R for Exploratory Data Analytics

Brief Introduction to R

What is R?

1. Powerful and widely used statistical software. A language developed by statisticians for statistics

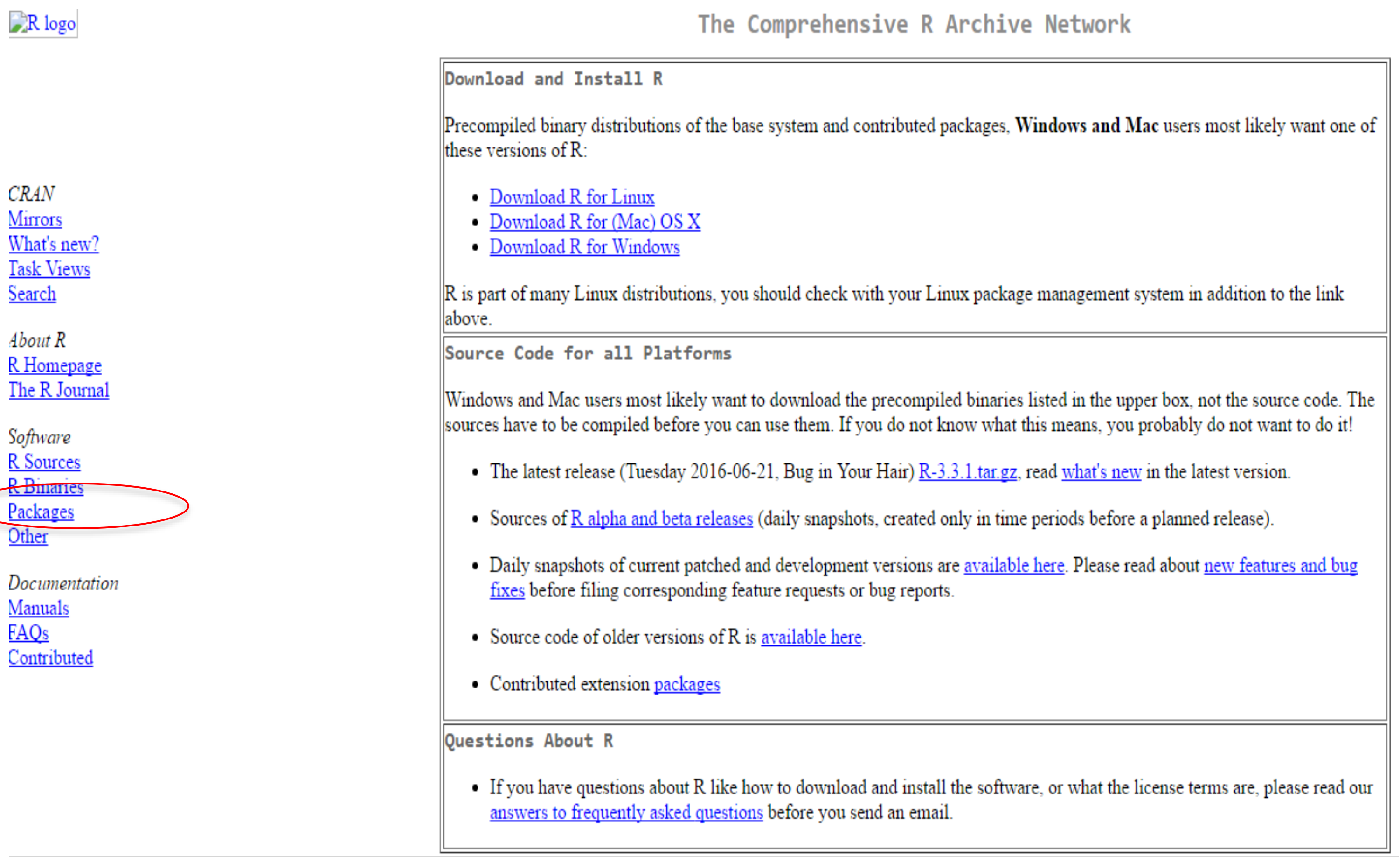
2. R is a GNU project. The source code for the R software environment is written primarily in C, Fortran, and R. R is freely available under the GNU General Public License

3. R language is maintained over a global network of web and http servers known as CRAN (Comprehensive R Archive Network) https://cran.r-project.org/mirrors.html

Downloading R for machine learning

1. Download and install in C:\Program Files\R\R-3.5.1
2. R Studio - IDE for R https://www.rstudio.com/products/rstudio/download2/#download

CRAN … an introduction



R logo

The Comprehensive R Archive Network

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (Tuesday 2016-06-21, Bug in Your Hair) R-3.3.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

Packages for machine learning – https://cran.r-project.org/index.html

R & R Studio

R – Some KeyPoints

1. Workspace - The workspace is the area of your computer's memory (RAM) where R is storing the variables and other things you are currently working with

2. To see the contents of the workspace issue a ls() or objects() command at the prompt

3. Keep your workspace clean. Remove objects after use to release memory

4. Caution- In R removing stuff from workspace is permanent. It cannot be recovered

5. If you wish to re-use the object, save it before removing it. save (object_name, file =“myRobjects.data")

6. If you wish to save the entire workspace, save.image(). This will save the workspace in current working director in a hidden file .Rdata

7. R uses a working directory on the hard disk which is the default directory where it will look for user defined files.

8. To see the contents of current working directory – dir(). To check the current working directory getwd(). To set it setwd(give path in quotes) e.g setwd(“c:/mydocuments”)

R – Some KeyPoints

9. SearchPath() – when you do any operations on an object, R looks for that object in the search path. It will also look for the function you are executing in the search path.

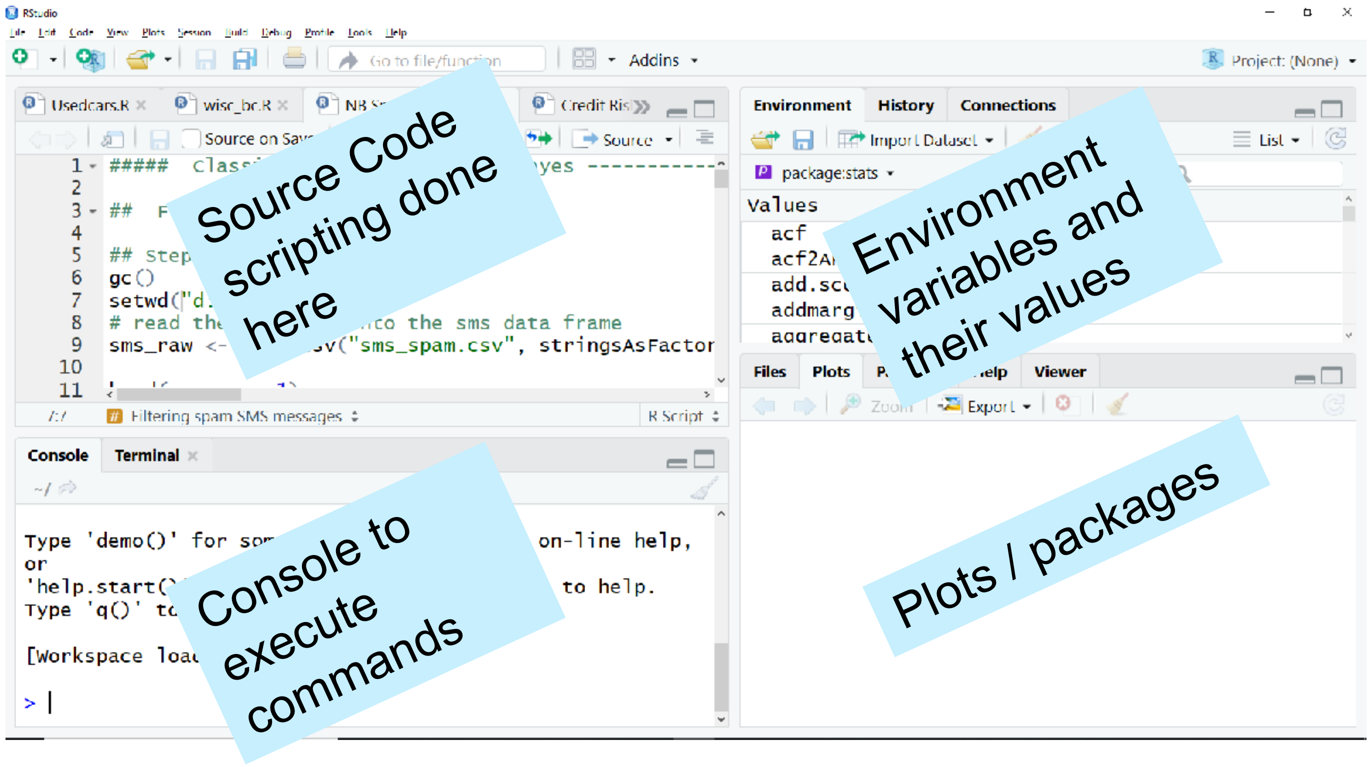
10. If the object is not found in the search path, it will return an error "object not found" or "function not found"

11. Search() at the prompt will list all the packages and global env (workspace) in which the object and the function will be searched. One can add new folders / packages to the search path

12. To get help on any function anywhere in the code, type help(“function name”) e.g. help("mean"). This will open up the help manual on the “Help” tab on the right

Rstudio – Some KeyPoints

1. RStudio is an integrated development environment (IDE) that allows you to interact with R more readily



```
##### Class                    yes -----------

## F

## Step
gc()
setwd("d.
# read the                nto the sms data frame
sms_raw <-             sv("sms_spam.csv", stringsAsFactor
```

```
Values
  acf
  acf2AR
  add.sc
  addmarg
  aggregat
```

```
Type 'demo()' for som                on-line help,
or
'help.start()                         to help.
Type 'q()' to

[Workspace loa

> |
```

Rstudio – Some KeyPoints

2. Before you begin working in R, you should set your working directory (a folder to hold all of your project files); for example, “C:\workspace\…”

3. This directory is the location where all your input data-sets are be stored

2. It also serves as the default location for plots and other objects exported from R

3. If set, one can import data into R with just a file name. To change the working directory in RStudio, select the **Files Tab > More > Set As Working Directory**

4. To check what the current working directory is, use getwd() in R. To change the workdirectory in R, use setwd(“c:/… “) **NOTE:** the forward slash

5. To import a file, say a csv, sample_df <- read.csv("C:/workspace/sand_example.csv")

6. Always view the data to ensure R has imported the data properly (though R can import data from Excel too, that may be risky given the default behavior of Excel) For e.g. if you enter 11 June in a cell, it becomes 6/11/2018 though it is displayed as 11 June (Ref: https://www.washingtonpost.com/news/wonk/wp/2016/08/26/an-alarming-number-of-scientific-papers-contain-excel-errors/?utm_term=.8a4a41aadf4f)

Rstudio – Some KeyPoints

2. Before executing R functions at the Console, they are usually written down (or scripted in the scripting window (“Source Code”)

3. The scripting window reflects the full flow / sequence of functions that make a complete job. It helps to document the work

4. In R, several types of file make one job. The file types include: workspace, script, history, and graphics. It is important to save these files regularly.

