Introduction to Data Science and R

Hamid Abdulsalam

Plans for Today

1. **Introduction**
2. **What is Data Science?**
3. **Why Study Data Science?**
4. **Why R ?**
5. **Getting Started with R**

Introduction

About the Instructor

- Lead Data Scientist @ Factual Analytics
- Data Science Facilitator @ Semicolon Africa
- Co-Founder Lagos R User Group
- B.Sc in Statistics (UNN), M.Sc Statistics (ABU)
- Statistical Tools includes R, Sas ,SPSS and Stata
- Over 400+ data analytics projects done.

Contact:

- hamid@factual.ng

About the Participants

- Introduce yourselves?
- What are your expectations from this Course?
- Have you ever used R or Rstudio before?

Some Data Science Applications

- Banks wants to Know if a customer will default on loan or not
- Cancer status of a patient
- Telecommunication company wants to know if a subscriber will churn or out
- Through datascience airlines are able to offer different prices per person, per moment, per location.
- Uber, Taxify , Snapchat, facebook are all driven by data science
- Loan applications such as Paylater, Branch, QuickMoney are data science driven

What is Data Science?

Data science involves the extraction and analysis of vital data from different sources with the sole aim of bring insights from the data. Most businesses these days rely heavily on data science. The future is all about Artificial Intelligience and machine learning.

What do data scientist do?

A data scientist's main objective is to organize and analyze large amounts of data, often using software specifically designed for the task. The final results of a data scientist's analysis needs to be easy enough for all invested stakeholders to understand — especially those working outside of IT.

Three Main tasks of a Data Scientist

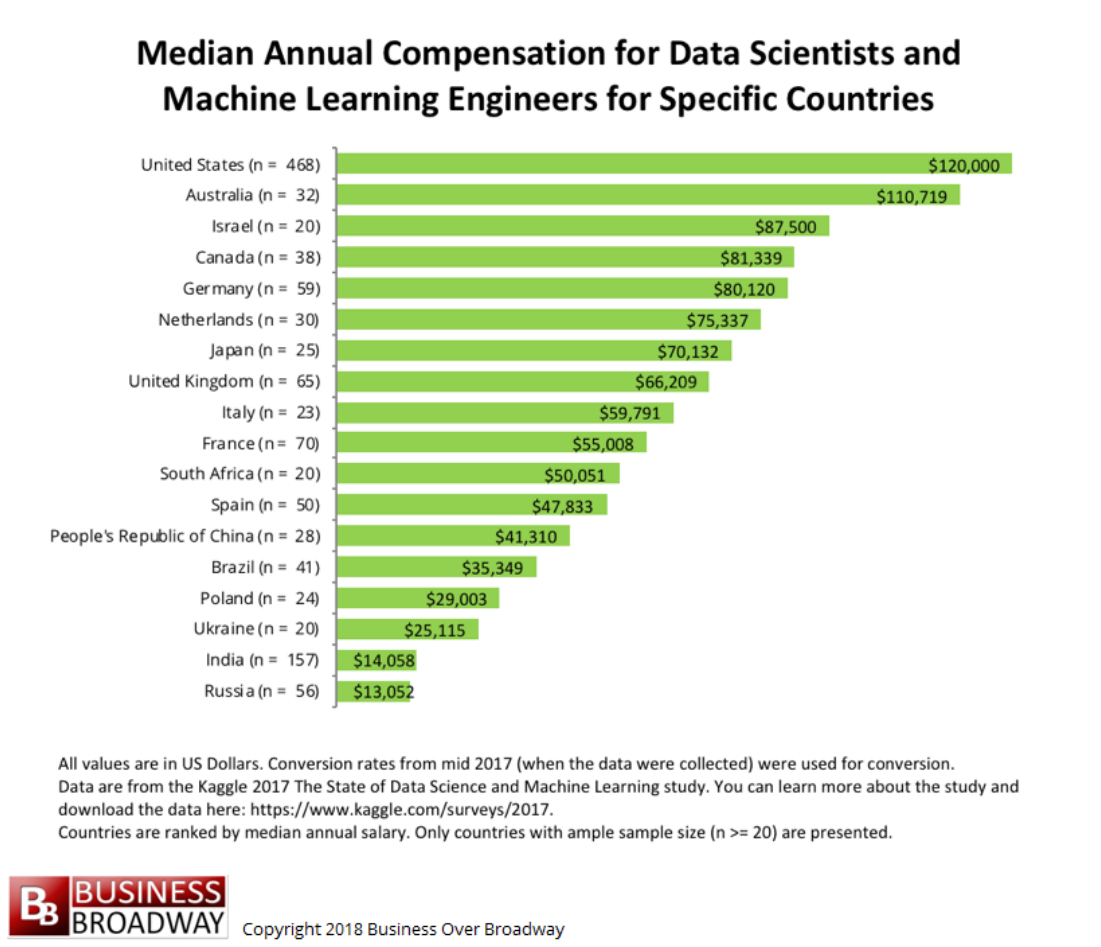
1. Preparing the data for analytics (Data Munging)
    - Gathering, cleaning, transforming, loading, filtering, deleting, merging e.t.c
2. Implementing the model (Statistics)
3. Communicating the results (Presentation of Reports)

Why Study Data Science?

