Data Analytics

Course Taught at IIFT

Session 7: Basics of R and RStudio

Dr. Tanujit Chakraborty

www.ctanujit.org

History of R…

- R is an open source programming language and software environment for statistical computing and graphics.
- The R language is widely used among statisticians and data scientists for developing statistical and data analytics tools.
- Modelled after S & S-plus, developed at AT&T labs in late 1980s.
- R project was started by Robert Gentleman and Ross Ihaka Department of Statistics, University of Auckland (1995).
- Currently maintained by R core development team – an international team of volunteer developers (since 1997).

Download R & RStudio

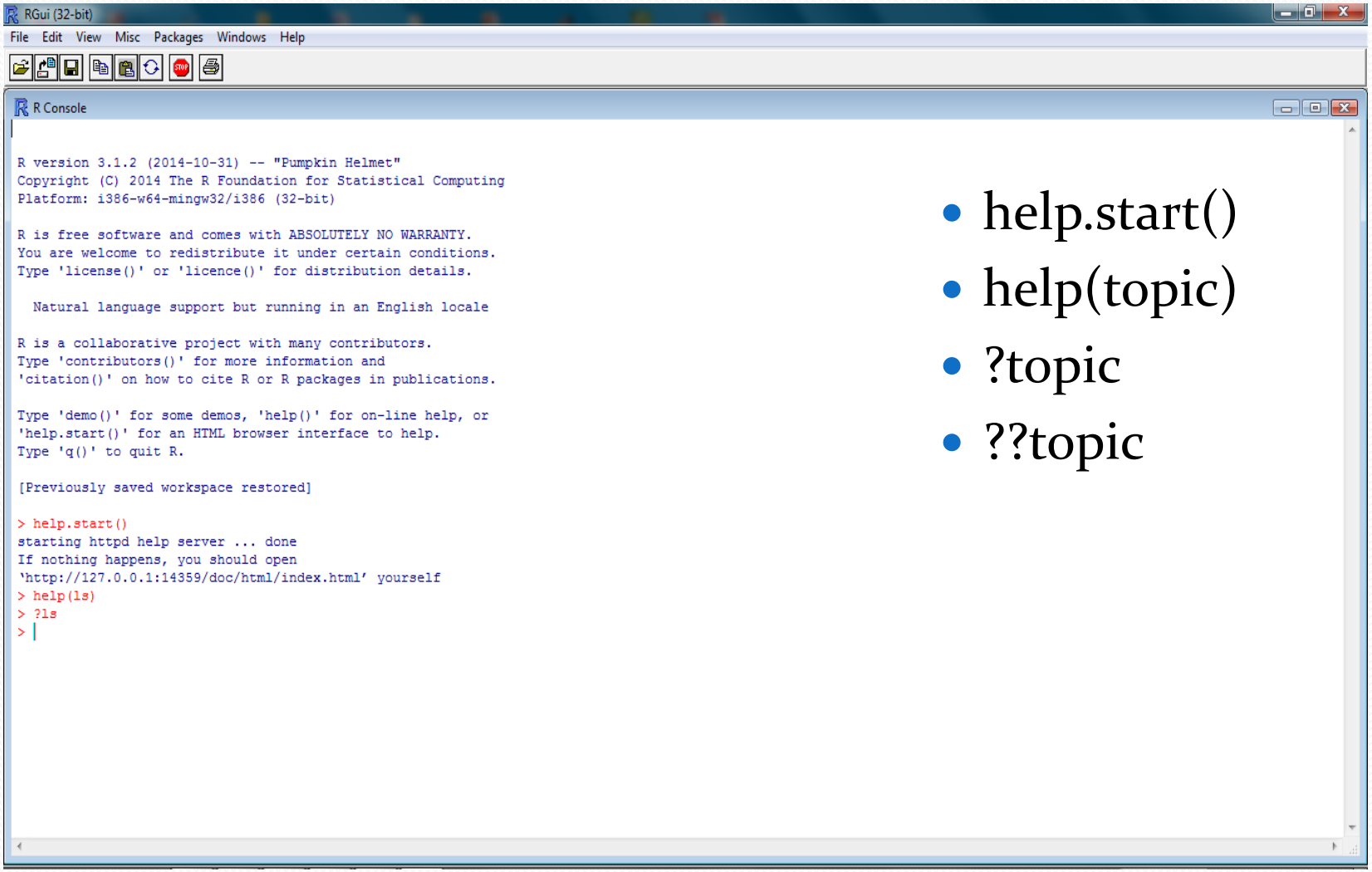
- http://www.r-project.org/
- http://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf
- Download R: http://cran.r-project.org/bin/
- Download RStudio: http://www.rstudio.com/ide/download/desktop

INSTALLATION

1.Download R software from http://cran.r-project.org/bin/windows/base/
2.Run the R set up (exe) file and follow instructions
3.Double click on the R icon in the desktop and R window will open
4.Download RStudio from http://www.rstudio.com/
5.Run R studio set up file and follow instructions
6.Click on R studio icon, R Studio IDE Studio will load
7.Tools – Global Options – Appearances – Change Colour Size Theme
(if you wish to change the background, not a mandatory step)
4.Go to R-Script (Ctrl + Shift + N)
5. Write 'Hello World !'
6.Save & Run (Ctrl + Enter)

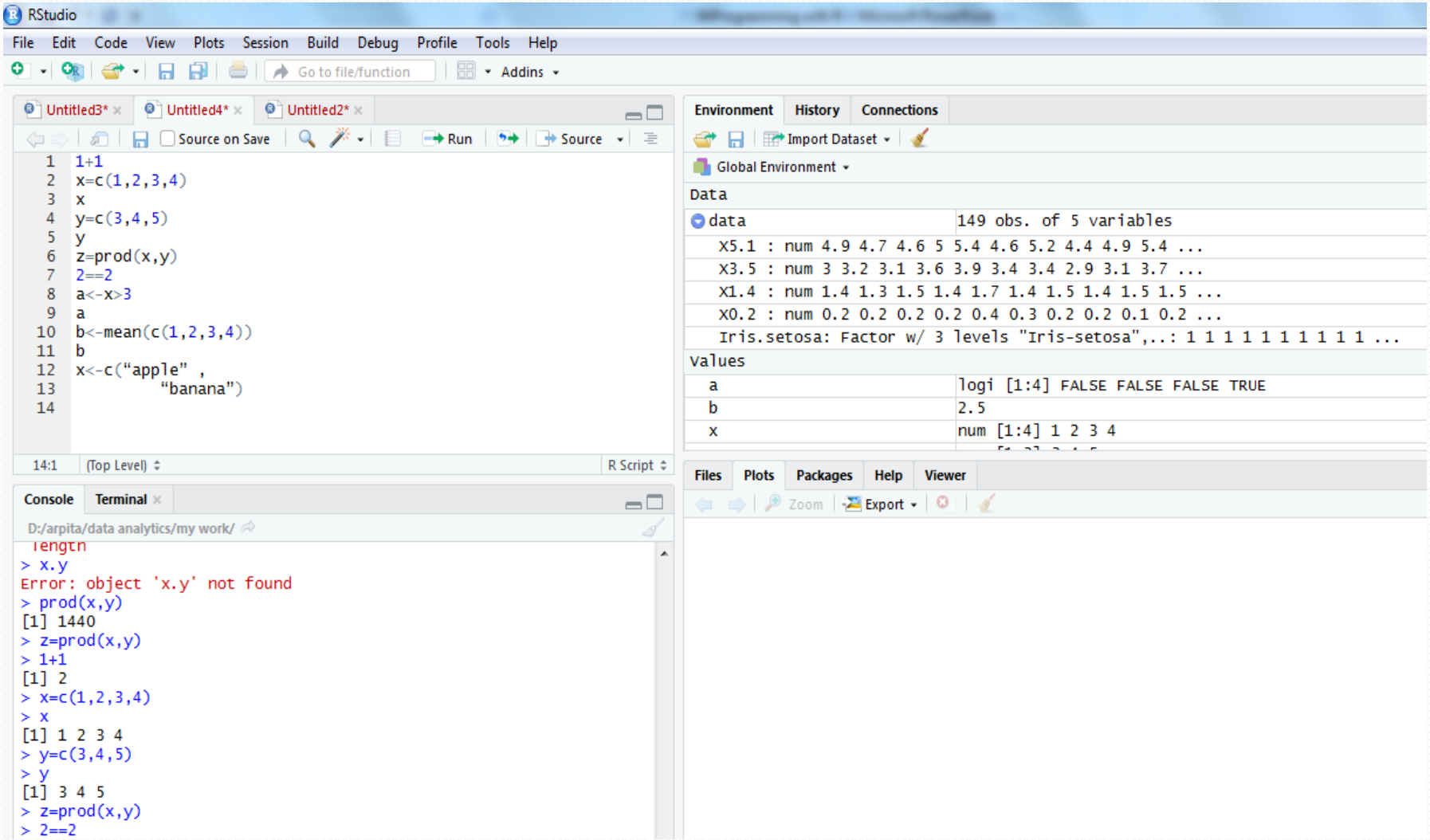
Congrats ! You have written your very first R-Program

Getting help from R console



- help.start()
- help(topic)
- ?topic
- ??topic

R command in integrated environment



How to use R for simple Mathematics

- > 3+5
- > 12 + 3 / 4 – 5 + 3*8
- > (12 + 3 / 4 – 5) + 3*8
- > pi * 2^3 – sqrt(4)
- >factorial(4)
- >log(2,10)
- >log(2, base=10)
- >log10(2)
- >log(2)

Note

- **R ignores spaces**

How to store results of calculations for future use

- > x = 3+5
- > x
- > y = 12 + 3 / 4 – 5 + 3*8
- > y
- > z = (12 + 3 / 4 – 5) + 3*8
- > z
- > A <- 6 + 8 ## no space should be between < & -
- > a ## Note: R is case sensitive
- >A

Using C command

- > data1 = c(3, 6, 9, 12, 78, 34, 5, 7, 7) ## numerical data
- > data1.text = c('Mon', 'Tue', "Wed") ## Text data
- ## Single or double quote both ok
- ##copy/paste into R console may not work
- > data1.text = c(data1.text, 'Thu', 'Fri')

Scan command for making data

- **> d3 = scan(what = 'character')**
- 1: mon
- 2: tue
- 3: wed thu
- 5:
- **> d3**
- [1] "mon" "tue" "wed" "thu"
- **> d3[2]**
- [1] "tue"
- **> d3[2]='mon'**
- **> d3**
- [1] "mon" "mon" "wed" "thu"
- > d3[6]='sat'
- > d3
- [1] "mon" "mon" "wed" "thu" NA "sat"
- > d3[2]='tue'
- > d3[5] = 'fri'
- > d3
- [1] "mon" "tue" "wed" "thu" "fri" "sat"

Concept of working directory

- >getwd()
- [1] "C:\Users\DA\R\Database"
- > setwd('D:\Data Analytics\Project\Database)
- > dir() ## working directory listing
- >ls() ## Workspace listing of objects
- >rm('object') ## Remove an element "object", if exist
- > rm(list = ls()) ## Cleaning

Reading data from a data file

- > setwd("D:/Tanujit/data analytics/my work") #Set the working directory to file location
- > getwd()
- [1] "D:/Tanujit/data analytics/my work“
- > dir()
- rm(list=ls(all=TRUE)) # Refresh session
- > data=read.csv('iris.csv', header = T, sep=",")
- (data = read.table(‘iris.csv', header = T, sep = ','))
- > ls()
- [1] "data"
- > str(data)
- 'data.frame': 149 obs. of 5 variables:
- $ X5.1 : num 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 5.4 ...
- $ X3.5 : num 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 ...
- $ X1.4 : num 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 ...
- $ X0.2 : num 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 0.2 ...
- $ Iris.setosa: Factor w/ 3 levels "Iris-setosa",..: 1 1 1 1 1 1 1 1 1 1 ...