5. R scripts -
   a. To save R scripts in RStudio, simply **click the save button** from your R script tab. Save scripts with the .R extension

6. R workspace
   a. The R workspace consists of all the data objects created or loaded during R session.

   b. R saves a file named .RData to the working directory when we quit the session and respond positively to it’s query on whether we want to save the workspace

.

Rstudio – Some KeyPoints

c. Saved .Rdata can be used to reload the workspace and get all the data back in subsequent session, all of the commands typed in previous session will be accessible by using the up-arrow and down-arrow keys on your keyboard.

d. You can also save or load your workspace at any time during your R session from the menu by clicking Session>Save Workspace As.., or the save button on the Environment Tab

8. R history file -
   a. An R history file is a copy of all key strokes. It is like a log file of the functions/ commands executed in a session.

   b. Rhistory file is simply a text file that lists all of the commands executed in the session. It does not store the results of the commands

   c. To load or save your R history <u>from the History Tab click the **Open File**</u> or **Save** button. If you load an Rhistory file, your previous commands will again become available with the up-arrow and down-arrow keys

   d. Can save and load using functions in the script
      - savehistory(file = “model1.Rhistory")
      - loadhistory(file = “model1.Rhistory")

R Data Structures

R Data Structures

R offers multiple data structures –

1. atomic vector
2. list
3. matrix
4. data frame
5. Factors

Vector refers to a collection of data elements. There are two types of vectors

1. Atomic vector (a vector consisting of data elements of a single type)
2. Lists (often lists are referred to as something different from vectors. It is different from atomic vectors)

Atomic vectors can be of different types / modes

1. Character
2. Logical
3. Integer
4. numeric

Most used R data structures in machine learning

1. x <- c(98.1, 98.6, 101.4)
    a. Will create a numeric atomic vector
    b. typeof(x) will return “double”

2. x <- c(1,2,3,4)
    a. will create a numeric vector! Not integer vector
    b. typeof(x) will return “double”

3. x <- c(1L, 2L, 3L, 4L)
    a. Will create vector of integers
    b. typeof(x) will return “integers”

4. Emp <- list(c(1, “Ganu", 54, "ML"))
    a. Will create a list of values
    b. Typeof(Emp) will return “list”

Most used R data structures in machine learning

5. Gender <- factor(c("Male", "Female"))
   a. Will create a factor variable with two values
   b. typeof(Gender) will return “integer”!
   c. Factor variables are internally stored as numbers
   d. Female will get code of 0 and Male of 1 given the ASCII order

6. dat <- data.frame(id = letters[1:10], x = 1:10, y = 11:20)
   a. Will create a dataframe with three columns
   b. typeof(dat) will return “list”
   c. even when all columns are of same type, typeof(dat) will return “list”

7. m <- matrix(nrow = 2, ncol = 2)
   a. Will create a matrix of dimension 2 X 2
   b. Typeof(m) will return “logical”! depends on what type of data is store in the matrix
   c. m <- matrix(data= c(2, 4, 6, 8), nrow = 2, ncol = 2,) will be of type double
   d. m <- matrix(data= c(2L, 4L, 6L, 8L), nrow = 2, ncol = 2,) wile be of integer type

R datatypes are objects!

1. Everything in R is an object. Variables are assigned R-Objects and the data type of the R-object becomes the data type of the variable

2. While most programming languages have a single class system, R has three class systems which are S3, S4 and Reference class

3. S3 class is primitive. It lacks a formal definition and object of this class can be created simply by adding a class attribute to it

    a. Emp <- list(c(1, “Ganu", 54, "ML")) …. In this example “list” is S3 class

4. S4 class is an improvement over S3. These classes are formally defined with a fixed structure. This helps in making object of the same class look same

    a. setClass("student", slots=list(name="character", age="numeric", GPA="numeric"))
    b. s <- new("**<u>student</u>**",name="John", age=21, GPA=3.5)
    c. s is a S4 object
    d. s <- new("student",name="John", age=21, GPA=3.5, rank = 1) will return error
    e. Since “rank” is not part of the “student” class
    f. Each field in the class definition is called a slot

R datatypes are objects!

5. Reference class are internally implemented as S4 classes with an environment added to it. Unlike S3 and S4 classes, methods belong to class rather than generic functions.
   a. student <- setRefClass("stud",fields = list(name = "character", age = "numeric", GPA = "numeric"))
   b. To access individual field of the object, use $ e.g. s$age
   c. To modify the value of a field, s$age <- 40
   d. With reference class, only a single copy of the object (student in above example) exists. If we copy the object variable to another object say, student1, student1 is only a pointer
      i. s1 <- s
      ii. s1$age = 40
      iii. student$age will get modified to 40

Note: For our data science purpose, we will use only S3 objects

Data Ingestion into R

R interfaces to external data sources

1. To load .csv file with column headers in first row, fields comma separated, first row contains variable names, comma is separator note the / instead of \ on windows systems

   ```
   wbcd <- read.csv("wisc_bc_data.csv", stringsAsFactors = FALSE)
   ```

2. The read.table function is used when reading in ASCII files that contain rectangular data. T he default delimiter is blank space; other delimiters must be specified by using the sep option. For e.g.

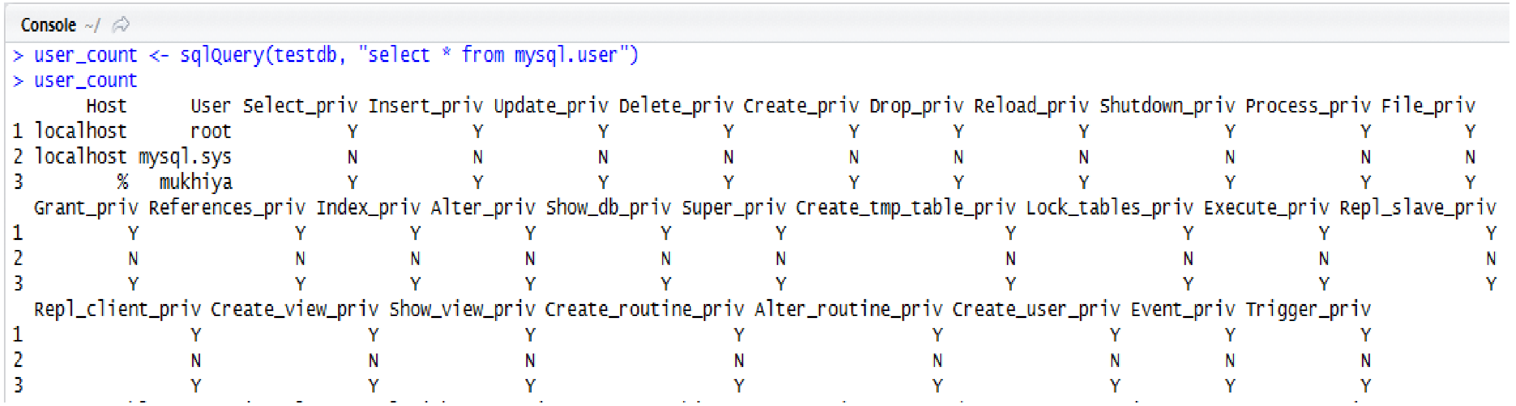
   ```
   wisc_df <- read.table("c:/wisc_bc_data.csv", header=TRUE, sep=",")
   ```

3. To read in the first worksheet from the workbook *abcd.xlsx where f*irst row contains variable names

   ```
   install.packages("XLConnect")
   library(XLConnect)
   wisc_df <- readWorksheetFromFile("c:/abcd.xlsx", sheet = 1)

   # read in the worksheet named wcbc
   wisc_df <- readWorksheetFromFile( "c:/abcd.xlsx", sheetName = “wcbc")
   ```

Data Ingesting into R from databases -

1. To ingest data from RDBMS, use RODBC or RJDBC
   a. install.packages("RODBC")
   b. library(RODBC)
   c. mycon <- odbcConnect("mysql", uid = "root", pwd = "root")
   d. emp <- sqlFetch(mycon, “testdb.employees") // testdb is a database on mysql and employees is a table in testdb.
   e. emp_count <- sqlQuery(myconn, "select count(*) from testdb.t1") // to access select rows from a table



```
> user_count <- sqlQuery(testdb, "select * from mysql.user")
> user_count
       Host      User Select_priv Insert_priv Update_priv Delete_priv Create_priv Drop_priv Reload_priv Shutdown_priv Process_priv File_priv
1 localhost      root           Y           Y           Y           Y           Y         Y           Y             Y            Y         Y
2 localhost mysql.sys           N           N           N           N           N         N           N             N            N         N
3         %   mukhiya           Y           Y           Y           Y           Y         Y           Y             Y            Y         Y
  Grant_priv References_priv Index_priv Alter_priv Show_db_priv Super_priv Create_tmp_table_priv Lock_tables_priv Execute_priv Repl_slave_priv
1          Y               Y          Y          Y            Y          Y                     Y                Y            Y               Y
2          N               N          N          N            N          N                     N                N            N               N
3          Y               Y          Y          Y            Y          Y                     Y                Y            Y               Y
  Repl_client_priv Create_view_priv Show_view_priv Create_routine_priv Alter_routine_priv Create_user_priv Event_priv Trigger_priv
1                Y                Y              Y                   Y                  Y                Y          Y            Y
2                N                N              N                   N                  N                N          N            N
3                Y                Y              Y                   Y                  Y                Y          Y            Y
```

