Introduction to Data Analysis – Week 1 SIT718

Delaram Pahlevani

Unit Outcome

This unit gives you an introduction to Data Analysis. In this first course you will need to become familiar with:

- ✓key ideas, laws and ethics related to data analysis
- ✓ how to begin installation and basic programming in R and RStudio.

As you progress through to courses two and three, you will develop a grounding in the processes and knowledge related to data summarisation, aggregation and modelling.

Course four is about optimisation and how to implement solutions based on scientific methods such as linear programming.

The Program will finish in course five with a glimpse into the exciting world of Game theory and the concept of zero sum game.

Big Data Overview

Data is created constantly at an ever-increasing rate. Mobile phones, social media, imaging technologies, all of these and more create new data that can be analysed. Three attributes stand out as defining big data characteristics:

- ✓ **huge volume of data**: big data can be billions of rows and millions of columns.
- ✓ **complexity of data types and structures**: big data reflects the variety of new data sources, formats and structures, including digital traces being left on the web and other digital repositories for subsequent analysis.
- ✓ **speed of new data creation and growth**: big data can describe high velocity data with rapid data ingestion and near real time analysis.

Data Structures

1- Structured data

Date	Order	Amount	Price $
12.09	3314	4	210
15.09	3315	2	105
19.09	3316	8	420
21.09	3317	2	105
22.09	3318	4	210

2- Semi-structured data

textual data files with a discernible pattern that enables parsing e.g. XML (extensible markup language), Twitter feeds, HTML

Data Structures - Continued

3- quasi-structured data

textual data with erratic data formats that can be formatted with effort, tools and time e.g. Google search results, web clickstream data

4- Unstructured data

data that has no inherent structure e.g. PDFs, images, videos

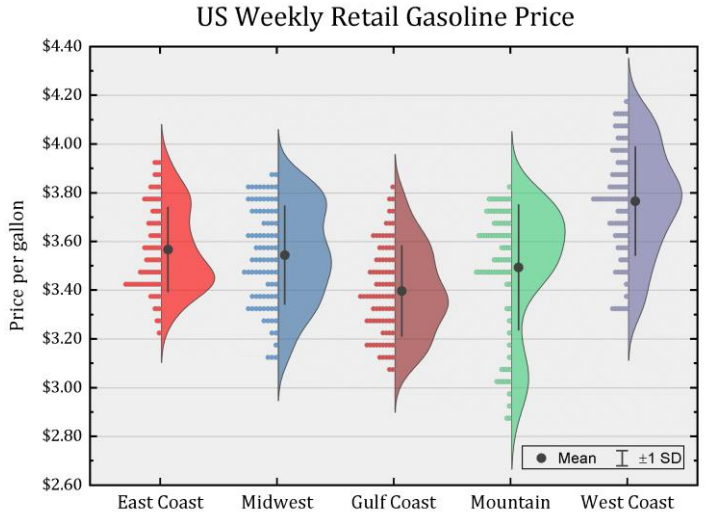
Visualising Data

Charts or Tables?

If we want to attract the attention of a more general users to the data and to highlight its characteristics, the histogram is more suitable.