Accessing elements from a file

- > data$X5.1
- [1] 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7
- > data$X5.1[7]=5.2
- > data$X5.1
- [1] 4.9 4.7 4.6 5.0 5.4 4.6 5.2 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7

#Note: This change has happened in workspace only not in the file.

- How to make it permanent?
- write.csv / write.table
- >write.table(data, file ='iris_mod.csv', row.names = FALSE, sep = ',')
- If row.names is TRUE, R adds one ID column in the beginning of file.
- So its suggested to use row.names = FALSE option
- >write.csv(data, file =='iris_mod.csv', row.names = TRUE) ## to test

Different data items in R

- **Vector**
- **Matrix**
- **Data Frame**
- **List**

Vectors in R

- >x=c(1,2,3,4,56)
- >x
- > x[2]
- > x = c(3, 4, NA, 5)
- >mean(x)
- [1] NA
- >mean(x, rm.NA=T)
- [1] 4
- > x = c(3, 4, NULL, 5)
- >mean(x)
- [1] 4

More on Vectors in R

- **>y = c(x,c(-1,5),x)**
- **>length(x)**
- **>length(y)**

There are useful methods to create long vectors whose elements are in arithmetic progression:

- **> x=1:20**
- **> x**

If the common difference is not 1 or -1 then we can use the seq function

- **> y=seq(2,5,0.3)**
- **> y**
- [1] 2.0 2.3 2.6 2.9 3.2 3.5 3.8 4.1 4.4 4.7 5.0
- **> length(y)**
- [1] 11

More on Vectors in R

- > x=1:5
- > mean(x)
- [1] 3
- > x
- [1] 1 2 3 4 5
- > x^2
- [1] 1 4 9 16 25
- > x+1
- [1] 2 3 4 5 6
- > 2*x
- [1] 2 4 6 8 10
- > exp(sqrt(x))
- [1] 2.718282 4.113250 5.652234 7.389056 9.356469

- It is very easy to add/subtract/multiply/divide two vectors entry by entry.
- > y=c(0,3,4,0)
- > x+y
- [1] 1 5 7 4 5
- > y=c(0,3,4,0,9)
- > x+y
- [1] 1 5 7 4 14
- Warning message:
- In x + y : longer object length is not a multiple of shorter object length
- > x=1:6
- > y=c(9,8)
- > x+y
- [1] 10 10 12 12 14 14

Matrices in R

- Same data type/mode – number , character, logical
- a.matrix <- matrix(vector, nrow = r, ncol = c, byrow = FALSE, dimnames = list(char-vector-rownames, char-vector-col-names))

dimnames is optional argument, provides labels for rows & columns.

- > y <- matrix(1:20, nrow = 4, ncol = 5)
- >A = matrix(c(1,2,3,4),nrow=2,byrow=T)
- >A
- >A = matrix(c(1,2,3,4),ncol=2)
- >B = matrix(2:7,nrow=2)
- >C = matrix(5:2,ncol=2)
- >mr <- matrix(1:20, nrow = 5, ncol = 4, byrow = T)
- >mc <- matrix(1:20, nrow = 5, ncol = 4)
- >mr
- >mc

More on matrices in R

- >dim(B) #Dimension
- >nrow(B)
- >ncol(B)
- >A+C
- >A-C
- >A%*%C #Matrix multiplication. Where will be the result?
- >A*C #Entry-wise multiplication
- >t(A) #Transpose
- >A[1,2]
- >A[1,]
- >B[1,c(2,3)]
- >B[,-1]

Lists in R

- Vectors and matrices in R are two ways to work with a collection of objects.
- Lists provide a third method. Unlike a vector or a matrix a list can hold different kinds of objects.
- One entry in a list may be a number, while the next is a matrix, while a third is a character string (like "Hello R!").
- Statistical functions of R usually return the result in the form of lists. So we must know how to unpack a list using the $ symbol.

Examples of lists in R

- >x = list(name="Tanujit", nationality="Indian", height=5.5, marks=c(95,45,80))

- >names(x)
- >x$name
- >x$hei #abbreviations are OK
- >x$marks
- >x$m[2]

Data frame in R

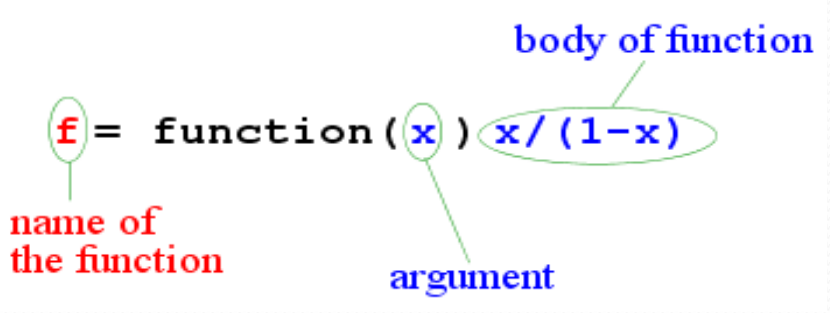
A data frame is more general than a matrix, in that different columns can have different modes (numeric, character, factor, etc.).

- >d <- c(1,2,3,4)
- >e <- c("red", "white", "red", NA)
- >f <- c(TRUE,TRUE,TRUE,FALSE)
- >myframe <- data.frame(d,e,f)
- >names(myframe) <- c("ID","Color","Passed") # Variable names
- >myframe
- >myframe[1:3,] # Rows 1 , 2, 3 of data frame
- >myframe[,1:2] # Col 1, 2 of data frame
- >myframe[c("ID","Color")] #Columns ID and color from data frame
- >myframe$ID # Variable ID in the data frame

Factors in R

- In R we can make a variable is nominal by making it a factor.
- The factor stores the nominal values as a vector of integers in the range [1… k] (where k is the number of unique values in the nominal variable).
- An internal vector of character strings (the original values) mapped to these integers.
- # Example: variable gender with 20 "male" entries and
  # 30 "female" entries
  >gender <- c(rep("male",20), rep("female", 30))
  >gender <- factor(gender)
  # Stores gender as 20 1's and 30 2's
- # 1=male, 2=female internally (alphabetically)
  # R now treats gender as a nominal variable
  >summary(gender)

Functions in R



- >g = function(x,y) (x+2*y)/3
- >g(1,2)
- >g(2,1)

BASIC TASKS

Matrix multiplication – Code

Read the matrix A and B

```
A = matrix(c(21,57,89,31,7,98), nrow =2, ncol=3, byrow = TRUE)
B = matrix(c(24, 35, 15, 34, 56,25), nrow = 3, ncol = 2, byrow = TRUE)
```

Multiplication of matrices

```
C = A%*%B
C
```

Determinant – R Code

```
A = matrix(c(51, 10, 23, 64), nrow = 2, ncol =2, byrow =TRUE)
det(A)
```

Matrix Inverse – R code

```
A = matrix(c(51, 10, 23, 64), nrow = 2, ncol =2, byrow =TRUE)
solve(A)
```

BASIC TASKS

Eigen values and Eigen vectors – R Code

```
A = matrix(c(1, -2, 3, -4), nrow = 2, ncol = 2, byrow = TRUE)
eigen(A)
```

Generating 5 Random Numbers – R Code

```
x = rnorm(5, mean = 0, sd = 1)
x
```

Functional Help

```
?rnorm()
```

Package Installation

```
install.packages("ggplot")
```

Library Call (for use)

```
library(ggplot)
```

DESCRIPTIVE STATISTICS

Exercise 1: The monthly credit card expenses of an individual in 1000 rupees is given in the file Credit_Card_Expenses.csv.

a.Read the dataset to R studio

b.Compute mean, median minimum, maximum, range, variance, standard deviation, skewness, kurtosis and quantiles of Credit Card Expenses

c.Compute default summary of Credit Card Expenses

d.Draw Histogram of Credit Card Expenses

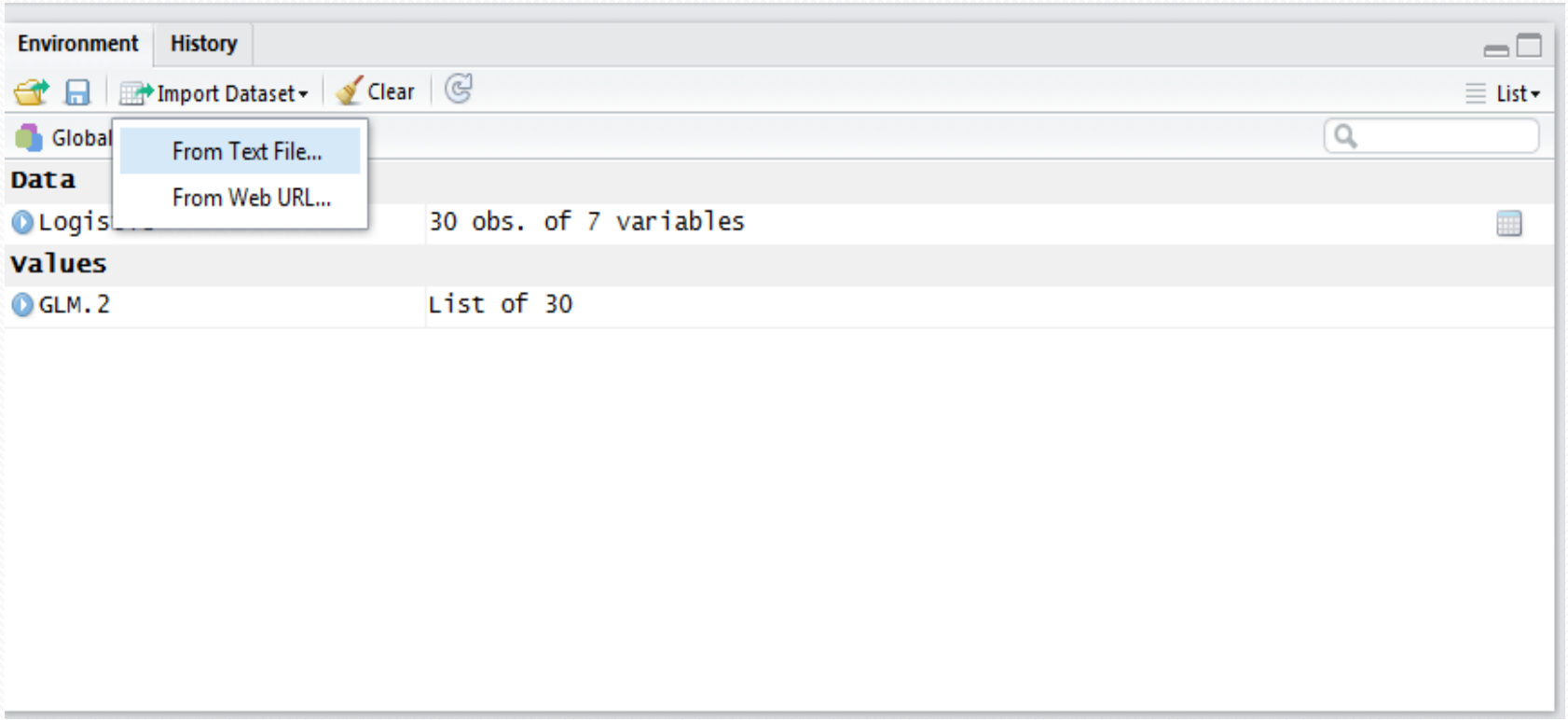
DESCRIPTIVE STATISTICS

The monthly credit card expenses of an individual in 1000 rupees is given below. Kindly summarize the data