Note: There are 20 different Read options that R can use to ingest data other than sqlFetch and sqlQuery. Refer too the help on “Read” and “Sql”

Exploratory Data Analytics

Exploring Data – Key activities – Know your data

1. Irrespective of where the data is stored, NoSql or HDFS or flat file or RDBMS…browse the data. If data too large, browse a sample.
2. Personal inspection helps get an idea of problems ahead (the problems may not always be visible at first sight)
3. Make sure you understand each column of the data with no ambiguity. If you find acronyms, make sure you understand their real meaning
4. For each attribute check the following –
    a. If of numeric type,
        i. is the min and max value within acceptable range. For e.g. if we are looking at “work_experience” and min is 0 while max is 15, is the 0 a missing value? Why is the max capped at 15? Why no one above 15 years
        ii. Is the range i.e (max – min) reasonable. Is it too different from expected range
        iii. If it is an “id” column, sort the data on “id” and look for breaks. If breaks in id values found, where are the other records? Was sampling biased

    If categorical type,

    1. Are the distinct categories as expected?
    2. Typos in category values? Such as “Very Good” and “VeryGood”
    3. Any missing values
5. Note down all the observation on each column. Observations that reflect potential problems

Exploring Data – Key activities – Lab-1

1. Open Wisconsin Cancer dataset

2. Set filter on all the columns

3. Start with "id" column. Sort the data on ID column. Do not save the file.
   a. What do you observe?
   b. Any questions that you would like to ask?
   c. What could your observations indicate? And what are the consequences if true.

4. Take the "Diagnosis" column. Check the distinct values. (**Note:** this is our target column)
   a. What do you observe?
   b. How does your observation matter?

5. In the "Radius" column, observe the range of values. Any questions? (we do not have the domain expertise to comment on this. The idea is to assess the validity of the range….is it possible). For instance take the highest value. How many records are there with this value?

6. What is your observation on the "concavity" and "points_mean"

Exploring Data – Key activities – Descriptive Analytics

1. Descriptive analytics refers to all the techniques applied to understand what the data is telling.

2. This helps fine tune our understanding of the data beyond the visual browsing

3. It involves analysing the columns individually (univariate analysis), analysing two variables together (bi-variate) and multiple variables together (multivariate analysis)

Exploring Data – Key activities – Descriptive Analytics – Univariate Analysis

1. We analyse each column one by one as discussed earlier. The analytical techniques depend upon the type of column, categorical or continuous

2. Start by identifying missing values, data pollution in the column. As discussed earlier, a visual inspection by setting the filters will help if it is a XL sheet

3. Numerical columns missing values can be treated in multiple ways –
   a. Replace missing value with central value
   b. Estimate the missing value based on other attributes
   c. Estimate the missing value based on other similar records
   d. Consider only records with no missing value for analysis of that column
   e. Drop records with missing values

4. Categorical columns missing values can be treated in multiple ways –
   a. Replace missing value with modal value (if few missing values and unique strong mode)
   b. Guess the missing value from other parameters
   c. Guess the missing value from other similar records
   d. Drop the records with missing values

Note: Always create new column or a new dataframe when imputing values for missing values for the research to be repeatable

Exploring Data – Key activities – Descriptive Analytics – Univariate Analysis

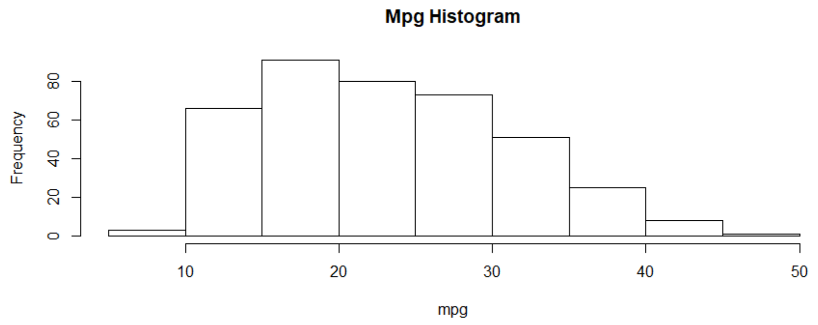
5. Identify outliers in numerical columns –
   a. Outliers are data points that lie far away from the central values
   b. They also lie far away from rest of the data points on that column
   c. Could be caused by mistake, true but rare values or mix-up of data from different sources (to be discussed later)
   d. Outliers have a negative impact on the final model making them ineffective in production

6. Box plot –
   a. Pictorial way of representing the central values and spread on a dimensions
   b. Displayed as a rectangle with median, Q1 and Q2 along with wiskers and outliers
   c. Comparing box plots helps analyse the differences in the way data is spread on various dimensions

7. Histogram –
   a. Pictorial way to represent the shape of spread
   b. Easily shows the skew in the distribution
   c. Helps compare the spread around the central value
   d. Identify long tails

Exploring Data – Key activities – Lab-2

1. Plot historgram for mpg, cyl, hp, wt, acc, columns of the mpg_df
   a. hist(autodf$mpg, main = "Mpg Histogram", xlab = "mpg")
   b. hist(autodf$cyl, main = "Cylinder Histogram", xlab = "Cyl")
   c. hist(autodf$hp, main = "HP Histogram", xlab = "HP")
   d. hist(autodf$wt, main = "Weight Histogram", xlab = "Weight")
   e. hist(autodf$acc, main = "Acceleration Histogram", xlab = "Acceleration")



2. Comment on the shape of the distribution in terms of skew, tail and length of the tail

3. Draw boxplot for the same column. Are there any outliers

Exploring Data – Key activities – Descriptive Analytics – Bivariate Analysis

1. Univariate analysis alone is not enough. We need to also understand the data in terms of central values and spread across dimensions

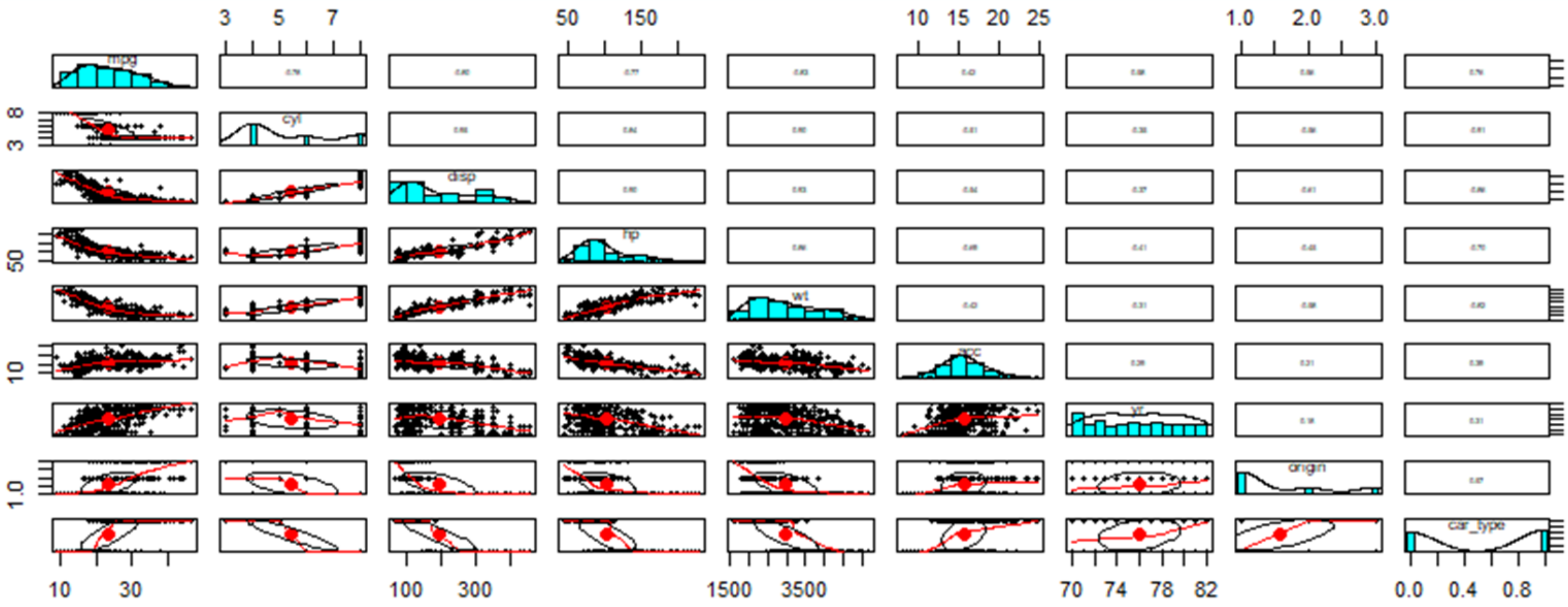
2. Bi-variate analysis helps understand how one variable influences other? It helps identify redundant attributes i.e. attributes carrying same information

3. It helps identify how the attributes influence the target variable and thus identify those attributes which are likely to be good predictors of the target variable and those which are not

4. We also get to know whether the attributes have a simple linear or non linear relation with the target variable

5. “Psych” library gives a facility to do bivariate analysis using pair panels

Exploring Data – Key activities – Descriptive Analytics – Bivariate Analysis



6. Pair panel is a square matrix with as many rows as columns\
7. Every row is also a column (row name, column name is reflected in the diagonal)
8. Diagonal elements are a row Vs itself and make no sense to analyse the relationship
9. Diagonals show how data is distributed on the given column
10. Below the diagonal is the relationship between columns, shown as scatter plot and Loess curve and above the diagonal is the same info in number (R value)

Exploring Data – Key activities – Lab-3

Objective – Explore car-mpg.csv data set

1. Open car-mpg.csv

2. Set filter on all the columns

3. Start with “mpg” column.
   a. What do you observe? Min value 9.0 , max 46.6 - Range = max – min = 37.6 units
   b. Any comments? Keep in mind, larger the spread less reliable the central values!

