continuous(range=c(1,5))
```

and whole code will be as below :

```
t<-ggplot(dataset, aes(x=cty, y=hwy,size=cyl)) + geom_point(pch=23)+scale_size_continuous(range=c(1,5))
```

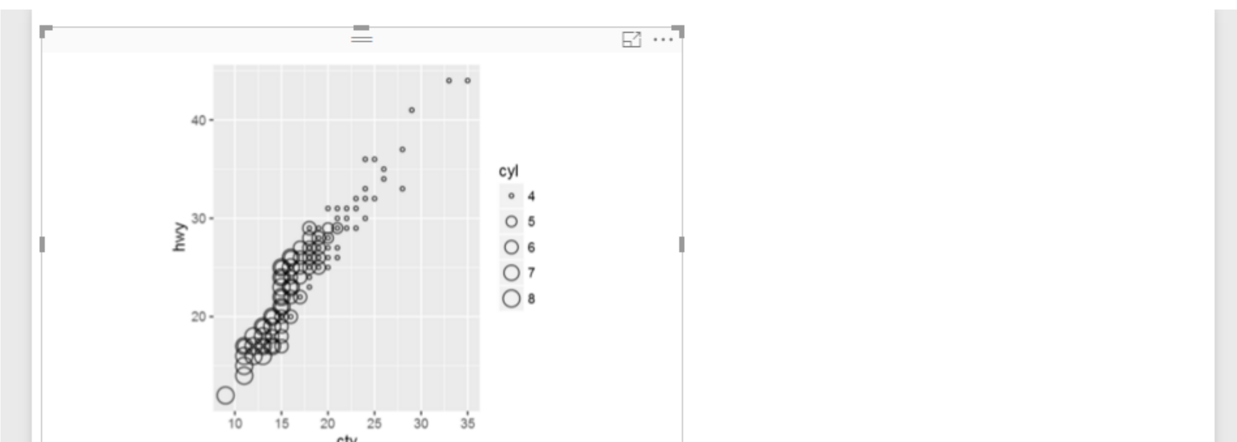
in the scale_size_continues(range=c(1,5)) we specify the difference between lowest value and highest one is 5, I am going to make this difference bigger by change it from 5 to 10

so the result will be as below picture:

R script editor

```
# Create dataframe
# dataset <- data.frame(cty, hwy, cyl)

# Remove duplicated rows
# dataset <- unique(dataset)
library(ggplot2)
t<-ggplot(dataset, aes(x=cty, y=hwy,size=cyl)) + geom_point(pch=21)+scale_size_continuous(range=c(1,10))
t
```

so now in picture the difference is much more obvious. Finally we have below picture



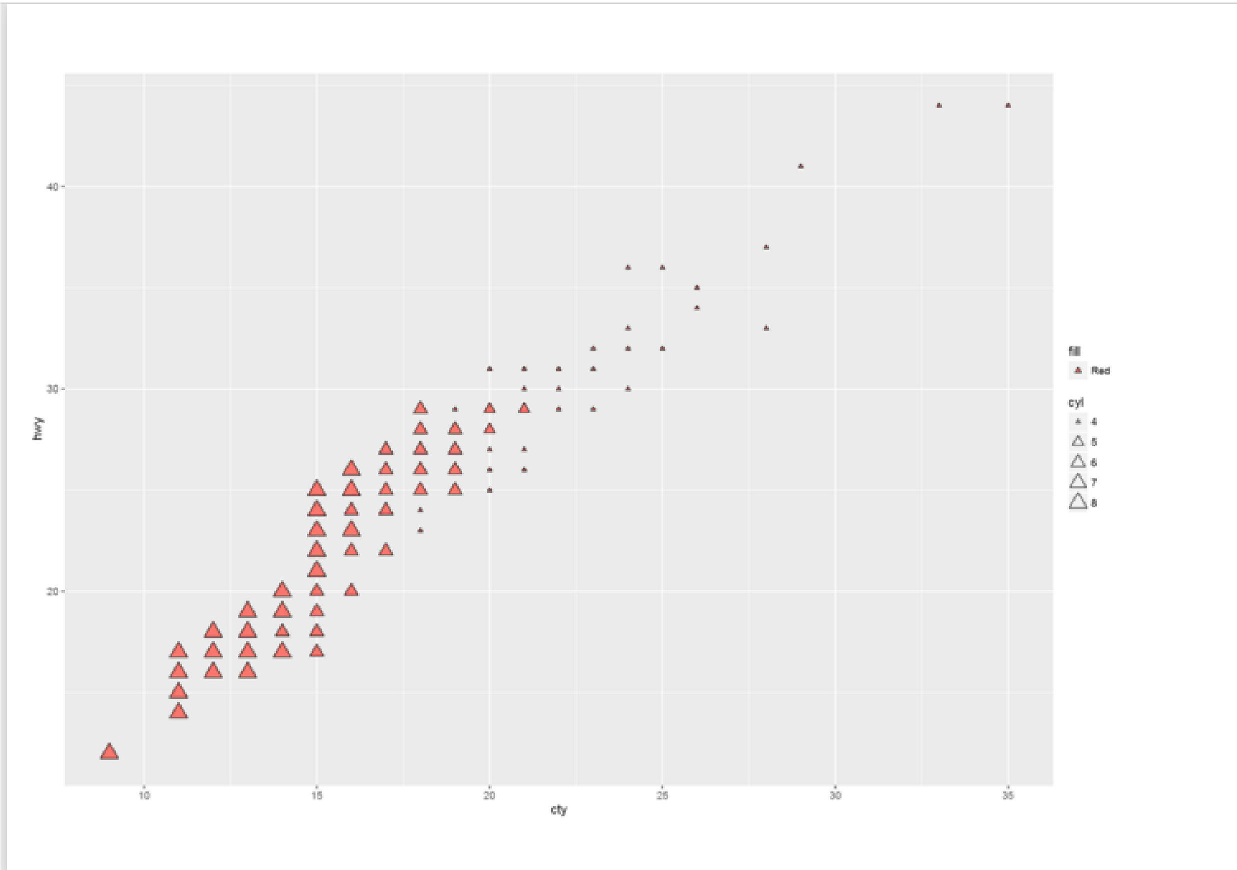
R script editor

```
# Create dataframe
# dataset <- data.frame(cty, hwy, cyl)

# Remove duplicated rows
# dataset <- unique(dataset)
library(ggplot2)
t<-ggplot(dataset, aes(x=cty, y=hwy,size=cyl)) + geom_point(pch=21)+scale_size_continuous(range=c(1,5))
t
```

in the other example I have changed “pch” value to 24 and I add another code inside of ” aes” function name “fill=Red” that means I want rectangle filled in red colour instead

```
t<-ggplot(dataset, aes(x=cty, y=hwy,size=cyl,fill="Red")) + geom_point(pch=24)+scale_size_continuous(range=c(1,5))
```

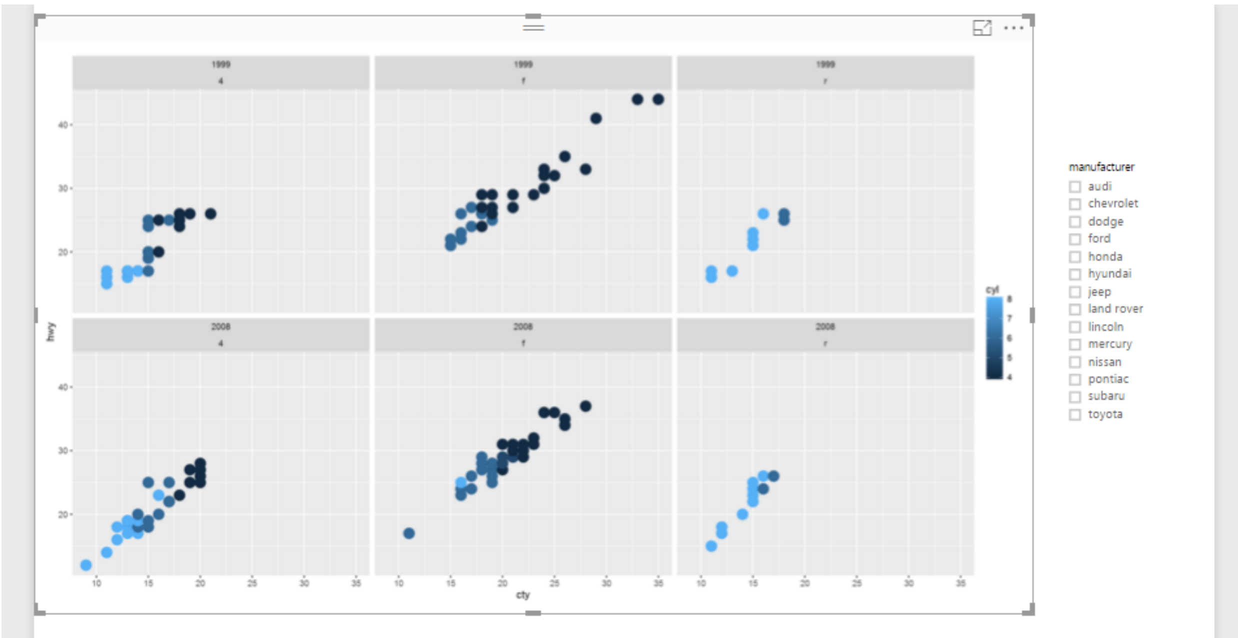
then I have below chart:



It is possible to show 5 different variables in just one chart, by using facet command in R. This will help us to have more dimension in our chart, This will be explained in the next post (Part 2).

3-Have More Charts by writing R codes inside Power BI: Part 2

Published Date : April 8, 2017



In the previous post (Part 1) I have explained how to write a simple scatter chart inside Power BI. Now in this post, I am going to show how to present 5 different values in just one chart via writing R scripts.

I will continue the codes that I wrote in the previous post as below :

```
library(ggplot2)
t<-ggplot(dataset, aes(x=cty, y=hwy,size=cyl,fill="Red")) + geom_point(pch=24)+scale_size_continuous(range=c(1,5))
t
```

In previous post ,we have shown 3 variables : speed in city, speed in highway, and number of car's cylinder.

In this post, I am going to show variable "year" and type of "Drive" of each car plus what we have.

first, I have to change the above code as below:

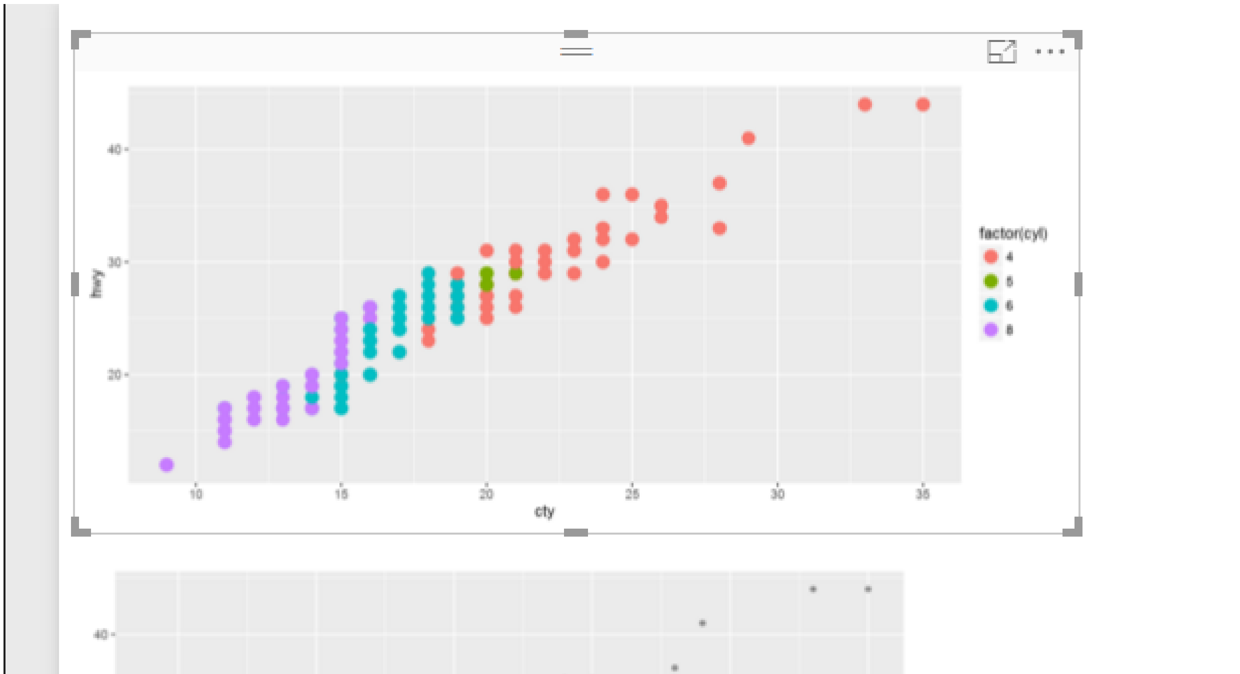
```
t<-ggplot(dataset, aes(x=cty, y=hwy,colour = factor(cyl))) + geom_point(size=4)
```

Before that, I want to do some changes in the chart. I changed the "aes" function argument. I replaced the "Size" argument with "Colour". that means, I want to differentiate car's cylinder values not just by Cycle size, but I am going to show them by allocating them different colours. so I changes the "aes" function as above.

so by changing the codes as below

```
library(ggplot2)

t<-ggplot(dataset, aes(x=cty, y=hwy,colour = factor(cyl))) + geom_point(size=4)
```

we will have below chart:



R script editor

```
# Remove duplicated rows
# dataset <- unique(dataset)
library(ggplot2)

t<-ggplot(dataset, aes(x=cty, y=hwy,colour = factor(cyl))) + geom_point(size=4)

t
```

Now I want to add other layer to this chart. By adding year and car drive option to the chart. To do that first choose year and drv from data field in power BI. As I have mentioned before, now the dataset variable will hold data about speed in city, speed in highway, number of cylinder, years of cars and type of drive.

I am going to use another function in the ggplot packages name “facet_grid” that helps me to show the different facet in my scatter chart. In this function, year and drv (driver) will be shown against each other.

```
facet_grid(year ~ drv)
```

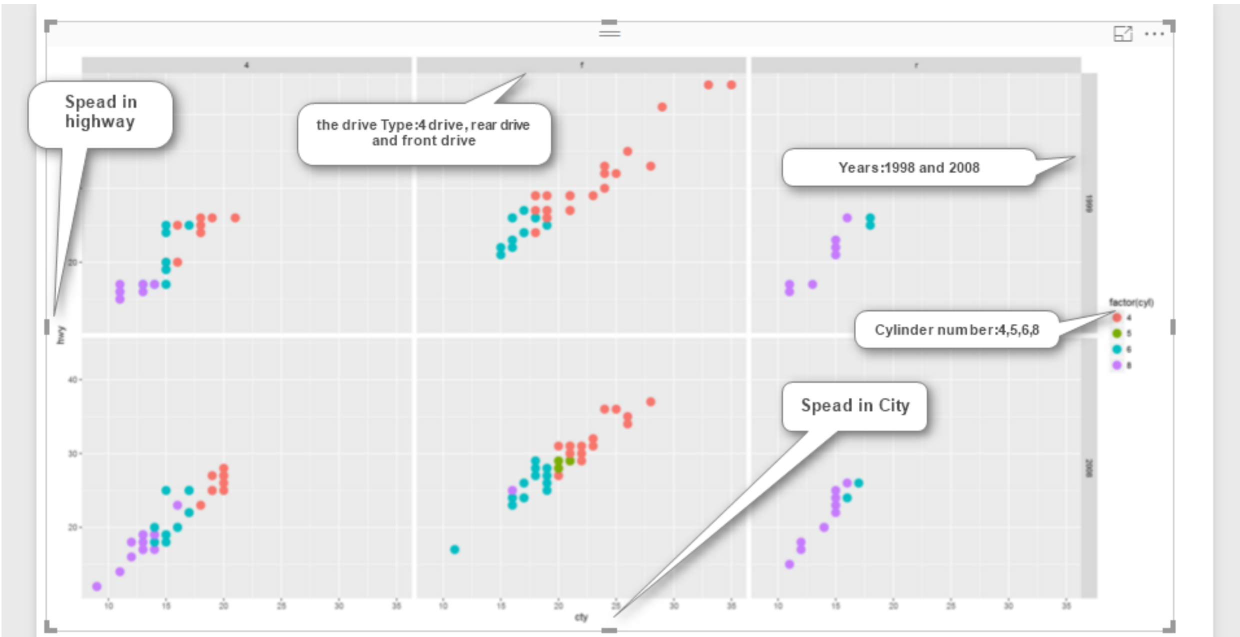
To do that, I am going to use above code to add another layer to my previous chart.

```
t<-ggplot(dataset, aes(x=cty, y=hwy,colour = factor(cyl))) + geom_point(size=4)

t<-t + facet_grid(year ~ drv)
```

t

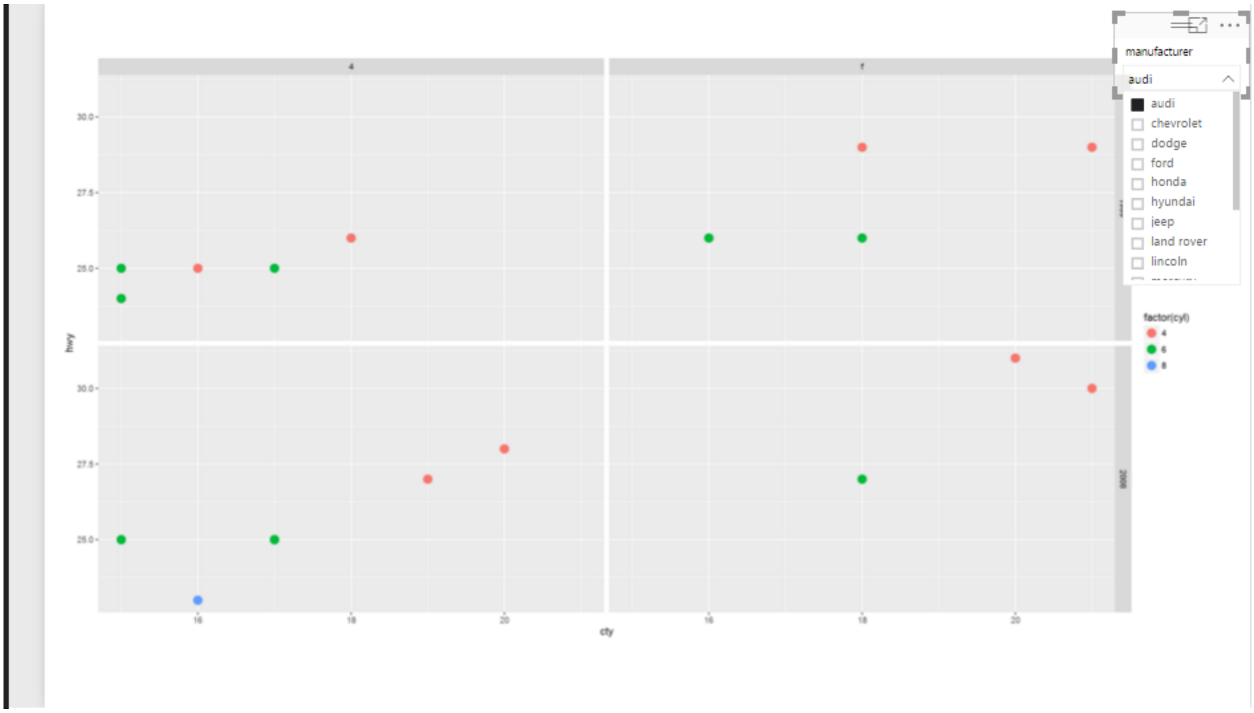
so I add another layer to variable “t” as above.

now the visualization will be look like as below: as you can see the car’s speed in the highway and city in y and x axis. Also, we have cylinder as colour and drive and year as facet.

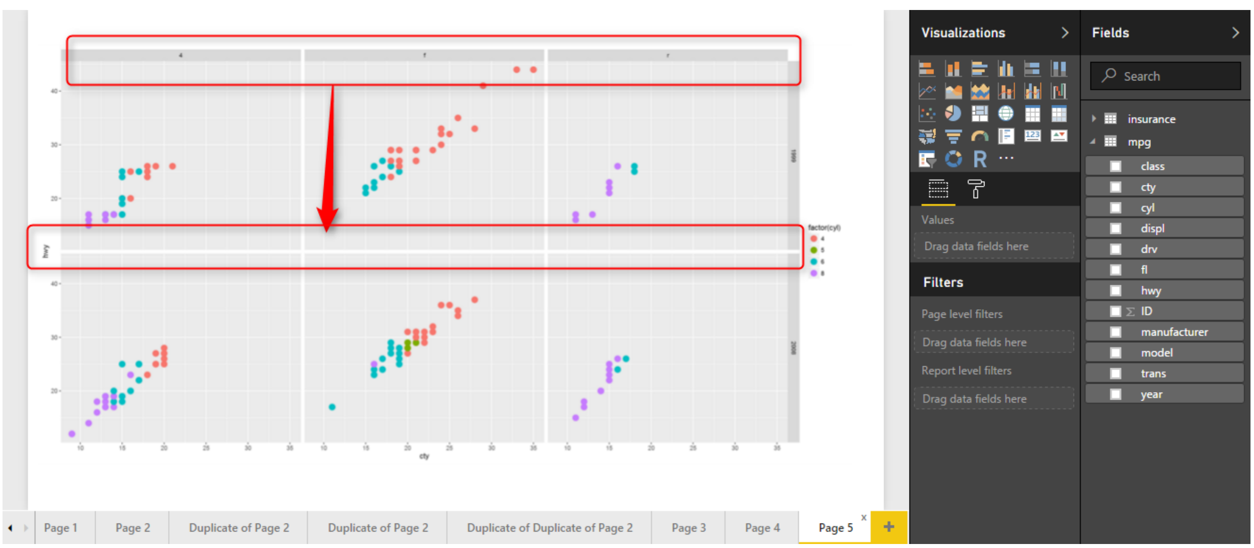


Now imagine, I am going to add a slicer to filter “car brands”, and also to see these five variables against each other in one chart.

So, we will have below chart:



I am going to have some more fun with chart, I need to show the drive type in all regions not just the three above (see below image)



In this case I am able to use the another facet function instead of facet_grid(year ~ drv) I am going to use other function name facet_wrap(year~ drv) which help me to do that.

Hence, I change the codes as below:

t<-t + facet_wrap(year~ drv)

Moreover, I want to show the car's cylinder type not just by different colour, but also with same colour just different shading. so I will replace the argument inside the aes function as below

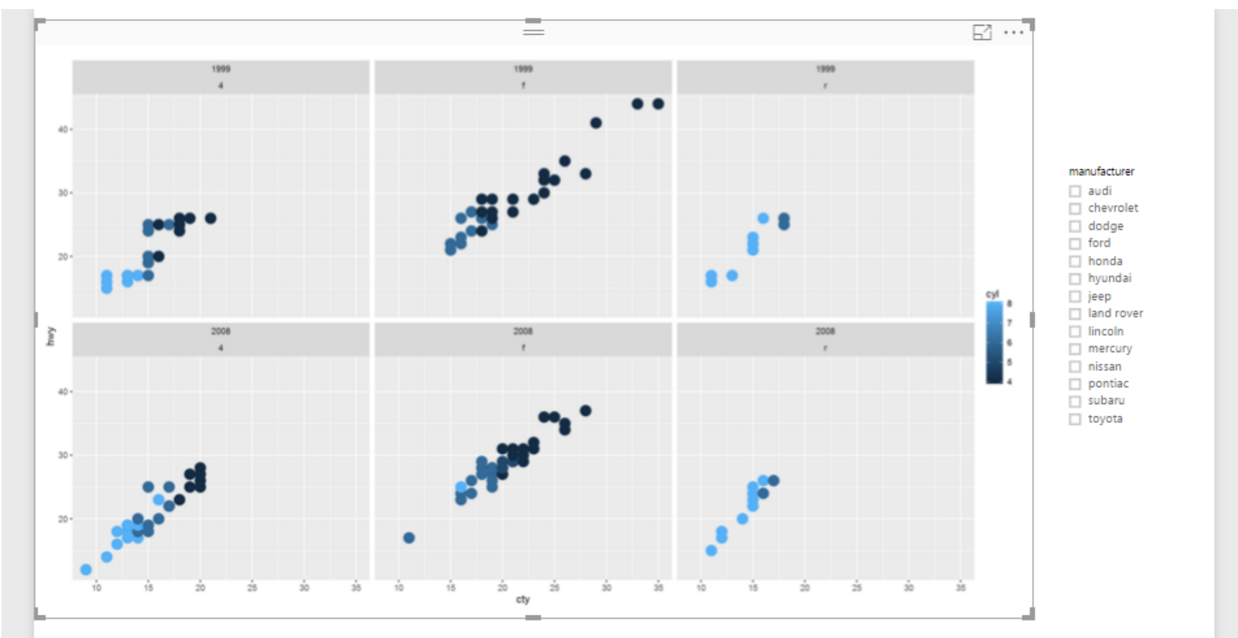
```
aes(x=cty, y=hwy,color=cyl))
```

instead of aes(x=cty, y=hwy,colour = factor(cyl))

so finally the code will be look like as below

```
library(ggplot2)

t<-ggplot(dataset, aes(x=cty, y=hwy,color=cyl)) + geom_point(size=5)

t<-t + facet_wrap(year~ drv)
t
```

Finally,I will have below picture, as you can see in the below image, we have title for each group as for the top left 1999 and 4drive, for top and middle we have 1999, and r drive and so forth.



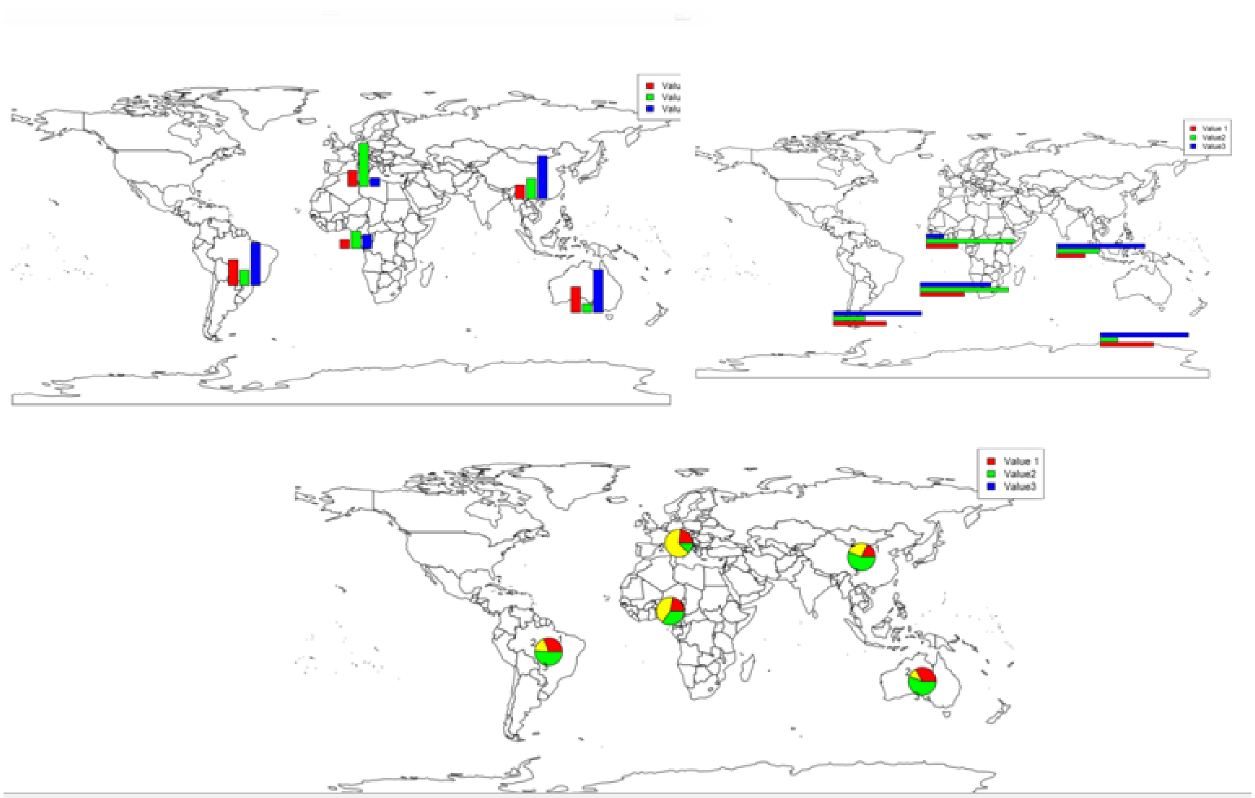
In future posts, I will show some other visuals that we have in ggplot2 package, which help us to have more fun in power BI.

Enter Your Email to download the file (required)

[Download]

4-Have More Charts by writing R codes inside Power BI: Part 3

Published Date : April 10, 2017



In the previous parts ([Part 1](#) and [Part 2](#)) , I have shown how to draw a chart in the power BI ([Part 1](#)) visualization. Also, in [Part 2](#) I have shown how to present 5 different variables in just one single chart. In this post, I will show how to show some sub plots in a map chart. Showing pie chart already is possible in power BI map. In this post I am going to show how to show bar chart, pie chart and so other chart type in a map.

For this post, I have used the information and codes available in [1] and [2], which was so helpful!.

This may happen that we want to have some subplots in a map, in R you are able to show different types of charts in a map as a subplot.

To start, first setup your power BI as [part 1](). We need below library first to be installed in R software. Then you should use them in Power BI by referring to them as below.

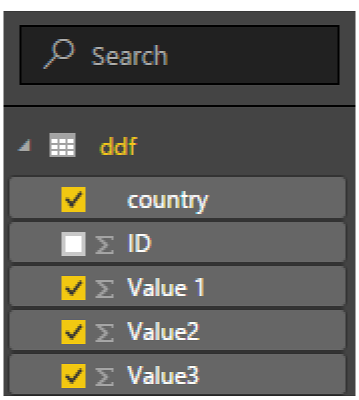
```
library(rworldmap)
library(sp)
library(methods)
library(TeachingDemos)
require(sp)
```

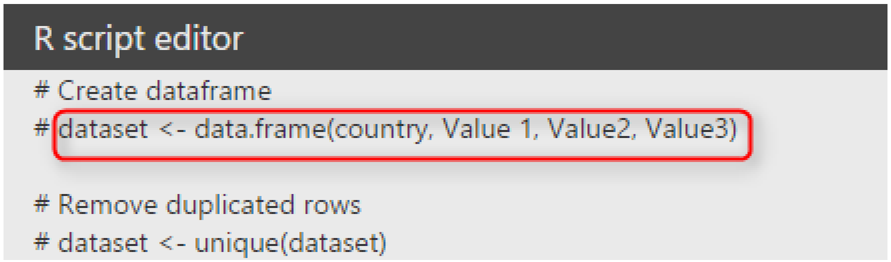
Next we need data to show on the map. I have a dataset about different countries as below :

ID	country	Value 1	Value2	Value3
1	Nigeria	5	10	8
2	China	8	12	25
3	Brazil	15	9	25
4	Italy	9	25	5
5	Australia	15	5	25

I have 3 different random columns for each country, as you can see in above picture (I just pick that data from reference number[1]). I am going to create a chart to show these 3 values (value1, value2, and value3) in map for each country.

In Power BI visualization, first select the dataset (country, value1, value2, and value3). This data will store in variable "Dataset" in R script editor as you can see in below image.





I put the "Dataset" content into new variable name "ddf" (see below)

```
ddf =dataset
```

The second step is about finding the latitude and longitude of each country using function "joincountrydata2map". this function gets the dataset "ddf" in our case as first argument, then based on the name of the country "joincode="NAME" and in ddf dataset "country column" (third argument) will find the country location specification (lat and lon)for showing in the map. We store the result of the function in the variable "sPDF"

```
sPDF <- joinCountryData2Map(ddf    , joinCode = "NAME"    , nameJoinColumn = "country"    , verbose = TRUE)
```

Hence, I am going to draw an empty map first by below code

```
plot(getMap())
```

Now I have to merg the data to get the location information from “sPDF” into “ddf”. To do that I am going to use” merge” function. As you can see in below code, first argument is our first dataset “ddf” and the second one is the data on Lat and Lon of location (sPDF). the third and forth columns show the main variables for joining these two dataset as “ddf” (x) is “country” and in the second one “sPDF” is “Admin”. the result will be stored in “df” dataset

```
df <- merge(x=ddf, y=sPDF@data[sPDF@data$ADMIN, c("ADMIN", "LON", "LAT")], by.x="country", by.y="ADMIN", all.x=TRUE)
```

Also, we need the “TeachingDemos” library as well.

```
require(TeachingDemos)
```

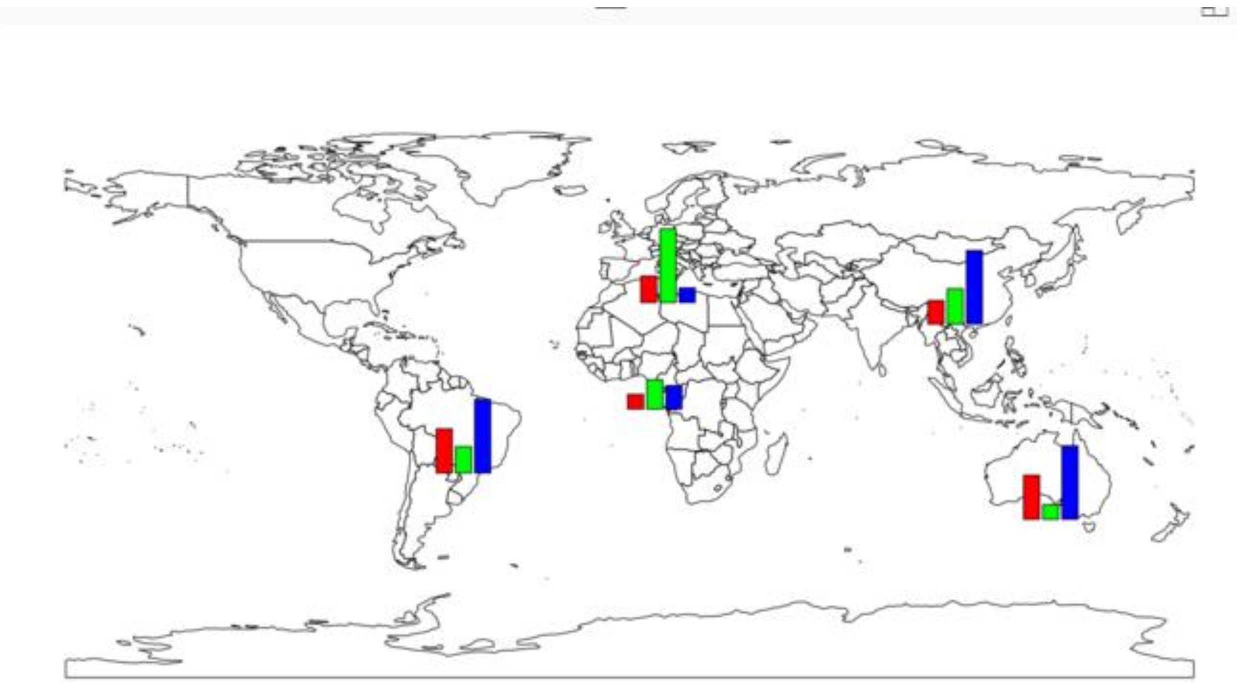
I am going to draw a simple bar chart that show the value1, Value2, and Value 3 for each country. So I need a loop structure to draw barchart for each country as below. I wrote “for(I in 1:nrwo(df)) that means draw barchart for all countries we have in “df” then I called a subplot as main function that inside I defined the barplot().

```
for (i in 1:nrow(df))
  subplot(barplot(height=as.numeric(as.character(unlist(df[i, 2:4], use.names=F)
                                                )
                              ),
                  axes=F,
                col=rainbow(3),
```

```
            ylim=range(df[,2:4])
          ),
    x=df[i, 'LON'], y=df[i, 'LAT'], size=c(.6, .6)
    )
```

barplot() get values for height of each bar chart as a number (as.number). also I fetch the data related to “df” dataset from row number “i”, for columns from 2 to 4 (value 1 to value 3). To stet the colouring of the bar chart we use (col=rainbow(3)). “Y” axis should range values from “df” dataset for dataset df[,2:4]. the “x” axis get the latitude and longitude. The size of the bar chart can be changed by function “size=c(,)”.

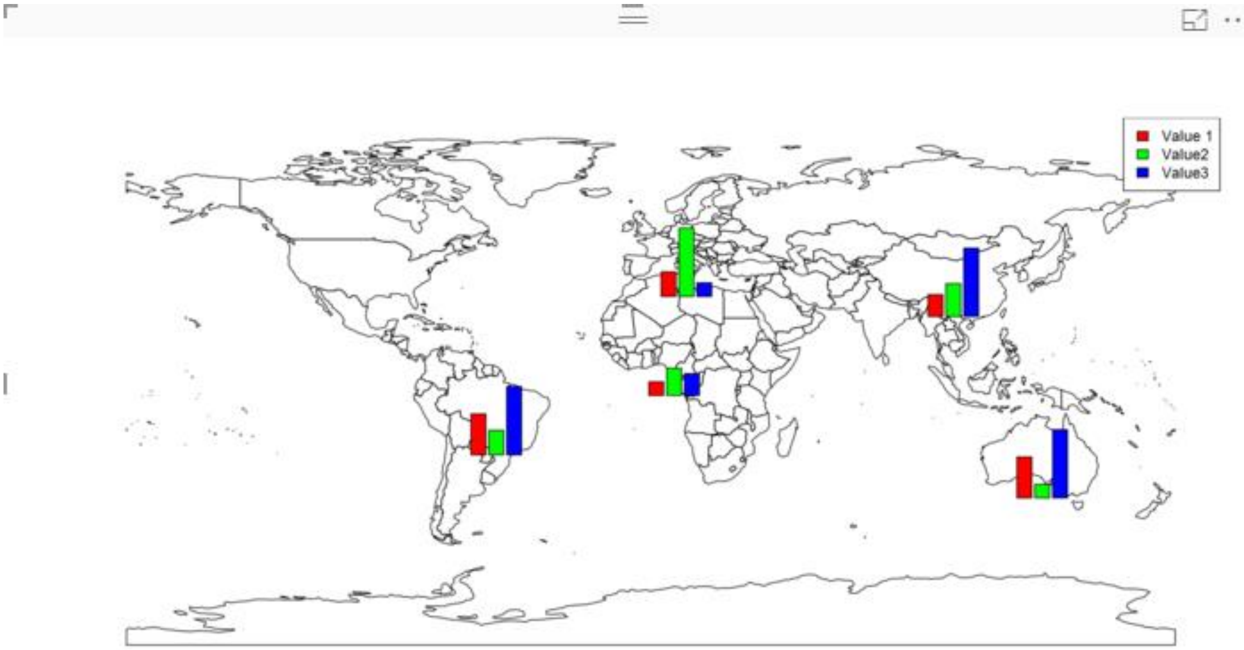
then we have below picture:



To have better map, we need a legend on the side of the map. To do that I am using a function named “legend” that the first argument is the name of the legend as “top right”. the legend values comes from “df” dataset. we using the same colouring we have for bar chart.

```
legend("topright", legend=names(df[, 2:4]), fill=rainbow(3))
```

so at the end we have below chart



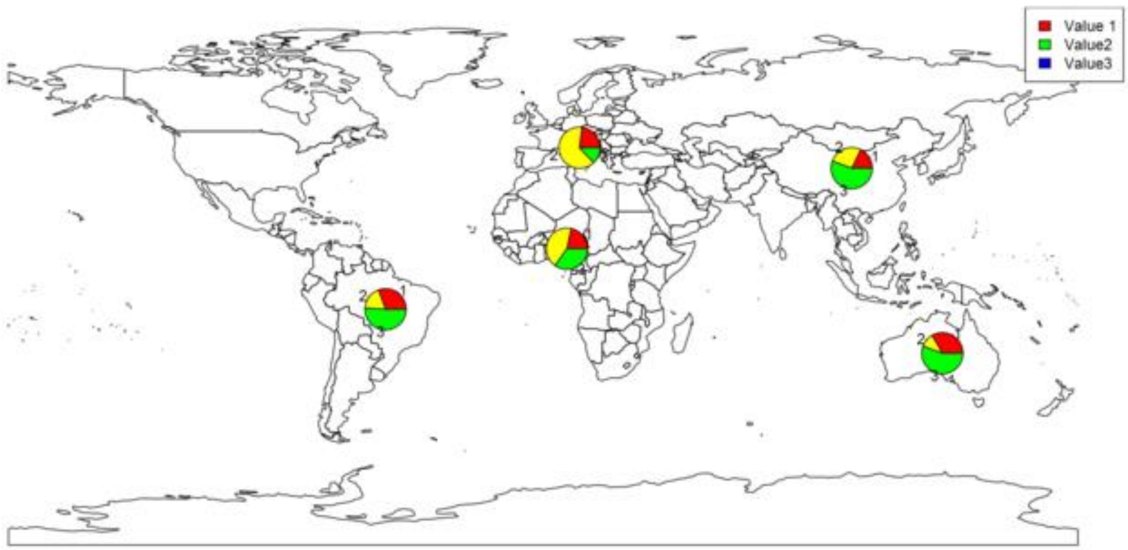
Now imagine that we want to have another type of chart in map as pie char or horizontal bar chart.

To do this, I need just to change the chart in Subplot as below