Graphs

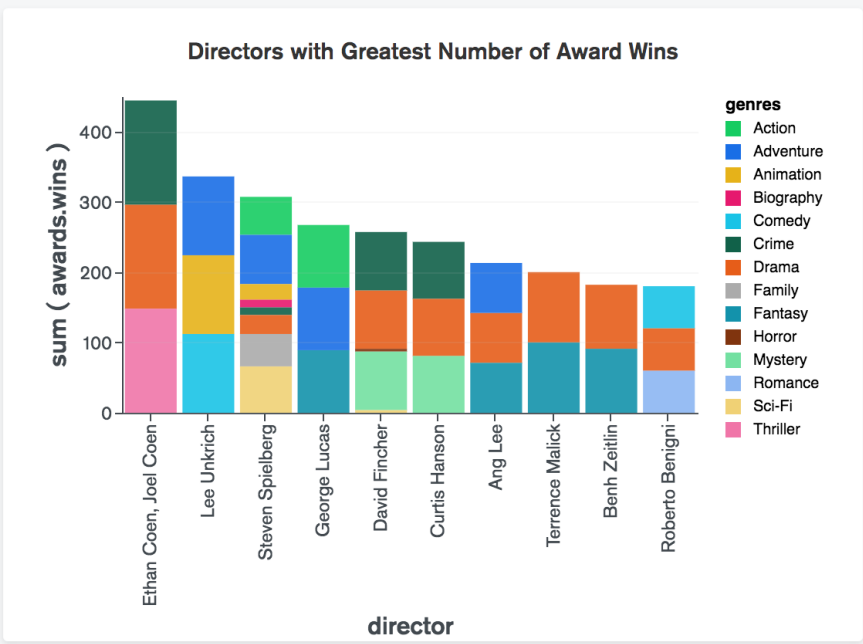
Graphs, charts, isobars, and other forms of visual representation of data are extremely valuable for showing patterns over time which a person would have difficulty in tracing from a table of numbers.

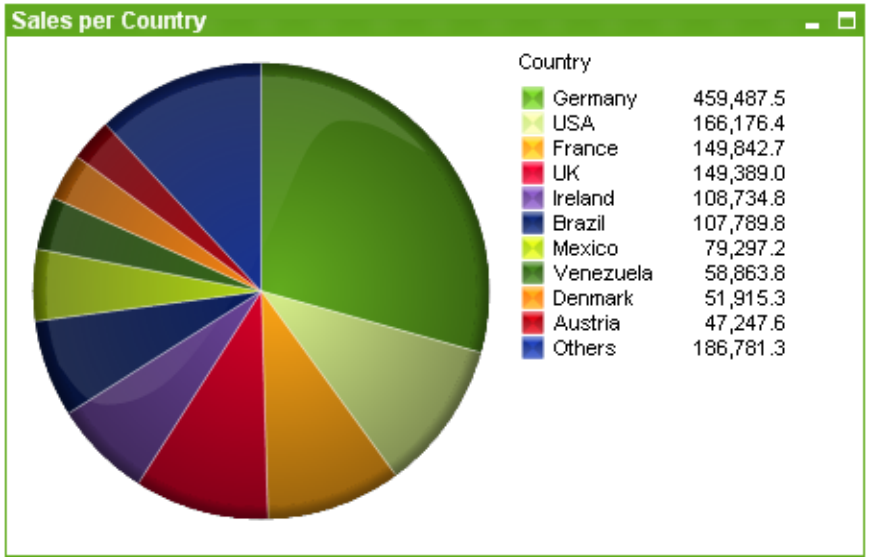


Visualising Data

Bar Charts

Bar charts allow comparisons to be shown between different groups, such as income of workers in different regions, or taste of age groups for fast food. They can be used like graphs to monitor changes over time.