Month	Credit Card Expenses	Month	Credit Card Expenses
1	55	11	63
2	65	12	55
3	59	13	61
4	59	14	61
5	57	15	57
6	61	16	59
7	53	17	61
8	63	18	57
9	59	19	59
10	57	20	63

DESCRIPTIVE STATISTICS

Reading a csv file to R Studio

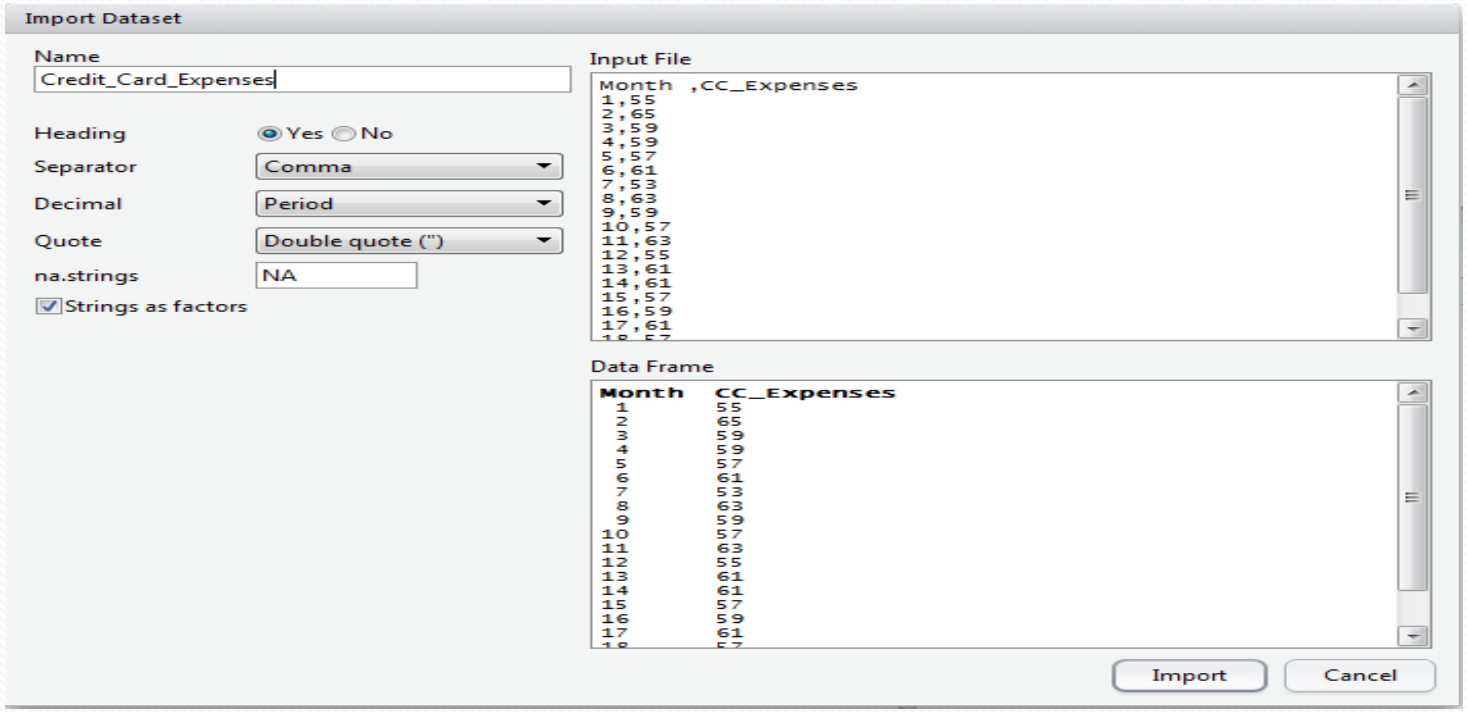


The file open dialog box will pop up

Browse to the file

DESCRIPTIVE STATISTICS

Reading a csv file to R Studio



Click Import button

R studio will read the data set to a data frame with specified name

DESCRIPTIVE STATISTICS

Reading a csv file to R Studio : Source code

- Credit_Card_Expenses <- read.csv("C:/Desktop/Data/Credit_Card_Expenses.csv")

To change the name of the data set to : mydata

```
> mydata = Credit_Card_Expenses
```

To display the contents of the data set

```
> print(mydata)
```

To read a particular column or variable of data set to a ne variable Example: Read CC_Expenses to CC

```
>CC = mydata$CC_Expenses
```

DESCRIPTIVE STATISTICS

Reading data from MS Excel formats to R Studio

Format	Code
Excel	library(xlsx) mydata <- read.xlsx("c:/myexcel.xlsx", "Sheet1")

Reading data from databases to R Studio

Function	Description
odbcConnect(*dsn*, uid="", pwd="")	Open a connection to an ODBC database
sqlFetch(*channel*, *sqtable*)	Read a table from an ODBC database into a data frame
sqlQuery(*channel*, *query*)	Submit a query to an ODBC database and return the results
sqlSave(*channel*, *mydf*, tablename = *sqtable*, append = *FALSE*)	Write or update (append=True) a data frame to a table in the ODBC database
sqlDrop(*channel*, *sqtable*)	Remove a table from the ODBC database
close(*channel*)	Close the connection

DESCRIPTIVE STATISTICS

Operators - Arithmetic

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2

DESCRIPTIVE STATISTICS

Operators - Logical

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
! =	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

DESCRIPTIVE STATISTICS

Descriptive Statistics

Computation of descriptive statistics for variable CC

Function	Code	Value
Mean	> mean(CC)	59.2
Median	> median(CC)	59
Standard deviation	> sd(CC)	3.105174
Variance	> var(CC)	9.642105
Minimum	> min(CC)	53
Maximum	> max(CC)	65
Range	> range(CC)	53.65

DESCRIPTIVE STATISTICS

Descriptive Statistics

Function	Code
Quantile	> quantile(CC)

Output					
Quantile	0%	25%	50%	75%	100%
Value	53	57	59	61	65

Function	Code
Summary	>summary(CC)

Output					
Minimum	Q1	Median	Mean	Q3	Maximum
53	57	59	59.2	61	65

DESCRIPTIVE STATISTICS

Descriptive Statistics

Function	Code
describe	> library(psych) > describe(CC)

Output	
Statistics	Values
N	20
mean	59.2
sd	3.11
median	59
Trimmed	59.25
mad	2.97
min	53
Max	65
Range	12
Skew	-0.08
Kurtosis	-0.85
se	0.69

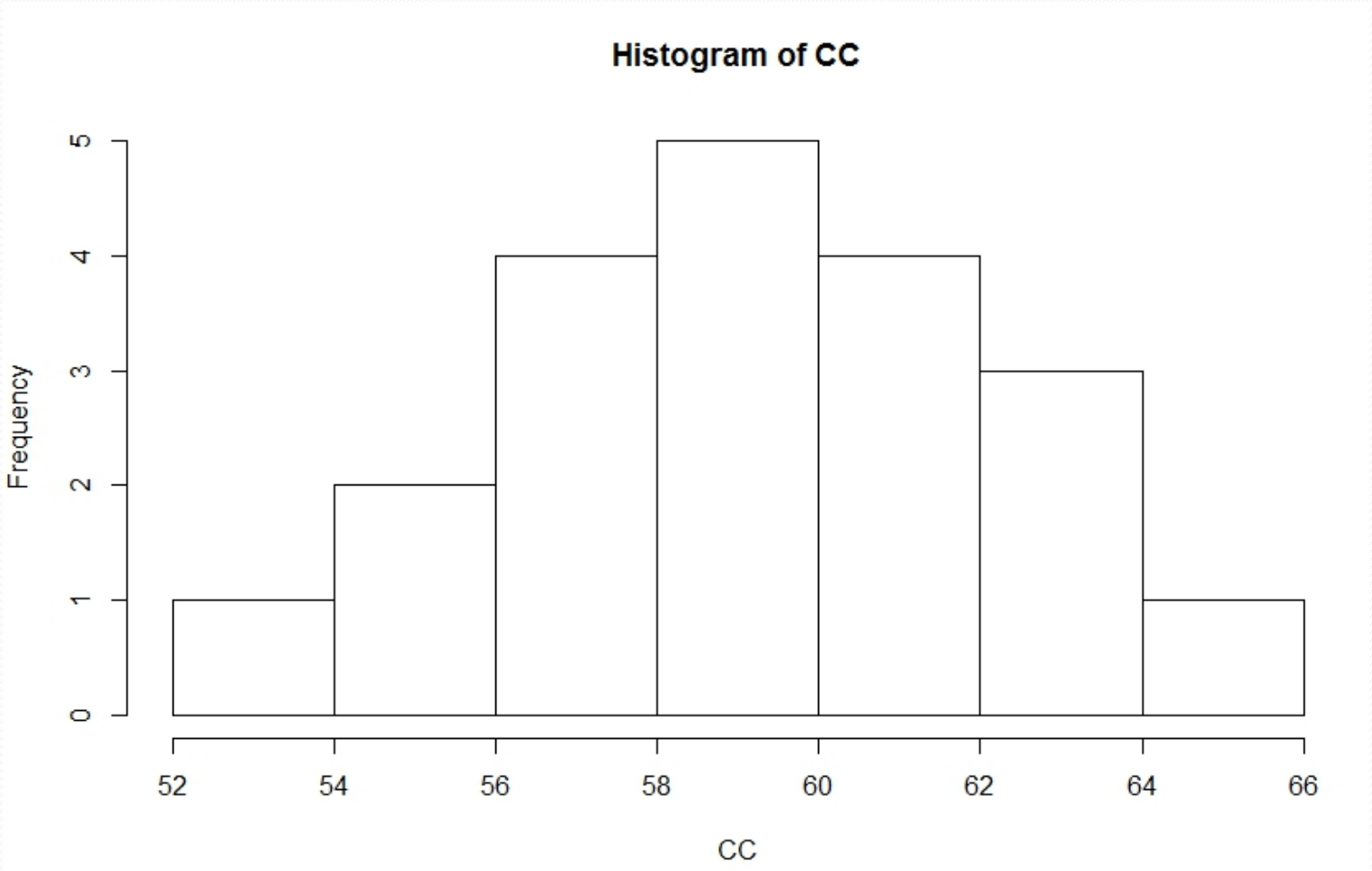
DESCRIPTIVE STATISTICS

Graphs

Graph	Code
Histogram	> hist(CC)
Histogram colour (“Blue”)	> hist(CC,col="blue")
Dot plot	> dotchart(CC)
Box plot	> boxplot(CC)
Box plot colour	> boxplot(CC, col="dark green")