```
subplot(pie(as.numeric(as.character(unlist(df[i, 2:4], use.names=F))),
```

Just replace the bar chart with pie chart (use above codes).

so we will have below chart

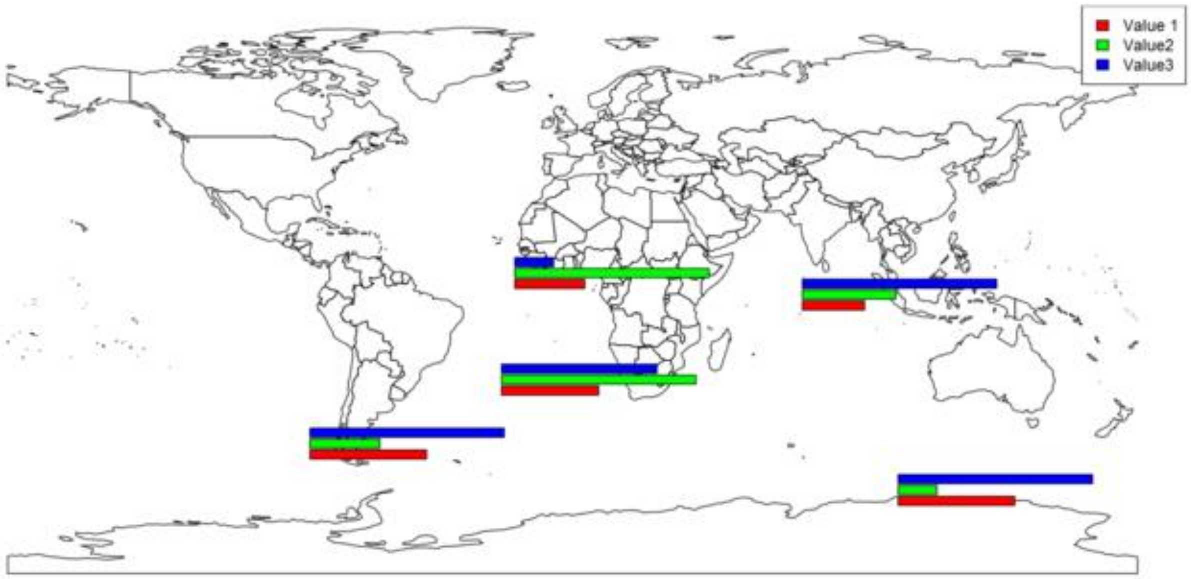


Or if we want to have a horizontal bar chart we need to just change our code as below

subplot(barplot(height=as.numeric(as.character(unlist(df[i, 2:4], use.names=F))), horiz = TRUE,

as"horiz=true"

and we have below chart



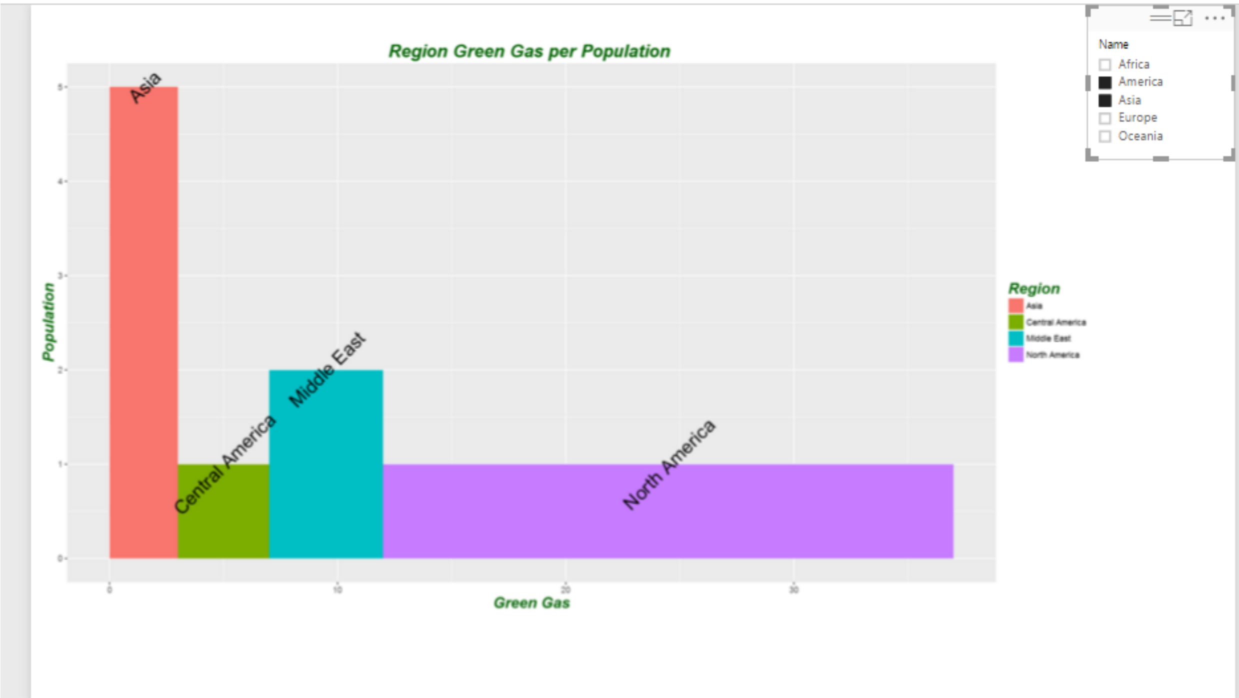
there are possibility to add other types of charts in the map as well!

[1] http://stackoverflow.com/questions/24231569/bars-to-be-plotted-over-map

[2]https://www.stat.auckland.ac.nz/~ihaka/120/Lectures/lecture16.pdf

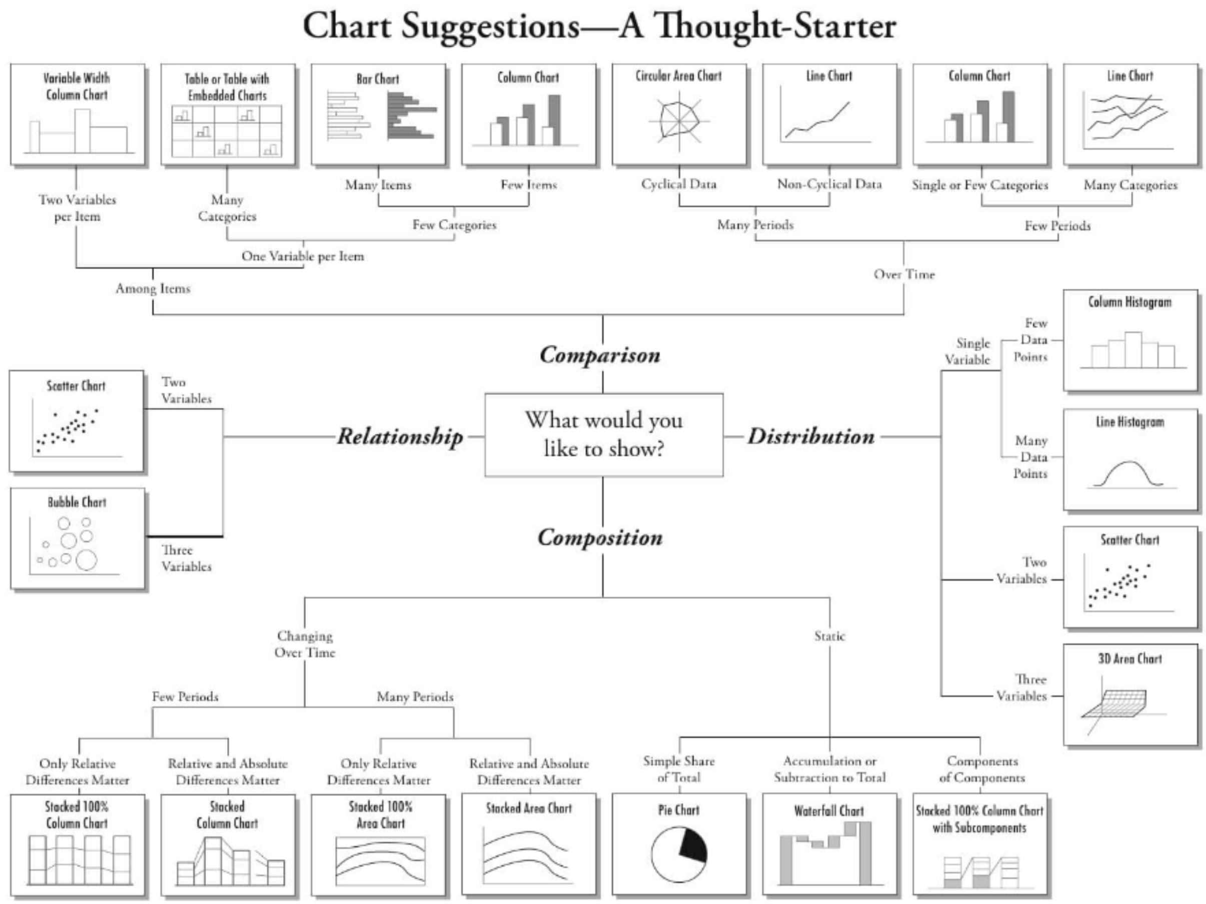
5-Variable Width Column Chart, writing R codes inside Power BI: Part 4

Published Date : April 21, 2017



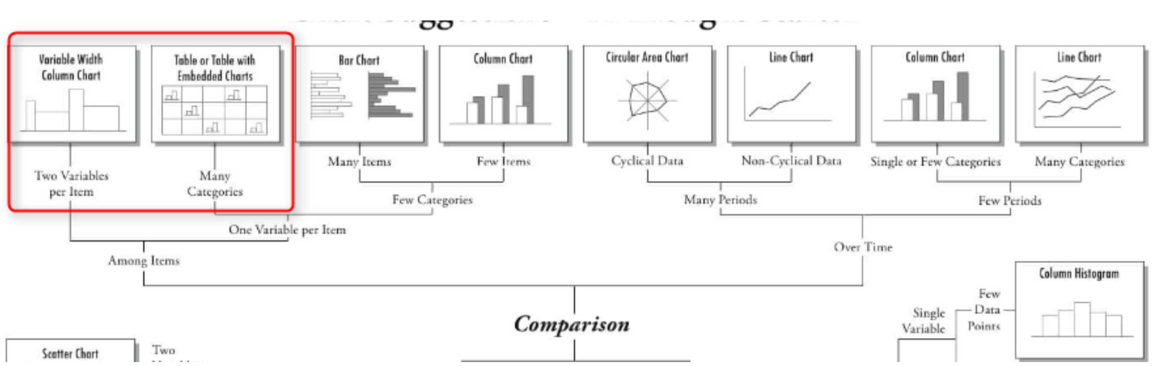
In the part 1, I have explained how to use R visualization inside Power BI. In the second part the process of visualization of five dimension in a single chart has been presented in Part 2, and finally in the part 3 the map visualization with embedded chart has been presented.

In current post and next ones I am going to show how you can do data comparison, variable relationship, composition and distribution in Power BI.



Comparison is one of the main reasons of data visualization, about comparing data to see the changes and find out the difference between values. This comparison is mainly about comparing data **Over time** or by other **Items.**

For comparison purpose, most of the charts are available in Power BI Visualization, just two of them are not :**Variable Width Column Chart** and **Table with Embedded Chart**.



In this post I will talk about how to draw a "**Variable Width Column Chart**" inside Power BI using R scripts.

In the next Post I will show how to draw a **Table with Embedded Chart"** in the next post.
Variable Width Column Chart

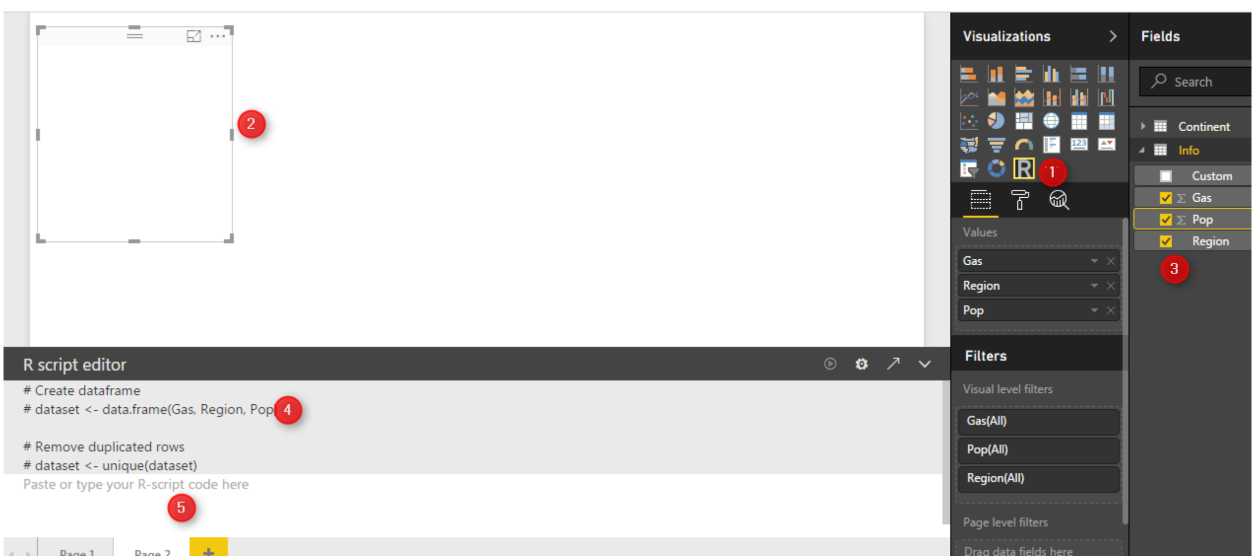
you will be able to have a column chart with different width inside Power BI by writing R codes inside R scripts editor.

to start I have some dummy data about amount of green gas and population of each region .

Region	Gas	Pop	Custom
North America	25	1	1
Oceania	19	3	4
Europe	11	2	5
Africa	6	2	3
Middle East	5	2	2
Central America	4	1	1
Asia	3	5	2

in the above table, "Pop" stands for "Population" "Gas" stands for "Green Gas" amount.

To start, first click on the "R" visualization icon in the right side of the page in power bi desktop. Then, you will see a blank visualization frame in the middle of the report. Following, click on the required fields at the right side of the report to choose them for showing in the report. Click on the " Gas", "Pop" and "Region" fields.



at the bottom of the page, you will see R scripts editor that allocate these three fields to a variable named “dataset”(number 4).

We define a new variable name “df” which will store a data frame (table) that contains information about region, population and gas amount.

df<-data.frame(Region=dataset$Region,Population= dataset$Pop,width=dataset$Gas) as you can see in above code, I have used “$” to access the fields stored in “dataset” variable.

now, I am going to identify the width size of the each rectangle in my chart, using below codes

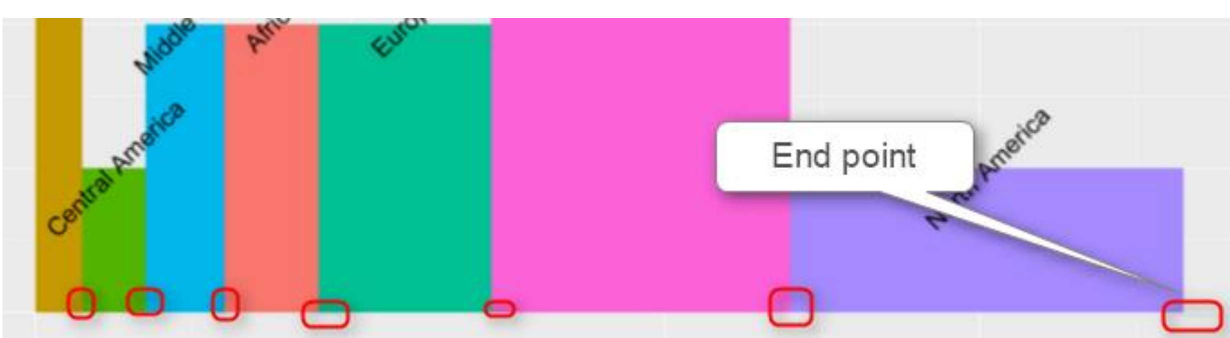
df$w <- cumsum(df$width) #cumulative sums.
df$wm <- df$w – df$width
df$GreenGas<- with(df, wm + (w – wm)/2)

the first step is to identify the start point and end point of each rectangle.

“cumsum” a cumulative sum function that calculate the width of each region in the graph from 0 to its width. This calculation give us the end point (width) of each bar in our column width bar chart (see below chart and table).

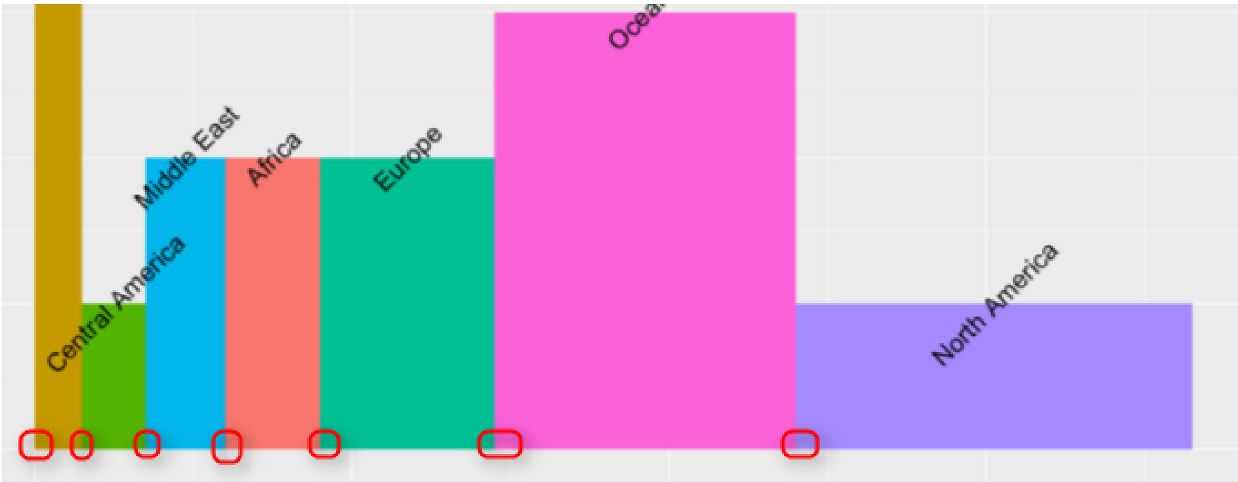
so I will have below numbers

A	B	C	D
Region	Gas	Pop	width from 0
North Am	25	1	25
Oceania	19	[illegible]	=B2+B3
Europe	11	2	30
Africa	6	2	17
Middle Ea	5	2	11
Central Ar	4	1	9
Asia	3	5	7



so we store this amount in df$w variable.

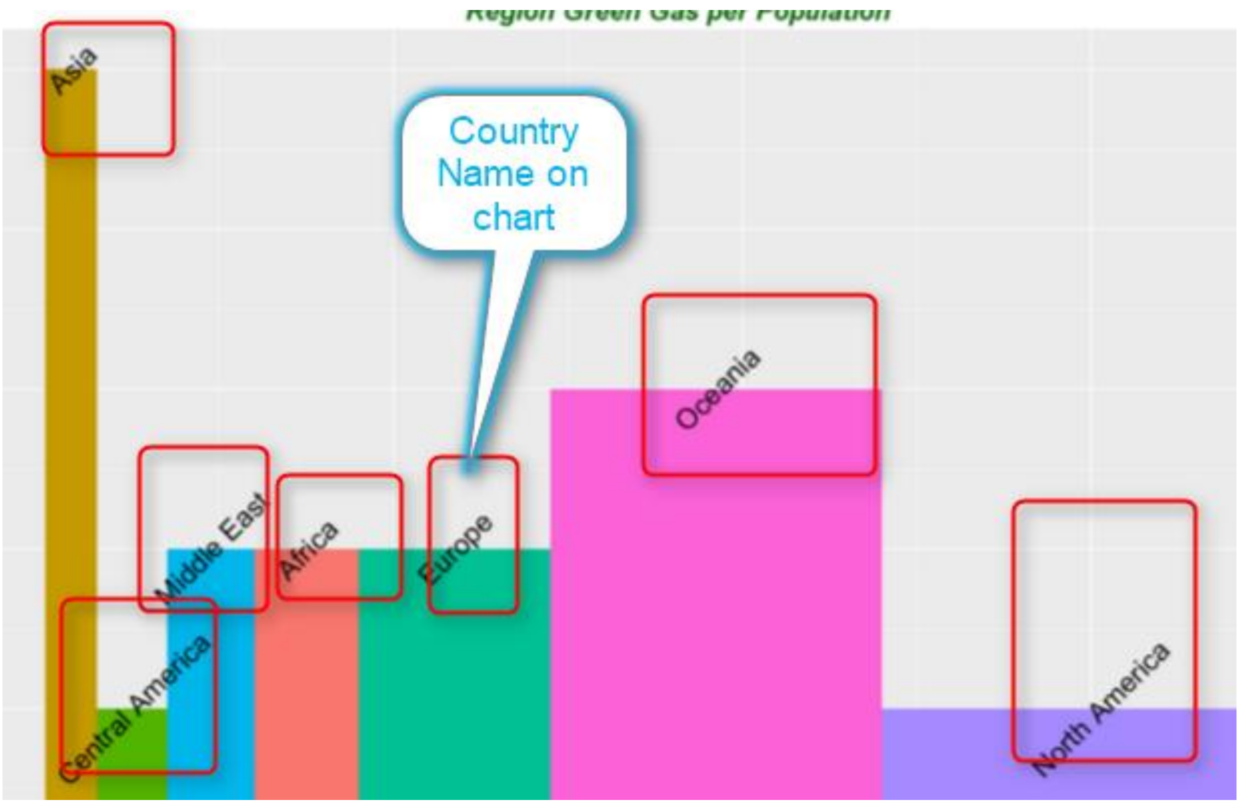
next, we have to calculate the start point of each bar chart (see below chart) using : df$wm <- df$w – df$width



Moreover, we also calculate the start point of width for each bar as below

Region	Gas	Pop	End poit of bar chart	start of bar chart
North Am	25	1	25	0
Oceania	19	3	44	25
Europe	11	2	30	19
Africa	6	2	17	11
Middle Ea	5	2	11	6
Central Ar	4	1	9	5
Asia	3	5	7	4

Another location that we should specify before hand is the rectangle lable. I prefer to put them in middle of the bar chart.



We want the label text to be located in the middle of each bar, so we calculate the middle of width (average) of each bar as above and put the value in df$GreenGas. This variable holds the location of each labels.

So we have all data ready to plot the chart. we need "ggplot2" library for this purpose

library(ggplot2)

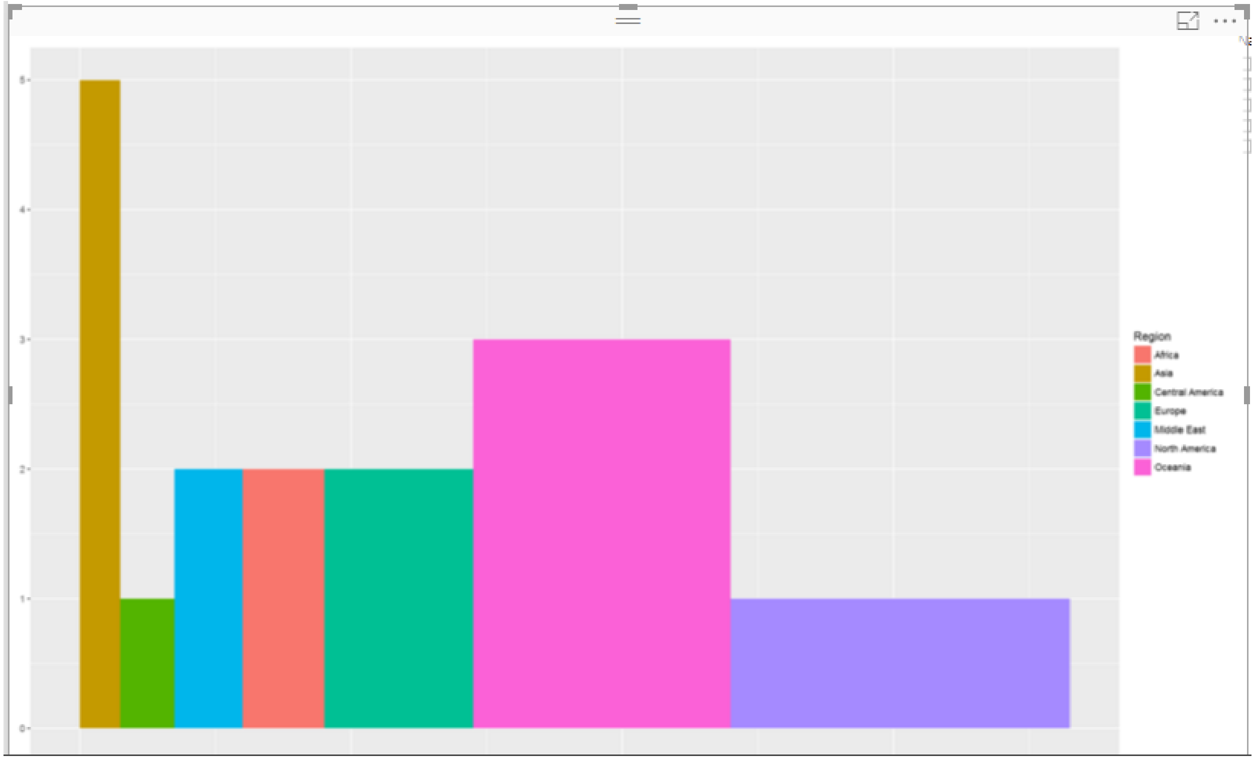
we call function "ggplot" and we pass the first argument to it as our dataset. the result of the function will be put in variable "P"
p <- ggplot(df, aes(ymin = 0))

Next, we call another function as below, this function draws a rectangle for us. This get the start point of each rectangle (bar) as "wm" and the end point of them "w" (we already calculated them in above and the third argument is the range of the "Population" number. The last argument is that the colouring each rectangle based on their related region.

p1 <- p + geom_rect(aes(xmin = wm, xmax = w, ymax = Population, fill = Region))

p1

If we run the above codes we will have below chart



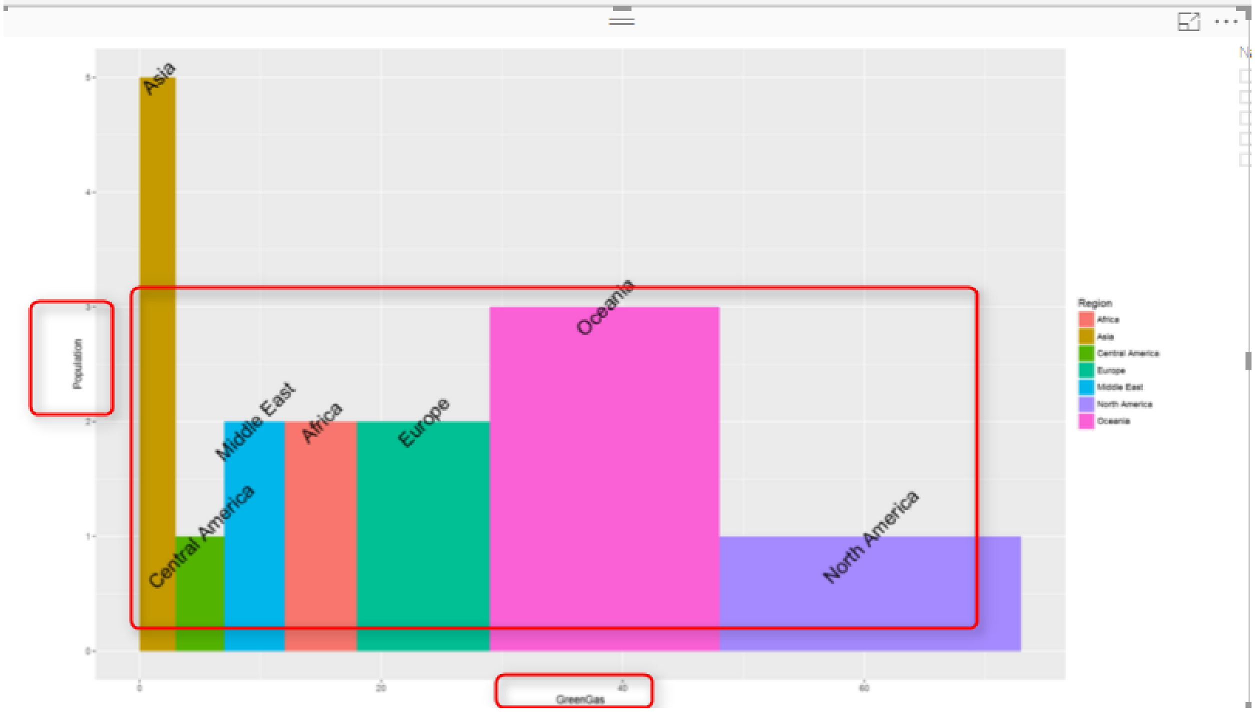
Our chart does not have any label. So we need to add other layer to p1 as below .

p2<-p1 + geom_text(aes(x = GreenGas, y = Population, label = Region),size=7,angle = 45)

p2

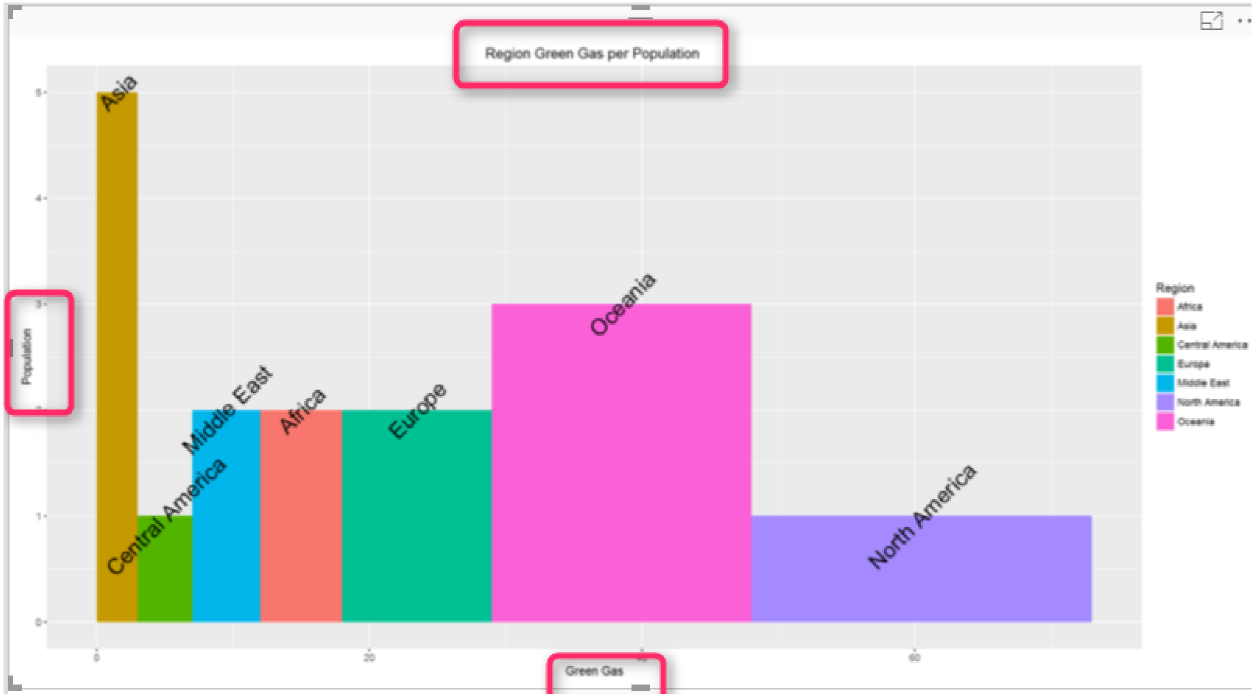
geom_text function provides the title for x and y axis and also for the each rectangle using "aes" function aes(x = GreenGas, y = Population, label = Region).

However you are able to change the size of the text label for each bar (rectangle) by adding a parameter " size". for instance I can change it to 10. or you can specify the angel of the label (rotate) it by another parameter as "angel" in this chart I put it to be "45" degree.



I am still not happy with this chart, I want to change the title of x and y axis and also add a title to chart. To do this, I add another layer to what I have. I call a function name "labs" which able me to specify the chart title, x and y axis name as below

p3<-p2+labs(title = "Region Green Gas per Population ", x = "Green Gas", y = "Population")
p3



the problem with this chart is that the title and text in x or y axis are not very clear. As a result I define a theme for each of them. To define a theme for chart title and the text I am going to use a function name " element_text".

This function enables me to define some theme for my chart as below:

blue.bold.italic.10.text <- element_text(face = "bold.italic", color = "dark green", size = 10)

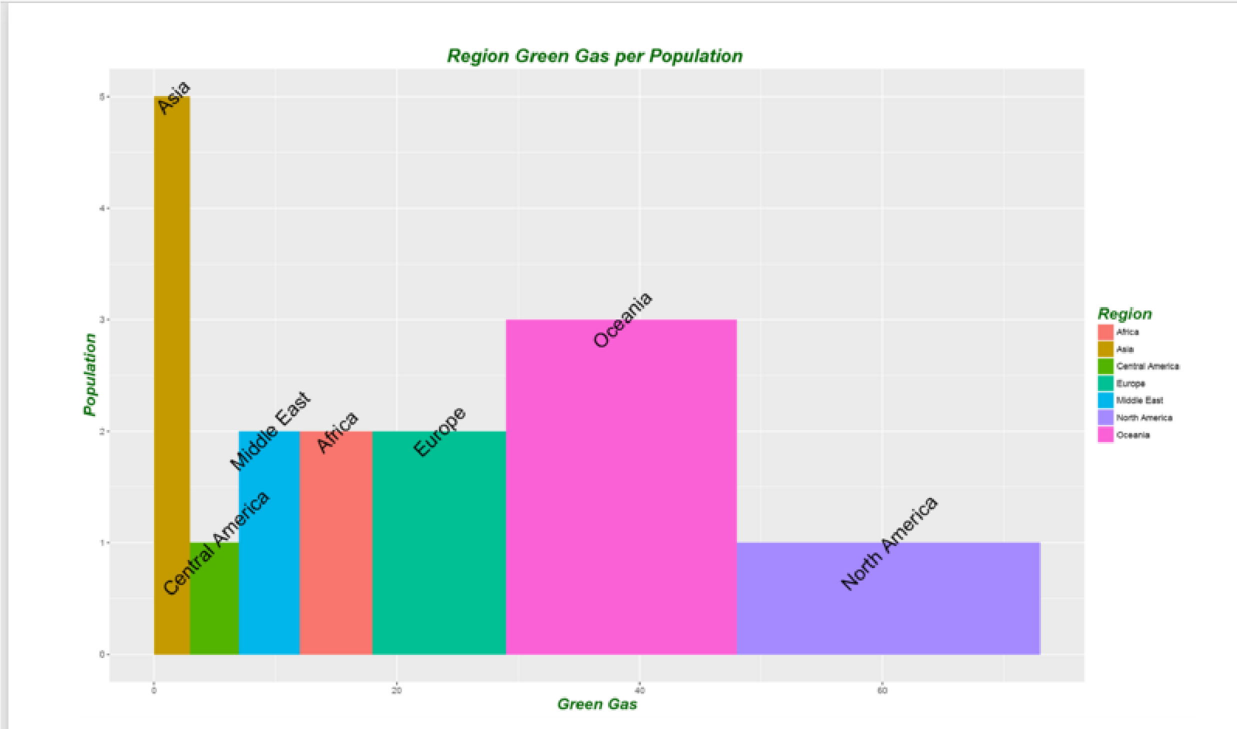
So I want the text (title chart , y and x text) be bold and italic. also the color be "dark green" and the text size be 10.

I have defined this theme. Now, I have to allocate this theme to title of chart and axis text.

To do this, I call another function (Theme) and add this as another layer. Theme function I specify the axis title should follow the rules and theme I created in above code "blue.bold.italic.10.text" . the second argument in "theme" function allocated this created theme to the chart title as below

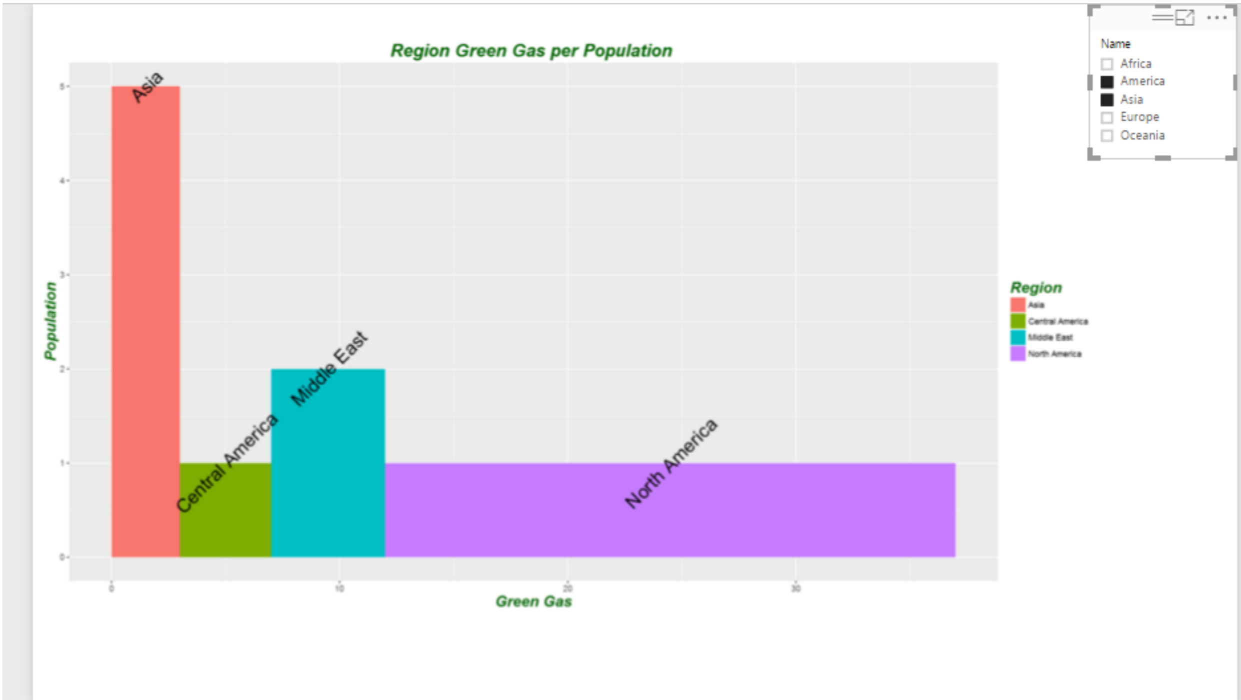
p4<-p3+theme(axis.title = blue.bold.italic.10.text, title =blue.bold.italic.10.text)

p4



There are other factor that you are able to change by adding more layers and function to make your chart look better.

Now I want to analyse the green gas per each continent. So I am going to add another slicer to my report to check it. so for example I just want to compare these numbers for "America" and "Asia" continent in my chart. the result is shown in below chart.



The overall code would be look like below :

```
df<-data.frame(Region=dataset$Region,Population= dataset$Pop,width=dataset$Gas)
df$w <- cumsum(df$width) #cumulative sums.
df$wm <- df$w – df$width
df$GreenGas<- with(df, wm + (w – wm)/2)
library(ggplot2)
p <- ggplot(df, aes(ymin = 0))
p1 <- p + geom_rect(aes(xmin = wm, xmax = w, ymax = Population, fill = Region))
p2<-p1 + geom_text(aes(x = GreenGas, y = Population, label = Region),size=7,angle = 45)
p3<-p2+labs(title = “Region Green Gas per Population ”, x = “Green Gas”, y = “Population”)
blue.bold.italic.10.text <- element_text(face = “bold.italic”, color = “dark green”, size = 16)
p4<-p3+theme(axis.title = blue.bold.italic.10.text, title =blue.bold.italic.10.text)
p4
```

I will post the Power Bi file to download soon here.

Enter Your Email to download the file (required)

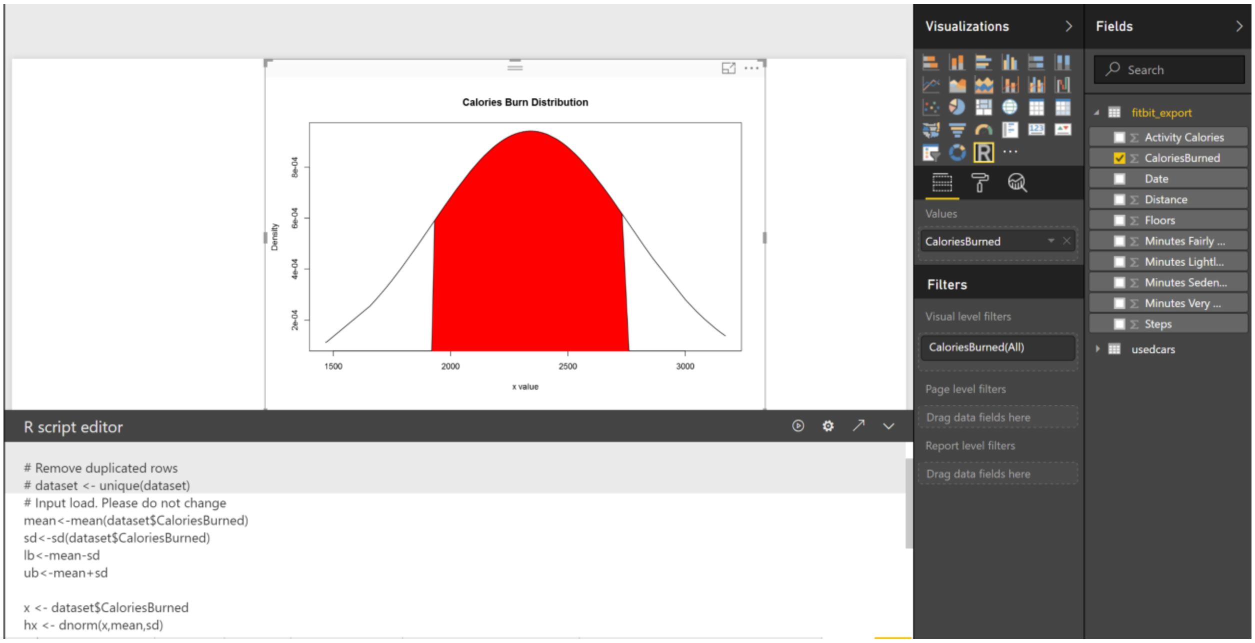
[1]https://i1.wp.com/www.tatvic.com/blog/wp-content/uploads/2016/12/Pic_2.png

[2]http://ggplot2.tidyverse.org/reference/geom_text.html

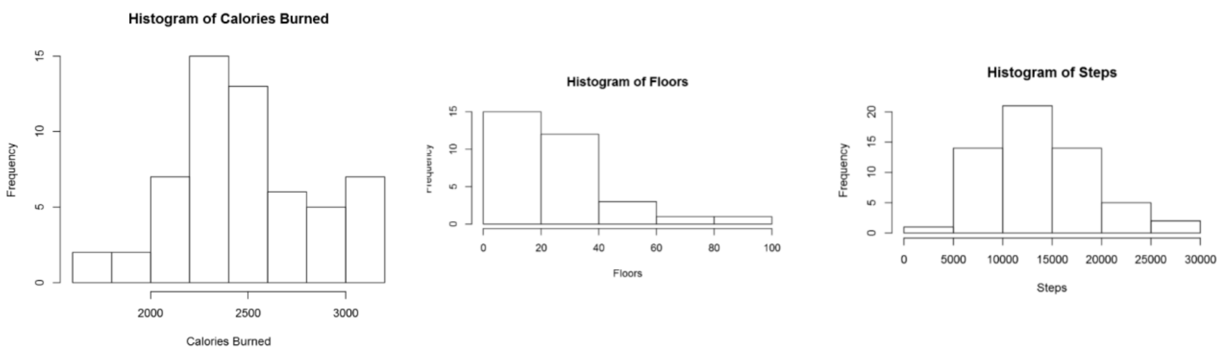
[3]http://stackoverflow.com/questions/14591167/variable-width-bar-plot

6-Visualizing Data Distribution in Power BI – Histogram and Norm Curve -Part 5

Published Date : May 29, 2017



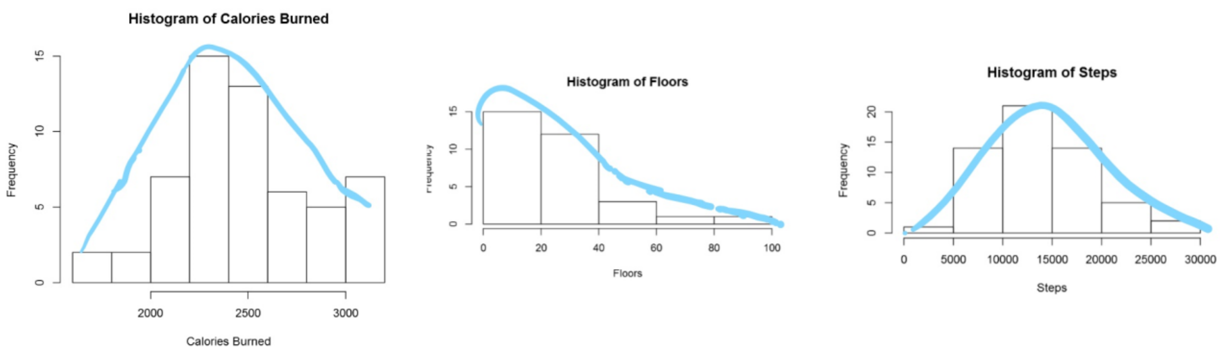
In the Part 1 I have explained some of the main statistics measure such as **Minimum, Maximum, Median, Mean, First Quantile, and Third Quantile.** Also, I have shown how to draw them in Power BI, using R codes. (we have Boxplot as a custom visual in power BI see :https://powerbi.microsoft.com/en-us/blog/visual-awesomeness-unlocked-box-and-whisker-plots/). However, to see the data distribution another way is to draw a histogram or normal curve. The spread of the numeric variable can be check by the histogram chart. Histogram uses any number of bins of an identical width. Below picture shows the data distribution for my Fitbit data (Floors, Calories Burned, and Steps).



to create a histogram chart, I wrote blew R code.

hist(dataset$Floors, main = "Histogram of Floors", xlab = "Floors")

In the above picture, the first chart shows the data distribution for my calories burn during three months. As you can see, most of the time I burned around 2200 to 2500 calories, also less than 5 times I burned calories less than 2000 calories. If you look at the histogram charts you will see each of them has different shape. As you can see the number of floors increases further to the right. While calories burn and number of floors tend to be evenly divided on both sides of the middle. This behaviour is called **Skew.** This help us to find the data distribution, as you can see the data distribution has a Bell Shape, which we call it **Normal Distribution.** Most of the world data follow the normal distribution trend. Data distribution can be identified by two parameters: Centre and Spread.



The centre of the data is measured by **Mean** value, which is the data average. Spread of data can be measured by **Standard deviation.**

So What is **Standard Deviation!** Standard deviation can be calculated from **Variance.** Variance is :"the average of the squared differences between each value and the mean value"[1] in other word to calculate the variance, for each point of data we call (Xi) we should find its distance from mean value (µ). to calculate distance we follow the formula as :

$$\mathrm{Var}(X) = \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

the distance between each element can be calculate by (Xi-µ)^2 (Number 1 in above Formula). Then for each point we have to calculate this distance and find the average. So we have summation of (Xi-µ)^2 for all the points and then divided by number of the points (n). **σ^2** is variance of data. Variance or Var(X) is the distance of all point from the mean value. The **Standard Deviation** is sqrt of Var(X). that is **σ.** So if data is so distributed and has more distance from Mean value then we have bigger Standard Deviation.

To draw normal curve in Power BI I wrote the blow codes. First, I calculate the Average and Standard Deviation as

mean<-mean(dataset$CaloriesBurned)

sd<-sd(dataset$CaloriesBurned)

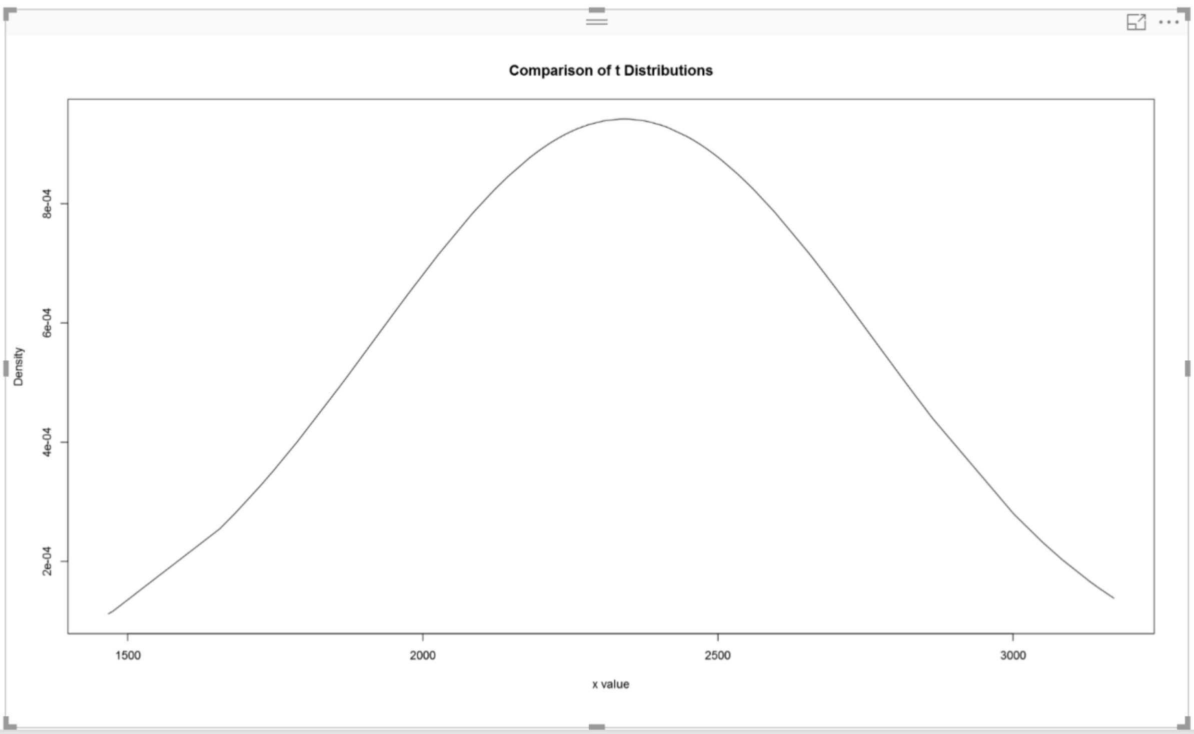
Then I used the "**dnorm"** function to create a norm curve as below:

y<-dnorm(dataset$CaloriesBurned,meanval,sdval)

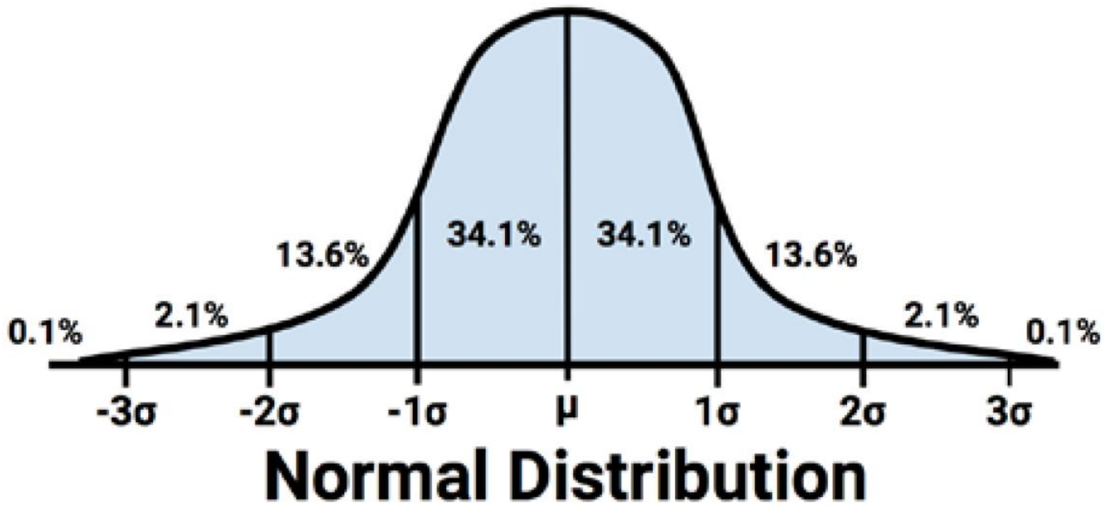
Then I draw anorm curve using Plot function :

plot(dataset$CaloriesBurned, y, xlab="x value", ylab="Density", type="l",main="Comparison of t Distributions")

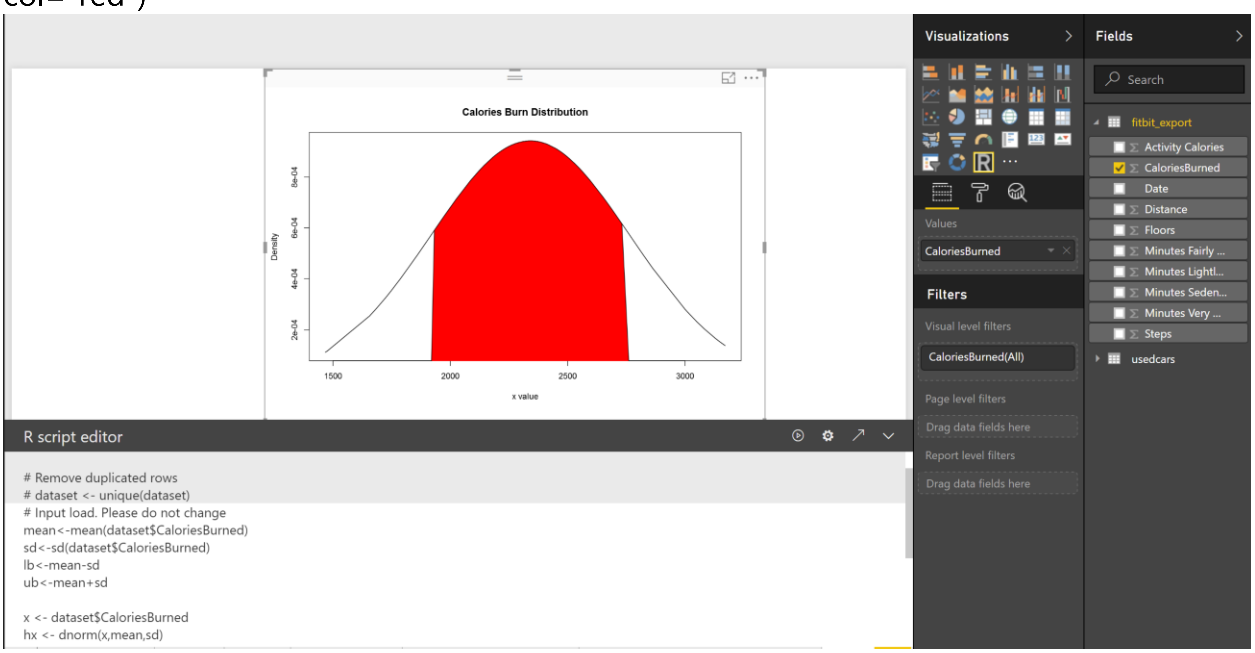
Then the following picture will be shown as below



According to [1], the 68-95-99.7 rule states that 68 percent of the values in a normal distribution fall within one sd of the mean, while 95 percent and 99.7 percent of the values fall within two and three standard deviations, respectively[1].



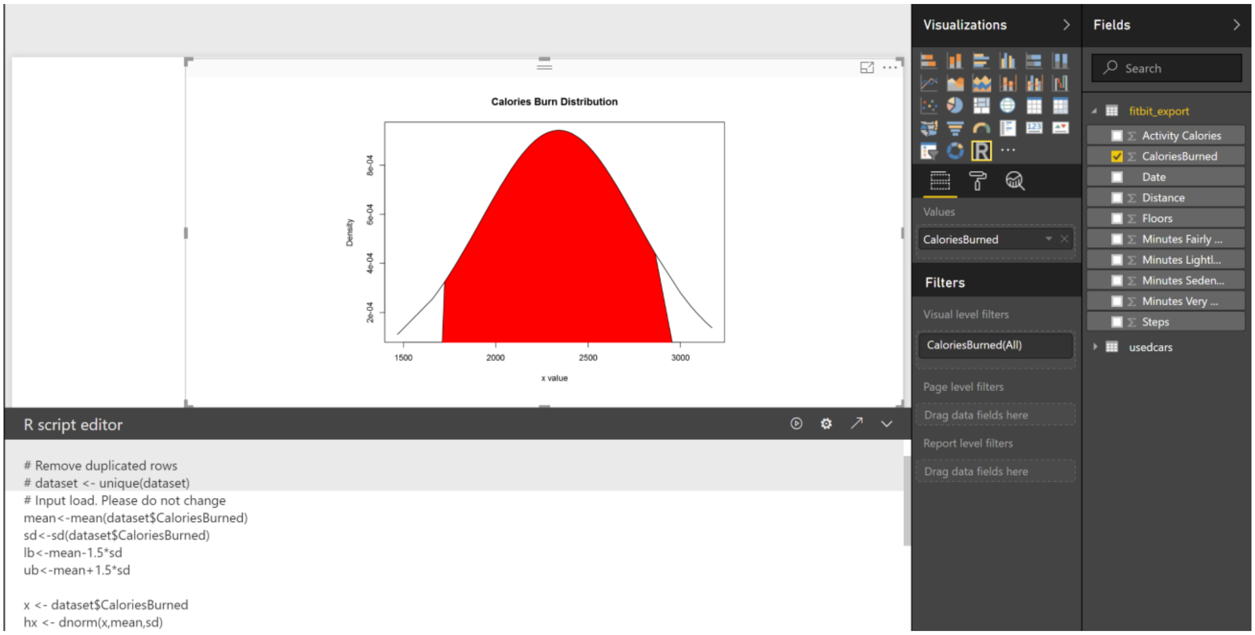
As you can see 68% of data is located between -sd and +sd. the 95% of data is located between -2sd and +2sd. Then, 99% of data has been located between -3sd and +3sd. to draw and identify the 68% of data I add other calculation to R code in Visualization as below: first I set a range of value for range (average-standard deviation, average +standard deviation) lower bound (lb) is average- standard deviation lb<-mean-sd for upper bound (ub) we calculate it as below ub<-mean+sd i <- x >= lb & x <= ub Then I draw a **Polygon** to show the 68% of data, so will be as below: polygon(c(lb,x[i],ub), c(0,hx[i],0),
col="red")



also we can specify the 98% of data distribution by writing the below code:

lb<-mean-1.5*sd ub<-mean+1.5*sd

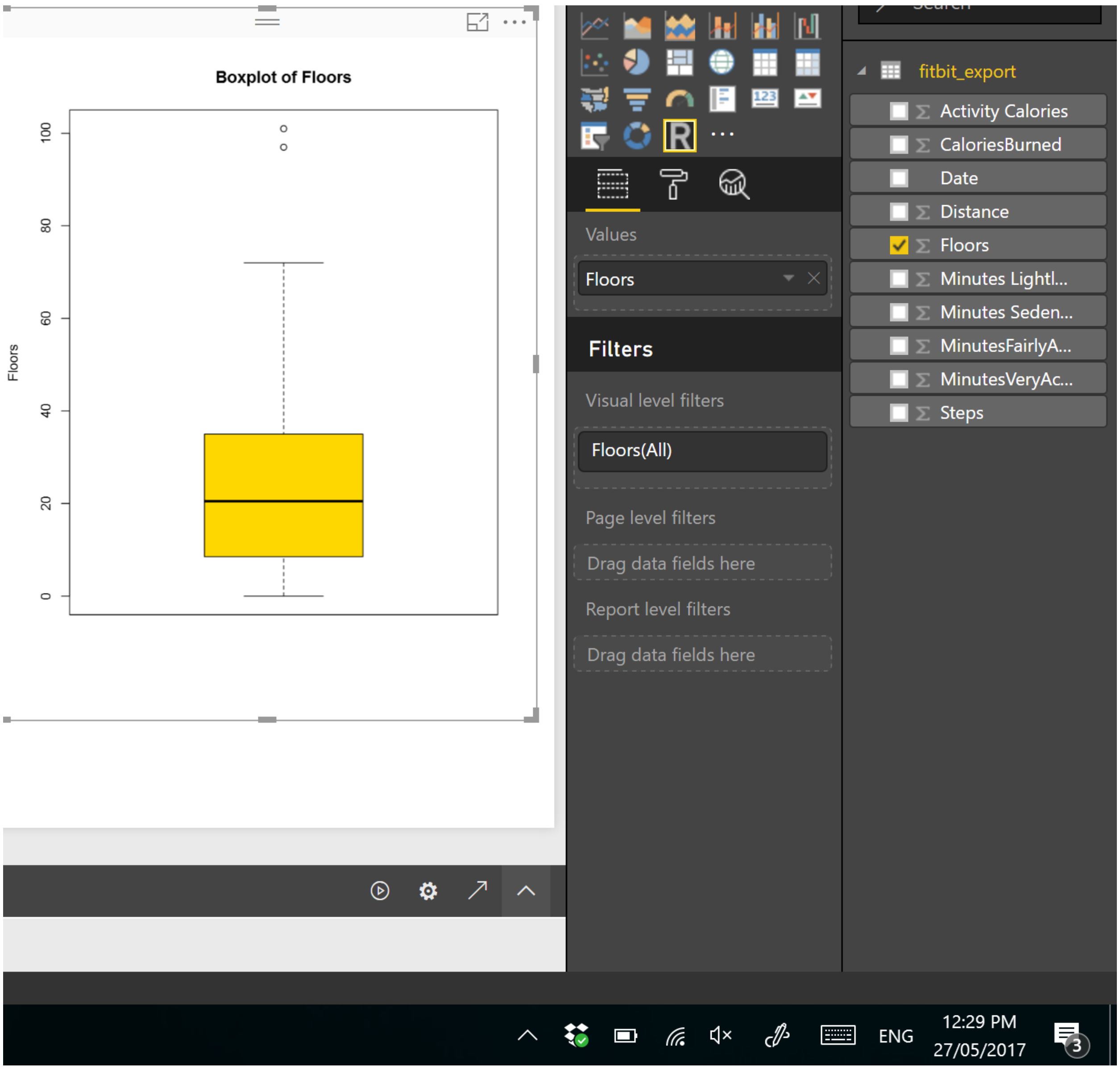
the result will be as below



[1].[Machine Learning with R,Brett Lantz, Packt Publishing,2015](#).

7-Visualizing Numeric Variables in Power BI – boxplots -Part 6

Published Date : May 27, 2017



In this post and next one, I am going to show how to see data distribution using some visuals like histogram, boxplot and normal distribution chart.

It is always important to have a holistic perspective regarding the minimum, maximum, middle, outliers of our data in one picture.

One of the chart that helps us to have a perspective regard these values in **"Box Plot" in R.**

For example, I am going to check my Fitbit data to see what is data **min**, **max**, **meadian**, **outlier**, **first and third quadrant** data using Box-Plot chart.

I draw a box-pot chart for checking the statistic measure number of floors I did in last 3 months, in PowerBI using R scripts.

The below codes can be used for drawing the box-plot chart. So, I choose the **Floors** field from power bi fields and then in R scripts I refer to ii.