4. Take the “Cylinder” column. Check the distinct values
   a. What do you observe?
   b. What is the frequency of each value?

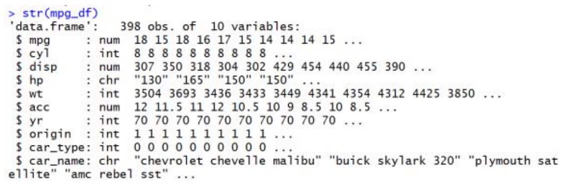
5. Comment on the “disp”, “hp”, “wt” and “acc” columns

6. What is your strategy on replacing missing values in the “hp” column?

Exploring Data – Key activities – Lab-3

1. Load car-mpg.csv into mpg_df dataframe
    1. setwd("D:/RData")   // set this to the path where you stored your data file
    2. mpg_df <- read.csv("car-mpg.csv", stringsAsFactors = FALSE

2. Understand the dataframe structure and column types
    1. str(mpg_df)



```
> str(mpg_df)
'data.frame':   398 obs. of  10 variables:
 $ mpg     : num  18 15 18 16 17 15 14 14 14 15 ...
 $ cyl     : int  8 8 8 8 8 8 8 8 8 8 ...
 $ disp    : num  307 350 318 304 302 429 454 440 455 390 ...
 $ hp      : chr  "130" "165" "150" "150" ...
 $ wt      : int  3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
 $ acc     : num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
 $ yr      : int  70 70 70 70 70 70 70 70 70 70 ...
 $ origin  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ car_type: int  0 0 0 0 0 0 0 0 0 0 ...
 $ car_name: chr  "chevrolet chevelle malibu" "buick skylark 320" "plymouth sat
ellite" "amc rebel sst" ...
```

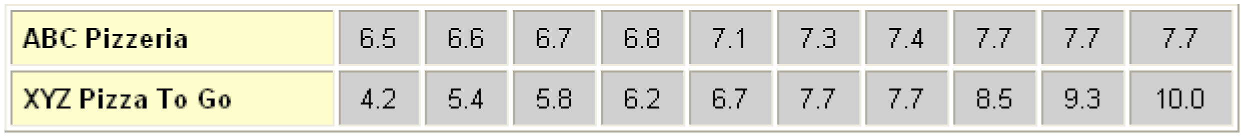
3. Some columns are not of the right type. For e.g. Cyl is a category/ factor not int, hp is numeric type not char, origin is a factor not int….
    1. mpg_df$origin <- factor(mpg_df$origin, levels=c(0,1,2), labels=c("Americas", "Europe", "Asia"))
    2. mpg_df$car_type <- factor(mpg_df$car_type, levels=c(0,1), labels=c("Automatic","Manual"))
    3. mpg_df$cyl <- factor(mpg_df$cyl,levels=c(3,4,5,6,7,8),labels=c("3cyl","4cyl","5cyl", "6cyl", "7cyl", "8cyl"))

4. Ensure the columns are of the right type using str(mpg_df) again

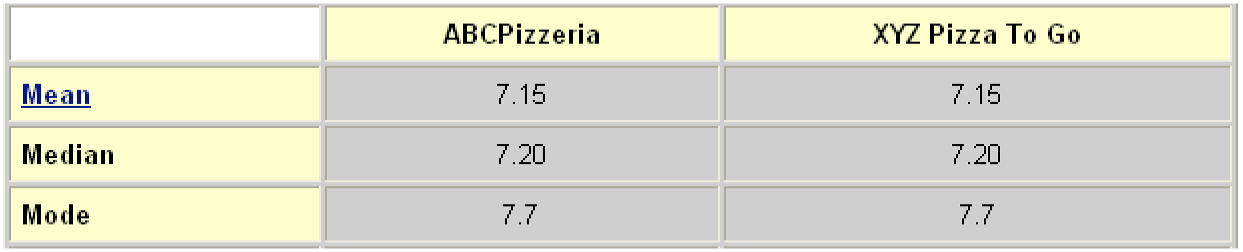
Descriptive Statistics

Descriptive statistics – central value and spread

Time to prepare the pizza



ABC Pizzeria	6.5	6.6	6.7	6.8	7.1	7.3	7.4	7.7	7.7	7.7
XYZ Pizza To Go	4.2	5.4	5.8	6.2	6.7	7.7	7.7	8.5	9.3	10.0



	ABCPizzeria	XYZ Pizza To Go
Mean	7.15	7.15
Median	7.20	7.20
Mode	7.7	7.7

1. ABC pizzeria is much more consistent than XYZ pizza in terms of time to prepare
   a. If you want to be sure about the amount of time you will need to get your lunch you will go to ABC Pizza
   b. Going to XYZ Pizza will make one less sure about the time required to gather the lunch
   c. However, if you are a risk taker, you may go to XYZ Pizza even when under time pressure as there is a chance of getting the lunch within 4.2 minutes. Saving a full 2+ minutes compared to ABC!

- In short, central tendency alone is not important, one should know the spread also to take any decision

Descriptive statistics - Variance

1. Variance is measured as the average of sum of squared difference between each data point (represented by xi) and the mean represented by μ

2. N represents the number of data points (when number of data points is small (lesser than 30), we replace N with N-1 (degrees of freedom) in the denominator only

3. Sum of difference between each data point and mean will always be zero. Hence square the difference and sum as shown above

$$\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$$

Descriptive statistics - Standard Deviation

1. Standard deviation, derived from variance is also a measure of spread. It is the square root of the variance.

$$\sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}}$$

1. Square root brings the units of measurement of spread to same as the units of central tendency

2. Continuing with the previous example, standard deviation is square root of 106.28 = 10.30

3. When the number of data points is small (less than 30), we replace N with N-1(degrees of freedom) in the denominator

Descriptive statistics – spread summary

1. Spread is measure of scatter of data points across the central value

2. Spread of data is measured using:
   a. Range
   b. Quartiles
   c. Variance
   d. Standard deviation

3. Range is the difference between highest and lowest value in a data set

4. Quartiles are physical division of the data set into four quarters

5. Second quartile is also known as median

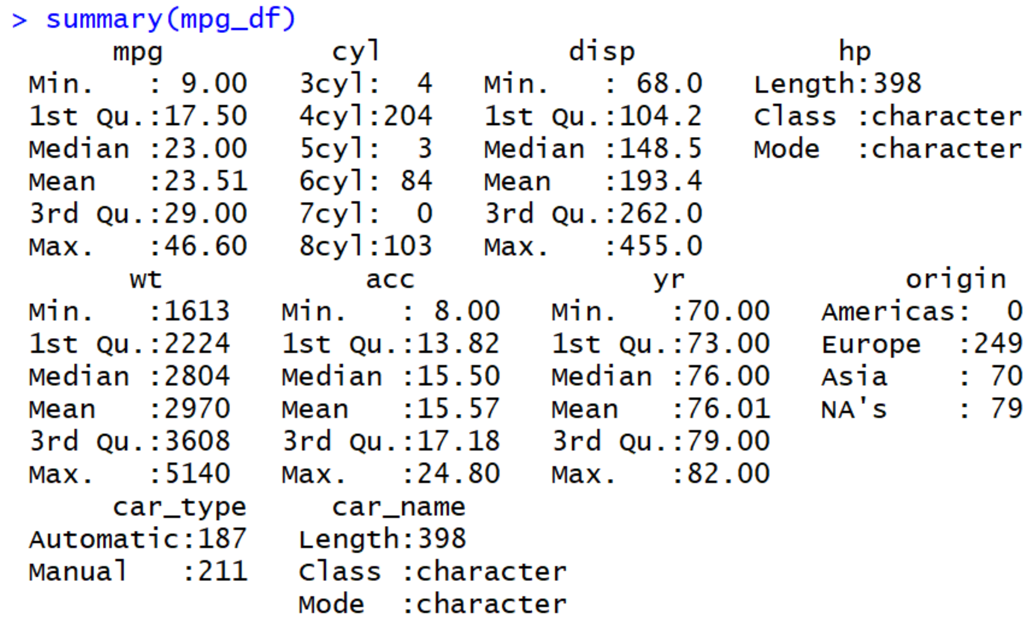
6. Variance is average of the sum of squared difference between data points and mean

7. Standard deviation is the square root of variance

Exploring Data – Key activities – Lab 4

1. Get the descriptive statistics for the columns
    1. summary(mpg_df)

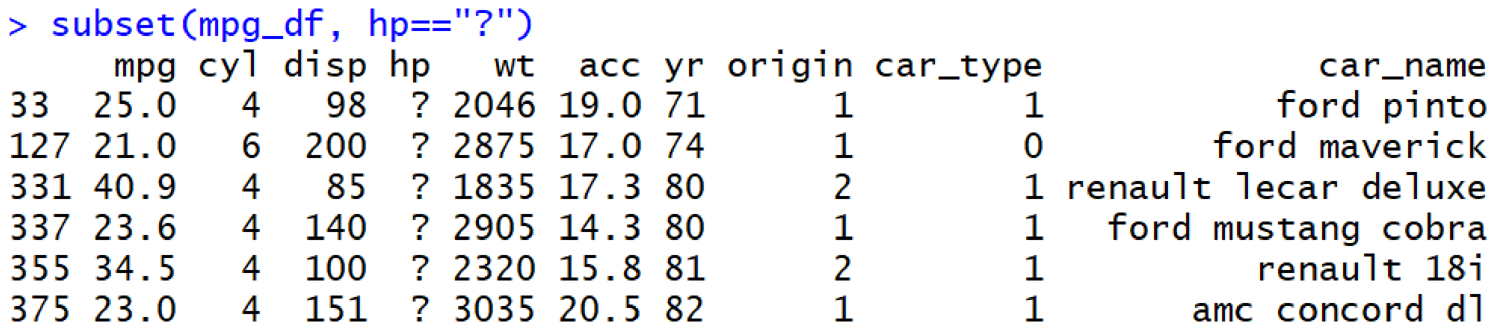
2. Analyse the output to understand how the data is distributed on each dimension



```
> summary(mpg_df)
      mpg             cyl           disp              hp           
 Min.   : 9.00   3cyl:  4   Min.   : 68.0   Length:398        
 1st Qu.:17.50   4cyl:204   1st Qu.:104.2   Class :character  
 Median :23.00   5cyl:  3   Median :148.5   Mode  :character  
 Mean   :23.51   6cyl: 84   Mean   :193.4                     
 3rd Qu.:29.00   7cyl:  0   3rd Qu.:262.0                     
 Max.   :46.60   8cyl:103   Max.   :455.0                     
       wt            acc              yr              origin   
 Min.   :1613   Min.   : 8.00   Min.   :70.00   Americas:  0  
 1st Qu.:2224   1st Qu.:13.82   1st Qu.:73.00   Europe  :249  
 Median :2804   Median :15.50   Median :76.00   Asia    : 70  
 Mean   :2970   Mean   :15.57   Mean   :76.01   NA's    : 79  
 3rd Qu.:3608   3rd Qu.:17.18   3rd Qu.:79.00                 
 Max.   :5140   Max.   :24.80   Max.   :82.00                 
      car_type      car_name        
 Automatic:187   Length:398        
 Manual   :211   Class :character  
                 Mode  :character  
```