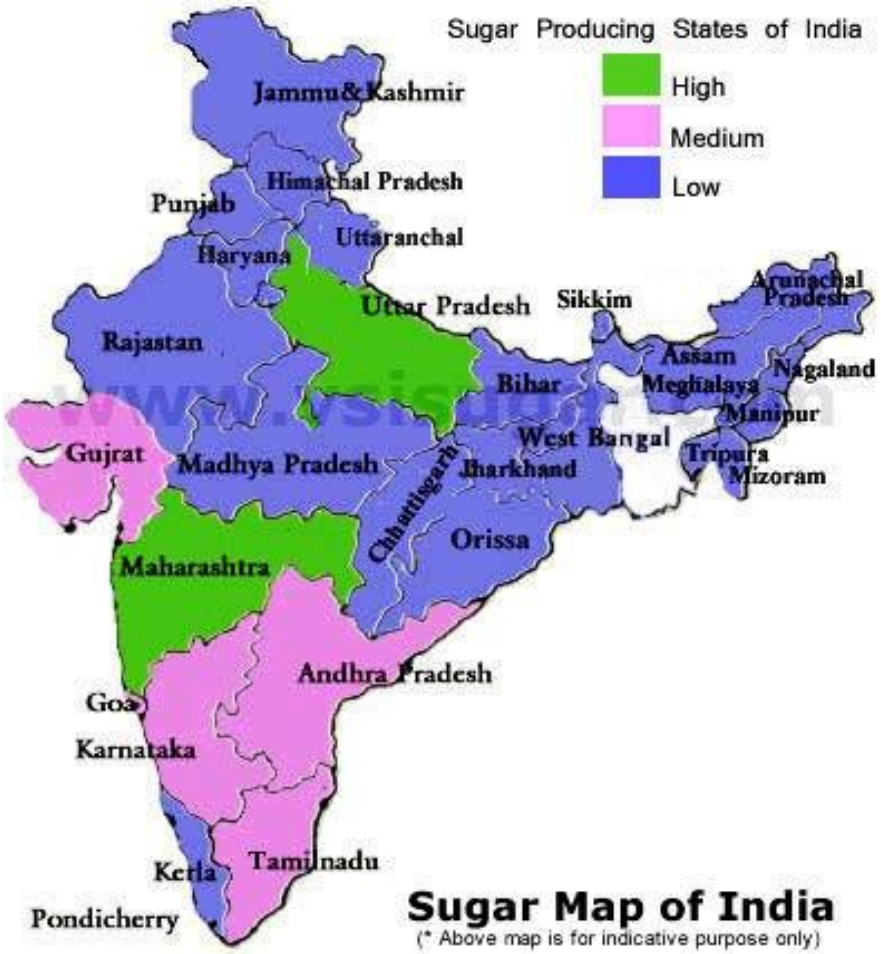




Visualising Data

Pie charts:

Pie charts are an alternative to bar charts for showing the composition of a total. Two or three pie charts can be placed alongside each other to illustrate the changing composition.



Visualising Data

Maps:

Heat maps, colour maps, or statistical maps, are used to represent changes in say, the temperature over a geographical area, or variation of properties (for example speed of athletes) in different places of the football field, or in general a variation of property over a geographical area.

General Principles for Representing Tables

- ✓Tables should be **titled** and, if more than one, numbered for easy reference.
- ✓Titles and column and row headings should be kept brief. Footnotes or text in the main body of the document may be used to elaborate on definitions, highlight unusual observations, etc.
- ✓The full source of the data should always be given - usually at the bottom of the table, so that users can trace the original data.
- ✓Capital letters are usually only used for the initial letter of the first word in a row or column heading.
- ✓Precision of numbers should be consistent, i.e how many decimal places. As a general rule, round numbers to two effective units unless you have reason to believe greater accuracy is required. Two effective units would be to round 161, 192, 204, to 160, 190, 200 respectively.
- ✓Give row and column totals or averages or any summary calculations that you believe the user would be interested in.
- ✓Put the numbers to be most often compared next to each other in columns rather than rows - the eye reads down columns more easily than across rows (Wright and Fox 1970).
- ✓In the text, give a summary in words of the main points in the table.
- ✓In general, do not use tables to show broad trends or relationships. Charts are much better.

RStudio

R is the statistical programming language most commonly used by professional data scientists, statisticians and data analysts.

How to download R?

Windows users:

1- Open an internet browser and go to www.r-project.org.

2- Click the "download R" link in the middle of the page under "Getting Started."

3- Select a CRAN location (a mirror site) and click the corresponding link.

4- Click on the "Download R for Windows" link at the top of the page.

5- Click on the "install R for the first time" link at the top of the page.

6- Click "Download R for Windows" and save the executable file somewhere on your computer. Run the .exe file and follow the installation instructions.

7- Now that R is installed, you need to download and install RStudio.

RStudio

Now it’s time to install RStudio

Windows users:

- Go to www.rstudio.com and click on the "Download RStudio" button.
- Click on "Download RStudio Desktop."
- Click on the version recommended for your system, or the latest Windows version, and save the executable file. Run the .exe file and follow the installation instructions.

Let’s get started with RStudio

Basic Mathematical Operations

Basic arithmetic operations are present in R and are the same as what you may have used previously in programs such as Excel.