```
boxplot(dataset$Floors, main="Boxplot of Floors",                ylab="Floors")
```

As you can see in above code, there is a function name "**boxplot**" which help me to draws a box plot. It gets the (**dataset$Floors**) as the first argument. Then, it gets the named of the chart as the second argument and the y axis name as the third inputs.

I have run the code in Power BI and I the below chart appear in PowerBI.

this chart shows the minimum and maximum of the number of floors I did in last three months. As you can see in the picture the minimum number of floors was "0" (the line at the bottom on the chart)and maximum is "70" (the line at the top of the chart). However

I am able to see the median of data (middle value) is around 20 (the bold line in middle of the chart).

What is median!

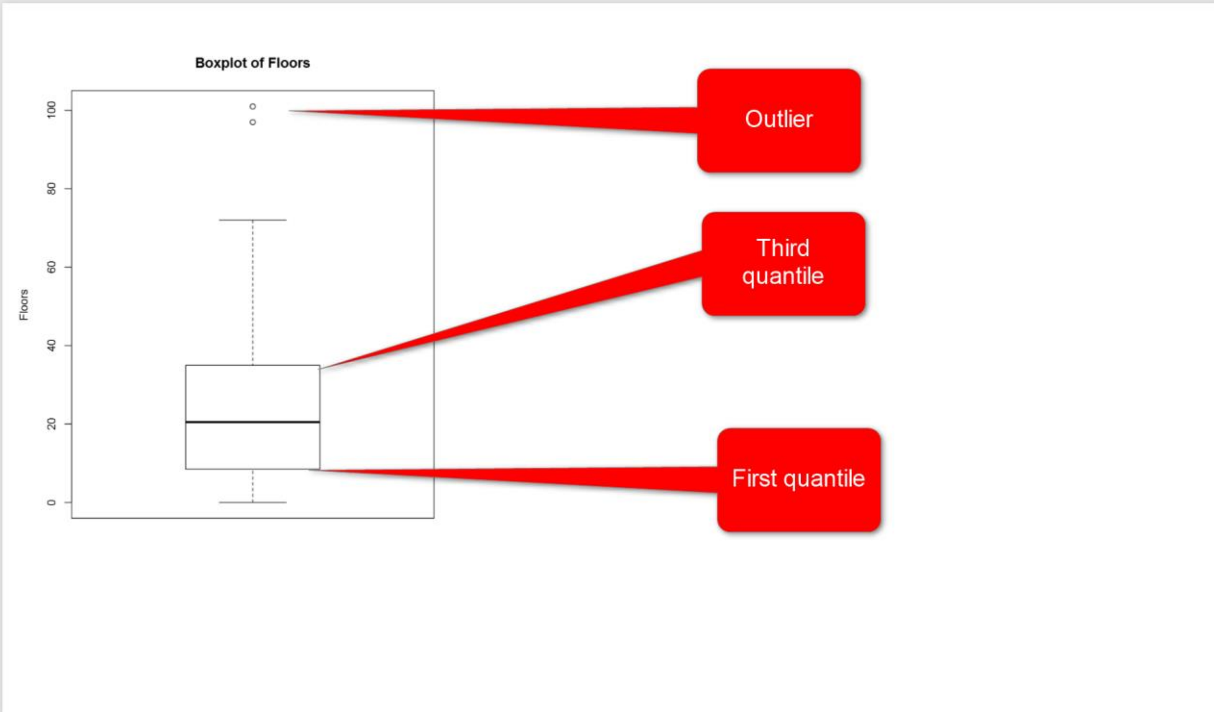
Imagine we have a dataset as (1,4,7,9,16,22,34,45,67) it is a sorted dataset, find the number that physically placed in middle of the dataset, I think **16** is physically located in middle of the list, so the median of this dataset is **16**. Median is not the mean value. Mean or average is summation of all data divided by number of data for the sample dataset is **22**, so **22#16**! that means mean is not equal to Median, lets change the dataset a bit :(1,4,7,9,16,22,**25,30,35**), median is still 16, but mean change:**16.5**. so in the second dataset I exclude the outlier (45,67) and it impacts on the mean not median!

Note: if we have lots of outlier in both side of our data range mean will be impact, more outlier at the upper range of our data (as above example) we have bigger mean than median, or if we have more lower outliers our mean value will be lower than median.

In the boxplot we just able to see the median value.

First Quarter and Third Quarter

We have two other measure name as **first and third quantile**. **First quantile** (see below picture), is the median value for the data range from minimum of data to median of data, so above example we just look at the data range from (1,4,7,9,16) and we find the median which is 7 so the first quantile is 7. Third quantile is the median for data range from (median to maximum).

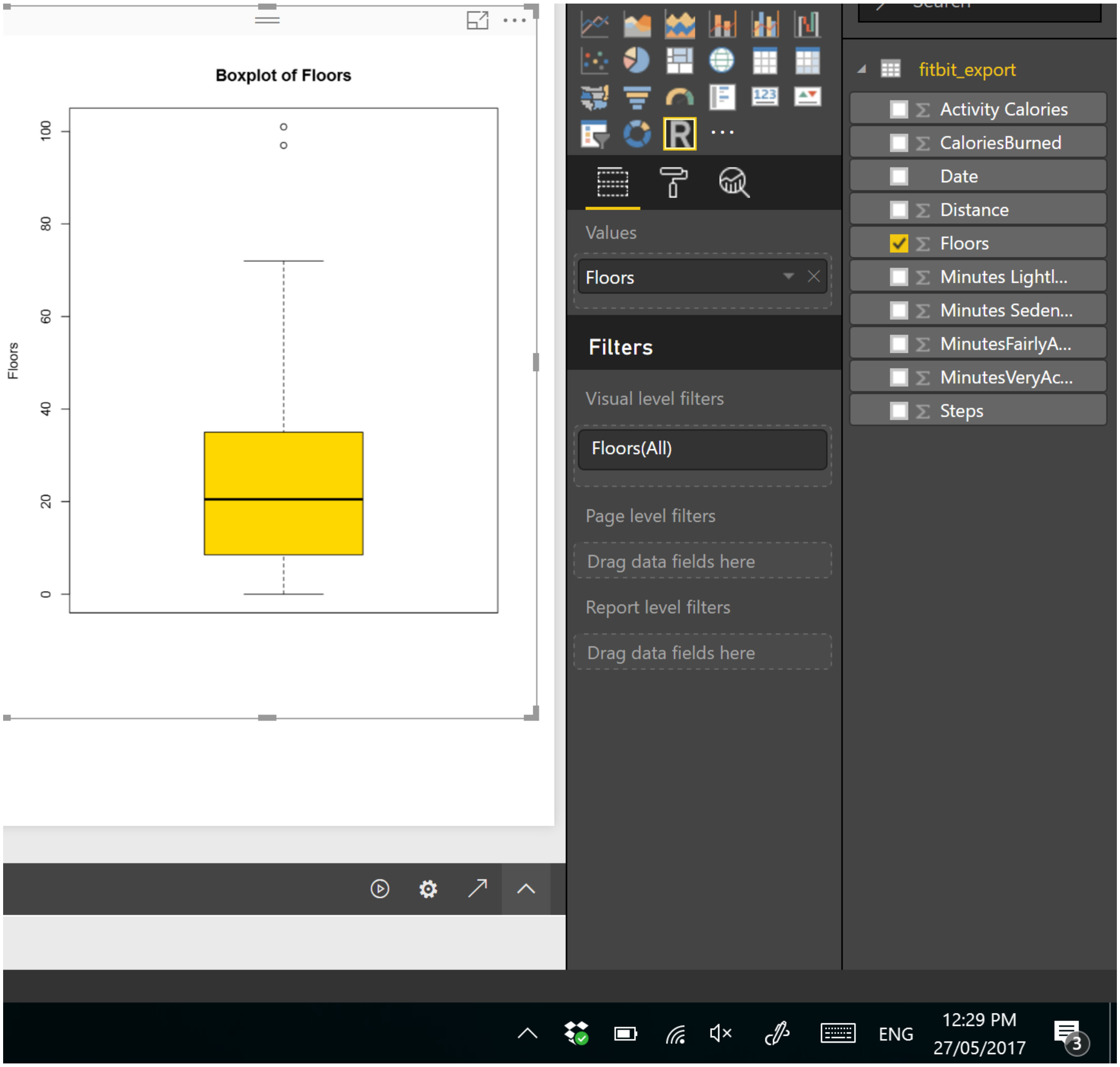


In above picture, you see two lines in the middle of the picture, they are first and third quantiles. the bold line is median. So, for my Fitbit data and for number of floors I have 10 for **First Quantile,** and for the **Third Quantile** we have 35.

However you see, there are some "not filled dot" in above of the chart that shows the outliers for floors number, they will impact on the mean value but not on median value. In Fitbit dataset, occasionally, I did 100 floors, which is a shame! :D, sometimes it is good to remove outliers data from charts to make data more smooth, so for machine learning analysis to get a better result some times it is good to remove them.

If you need more color change the code as below

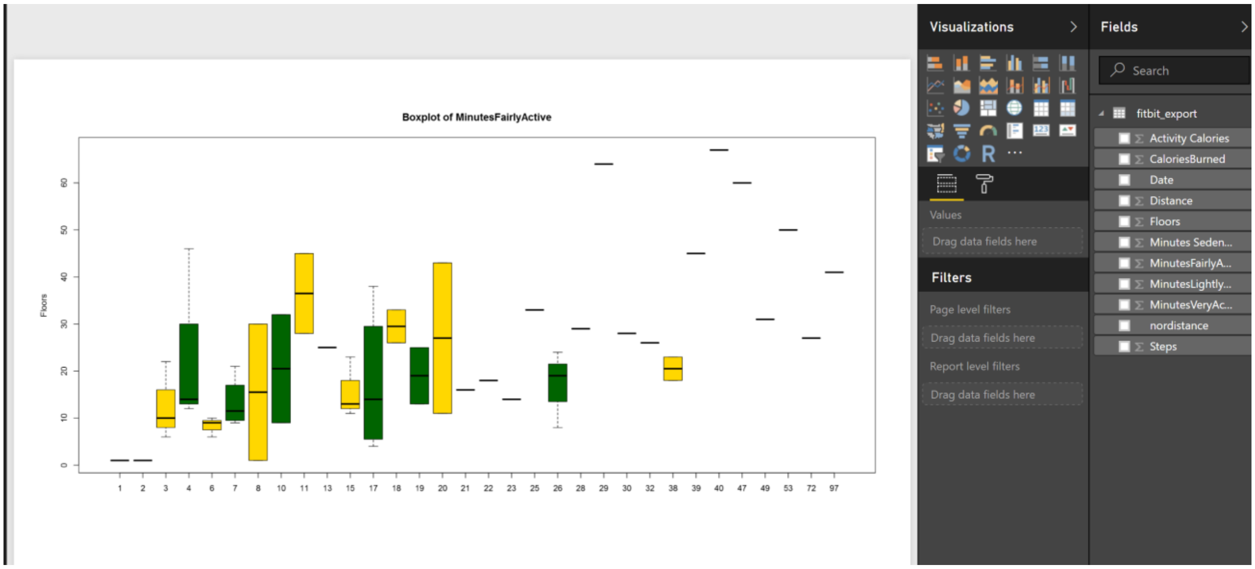
boxplot(dataset$Floors,main="BoxplotofFloors", ylab="Floors", **col=(c("gold")))**



or sometimes, you prefer to compare two attributes together then, for instance I am interested to see what is the median.

boxplot(**MinutesFairlyActive~Floors**, data=dataset,main="Boxplot of MinutesFairlyActive", ylab="Floors", col=(c("**gold","dark green**")))

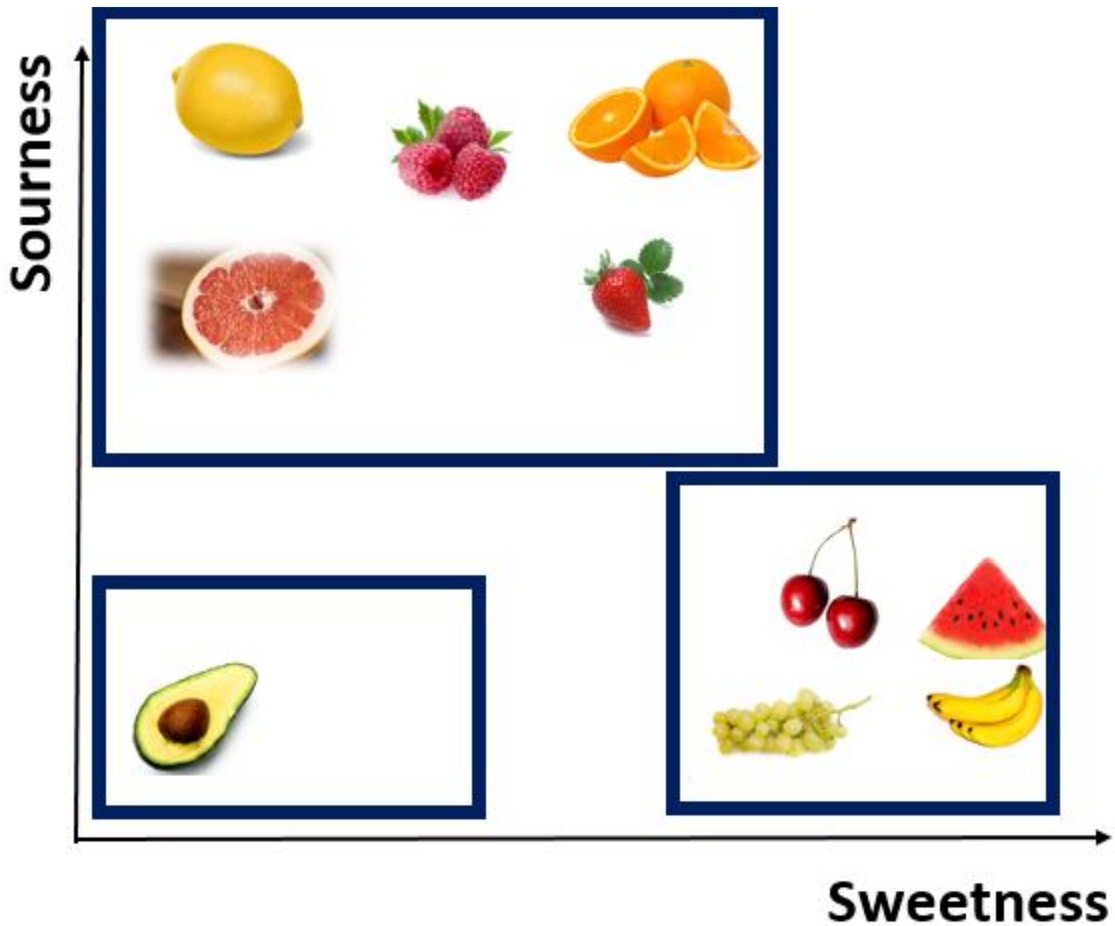
So to compare the statistics of minutes that I was fairly active to number of Floors, I change the code a bit, and compare them against each other, also I add another color to show them as below



In next post, I will talk about histogram that also show the data distribution and normal curve in detail!

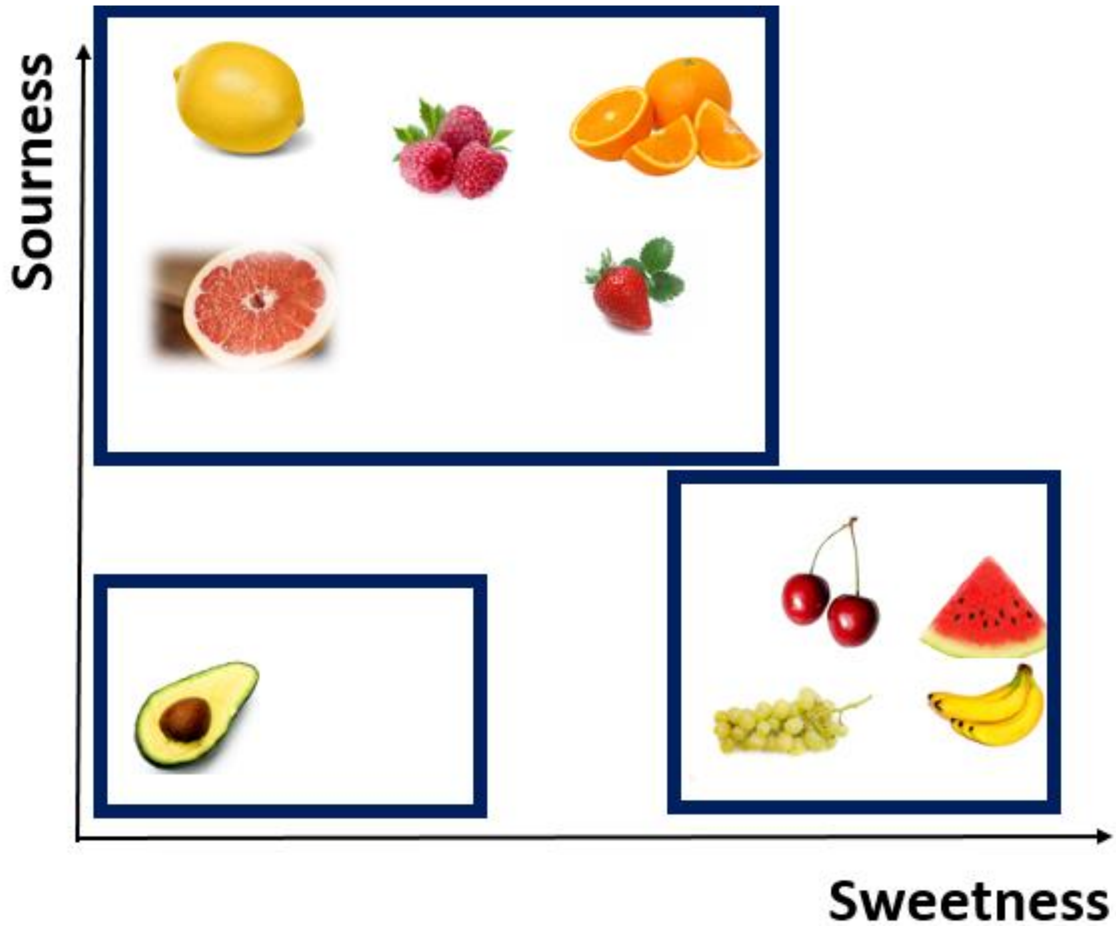
8-Prediction via KNN (K Nearest Neighbours) Concepts: Part 1

Published Date : March 22, 2017



K Nearest Neighbor (KNN) is one of those algorithms that are very easy to understand and has a good accuracy in practice. KNN can be used in different fields from health, marketing, finance and so on [1]. KNN is easy to understand and also the code behind it in R also is easy to write. In this post, I will explain the main concept behind KNN. Then in Part 2 I will show how to write R codes for KNN. Finally in the Part 3 the process of how run KNN in Power BI data will be explained.

To understand the KNN concepts, consider below example: We are designing a game for children below 6 . First, we asked them to close their eyes and then to taste a fruit, and identify is it sour or sweet. Based on their answers, we have below diagram:



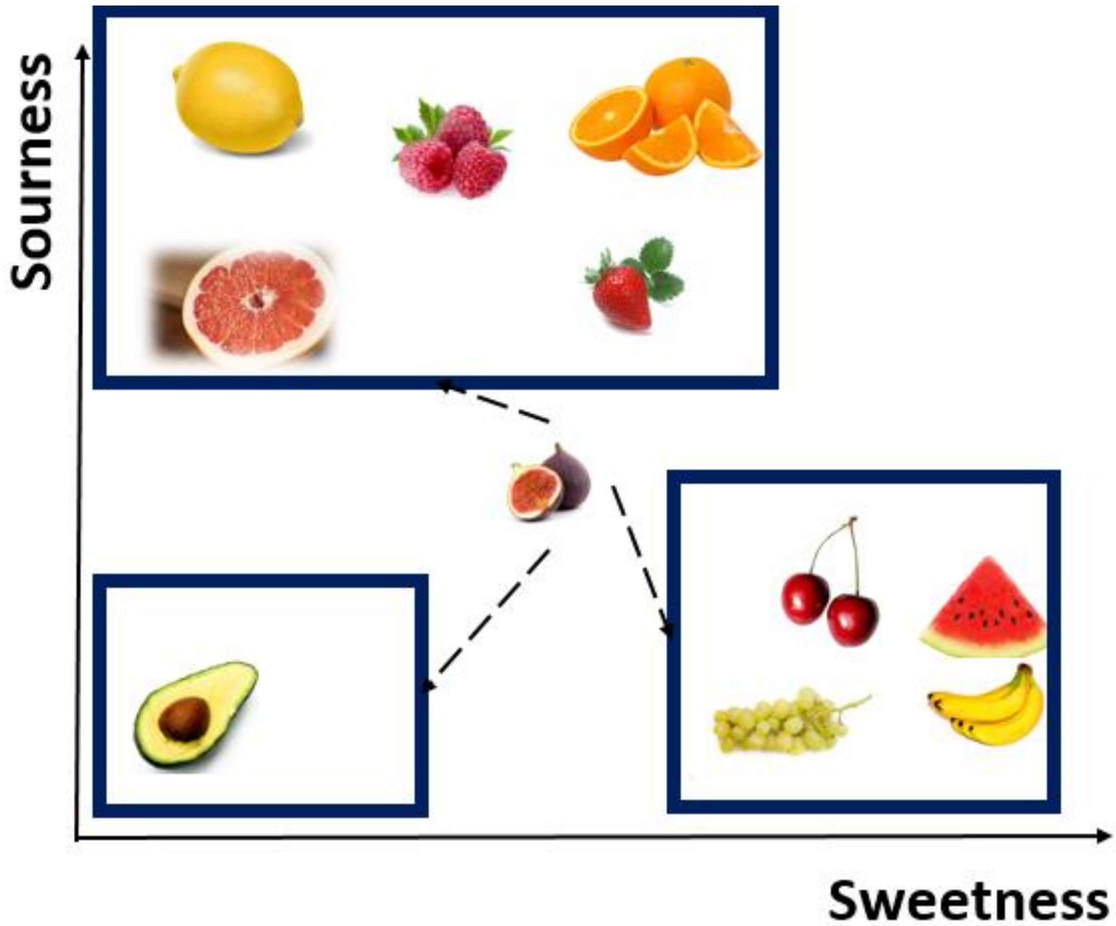
As you can see we have three main groups based on the level of sweetness and sourness. we asked children to put a number of sweetness and sourness for each fruit in scale of 1 - 10. We have below numbers. As you can see Lemon for example, has the high number in Sourness and low number in sweetness. Whist, Watermelon has high number (9) in sweetness and number 1 for sourness. (this is a example maybe the number is not correct, the aim of this example to show the concepts behind the KNN)

Fruite	Sweetness	Sourness	Fruite Type
Lemon	1	9	Sour
Grapfruite	2	8	Sour
Orange	3	7	Sour
Rasberry	2	8	Sour
Cherry	6	4	Sweet
banana	9	1	Sweet
Grapes	8	2	Sweet
Watremelon	9	1	Sweet
Avacado	1	1	None
Strawberry	5	5	Sour

Imagine that we have a fruit that is not in above list, we want to identify the nearness of that fruit to others and then identify it is a sweet fruit or sour one. Consider figs as example. to identify it is a sweet or sour fruit, we have some number of its level of sourness and sweetness as below

Fruite	Sweetnes	Sourness	Fruite Typ
Fig	7	3	Sour

as you can see for sweetness it is 7 and for sourness it is 3

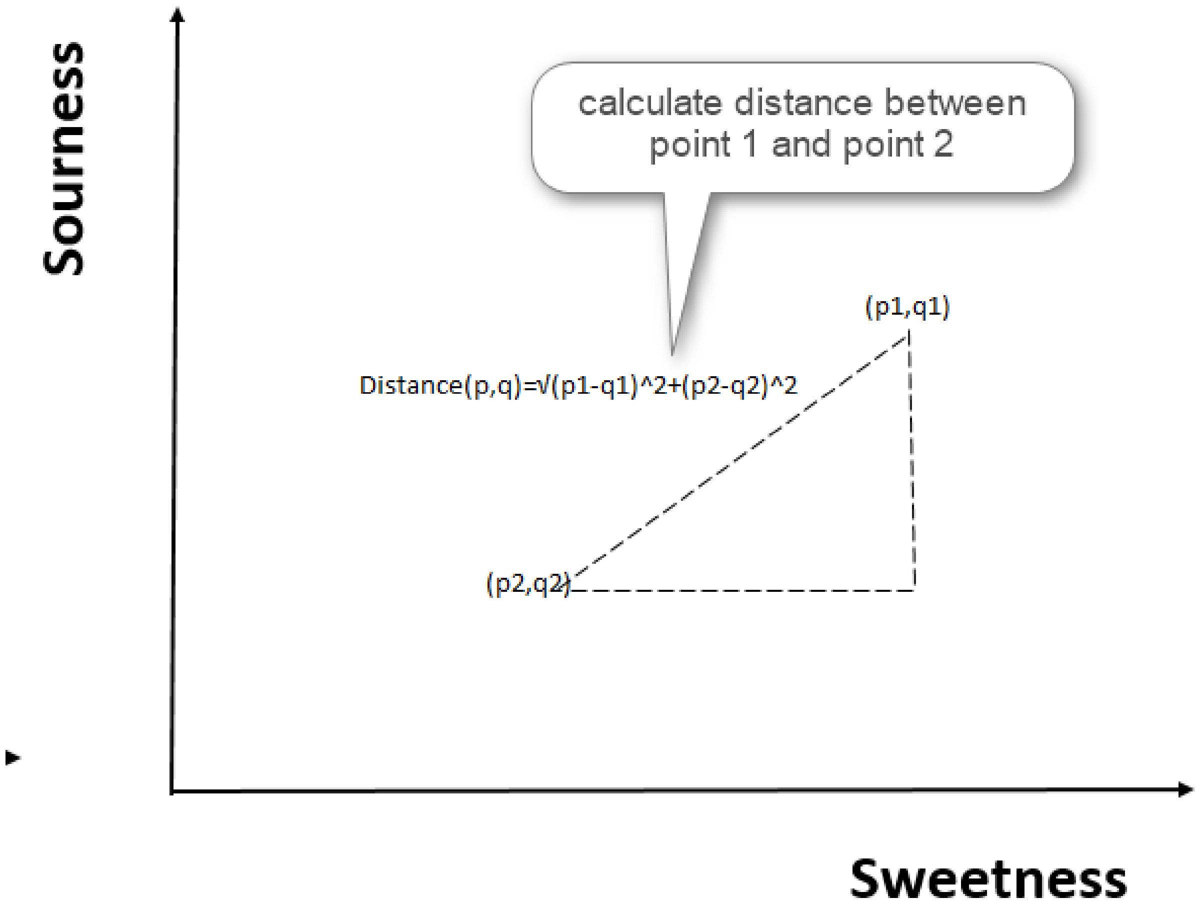


To find which fruit is near to this one, we should calculate the distance between Figs and other fruits.

From mathematics perspective, to find out distance between two points, we use the Euclidean distance formula as below:

Distance(p,q)=√(p1-q1)^2+(p2-q2)^2+...+(pi-qi)^2

Example= distance(Fig, Lemon)=√(7-1)^2+(3-9)^2=8.2



For calculating the distance between Figs and Lemon, we first subtract their dimensions (above formula)

Distance between Fig and Lemon is 8.2. Now we are going to calculate this distance for all other fruits. From below table, the distance between Cherry and Grapes is so close to Figs (distance 1.41)

Fruite	Sweetnes	Sourness	Fruite Type	Distance to Fig
Lemon	1	9	Sour	sqrt((1-7)^2+(9-3)^2)=8.4
Grapfruite	2	8	Sour	sqrt((2-7)^2+(8-3)^2)=7.1
Orange	3	7	Sour	sqrt((3-7)^2+(7-3)^2)=5.6
Rasberry	2	8	Sour	sqrt((2-7)^2+(8-3)^2)=7.1
Cherry	6	4	Sweet	sqrt((6-7)^2+(4-3)^2)=1.41
banana	9	1	Sweet	sqrt((9-7)^2+(1-3)^2)=2.82
Grapes	8	2	Sweet	sqrt((8-7)^2+(2-3)^2)=1.41
Watremelon	9	1	Sweet	sqrt((9-7)^2+(1-3)^2)=2.44
Avacado	1	1	None	sqrt((1-7)^2+(1-3)^2)=6.3
Strawberry	5	5	Sour	sqrt((5-7)^2+(5-3)^2)=2.82

Hence, Cherry and Grape are closet neighbor to Fig, we call them the first Nearest Neighbor. Watermelon with 2.44 is the Second Nearest Neighbor to Figs. the third nearest neighbor is strawberry and banana.

as you see in this example we calculate 8 nearest neighbor.

8 nearest neighbor for this example is Lemon with 8.4 distance. there is a lot distance between Lemon and Figs, so it is not correct to consider Lemon as nearest Neighbor. to find the best number for k(number of neighbors) we have consider the square root of the number of observations in our example. For instance,we have 10 observations which Square root is 3, so we have 3 nearest neighbors based on distance as first neighbor(Cherry and Grapes), second neighbor(Watermelon) and third is (Banana and Strawberry).

Because all of these are Sweet fruits, we consider Figs as a sweet one.

So in any example we calculate the distance of items to others categories. there other methods for calculating the distance.

KNN, has been used to predict a group for new items. for example:
1. predict that a customer will stay with us or not (new customer belong to with group: stay or leave)

2. image processing, if an uploaded picture of animal is related t birds, cats, and so on.

In the next post I will explain the related R codes for KNN .

[1]https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/

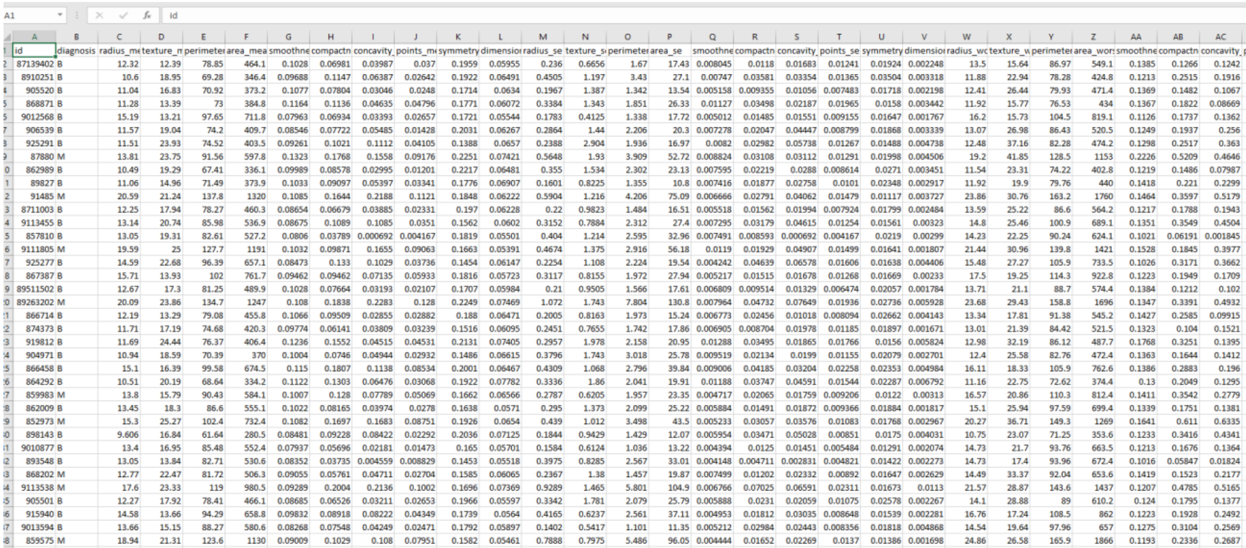
[2].Machine Learning with R,Brett Lantz, Packt Publishing,2015.

9-Prediction via KNN (K Nearest Neighbours) R codes: Part 2

Published Date : March 23, 2017

In the previous post (Part 1), I have explained the concepts of KNN and how it works. In this post, I will explain how to use KNN for predict whether a patient with Cancer will be Benign or Malignant. This example is get from Brett book[1]. Imagine that we have a dataset on laboratory results of some patients that some of them already Benign or Malignant. See below picture.

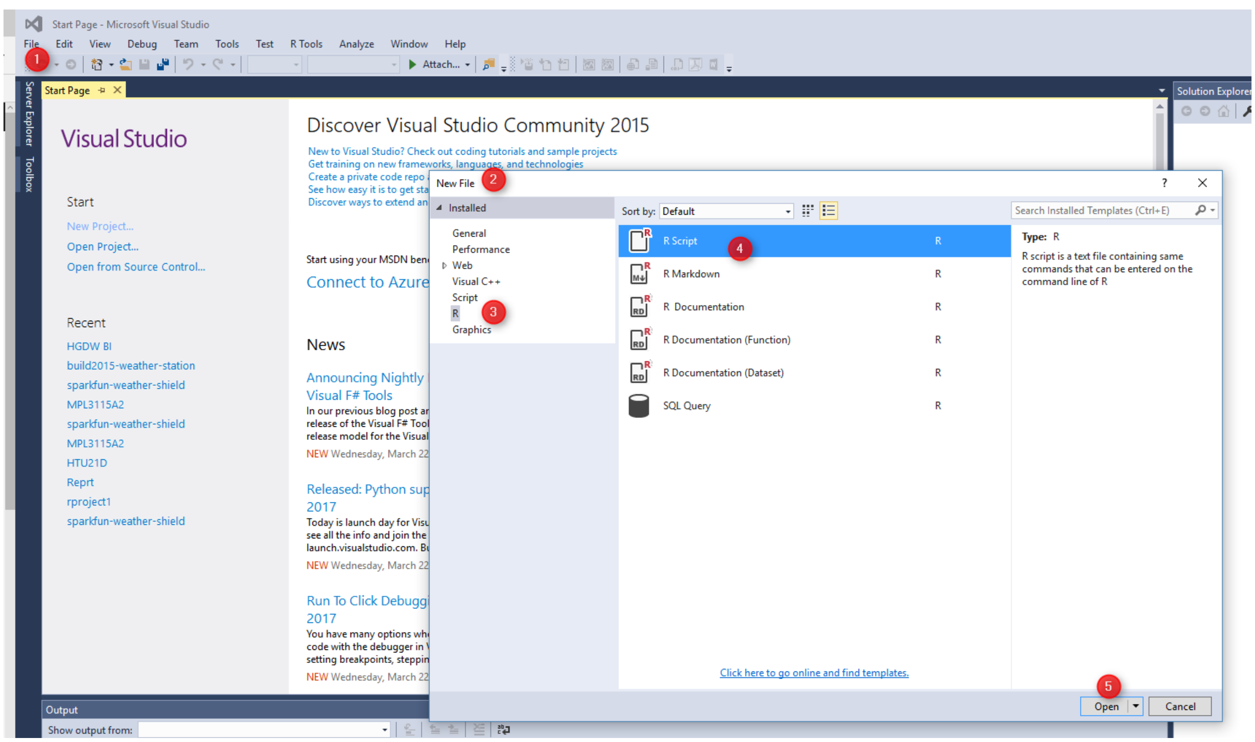
the first column is patient ID, the second one is the diagnosis for each patient: B stand for Benign and M stand for Malignant. the other columns are the laboratory results (I am not good on understanding them!)



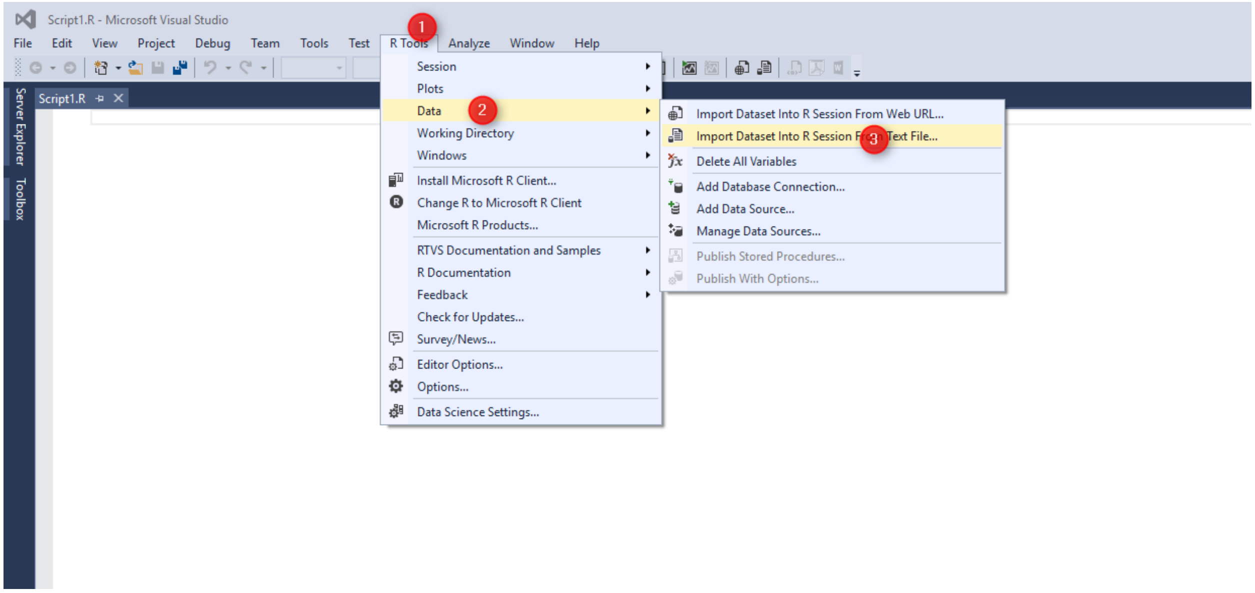
id	diagnosis	radius_me	texture_m	perimeter	area_mea	smoothne	compactn	concavity_	points_me	symmetry	dimensior	radius_se	texture_s	perimeter	area_se	smoothne	compactn	concavity_	points_se	symmetry	dimensior	radius_wc	texture_w	perimeter	area_wors	smoothne	compactn	concavity_
87139402	B	12.32	12.39	78.85	464.1	0.1028	0.06981	0.03987	0.037	0.1959	0.05955	0.236	0.6656	1.67	17.43	0.008045	0.0118	0.01683	0.01241	0.01924	0.002248	13.5	15.64	86.97	549.1	0.1385	0.1266	0.1242
8910251	B	10.6	18.95	69.28	346.4	0.09688	0.1147	0.06387	0.02642	0.1922	0.06491	0.4505	1.197	3.43	27.1	0.00747	0.03581	0.03354	0.01365	0.03504	0.003318	11.88	22.94	78.28	424.8	0.1213	0.2515	0.1916
905520	B	11.04	16.83	70.92	373.2	0.1077	0.07804	0.03046	0.0248	0.1714	0.0634	0.1967	1.387	1.342	13.54	0.005158	0.009355	0.01056	0.007483	0.01718	0.002198	12.41	26.44	79.93	471.4	0.1369	0.1482	0.1067
868871	B	11.28	13.39	73	384.8	0.1164	0.1136	0.04635	0.04796	0.1771	0.06072	0.3384	1.343	1.851	26.33	0.01127	0.03498	0.02187	0.01965	0.0158	0.003442	11.92	15.77	76.53	434	0.1367	0.1822	0.08669
9012568	B	15.19	13.21	97.65	711.8	0.07963	0.06934	0.03393	0.02657	0.1721	0.05544	0.1783	0.4125	1.338	17.72	0.005012	0.01485	0.01551	0.009155	0.01647	0.001767	16.2	15.73	104.5	819.1	0.1126	0.1737	0.1362
906539	B	11.57	19.04	74.2	409.7	0.08546	0.07722	0.05485	0.01428	0.2031	0.06267	0.2864	1.44	2.206	20.3	0.007278	0.02047	0.04447	0.008799	0.01868	0.003339	13.07	26.98	86.43	520.5	0.1249	0.1937	0.256
925291	B	11.51	23.93	74.52	403.5	0.09261	0.1021	0.1112	0.04105	0.1388	0.0657	0.2388	2.904	1.936	16.97	0.0082	0.02982	0.05738	0.01267	0.01488	0.004738	12.48	37.16	82.28	474.2	0.1298	0.2517	0.363
87880	M	13.81	23.75	91.56	597.8	0.1323	0.1768	0.1558	0.09176	0.2251	0.07421	0.5648	1.93	3.909	52.72	0.008824	0.03108	0.03112	0.01291	0.01998	0.004506	19.2	41.85	128.5	1153	0.2226	0.5209	0.4646
862989	B	10.49	19.29	67.41	336.1	0.09989	0.08578	0.02995	0.01201	0.2217	0.06481	0.355	1.534	2.302	23.13	0.007595	0.02219	0.0288	0.008614	0.0271	0.003451	11.54	23.31	74.22	402.8	0.1219	0.1486	0.07987
89827	B	11.06	14.96	71.49	373.9	0.1033	0.09097	0.05397	0.03341	0.1776	0.06907	0.1601	0.8225	1.355	10.8	0.007416	0.01877	0.02758	0.0101	0.02348	0.002917	11.92	19.9	79.76	440	0.1418	0.221	0.2299
91485	M	20.59	21.24	137.8	1320	0.1085	0.1644	0.2188	0.1121	0.1848	0.06222	0.5904	1.216	4.206	75.09	0.006666	0.02791	0.04062	0.01479	0.01117	0.003727	23.86	30.76	163.2	1760	0.1464	0.3597	0.5179
8711003	B	12.25	17.94	78.27	460.3	0.08654	0.06679	0.03885	0.02331	0.197	0.06228	0.22	0.9823	1.484	16.51	0.005518	0.01562	0.01994	0.007924	0.01799	0.002484	13.59	25.22	86.6	564.2	0.1217	0.1788	0.1943
9113455	B	13.14	20.74	85.98	536.9	0.08675	0.1089	0.1085	0.0351	0.1562	0.0602	0.3152	0.7884	2.312	27.4	0.007295	0.03179	0.04615	0.01254	0.01561	0.00323	14.8	25.46	100.9	689.1	0.1351	0.3549	0.4504
857810	B	13.05	19.31	82.61	527.2	0.0806	0.03789	0.000692	0.004167	0.1819	0.05501	0.404	1.214	2.595	32.96	0.007491	0.008593	0.000692	0.004167	0.0219	0.00299	14.23	22.25	90.24	624.1	0.1021	0.06191	0.001845
9111805	M	19.59	25	127.7	1191	0.1032	0.09871	0.1655	0.09063	0.1663	0.05391	0.4674	1.375	2.916	56.18	0.0119	0.01929	0.04907	0.01499	0.01641	0.001807	21.44	30.96	139.8	1421	0.1528	0.1845	0.3977
925277	B	14.59	22.68	96.39	657.1	0.08473	0.133	0.1029	0.03736	0.1454	0.06147	0.2254	1.108	2.224	19.54	0.004242	0.04639	0.06578	0.01606	0.01638	0.004406	15.48	27.27	105.9	733.5	0.1026	0.3171	0.3662
867387	B	15.71	13.93	102	761.7	0.09462	0.09462	0.07135	0.05933	0.1816	0.05723	0.3117	0.8155	1.972	27.94	0.005217	0.01515	0.01678	0.01268	0.01669	0.00233	17.5	19.25	114.3	922.8	0.1223	0.1949	0.1709
89511502	B	12.67	17.3	81.25	489.9	0.1028	0.07664	0.03193	0.02107	0.1707	0.05984	0.21	0.9505	1.566	17.61	0.006809	0.009514	0.01329	0.006474	0.02057	0.001784	13.71	21.1	88.7	574.4	0.1384	0.1212	0.102
89263202	M	20.09	23.86	134.7	1247	0.108	0.1838	0.2283	0.128	0.2249	0.07469	1.072	1.743	7.804	130.8	0.007964	0.04732	0.07649	0.01936	0.02736	0.005928	23.68	29.43	158.8	1696	0.1347	0.3391	0.4932
866714	B	12.19	13.29	79.08	455.8	0.1066	0.09509	0.02855	0.02882	0.188	0.06471	0.2005	0.8163	1.973	15.24	0.006773	0.02456	0.01018	0.008094	0.02662	0.004143	13.34	17.81	91.38	545.2	0.1427	0.2585	0.09915
874373	B	11.71	17.19	74.68	420.3	0.09774	0.06141	0.03809	0.03239	0.1516	0.06095	0.2451	0.7655	1.742	17.86	0.006905	0.008704	0.01978	0.01185	0.01897	0.001671	13.01	21.39	84.42	521.5	0.1323	0.104	0.1521
919812	B	11.69	24.44	76.37	406.4	0.1236	0.1552	0.04515	0.04531	0.2131	0.07405	0.2957	1.978	2.158	20.95	0.01288	0.03495	0.01865	0.01766	0.0156	0.005824	12.98	32.19	86.12	487.7	0.1768	0.3251	0.1395
904971	B	10.94	18.59	70.39	370	0.1004	0.0746	0.04944	0.02932	0.1486	0.06615	0.3796	1.743	3.018	25.78	0.009519	0.02134	0.0199	0.01155	0.02079	0.002701	12.4	25.58	82.76	472.4	0.1363	0.1644	0.1412
866458	B	15.1	16.39	99.58	674.5	0.115	0.1807	0.1138	0.08534	0.2001	0.06467	0.4309	1.068	2.796	39.84	0.009006	0.04185	0.03204	0.02258	0.02353	0.004984	16.11	18.33	105.9	762.6	0.1386	0.2883	0.196
864292	B	10.51	20.19	68.64	334.2	0.1122	0.1303	0.06476	0.03068	0.1922	0.07782	0.3336	1.86	2.041	19.91	0.01188	0.03747	0.04591	0.01544	0.02287	0.006792	11.16	22.75	72.62	374.4	0.13	0.2049	0.1295
859983	M	13.8	15.79	90.43	584.1	0.1007	0.128	0.07789	0.05069	0.1662	0.06566	0.2787	0.6205	1.957	23.35	0.004717	0.02065	0.01759	0.009206	0.0122	0.00313	16.57	20.86	110.3	812.4	0.1411	0.3542	0.2779
862009	B	13.45	18.3	86.6	555.1	0.1022	0.08165	0.03974	0.0278	0.1638	0.0571	0.295	1.373	2.099	25.22	0.005884	0.01491	0.01872	0.009366	0.01884	0.001817	15.1	25.94	97.59	699.4	0.1339	0.1751	0.1381
852973	M	15.3	25.27	102.4	732.4	0.1082	0.1697	0.1683	0.08751	0.1926	0.0654	0.439	1.012	3.498	43.5	0.005233	0.03057	0.03576	0.01083	0.01768	0.002967	20.27	36.71	149.3	1269	0.1641	0.611	0.6335
898143	B	9.606	16.84	61.64	280.5	0.08481	0.09228	0.08422	0.02292	0.2036	0.07125	0.1844	0.9429	1.429	12.07	0.005954	0.03471	0.05028	0.00851	0.0175	0.004031	10.75	23.07	71.25	353.6	0.1233	0.3416	0.4341
9010877	B	13.4	16.95	85.48	552.4	0.07937	0.05696	0.02181	0.01473	0.165	0.05701	0.1584	0.6124	1.036	13.22	0.004394	0.0125	0.01451	0.005484	0.01291	0.002074	14.73	21.7	93.76	663.5	0.1213	0.1676	0.1364
893548	B	13.05	13.84	82.71	530.6	0.08352	0.03735	0.004559	0.008829	0.1453	0.05518	0.3975	0.8285	2.567	33.01	0.004148	0.004711	0.002831	0.004821	0.01422	0.002273	14.73	17.4	93.96	672.4	0.1016	0.05847	0.01824
868202	M	12.77	22.47	81.72	506.3	0.09055	0.05761	0.04711	0.02704	0.1585	0.06065	0.2367	1.38	1.457	19.87	0.007499	0.01202	0.02332	0.00892	0.01647	0.002629	14.49	33.37	92.04	653.6	0.1419	0.1523	0.2177
9113538	M	17.6	23.33	119	980.5	0.09289	0.2004	0.2136	0.1002	0.1696	0.07369	0.9289	1.465	5.801	104.9	0.006766	0.07025	0.06591	0.02311	0.01673	0.0113	21.57	28.87	143.6	1437	0.1207	0.4785	0.5165
905501	B	12.27	17.92	78.41	466.1	0.08685	0.06526	0.03211	0.02653	0.1966	0.05597	0.3342	1.781	2.079	25.79	0.005888	0.0231	0.02059	0.01075	0.02578	0.002267	14.1	28.88	89	610.2	0.124	0.1795	0.1377
915940	B	14.58	13.66	94.29	658.8	0.09832	0.08918	0.08222	0.04349	0.1739	0.0564	0.4165	0.6237	2.561	37.11	0.004953	0.01812	0.03035	0.008648	0.01539	0.002281	16.76	17.24	108.5	862	0.1223	0.1928	0.2492
9013594	B	13.66	15.15	88.27	580.6	0.08268	0.07548	0.04249	0.02471	0.1792	0.05897	0.1402	0.5417	1.101	11.35	0.005212	0.02984	0.02443	0.008356	0.01818	0.004868	14.54	19.64	97.96	657	0.1275	0.3104	0.2569
859575	M	18.94	21.31	123.6	1130	0.09009	0.1029	0.108	0.07951	0.1582	0.05461	0.7888	0.7975	5.486	96.05	0.004444	0.01652	0.02269	0.0137	0.01386	0.001698	24.86	26.58	165.9	1866	0.1193	0.2336	0.2687

We want to create a prediction models for a new patient with specific laboratory results, we want to predict whether this patient will be Benign or Malignant.

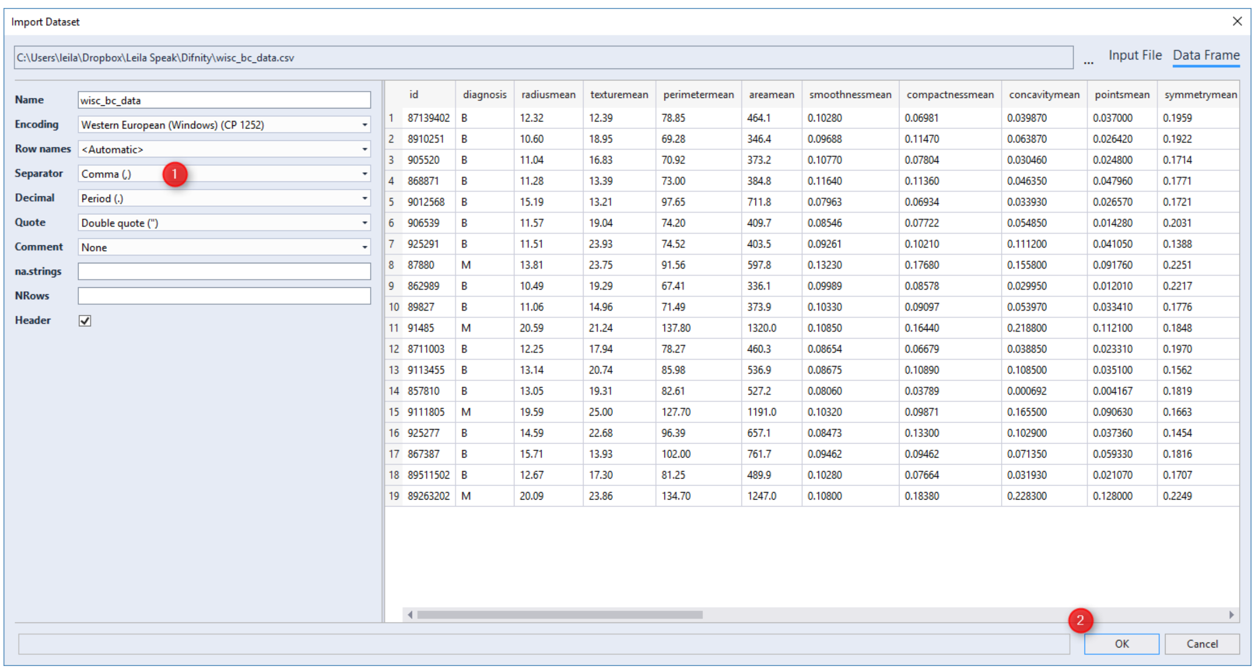
For this demo, I will use R environment in Visual Studio. Hence, after opening Visual Studio 2015, select File, New file and then under the General tab find "R". I am going to write R codes in R scripts (Number 4) and then create a R scripts there.



After creating an empty R scripts. Now I am going to import data. choose "R Tools", then in Data menu, then click on the "Import Dataset into R session".



You will see below window. It shows all the columns and the sample of data. The CSV file that I am using for this post has been produced by [1]. It is a CSV file with delimiter (number 1) by Comma.



	id	diagnosis	radiusmean	texturemean	perimetermean	areamean	smoothnessmean	compactnessmean	concavitymean	pointsmean	symmetrymean
1	87139402	B	12.32	12.39	78.85	464.1	0.10280	0.06981	0.039870	0.037000	0.1959
2	8910251	B	10.60	18.95	69.28	346.4	0.09688	0.11470	0.063870	0.026420	0.1922
3	905520	B	11.04	16.83	70.92	373.2	0.10770	0.07804	0.030460	0.024800	0.1714
4	868871	B	11.28	13.39	73.00	384.8	0.11640	0.11360	0.046350	0.047960	0.1771
5	9012568	B	15.19	13.21	97.65	711.8	0.07963	0.06934	0.033930	0.026570	0.1721
6	906539	B	11.57	19.04	74.20	409.7	0.08546	0.07722	0.054850	0.014280	0.2031
7	925291	B	11.51	23.93	74.52	403.5	0.09261	0.10210	0.111200	0.041050	0.1388
8	87880	M	13.81	23.75	91.56	597.8	0.13230	0.17680	0.155800	0.091760	0.2251
9	862989	B	10.49	19.29	67.41	336.1	0.09989	0.08578	0.029950	0.012010	0.2217
10	89827	B	11.06	14.96	71.49	373.9	0.10330	0.09097	0.053970	0.033410	0.1776
11	91485	M	20.59	21.24	137.80	1320.0	0.10850	0.16440	0.218800	0.112100	0.1848
12	8711003	B	12.25	17.94	78.27	460.3	0.08654	0.06679	0.038850	0.023310	0.1970
13	9113455	B	13.14	20.74	85.98	536.9	0.08675	0.10890	0.108500	0.035100	0.1562
14	857810	B	13.05	19.31	82.61	527.2	0.08060	0.03789	0.000692	0.004167	0.1819
15	9111805	M	19.59	25.00	127.70	1191.0	0.10320	0.09871	0.165500	0.090630	0.1663
16	925277	B	14.59	22.68	96.39	657.1	0.08473	0.13300	0.102900	0.037360	0.1454
17	867387	B	15.71	13.93	102.00	761.7	0.09462	0.09462	0.071350	0.059330	0.1816
18	89511502	B	12.67	17.30	81.25	489.9	0.10280	0.07664	0.031930	0.021070	0.1707
19	89263202	M	20.09	23.86	134.70	1247.0	0.10800	0.18380	0.228300	0.128000	0.2249

After importing the dataset, now we are going to see the summary of data by Function "STR". this function shows the summary of column's data and the data type of each column.

```
str(wisc_bc_data)
```

the result will be:

```
data.frame':	569 obs. of  32 variables:
 $ id             : int  87139402 8910251 905520 868871 9012568 906539 925291 87880 862989 89827 ...
 $ diagnosis       : Factor w/ 2 levels "B","M": 1 1 1 1 1 1 1 2 1 1 ...
 $ radius_mean      : num  12.3 10.6 11 11.3 15.2 ...
 $ texture_mean     : num  12.4 18.9 16.8 13.4 13.2 ...
 $ perimeter_mean   : num  78.8 69.3 70.9 73 97.7 ...
 $ area_mean        : num  464 346 373 385 712 ...
 $ smoothness_mean  : num  0.1028 0.0969 0.1077 0.1164 0.0796 ...
 $ compactness_mean : num  0.0698 0.1147 0.078 0.1136 0.0693 ...
 $ concavity_mean   : num  0.0399 0.0639 0.0305 0.0464 0.0339 ...
 $ points_mean      : num  0.037 0.0264 0.0248 0.048 0.0266 ...
 $ symmetry_mean    : num  0.196 0.192 0.171 0.177 0.172 ...
 $ dimension_mean   : num  0.0595 0.0649 0.0634 0.0607 0.0554 ...
 $ radius_se        : num  0.236 0.451 0.197 0.338 0.178 ...
 $ texture_se       : num  0.666 1.197 1.387 1.343 0.412 ...
 $ perimeter_se     : num  1.67 3.43 1.34 1.85 1.34 ...
 $ area_se          : num  17.4 27.1 13.5 26.3 17.7 ...
 $ smoothness_se    : num  0.00805 0.00747 0.00516 0.01127 0.00501 ...
 $ compactness_se   : num  0.0118 0.03581 0.00936 0.03498 0.01485 ...
 $ concavity_se     : num  0.0168 0.0335 0.0106 0.0219 0.0155 ...
 $ points_se        : num  0.01241 0.01365 0.00748 0.01965 0.00915 ...
 $ symmetry_se      : num  0.0192 0.035 0.0172 0.0158 0.0165 ...
 $ dimension_se     : num  0.00225 0.00332 0.0022 0.00344 0.00177 ...
 $ radius_worst     : num  13.5 11.9 12.4 11.9 16.2 ...
 $ texture_worst    : num  15.6 22.9 26.4 15.8 15.7 ...
 $ perimeter_worst  : num  87 78.3 79.9 76.5 104.5 ...
 $ area_worst       : num  549 425 471 434 819 ...
 $ smoothness_worst : num  0.139 0.121 0.137 0.137 0.113 ...
 $ compactness_worst: num  0.127 0.252 0.148 0.182 0.174 ...
 $ concavity_worst  : num  0.1242 0.1916 0.1067 0.0867 0.1362 ...
 $ points_worst     : num  0.0939 0.0793 0.0743 0.0861 0.0818 ...
 $ symmetry_worst   : num  0.283 0.294 0.3 0.21 0.249 ...
 $ dimension_worst  : num  0.0677 0.0759 0.0788 0.0678 0.0677 ...
>
```

Now we want to keep the original dataset, so we put data in a temp variable “wbcd”

```
wbcd <- wisc_bc_data
```

The first column of data “id” could not be that much important in prediction, so we eliminate the first column from dataset.

```
wbcd<-wbcd[-1]
```

We want to look at the statistical summary of each column: such as min, max, mid, mean value of each columns.

summary(wbcd)

The result of running the code will be as below, as you can see for first column (we already delete the id column), we have 357 cases that are Benign and 212 Malignant cases. also for all other laboratory measurement we can see the min, max, median, mean. 1st Qu, and 3rd Qu.

diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean points_mean symmetry_mean dimension_mean
B:357 Min. : 6.981 Min. : 9.71 Min. : 43.79 Min. : 143.5 Min. :0.05263 Min. :0.01938 Min. :0.00000 Min. :0.00000 Min. :0.1060 Min. :0.04996
M:212 1st Qu.:11.700 1st Qu.:16.17 1st Qu.: 75.17 1st Qu.: 420.3 1st Qu.:0.08637 1st Qu.:0.06492 1st Qu.:0.02956 1st Qu.:0.02031 1st Qu.:0.1619 1st Qu.:0.05770
Median :13.370 Median :18.84 Median : 86.24 Median : 551.1 Median :0.09587 Median :0.09263 Median :0.06154 Median :0.03350 Median :0.1792 Median :0.06154
Mean :14.127 Mean :19.29 Mean : 91.97 Mean : 654.9 Mean :0.09636 Mean :0.10434 Mean :0.08880 Mean :0.04892 Mean :0.1812 Mean :0.06280
3rd Qu.:15.780 3rd Qu.:21.80 3rd Qu.:104.10 3rd Qu.: 782.7 3rd Qu.:0.10530 3rd Qu.:0.13040 3rd Qu.:0.13070 3rd Qu.:0.07400 3rd Qu.:0.1957 3rd Qu.:0.06612
Max. :28.110 Max. :39.28 Max. :188.50 Max. :2501.0 Max. :0.16340 Max. :0.34540 Max. :0.42680 Max. :0.20120 Max. :0.3040 Max. :0.09744
radius_se texture_se perimeter_se area_se smoothness_se compactness_se concavity_se points_se symmetry_se dimension_se
Min. :0.1115 Min. :0.3602 Min. : 0.757 Min. : 6.802 Min. :0.001713 Min. :0.002252 Min. :0.00000 Min. :0.000000 Min. :0.007882 Min. :0.0008948
1st Qu.:0.2324 1st Qu.:0.8339 1st Qu.: 1.606 1st Qu.: 17.850 1st Qu.:0.005169 1st Qu.:0.013080 1st Qu.:0.01509 1st Qu.:0.007638 1st Qu.:0.015160 1st Qu.:0.0022480
Median :0.3242 Median :1.1080 Median : 2.287 Median : 24.530 Median :0.006380 Median :0.020450 Median :0.02589 Median :0.010930 Median :0.018730 Median :0.0031870
Mean :0.4052 Mean :1.2169 Mean : 2.866 Mean : 40.337 Mean :0.007041 Mean :0.025478 Mean :0.03189 Mean :0.011796 Mean :0.020542 Mean :0.0037949
3rd Qu.:0.4789 3rd Qu.:1.4740 3rd Qu.: 3.357 3rd Qu.: 45.190 3rd Qu.:0.008146 3rd Qu.:0.032450 3rd Qu.:0.04205 3rd Qu.:0.014710 3rd Qu.:0.023480 3rd Qu.:0.0045580
Max. :2.8730 Max. :4.8850 Max. :21.980 Max. :542.200 Max. :0.031130 Max. :0.135400 Max. :0.39600 Max. :0.052790 Max. :0.078950 Max. :0.0298400
radius_worst texture_worst perimeter_worst area_worst smoothness_worst compactness_worst concavity_worst points_worst symmetry_worst dimension_worst
Min. : 7.93 Min. :12.02 Min. : 50.41 Min. : 185.2 Min. :0.07117 Min. :0.02729 Min. :0.0000 Min. :0.00000 Min. :0.1565 Min. :0.05504
1st Qu.:13.01 1st Qu.:21.08 1st Qu.: 84.11 1st Qu.: 515.3 1st Qu.:0.11660 1st Qu.:0.14720 1st Qu.:0.1145 1st Qu.:0.06493 1st Qu.:0.2504 1st Qu.:0.07146
Median :14.97 Median :25.41 Median : 97.66 Median : 686.5 Median :0.13130 Median :0.21190 Median :0.2267 Median :0.09993 Median :0.2822 Median :0.08004
Mean :16.27 Mean :25.68 Mean :107.26 Mean : 880.6 Mean :0.13237 Mean :0.25427 Mean :0.2722 Mean :0.11461 Mean :0.2901 Mean :0.08395
3rd Qu.:18.79 3rd Qu.:29.72 3rd Qu.:125.40 3rd Qu.:1084.0 3rd Qu.:0.14600 3rd Qu.:0.33910 3rd Qu.:0.3829 3rd Qu.:0.16140 3rd Qu.:0.3179 3rd Qu.:0.09208

```
 Max.   :36.04   Max.   :49.54   Max.   :251.20   Max.   :4254.0   Max.   :0.22260   Max.   :1.05800   Max.   :1.2520  
Max.   :0.29100   Max.   :0.6638   Max.   :0.20750  
>
```

Data Wrangling

first of all, we want to have a dataset that is easy to read. the first data cleaning is about replacing "B" value with Benign and "M" value with Malignant in diagnosis column. This replacement makes the data to be more informative. Hence we employ below code:

```
wbcd$diagnosis<- factor(wbcd$diagnosis, levels = c("B", "M"), labels = c("Benign", "Malignant"))
```

Factor is a function that gets the column name in a dataset, and we can identify the labels with out consuming memories)

There is another issue in the data: numbers are not normalized!

What is data normalization: that mean they are not on the same scale. For instance for radius mean all numbers between 6 to 29 while for column smoothness_mean is between 0.05 to 0.17. for performing the predict analysis using KNN, as we use distance calculation (Part 1), it is important all numbers should be in same range[1].

Normalization can be done by below formula

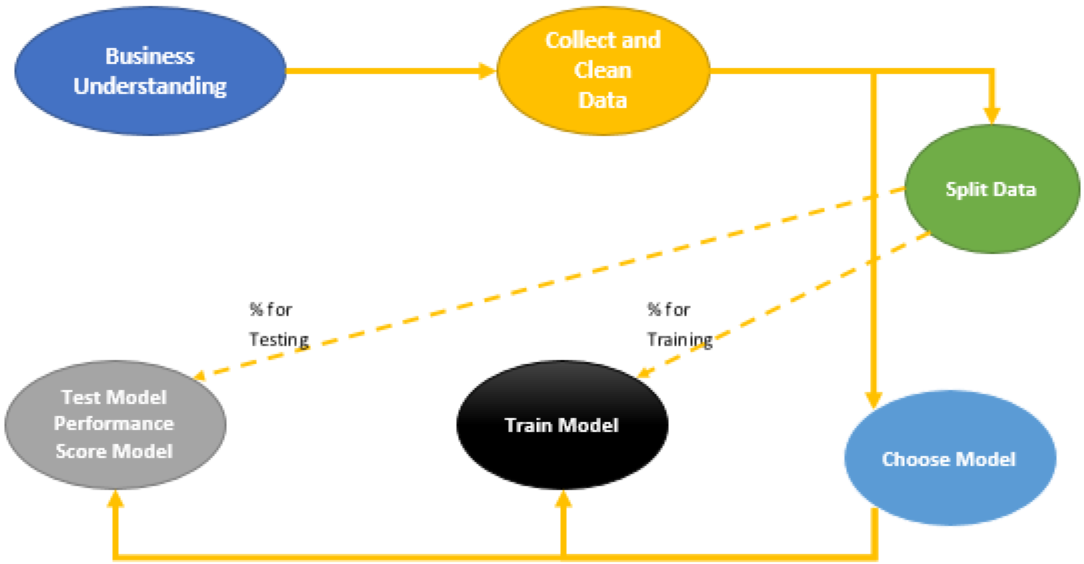
```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
```

Now we are going to apply this function in all numeric columns in wbcd dataset. There is a function in R that apply a function over a dataset:

```
wbcd_n <- as.data.frame(lapply(wbcd[2:31], normalize))
```

"lapply" gets the dataset and function name, then apply the function on all dataset. In this example because the first column is text (diagnosis), we apply "normalize" function on columns 2 to 31.Now our data is ready for creating a KNN model.

From machine learning process we need a dataset for training model and another for testing model (from Market basket analysis post)



Hence, we should have two different dataset for train and test. In this example, we going to have row number 1 to 469 for training and creating model and from row number 470 to 569 for testing the model.

```
wbcd_train <- wbcd_n[1:469, ]
 wbcd_test <- wbcd_n[470:569, ]
```

wbcd_train we have 469 rows of data and the rest in wbcd_test. Also we need the prediction label for result

```
wbcd_train_labels <- wbcd[1:469, 1]
wbcd_test_labels <- wbcd[470:569, 1]
```

Now data is ready, we are going to train our model and create KNN algorithm.

For using KNN there is a need to install package "Class"

```
install.packages("class")
```

Now we able to call function KNN to predict the patient diagnosis. KNN function accept the training dataset and test dataset as second arguments. moreover the prediction label also need for result. we want to use KNN based on the discussion on Part 1, to identify the number K (K nearest Neighbour), we should calculate the square root of observation. Here for 469 observation the K is 21.

```
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test,cl= wbcd_train_labels, k = 21)
```

The result is that "wbcd_test_pred" holds the result of the KNN prediction.

```
[1] Benign    Benign    Benign    Benign    Malignant Benign    Malignant Benign    Malignant Benign
Malignant Benign    Malignant Malignant Benign    Benign    Malignant Benign
 [19] Malignant Benign    Malignant Malignant Malignant Malignant Benign    Benign    Benign    Benign
Malignant Malignant Malignant Benign    Malignant Malignant Benign    Benign
 [37] Benign    Benign    Benign    Malignant Malignant Benign    Malignant Malignant Benign    Malignant
Malignant Malignant Malignant Malignant Benign    Benign    Benign    Benign
 [55] Benign    Benign    Benign    Benign    Malignant Benign    Benign    Benign    Benign    Benign
Malignant Malignant Benign    Benign    Benign    Benign    Benign    Malignant
 [73] Benign    Benign    Malignant Malignant Benign    Benign    Benign    Benign    Benign    Benign    Benign
Malignant Benign    Benign    Malignant Benign    Benign    Benign
 [91] Benign    Malignant Benign    Benign    Benign    Benign    Benign    Malignant Benign    Malignant
Levels: Benign Malignant
```

We want to evaluate the result of the model by installing "gmodels" a packages that shows performance evaluation.

```
install.packages("gmodels")
require("gmodels")
library("gmodels")
```

We employ a function name "CrossTable". It gets label as first input, the prediction result as second argument.

```
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred,
prop.chisq = FALSE)
```

The result of "Cross table" as below. We have 100 observation. The tables show the result of evaluation and see how accurate the KNN prediction is. The first row and first column shows the true positive (**TP**) cases, means the cases that already Benign and KNN predicts Benign. The first row and second column shows number of cases that already Benign and KNN predict they are Malignant (**TN**). The second row and first column is Malignant in real world but KNN predict they are Benign (**FP**). Finally the last column and last row is False Negative (**FN**) that means cases that they Malignant and KNN predict as Malignant.