Exploring Data – Key activities – Lab-4

3. Hp column is showing up as character. This clearly indicates problem in the data
   1. subset(mpg_df, hp=="?")



```
> subset(mpg_df, hp=="?")
     mpg cyl disp hp   wt  acc yr origin car_type             car_name
33  25.0   4   98  ? 2046 19.0 71      1        1           ford pinto
127 21.0   6  200  ? 2875 17.0 74      1        0        ford maverick
331 40.9   4   85  ? 1835 17.3 80      2        1 renault lecar deluxe
337 23.6   4  140  ? 2905 14.3 80      1        1   ford mustang cobra
355 34.5   4  100  ? 2320 15.8 81      2        1         renault 18i
375 23.0   4  151  ? 3035 20.5 82      1        1        amc concord dl
```

4. We have to replace “?” (missing values) with estimated values.

Exploring Data – Key activities – Lab-4

1. setwd("D:/OML/R based ML/RData")

2. autodf <- read.csv("car-mpg.csv", stringsAsFactors = FALSE)

3. head(autodf) # Check whether data loads

4. lapply(autodf, class) # Check the column types "Cyl", "Origin", "car_type" are category
5.   # horsepower is supposed to be numeric. That it is character, shows data problem

6. summary(autodf) # basic statistical information on all columns. Any missing column indicates problem

7. subset(autodf, is.na(as.numeric(hp))) #hp col has data problem. Let us check what is the problem
8. #Convert colums into appropriate types

9. mpg_df$cyl <- factor(mpg_df$cyl,levels=c(3,4,5,6,7,8), labels=c("3cyl","4cyl","5cyl", "6cyl", "7cyl", "8cyl"))

10. mpg_df$car_type <- factor(mpg_df$car_type, levels=c(0,1), labels=c("Automatic","Manual"))

11. mpg_df$origin <- factor(mpg_df$origin, levels=c(0,1,2), labels=c("Americas", "Europe", "Asia"))

12. autodf$hp <- as.numeric(autodf$hp) #Convert the hp data type to numeric from char

13. sum(is.na(autodf$hp)) #Check for number of NA's in horsepower attribute

Exploring Data – Key activities – Lab-4

14. median(autodf$hp,na.rm=T) #Check for median and replace NA values with median

15. # All those records where autodf$hp column is na, replace with Median calculated after removing na
16. autodf$hp[is.na(autodf$hp)]<-median(autodf$hp,na.rm=T)

17. summary(autodf) # This time we should be able to see the hp column statistics

18. autodf <- autodf[,-10]   # Drop the car name column.
19. head(autodf) # Check the columns now

20. plot(autodf) # a pair plot facility ... not appealing

21. install.packages("psych") # Install this package for pair plot
22. library(psych) # load the library

23. pairs.panels(autodf[c("mpg", "cyl", "disp", "hp", "wt", "acc", "yr", "origin", "car_type")])

24. # the various dimensions seem to have some influence on mpg. Let us explore each dimensions individually (Univariate Analysis)

Exploring Data – Key activities – Lab-4

25) hist(autodf$mpg, main = "Mpg Histogram", xlab = "mpg")

26) hist(autodf$cyl, main = "Cylinder Histogram", xlab = "Cyl")

27) hist(autodf$hp, main = "HP Histogram", xlab = "HP")

28) hist(autodf$wt, main = "Weight Histogram", xlab = "Weight")

29) hist(autodf$acc, main = "Acceleration Histogram", xlab = "Acceleration")

30) hist(autodf$yr, main = "Year Histogram", xlab = "Year")

31) hist(autodf$origin, main = "Origin", xlab = "Origin")

32) # Some histograms are showing long tails. Let us analyze them using boxplot to check for outliers.

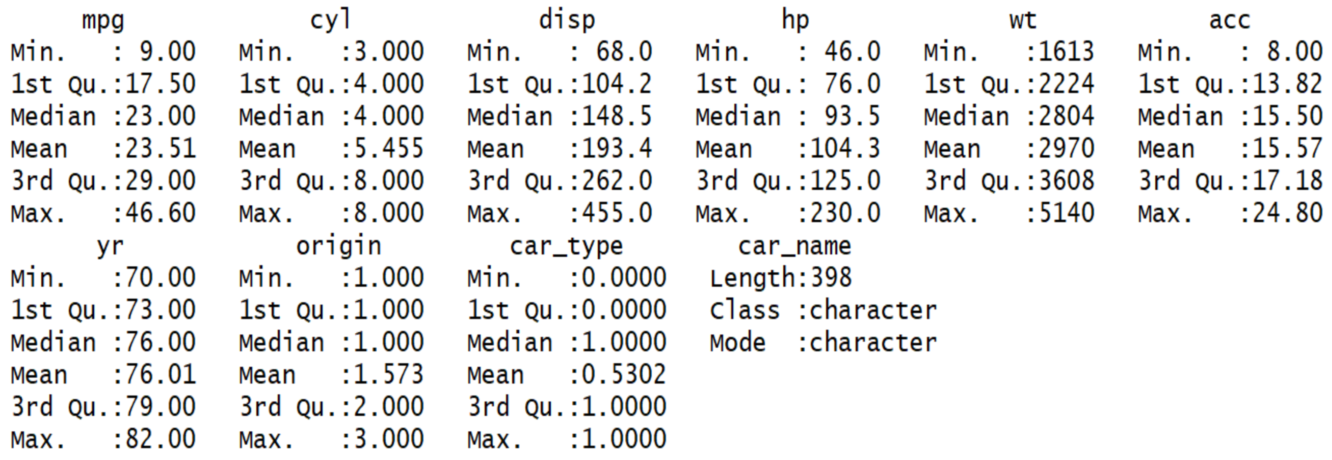
33) boxplot(mpg ~ cyl, data = autodf, xlab = "Number of Cylinders", ylab = "Miles Per Gallon", main = "Mileage Data")

34) boxplot(mpg ~ origin, data = autodf, xlab = "Origin", ylab = "Miles Per Gallon", main = "Mileage Data")

Exploring Data – Key activities – Descriptive Analytics

1. Use descriptive statistics to understand your data better by studying the
   a. Central values and their differences
   b. Spread of data on each attributes
   c. Skew in the data distribution
   d. Identify potential outliers

2. To get the descriptive statistical information
   a. summary(autodf)



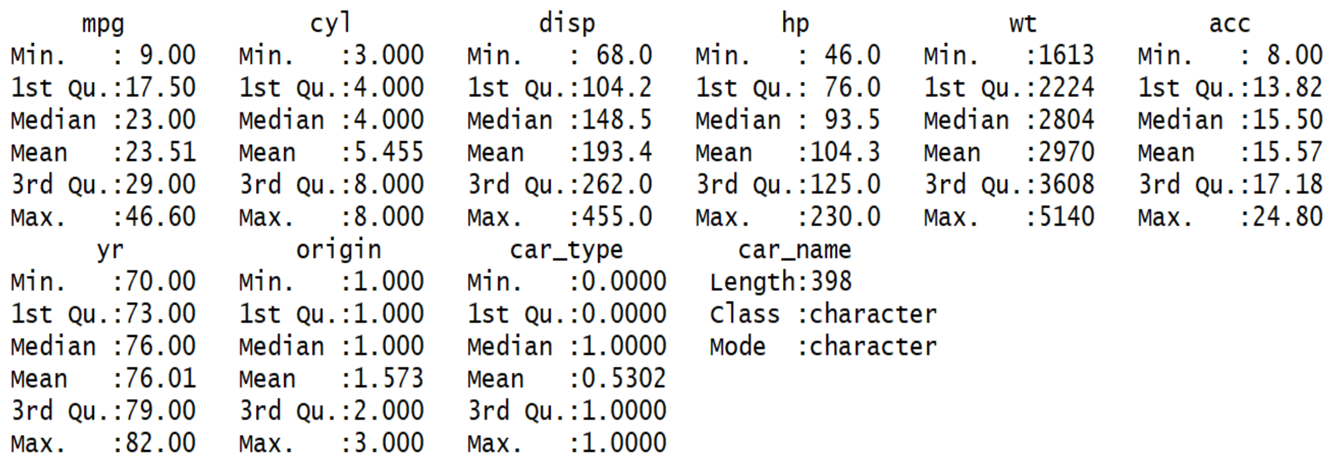
```
      mpg             cyl             disp             hp               wt            acc
 Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613   Min.   : 8.00
 1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.2   1st Qu.: 76.0   1st Qu.:2224   1st Qu.:13.82
 Median :23.00   Median :4.000   Median :148.5   Median : 93.5   Median :2804   Median :15.50
 Mean   :23.51   Mean   :5.455   Mean   :193.4   Mean   :104.3   Mean   :2970   Mean   :15.57
 3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0   3rd Qu.:125.0   3rd Qu.:3608   3rd Qu.:17.18
 Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140   Max.   :24.80
       yr            origin          car_type          car_name
 Min.   :70.00   Min.   :1.000   Min.   :0.0000   Length:398
 1st Qu.:73.00   1st Qu.:1.000   1st Qu.:0.0000   Class :character
 Median :76.00   Median :1.000   Median :1.0000   Mode  :character
 Mean   :76.01   Mean   :1.573   Mean   :0.5302
 3rd Qu.:79.00   3rd Qu.:2.000   3rd Qu.:1.0000
 Max.   :82.00   Max.   :3.000   Max.   :1.0000
```

Exploring Data – Key activities – Descriptive Analytics

3. Check if all the columns you expected are reflected in the output. If any numeric column is missing, most likely it is due to some data pollution

4. Compare the mean and the median in all the columns. If mean and median coincide (rare but possible) then there is no skew (deviation from symmetrical distribution) For e.g. in the “mpg” column the mean and median are almost same.