Addition: 5+2 represents 5+2

Subtraction: 6-8 represents 6−8

Multiplication: 3*4 represents 3×4

Division: 3/4 represents 3÷4

Powers: 3^4 represents 81

Your task:
Implement all mathematical operations on RStudio.

Assignment of Variables using RStudio

To assign a value or data to a variable, we use the equals sign =. In older R tutorials you may also see the expression <- used, which performs the same action. After we've assigned a value to a variable, we can use that variable in further calculations.

Your task:
Try entering the following into the console:

```
a = 3
b = 7
the.value = 12
the.index = 2
the.index_2 = 3
```

Now evaluate the following commands:

```
a*b
the.value * a
a^the.index + b^the.index_2
```

Vectors and Arrays:

Vectors and arrays are terms used to describe a set of numbers in some particular arrangement e.g. [1,5,2,8]

Using RStudio: c(3,2,1,0)

Your task:
Try entering the following commands into the R console:

```
4:8
array(0,3)
array(1,10)
array(2,200)
array(c(1,2,3), 10)
array(c(4,5,6), 12)
array(0, c(3,4))
```

Manipulating Vectors & Arrays

We can replace one or multiple values in an array. After each following command, type a in the console and press enter to see how the array a changes after each command

Your task:
Try entering the following commands into the R console:

```
a = array(0,20)
a[5] = 1
a[c(3,7,11)] = c(2,6,1)
a[17:20] = c(1,2,1,4)
```

Final expected output:

```
0 0 2 0 1 0 6 0 0 0 1 0 0 0 0 0 1 2 1 4
```

Manipulating Vectors & Arrays - Continued

We can combine two or more vectors into a matrix using cbind (column bind) or rbind (row bind). We'll want to use these when we want to combine rows or columns or data into a separate matrix.

Your task:
Run the following in the console and see how the vectors are combined. Try playing around with different size vectors and see what happens.

```
cbind(c(1,2,3,7,9), c(21, 2, 1, 5, 6))
cbind(c(1,2,3,7,9), c(1,9,7,2,1), array(6,5))
rbind(c(3,6,1,92), c(10,3,1))
rbind(c(3,6,1,92), c(3,2,1,8))
rbind(c(3,6,1,92), c(4,1,12,1,2))
```

Basic Operations on Vectors

Example 1:

c(1,2,3)+4 : 5 6 7

c(1,2,3)*3: 3 6 9

c(1,2,3)^2: 1 4 9

Example 2:

v1 = c(1,6,7,9)

v2 = c(-1,2,1,-2)

Now check the following commands against the expected output:
v1+v2 should give you 0 8 8 7
v1-v2 should give you 2 4 6 11
v1*v2 should give you −1 12 7 −18
v1/v2 should give you −1 3 7 −4.5
v1^v2 should give you 1 36 7 0.01234568

Existing Functions:

Assign the following vectors:

a = c(1,8,3,9)

b = c(2,2,1,1)

d = c(3,4,6,81,9)

Now check the following functions:

- ❑ sum(a) should give you 21
- ❑ prod(d) should give you 52488
- ❑ length(b) should give you 4
- ❑ sum(a*b) should give you 30
- ❑ sum(a^b) should give you 77
- ❑ min(d) should give you 3
- ❑ max(a,d) should give you 81
- ❑ prod(a)*(1/length(b)) should give you 54
- ❑ min(max(a), max(b)) should give you 2

Creating new functions

Creation of a basic function essentially consists of 3 components:

Predefining the function inputs

A sequence of calculations

Return of an output

Example 1:

```
our.mean <- function(x) {
sum(x)/length(x)
}
x= c(1,7,3,4)
our.mean(x)
```

Example 2:

```
our.mean.2 <- function(x) {
n <- length(x)
s <- sum(x)
output <- s/n
output
}
```

Geometric and Harmonic mean

```
GM
GM <- function(x) {
prod(x)^(1/length(x))
}
x= c(1,7,3,4)
GM(x)
```

```
HM
HM <- function(x) {
length(x)/sum(1/x)
}
```