```
Total Observations in Table:  100


                 | wbcd_test_pred
wbcd_test_labels |    Benign | Malignant | Row Total |
-----------------|-----------|-----------|-----------|
          Benign |        61 |         0 |        61 |
                 |     1.000 |     0.000 |     0.610 |
                 |     0.968 |     0.000 |           |
                 |     0.610 |     0.000 |           |
-----------------|-----------|-----------|-----------|
```

```
    Malignant |         2 |        37 |        39 |
              |     0.051 |     0.949 |     0.390 |
              |     0.032 |     1.000 |           |
              |     0.020 |     0.370 |           |
-----------------|-----------|-----------|-----------|
 Column Total |        63 |        37 |       100 |
              |     0.630 |     0.370 |           |
-----------------|-----------|-----------|-----------|
```

So as much as TP and FN is the higher the prediction is better. In our example TP is 61 and FN is 37, moreover the TN and TP is just 0 and 2 which is good.

To calculate the accuracy we should follow the below formula:

accuracy <- (tp + tn) / (tp + fn + fp + tn)

Accuracy will be (61+37)/(61+37+2+0)=98%

In the next post I will explained how to perform KNN in Power BI (data wrangling, modelling and visualization).

[1].Machine Learning with R,Brett Lantz, Packt Publishing,2015.

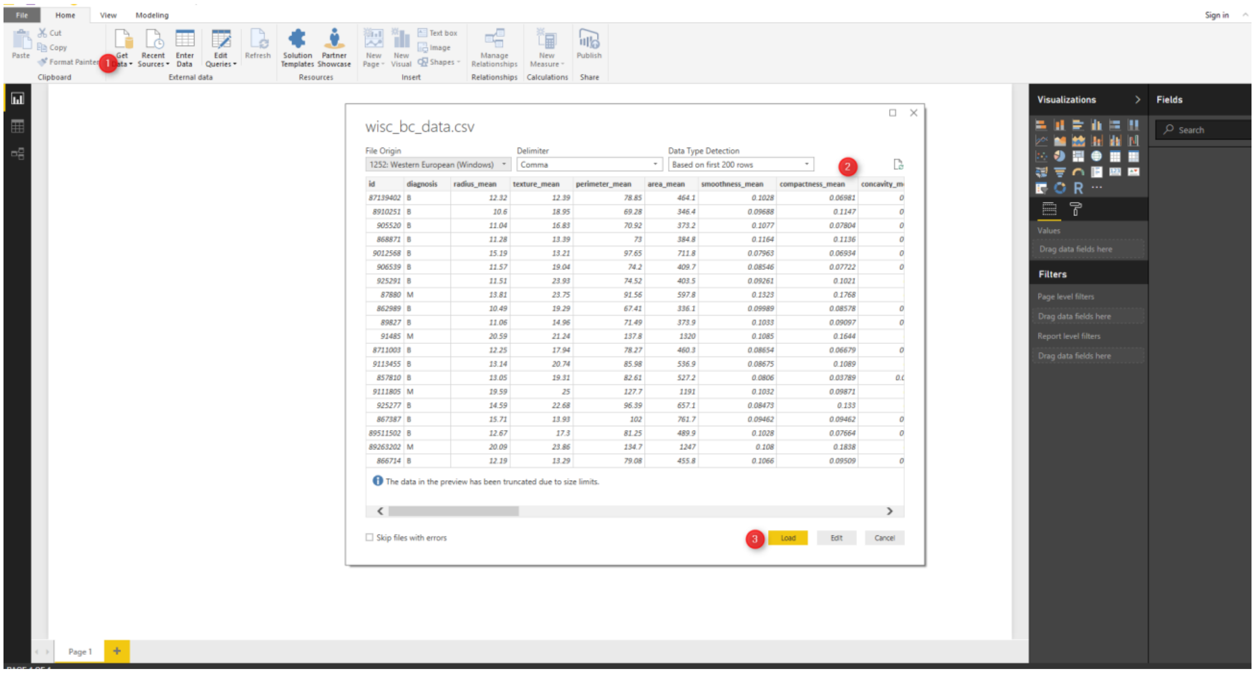
10-Prediction via KNN (K Nearest Neighbours) KNN Power BI: Part 3

Published Date : March 24, 2017

K Nearest Neighbour (KNN) is one of those algorithms that are very easy to understand and it has a high level of accuracy in practice. In Part One of this series, I have explained the KNN concepts. In Part 2 I have explained the R code for KNN, how to write R code and how to evaluate the KNN model. In this post, I want to show how to do KNN in Power BI.

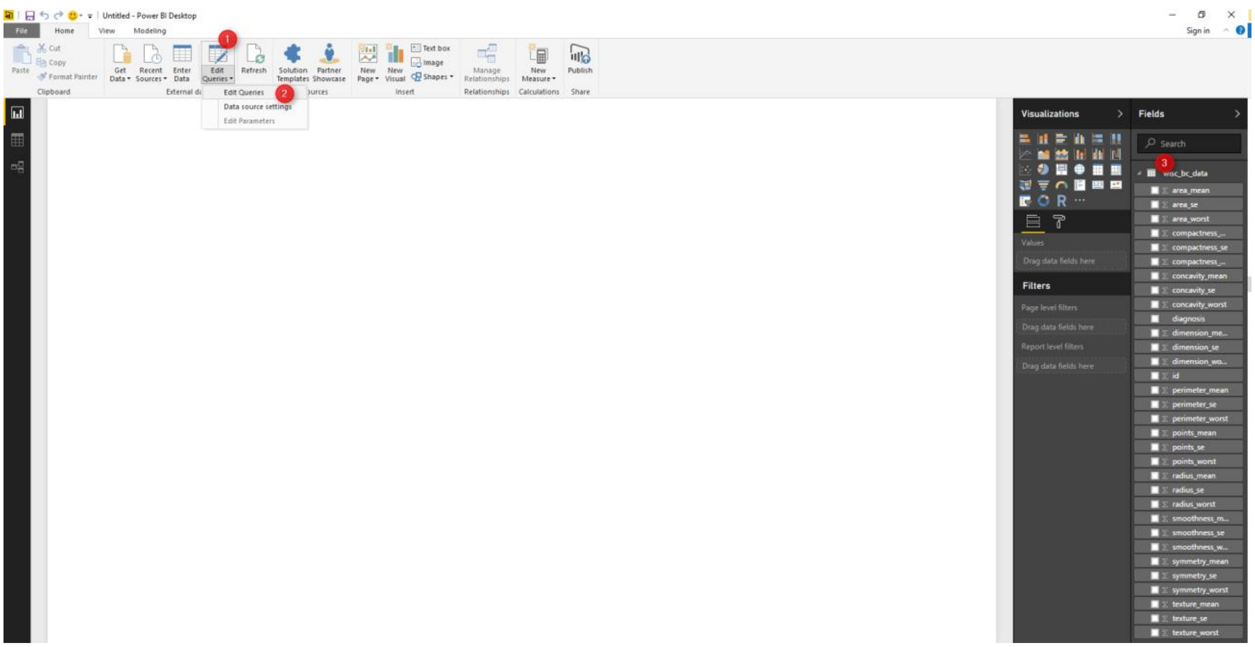
If you do not have Power BI Desktop, install it from https://powerbi.microsoft.com/en-us/

In power BI, Click on "Get Data" to import the data into Power BI, the data set . we have used the same dataset as in Part Two. The dataset contains the patient data such as : their diagnosis and laboratory results (31 columns).



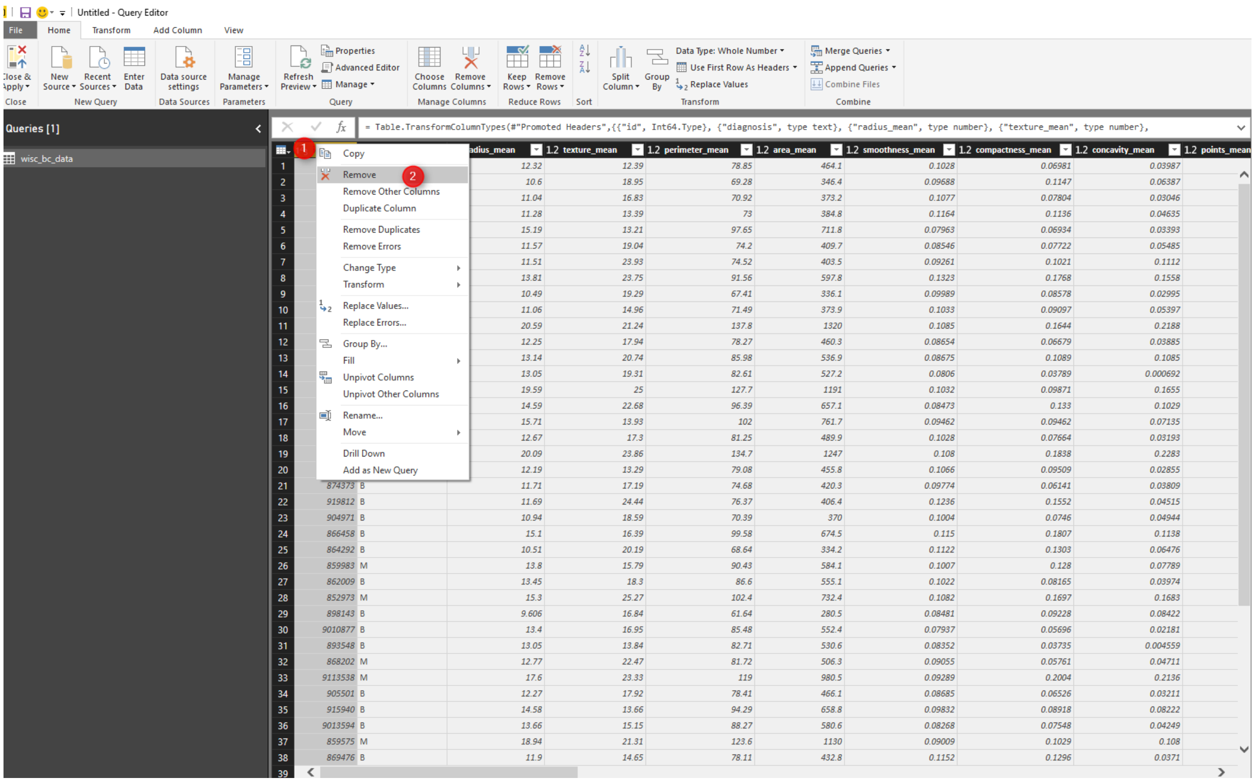
You will see (number 3) data in right side of the Power BI.

We want to clean the data first, hence we click on the "Edit Queries" in Power BI to do some data cleaning and also apply R scripts for KNN Model creation (Number 1 and 2).

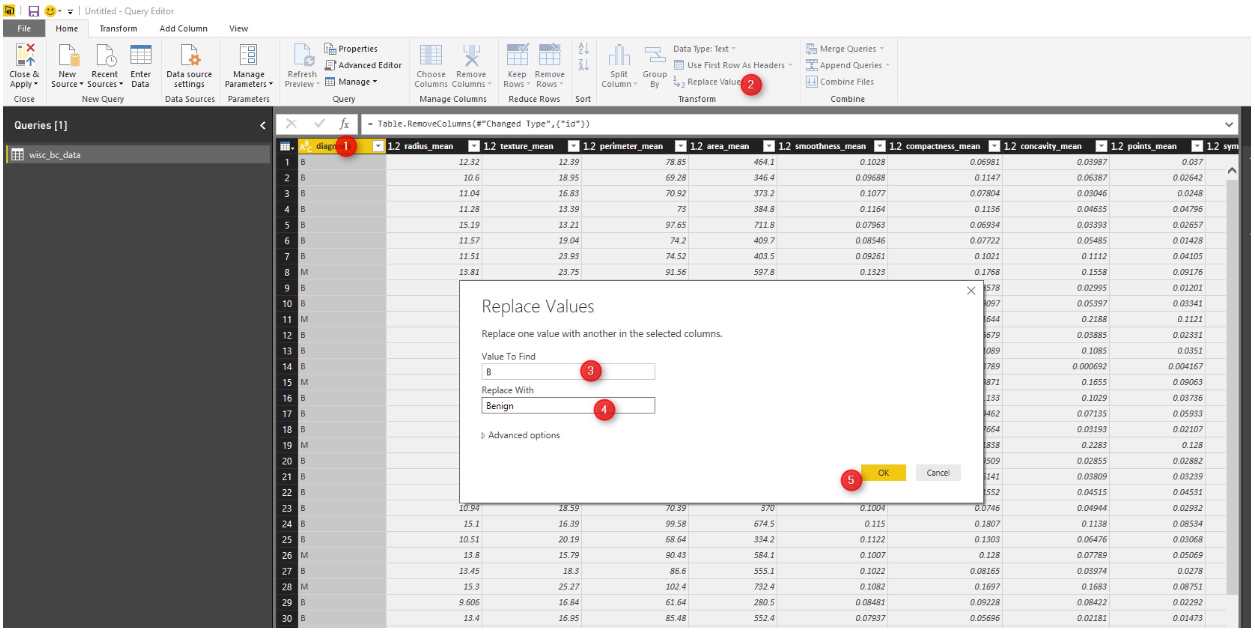


By clicking on the "Edit Query", we will see the "Query Editor" windows.

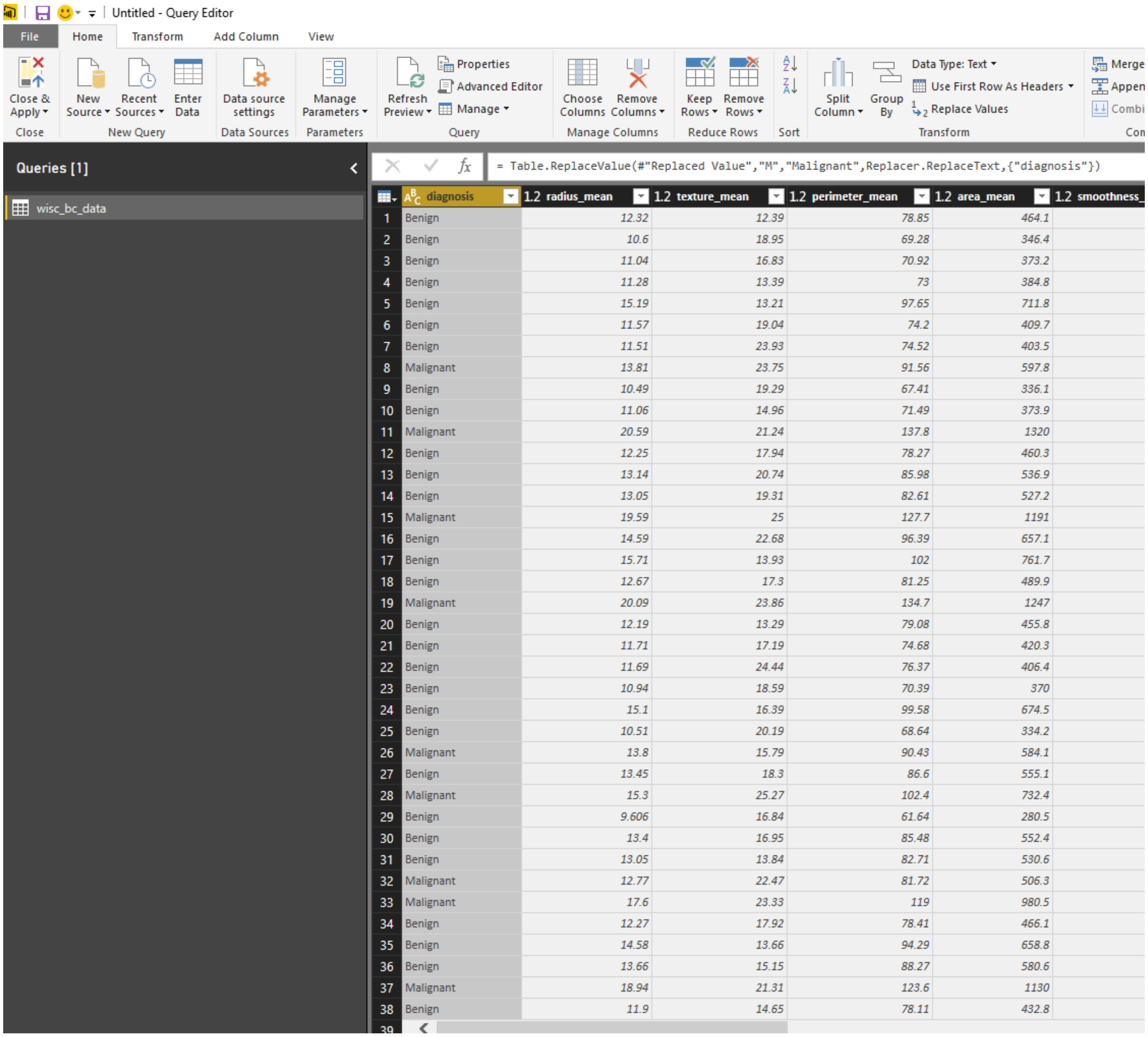
First of all, we want to remove the "ID" column. ID attributes does not have impact on prediction results. Hence, we right click on the "ID" column in power bi (number 1 and 2). and remove the ID column from data set.



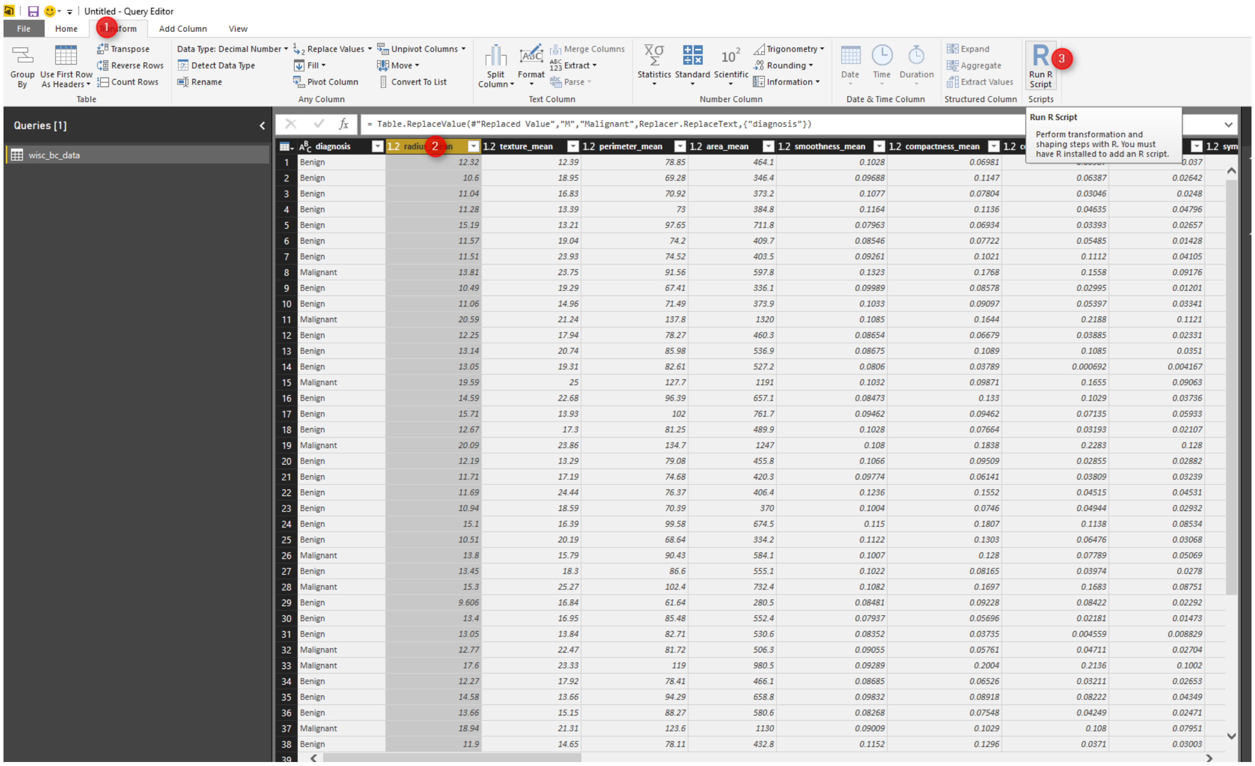
Another data cleaning approach is about replacing "B" value with "Benign " and "M" with "Malignant" in Diagnosis column. To do that, we right click on the diagnosis column (number 1). Then click on the "Replace Value" (number 2) in Transform tabe. In replace values place, for "Value To Find" type "B" (number 3) then "Replace With" the "Benign". Do the same for Malignant.



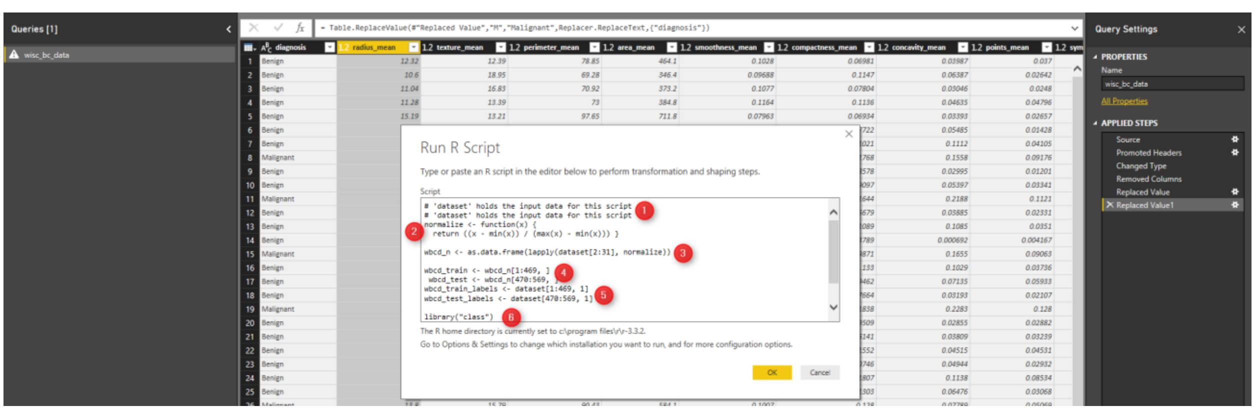
The result of the applied query will be look like below picture:



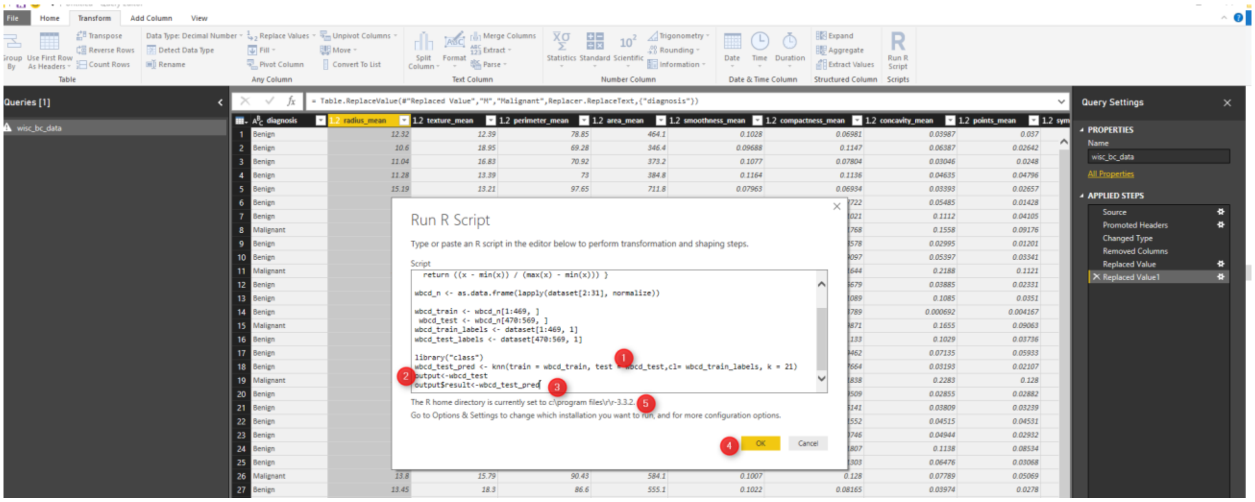
Another data cleaning is data normalization. Normalization has been explained in [Part Two](). We want to convert all the measurement value to the same scale. Hence we click on “Transform” tab, then in transform tab click on the data set (any numeric column). In this step we are using R scripts to normalize data. So, we click on the R scripts to perform normalization.



We write the same code we have in Part 2. the whole data (wis_bc_data) will be hold in "dataset". for doing normalization, we first write a function (number 1) and the function will be store in"normalize" variable. Then we apply normalize function to dataset. (number 3). We want to apply the function on numeric data not text (diagnosis column) hence, we refer to dataset[2:31] that means apply function on column 2 to column 31. the result of the function is data frame that will be sored in "wbcd_n" variable.

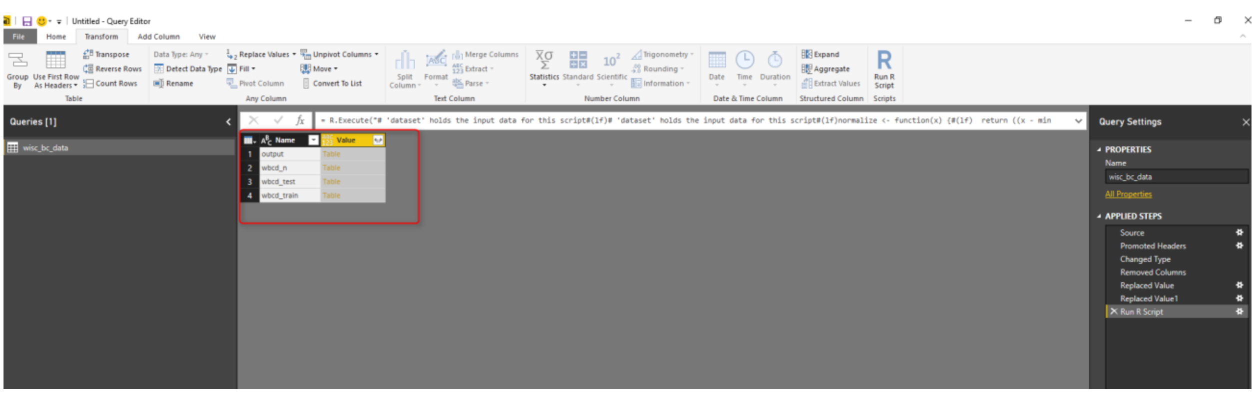


In next step, we are going to create test and train data set (Part 2). In this example, we going to put aside row number 1 to 469 for training and creating model and from row number 470 to 569 for testing the model. Finally, data is ready. Now we are able to train our model and create KNN algorithm.

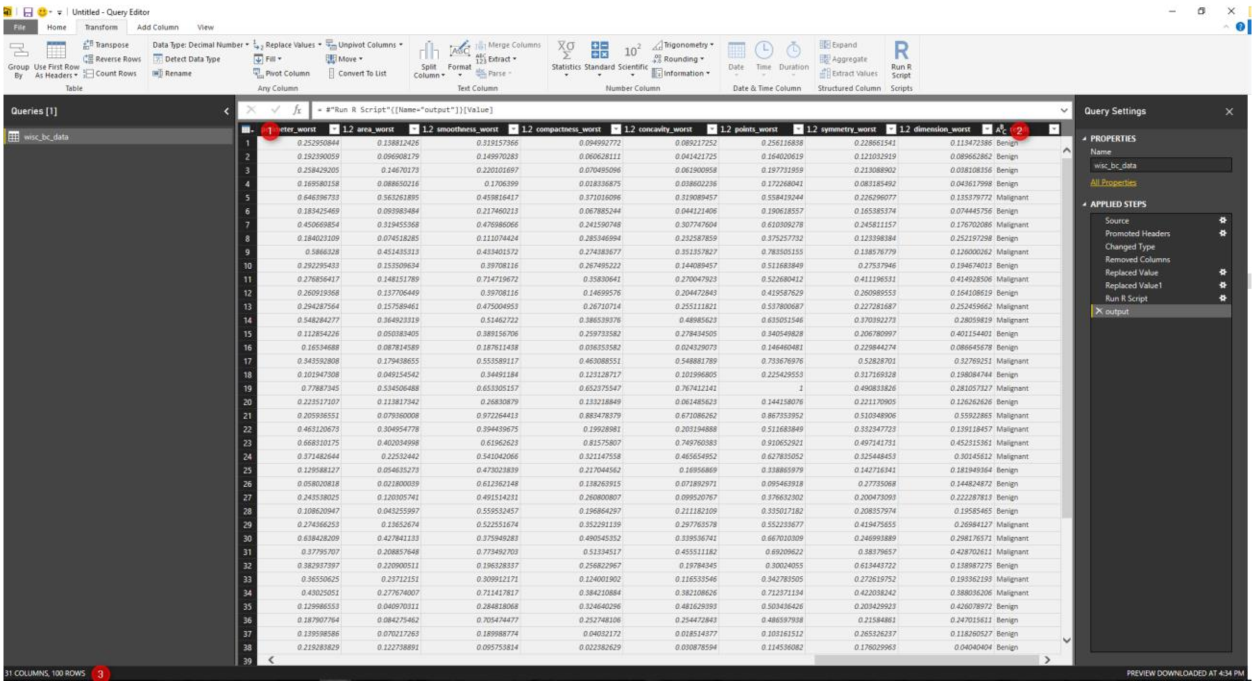


We already installed package “Class” in R studio. Now we are able to call function KNN to predict the patient diagnosis. KNN function accept the training data set and test data set as second arguments. Moreover the prediction label is also needed for result. We want to use KNN based on the discussion on Part 1, to identify the number K (K nearest Neighbor), we should calculate the square root of observation. Here for 469 observation the K is 21. the result is “wbcd_test_pred” holds the result of the KNN prediction. however, we want to have the result beside the real data so we store the test data set in “output” variable, then add the separate column to store the prediction result (number 3). Then click on “OK” to apply the R scripts on data.

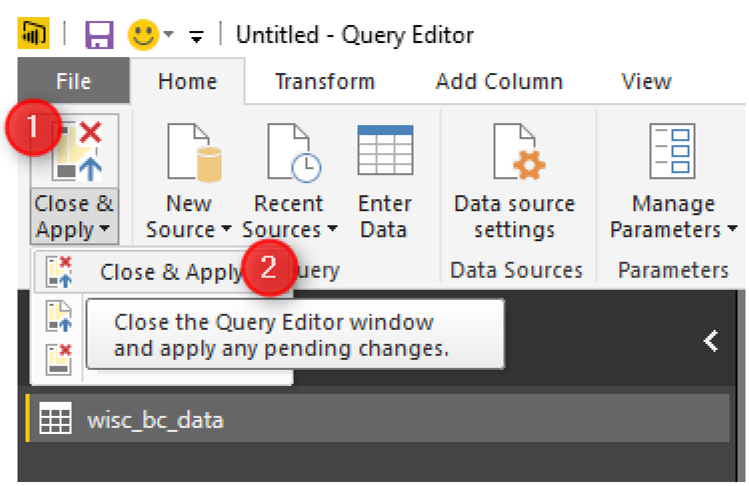
We will see below output:



For each output (Data frame in R scripts), we will have a table value. we store the final result in “output” in our R scripts, so we click on the Value (table) in front of the “output” name (see above picture). The result will be similar to below picture. In output, we will have the “Test” dataset and in the last column, we will have the prediction results. If you look at the left bottom side, you will see 100 rows that are the number of test case we have.



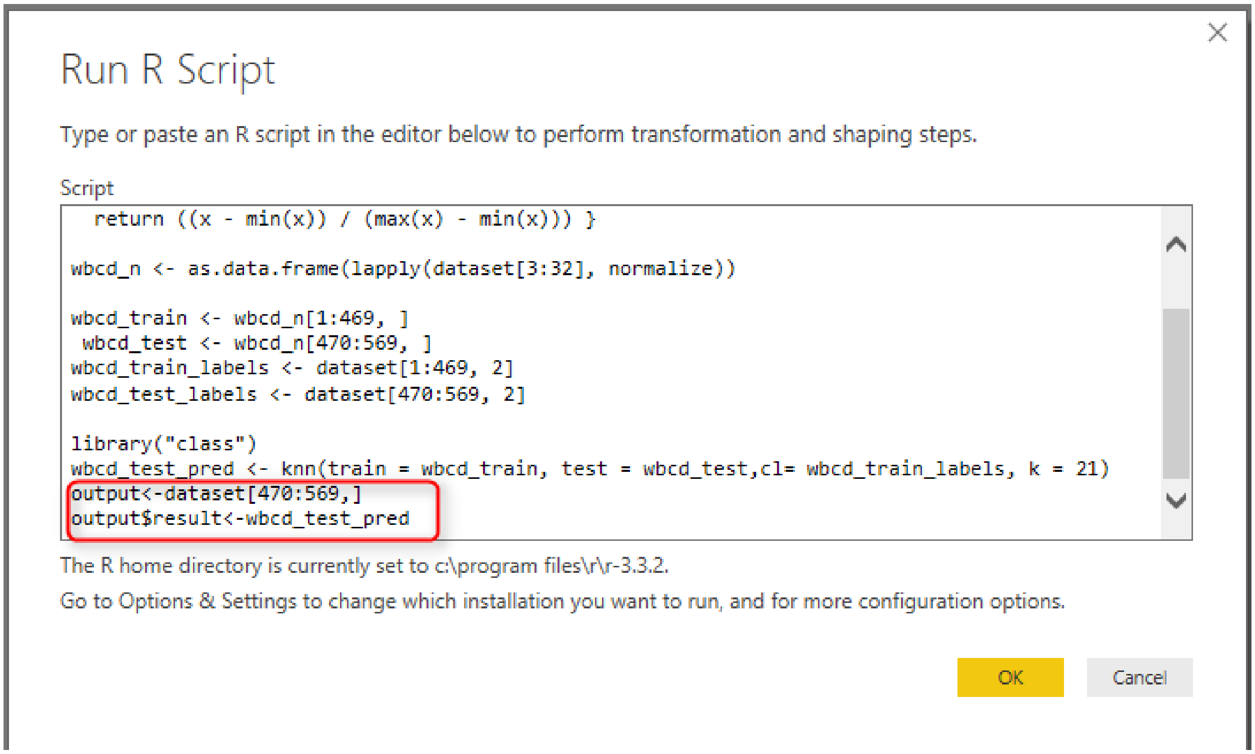
After R transformation, just click on the "Close&Apply" to see the result in visualization.



However, sometimes you want also see the patient ID in result data set. So what we do is to change the R code, as below :

output<-dataset[470:569]

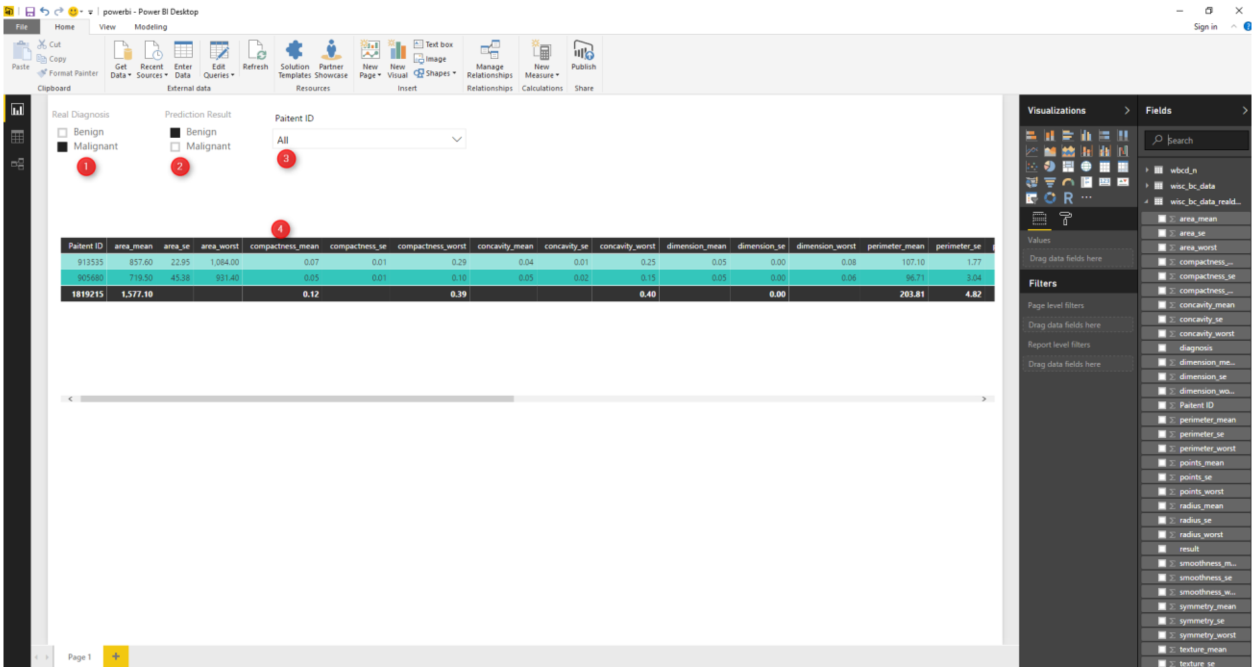
That means, we just get the whole data set from row 470 to 569. This data set is original one that contains the patient ID.



Hence, just close and apply the query.

In the below visualization, I have shown the result of prediction and real data. In current dataset we have patient ID, real data about the doctor's diagnosis and the predicted diagnosis by KNN. In Real Diagnosis filter (picture:number 1) we are able to select patient that "Malignant" and in second filter we are able to choose the predicted result "Benign". this will show us the cases that prediction could not work properly.

we have 2 cases that the KNN predict in wrong way. as you see in below picture.



In the next series, I will talk about the other algorithms such as Neural Network (Deep Learning), Time series, and Decision Tree.

Moreover, there is an upcoming series on Azure ML which will be start soon.

11-Make Business Decisions: Market Basket Analysis Part 1

Published Date : February 14, 2017

Market Basket analysis (Associative rules), has been used for finding the purchasing customer behaviour in shops & stores to show the related items that were sold together. This approach is not just used for marketing related products, but also for finding rules in health care, policies, events management so on and so forth.
In this Post I will explain how Market Basket Analysis can be used, how to write it in R and come up with good rules.

In next post I will show how to write Associative Rules in Power BI using R scripts.

What is Market Basket Analysis (Concepts)?

This analysis examines customer purchased behavior. For instance, it suggests that customers often purchase shampoo and conditioner together. From marketing perspective , it helps promote Shampoo and lead customers to purchase conditioner as well. From sales perspective, by putting Shampoo beside Conditioner on the shelf, there is a higher chance that people purchase both.

Association rules is another name for Market Basket analysis. Association rules are in the form if X then Y. For example: 60% of those who buy life insurance also buy health insurance. In another example, 80% of those who buy books on-line also buy music on-line. Also, 50% of those who have high blood pressure and are overweight have high cholesterol [2]. Other examples such as;

- Searching for interesting and frequently occurring patterns of DNA and protein sequences in cancer data.
- Finding patterns of purchases or medical claims that occur in combination with fraudulent credit card or insurance use.
- Identifying combinations of behavior that precede customers dropping their cellular phone service or upgrading their cable television package.

You will see these rules are not just about the shopping, but also can be applied in health care, insurance, and so forth.

Measuring rule interest – support and confidence

To create appropriate rules we should, it is important to identify the support and confidence measure. Support measure is: The support of an item set or rule measures how frequently it occurs in the data[2].

For instance, imagine we have below transaction items from a shopping store for last hours,

Customer 1: Salt, pepper, Blue cheese

Customer 2: Blue Cheese, Pasta, Pepper, tomato sauce

Customer 3: Salt, Blue Cheese, Pepperoni, Bacon, egg

Customer 4: water, Pepper, Egg, Salt

we want to know how many times customer purchase pepper and salt together the support will be : from out four main transactions (4 customers), 2 of them purchased salt and pepper together. so the support will be 2 divided by 4 (all number of transaction. 2/4 (0.50).

$$\text{support}(X) = \frac{\text{count}(X)}{N}$$

x: frequency of an item occurs

N: total Transaction

Another important measure is about the rule's confidence that is a measurement of its predictive power or accuracy.

for example we want to know what is the probability that people purchase Salt then they purchase Pepper or wise versa.

Confidence (Salt–>Pepper), we have to calculate the frequency of purchasing Salt (Support (Salt))

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X, Y)}{\text{support}(X)}$$

then we calculate the purchase frequency(Support) of both Salt and Pepper (we already calculated it).

then The Support (Salt, Peppers) should be divided by Support(Salt):

the Support for Purchasing Salt is 3 out of 4 (0.75)

Support for Purchasing Salt and Pepper is 0.5

by dividing these two number (0.5 Divide to 0.75) we will have: 0.6

So we can say in 60% of time (based on our dataset), if Customer purchase Sale, they will Purchase Pepper.
so we have a rule that "if Customer purchase Salt–>then with 60% of time they will purchase peppers"

However, this percentage is not valid for "Purchasing Pepper then Salt"

To calculate this, we should first calculate the Support for Pepper which is 0.75

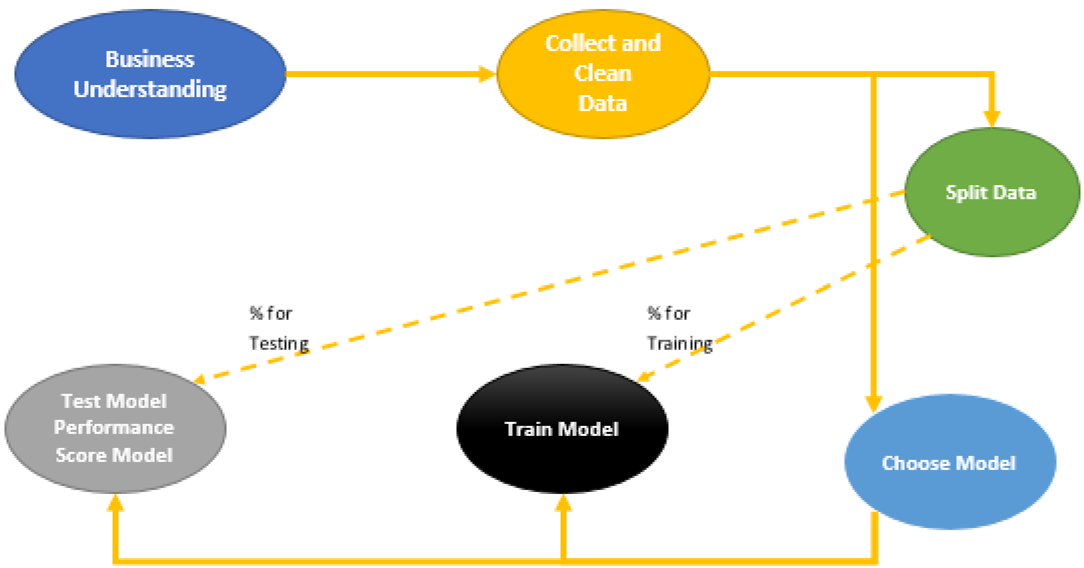
then, Divide the Support (Pepper and Sale) to Support (Pepper)=1

So we have below Rules:

"People who purchase Pepper–> will purchase Salt in 100% of time"

Market Basket Analysis in R

for doing the market basket analysis we follow below diagram:



First step- is to identity the business problem: in our case is to identify the shopping list items that have been purchased most together by our customers.

Second Step- Gather data and find relevant attributes. we have a data set about 169 customers who purchased some item from grocery stores. The collected data should be formatted.

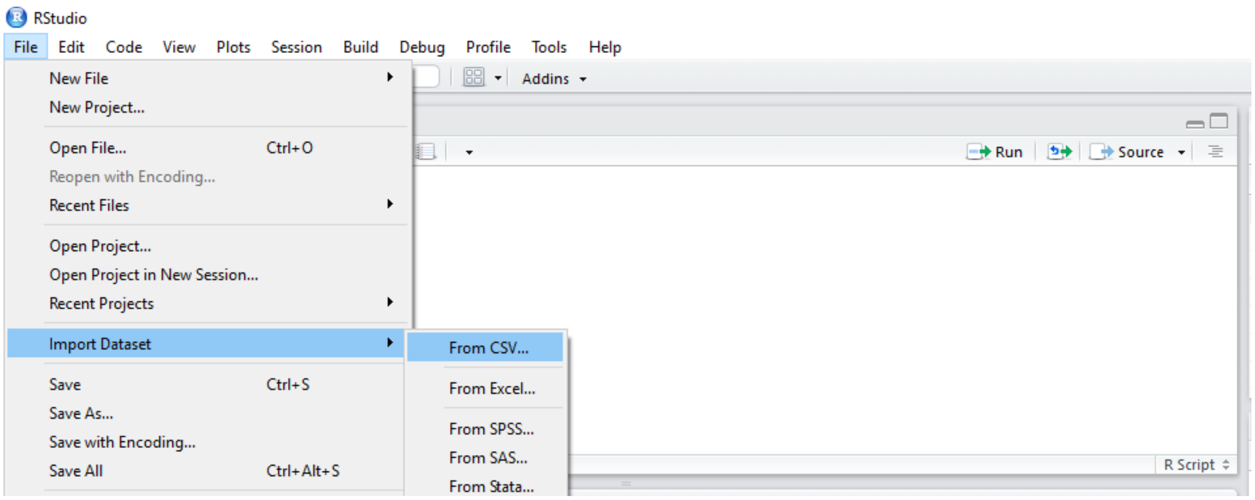
Third Step- Select the machine learning algorithm regarding to the business problems and available data. In this model because we are going find the rules in customer shopping behaviour that help us to market more on those items. Hence, we choose Associative Rules.

After selection of algorithm, we have to pass some percentage of data (more than 70%) for learning. Algorithm will learns from current Data. The remaining part of data have been used for testing the accuracy of algorithms.

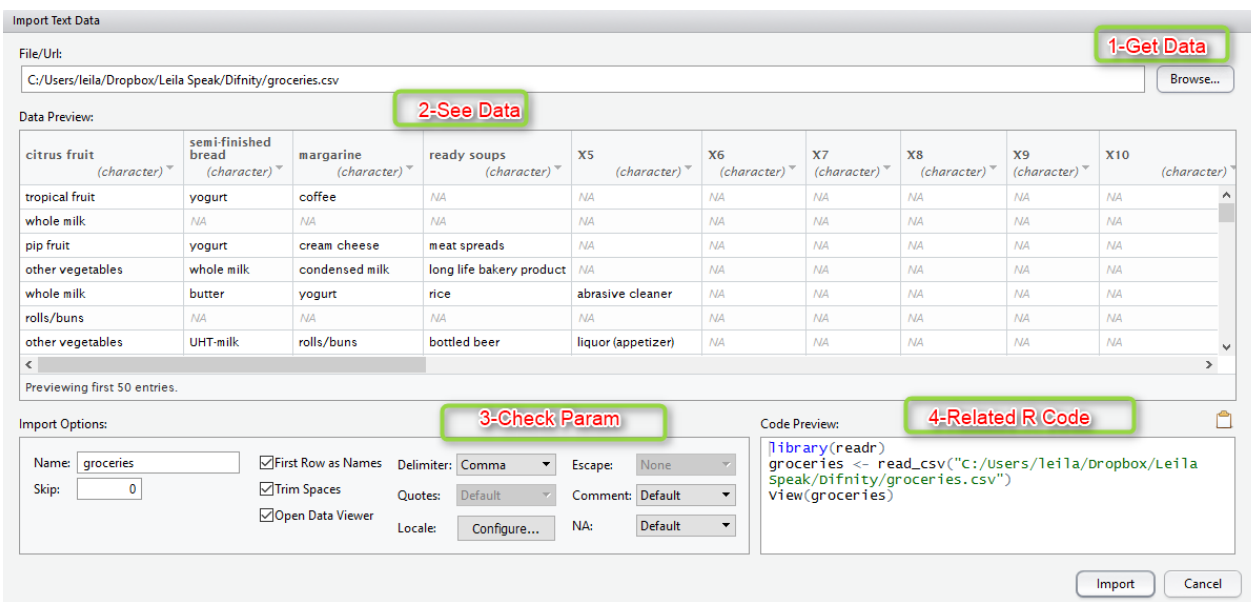
Step 1- Get Data, Clean Data and Explore Data

We have some data about the Groceries transaction in a shopping store

The first step to do machine learning in R is to import the data set into R.



Now we should brows our CSV file.



The first five rows of the raw grocery.csv file are as follows:
1-citrus fruit,semi-finished bread,margarine,ready soups
2-tropical fruit,yogurt,coffee
3-whole milk
4-pip fruit,yogurt,cream cheese,meat spreads
5- other vegetables,whole milk,condensed milk,long life bakery product

After importing Data, we will see that the imported dataset is not in form of tables, there is no column name

	citrus fruit	semi-finished bread	margarine	ready soups	X5	X6	X7
1	tropical fruit	yogurt	coffee	NA	NA	NA	NA
2	whole milk	NA	NA	NA	NA	NA	NA
3	pip fruit	yogurt	cream cheese	meat spreads	NA	NA	NA
4	other vegetables	whole milk	condensed milk	long life bakery product	NA	NA	NA
5	whole milk	butter	yogurt	rice	abrasive cleaner	NA	NA
6	rolls/buns	NA	NA	NA	NA	NA	NA
7	other vegetables	UHT-milk	rolls/buns	bottled beer	liquor (appetizer)	NA	NA

The first step is to create a sparse matrix from data in R (See the Sparse Matrix explanation in https://en.wikipedia.org/wiki/Sparse_matrix).

Each row in the sparse matrix indicates a transaction. The sparse matrix has a column (that is, feature) for every item that could possibly appear in someone's shopping bag. Since there are 169 different items in our grocery store data, our sparse matrix will contain 169 columns [2].

To create a data set that able to do associative rules, we have install "arules".

the below codes help us to that :

```
install.packages("arules")
library(arules)
groceries <- read.transactions("File address", sep = ",")
```

So to see the summary of data we run Summary(groceries)

```
> summary(groceries)
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.02609146

most frequent items:
      whole milk other vegetables       rolls/buns             soda           yogurt          (Other)
            2513             1903             1809             1715             1372            34055

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46   29   14   14    9   11    4
  23   24   26   27   28   29   32
   6    1    1    1    1    3    1
```

9835 Customer and 169 items

Now we are going to inspect the first five rows in groceries data frame. using "inspect" function to see the first customer's transactions:

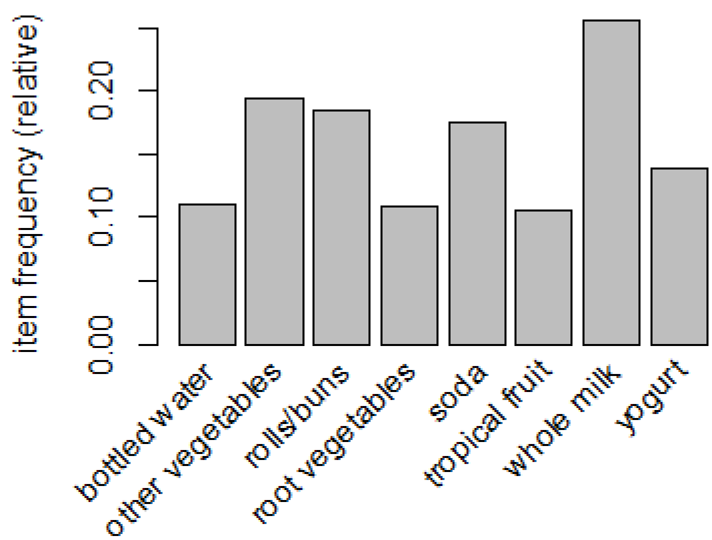
```
> inspect(groceries[1:5]) #first five transactions can be viewed
    items
[1] {citrus fruit,margarine,ready soups,semi-finished bread}
[2] {coffee,tropical fruit,yogurt}
[3] {whole milk}
[4] {cream cheese,meat spreads,pip fruit,yogurt}
[5] {condensed milk,long life bakery product,other vegetables,whole milk}
```

Now we want to draw a bar chart that depict the proportion of transactions containing certain items.