5. Since the mean is slightly greater than median, it indicates a skew though not significant



```
      mpg              cyl             disp              hp               wt             acc       
 Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613   Min.   : 8.00  
 1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.2   1st Qu.: 76.0   1st Qu.:2224   1st Qu.:13.82  
 Median :23.00   Median :4.000   Median :148.5   Median : 93.5   Median :2804   Median :15.50  
 Mean   :23.51   Mean   :5.455   Mean   :193.4   Mean   :104.3   Mean   :2970   Mean   :15.57  
 3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0   3rd Qu.:125.0   3rd Qu.:3608   3rd Qu.:17.18  
 Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140   Max.   :24.80  
       yr            origin          car_type         car_name        
 Min.   :70.00   Min.   :1.000   Min.   :0.0000   Length:398        
 1st Qu.:73.00   1st Qu.:1.000   1st Qu.:0.0000   Class :character  
 Median :76.00   Median :1.000   Median :1.0000   Mode  :character  
 Mean   :76.01   Mean   :1.573   Mean   :0.5302                     
 3rd Qu.:79.00   3rd Qu.:2.000   3rd Qu.:1.0000                     
 Max.   :82.00   Max.   :3.000   Max.   :1.0000                     
```

Visual Techniques

Exploring Data – Key activities – ggplot

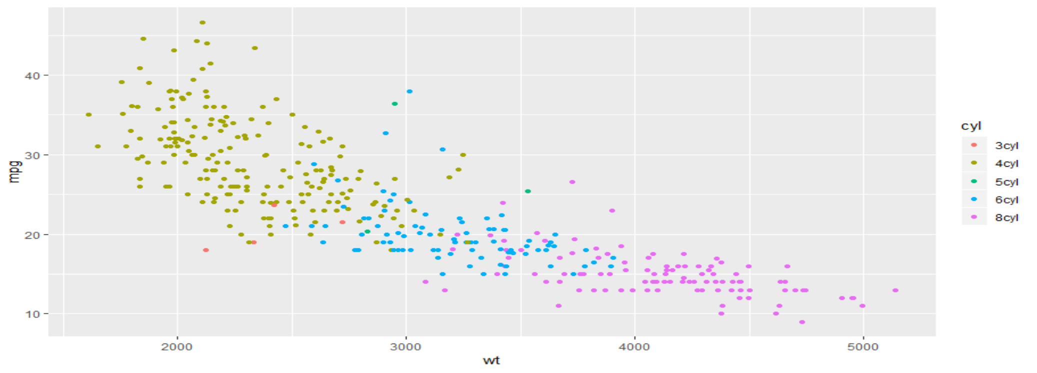
1. ggplot implements “grammar of graphics”, a coherent system for describing and building graphs. Ref A layered grammar of graphics

2. ggplot is a core library of a package called tidyverse. Install and load this library

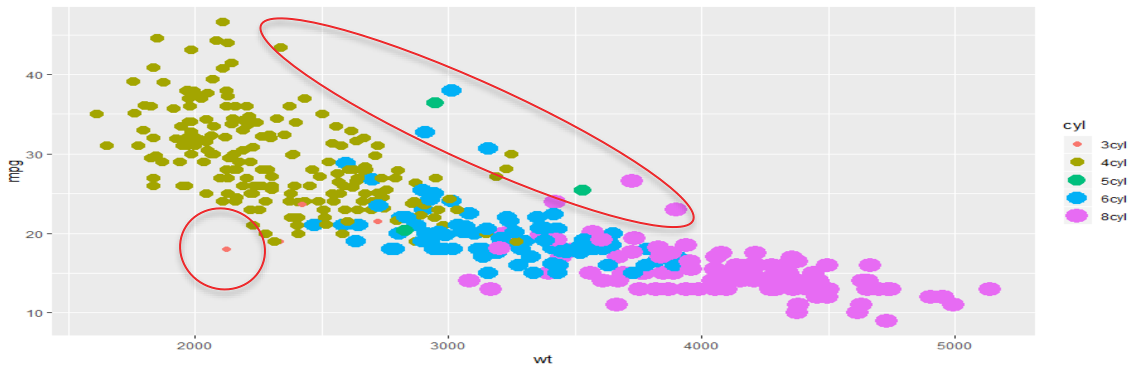
3. Plotting begins with a call to ggplot where ggplot() creates a coordinate system. Layers of information are added to this system
   a. ggplot(data=autodf)+ geom_point(mapping= aes(x = wt, y= mpg, color=origin))
   b. The first argument of ggplot is the dataset to use. This is used to build the coordinate system
   c. The function geom_point() adds a layer of points for every instance of the combination of values of the coordinates found in the dataset
   d. The geom function takes a mapping argument. This defines how the variables in the dataset are mapped to visual properties
   e. The mapping argument is always paired with another function aes() which stands for aesthetics
   f. The aes function defines the X coordinate and the Y coordinate and other visual properties such as color

Exploring Data – Key activities – ggplot

4. ggplot(data=autodf)+ geom_point(mapping= aes(x = wt, y= mpg, color=cyl))



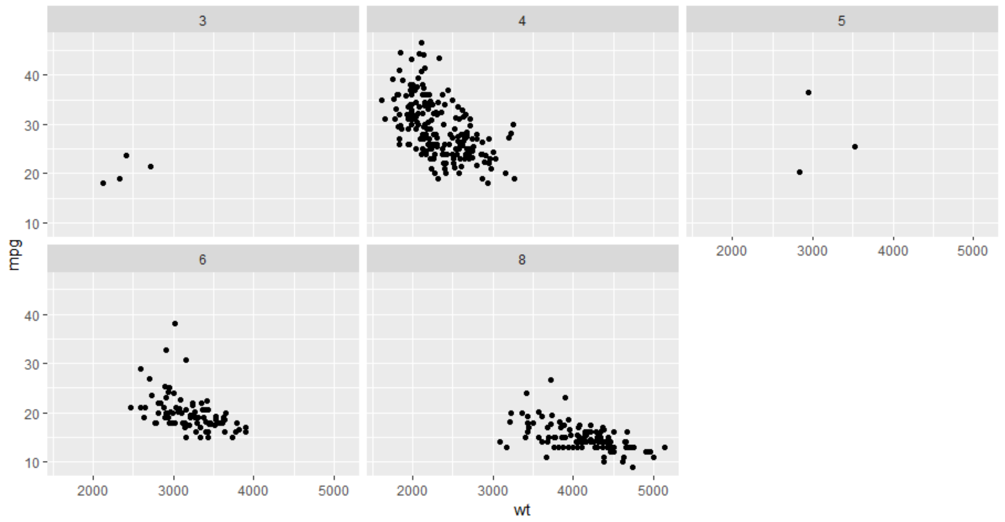
5. We can even map the column “cyl” to size. In that case the size will reflect the class of the car in terms of cylinders.
6. ggplot(data=autodf)+ geom_point(mapping= aes(x = wt, y= mpg, color= cyl, size=cyl))



Note the outliers

Exploring Data – Key activities – ggplot

7. Facets – sometimes its useful to split the graphics based on certain criteria such as cylinder and observe the function within each graph separately. Facets helps –
   a. ggplot(data=autodf)+ geom_point(mapping= aes(x = wt, y= mpg)) + facet_wrap(~ cyl, nrow=2)



8. Facets allow us to explore hidden relations for e.g. in the image above, it is clear that the degree of association between mpg and wt is different for diff cylinder cars.
   a. Cars with more cylinders have flatter relation while cars with less cylinders have a steeper negative slope

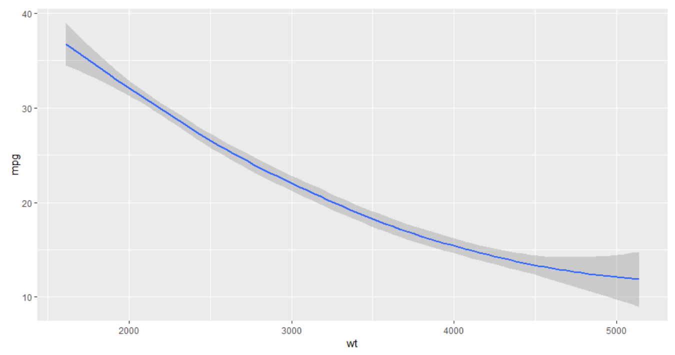
Exploring Data – Key activities – ggplot

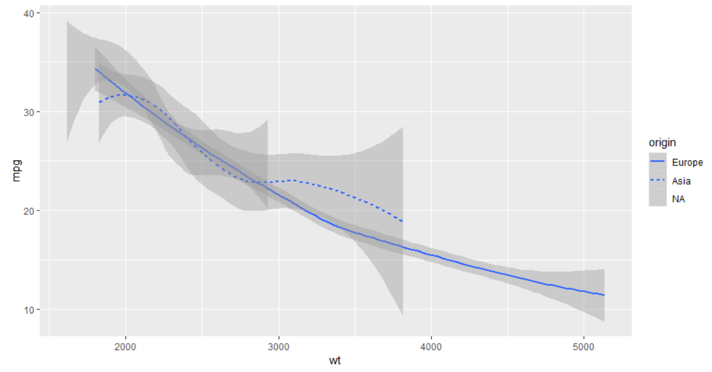
9. Geom is a geometrical object that is used to represent a data in the ggplot