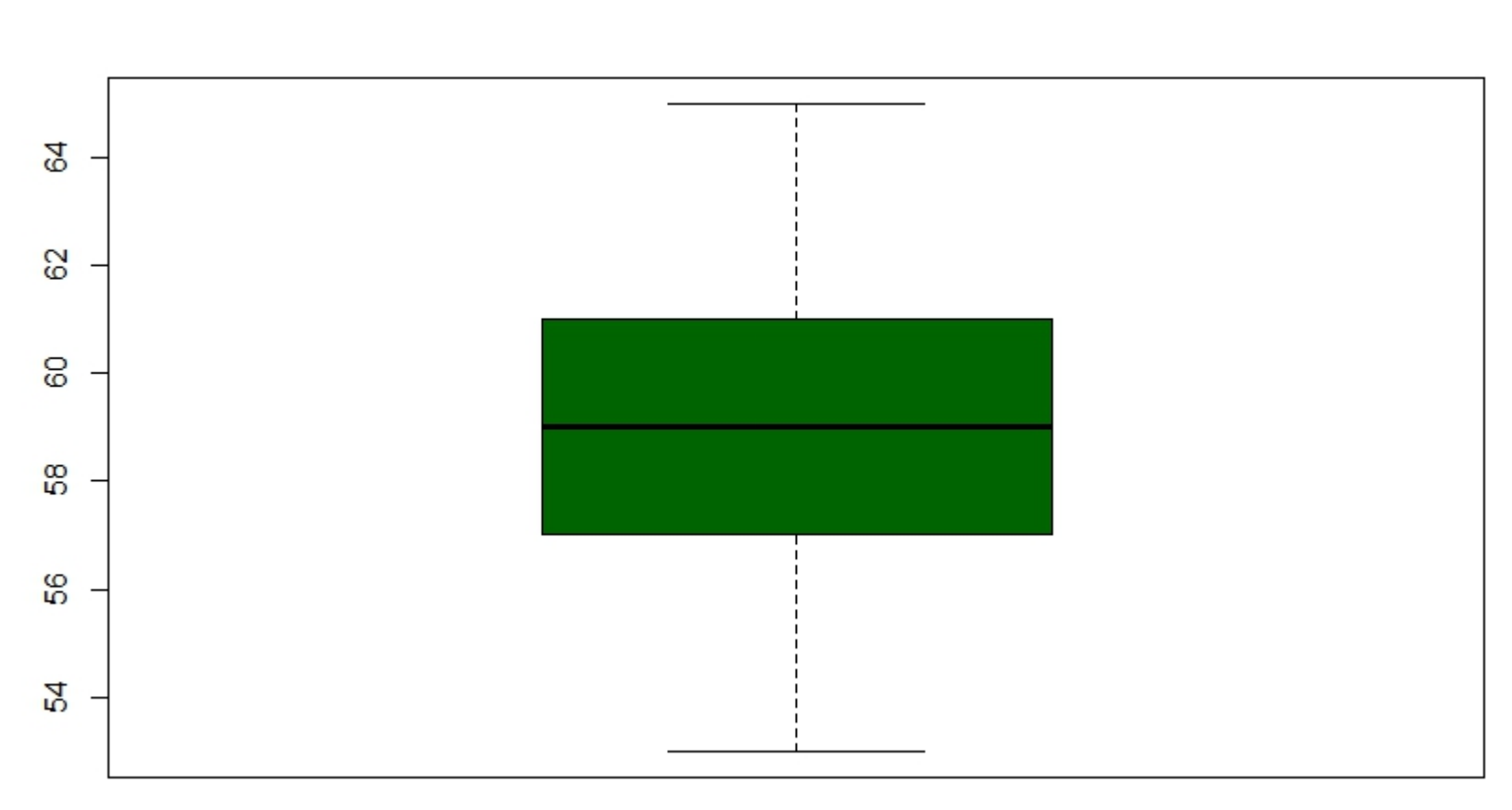
DESCRIPTIVE STATISTICS

Histogram : Variable - CC



DESCRIPTIVE STATISTICS

Box plot : Variable - CC



DATA VISUALIZATION

DATA VISUALIZATION

With ever increasing volume of data, it is impossible to tell stories without visualizations. Data visualization is an art of how to turn numbers into useful knowledge.

Popular Data Visualization Techniques:

1. Scatter Plot
2. Histogram
3. Bar & Stack Bar Chart
4. Box Plot
5. Area Chart
6. HeatMap
7. Correlogram

DATA VISUALIZATION

We'll use 'Big_Mart_Dataset.csv' example as shown below to understand how to create visualizations.

Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
FDA15	9.300	Low Fat	0.016047301	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Typ
DRC01	5.920	Regular	0.019278216	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Typ
FDN15	17.500	Low Fat	0.016760075	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Typ
FDX07	19.200	Regular	0.000000000	Fruits and Vegetables	182.0950	OUT010	1998		Tier 3	Grocery Store
NCD19	8.930	Low Fat	0.000000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Typ
FDP36	10.395	Regular	0.000000000	Baking Goods	51.4008	OUT018	2009	Medium	Tier 3	Supermarket Typ
FDO10	13.650	Regular	0.012741089	Snack Foods	57.6588	OUT013	1987	High	Tier 3	Supermarket Typ
FDP10	NA	Low Fat	0.127469857	Snack Foods	107.7622	OUT027	1985	Medium	Tier 3	Supermarket Typ
FDH17	16.200	Regular	0.016687114	Frozen Foods	96.9726	OUT045	2002		Tier 2	Supermarket Typ
FDU28	19.200	Regular	0.094449590	Frozen Foods	187.8214	OUT017	2007		Tier 2	Supermarket Typ
FDY07	11.800	Low Fat	0.000000000	Fruits and Vegetables	45.5402	OUT049	1999	Medium	Tier 1	Supermarket Typ
FDA03	18.500	Regular	0.045463773	Dairy	144.1102	OUT046	1997	Small	Tier 1	Supermarket Typ
FDX32	15.100	Regular	0.100013500	Fruits and Vegetables	145.4786	OUT049	1999	Medium	Tier 1	Supermarket Typ
FDS46	17.600	Regular	0.047257328	Snack Foods	119.6782	OUT046	1997	Small	Tier 1	Supermarket Typ
FDF32	16.350	Low Fat	0.068024300	Fruits and Vegetables	196.4426	OUT013	1987	High	Tier 3	Supermarket Typ

DATA VISUALIZATION

1. Scatter Plot: It is used to see the relationship between two continuous variables. In our above mart dataset, if we want to visualize the items as per their cost data, then we can use scatter plot chart using two continuous variables, namely Item_Visibility & Item_MRP as shown.

Read data and simple scatter plot using function ggplot() with geom_point().

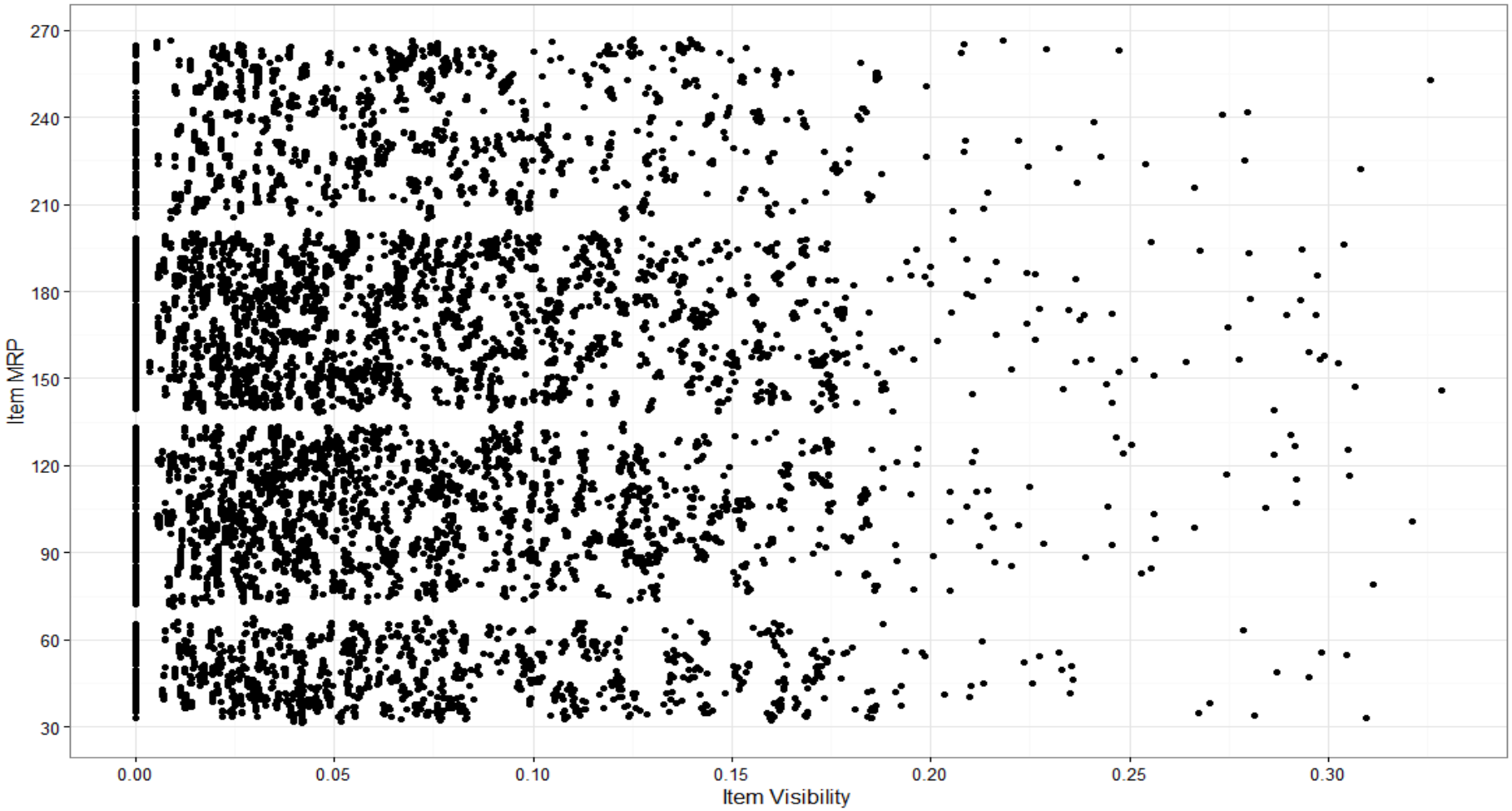
```
> train <- read.csv("C:/Users/Data/Big_Mart_Dataset.csv")

> view(train)

> library(ggplot2)

> ggplot(train, aes(Item_Visibility, Item_MRP)) + geom_point() +
scale_x_continuous("Item Visibility", breaks = seq(0,0.35,0.05))+
scale_y_continuous("Item MRP", breaks = seq(0,270,by = 30))+ theme_bw()
```