```
itemFrequencyPlot(groceries, support = 0.1)
```

itemfrequencyPlot is a function that draw a bar chart based on the item frequency in transaction list. The first parameter is our dataset, the second parameter is support which is a numeric value. in this example we set the support as 0.1, which means only display items which have a support of at least 0.1.

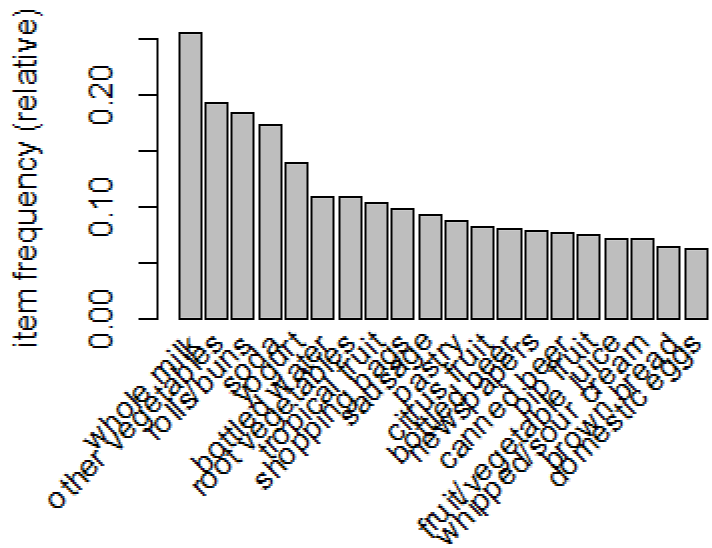
The result is like below:



Also if we are interested to just see the top 20 items that have been purchased more, we can used the same function but with different inputs as below:

```
itemFrequencyPlot(groceries, topN =20)
```

the topN arguments is set to 20 that means just main 20 items.



as you can see in above diagram, whole milk have been purchased more, then Vegetables and so forth.

Step 2- Create Market Basket Analysis Model

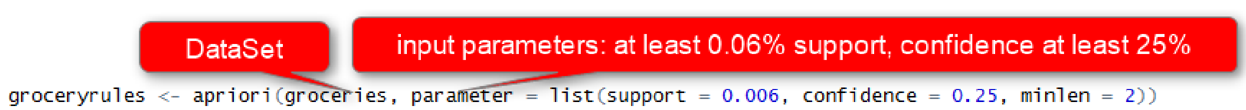
We are going to use apriori algorithm in R.

install.packages("apriori")

now we able to call the

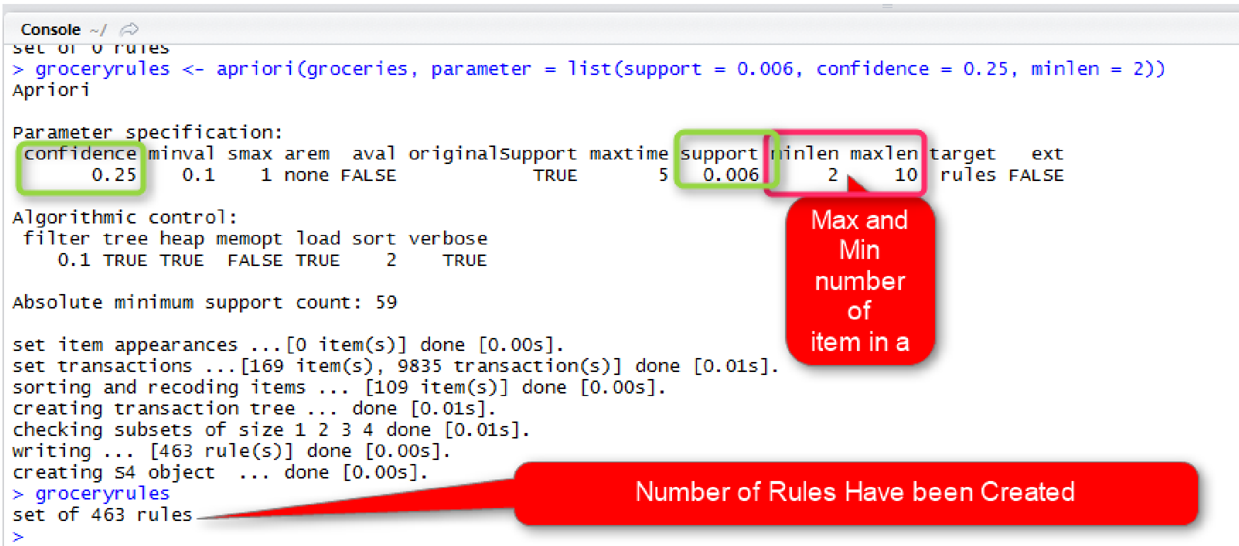
groceryrules <- apriori(groceries, parameter = list(support = 0.006, confidence = 0.25, minlen = 2))

we call function *apriori()*

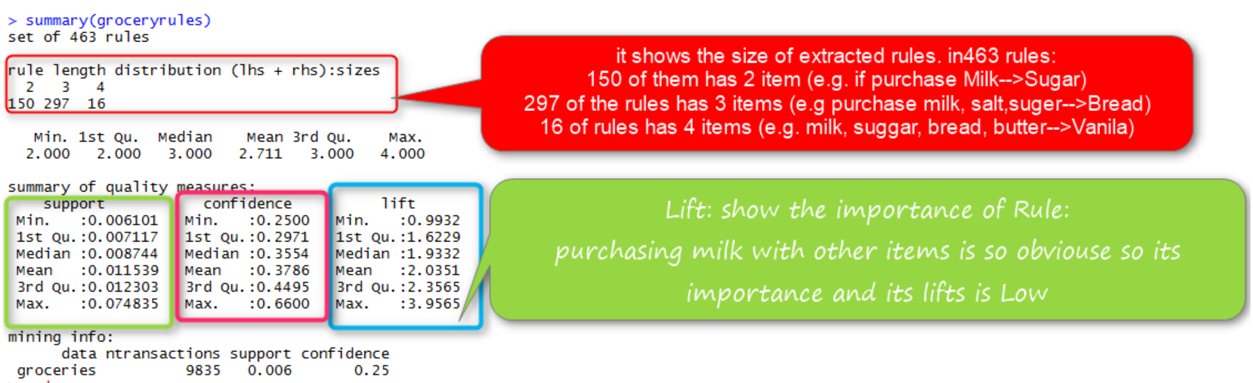


it gets groceries data set as first input, the list of parameters (such as minimum supports which is 0.06%, the confidence that is 25% and the minimum length of the rule 2) as second inputs.

the result of running this code will be



in above picture, we got about 463 rules



by applying summary function we will have summary of Support, Confidence and Lift.

Lift is another measure that shows the importance of the a rule, that means how much we should pay attention to a rule.

Now, we are going to fetch the 20 most important rules by using *inspect()* function

```
> inspect(sort(groceryrules, by = "lift")[1:20])
     lhs                                            rhs                  support     confidence lift
[1]  {herbs}                                     => {root vegetables}    0.007015760 0.4312500  3.956477
[2]  {berries}                                   => {whipped/sour cream} 0.009049314 0.2721713  3.796886
[3]  {other vegetables,tropical fruit,whole milk} => {root vegetables}   0.007015760 0.4107143  3.768074
[4]  {beef,other vegetables}                     => {root vegetables}    0.007930859 0.4020619  3.688692
[5]  {other vegetables,tropical fruit}           => {pip fruit}          0.009456024 0.2634561  3.482649
[6]  {beef,whole milk}                           => {root vegetables}    0.008032537 0.3779904  3.467851
[7]  {other vegetables,pip fruit}                => {tropical fruit}     0.009456024 0.3618677  3.448613
[8]  {pip fruit,yogurt}                          => {tropical fruit}     0.006405694 0.3559322  3.392048
[9]  {citrus fruit,other vegetables}             => {root vegetables}    0.010371124 0.3591549  3.295045
[10] {other vegetables,whole milk,yogurt}        => {tropical fruit}     0.007625826 0.3424658  3.263712
[11] {other vegetables,whole milk,yogurt}        => {root vegetables}    0.007829181 0.3515982  3.225716
[12] {tropical fruit,whipped/sour cream}         => {yogurt}             0.006202339 0.4485294  3.215224
[13] {other vegetables,tropical fruit,whole milk} => {yogurt}            0.007625826 0.4464286  3.200164
[14] {other vegetables,rolls/buns,whole milk}    => {root vegetables}    0.006202339 0.3465909  3.179778
[15] {frozen vegetables,other vegetables}        => {root vegetables}    0.006100661 0.3428571  3.145522
[16] {other vegetables,tropical fruit}           => {root vegetables}    0.012302999 0.3427762  3.144780
[17] {sliced cheese}                             => {sausage}            0.007015760 0.2863071  3.047435
[18] {other vegetables,tropical fruit}           => {citrus fruit}       0.009049314 0.2521246  3.046248
[19] {beef}                                      => {root vegetables}    0.017386884 0.3313953  3.040367
[20] {citrus fruit,root vegetables}              => {other vegetables}   0.010371124 0.5862069  3.029608
```

Ee employ inspect function to fetch the 20 most important rules. As you can see the most important rule is that people who purchase Herbs—>will purchase Root Vegetable

as any machine Learning algorithm, the first step is to Train data.

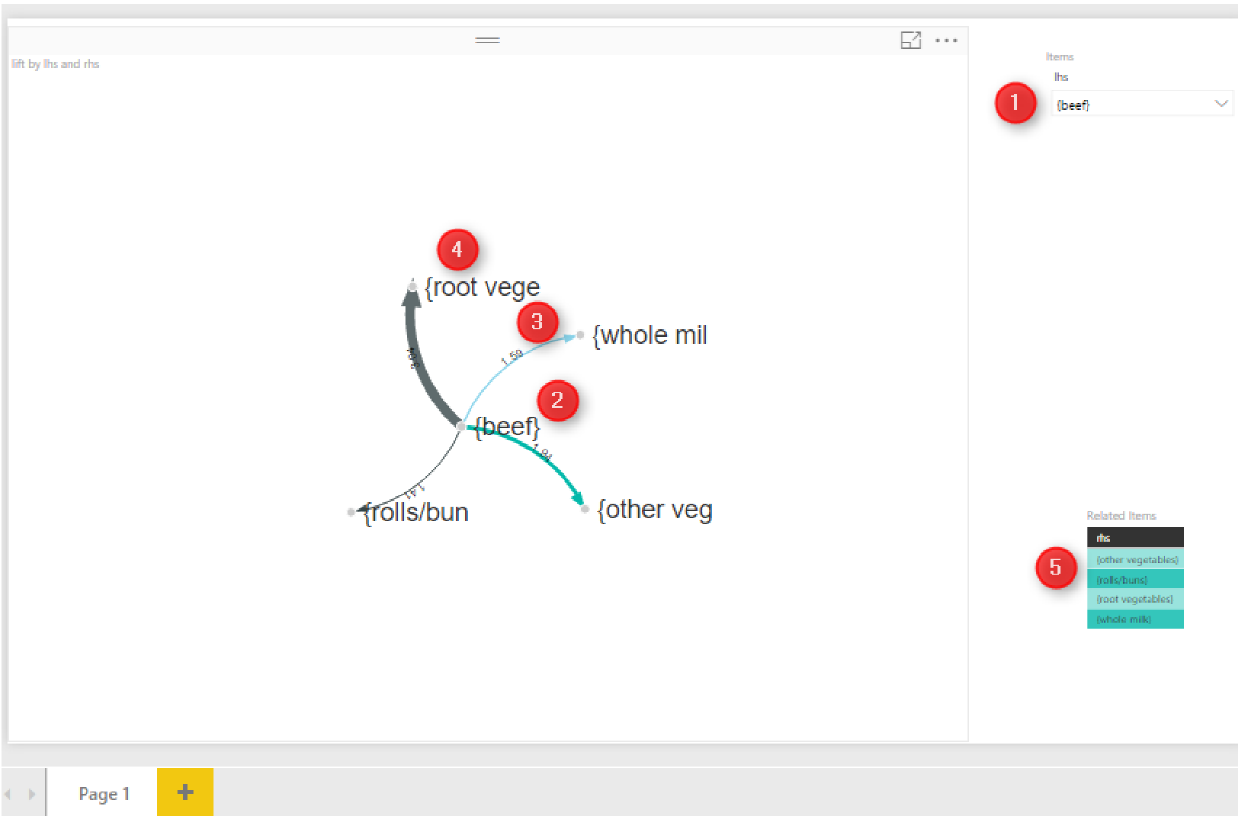
[1].
http://www.ms.unimelb.edu.au/~odj/Teaching/dm/1%20Association%20Rules%2008.pdf

[2].Machine Learning with R,Brett Lantz, Packt Publishing,2015.

Stay Tuned for the Next Part.

12-Make Business Decisions: Market Basket Analysis Part 2

Published Date : March 21, 2017



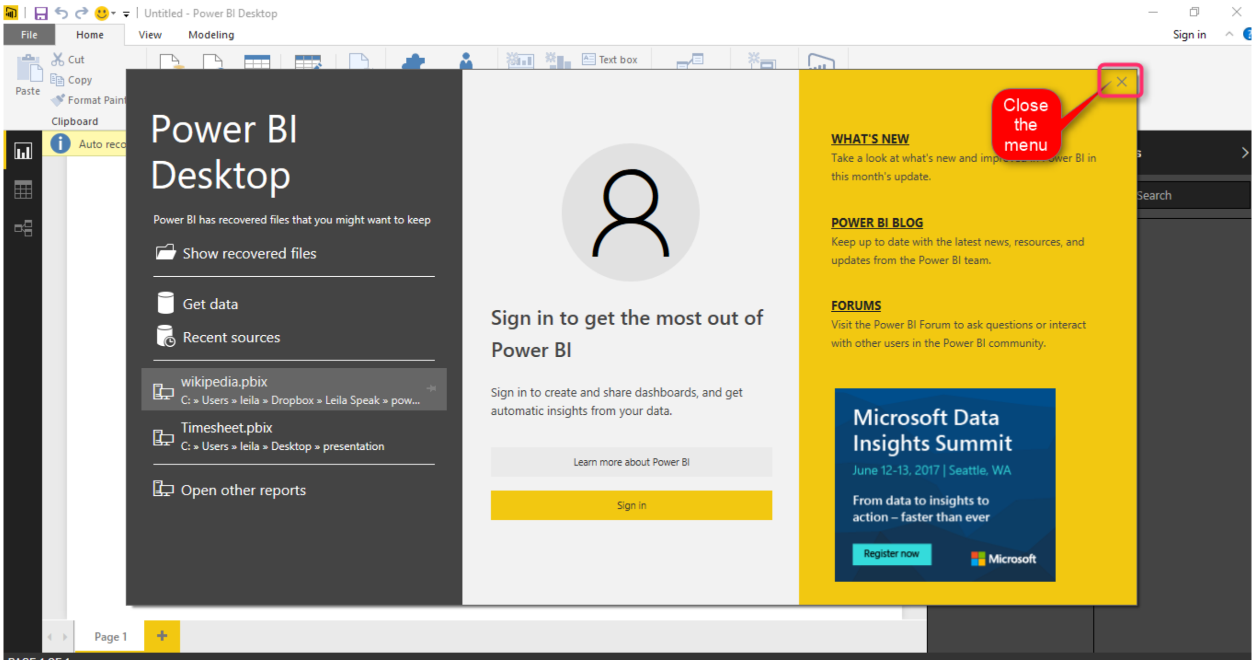
In the Part one I have explained the main concepts of Market basket analysis (associative Rules) and how to write the code in R studio. In this post I will explained the process of doing market basket analysis in Power BI.

for doing this post I have used the data set from [1].

Power BI Desktop, is a self service BI tool. you can download it from below link;

https://powerbi.microsoft.com/en-us/

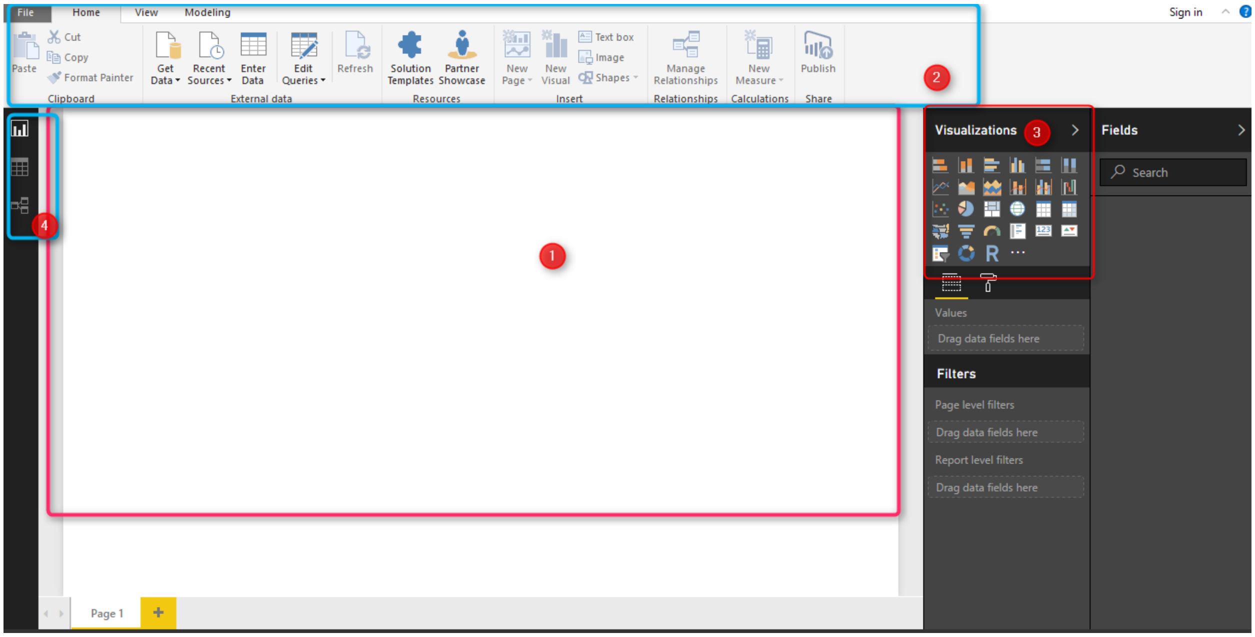
to do the market basket analysis, I first create a new Power BI file



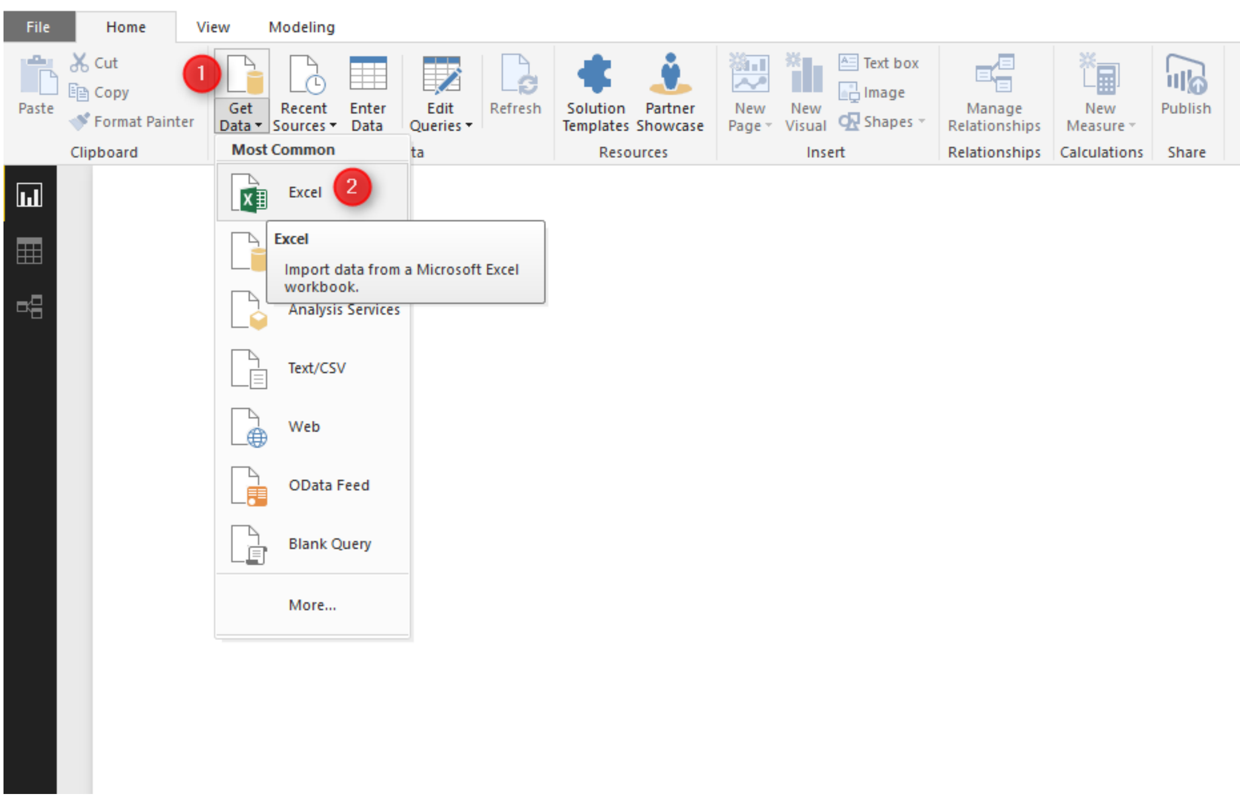
Power BI is a great tools for visualization and cleaning data, most of data wrangling and data cleaning like remove missing variables, replace values, remove columns , etc. can happen here.

The middle area is for creating reports (Number 1). At the top, the main tools for creating reports, data wrangling, and so forth is located in number 2.

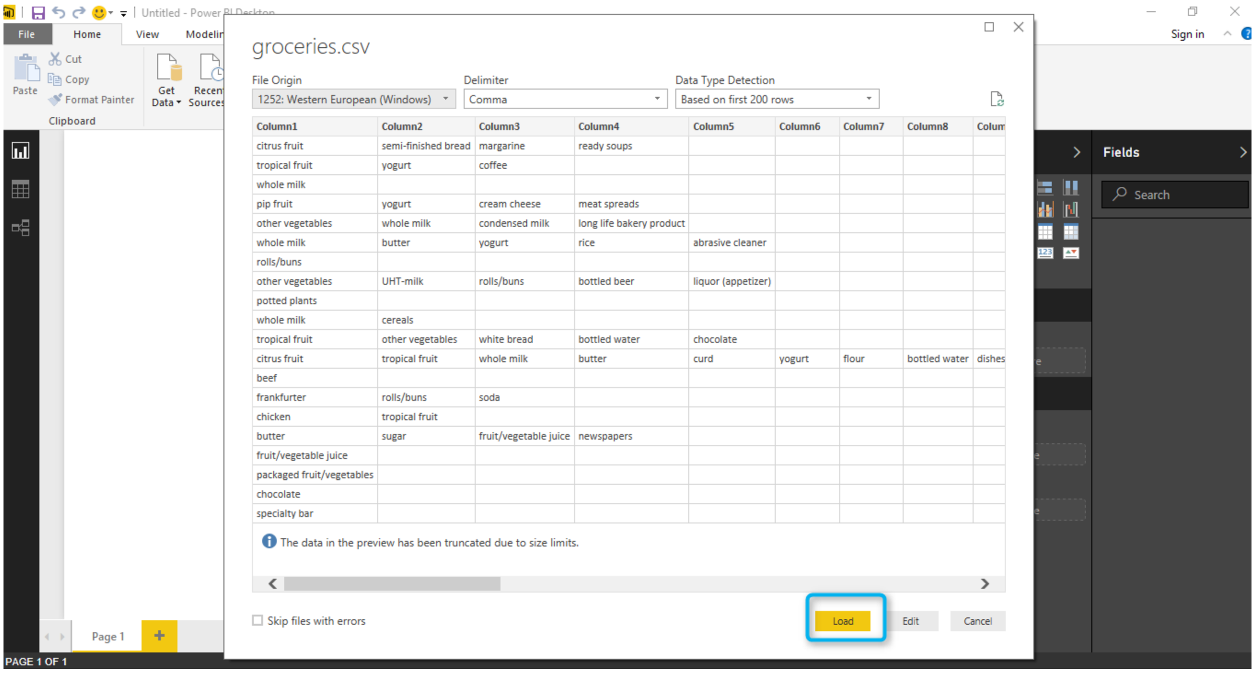
Moreover, the report elements like bar chart pie chart and so on has been shown in right side (number 3). finally, we able to see the relationship between tables and data in left side (number 4)



We get data from excel (local PC), so click on Get data in top menu and choose Excel from sources, as in picture below.



After loading the data set, now we can see the transaction shopping data for each customers (see below pictures).



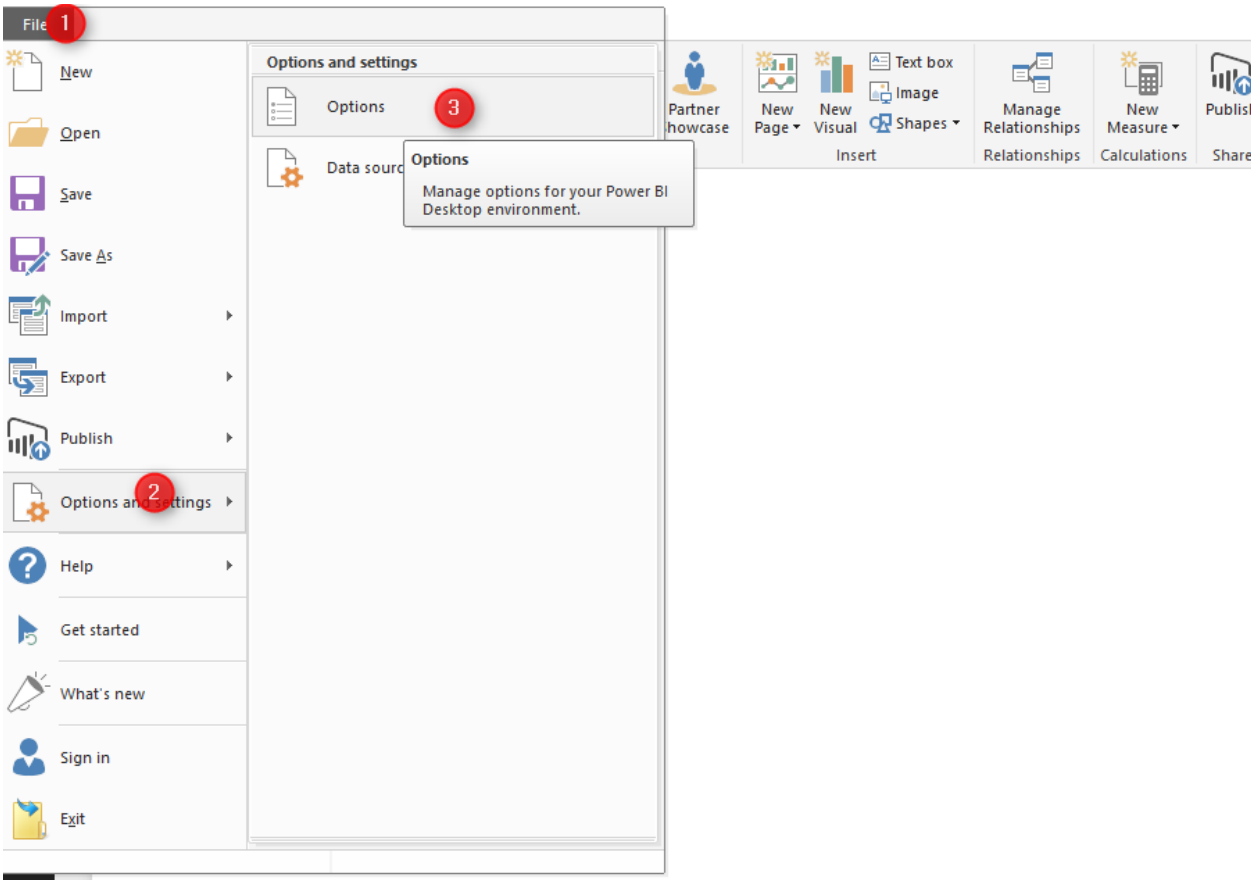
So, we click on load option to get data from local pc into power bi.

Then we want to do Market Basket analysis on data to get more insight out of it. In power bi, it possible to write R code!

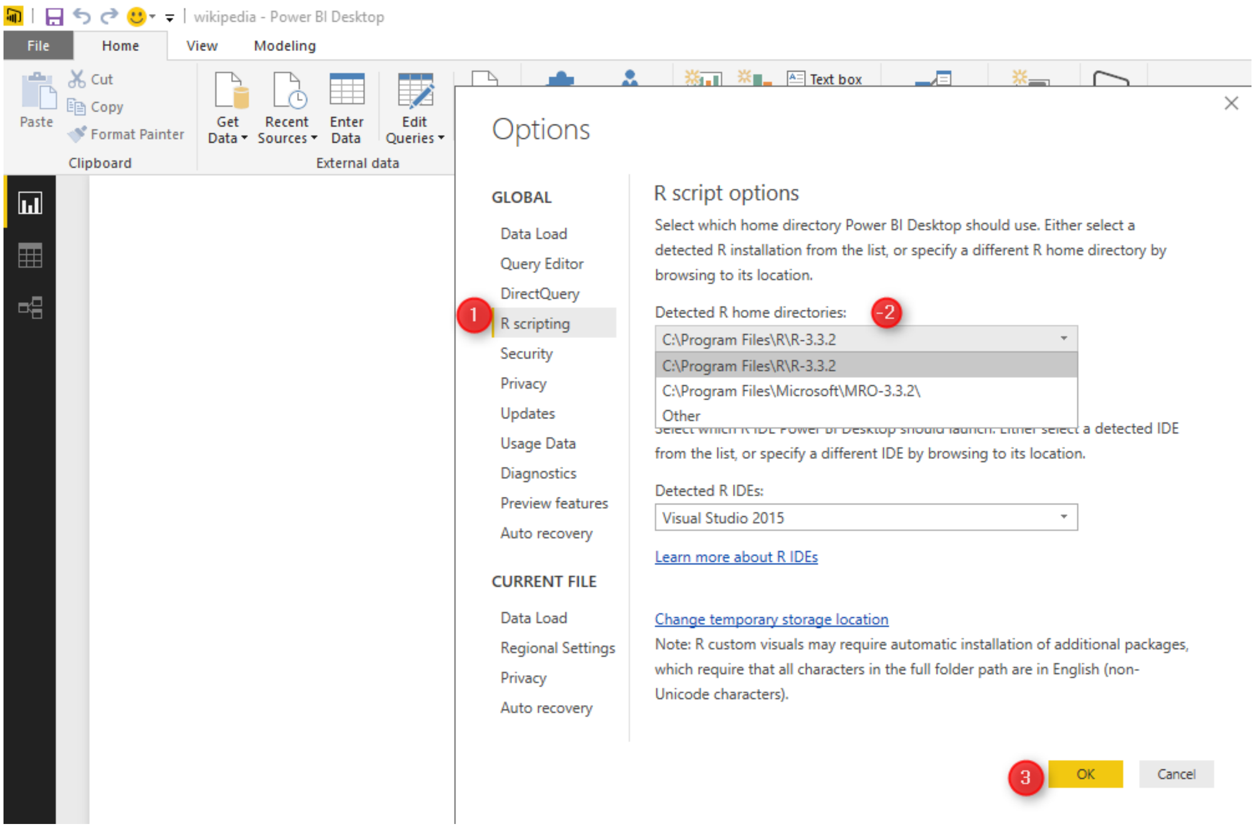
First you should install a version of R in your pc. I already installed Microsoft R Open 3.3.2. from

https://mran.microsoft.com/open/

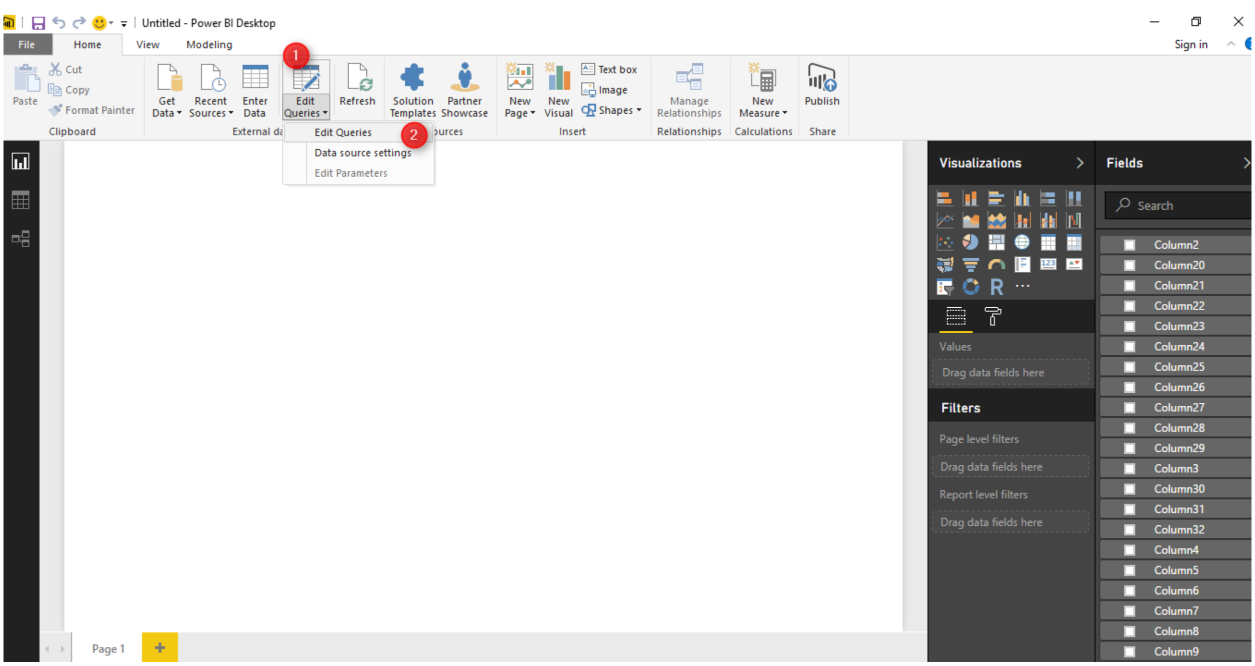
After installing R in your machine, in power BI you should specify the R version. to do that, in Power BI, click on “File”, then “Options” (below picture)



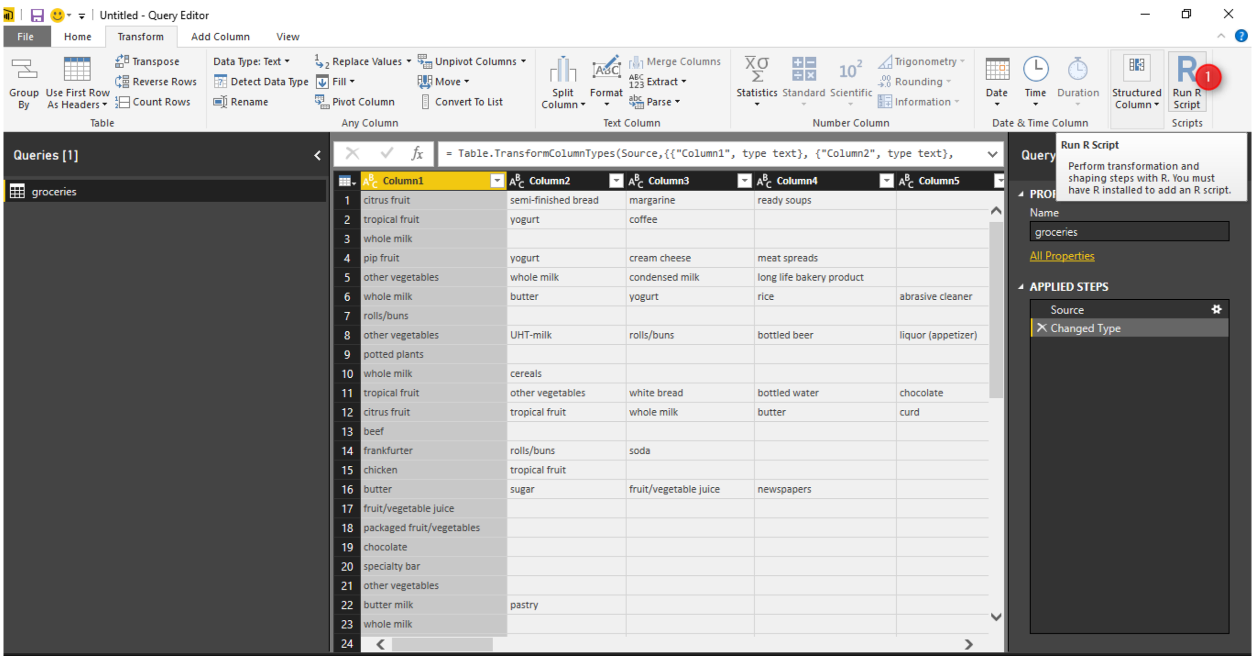
Then in “Global”, find he “R scripting”. In R scripting, Power Bi automatically detect the available R version in your PC. It is important that you select the R version that you already tested your R code there. as when we install a package in R, in Power BI that package is also become available, so there should be some connectivity between R version that you run your code and then one you select in power BI.



Now we want to write Market Basket analysis code in Power BI. To do this we have to click on "Edit Query" and then choose " Edit Queries" from there.



After selecting “edit query” , you will see the query editor environment. in top right, there is a “R transformation” icon.



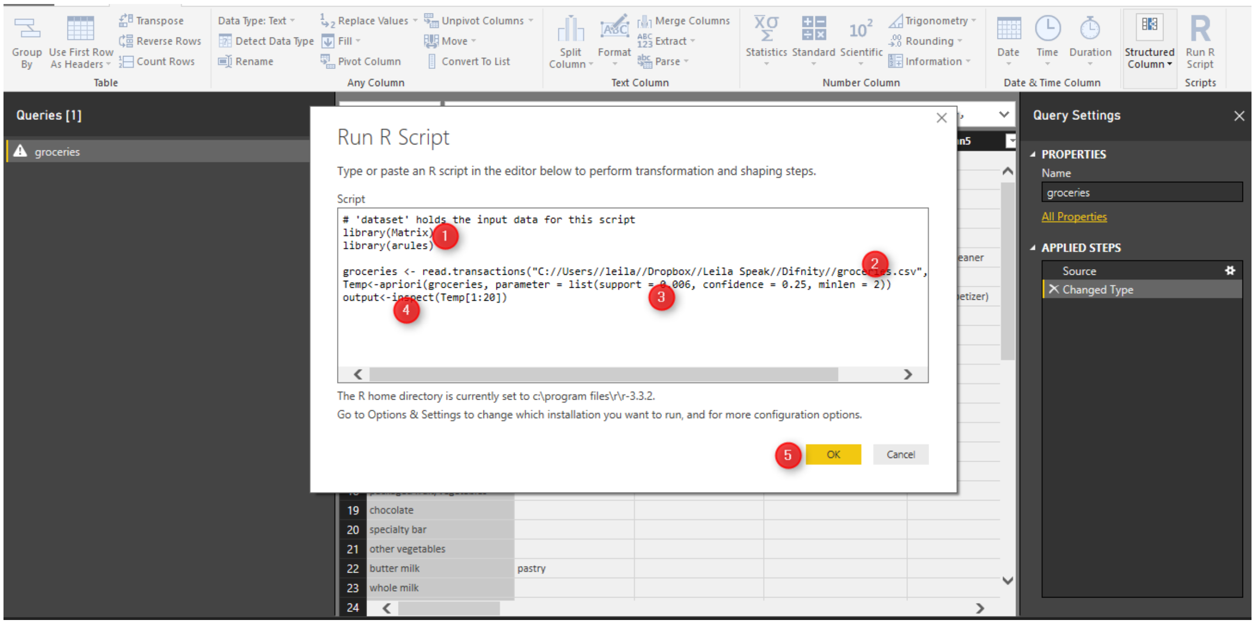
By clicking on the “R transformation”, R editor will show up as a new windows will show up, you can past your code here. There are couple of things that you should consider.

1. there is an error message handling but always recommended to run and be sure your code work in R studio first (in our example we already tested it in Part 1).

2. all data is holding in variable “dataset”.

3. you do not need to write “install.packages” to get packages here, but you should first install required packages into your R editor and here just call “library(package name)”

We have below editor



I need two main libraries to do market basket analysis: “Matrix” and “arules”, the two lines of code to have these libraries are:

library(Matrix)
library(arules)

As the data is not in format of transaction I have to reload the data from my PC again to make them as transaction type by writing below code

groceries <- read.transactions("C://Users//leila//Dropbox//Leila Speak//Difnity//groceries.csv", sep = ",")

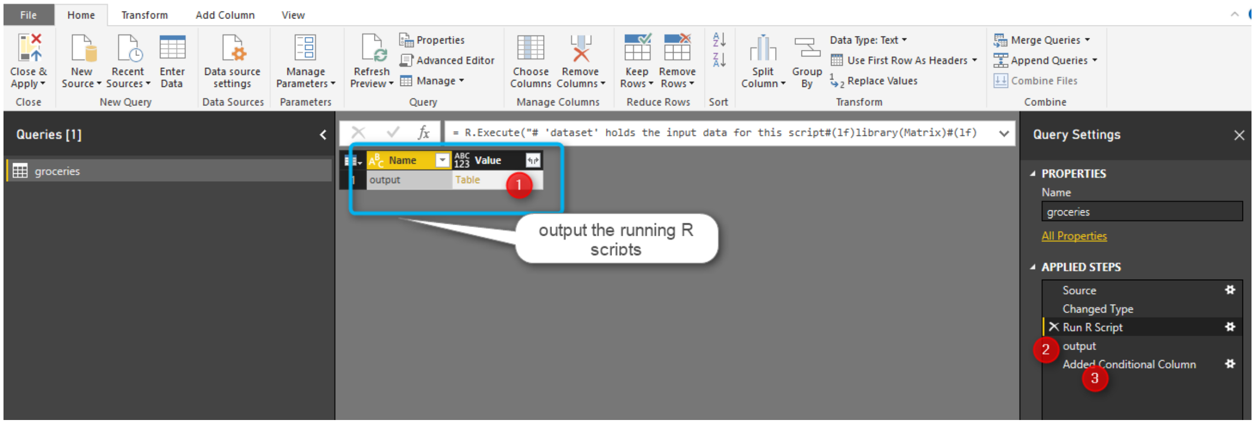
Then, call “apriori” function to find the rules in customers shopping behaviour. apriori gets the dataset “groceries” as input, also it accepts the parameters like support ,confidence , and minlen. the output of the function will be in “Temp” variable

Temp<-apriori(groceries, parameter = list(support = 0.006, confidence = 0.25, minlen = 2))

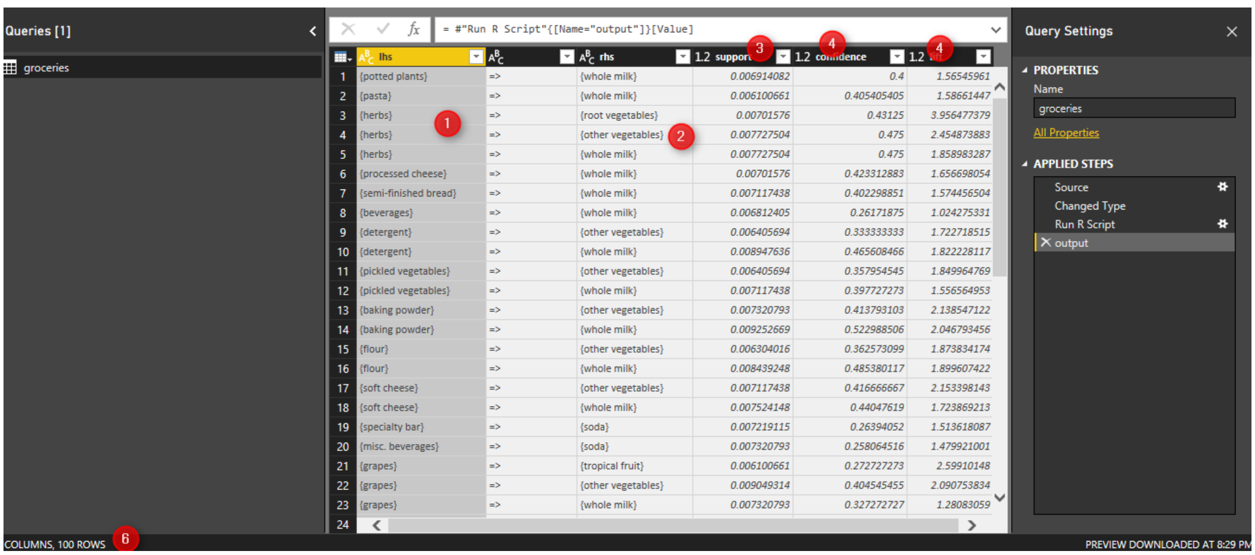
now we inspect the first 100 rules by calling the “inspect” function and put the output of function in “Output” variable.

output<-inspect(Temp[1:100])

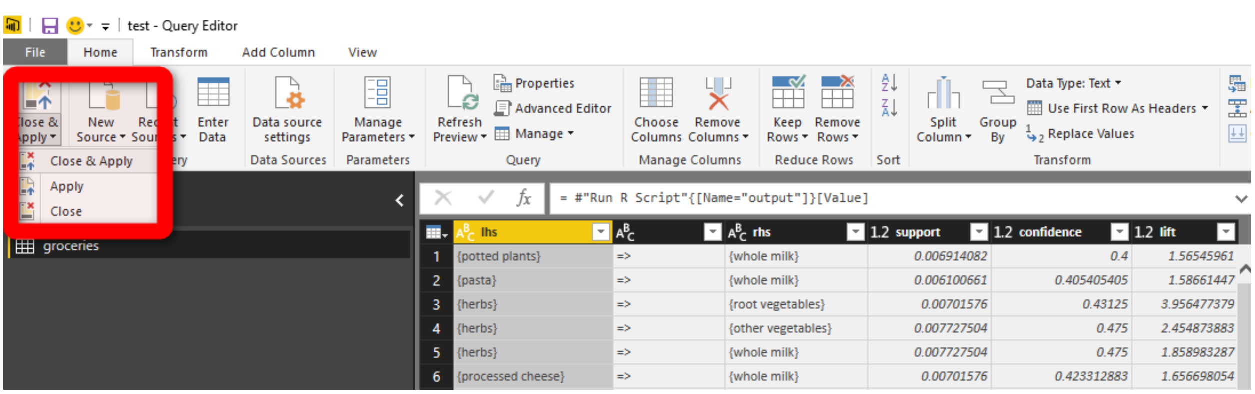
output variable is the result of the query.



After clicking on the “Output” (above picture number 1), we will see the below results. the result of finding rules of customer behaviour are shown in below image. the first column (lhs) is the main item that people purchase the third column (rhs) is the related items to (lhs). the support, confidence, and lift measures has been show in column forth to sixth.

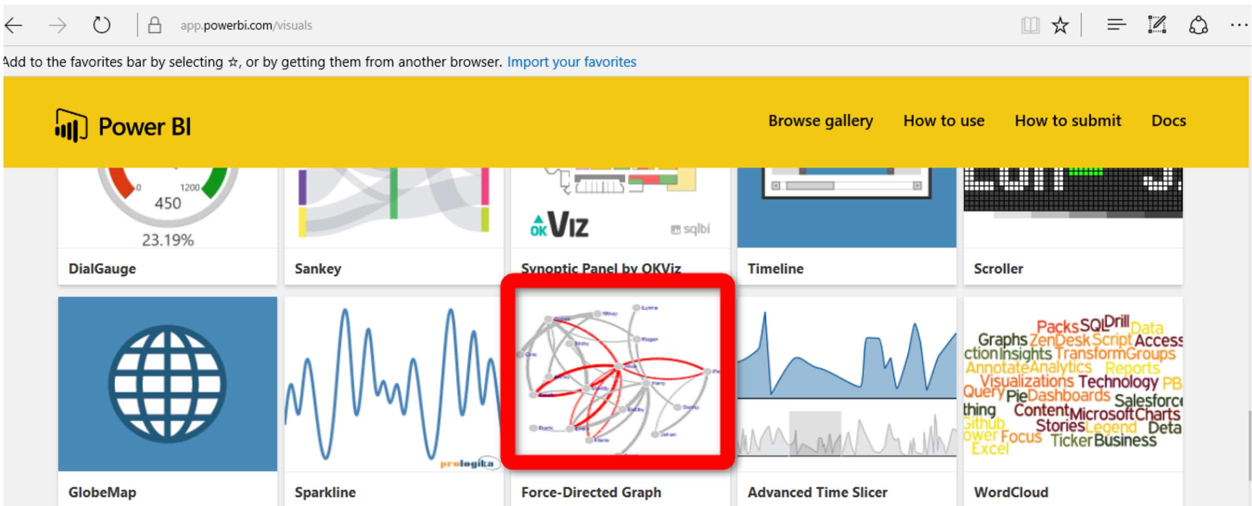


In total, 100 rules has been shown (number 6). Finally click “close and apply” on top left side (see below picture)

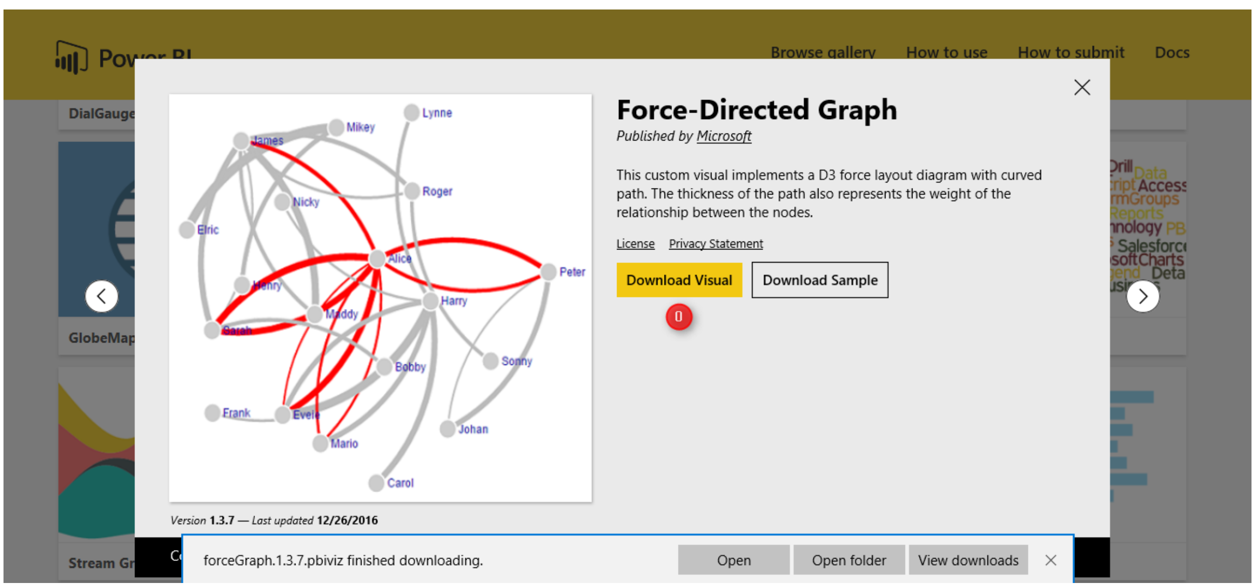


Now we can create a visualization for showing items and related items in Power BI visualization part.

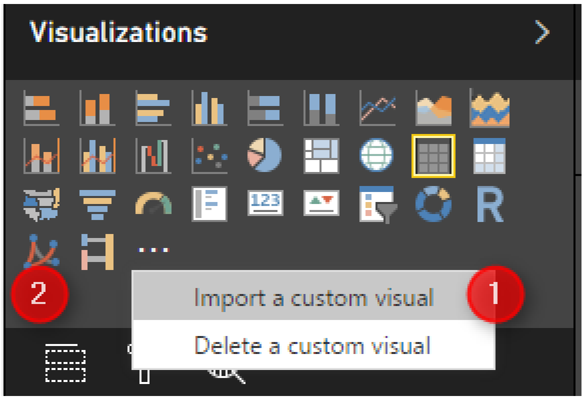
I am using the custom visualization from power BI website name as "Forced-Directed Graph" to show the relationships.



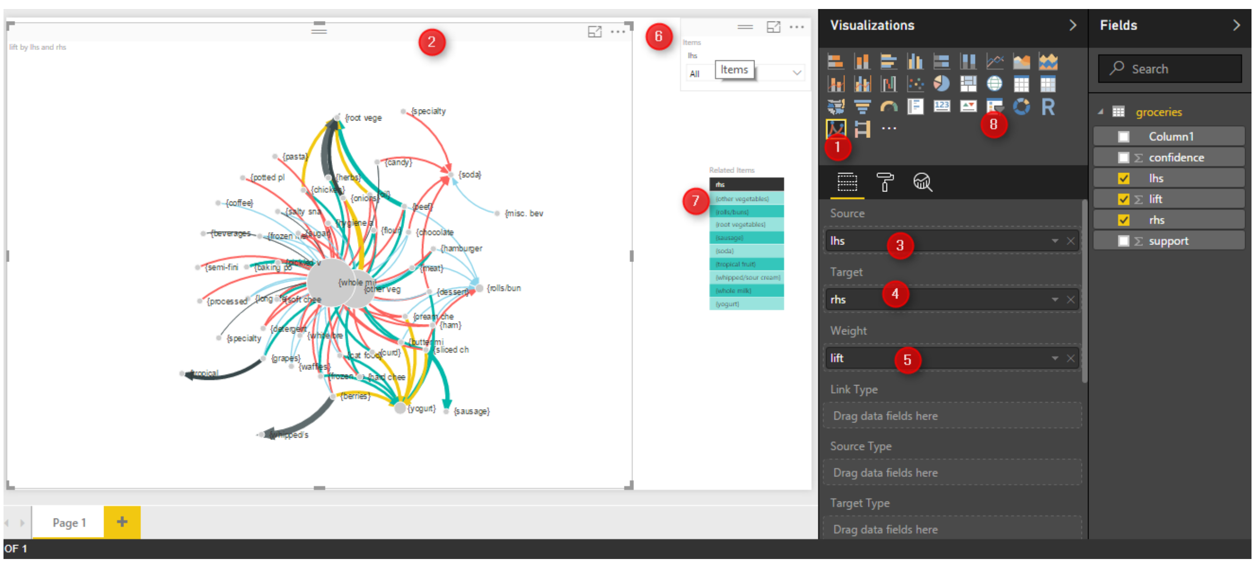
just click on the visualization and download it.



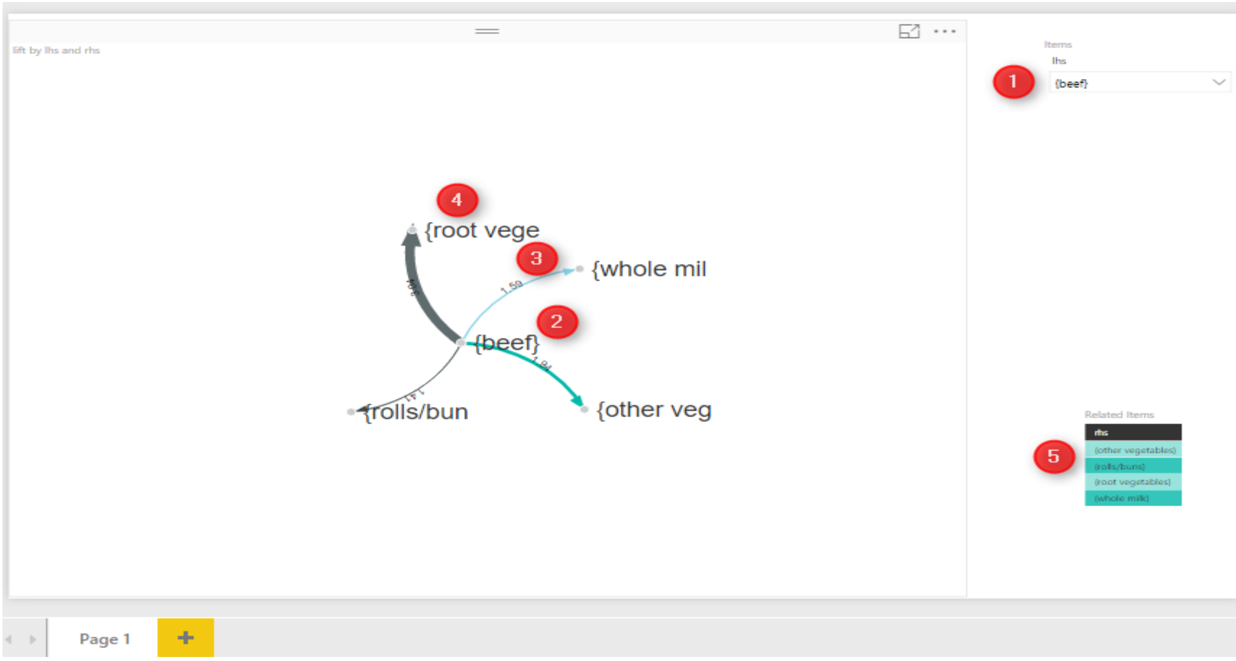
Import it to visualization part once download is completed. First click on the 3 dots and select the "Import a customer visual" and import the downloaded one.



Then in visualization first click on the right side and visualization part on the "Forced-Direct" chart (number 1) then in the right side in "Source" (number 3,4,5) bring the lhs, rhs and lift. this char shows the lhs (main items) as main node, the rhs (the related items in shopping basket) as the related nodes. the thickness of the line between the nodes, show the lift value. the bigger value for lift the thicker line. and the product is much more importance. I had also a drop down list for the items.



By selecting for example "beef" from dropdown list, the graphs show the related items to beefs. such as "root vege, whole mil, rolls, and Veg. the importance and possibility of purchasing these items has been shown by the thickness of the line so in this example "Root Veg" is main rules and has much more importance than the others.

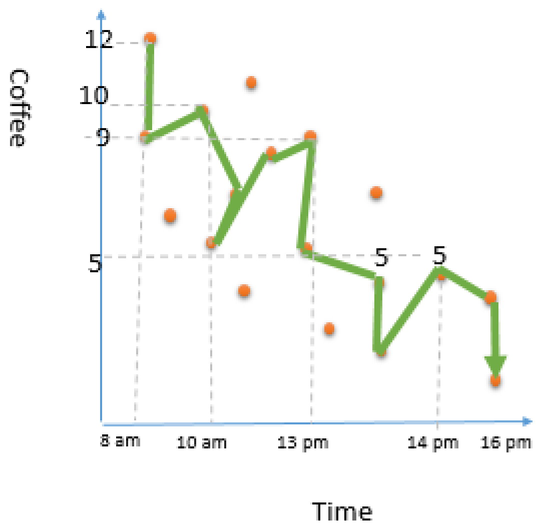


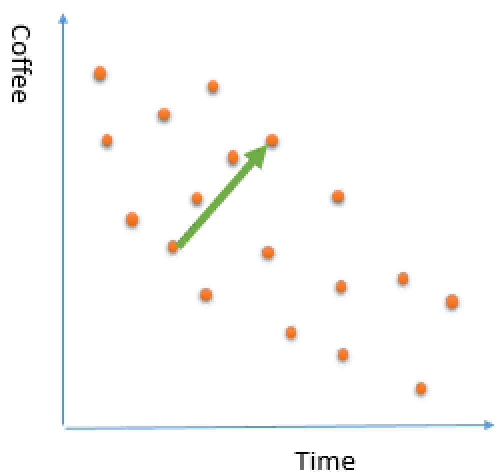
There are some other useful visualization that can be used for showing the customer shopping behavior .

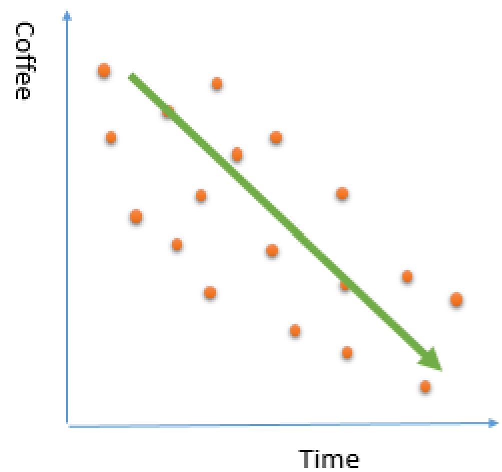
[1]Machine Learning with R,Brett Lantz, Packt Publishing,2015

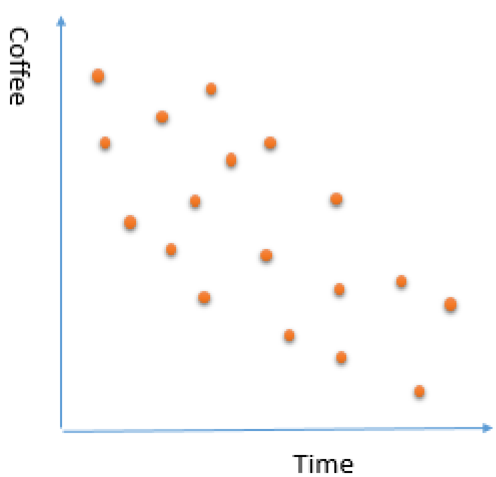
13-Over fitting and Under fitting in Machine Learning

Published Date : April 20, 2017


Coffee
12
10
9
5
5
5
8 am
10 am
13 pm
14 pm
16 pm
Time


Coffee
Time

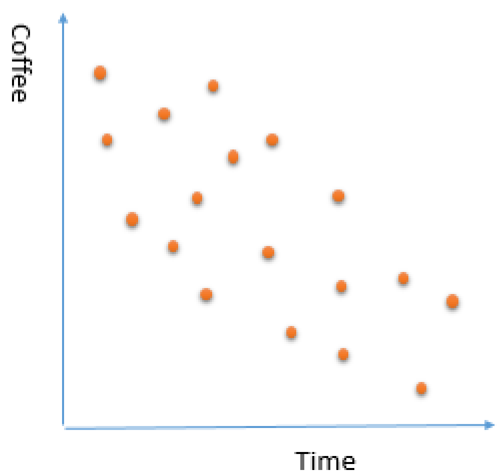

Coffee
Time


Coffee
Time

The main aim of machine learning is to learn from past data that able us to predict the future and upcoming data.

It is so important that chosen algorithm is able to mimic the actual behaviour of data. in the all different machine learning algorithms, there is a way to enhance the prediction by better learning from data behaviour.

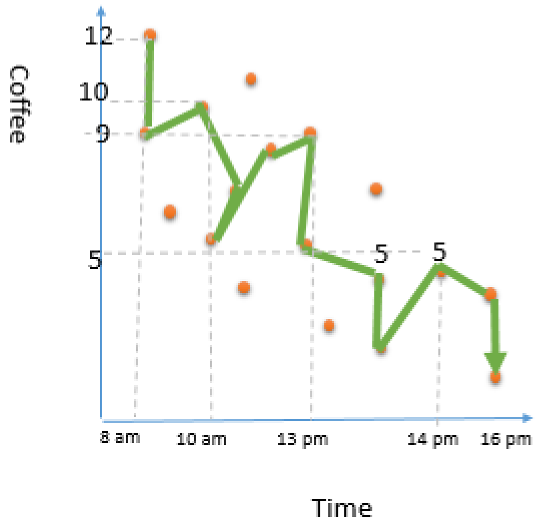
However, in most machine learning experience, we will face two risks: **Over Fitting and Under Fitting**.
I will explain these two concepts via an example below.
Imagine that we have collected information about the number of coffees that have been purchased in a café from 8am to 5pm.
we spotted below chart



We want to employ a machine learning algorithm that learn from the current purchased data to predict what is the coffee consumption during a working day. This will help us to have a better prediction on how much coffee we will sell in each hour for other days.
For learning from past data there is three main ways .
1- Considering all purchased data. for instance:

- 12 cups at 8am
- 10 cups at 10am
- 5 cups at 10.30 am
- 10 cups in 13 pm

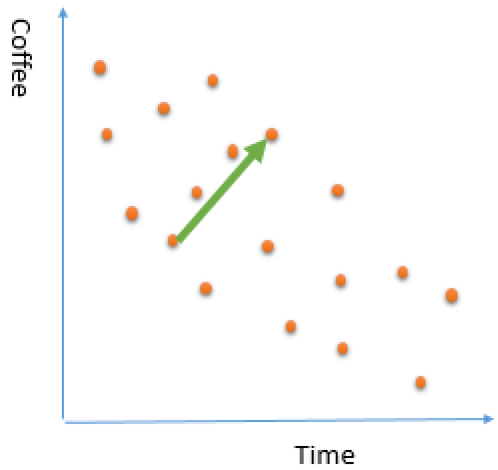
- 5 cups in 13.20 pm
- 3 cups in 13.40 pm
- 5 cups in 14 pm ?



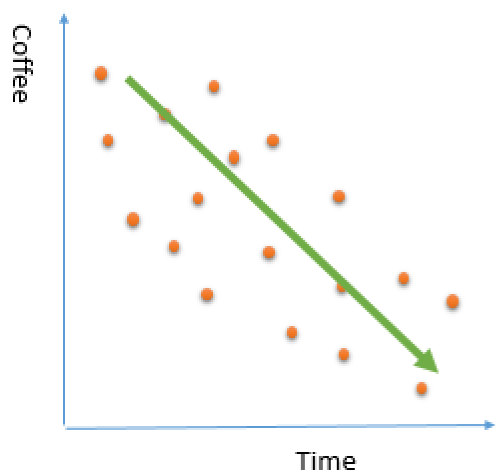
you see the number of the selling coffee change and fluctuate, so if we consider all the coffee purchased points, it is very hard to find a pattern and we have lots of "if and else" condition that makes learning process hard. for example, number of purchased coffee dropped suddenly at 8.30 am, which is not that much make sense. Because, in the morning people are more in coffee. Thus, by considering all the points, we are going to have some **noises** in data. This approach, will be have bad impact on the learning process. and we not able to come up with a good training data.

2- Considering a small portion of data. Sometimes we just consider a small portion of data that is not able to explain all behaviour. Imagine that in the above example , we just look at the coffee consumption from 11am to 2pm. as you can see in the below picture, there is a increase in the coffee consumption. Is this trend apply to other times? Is it correct to say for future days, number of purchased coffee will be increase by time of the day? not (at least our data not showing this)

This is an example of biased that we just consider a sample portion of data for explaining or learning from data. another name for this behaviour is "under fitting". this behaviour also lead to a poor prediction.



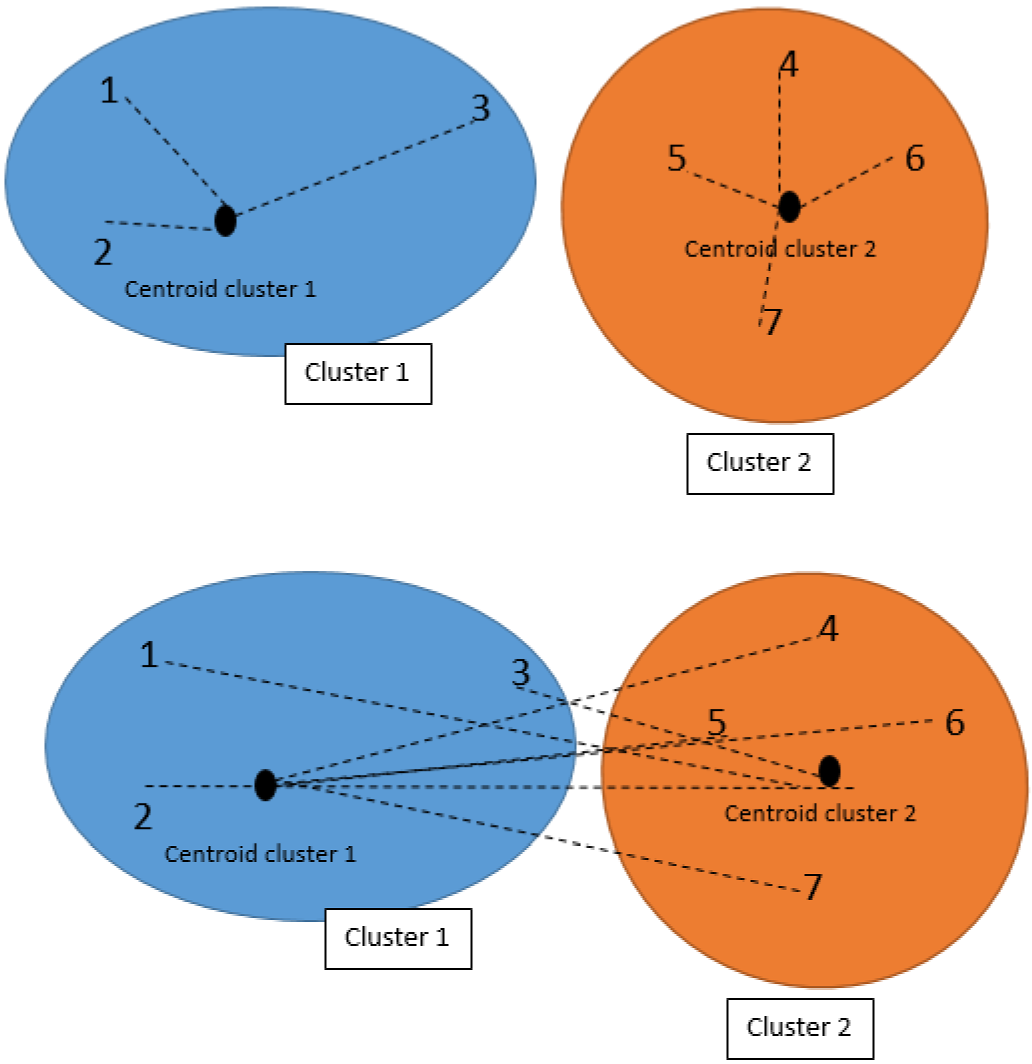
3- In real world, our data set there is a decrease in consumption of coffee during day. a good prediction model will find below trend in data. not a specific view of data point which not include all data.



each algorithms provides an approach to avoid these two risks. to see how to avoid over fitting or under fitting in the KNN algorithm, check the Post for k nearest neighbour, selection of variable "k" (number of neighbour). Or in the next post for identifying the number of clusters.

14-Clustering Concepts , writing R codes inside Power BI: Part 1

Published Date : May 1, 2017



Sometimes we just need to see the natural trend and behavior of data without doing any predictions. we just want to check how our business data can be naturally grouped. According to the Wikipedia, Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

For instance, we are interested in grouping our customers based on their purchase behaviour, demographic information [1]. Or in science example, we want to cluster the number and severity of earth quick happened in New Zealand for the past 10 years, or for medical purpose, we want to classify our patients with cancer based on their laboratory results. Clustering is non-supervised learning that means we are not going to assign any label for algorithm.

In this post, I am going to explain the main concepts behind the k-mean clustering and then in next post I will show you how I can use clustering to classify my Fitbit data using Power BI report.

Fitbit is an activity tracker, wireless-enable wearable technology device that measures data such as: number of steps, heart rate, quality of sleep, steps climbed, calorie burned, etc.

My history of my activities are downloaded in Excel format from Fitbit website (see below page)

As you can see, information such as calories burned, number of steps, distance, Floors, minute activities and so forth have been recorded.

A	B	C	D	E	F	G	H	I	J
Date	Calories B	Steps	Distance	Floors	Minutes S	Minutes L	Minutes F	Minutes V	Activity Ca
7/05/2016	1,730	2,758	1.84	0	1,328	112	0	0	346
8/05/2016	2,144	8,017	5.24	13	1,063	257	0	0	871
9/05/2016	3,000	24,257	15.84	53	946	303	50	61	1,912
10/05/2016	3,171	26,955	17.6	72	469	421	27	78	2,194
11/05/2016	2,083	6,098	3.98	3	1,251	175	6	8	715
12/05/2016	1,747	3,261	2.13	1	1,364	76	0	0	253
13/05/2016	1,787	10,932	7.14	6	1,350	78	10	2	327
14/05/2016	1,471	23	0.02	0	1,440	0	0	0	0
15/05/2016	1,747	2,615	1.71	0	1,152	134	0	0	392
16/05/2016	2,346	11,578	7.76	15	1,180	198	23	39	1,099
17/05/2016	2,653	15,067	9.95	38	1,121	230	23	66	1,444
18/05/2016	2,449	13,254	8.76	17	1,146	225	21	48	1,234
19/05/2016	2,123	8,704	5.68	10	1,252	148	9	31	794
20/05/2016	1,467	3,973	2.59	11	1,440	0	0	0	0
21/05/2016	1,948	5,557	3.63	0	1,236	183	9	12	652
22/05/2016	2,230	10,075	6.73	8	724	260	1	17	1,020
23/05/2016	2,258	8,902	5.96	5	1,440	0	0	0	0
24/05/2016	2,505	12,988	8.59	101	1,440	0	0	0	0
25/05/2016	2,483	13,786	9	26	1,237	175	8	20	820
26/05/2016	1,974	6,703	4.38	17	1,286	123	4	27	641
27/05/2016	1,467	84	0.05	0	1,440	0	0	0	0

fitbit_export

Now, I imported these data into the Power BI desktop for data transformation to remove unnecessary columns. Finally, I came up with the below table!

Calories.Burned	Steps	Floors	Minutes.Fairly.Active
1730	2758	0	0
2144	8017	13	0
3000	24257	53	50
3171	26955	72	27
2083	6098	3	6
1747	3261	1	0
1787	10932	6	10
1471	23	0	0
1747	2615	0	0
2346	11578	15	23
2653	15067	38	23
2449	13254	17	21
2123	8704	10	9
1467	3973	11	0
1948	5557	0	9
2230	10075	8	1
2258	8902	5	0
2505	12988	101	0
2483	13786	26	8
1974	6703	17	4
1467	84	0	0
1475	39	0	0
2383	12024	8	30
1874	4339	1	0

My aim is to see grouped data based on the calories burn, step number, floors and active minute. This helps me to see while I have high calories burned, did I have also high number of steps or just because of number of activities have high calories burnt.

Before explaining how I am going to use k-mean clustering to group my Fitbit data, first in this post, let me show an example on how k-mean clustering works. I will explain the concepts by using the good example provided by this blog [2].

There are different clustering approaches that proposed by different researchers. One of the popular ones is k-mean clustering. In **K-mean** clustering, **K** stands for number of clusters that we want to have from data. **Mean** is the mean of the clusters (centroid). That means we classify the data based on their average distance to the center of each cluster.

Imagining that we have a series of data as below, each individual with two set of value : A and B

Individuals	Value A	Value B
1	1	1
2	1.5	2
3	3	4
4	5	7
5	3.5	5
6	4.5	5
7	3.5	4.5

to cluster this dataset, we decided to have just two clusters, so it is a 2-mean (k-mean). first, I created 2 clusters based on the smallest and largest values. The smallest value is individual 1 with A & B (1,1). and the largest one is individual 4 with (5 ,7). we consider individual one is cluster one and individual 4 is cluster 2.

Then, we have two clusters with the below specifications

Cluster	Individuals	Mean (centroid)	Vector
1	1	(1,1)	
2	4	(5,7)	

Next step is to find the distance of other each individual from each of the two clusters. for example, for individual 2 we have to calculate its distance to cluster 1 (which currently just has individual one) and also calculate individual 2 distance value to cluster 2 (which has individual 4). I am using Euclidean distance to calculate distances.

Cluster	distance to cluster
1	sqrt ((1-1.5)^2+(1-2)^2)=1.11

2 sqrt ((5-1.5)^2+(7-2)^2)=6.11

So we choose the cluster 1(1.11) because it has the closet distance to the individual 2 than cluster 2 (6.11). At the beginning, the mean of cluster 1 was (1,1) since it only included individual 1. Now by adding individual 2 to cluster 1 we need a new "mean value" for cluster 1. we call the mean value of cluster as "centroid".

The mean (centroid) cluster 1 is the average of vectors for individual 1 and 2, so the new centroid for cluster 1 can be calculated via ((1+1.5)/2, (1+2)/2)=(1.2,1.5).

We did all of the above processes for all individuals, and came up with the below table of results. If I want to explain the process, it will be as below:

In step one, subject 1 compared with subject 1 (itself), so the centroid is individual value (1 1) and we have just one element in cluster 1.

The same for subject 4 in step 1.the centroid in cluster 2 will be the individual 4 value.

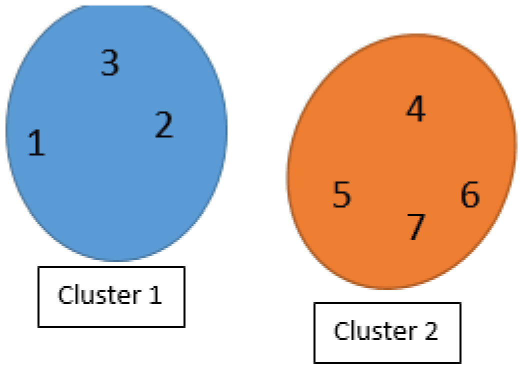
In step 2 we found that subject 3 has closet distance with subject 1 than 4. so we updated the centroid for cluster 1 (mean of the subject 1 and 3), we came up with the new centroid as (1.2, 1.5). and in the cluster 1 we have 2 elements and in the cluster 2 just one.

We followed the steps till all subjects were allocated to a cluster. In the step 6, which is final step, we have 3 elements in the cluster 1, and 4 elements in the cluster 2.

	Cluster 1		Cluster 2	
Step	Individual	Mean Vector (centroid)	Individual	Mean Vector (centroid)
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

Individuals 1,2, and 3 belong to cluster 1 and individuals 4,5,6,7 belongs to cluster 2.

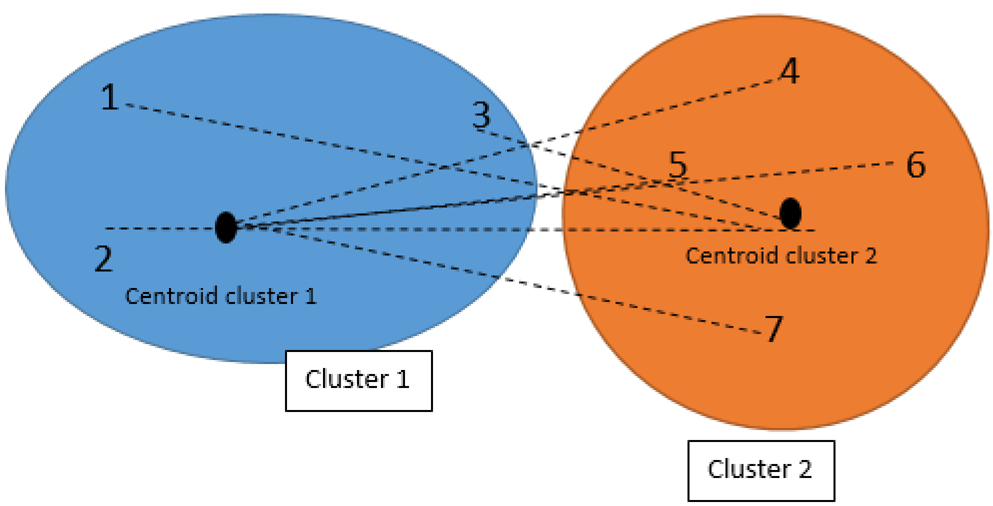
The best clustering result is when each individual has the closet distance to its cluster's mean (centroid). In our example the individual 1,2,3 should have the closet distance to their centroid which is (1.8,2.3). also, they should have the longest distance to other cluster's centroid.



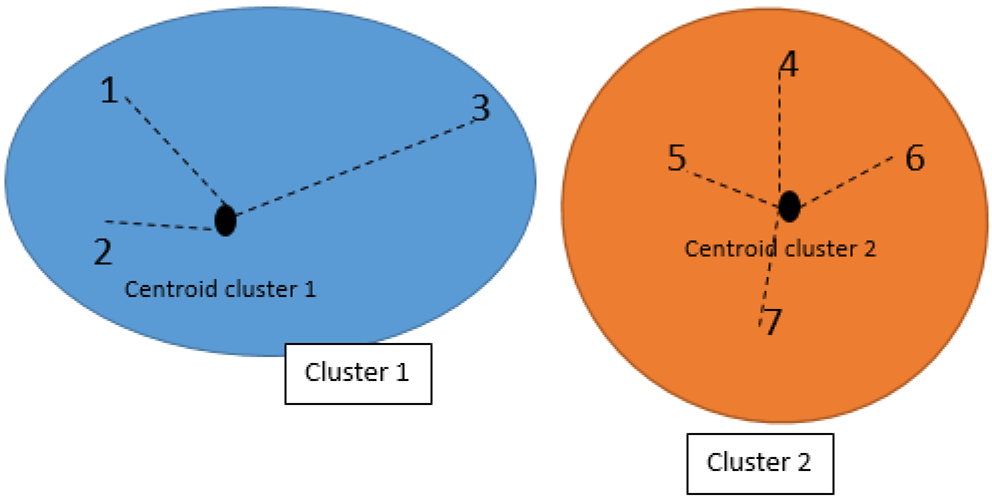
However, clustering is not completed! We should find the distance between each element with its cluster centroid (step 6 in above table) and also with another cluster centroid. Maybe some elements belong to other clusters.

So first we check the distance with other Clusters (see first picture). Then we calculate the distance with another cluster centroid

The below picture shows the distance of each individual to the centroid of the other clusters (not their own):



The below picture shows the distance of each individual with their own cluster's centroid.



Finally, we came out with the below numbers. as you can see in below table, the individual 3 now has closet distance to the cluster 2 than Cluster 1.

Individual	Distance to mean (centroid) of Cluster 1	Distance to mean (centroid) of Cluster 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

So, now we should rearrange the clustering as below.

Individual 1 and 2 belong to cluster 1, and individual 3,4,5,6,7 belongs to cluster 2.

This example explains the overall process of clustering. So, now I am able to show you how I applied clustering algorithm on my fit bit data in the Power BI in the next post.

[1] http://blogs.sas.com/content/subconsciousmusings/2016/05/26/data-mining-clustering/#prettyPhoto/0/

[2] http://mnemstudio.org/clustering-k-means-example-1.htm

15-K-mean clustering In R, writing R codes inside Power BI: Part 2

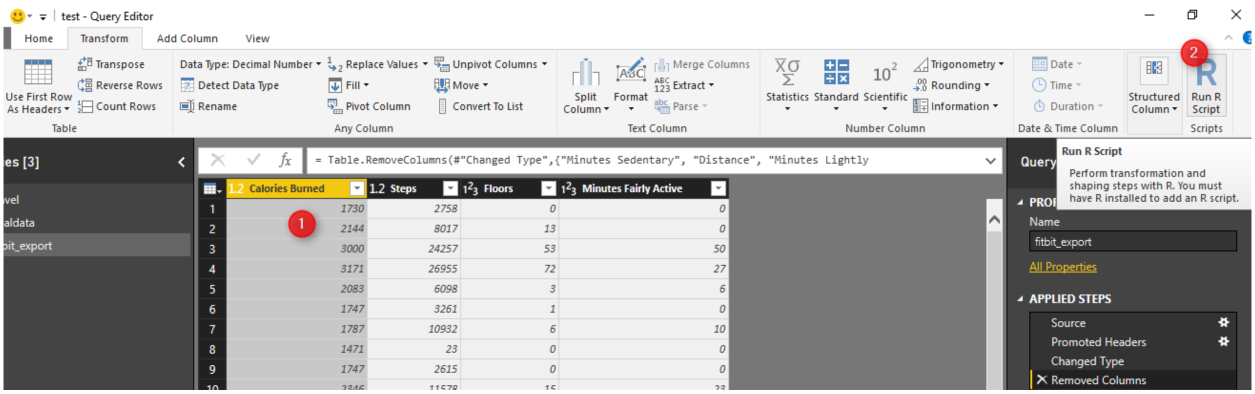
Published Date : May 2, 2017

In the previous post,I have explained the main concepts and process behind the K-mean clustering algorithm. Now I am going to use this algorithm for classifying my Fitbit data in power BI.

Calories.Burned	Steps	Floors	Minutes.Fairly.Active
1730	2758	0	0
2144	8017	13	0
3000	24257	53	50
3171	26955	72	27
2083	6098	3	6
1747	3261	1	0
1787	10932	6	10
1471	23	0	0
1747	2615	0	0
2346	11578	15	23
2653	15067	38	23
2449	13254	17	21
2123	8704	10	9
1467	3973	11	0
1948	5557	0	9
2230	10075	8	1
2258	8902	5	0
2505	12988	101	0
2483	13786	26	8
1974	6703	17	4
1467	84	0	0
1475	39	0	0
2383	12024	8	30
1874	4339	1	0

As I have explained in part 1, I gathered theses data from Fitbit application and I am going to cluster them using k-mean clustering. My aim is to group data based on the calories burned, number of steps, floors and active minute. This will help me categories my activities into three five main groups as "Lazy days", "Working Days", "Some Activities", "Active Days", and "Extremely Active" days.

First, in power BI, I clicked on "Edit Query". Then I choose the "Run R Script" icon.

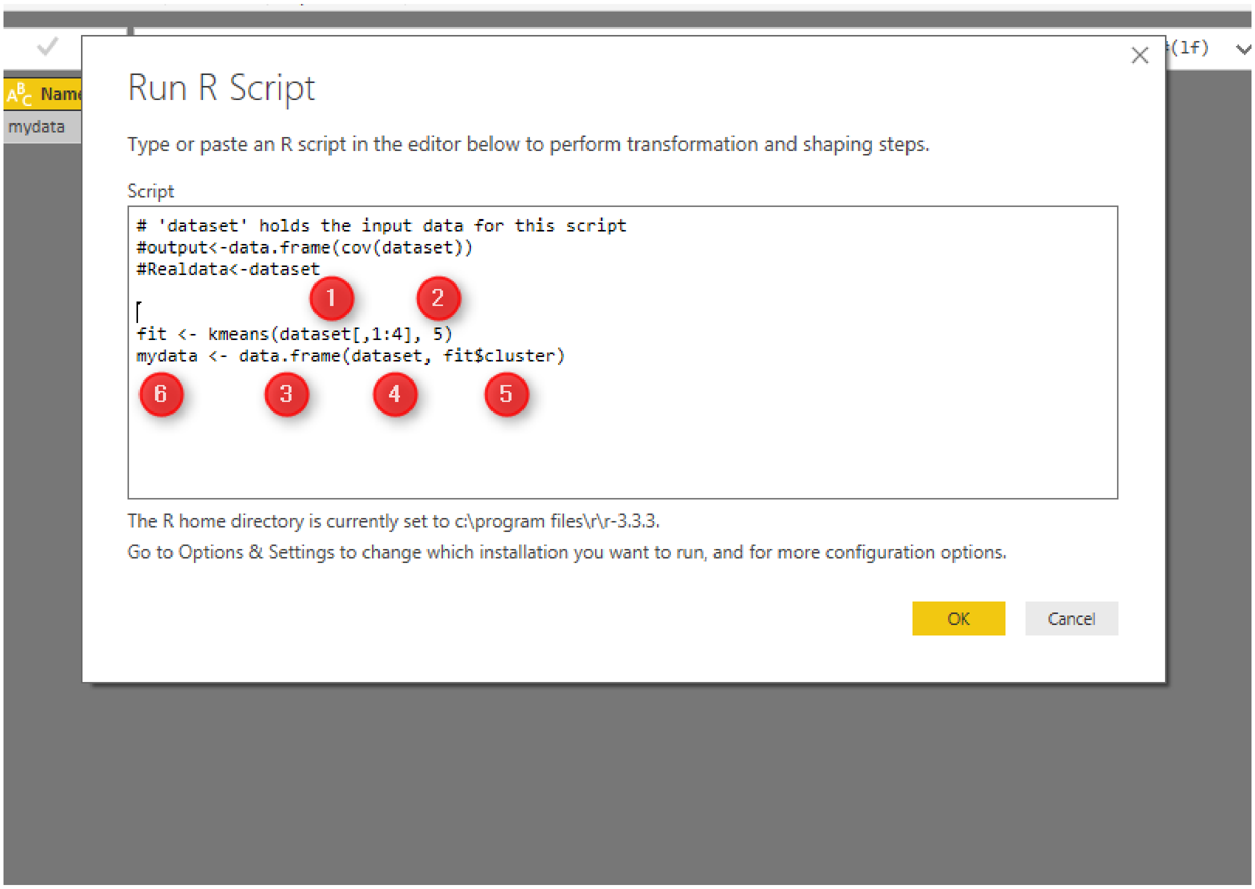


Next, write below codes in the R editor (see below picture).

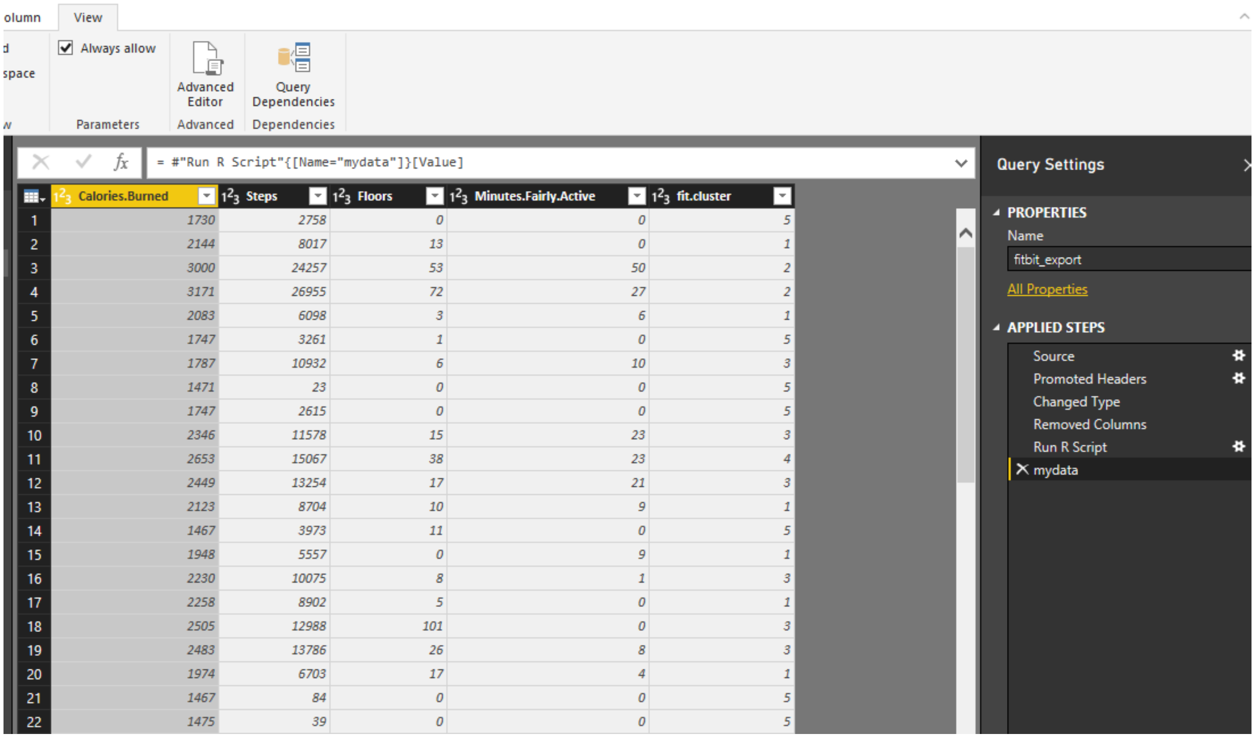
As you can see the data (fitbit data) is in variable "dataset".

K-means function in R helps us to do k-mean clustering in R. The first argument which is passed to this function, is the dataset from Columns 1 to 4 (dataset[,1:4]). The second argument is the number of cluster or centroid, which I specify number 5. There is some approach to find the best number of cluster (which will be explain later).

Tow the result of clustering will be stored in "fit" variable. Moreover, to see the result in power BI I need to convert dataset to "data.frame" format. so I called the function "data.frame" which gets "dataset" as the first argument. Moreover, the fit$cluster (result of k-mean clustering) will be added as a new column to the original data. These data will be stored in "mydata" variable. If, I push the ok bottom, I will have the result of clustering as new dataset.



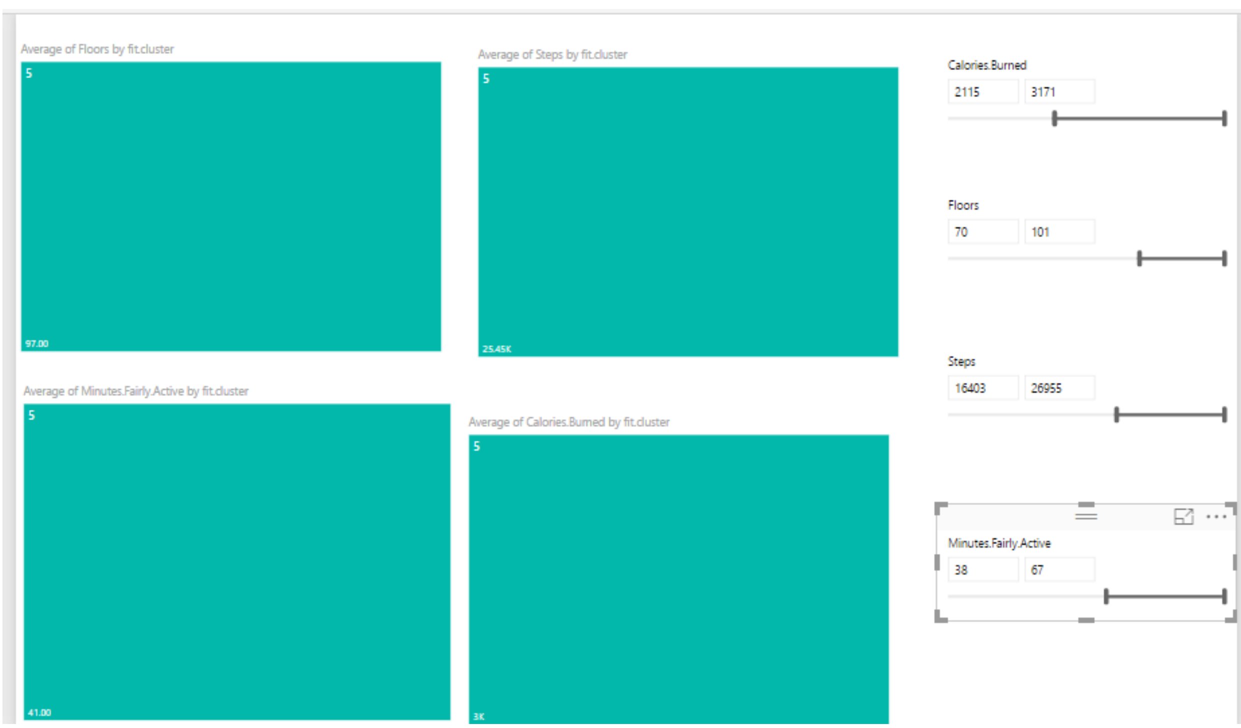
see below picture. In below data, each row has been allocated to a specific cluster.



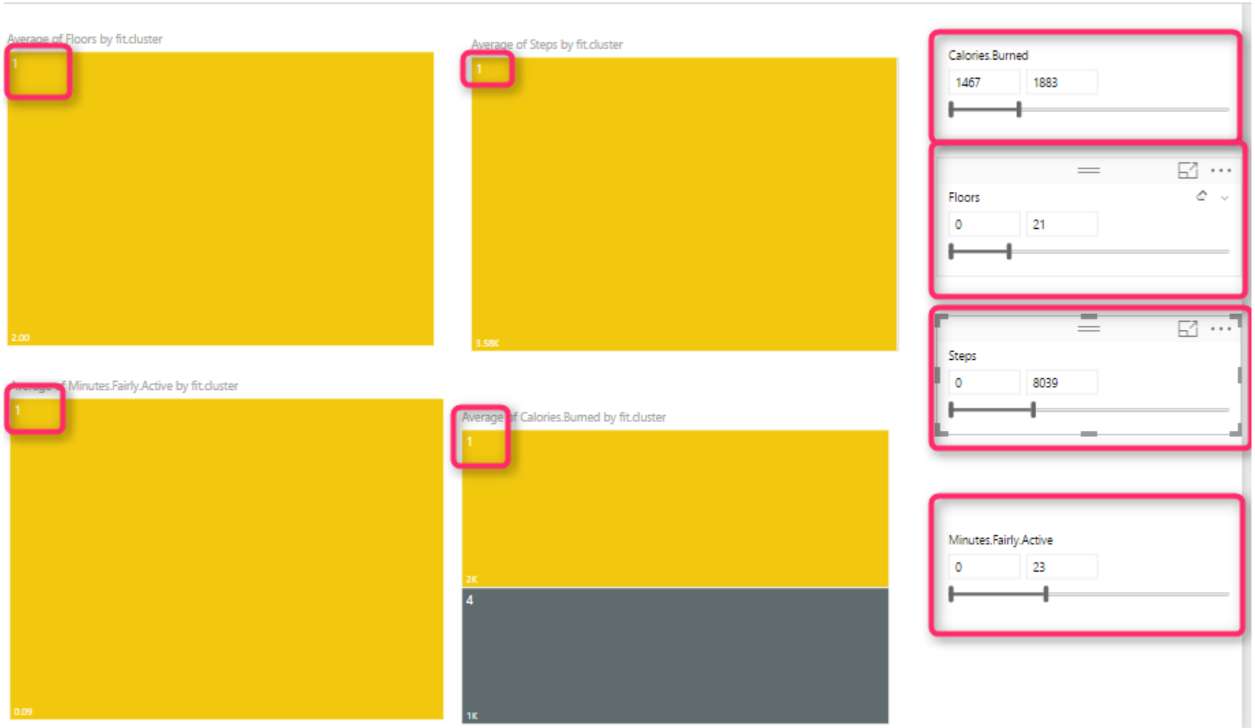
We have run clustering, I am going to show the results in power BI report, using power BI amazing visualization tools!

Created 4 different slicers to show “calories”, “floors”, “steps” and “activities”, I have four different heat map charts. Chart number 1 shows the average of” number of Floors” I did by different clusters, number 2 shows “average number of steps” by clusters, number 3 shows “average number of active minutes” by clusters, and finally number 4 show the “average number of burned calories” by clusters.

I am going to see if I burned between 2000 and 3200 calories, with 70 to 101 floors, and then with 16000 to 26000 steps and be active for 38 to 67 minutes I belong to which cluster. as you can see it will be cluster 5.

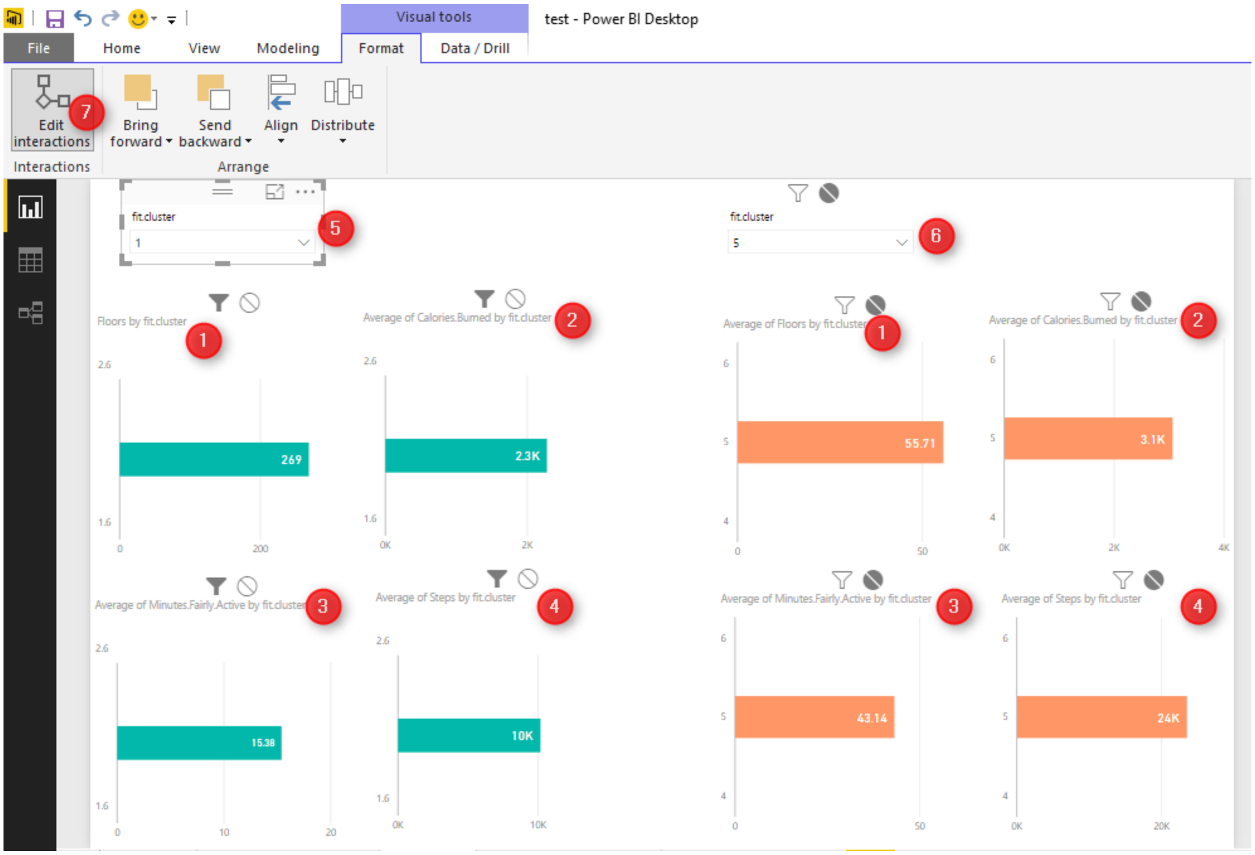


I did the same experiment and check different values to see the different cluster values as you can see below:



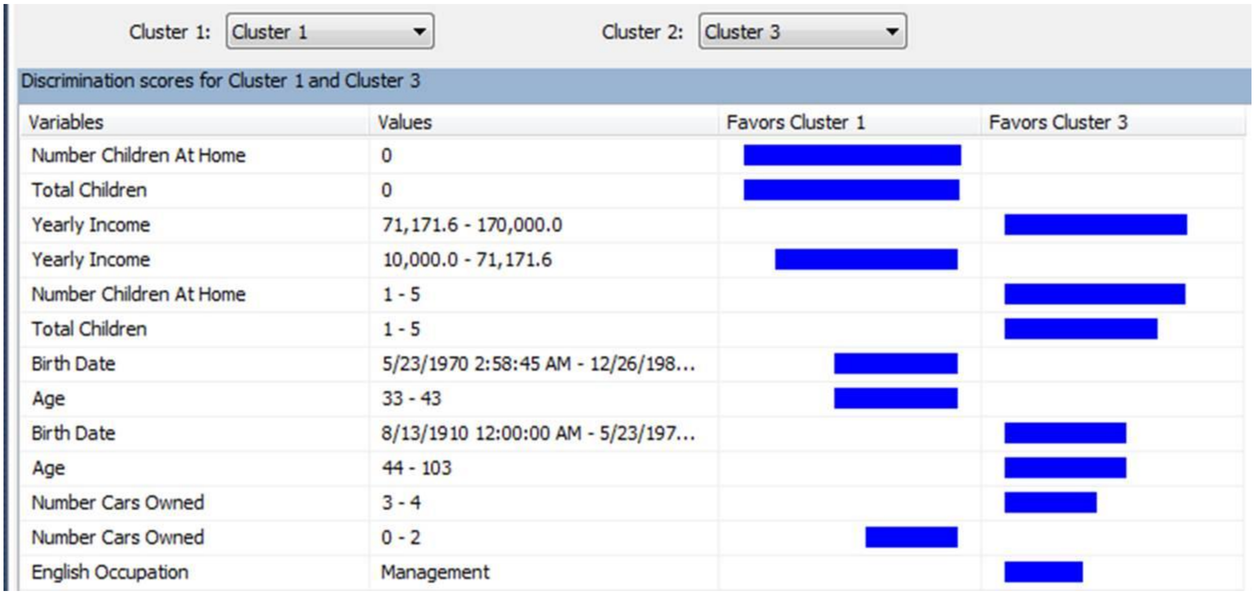
Another way of analysis can be done by comparing cluster 1 to cluster 2. To do that, I have created the below reports for comparing cluster's numbers. To create this report, I have two groups of charts (green and orange)

The first column chart (number 1) shows the "number of floors for each cluster", charts number 2 in each group show the average "burned calorie" by cluster, chart number 3 show the "average active minutes", and the last one shows the" average steps". Then, I created two different slicers to select cluster numbers. Number 5 for first cluster and number 6 for second one. As you know selecting a number in each slicer will have impact on the other charts. To prevent the impact of slicer number 5 on orange chart, I click on "Format" tab in power BI, then I choose the "Edit Interaction" option, now I am able to select by clicking on each slicer which chart should change and which not. In the following example, for first slicer I chose cluster number 1, and for the other slicer I chose number 5. I am going to compare their result together.



The good thing about using R with Power BI is that you can benefit from the great interactive and nice looking charts in Power BI and hence better analyzing data with R algorithms. There are many other different ways that we can analysis these results to get better understanding of our data.

This visualization reminds me the “data mining” tools we have earlier in “Microsoft SSAS” see below:



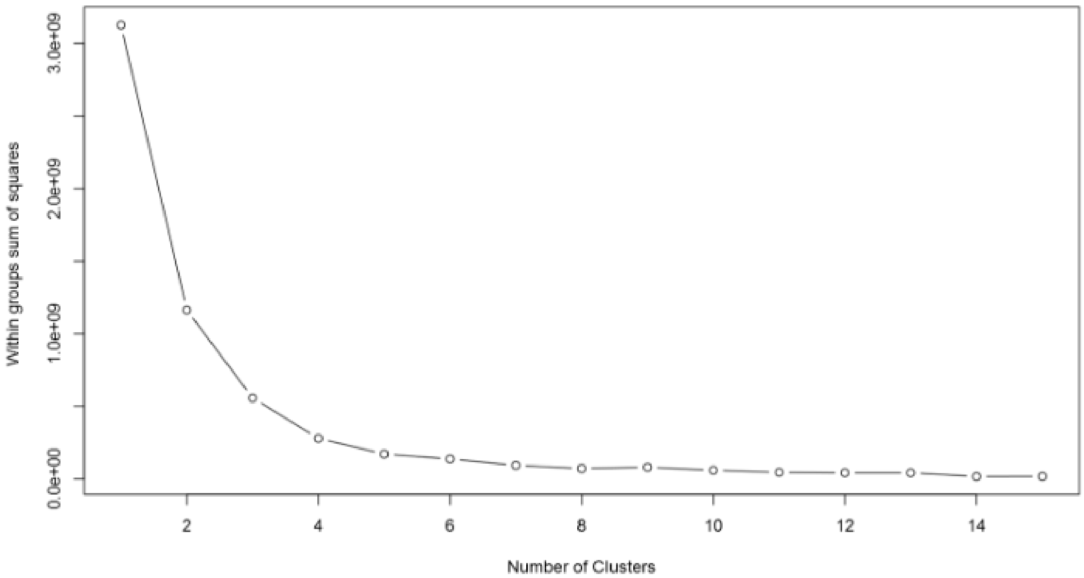
Cluster 1: Cluster 1 Cluster 2: Cluster 3

Discrimination scores for Cluster 1 and Cluster 3

Variables	Values	Favors Cluster 1	Favors Cluster 3
Number Children At Home	0	■	
Total Children	0	■	
Yearly Income	71,171.6 - 170,000.0		■
Yearly Income	10,000.0 - 71,171.6	■	
Number Children At Home	1 - 5		■
Total Children	1 - 5		■
Birth Date	5/23/1970 2:58:45 AM - 12/26/198...	■	
Age	33 - 43	■	
Birth Date	8/13/1910 12:00:00 AM - 5/23/197...		■
Age	44 - 103		■
Number Cars Owned	3 - 4		■
Number Cars Owned	0 - 2	■	
English Occupation	Management		■

16-Identifying Number of Cluster in K-mean Algorithm in Power BI: Part 3

Published Date : May 17, 2017



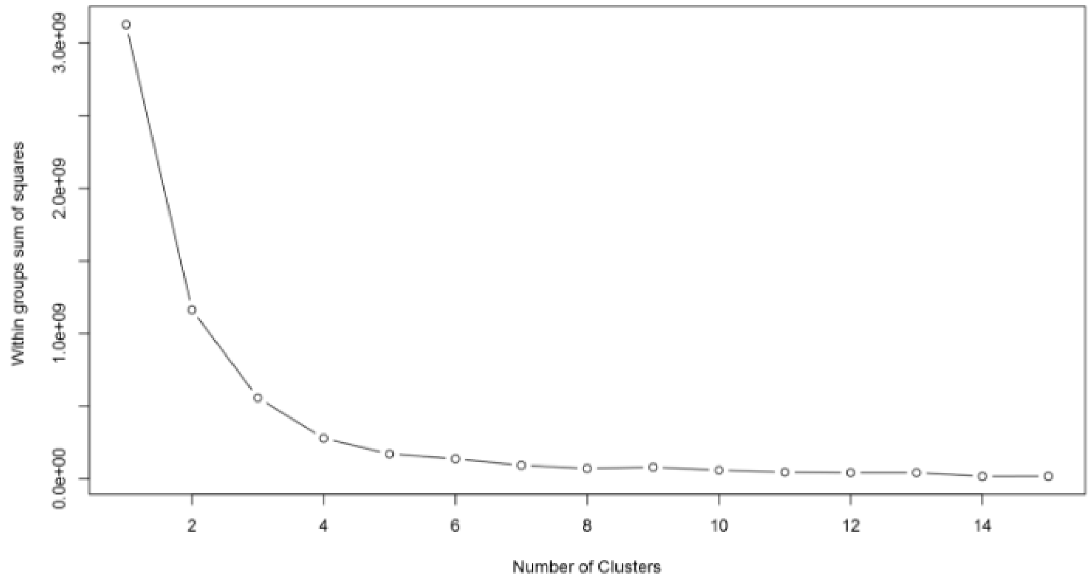
I have explained the main concept of the Clustering algorithm in [Post 1] and how to do cluster analysis in Power BI in [Part 2.]
In this post, I will explain how to identify the best number of cluster for doing cluster analysis by looking on the "elbow chart"

K-Mean clusters the data into *k* clusters. We need to identify the right number of clusters.

Elbow method is a way to validate the number of clusters to get the best performance. The idea of the elbow method is to run k-means clustering on the dataset for a range of K values.

The min concepts are to minimize the "sum of squared errors (SSE)" that is the distance of each object with the mean of each cluster. We try k from 1 to the number of observation and test the SSE.

Let's have a look on a "Elbow Chart".



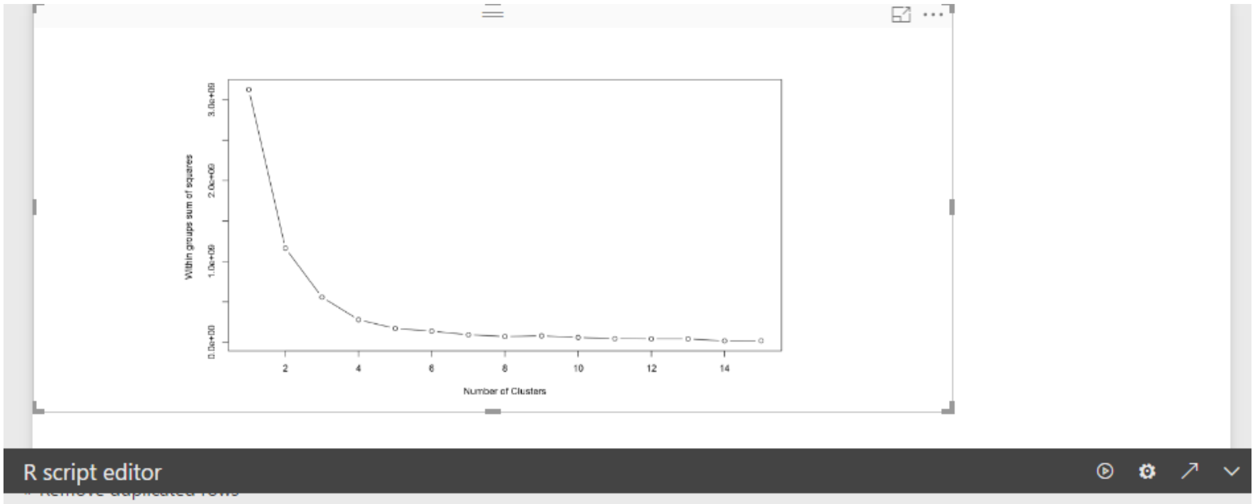
From above picture, Y axis is SSE that is the distance of objects from the cluster mean. The smaller SSE means that we have better cluster (see post part 1).

The number of cluster increase in X axis, SSE become smaller. But we need the minimum number of clusters with the minimum SSE, so from the above example, we choose the elbow of chart to find both minimum number of cluster and minimum SSE.

Back to example I have done in post part 2, I am going to show how to have Elbow chart in Power BI using R codes. The blow code help us to have elbow chart inside Power BI.