10. Plots are often described in terms of the geometrical objects used to plot. For e.g. bar charts use bars, boxplot use box, line charts use line. Scatter plots use points

11. To change the type of the plot, change the geom function. For instance to get a line plot-
    a. ggplot(data=autodf)+ geom_smooth(mapping= aes(x = wt, y= mpg))

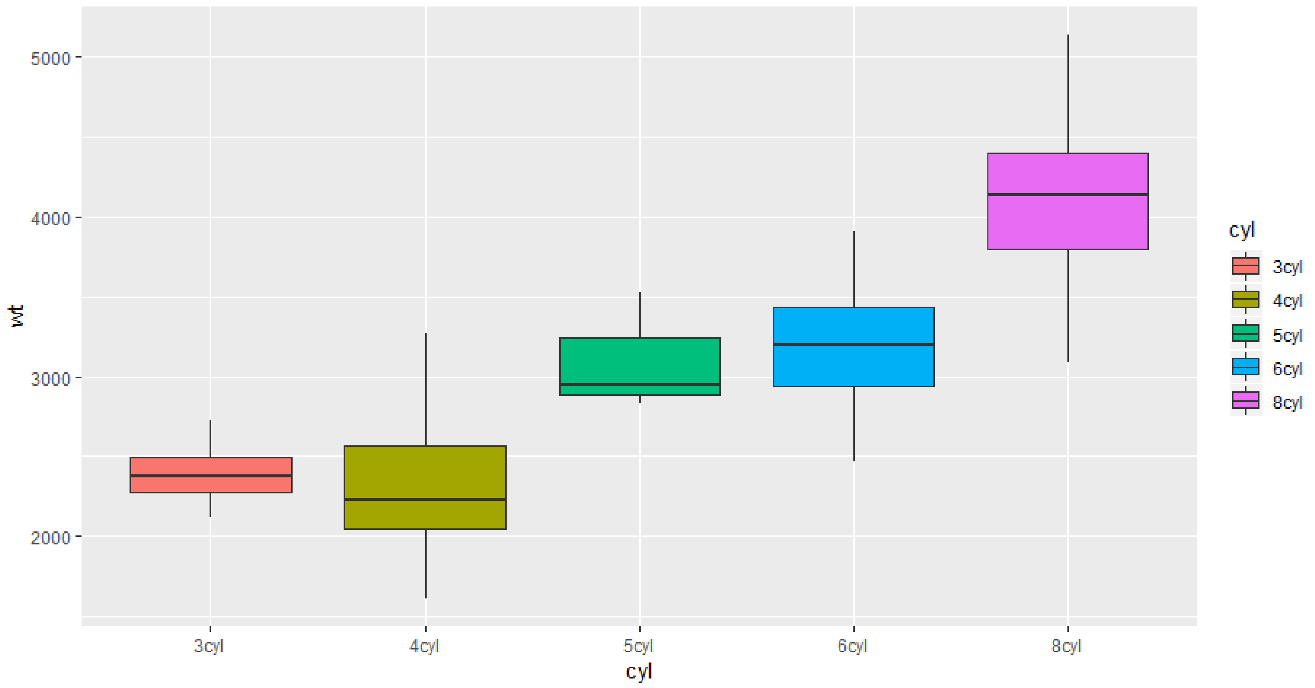
12. We can also plot smooth lines for different classes if any
    a. ggplot(data=autodf)+ geom_smooth(mapping= aes(x = wt, y= mpg, linetype=origin))





Exploring Data – Key activities – ggplot

13. Boxplots help in comparing distribution of data across different attributes and values. For e.g. how weight of automobile changes with number of cylinders…
    a. ggplot(data=autodf, aes(x=cyl, y=wt, fill=cyl)) + geom_boxplot()



14. The plot clearly shows that as cylinders increase, weight of the car increases

Preparing Data For Visualization Using dplyr

Exploring Data – Key activities – dplyr

1. To create visual plots we need the data in proper shape i.e. in form of rows and columns
2. Often the data may need some preparation before it can be used. For e.g. create new variables, create summaries, re-order the columns or rename the columns

3. Dplyr package helps us prepare the data for visualization

4. Basic operations in dplyr include
   a. filter() – to pick certain observations based on value
   b. arrange() – to reorder rows
   c. select() – to pick variables by names
   d. mutate() - create new variables
   e. summarize() – collapse many observations into one

head(flights)

5. All the basic functions above can be used along with group_by() operation which changes the scope of the functions from entire dataset to groups

Exploring Data – Key activities – dplyr

1. They dplyr operations provide verbs for data manipulations and work similarly –
    a. First argument is the data set
    b. Subsequent arguments describe what needs to be done
    c. The result is a new data frame
2. To explore dplr –
    a. install package nycflights13
    b. library(nycflights13)
    c. library(dplyr)
    d. head(flights)

3. filter() - selects few of the rows based on conditions provided
    a. jan_flights <- filter(flights, month==1, day==1)
    b. R provides comparison operators such as >, <, >=, <= , != , ==
    c. When using "==" make sure you are not using "=" and when comparing float with integer for e.g. sqrt(2)^2 == 2 will return false
    d. We can supply multiple arguments to filter using logical operators such as "And" , "Or"
        a. filter(flights, month==1 | month==12)
        b. filter(flights, month %in% c(1,12))

Exploring Data – Key activities – dplyr

4. The NA in various columns can impact many operations and result in NA. filter() will return only those rows which do not have NA in the filter column. If we wish the rows with “NA” to appear in the result
    a. filter(flights, is.na(month) | month ==1)

5. To arrange rows in any particular order of values of any columns
    a. arrange(flights, year, month, day) # by default the arrangement is ascending
    b. arrange(flights, desc(month), year, day) # to arrange on descending order one column

6. To select a few columns of choice
    a. select(flights, year, month, day)   # allows one to zoom into the required columns
    b. select(flights, year:day) # select all columns from year upto day
    c. select(flights, -(year:day))   # select all columns except columns year to day
    d. select(flights, starts_with("yea"))   # to select columns whose column names start with
    e. select(flights, ends_with("ear")) # column names end with
    f. select(flights, contains("yea"))

7. To add new variables into the dataframe / tibble

Exploring Data – Key activities – dplyr

7. To add new variables into the dataframe / tibble
   a. flight_sml <- select(flights, year:day, ends_with("delay"), distance, air_time)
   b. mutate(flight_sml, gain = arr_delay - dep_delay, speed = distance / arr_time *60)
   c. transmute(flight_sml, gain = arr_delay - dep_delay, speed = distance / arr_time *60) # if you want to keep only new columns

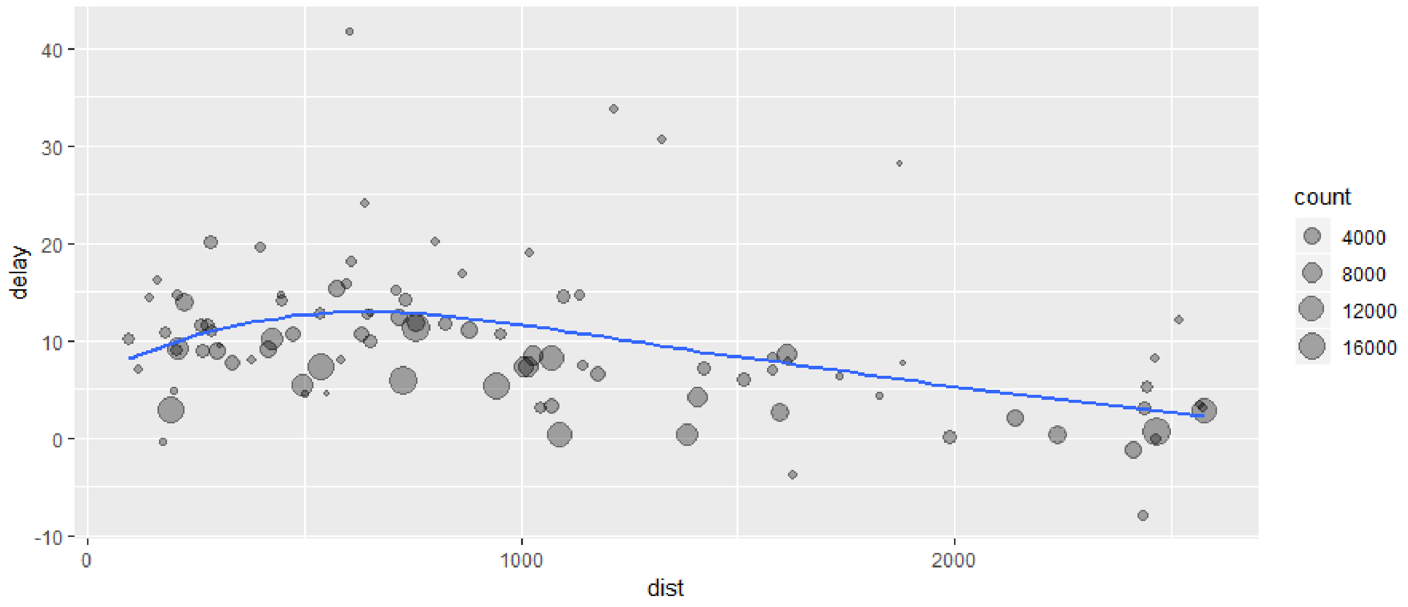
8. To collapse a dataframe into a single row i.e. summarize all the values-
   a. summarize(flights, delay= mean(dep_delay, na.rm=TRUE))
   b. Summarize function becomes useful when we pair it with groupby command

9. To group records –
   a. by_day <- group_by(flights, year, month, day)

10. To combine both summarize and group by – To get mean flight delay by day, month and year i.e. for each day of each month of each year –
    a. by_day <- group_by(flights, year, month, day)
    b. summarize(by_day, delay=mean(dep_delay, na.rm=TRUE))