DATA VISUALIZATION



DATA VISUALIZATION

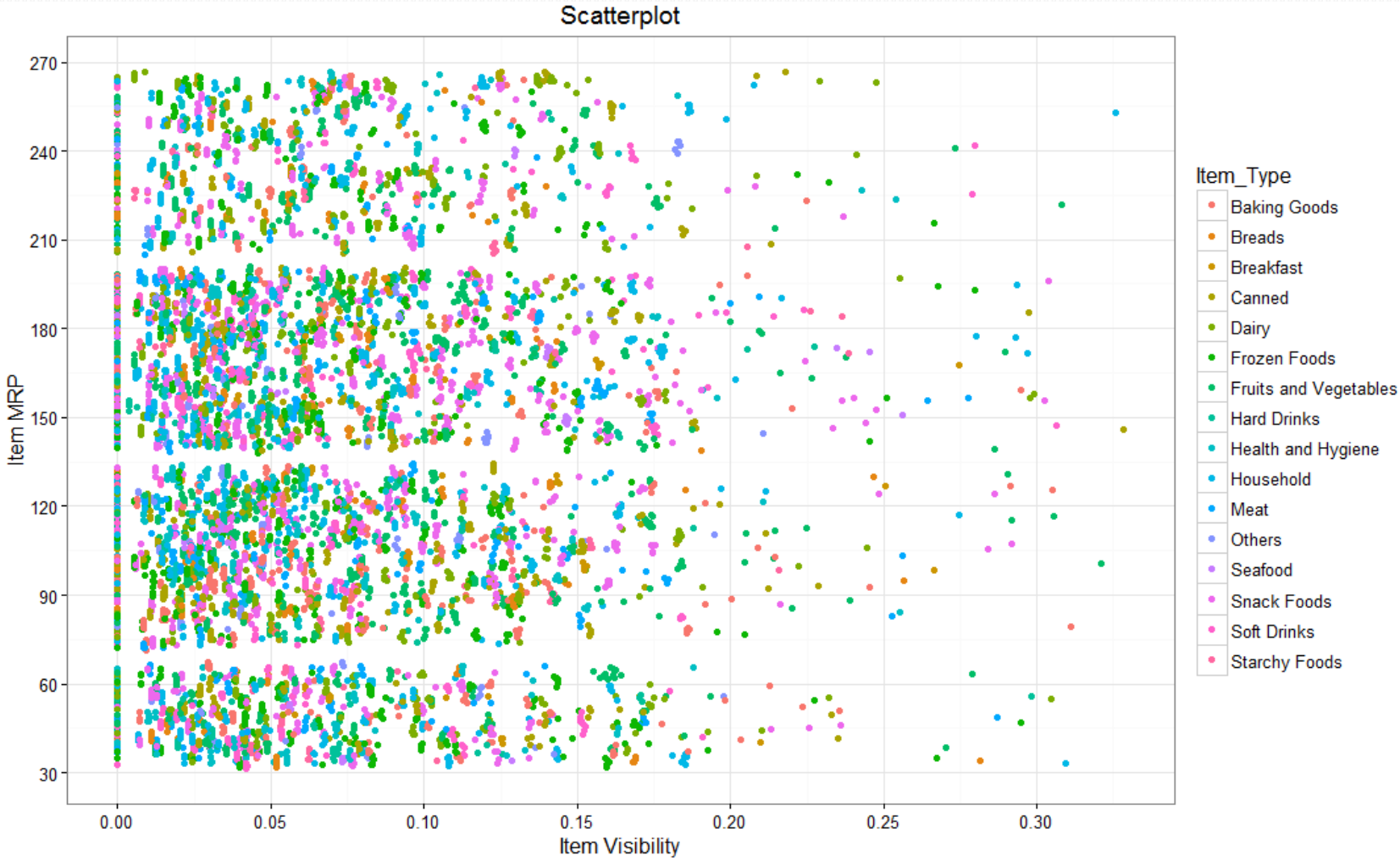
1. Scatter Plot: Now, we can view a third variable also in same chart, say a categorical variable (Item_Type) which will give the characteristic (item_type) of each data set. Different categories are depicted by way of different color for item_type in below chart.

Another scatter plot using function ggplot() with geom_point().

```
> library(ggplot2)

> ggplot(train, aes(Item_Visibility, Item_MRP)) + geom_point(aes(color = Item_Type)) + scale_x_continuous("Item Visibility", breaks = seq(0,0.35,0.05))+ scale_y_continuous("Item MRP", breaks = seq(0,270,by = 30))+ theme_bw() + labs(title="Scatterplot")
```

DATA VISUALIZATION



DATA VISUALIZATION

1. Scatter Plot: We can even make it more visually clear by creating separate scatter plots for each separate Item_Type as shown below.

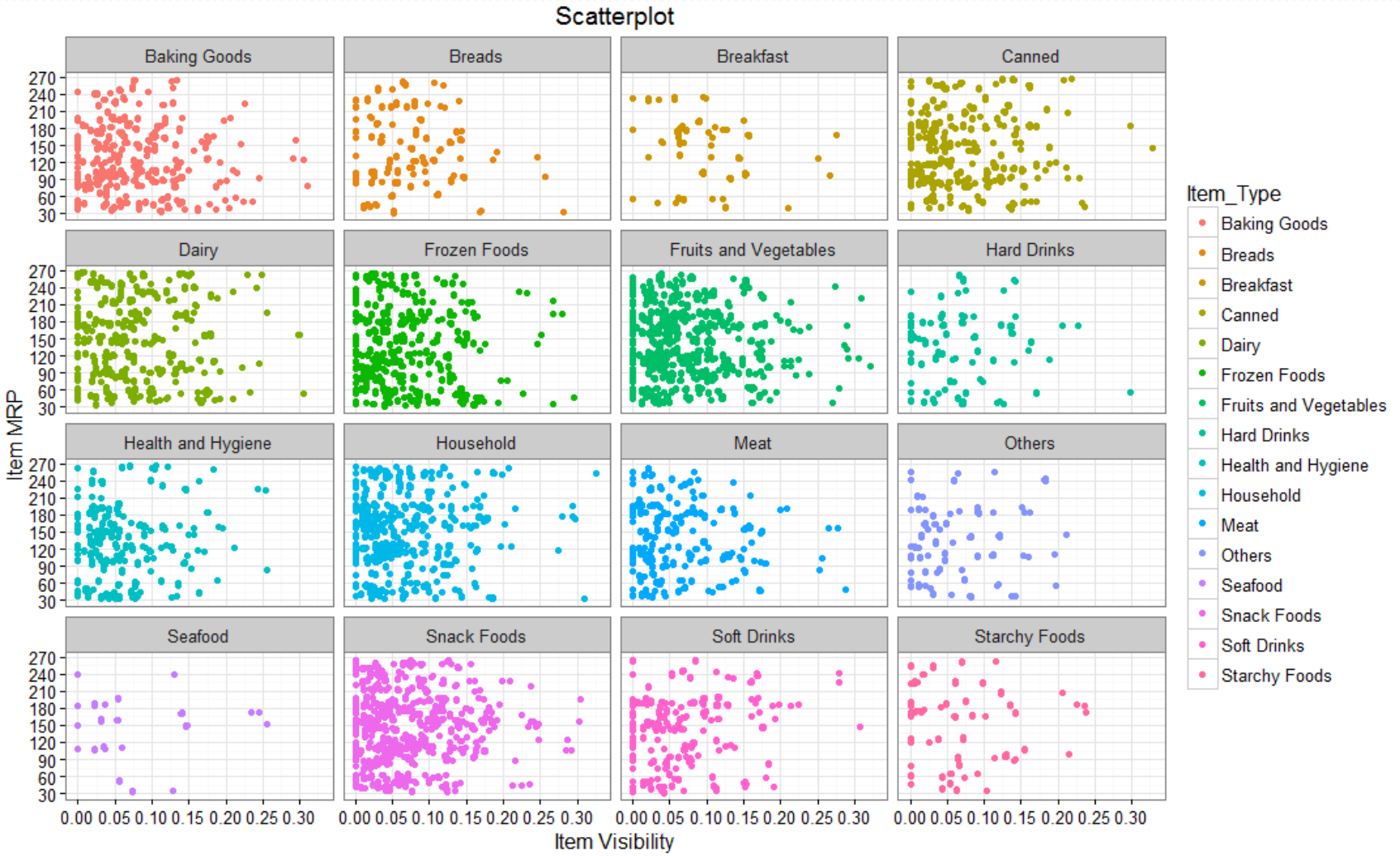
Another scatter plot using function ggplot() with geom_point().

- library(ggplot2)

- ggplot(train, aes(Item_Visibility, Item_MRP)) + geom_point(aes(color = Item_Type)) + scale_x_continuous("Item Visibility", breaks = seq(0,0.35,0.05))+ scale_y_continuous("Item MRP", breaks = seq(0,270,by = 30))+ theme_bw() + labs(title="Scatterplot") + facet_wrap(~ Item_Type)

Here, facet_wrap works well & wraps Item_Type in rectangular layout.

DATA VISUALIZATION



DATA VISUALIZATION

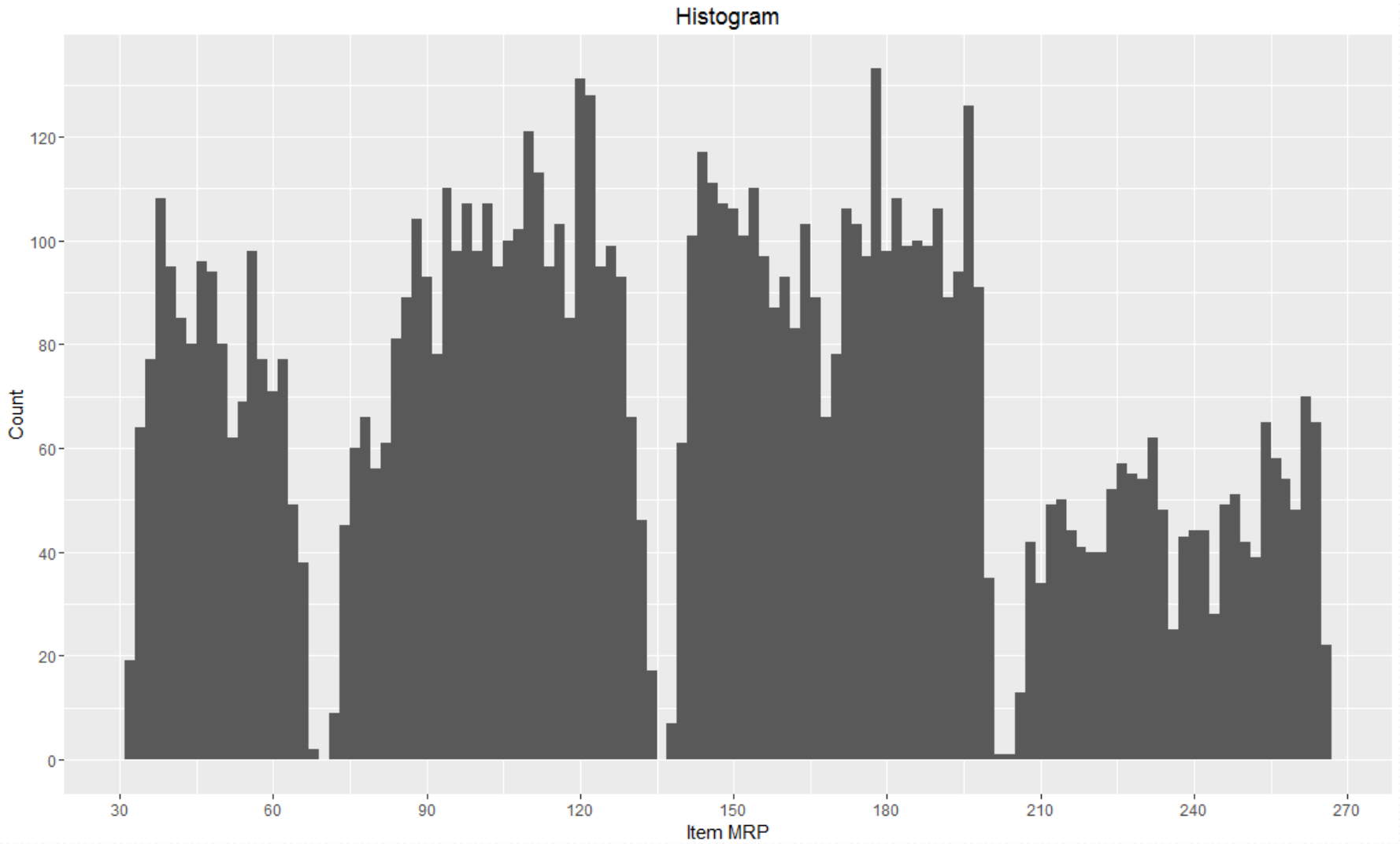
2. Histogram: It is used to plot continuous variable. It breaks the data into bins and shows frequency distribution of these bins. We can always change the bin size and see the effect it has on visualization.