```
wss <- (nrow(dataset[,1:4])-1)*sum(apply(dataset[,1:4],2,var))
for (i in 2:15) wss[i] <- sum(kmeans(dataset[1:4], centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters", ylab="Within groups sum of squares")
```

I write this code inside Power BI R editor visualization.



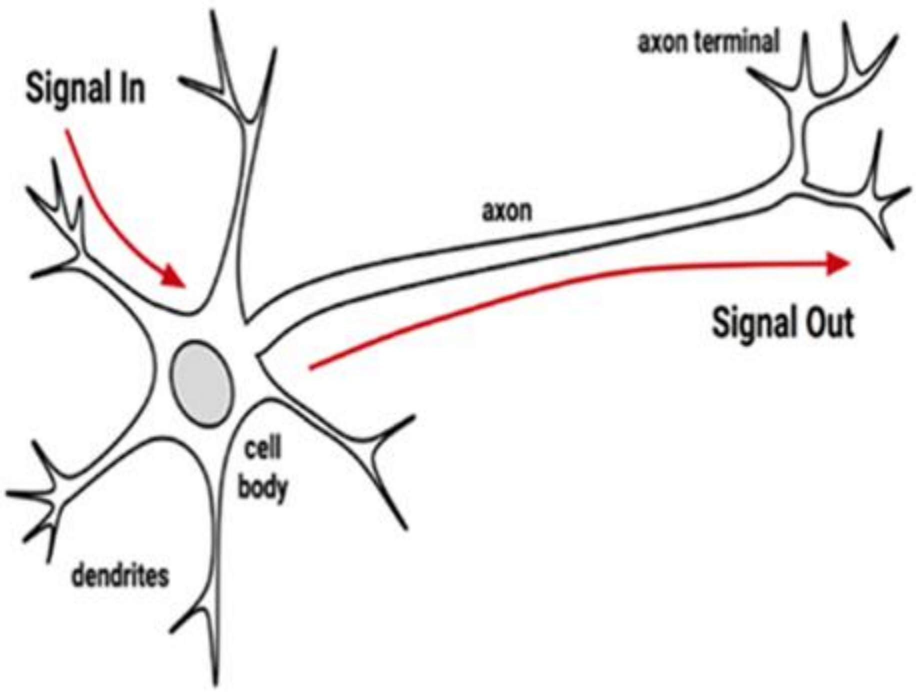