Exploring Data – Key activities – dplyr

7. Let us analyze the flight data to check whether delays is dependent on the distance flight has to cover….
   a. by_dest <- group_by(flights, dest)   # group by destinations
   b. delay <- summarize(by_dest, count= n(), dist = mean(distance, na.rm=TRUE),
   c. delay= mean(arr_delay, na.rm=TRUE))
   d. delay <- filter(delay, count > 20, dest !="HNL")
   e. ggplot(data = delay, mapping = aes(x = dist, y=delay)) + geom_point(aes(size=count), alpha=1/3) + geom_smooth(se = FALSE)



Handling outliers using MICE

Exploring Data – Key activities – Missing values using MICE

1. data <- airquality # a sample data that comes with the R package

2. data[4:10, 3] <- rep(NA, 7)   # introducing NA in rows 4 to 10 column 3

3. data[1:5, 4] <- NA   # introducing NA in rows 1 to 5 column 4

4. head(data)   # check the data for NA

5. summary(data)   # will show the basic statistics with NAs

6. pMiss <- function(x){sum(is.na(x)) / length(x) * 100} #creating a function and aliasing. This function will give percent of data missing in a column

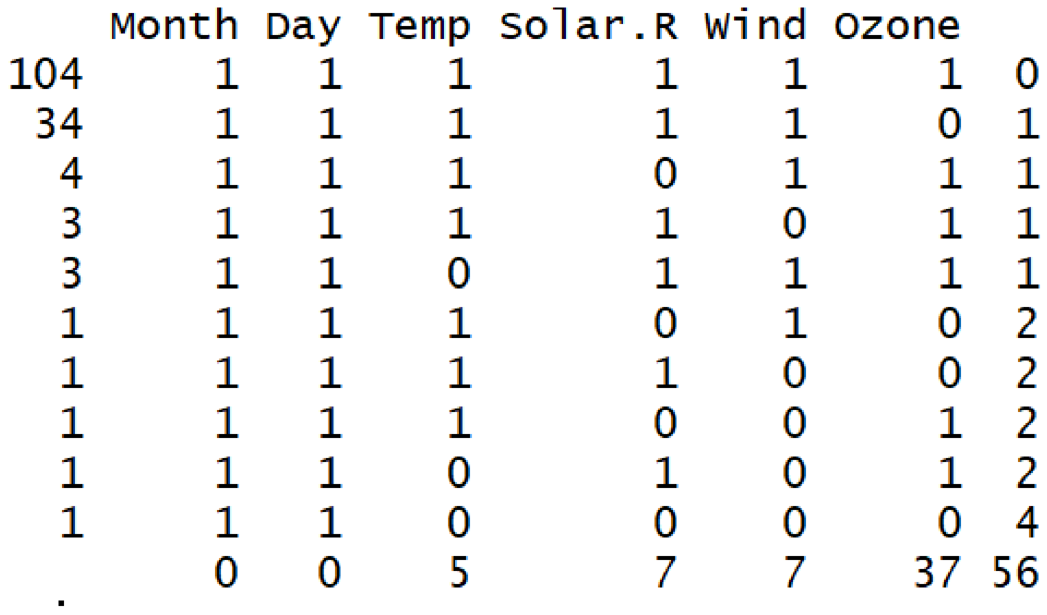
7. apply(data, 2, pMiss) # apply the function pmiss columnwise

Exploring Data – Key activities – Missing values using MICE

8. install.packages("mice")   # Multivariate Imputation by Chained Equations

9. library(mice)

10. md.pattern(data) # missing data pattern # visual matrix view of missing value pattern



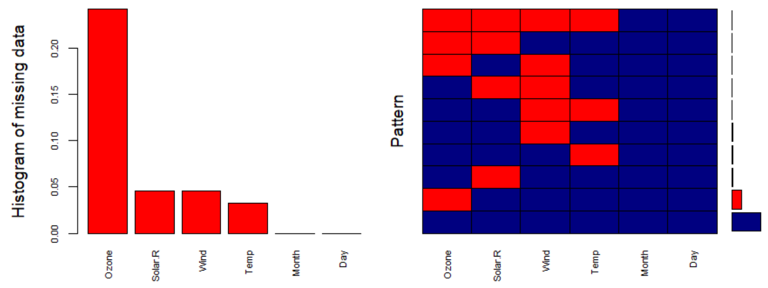
```
    Month Day Temp Solar.R Wind Ozone
104     1   1    1       1    1     1  0
 34     1   1    1       1    1     0  1
  4     1   1    1       0    1     1  1
  3     1   1    1       1    0     1  1
  3     1   1    0       1    1     1  1
  1     1   1    1       0    1     0  2
  1     1   1    1       1    0     0  2
  1     1   1    1       0    0     1  2
  1     1   1    0       1    0     1  2
  1     1   1    0       0    0     0  4
        0   0    5       7    7    37 56
```

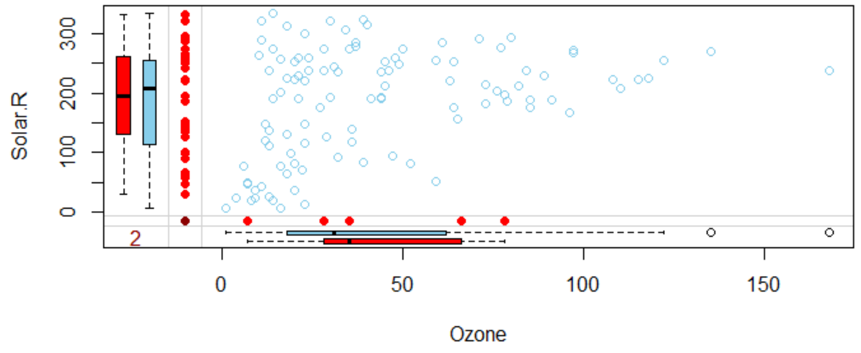
Exploring Data – Key activities – Missing values using MICE

11. install.packages("VIM")

12. library(VIM)

13. aggr_plot <- aggr(data, col=c('navyblue','red'), numbers=TRUE, sortVars=TRUE, labels=names(data), cex.axis=.7, gap=3, ylab=c("Histogram of missing data","Pattern"))
14. marginplot(data[c(1,2)])

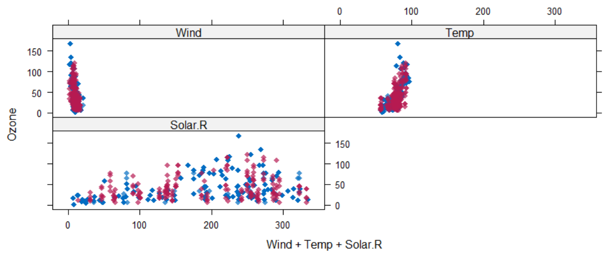


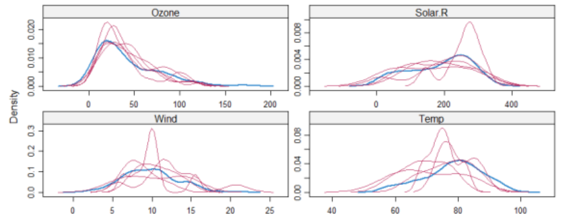


```
tempData <- mice(data,m=5,maxit=50,method='pmm',seed=500)
Summary(tempData)

# Get imputed data (for the Ozone variable)
tempData$imp$Ozone

# Possible imputation models provided by mice() are
methods(mice)

# What imputation method did we use?
tempData$meth

# Get completed datasets (observed and imputed)
completedData <- complete(tempData,1)
summary(completedData)
```

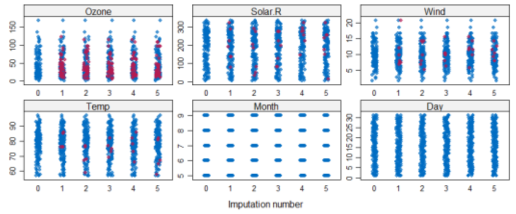
Exploring Data – Key activities – Missing values using MICE

19. xyplot(tempData,Ozone ~ Wind+Temp+Solar.R,pch=18,cex=1)





20. densityplot(tempData)



21. stripplot(tempData, pch = 20, cex = 1.2)

To be discussed later after learning to model

22. modelFit1 <- with(tempData,lm(Temp~ Ozone+Solar.R+Wind)) # imputing values with different models

23. summary(pool(modelFit1))

Exploring Data – Key activities – Missing values using MICE

19. xyplot(tempData,Ozone ~ Wind+Temp+Solar.R,pch=18,cex=1)

20. densityplot(tempData)

21. stripplot(tempData, pch = 20, cex = 1.2)

22. modelFit1 <- with(tempData,lm(Temp~ Ozone+Solar.R+Wind))
23. summary(pool(modelFit1))

Tidy data for machine learning …

1. It is often said that 80% of data analysis is spent on the process of cleaning and preparing the data (Dasu and Johnson 2003).

2. Data preparation is not just a first step, but must be repeated many over the course of analysis as new problems come to light or new data is collected.

3. Part of the challenge is the breadth of activities it encompasses: from outlier checking, to date parsing, to missing value imputation

4. Ref: http://vita.had.co.nz/papers/tidy-data.pdf

   Adobe Acrobat Document

5. Ref: https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

   Microsoft Office Word Document

6. https://statsguys.wordpress.com/2014/01/03/first-post/ - Data cleaning

7. http://datatechblog.com/2015/08/data-analysis-using-r-gathering-organizing-munging/

R Libraries / Packages

R Lib/Packages	Application		R Lib/Packages	Application
devtools	Used for package development		Neuralnet	Package for implementing ANN
dpylr	provides a set of tools for efficiently manipulating datasets in R		Kernlab	Kernel based machine learning for SVM
ggplot	Graph Grammar for plots in R		arules	Libraries for association rules mining
httr	Tools for working with URL and HTTP		Gmodels	Package for cross table analysis
Rodbc	To connect to RDBMS		Snowballc	R interface to C lib for word stemming
tidyr	Used to tidy data. Used in collaboration with dplyr with chain operator		E1071	Support Vector Machine package, NB Classifier, Bagging etc.
Class	Package for linear classifiers		c50	For decision tree model
Yaml	Yet Another Markup Language for data serialization		psych	Multivariate analysis, cluster analysis etc
tm	Text mining library with function to create corpus		Tidyverse	Set of packages for data handling and visualization