For Big_Mart_Dataset, if we want to know the count of items on basis of their cost, then we can plot histogram using continuous variable Item_MRP as shown below.

Histogram plot using function ggplot() with geom_ histogram()

```
> ggplot(train, aes(Item_MRP)) + geom_histogram(binwidth = 2)+
scale_x_continuous("Item MRP", breaks = seq(0,270,by = 30))+
scale_y_continuous("Count", breaks = seq(0,200,by = 20))+  labs(title =
"Histogram")
```

DATA VISUALIZATION



DATA VISUALIZATION

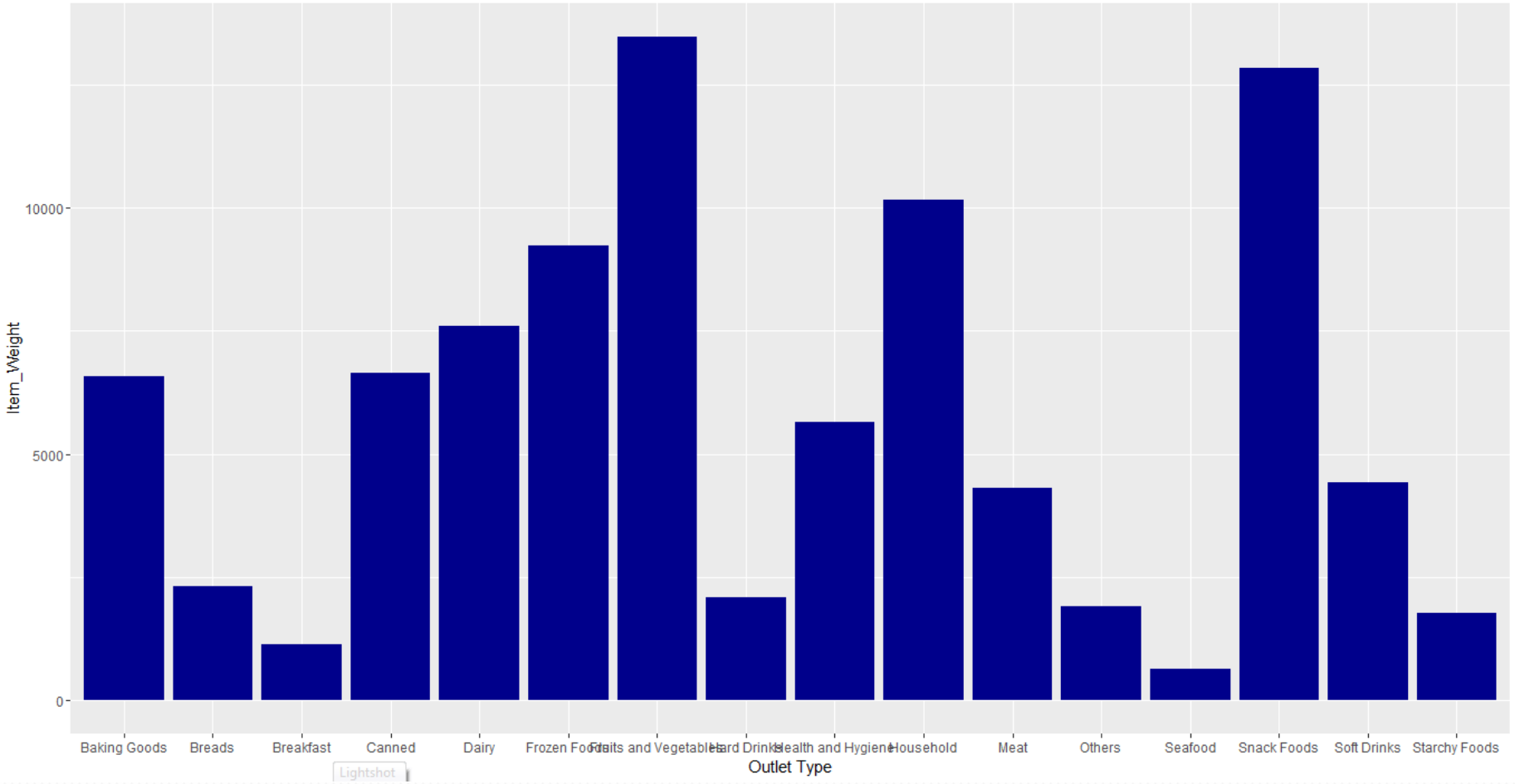
3. Bar Chart: It is used when you want to plot a categorical variable or a combination of continuous and categorical variable.

For Big_Mart_Dataset, if we want to know item weights (continuous variable) on basis of Outlet Type (categorical variable) on single bar chart as shown below.

Vertical Bar plot using function ggplot()

```
> ggplot(train, aes(Item_Type, Item_Weight)) + geom_bar(stat = "identity", fill = "darkblue") + scale_x_discrete("Outlet Type")+ scale_y_continuous("Item Weight", breaks = seq(0,15000, by = 500))+ theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) + labs(title = "Bar Chart")
```

DATA VISUALIZATION



DATA VISUALIZATION

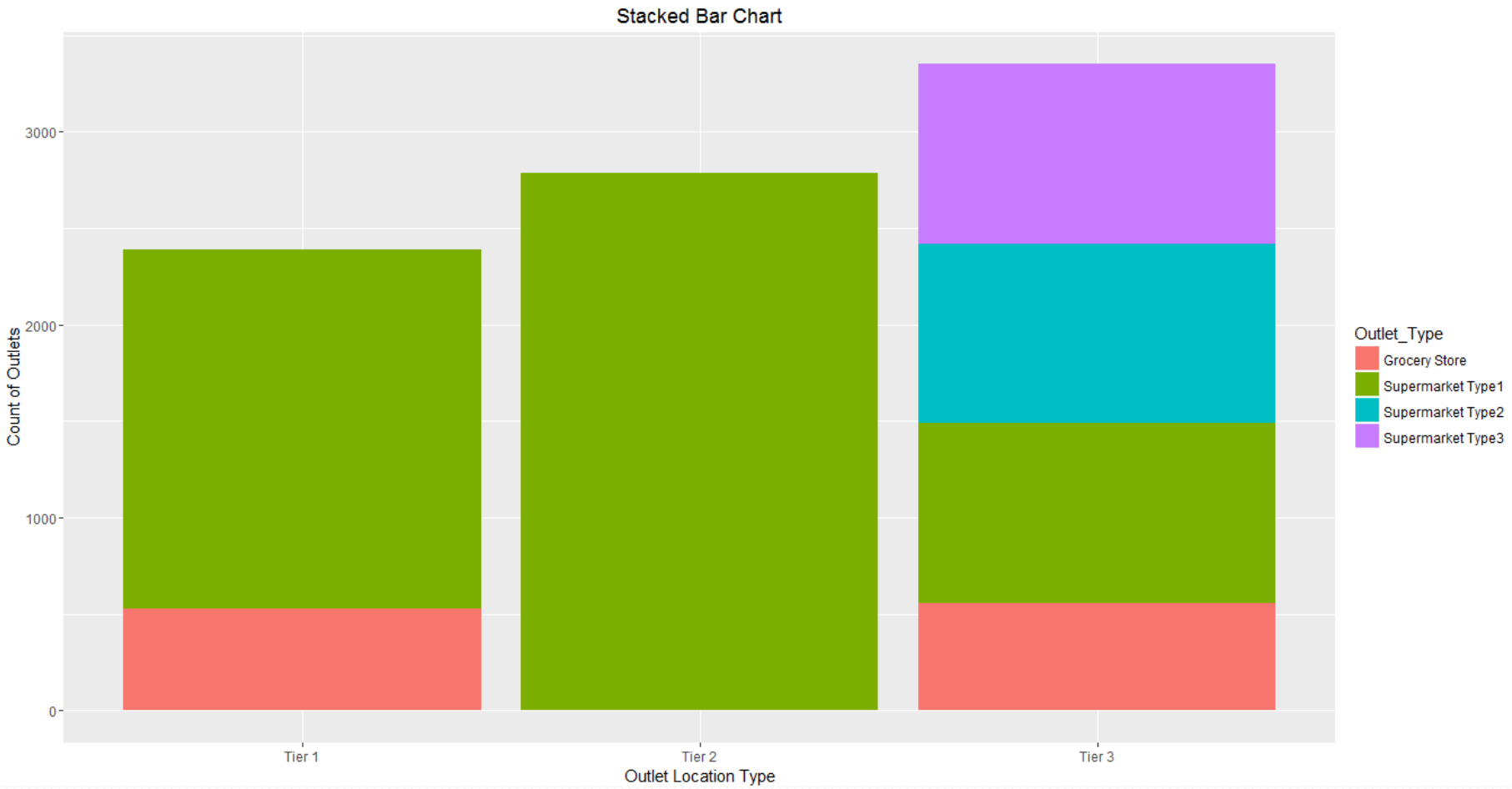
3. Stack Bar Chart: It is an advanced version of bar chart, used for visualizing a combination of categorical variables.

For Big_Mart_Dataset, if we want to know the count of outlets on basis of categorical variables like its type (Outlet Type) and location (Outlet Location Type) both, stack chart will visualize the scenario in most useful manner.