```
# dataset <- unique(dataset)
wss <- (nrow(dataset[,1:4])-1)*sum(apply(dataset[,1:4],2,var))
for (i in 2:15) wss[i] <- sum(kmeans(dataset[1:4],  centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters", ylab="Within groups sum of squares")
```

According to the above explanation, for clustering Fitbit data we need 4 or 3 cluster. which is minimum SSE and minimum number of Cluster. by applying this number, w should have better clustering.

[1]https://stats.stackexchange.com/questions/147741/k-means-clustering-why-sum-of-squared-errors-why-k-medoids-not

17-Neural Network Concepts Part 1

Published Date : June 26, 2017

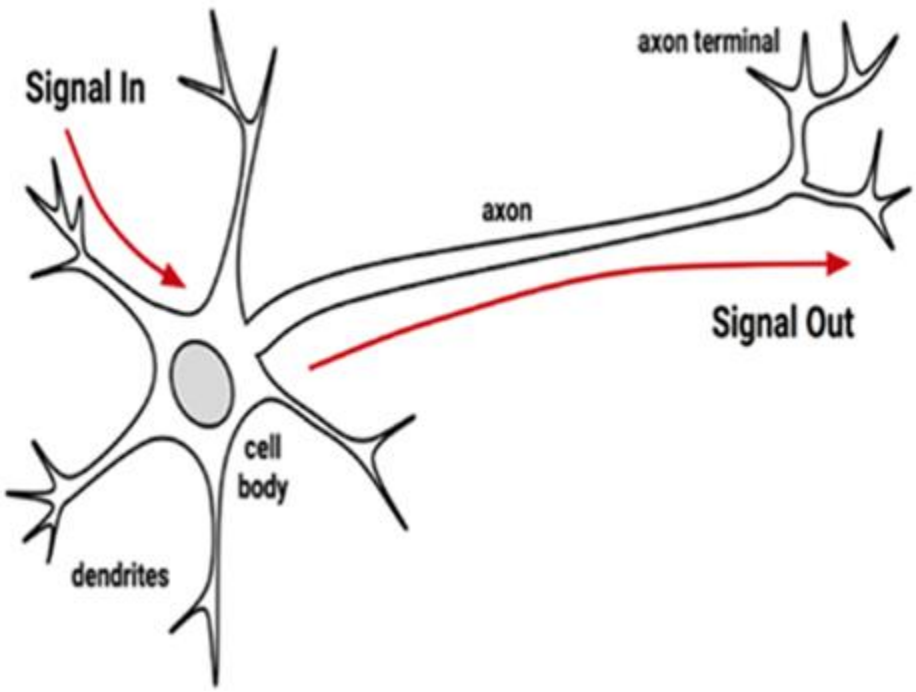


In this article and next one, I will share my understanding on Neural Network and how to write the related R code inside the Power BI.

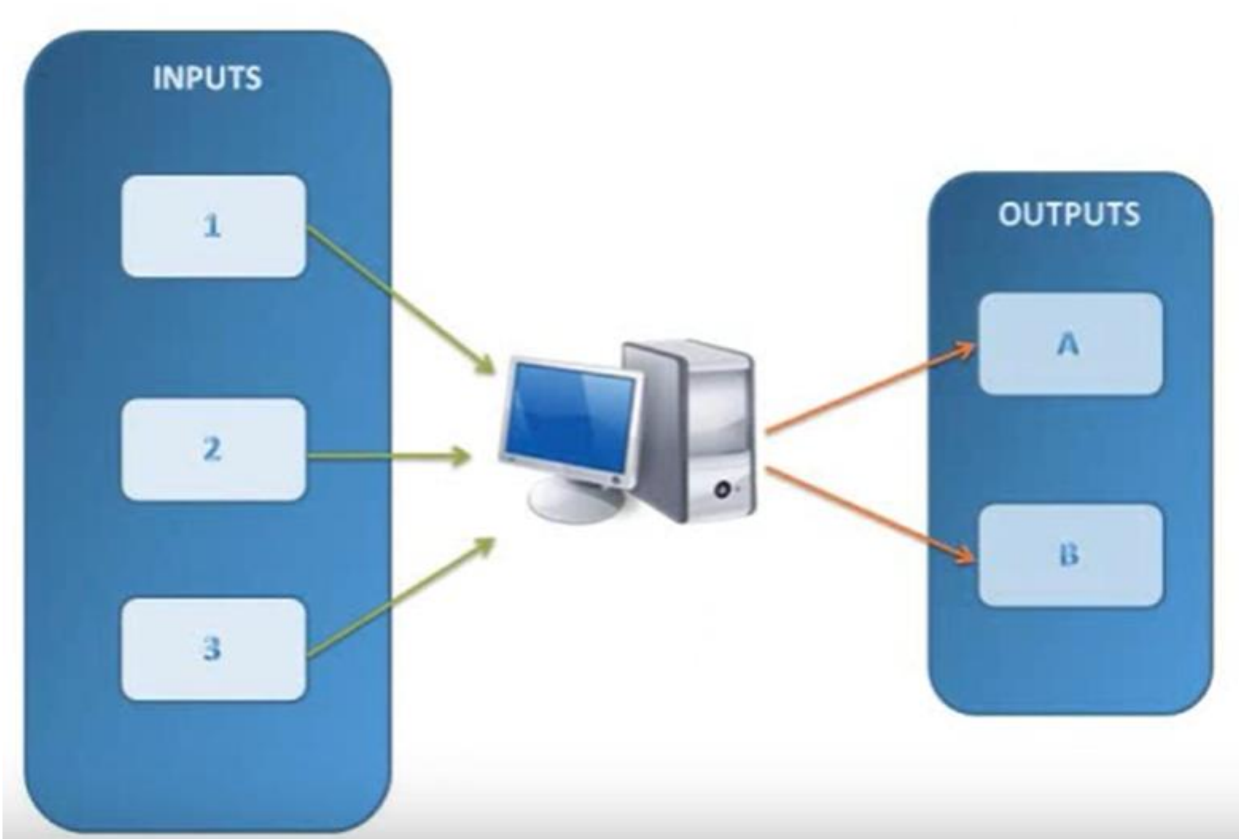
First, in this post I am going to explain the main concept of the Neural Network and how it works. The video https://www.youtube.com/watch?v=DG5-UyRBQD4&spfreload=10 helped me a lot to get better understand the main concept behind the neural network also the book that I put in reference was also a good source for it.

What we expect from a computer is that we provide some inputs and then we received outputs that match our needs. Scientist try to mimic the human brain for creating any intelligence machine. A machine that do the reasoning same way as human does.

The most important human brain element is neurons. Human brains consist of 75 million neurons. Each neuron is connected to other via a synapse. So, what we have in Neural network is some nodes that are connected to each other. In human body, if a neuron trigger by some external elements, it will pass the message from the receiver node to other nodes via synapsis.



Neural network mimic the same concepts from human brain. One node gets some inputs from the environment and then Neural Network model creates outputs. This process is so similar to the computer system behaviour.



In Neural Network, we have below components

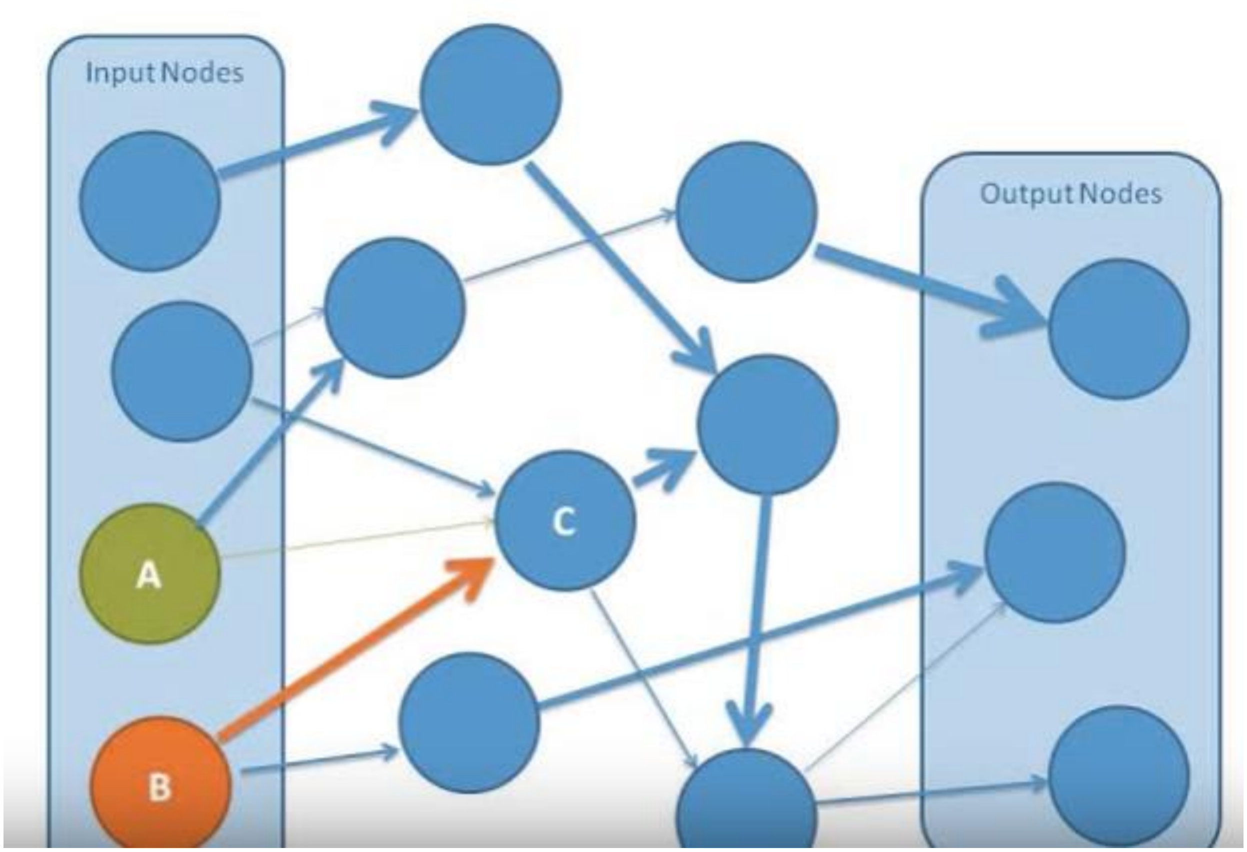
1- Set of inputs nodes

2- Set of output node/s

3-Some processing in middle to achieve a good result

4- The flow of information

5- The connection between nodes

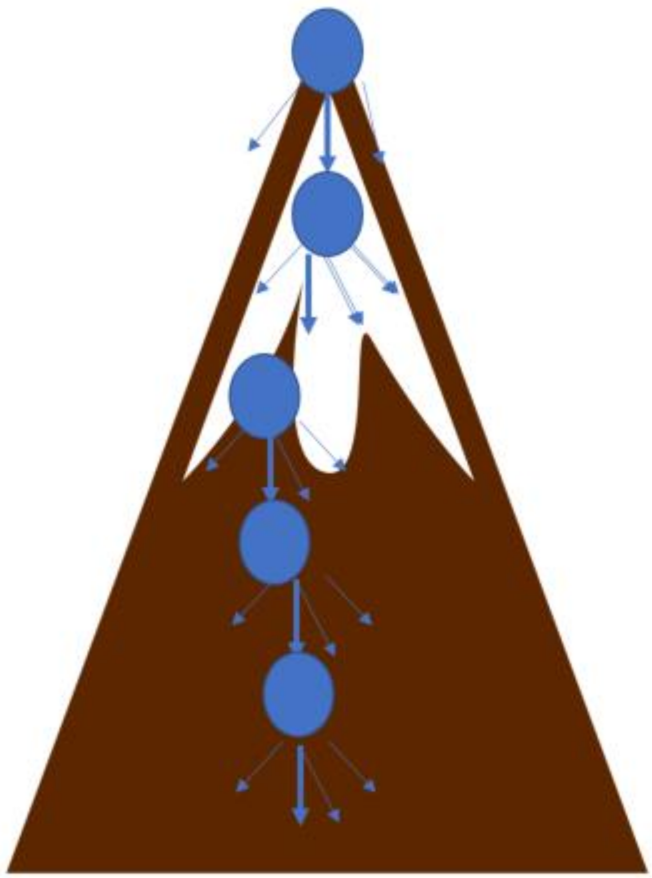


Some of the connections are more important than the other. That means they able to have more impact on the result than the others. In Neural Network, we call them Weights.

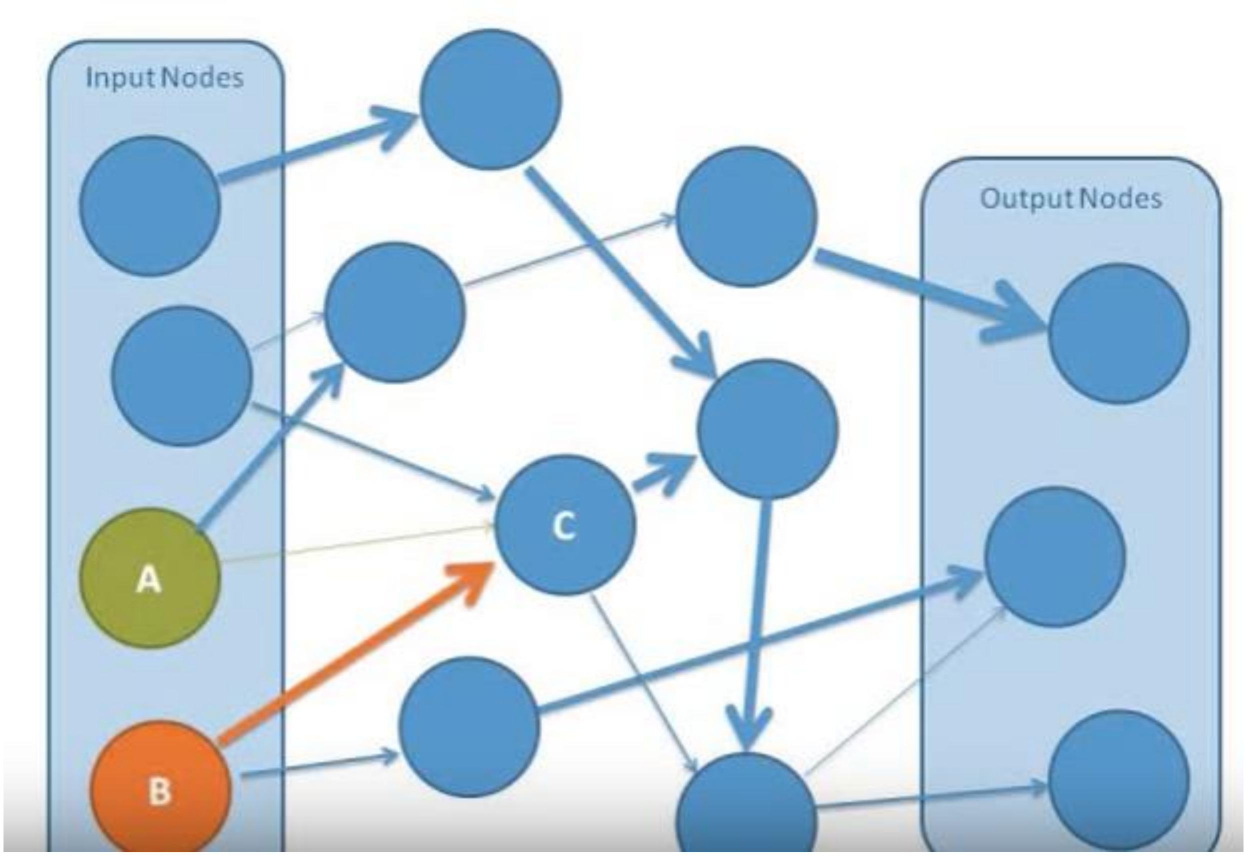
So what is a weight? There is a really good example in video (https://www.youtube.com/watch?v=BR9h47Jtqyw)

Imagine we want go to back from a hiking journey, we are in top and weather is foggy, so we only able to see the 1 meter ahead, so we can decide which direction we should go just for one meter ahead.

We put the first step now based on the location again we decided which direction we should go and tale the other steps, so in each step we evaluate the way and choose the best way till we come down the mountain.



These decision place can be seen as a node of decision that lead us to a better and closer point. In Neural Network, we have some hidden Nodes that do the main job! They found the best value for the output, they are using some function that we call that functions as “Activation function” for instance in below picture, Node C is a hidden node that take the values from node A and B. as you can see the weight (the better path) related to Node B as shown in tick line that means Node B may lead to get better results so Node C get input values from Node B not Node A.



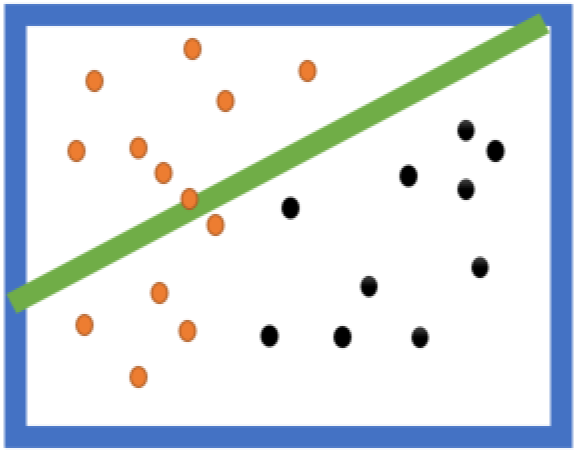
Different layouts of neural network has:

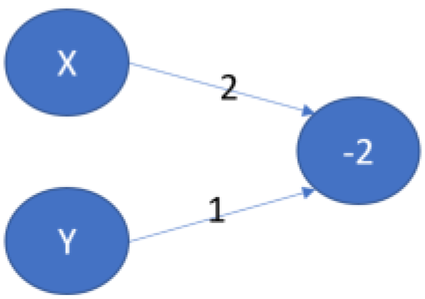
single-layer network: all input connected to one output via some link and specific weight without applying any function. this is a very simple and can be so similar to linear regression.

multilayer network: there maybe some hidden nodes, and most of the time they are fully connected. It means that every node in one layer is connected to every node in the next layer,

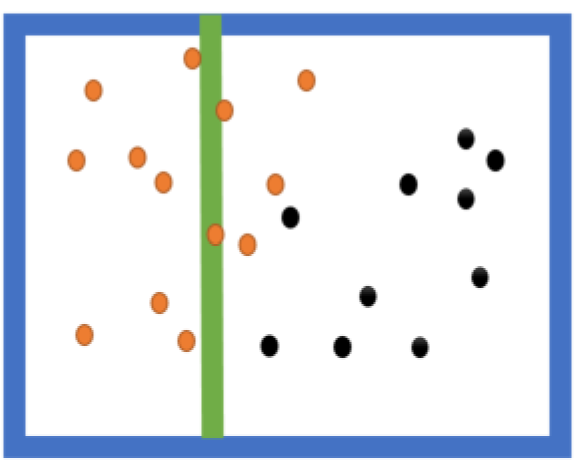
why we need hidden nodes? To answer this question, let's look at the below example:

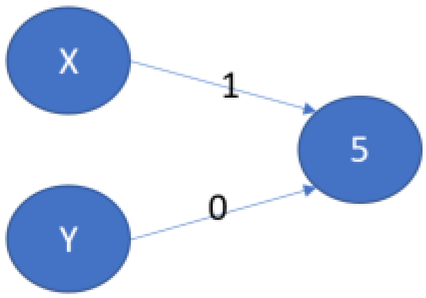
Imagine that we have a data like below picture. We aim to classify the data into two groups as black and orange. First, we apply formula 2x+y=-2 to separate them, as you see in the below picture, formal able to cover 60% of the classification, so still some orange dot is in the black area, the line is not that much accurate and not able to fully classify the data.





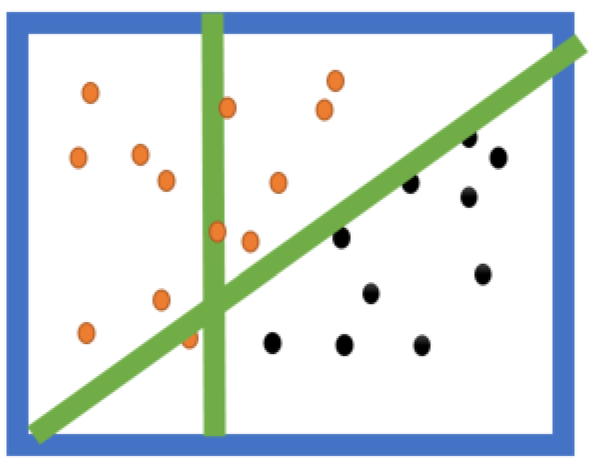
In the other linear formula we have this formula to classify the black and orange groups: x=5 so we have 1*x+0*y=5, this line also not that much able to classify the nodes, it able partially to do it.



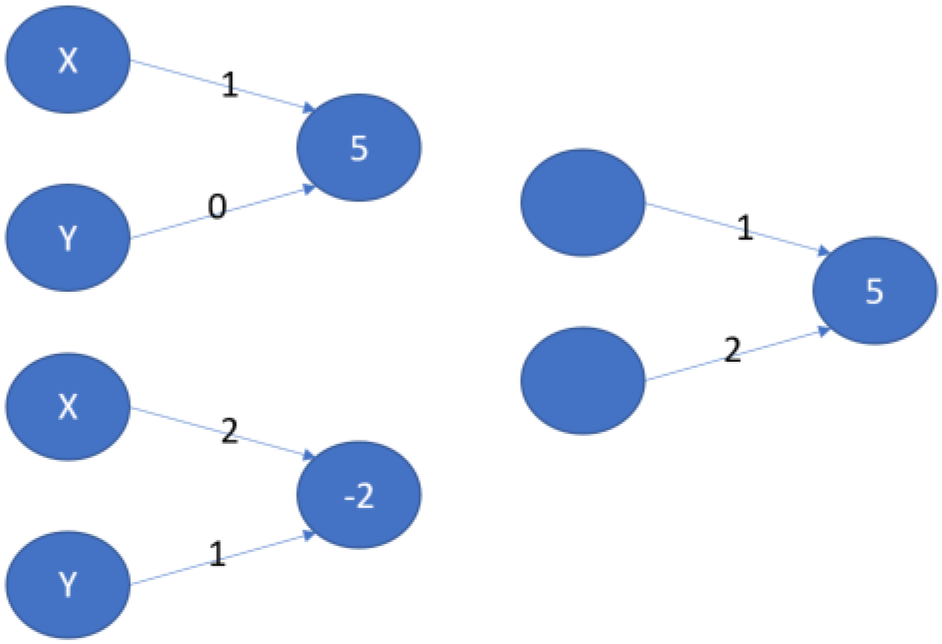


We have another formula that is combination of these two and are more effective in classifying the black nods from orange one, see below picture. I can sum formula1 and formula 2 to be able to classify better the black and orange nodes.

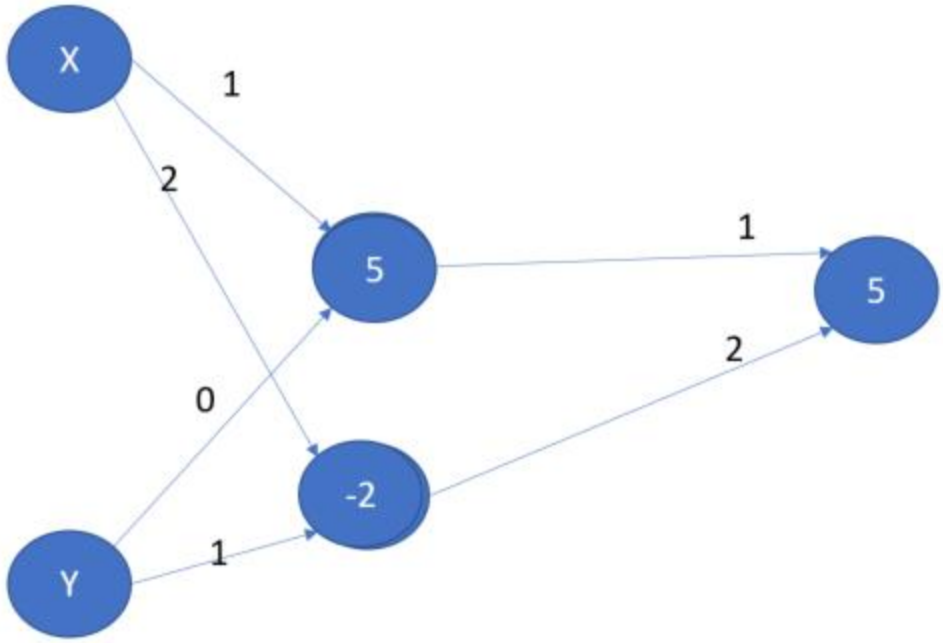
with formula will be : -8*x+y=-6

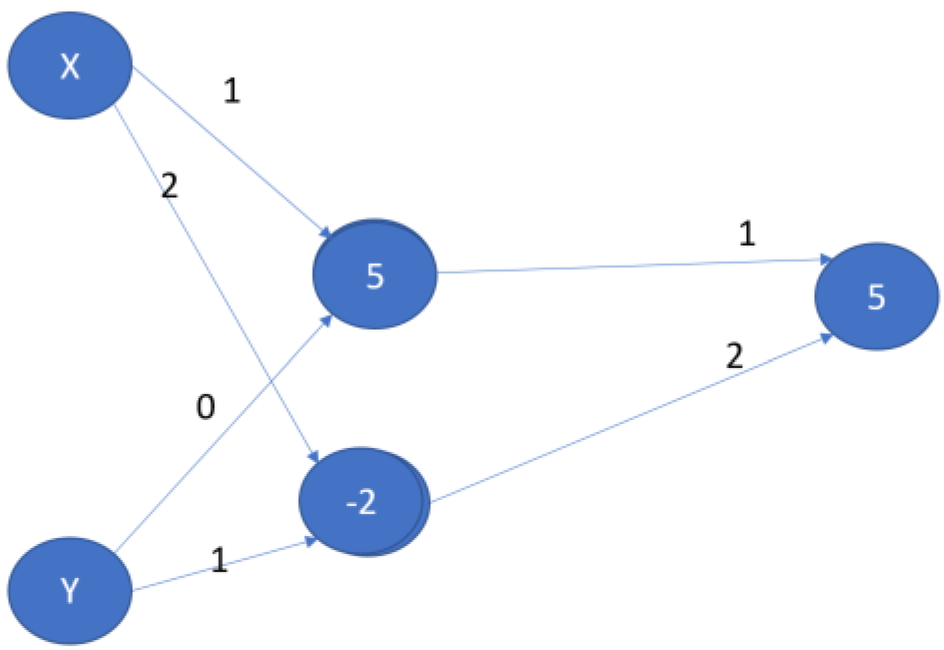


with this example, we first apply two function on our input values (X and Y). Then, we merge them to find a better formula that able to classify the data. We use some activation function to join these two formulas to reach a better result. Hence, we have below network:

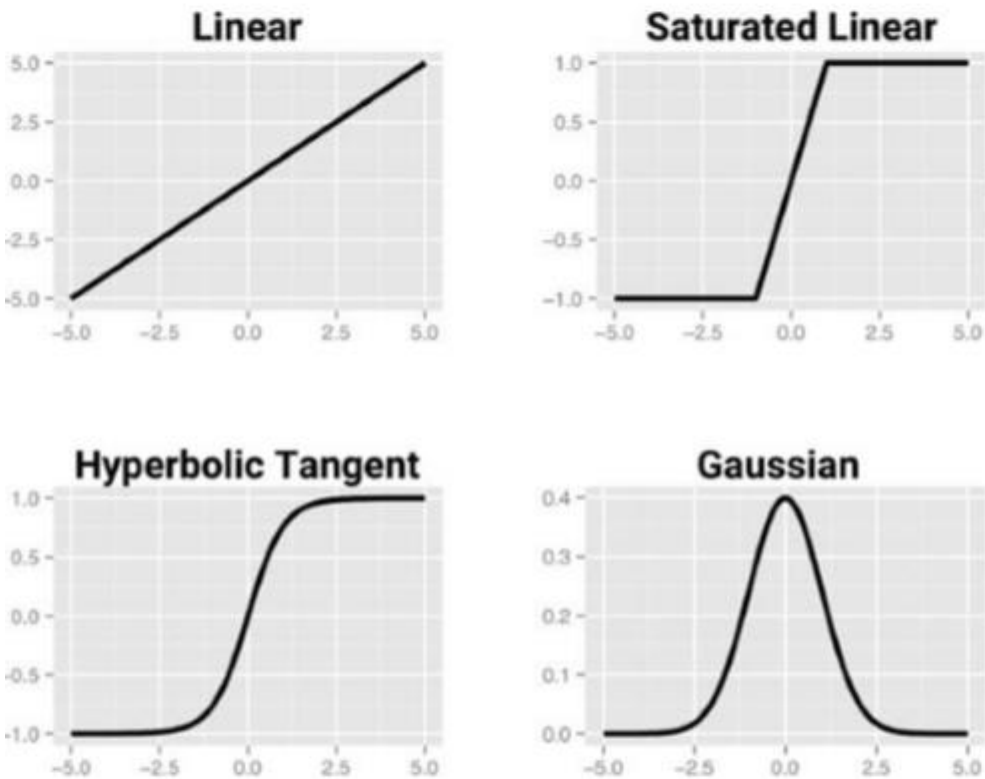


we connect the nodes and we have below fully structure Neural Network network.





What function we can use for activation and merging the nodes? there are many of these activation function such as linear, Saturated Linear, Hyperbolic Tangent, and Gaussian. I am not going to explain them as for this post and Next one we just want to use them, will discuss them in future.



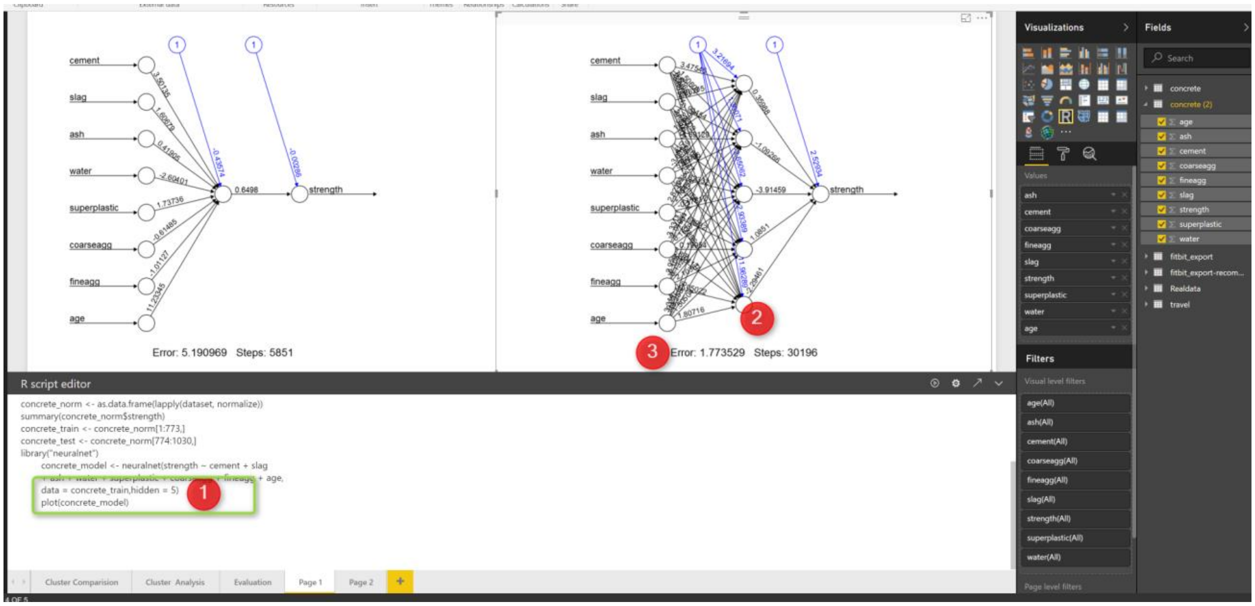
These are the main concepts behind the neural network. In the next post, I will show how to write code in R and Power BI. I found the blow videos good for understanding the main concepts.

https://www.youtube.com/watch?v=DG5-UyRBQD4&spfreload=10

https://www.youtube.com/watch?v=BR9h47Jtqyw

18-Neural Network R Codes in Power BI Part 2

Published Date : June 27, 2017



In the last article, I have explained the main concepts behind the neural network, in this post I will show how to apply neural network in a scenario in R and how to see the results and hidden layers in a plot. For this post, I got some great example from [1].

Scenario:

Concert has been use in many different structures such as bridge, apartment, roadways. For safety reason, the strength of concrete matters. Concrete strength depends on the materials that are used, such as: Cement, Slag, Ash, water.

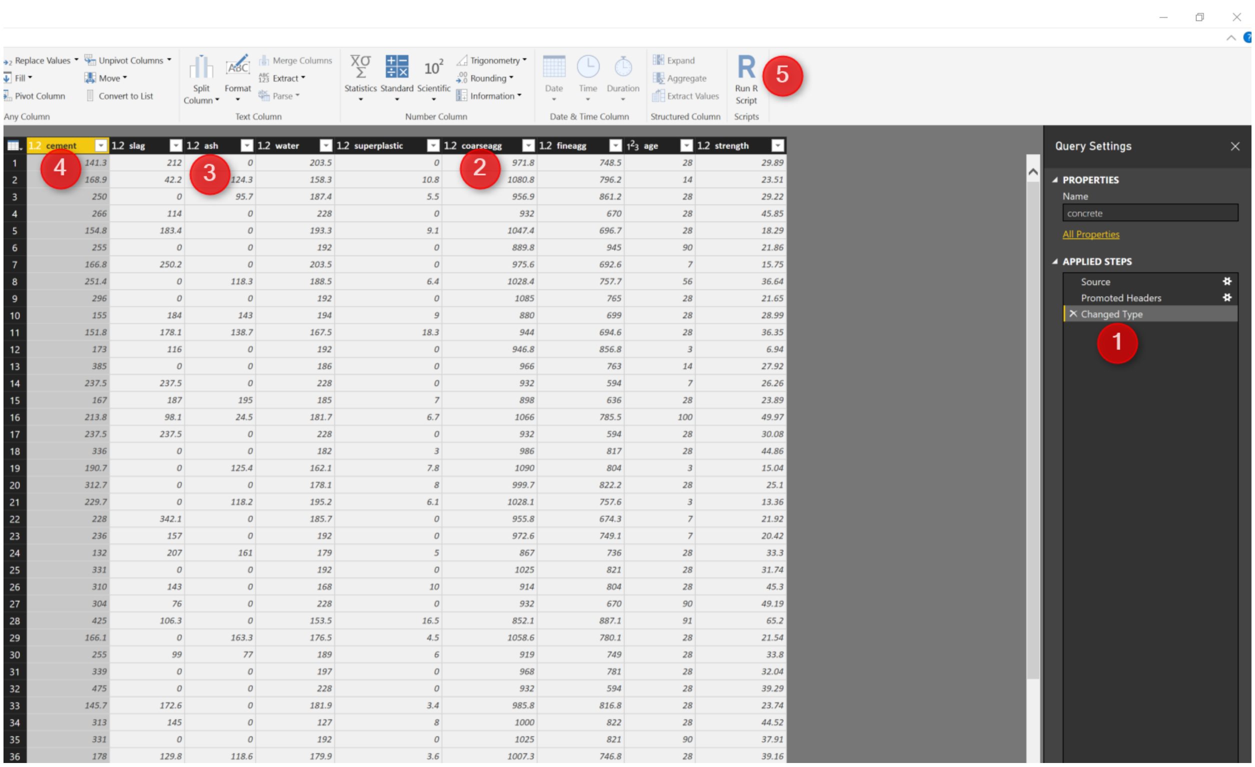
Our dataset below shows the ingredients of concrete.

A	B	C	D	E	F	G	H	I	J
cement	slag	ash	water	superplasti	coarseagg	fineagg	age	strength	
141.3	212	0	203.5	0	971.8	748.5	28	29.89	
168.9	42.2	124.3	158.3	10.8	1080.8	796.2	14	23.51	
250	0	95.7	187.4	5.5	956.9	861.2	28	29.22	
266	114	0	228	0	932	670	28	45.85	
154.8	183.4	0	193.3	9.1	1047.4	696.7	28	18.29	
255	0	0	192	0	889.8	945	90	21.86	
166.8	250.2	0	203.5	0	975.6	692.6	7	15.75	
251.4	0	118.3	188.5	6.4	1028.4	757.7	56	36.64	
296	0	0	192	0	1085	765	28	21.65	
155	184	143	194	9	880	699	28	28.99	
151.8	178.1	138.7	167.5	18.3	944	694.6	28	36.35	
173	116	0	192	0	946.8	856.8	3	6.94	
385	0	0	186	0	966	763	14	27.92	
237.5	237.5	0	228	0	932	594	7	26.26	
167	187	195	185	7	898	636	28	23.89	
213.8	98.1	24.5	181.7	6.7	1066	785.5	100	49.97	
237.5	237.5	0	228	0	932	594	28	30.08	
336	0	0	182	3	986	817	28	44.86	
190.7	0	125.4	162.1	7.8	1090	804	3	15.04	
312.7	0	0	178.1	8	999.7	822.2	28	25.1	
229.7	0	118.2	195.2	6.1	1028.1	757.6	3	13.36	
228	342.1	0	185.7	0	955.8	674.3	7	21.92	
236	157	0	192	0	972.6	749.1	7	20.42	
132	207	161	179	5	867	736	28	33.3	
331	0	0	192	0	1025	821	28	31.74	
310	143	0	168	10	914	804	28	45.3	
304	76	0	228	0	932	670	90	49.19	
425	106.3	0	153.5	16.5	852.1	887.1	91	65.2	
166.1	0	163.3	176.5	4.5	1058.6	780.1	28	21.54	
255	99	77	189	6	919	749	28	33.8	
339	0	0	197	0	968	781	28	32.04	
475	0	0	228	0	932	594	28	39.29	
145.7	172.6	0	181.9	3.4	985.8	816.8	28	23.74	
313	145	0	127	8	1000	822	28	44.52	
331	0	0	192	0	1025	821	90	37.91	
178	129.8	118.6	179.9	3.6	1007.3	746.8	28	39.16	
165	0	143.6	163.8	0	1005.6	900.9	14	16.88	
277.2	97.8	24.5	160.7	11.2	1061.7	782.5	14	47.71	
325	0	0	184	0	1063	783	7	17.54	

concrete

We are going to predict the concrete strength using neural network. Neural network can be used for predict a value or class, or it can be used for predicting multiple items. In this example, we are going to predict a value, that is concrete strength.

Data is first loaded in power bi, I am going to write some R codes in "Query Editor". First, we need to do some data transformations. As you can see in the below picture number 2,3 and 4, data is not on the same scale, we need to do data normalization before applying any machine learning. I am going to write a code for that (Already explained the normalization in post KNN). So, to write some R codes, I just click on the R transformation component (number 5).



I have used the below codes to normalized the dataset.

```
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

concrete_norm <- as.data.frame(lapply(dataset, normalize))
```

The same as any predictive model first we should provide some set of data for training and the other for testing as below.

```
concrete_train <- concrete_norm[1:773,]
```

```
concrete_test <- concrete_norm[774:1030,]
```

Next, I am going to call a package for Neural network that has been used a lot, name as "neuralnet". There are other packages for this purpose. I first install it using intall.packages command in my Rstudio.

```
library("neuralnet")
        concrete_model <- neuralnet(strength ~ cement + slag
        + ash + water + superplastic + coarseagg + fineagg + age,
        data = concrete_train)
```

This package has a function name (neuralnet) that create a model.

next, I am going to run the model against the training dataset for all 8 attributes as below

```
model_results <- compute(concrete_model, concrete_test[1:8])
predicted_strength <- model_results$net.result
```

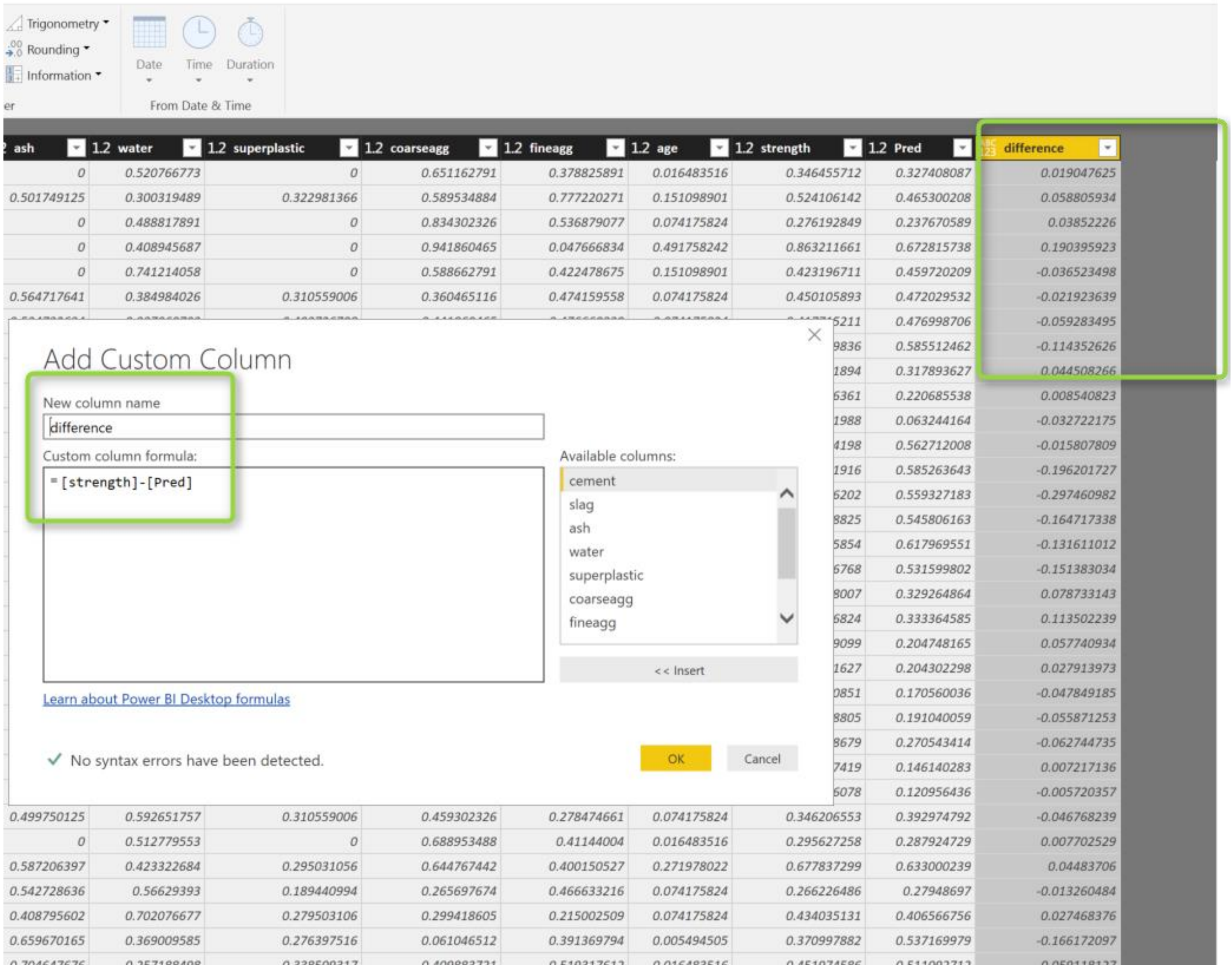
finally I create an output data frame dataset to show the result in Power BI

```
output<-dataset[774:1030,]
output$Pred<-predicted_strength
```

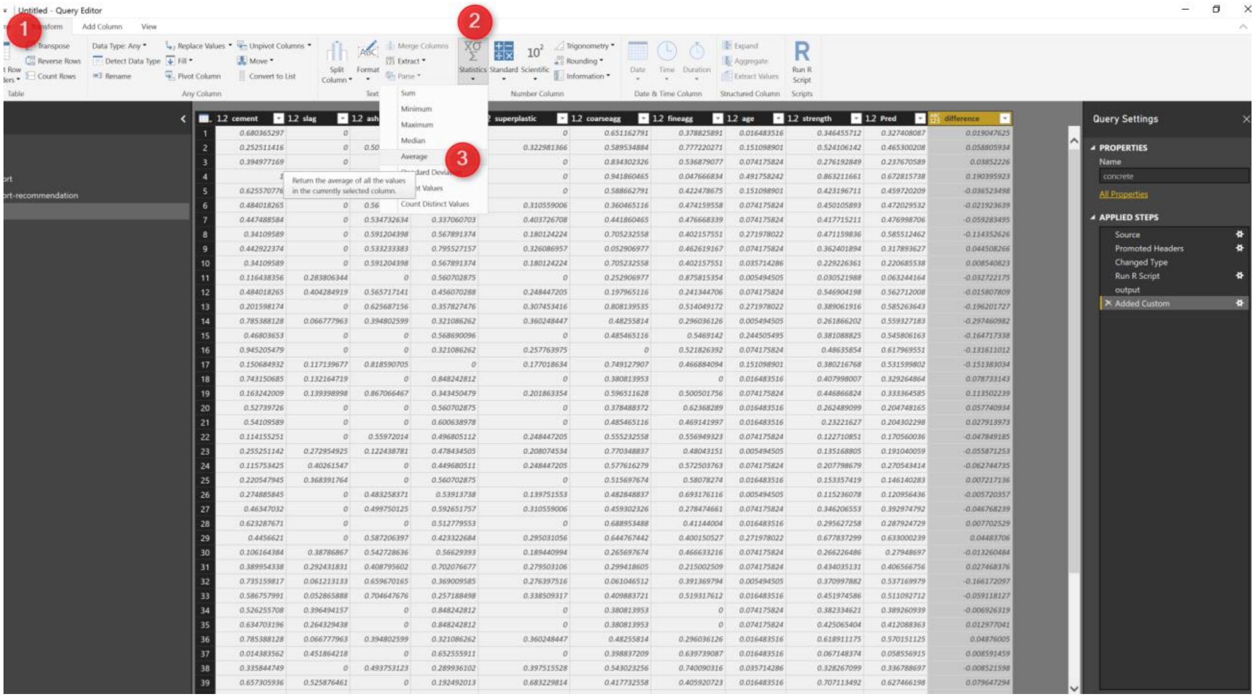
The output has been shown in the below picture. column 9 (strength) shows the real concrete strength, while the column 10 (pred) shows the prediction from neural net.

	cement	slag	ash	water	superplastic	coarseagg	fineagg	age	strength	Pred
1	400	0	0	187	0	1025	745	7	30.14	0.325850288
2	212.6	0	100.4	159.4	10.4	1003.8	903.8	56	44.4	0.465632258
3	275	0	0	183	0	1088	808	28	24.5	0.237054084
4	540	0	0	173	0	1125	613	180	71.62	0.671002974
5	376	0	0	214.6	0	1003.5	762.4	56	36.3	0.459093044
6	314	0	113	170	10	925	783	28	38.46	0.472427289
7	298	0	107	164	13	953	784	28	35.86	0.477760887
8	251.4	0	118.3	192.9	5.8	1043.6	754.3	100	40.15	0.587108177
9	296	0	106.7	221.4	10.5	819.2	778.4	28	31.42	0.315075624
10	251.4	0	118.3	192.9	5.8	1043.6	754.3	14	20.73	0.220328358
11	153	102	0	192	0	888	943.1	3	4.78	0.068072786
12	314	145.3	113.2	178.9	8	869.1	690.2	28	46.23	0.563779143
13	190.3	0	125.2	166.6	9.9	1079	798.9	100	33.56	0.587221586
14	446	24	79	162	11.6	967	712	3	23.35	0.560418925
15	307	0	0	193	0	968	812	90	32.92	0.546866614
16	516	0	0	162	8.3	801	802	28	41.37	0.618317098
17	168	42.1	163.8	121.8	5.7	1058.7	780.1	56	32.85	0.53454282
18	427.5	47.5	0	228	0	932	594	7	35.08	0.327606968
19	173.5	50.1	173.5	164.8	6.5	1006.2	793.5	28	38.2	0.333152204
20	333	0	0	192	0	931.2	842.6	7	23.4	0.203143343
21	339	0	0	197	0	968	781	7	20.97	0.203289536
22	152	0	112	184	8	992	816	28	12.18	0.171602395
23	213.8	98.1	24.5	181.7	6.7	1066	785.5	3	13.18	0.191031479
24	152.7	144.7	0	178.1	8	999.7	822.2	28	19.01	0.269408394
25	198.6	132.4	0	192	0	978.4	825.5	7	14.64	0.146753953
26	222.4	0	96.7	189.3	4.5	967.1	870.3	3	11.58	0.12265713
27	305	0	100	196	10	959	705	28	30.12	0.392717936
28	375	0	0	186	0	1038	758	7	26.06	0.286421657
29	297.2	0	117.5	174.8	9.5	1022.8	753.5	100	56.74	0.633703197
30	148.5	139.4	108.6	192.7	6.1	892.4	780	28	23.7	0.278002147
31	272.8	105.1	81.8	209.7	9	904	679.7	28	37.17	0.405820072
32	424	22	132	168	8.9	822	750	3	32.11	0.537589428
33	359	19	141	154	10.9	942	801	7	38.61	0.511604545
34	332.5	142.5	0	228	0	932	594	28	33.02	0.388373021
35	380	95	0	228	0	932	594	28	36.45	0.411418597
36	446	24	79	162	11.6	967	712	7	52.01	0.571282449
37	108.3	162.4	0	203.5	0	938.2	849	7	7.72	0.06415345
38	249.1	0	98.8	158.1	12.8	987.8	889	14	28.68	0.335272386
39	389.9	189	0	145.9	22	944.7	755.8	7	59.09	0.6278761
40	250	0	95.7	191.8	5.3	948.9	857.2	14	24.66	0.200271748

I created a custom column to see the differences between predicted and the orinal value as you see in below picture:

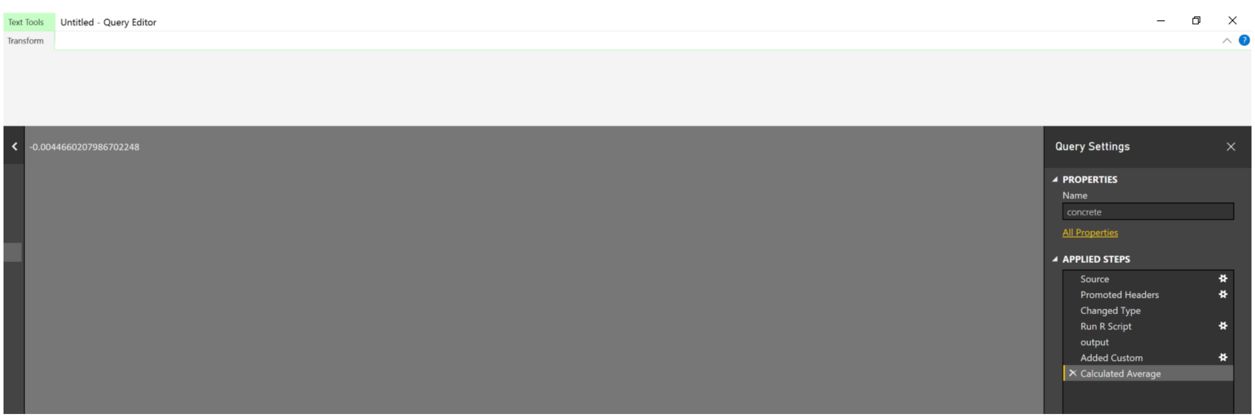


There are not much difference. If you just go to transform (number 1 in the below picture) and then choose the Average (number 2 and 3),



you will see the difference is not that much in average=-0.0044

which shows the prediction is good



However, you may be interested to see the plot in visualization and see the hidden nodes and other information like weights.

so I am going to report area and just copy and paste the code I run for the neural network

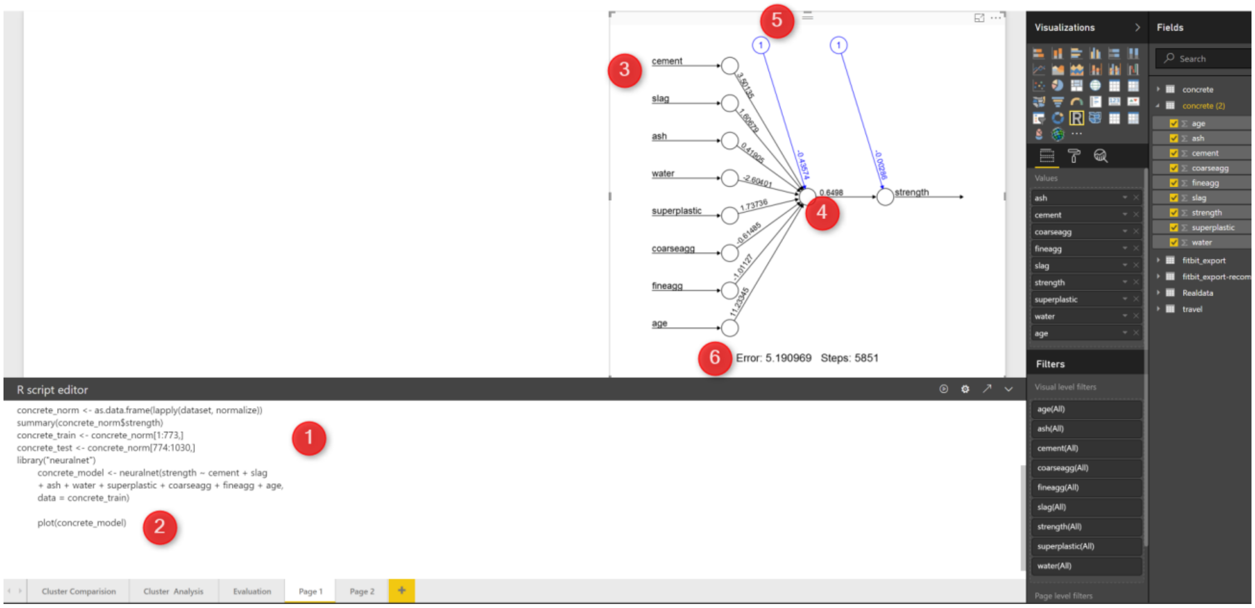
```
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
```

```
concrete_norm <- as.data.frame(lapply(dataset, normalize))
summary(concrete_norm$strength)
concrete_train <- concrete_norm[1:773,]
concrete_test <- concrete_norm[774:1030,]
library("neuralnet")
        concrete_model <- neuralnet(strength ~ cement + slag
        + ash + water + superplastic + coarseagg + fineagg + age,
        data = concrete_train)
```

just I add plot to show the model

```
plot(concrete_model)
```

the output will be like :



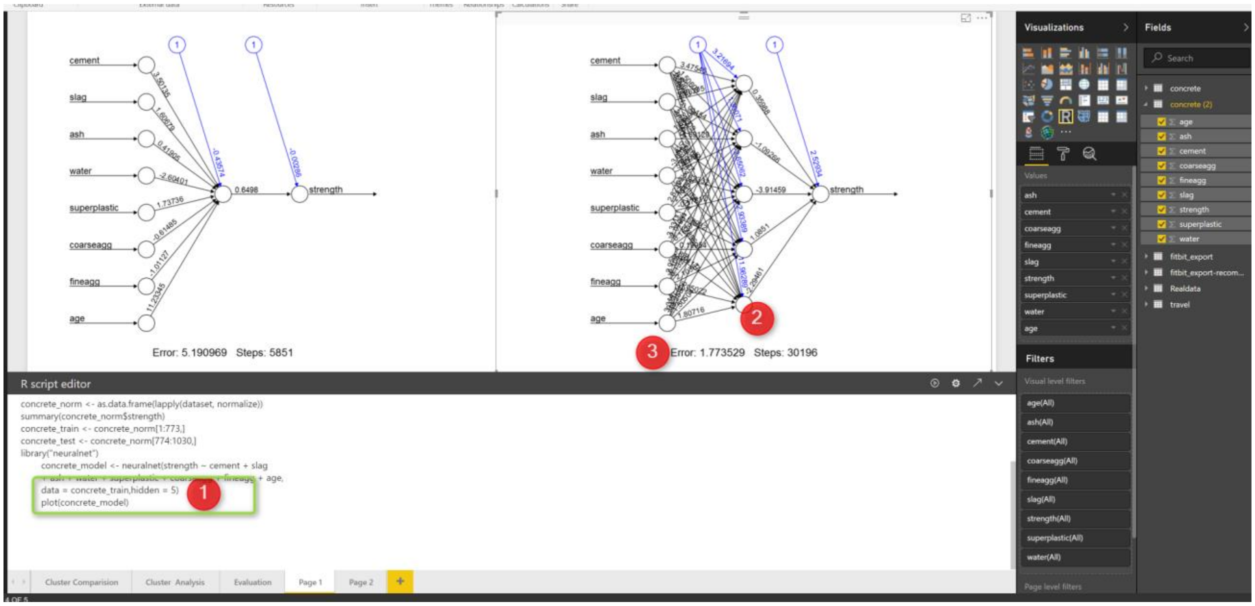
as you can see in the above picture, in number 1, I copied and pasted the code (I change the dataset name). then I simply use the plot (number 2) to show the neural network.

You see the network structure in visualization section. The number 3 show the input that we consider to predict the strength of the concrete. number 5 is a intercept or biased number. number 4 is the strength weight. and in number 6 at the bottom of the page you will the error and how many steps has gone to reach. this is a very simple network, just one hidden node, lets add some hidden node to this model just by changing the code as below :

```
concrete_model <- neuralnet(strength ~ cement + slag
```

```
+ ash + water + superplastic + coarseagg + fineagg + age,
data = concrete_train,hidden = 5)
```

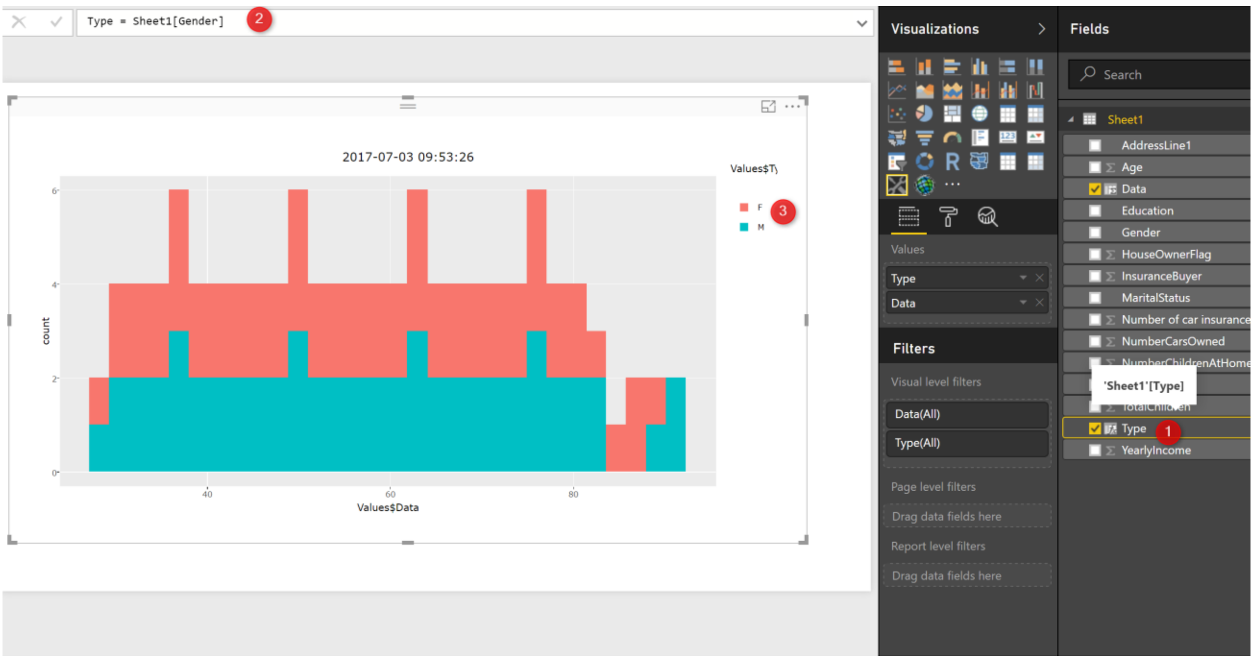
The only change is to add parameter "Hidden" to the neural net function (number 1). Then run the code and you will see another network that has 5 different hidden node. If you look at the error, you will see it is decreased so much! So always having some more hidden layer can be helpful but not that much, I could not find a rule to identify number of hidden node, but if you have any hidden node then we have the problem of over fitting see related post about over fitting



[1]Machine Learning with R,Brett Lantz, Packt Publishing,2015

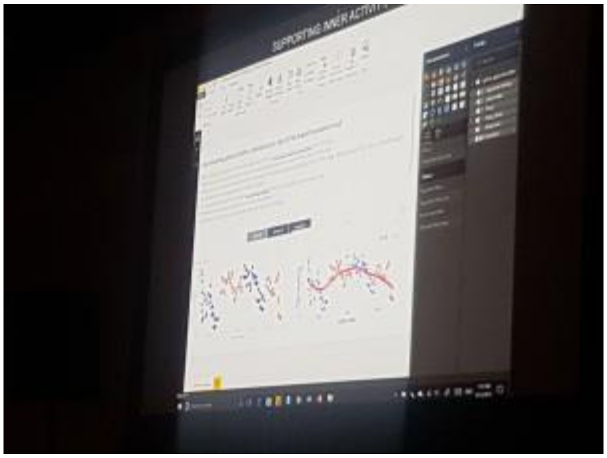
19-Interactive Charts using R and Power BI: Create Custom Visual Part 1

Published Date : July 3, 2017



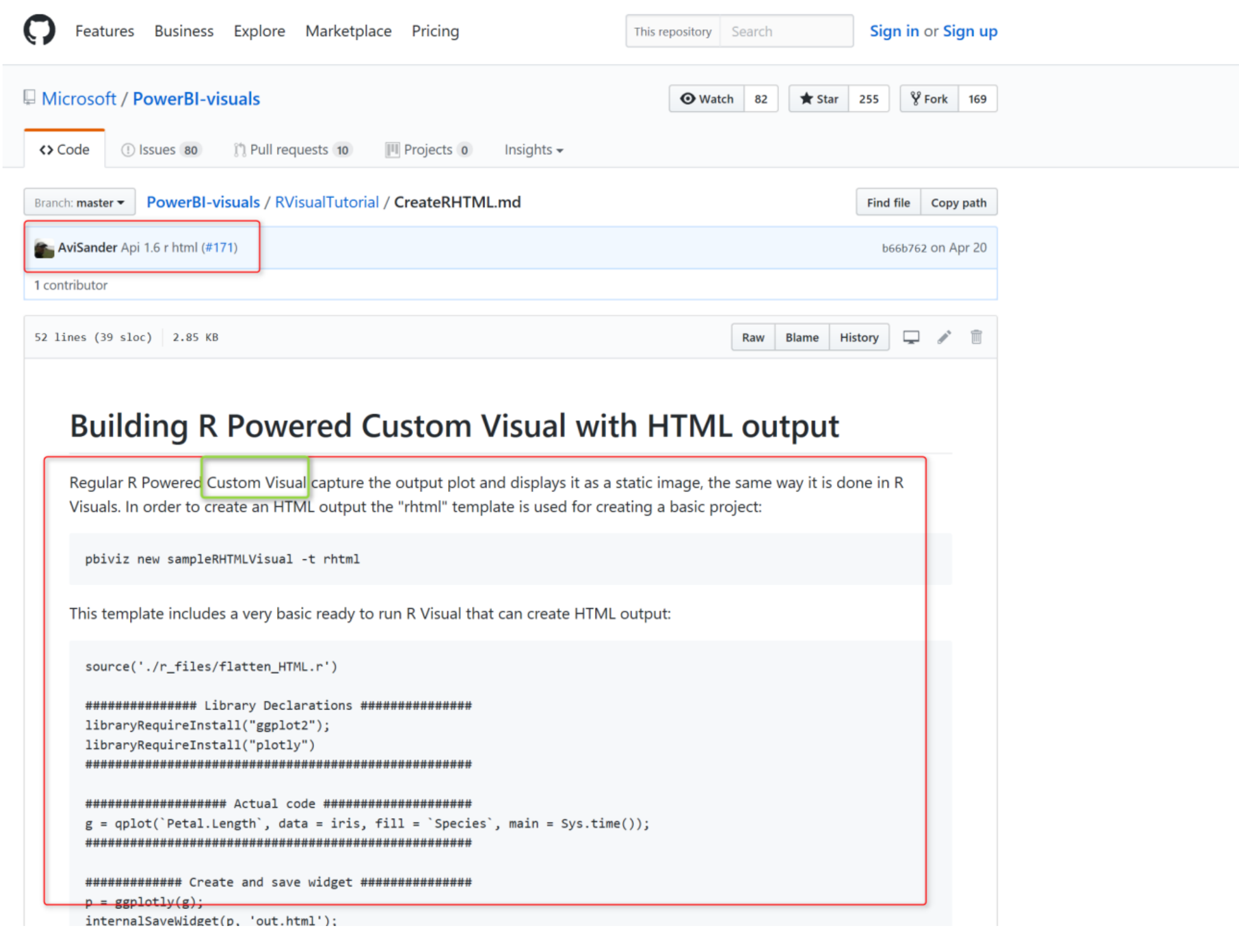
There is a possibility to use Plotly packages in Power BI.

So what is Plotly: Plotly is an R package for creating interactive web-based graphs via the open source JavaScript graphing library:https://plot.ly/r/getting-started/



this feature has been added recently and had been announce by Christian Berg in Microsoft Data insight summit 2017.

I started to search about it and I found an article by Avi Sander (https://github.com/Microsoft/PowerBI-visuals/blob/master/RVisualTutorial/CreateRHTML.md) in GitHub.



from the article I understand that I should create a custom Visual with R packages but How? I asked My dear friend Rudiger Hein about it, and he sent me a nice video created by [Stephanie Locke](#) and now I am going to have some post series on how to have more nice charts using Plotly in Power BI!

To have Plotly inside Power BI you should create a custom visual from your R codes.

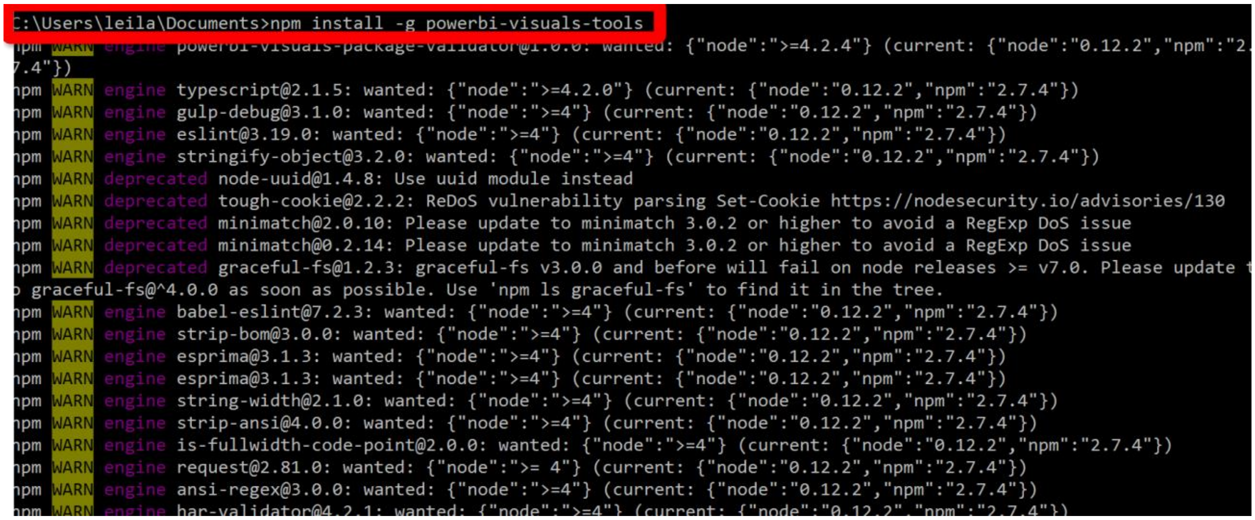
In this post I am going to show how to set up and create custom R visual inside Power BI, and in the next posts I will show some other nice charts!

1-first Step

First you should download NodeJS 4.0+ Required (5.0 recommended) from https://nodejs.org/

Then in command Prompt write the below codes to install it.

npm install -g powerbi-visuals-tools



you should see above message after writing "npm install -g powerbi-visuals-tools" in command prompt.

To confirm it was installed correctly you can run the command without any parameters which should display the help screen.

pbiviz

then you should see the information about Power BI Custom Visual Tools