Stack Bar Chart using function ggplot()

```
> ggplot(train, aes(Outlet_Location_Type, fill = Outlet_Type)) +
geom_bar()+labs(title = "Stacked Bar Chart", x = "Outlet Location Type", y =
"Count of Outlets")
```

DATA VISUALIZATION



DATA VISUALIZATION

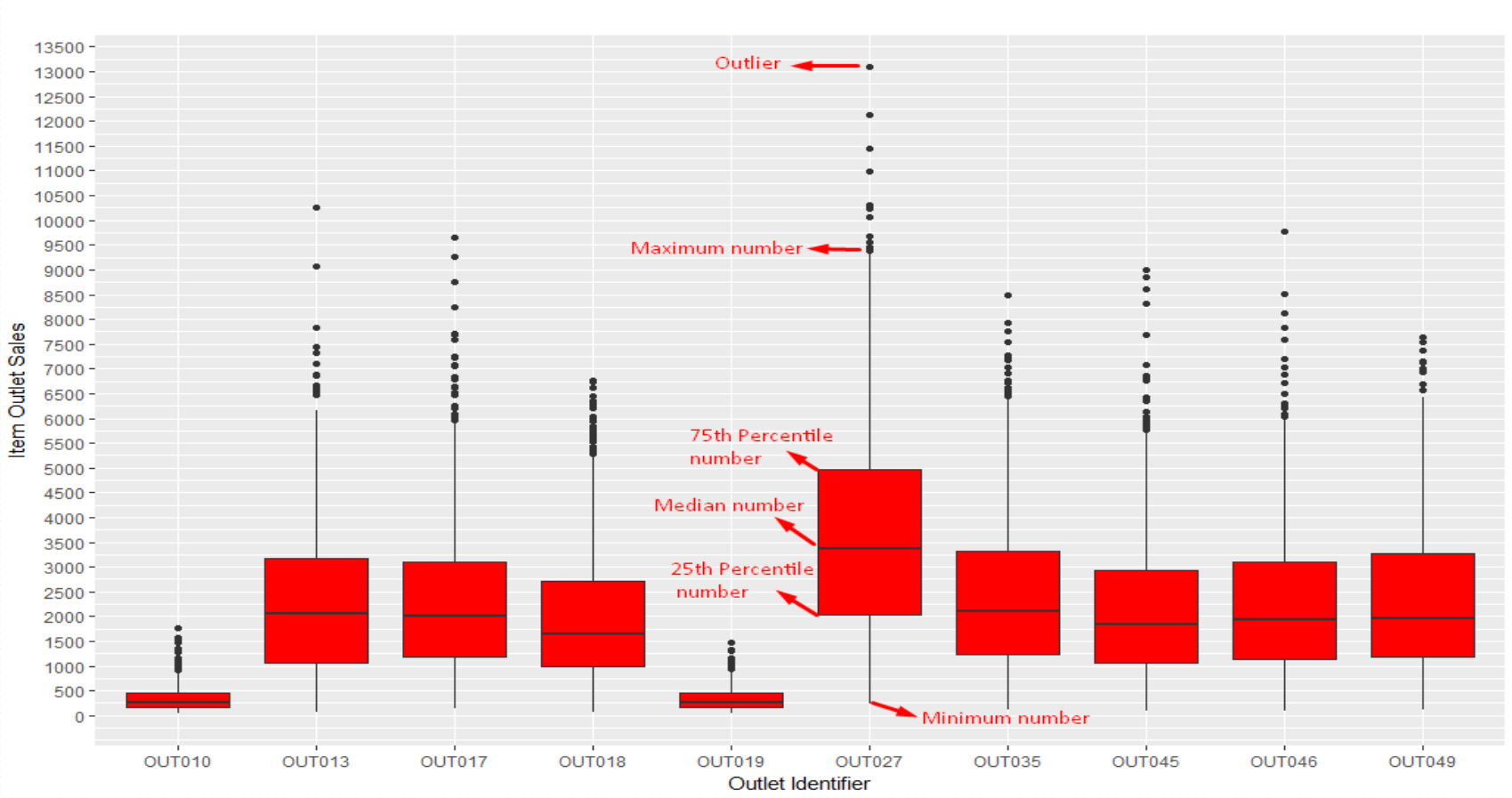
4. Box Plot: It is used to plot a combination of categorical and continuous variables. This plot is useful for visualizing the spread of the data and detect outliers. It shows five statistically significant numbers- the minimum, the 25th percentile, the median, the 75th percentile and the maximum.

For Big_Mart_Dataset, if we want to identify each outlet's detailed item sales including minimum, maximum & median numbers, box plot can be helpful. In addition, it also gives values of outliers of item sales for each outlet as shown in below chart.

R Code:

```
> ggplot(train, aes(Outlet_Identifier, Item_Outlet_Sales)) + geom_boxplot(fill = "red")+scale_y_continuous("Item Outlet Sales", breaks= seq(0,15000, by=500))+labs(title = "Box Plot", x = "Outlet Identifier")
```

DATA VISUALIZATION



The black points are outliers. Outlier detection and removal is an essential step of successful data exploration.

DATA VISUALIZATION

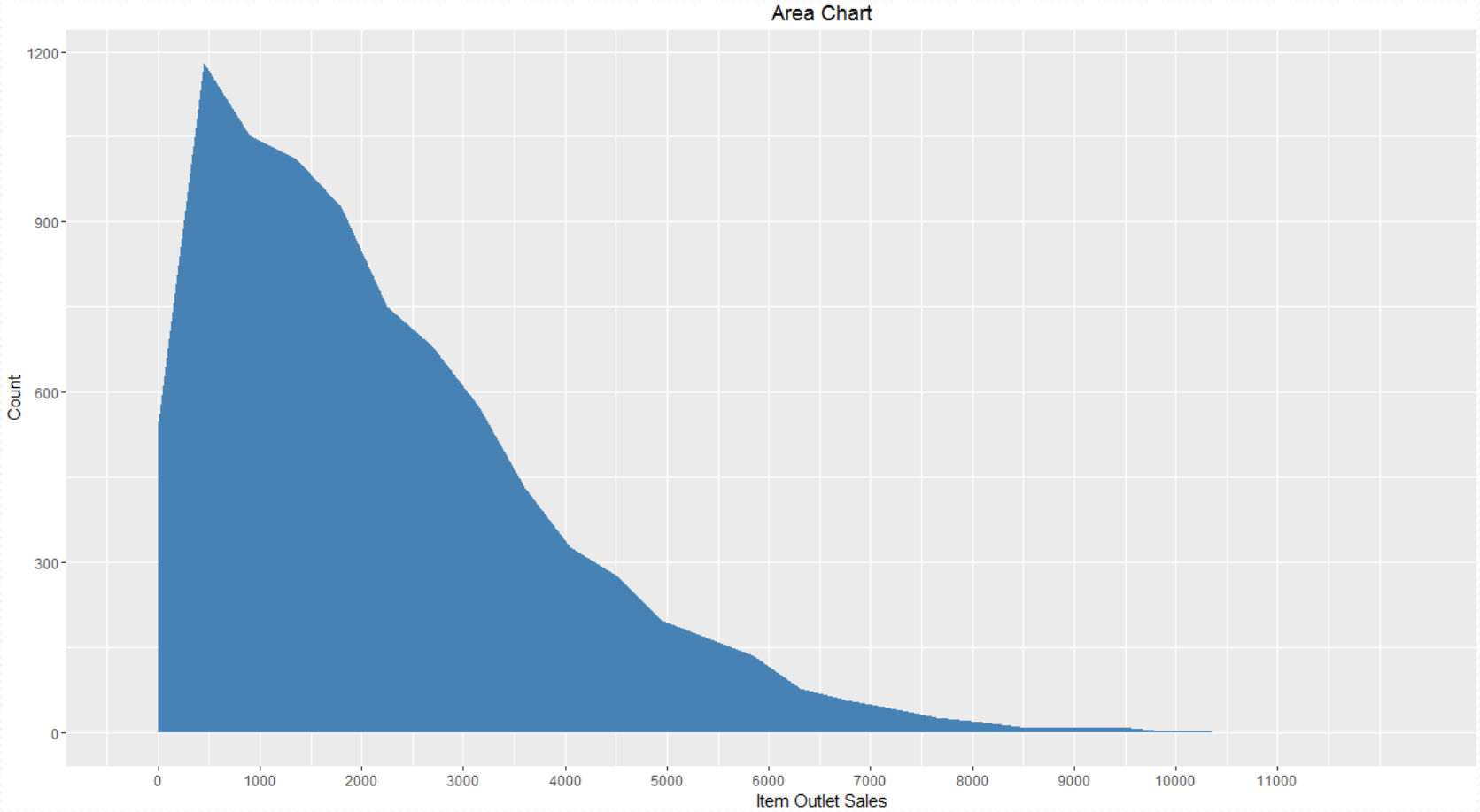
5. Area Chart: It is used to show continuity across a variable or data set. It is very much same as line chart and is commonly used for time series plots. Alternatively, it is also used to plot continuous variables and analyse the underlying trends.

For Big_Mart_Dataset, when we want to analyse the trend of item outlet sales, area chart can be plotted as shown below. It shows count of outlets on basis of sales.

R Code:

```
> ggplot(train, aes(Item_Outlet_Sales)) + geom_area(stat = "bin", bins = 30, fill = "steelblue") + scale_x_continuous(breaks = seq(0,11000,1000))+ labs(title = "Area Chart", x = "Item Outlet Sales", y = "Count")
```

DATA VISUALIZATION



Area chart shows continuity of Item Outlet Sales using function ggplot() with geom_area.

DATA VISUALIZATION

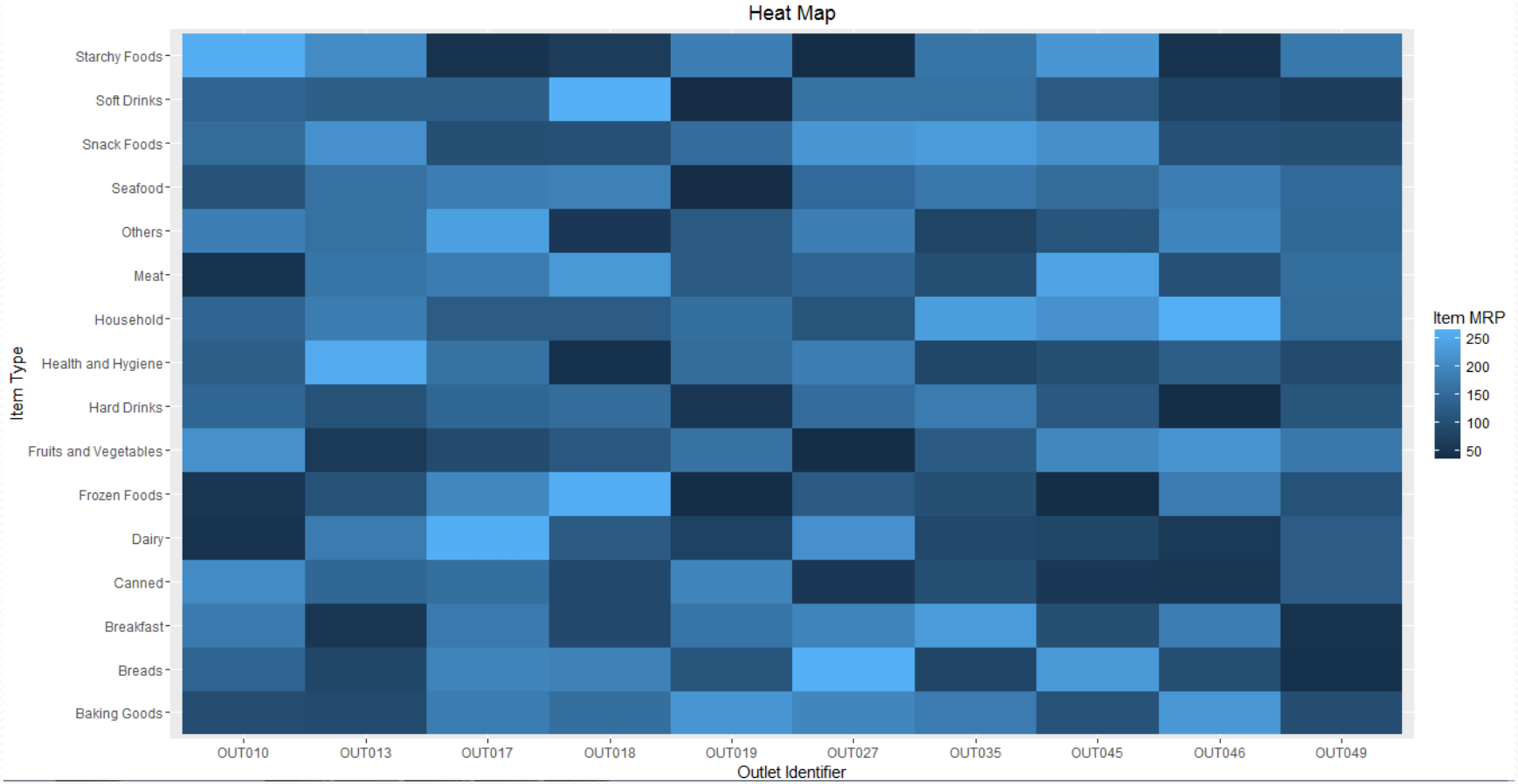
6. Heat Map: It uses intensity (density) of colours to display relationship between two or three or many variables in a two-dimensional image.

For Big_Mart_Dataset, if we want to know cost of each item on every outlet, we can plot heatmap as shown below using three variables Item MRP, Outlet Identifier & Item Type from our mart dataset.

R Code:

```
> ggplot(train, aes(Outlet_Identifier, Item_Type))+  geom_raster(aes(fill = Item_MRP))+  labs(title ="Heat Map", x = "Outlet Identifier", y = "Item Type")+ scale_fill_continuous(name = "Item MRP")
```

DATA VISUALIZATION



The dark portion indicates Item MRP is close 50.
The brighter portion indicates Item MRP is close to 250.

DATA VISUALIZATION

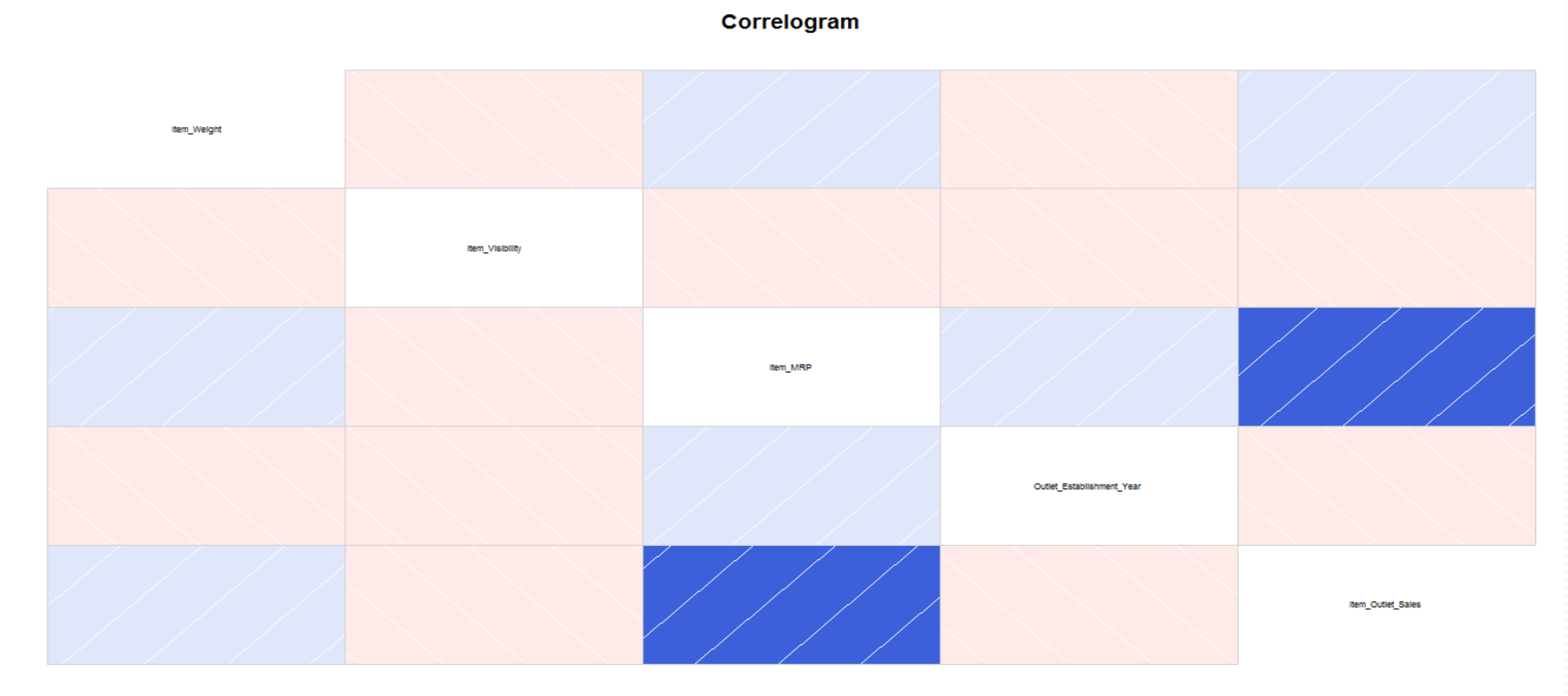
7. Correlogram: It is used to test the level of co-relation among the variable available in the data set. The cells of the matrix can be shaded or coloured to show the co-relation value.

For Big_Mart_Dataset, check co-relation between Item cost, weight, visibility along with Outlet establishment year and Outlet sales from below plot.

R Code for simple correlogram using function corrgram():

```
> install.packages("corrgram")
> library(corrgram)
> corrgram(train, order=NULL, panel=panel.shade, text.panel=panel.txt,
main="Correlogram")
```

DATA VISUALIZATION



- Darker the colour, higher the co-relation between variables. Positive co-relations are displayed in blue and negative correlations in red colour. Colour intensity is proportional to the co-relation value.
- We can see that Item cost & Outlet sales are positively correlated while Item weight & its visibility are negatively correlated.

DATA PRE-PROCESSING

DATA PREPROCESSING

1. Missing value replenishment
2. Transformation or normalization
3. Random Sampling

Missing Value Handling

Example: Suppose a telecom company wants to analyze the performance of its circles based on the following parameters

1. Current Month's Usage
2. Last 3 Month's Usage
3. Average Recharge
4. Projected Growth

The data set is given in next slide. (Missing_Values_Telecom Data)

Missing Value Handling

Example: Circle wise Data

SL No.	Current Month's Usage	Last 3 Month's Usage	Average Recharge	Projected Growth	Circle
1	5.1	3.5	99.4	99.2	A
2	4.9	3	98.6	99.2	A
3		3.2		99.2	A
4	4.6	3.1	98.5	9..2	A
5	5		98.4	99.2	A
6	5.4	3.9	98.3	99.4	A
7	7	3.2	95.3	98.4.	B
8	6.4	3.2	95.5	98.5	B
9	6.9	3.1	95.1	98.5	B
10		2.3	96	98.3	B
11	6.5	2.8	95.4	98.5	B
12	5.7		95.5	98.3	B
13	6.3	3.3		98.6	B
14	6.7	3.3	94.3	97.5	C
15	6.7	3	94.8	97.3	C
16	6.3	2.5	95	98.9	C
17		3	94.8	98	C
18	6.2	3.4	94.6	97.3	C
19	5.9	3	94.9	98.8	C

Missing Value Handling

Example: Read data and variables to R

```
> mydata = Missing_Values_Telecom
> cmusage = mydata[,2]
> l3musage = mydata[,3]
> avrecharge = mydata[,4]
```

Missing Value Handling

Option 1: Discard all records with missing values

```
>newdata = na.omit(mydata)
>write.csv(newdata,"E:/IIFT/newdata.csv")
```

SL.No.	Current.Month.s.Usage	Last.3.Month.s.Usage	Average.Recharge	Projected.Growth	Circle
1	5.1	3.5	99.4	99.2	A
2	4.9	3	98.6	99.2	A
4	4.6	3.1	98.5	9..2	A
6	5.4	3.9	98.3	99.4	A
7	7	3.2	95.3	98.4.	B
8	6.4	3.2	95.5	98.5	B
9	6.9	3.1	95.1	98.5	B
11	6.5	2.8	95.4	98.5	B
14	6.7	3.3	94.3	97.5	C
15	6.7	3	94.8	97.3	C
16	6.3	2.5	95	98.9	C
18	6.2	3.4	94.6	97.3	C
19	5.9	3	94.9	98.8	C

Missing Value Handling

Option 2: Replace the missing values with variable mean, median, etc

Replacing the missing values with mean

Compute the means excluding the missing values

```
>cmusage_mean = mean(cmusage, na.rm = TRUE)
>l3musage_mean = mean(l3musage, na.rm = TRUE)
> avrecharge_mean = mean(avrecharge, na.rm = TRUE)
```

Replace the missing values with mean

```
> cmusage[is.na(cmusage)]=cmusage_mean
> l3musage[is.na(l3musage)]= l3musage_mean
>avrecharge[is.na(avrecharge)]=avrecharge_mean
```

Missing Value Handling

Option 2: Replace the missing values with variable mean, median, etc

Replacing the missing values with mean

Replace the missing values with mean

```
> cmusage[is.na(cmusage)]=cmusage_mean
> l3musage[is.na(l3musage)]= l3musage_mean
>avrecharge[is.na(avrecharge)]=avrecharge_mean
```

Making the new file

```
> mynewdata = cbind(cmusage, l3musage, avrecharge, mydata[,5],mydata[,6])
> write.csv(mynewdata, "E:/ISI/mynewdata.csv")
```

Missing Value Handling

Replacing the missing values with men

Option 2: Replace the missing values with variable mean, median, etc

SL No	cmusage	l3musage	avrecharge	Proj Growth	Circle
1	5.1	3.5	99.4	11	1
2	4.9	3	98.6	11	1
3	5.975	3.2	96.14117647	11	1
4	4.6	3.1	98.5	1	1
5	5	3.105882353	98.4	11	1
6	5.4	3.9	98.3	12	1
7	7	3.2	95.3	6	2
8	6.4	3.2	95.5	7	2
9	6.9	3.1	95.1	7	2
10	5.975	2.3	96	5	2
11	6.5	2.8	95.4	7	2
12	5.7	3.105882353	95.5	5	2
13	6.3	3.3	96.14117647	8	2
14	6.7	3.3	94.3	3	3
15	6.7	3	94.8	2	3
16	6.3	2.5	95	10	3
17	5.975	3	94.8	4	3
18	6.2	3.4	94.6	2	3
19	5.9	3	94.9	9	3

TRANSFORMATION / NORMALIZATION

z transform:

Transformed data = (Data – Mean) / SD

Exercise : Normalize the variables in the Supply_Chain.csv ?

Read the files

```
>mydata  = Supply_Chain
> mydata = mydata[,2:7]
```

Normalize or standardize the variable

```
>mystddata = scale(mydata)
```

RANDOM SAMPLING

Example: Take a sample of size 60 (10%) randomly from the data given in the file bank-data.csv and save it as a new csv file?

Read the files

```
>mydata  = bank-data
> mysample = mydata[sample(1:nrow(mydata), 60, replace = FALSE),]
>write.csv(mysample,"E:/IIFT/mysample.csv")
```

Example: Split randomly the data given in the file bank-data.csv into sets namely training (75%) and test (25%) ?

Read the files

```
>mydata  = bank-data
>sample = sample(2, nrow(mydata), replace = TRUE, prob = c(0.75, 0.25))
> sample1 = mydata[sample ==1, ]
> sample2 = mydata[sample ==2,]
```

Any question?

You may also send your question(s) at ctanujit@gmail.com