```
:\Users\leila\Documents>pbiviz
   oms/+osyhdhyso/
   ym/       /+oshddhys+/
   ym/              /+oyhddhyo+/
   ym/                     /osyhdho
   ym/                           sm+
   ym/              yddy         om+
   ym/          shho /mmmm/      om+
    /    oys/ +mmmm /mmmm/       om+
   oso  ommmh +mmmm /mmmm/       om+
  ymmmy smmmh +mmmm /mmmm/       om+
  ymmmy smmmh +mmmm /mmmm/       om+
  ymmmy smmmh +mmmm /mmmm/       om+
  +dmd+ smmmh +mmmm /mmmm/       om+
        /hmdo +mmmm /mmmm/ /so+//ym/
              /dmmh /mmmm/ /osyhhy/
                //   dmmd
                      ++

      PowerBI Custom Visual Tool


 Usage: pbiviz [options] [command]


 Commands:

   new [name]        Create a new visual
   info              Display info about the current visual
   start             Start the current visual
   package           Package the current visual into a pbiviz file
   validate [path]   Validate pbiviz file for submission
   update [version]  Updates the api definitions and schemas in the current visual. Changes the version if specified
   help [cmd]        display help for [cmd]

 Options:

   -h, --help      output usage information
   -V, --version   output the version number
   --install-cert  Install localhost certificate


C:\Users\leila\Documents>
```

Yes, now you have the pbiviz in your machine

2-Second Step

I am going to follow the steps proposed by Avi Sander. we are going to create a “rhtml” template.

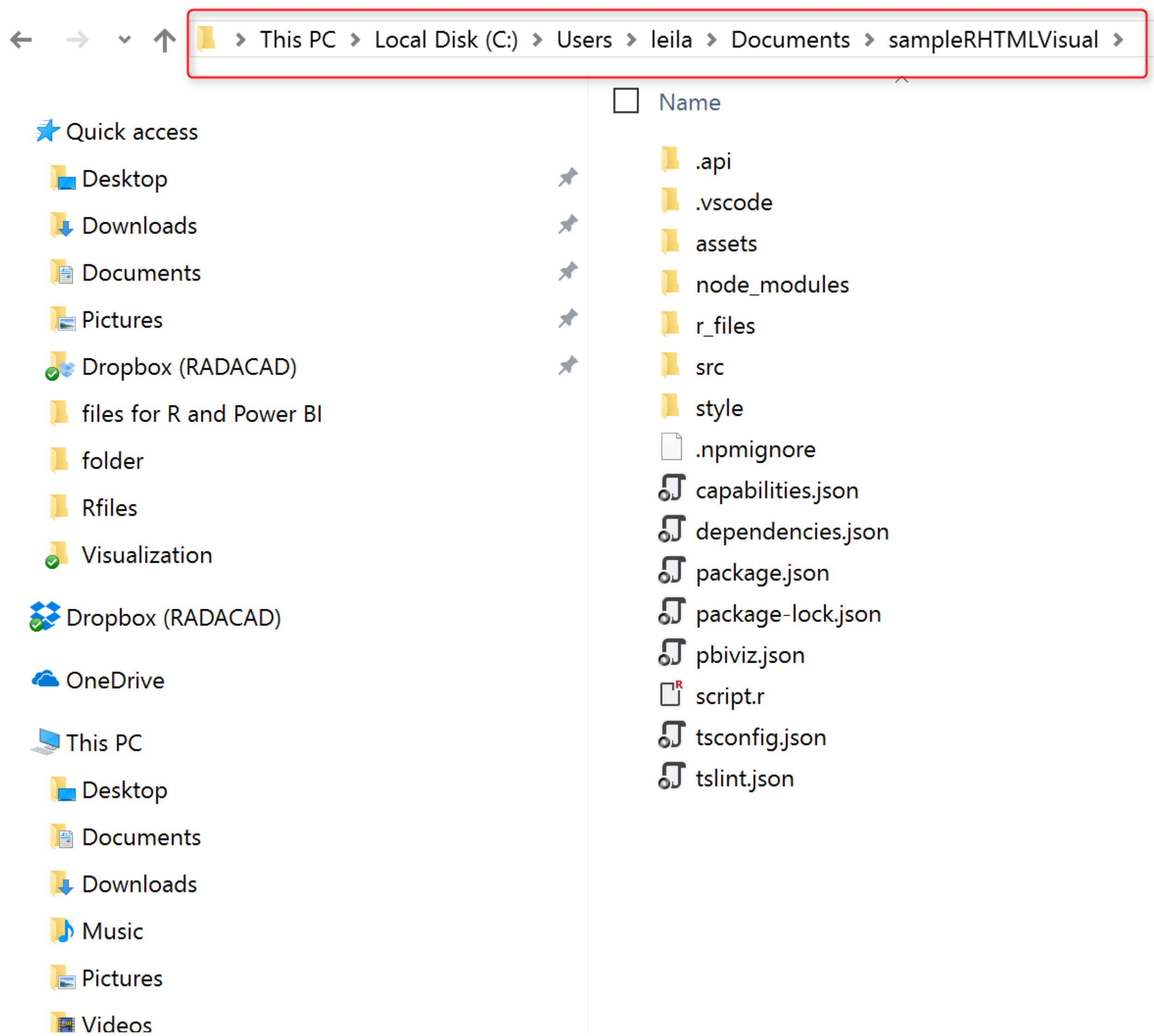
n command prompt type:

pbiviz new sampleRHTMLVisual -t rhtml

Now you should see that a new custom visual is installing and a new package has been created, but what is inside the packages

```
C:\Users\leila\Documents>pbiviz new sampleRHTMLVisual -t rhtml
 info   Creating new visual
 info   Installing packages...
 info   Installed packages.
 done   Visual creation complete

C:\Users\leila\Documents>
```

I check the folder that this packages has been create



this is a folder that provides a template for me to create other R custom visuals, please check the file “script.r” inside the folder.

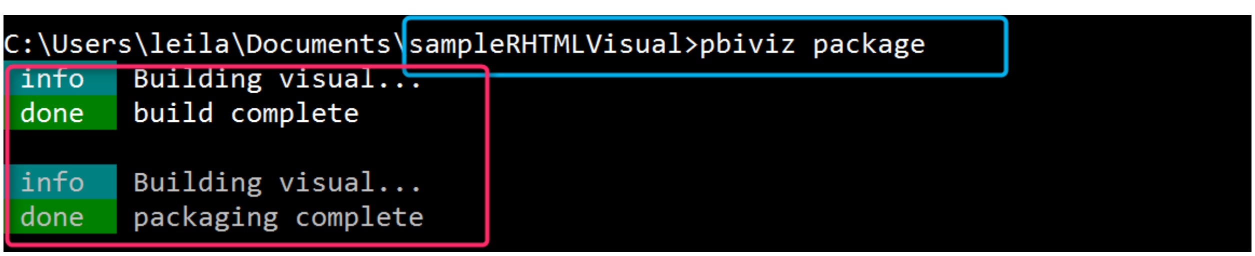
```
source('./r_files/flatten_HTML.r')

############### Library Declarations ###############
libraryRequireInstall("ggplot2");
libraryRequireInstall("plotly")
####################################################

################### Actual code ####################
g = qplot(`Petal.Length`, data = iris, fill = `Species`, main = Sys.time());
####################################################

############# Create and save widget ###############
p = ggplotly(g);
internalSaveWidget(p, 'out.html');
####################################################
```

From the above codes, you see that we need library “Plotly and ggplot2 to draw a simple ggplot2 chart that is more interactive because of Plotly.

The dataset is hard coded here for “iris” that is an open source dataset in R. The plot gets the data from iris dataset and shows the Petal.Lengh and the Species there. then we use the ggplotly function to show the data.

Now, we have an R scripts, which I am going to create a package from it first, and then I will write my own codes to create different charts.

I back to command prompt and I type just “pbiviz package” in the folder.



I open a new Power BI project and as you can see in the below video, you will be able to work with it now!

https://www.youtube.com/watch?v=uLikWluqg54&feature=youtu.be

3- Third Step

Now I am going to change the code to have my own R scripts that able to work with the code

I just change the code a bit as below:

```
source('./r_files/flatten_HTML.r')

############### Library Declarations ###############
libraryRequireInstall("ggplot2");
libraryRequireInstall("plotly")
####################################################

################### Actual code ####################
g = qplot(Values$Data, data = Values, fill = Values$Type, main = Sys.time());
####################################################

############# Create and save widget ###############
p = ggplotly(g);
internalSaveWidget(p, 'out.html');
```

As you can see in above code, I have changed the data field as value, and also the fill with Type, so in any code if I have a Data and type column (can be created via custom column in Power bi) I able to use the below codes

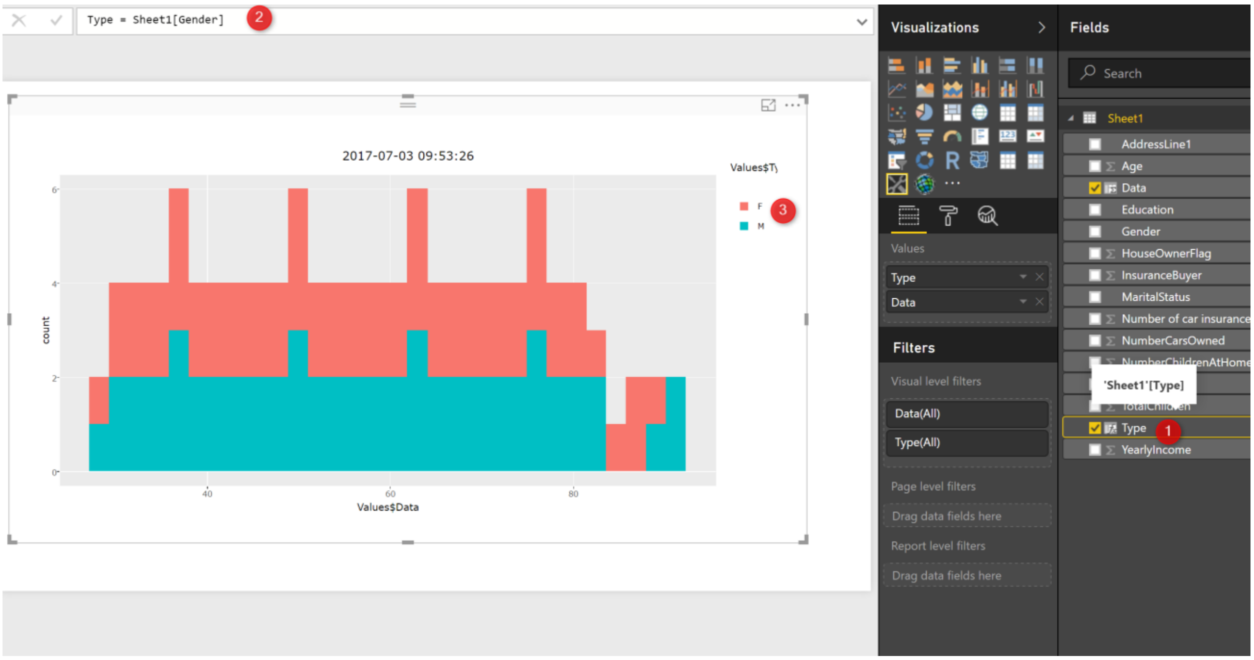
```
g = qplot(Values$Data, data = Values, fill = Values$Type, main = Sys.time());
```

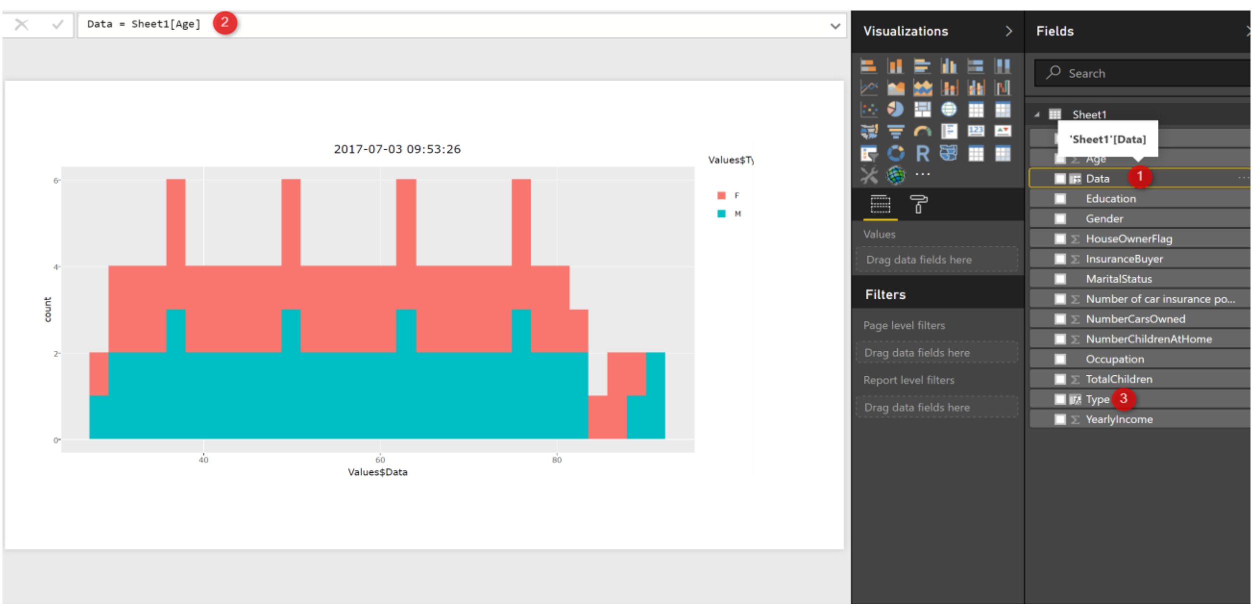
I just save the R script file, then I run the package again

"pbiviz package" in command prompt, then I add the chart to power bi as a custom visual and now I have the chart in my power bi visual!

After importing the custom visual, now I am going to show the gender and age of people from my insurance data.

I have to create a custom column with the name Data for age of people and another custom column with name Type for people age. Picture below, I created a column name Type that I put the gender for it, then I created another custom column for Age of people and I name it Data (in next picture)





I have a chart that it is interactive and you able to see the data details (see below video)

https://www.youtube.com/watch?v=wkRq6NiCf1g&feature=youtu.be

in next posts, I will show some exciting chart using Plotly in Power BI

References: https://www.npmjs.com/package/powerbi-visuals-tools

https://github.com/Microsoft/PowerBI-visuals/blob/master/RVisualTutorial/CreateRHTML.md

https://www.youtube.com/watch?v=_zd-UGfD2Os

20-Interactive Charts using R and Power BI: Create Custom Visual Part 2

Published Date : July 5, 2017

In the last post, I have explained the main process of creating the R custom visuals inside Power BI.

In this post, I am going to show how to:

1- Have more custom visuals

2- Different charts that we can have in Power BI

3- Explain some issues

Have more custom visuals

To have more R visuals, it is now possible to use Plotly Packages by following the Custom Visual Process inside Power BI.

In the last post, I did not explain how to create a custom visual for R and Plotly and how to have it inside the Power BI.

I decided to show all different R charts, that I have used before, as Interactive visual.

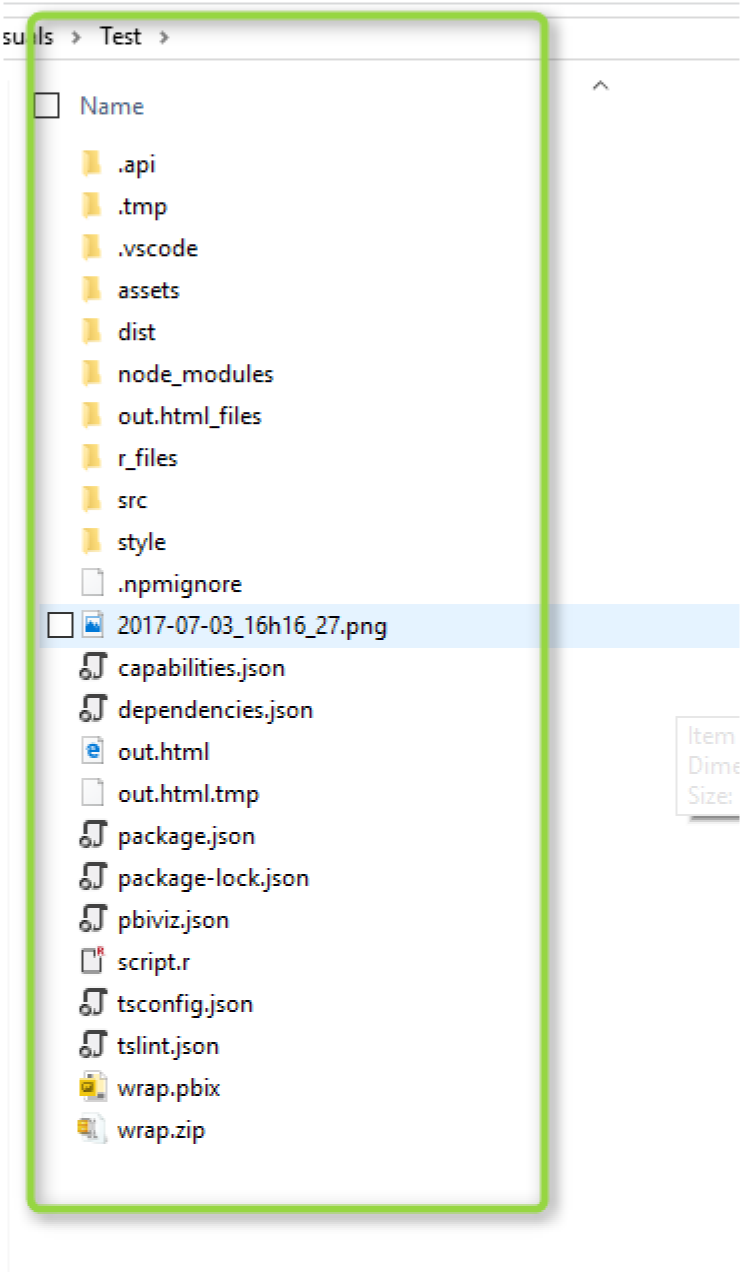
I am going to create a Jitter chart to show three variables in a chart. The main R codes are in my previous Post about having more charts in Power BI using R graphs.

I am going to show How to create two different R Custom Visual one for Facet chart and the other for normal Jitter Chart

Jitter Chart

What I done: First I went back to the main folder that we have from the sample Power BI package,

1- I copy the folder to create a new custom visual (as it was my first attempts I call it Test: D)



2-I change the "pbiviz.json" files content: as below code.

```
{
 "visual": {
  "name": "Test",
  "displayName": " Test ",
  "guid": "TestI216CAF192F6C439FAC22226710C3B3D4",
  "visualClassName": "Visual",
  "version": "1.0.0",
  "description": "",
```

```
    "supportUrl": "",
    "gitHubUrl": ""
  },
  "apiVersion": "1.7.0",
  "author": {
    "name": "",
    "email": ""
  },
  "assets": {
    "icon": "assets/icon.png"
  },
  "externalJS": [
    "node_modules/powerbi-visuals-utils-dataviewutils/lib/index.js"
  ],
  "style": "style/visual.less",
  "capabilities": "capabilities.json",
  "dependencies": "dependencies.json",
  "stringResources": []
}
```

3-Then I put my R scripts inside the existing file name “scripts.r” .

I have change the content as below to draw a simple jitter chart

The orange colour shows the changes I have made.

first: the main dataset will be stored inside the variable “Values” I put the value **Values$hwy** for the y axis and for x axis **Values$cty**. also I want to distinguish the number of cylinder of each car by different color **colour = Values$cyl**.

```
libraryRequireInstall("ggplot2");
libraryRequireInstall("plotly")
library("plotly")
library("ggplot2")
library("htmlwidgets")
####################################################

g=ggplot(Values, aes(x=c, y=Values$hwy,colour = Values$cyl)) + geom_jitter(size=4)
################### Actual code ####################
```

```
###################################################
############# Create and save widget ###############
p = ggplotly(g);
internalSaveWidget(p, 'out.html');
```

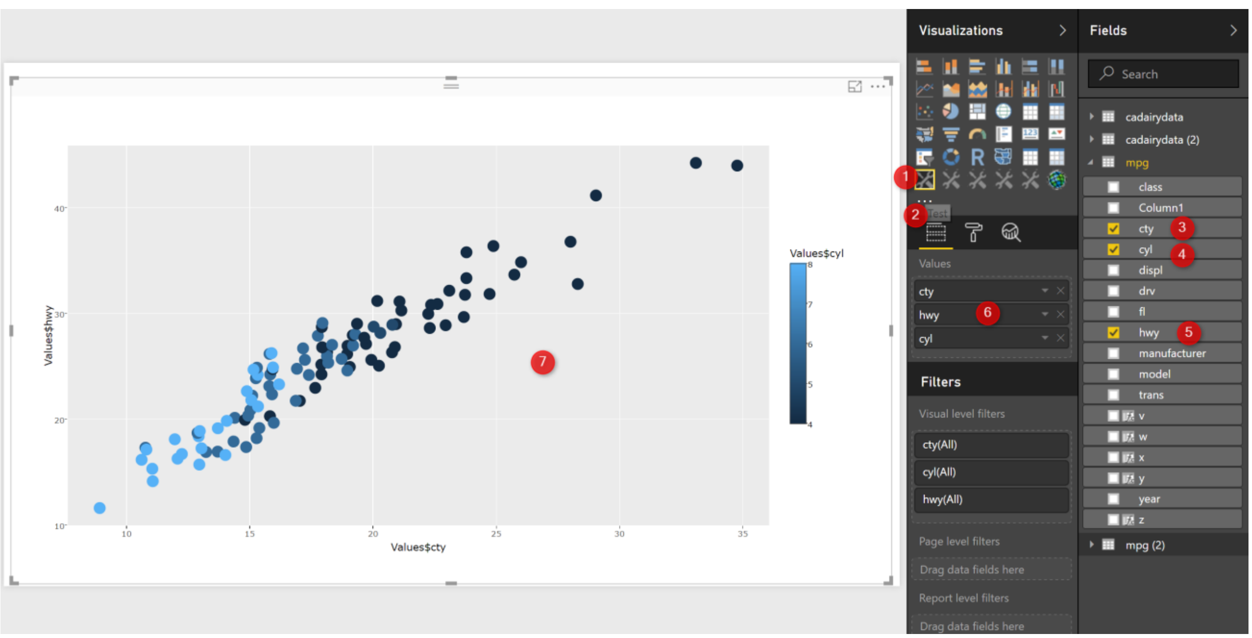
4-Now I can run the package and add it to power BI as a custom visual (as I have described in last post). See the blow picture. First, I have import the custom visual into power BI (number 1 and 2 in the below picture).

to do: I will show later how to change the icon of the custom visual.

Next the custom visual accepts the three-main variable name as "cty, hwy, and cyl". the variable should be name the same (number 3,4,5 and 6 in the below picture).

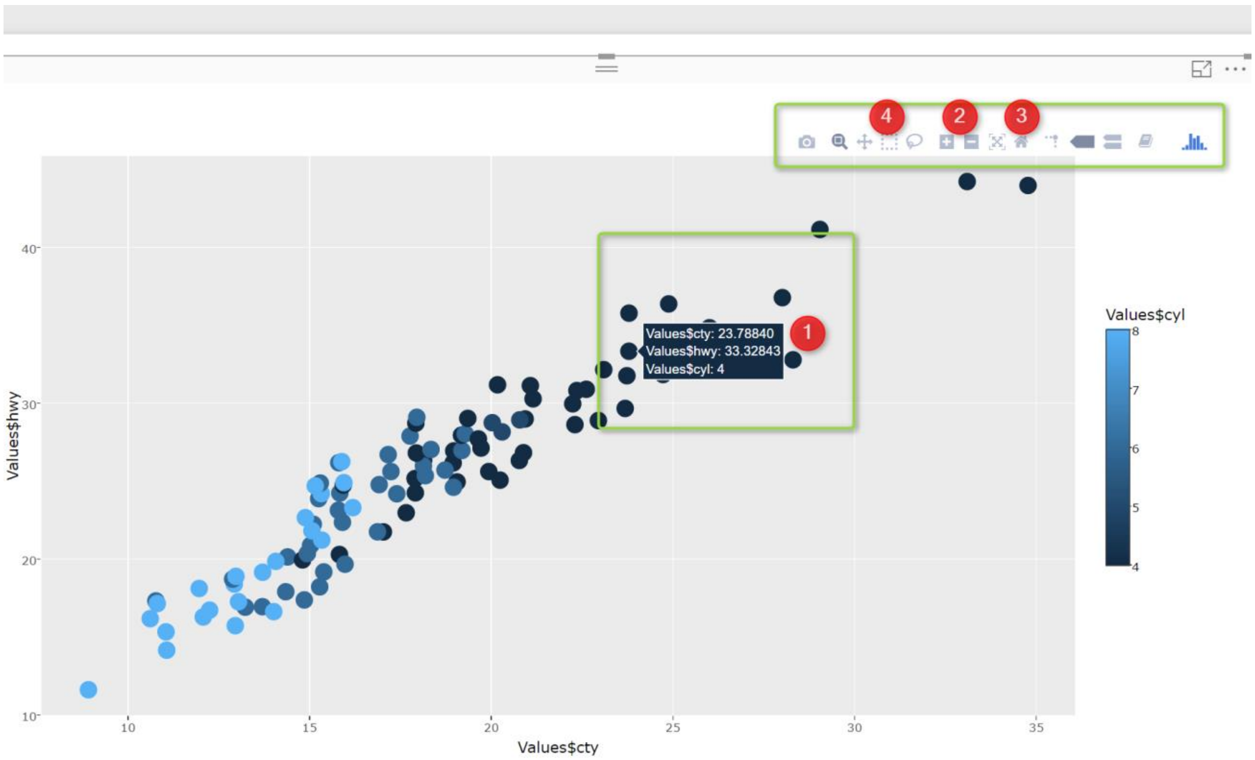
To do: in next chart I have replace them with (x,y,z,w,v) so it can be applied to all other charts and different datasets.

Finally, in number 7. you see the charts.

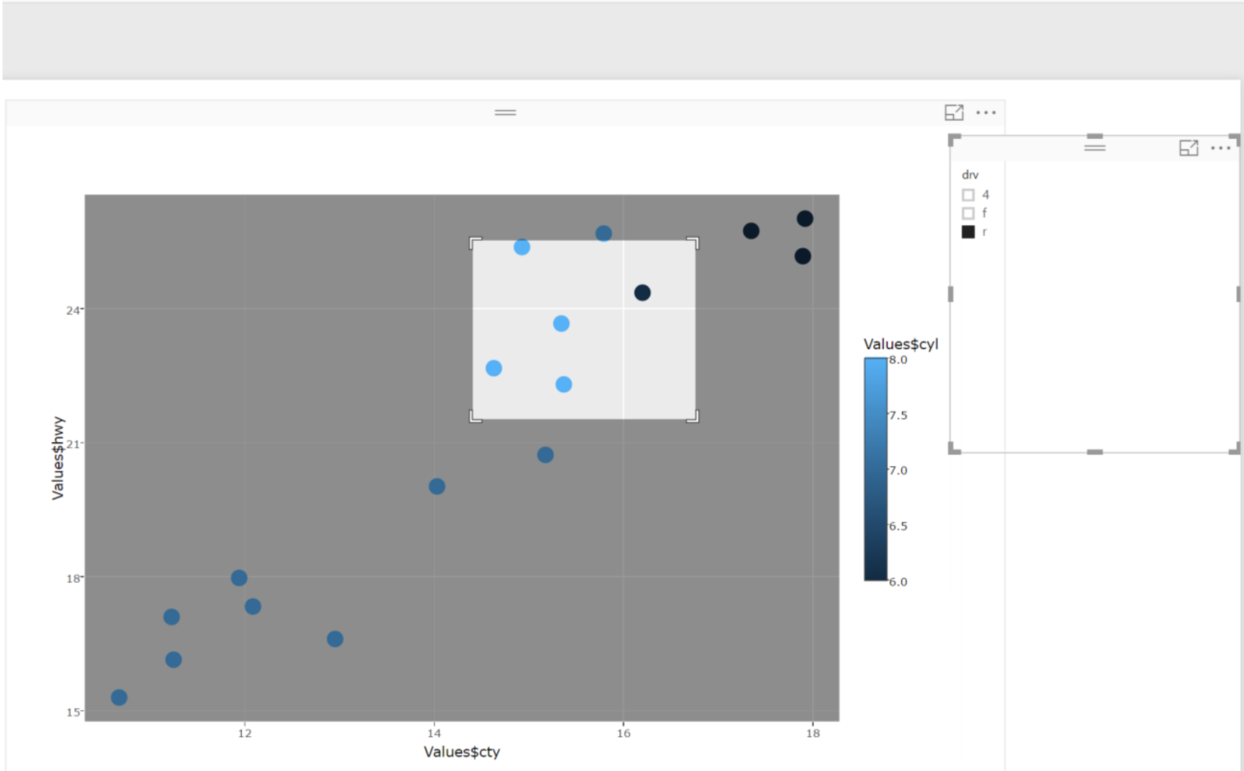


Now, just by hovering your mouse on charts you able to see the tooltips for speed in city, high way and number of cylinder (number 1).

moreover, there is a possibility to zoom in and zoom out and to select specific area of the charts.



I can select specific area of the chart and zoom in to see detail data, also this picture able to be interactive with Power BI slicer.



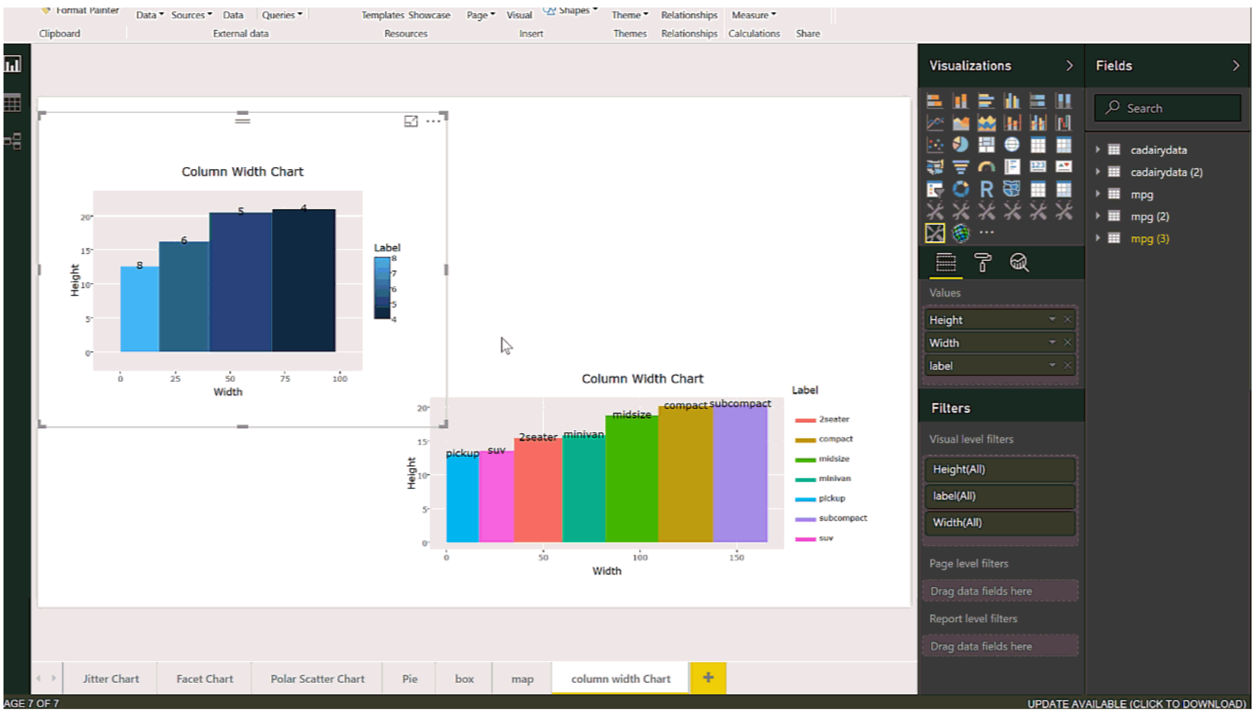
In the next post, I am going to show how to have more charts that we do not have normally in Power BI, make them as Custom Visual.

download the custom visual from below

[Jitter chart (23 downloads)]

21-Interactive Charts using R and Power BI: Create Custom Visual Part 3

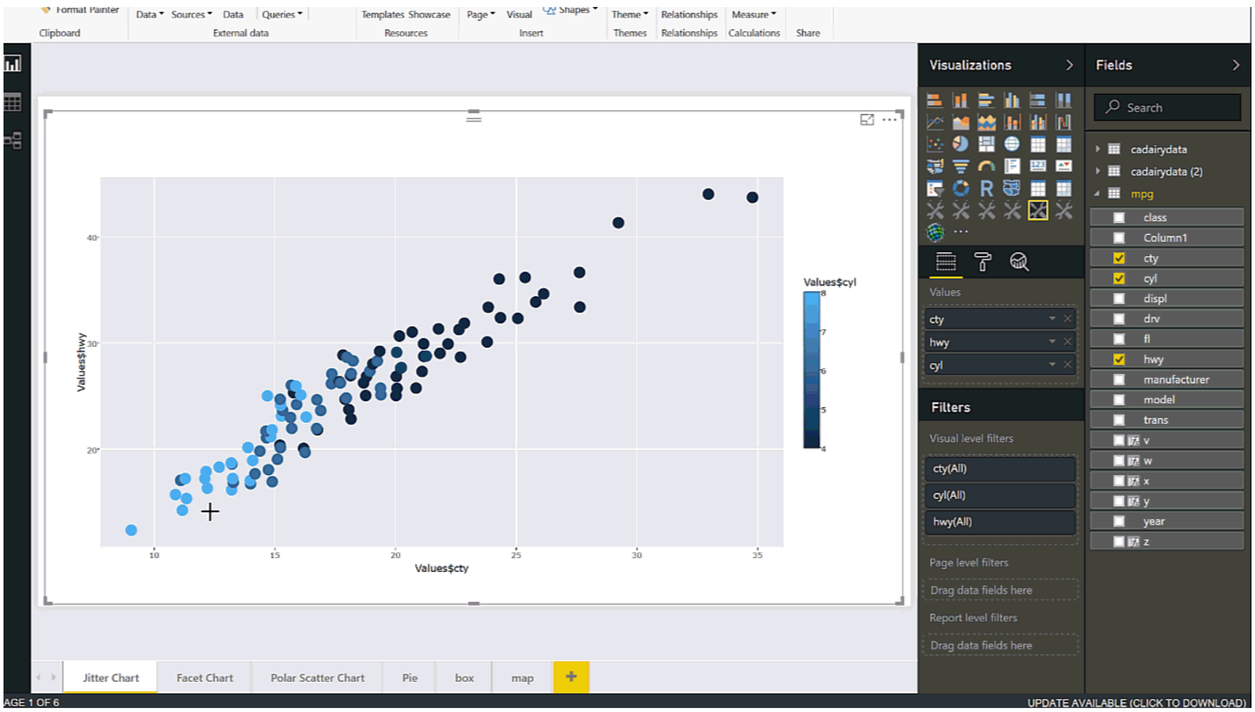
Published Date : July 10, 2017



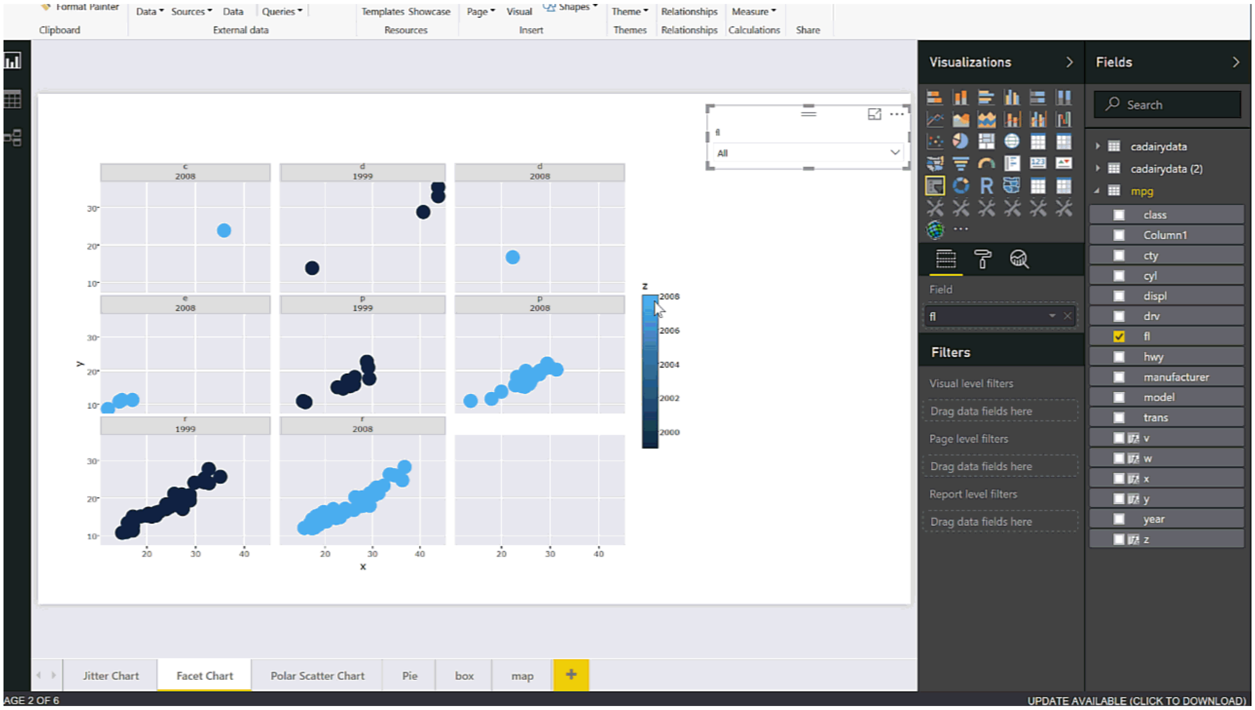
In the last two posts (Part 1 and 2), I have explained the main process of creating R custom Visual Packages in Power BI. There are some parts that still need improvement which I will do in the next posts. In this post, I am going to show different R charts that can be used in power BI and when we should use them for which data type, these are Facet jitter chart, Pie chart, Polar Scatter Chart, Multiple Box Plot, and Column Width Chart. I follow the same process I did in Post 1 and Post 2. Moreover, I add the related R scripts for each chart and will explain how and for what type of data use these graphs.

1-Jitter Chart

This chart is used to show all data points in a dataset. Three variables are shown at the same time in one chart: two numeric variables for x and y-axis and one factor variable with different colours. Picture below is a custom visual that shows the speed of the car in the city on the x axis, the car's speed in a high way in Y axis and the number of cylinders as factor variable in the chart legend.



It is possible to show 4 or 5 variables at the same time. One for the x-axis, y-axis, colour shade for factor data, two factor variables for Facet and different tiles. In the below picture you will see that I show the speed of the car in city and highway in x and y-axis, also I put the year of the cars into z variable. Moreover, I need two variables for different tiles. One for year and another factor variable for car's FL. This custom Visual get constant 5 variables as x,y,z,w, and v. I have a post on this chart, see http://radacad.com/have-more-charts-by-writing-r-codes-inside-power-bi-part-2

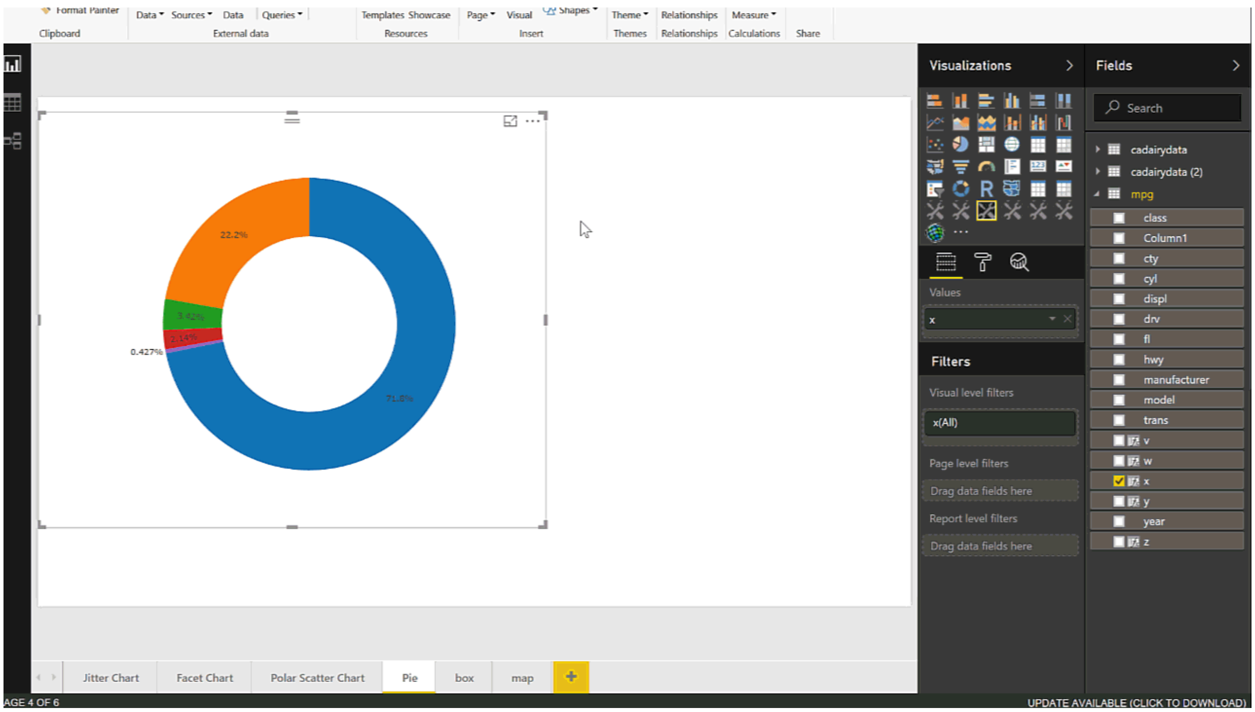


The code for creating chart has been shown in the below code

```
source('./r_files/flatten_HTML.r')

############### Library Declarations ###############
libraryRequireInstall("ggplot2");
libraryRequireInstall("plotly")
library("plotly")
library("ggplot2")
library("htmlwidgets")
####################################################
#g = plot_ly(mpg, x = mpg$cty, y = mpg$hwy, text = paste("Clarity: ", mpg$cyl),
#mode = "markers", color = mpg$cty, size = mpg$cty)
################### Actual code ####################
g=ggplot(Values, aes(x=x, y=y,color=z)) + geom_jitter(size=5)+facet_wrap(w~ v)
####################################################
#g =ggplot(mpg, aes(x=cty, y=hwy,color=cyl)) + geom_jitter(size=5)+facet_wrap(year~ drv)
############# Create and save widget ###############
p = ggplotly(g);
internalSaveWidget(p, 'out.html');
```

2-Pie Chart

Pie charts are able to show the composition of data. In this below example, I have shown how to show the composition of the car's speed in highway as a continues variable with grouping them based on FL of the car. This chart shows the labels inside the pie chart (in power bi it shows outside mainly)



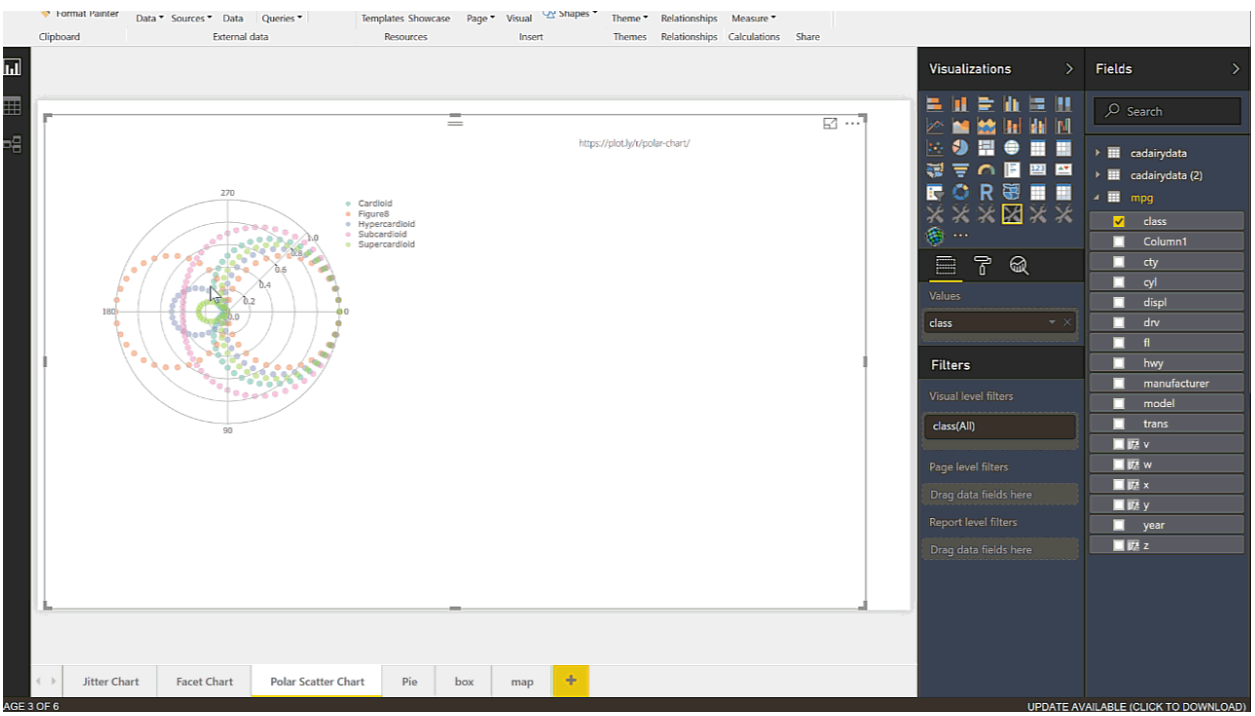
The code for generating the pie chart has been shown below

```
ource('./r_files/flatten_HTML.r')

############### Library Declarations ###############
libraryRequireInstall("ggplot2");
libraryRequireInstall("plotly")
library("plotly")
library("ggplot2")
library("htmlwidgets")
####################################################
#g = plot_ly(mpg, x = mpg$cty, y = mpg$hwy, text = paste("Clarity: ", mpg$cyl),
#mode = "markers", color = mpg$cty, size = mpg$cty)
#library(plotly)
```

```
# Get Manufacturer
g <- mpg %>%
  group_by(fl) %>%
  summarise(count = n()) %>%
  plot_ly(labels = ~fl, values = ~count) %>%
  add_pie(hole = 0.6)
  ############# Create and save widget ###############
p = ggplotly(g);
internalSaveWidget(p, 'out.html');
```

3-Polar Scatter Chart

This chart has been used to show two numeric variables, which one of the should have a wider range. For instance, one variable should be from 0 to 365 or 177 to -177 and the other variable should have a limited range for instance from 0 to 10 or from 0 to 1. We need another factor variable to show the colour. in the below gif, you will see we have a variable that ranges from 0 to 1 and the other one from 0 to 270. Also, we have a factor variable that shows the lines in the graph in different colours.



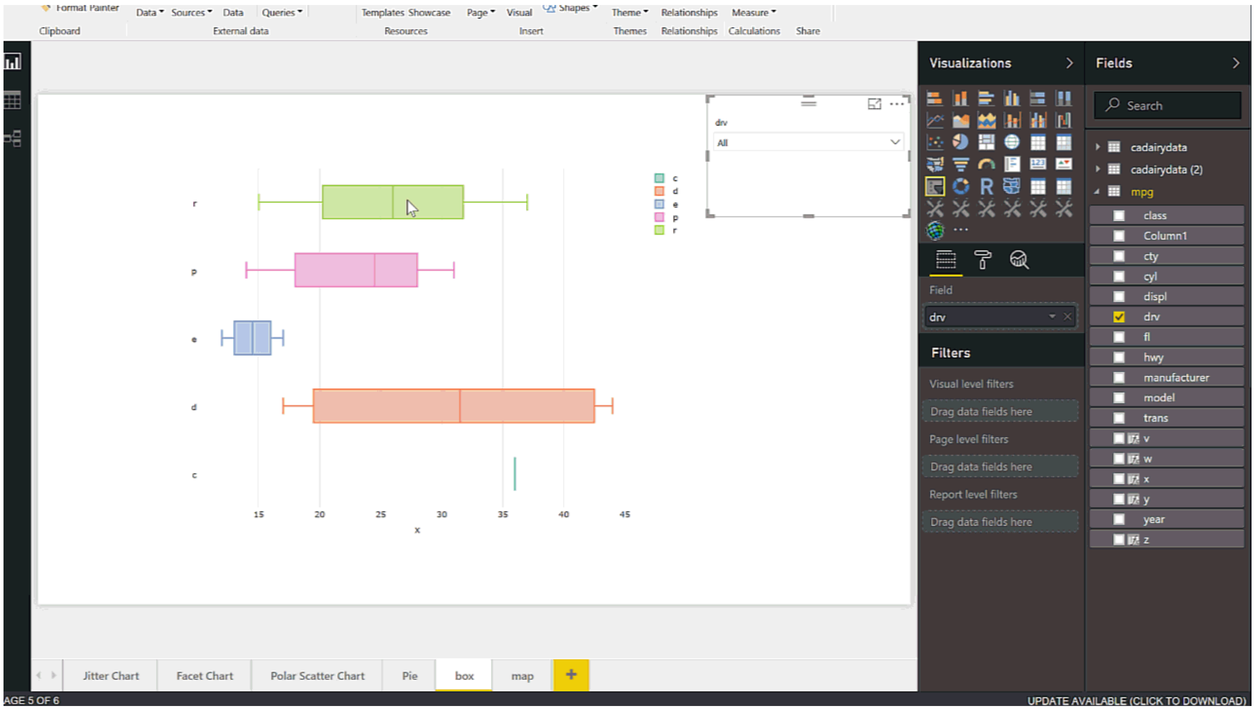
The code for generating the Polar chart has been shown below

```
source('./r_files/flatten_HTML.r')

############### Library Declarations ###############
libraryRequireInstall("ggplot2");
libraryRequireInstall("plotly")
library("plotly")
library("ggplot2")
library("htmlwidgets")
####################################################

#g=ggplot(Values, aes(x=Values$cty, y=Values$hwy,colour = Values$cyl)) + geom_jitter(size=4)
################### Actual code ####################

#p = ggplotly(g);

p <- plot_ly( plotly::mic, r = ~r, t = ~t, color = ~nms, alpha = 0.5, type = "scatter")

layout(p, title = " ", orientation = -90);

internalSaveWidget(p, 'out.html');
```

4-Box Plot

It always important to have a holistic perspective regarding the minimum, maximum, middle, outliers of our data in one picture. One of the charts that help us to have a perspective regarding these values in "Box Plot" in R. I have already a post on this concept and how to have it see: http://radacad.com/visualizing-numeric-variables-in-power-bi-boxplots-part-1

here I have shown how to make them more interactive via Plotly , having more plots in one chart regarding a factor variable.



The code for generating the Barchart has been shown below

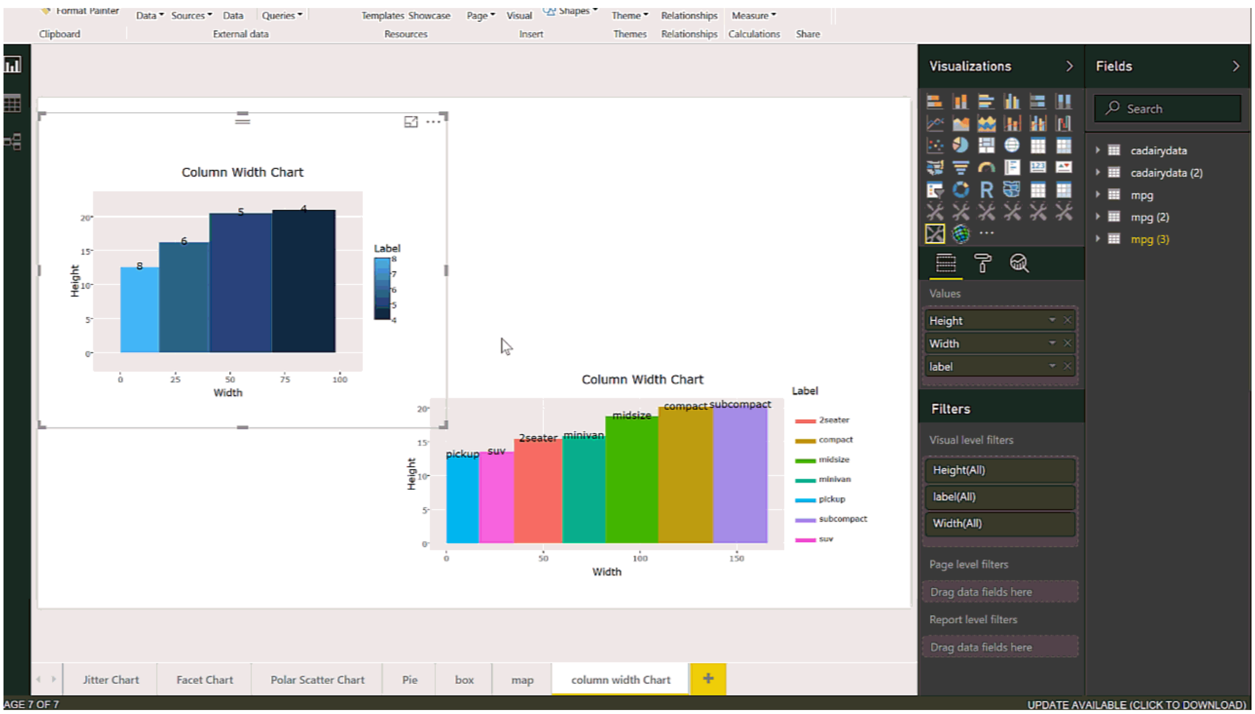
```
source('./r_files/flatten_HTML.r')

############### Library Declarations ###############
libraryRequireInstall("ggplot2");
libraryRequireInstall("plotly")
library("plotly")
library("ggplot2")
library("htmlwidgets")
####################################################


g <- plot_ly(Values, x = ~x, color = ~w, type = "box")



p = ggplotly(g);
internalSaveWidget(p, 'out.html');
```

5- Column Width Chart

For comparison purpose, most of the charts are available in Power BI Visualization, just two of them are not: Variable Width Column Chart and table with table embedded chart (I will show it next post hopefully). This chart helps us to do compare two variables in a bar chart. We have height and width of the bar chart. In a normal bar chart, the bar width is the same size as the process of creating this chart has been explained in the post: http://radacad.com/variable-width-column-chart-writing-r-codes-inside-power-bi-part-4

Here I am showing how I create a custom visual. The dataset is about car speed in city and highway plus the number of cylinders and their type of drive (four-wheel drive, rear drive or front drive).

To do that first in power BI I create two separate datasets that have been group by:1- the number of cylinders and 2- the type of drive.



Input load. Please do not change

#`dataset` = read.csv('C:/Users/leila/AppData/Local/Radio/REditorWrapper_0ec2c1f1-83e7-46bd-95fa-49bcf787902d/input_df_c8eb4c44-0bdd-4d95-98bb-9694dae86d4c.csv', check.names = FALSE, encoding = "UTF-8", blank.lines.skip = FALSE);

Original Script. Please update your script content here and once completed copy below section back to the original editing window

source('./r_files/flatten_HTML.r')

############### Library Declarations ###############

```
libraryRequireInstall("ggplot2");
libraryRequireInstall("plotly")
library("plotly")
library("ggplot2")
library("htmlwidgets")
library("dplyr")
library("ggplot2")
df<-data.frame(Label=Values$label,Height= Values$Height,width=Values$Width)
df$w <- cumsum(df$width) #cumulative sums.
df$wm <- df$w - df$width
df$GreenGas<- with(df, wm + (w - wm)/2)

p  <- ggplot(df, aes(ymin = 0))
p1 <- p + geom_rect(aes(xmin = wm, xmax = w, ymax = Height, fill = Label))
p2<-p1 + geom_text(aes(x = GreenGas, y = Height, label = Label),size=4,angle = 45)
p3<-p2+labs(title = "Column Width Chart", x = "Width", y = "Height")
#blue.bold.italic.10.text <- element_text(face = "bold.italic", color = "dark green", size = 16)
#p4<-p3+theme(axis.title = blue.bold.italic.10.text, title =blue.bold.italic.10.text)
g=p3;
p = ggplotly(g);
internalSaveWidget(p, 'out.html');
```