Data scientist has become one of the key careers of the century. What's more, according to the Harvard Business Review, it's the sexiest job of the 21st century.

Job opportunitites

- Data analysis skills are highly valued globally



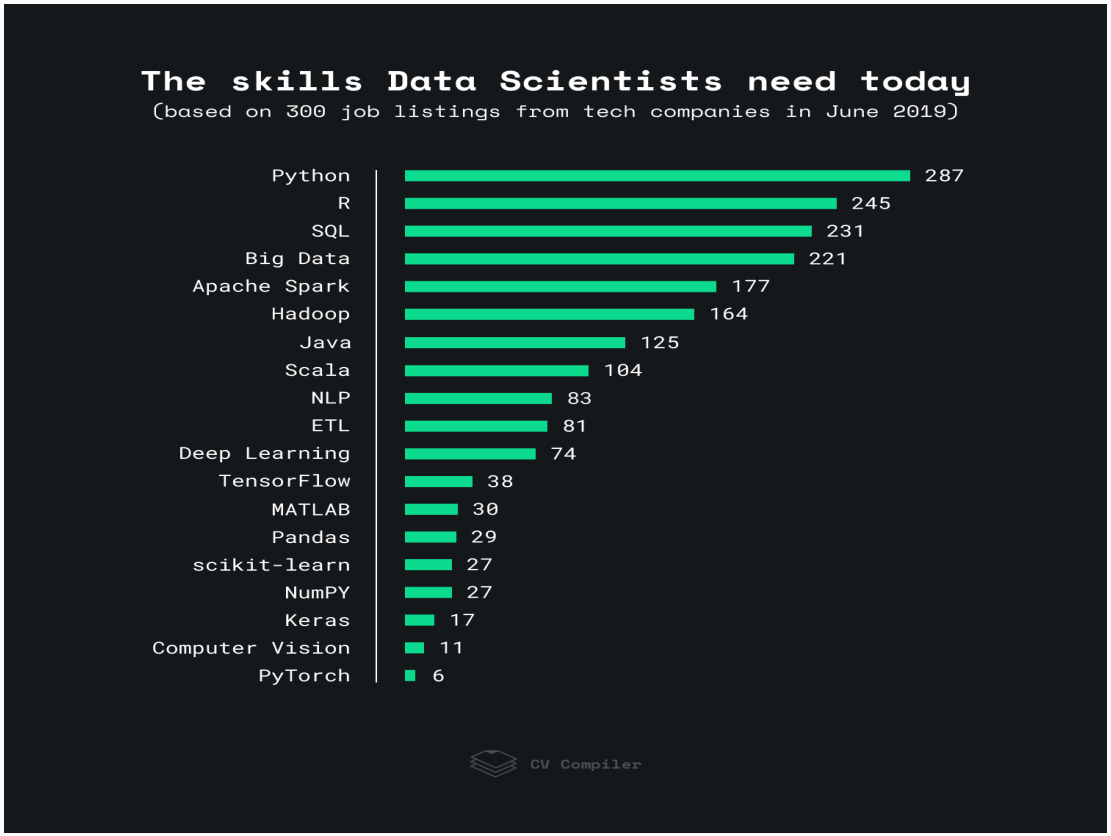
Business Broadway: Median Annual Income of Data Scientists 2018

Why R?

- Open source: Easily downloadable and doesn't come with a price
- Thousands of Packages hosted on CRAN
- Rich online Community
- Number one software for data analytics
- Visualisation (Plots) is one of the best
- You can easily get help online i.e stackoverflow, stackexchange

Why R?

- Top programming languages for data science.



KDNuggets 2019

Getting Started With R

We will get started by going through some of the fundamentals of R.

`help()` function

- From the R "Console" you can use the `help()` function or `?`. For example, try the following two commands (which give the same result):

```
help(mean)
?mean
```

R is case sensitive

`height` and `height` are different:

```
height <- 10
Height <- 50
```

```
> height
[1] 10
> Height
[1] 50
```

Objects in R

There are five basic classes of objects in R:

- character
- numeric (real numbers)
- integer
- complex
- logical (True/False)

The most basic object is a vector

- A vector can only contain objects of the same class e.g (Only Characters or only numbers). An exception to this is a *list* which can contain objects of different classes.

Vectors are created using c() function in R

Numbers in R

- R treats numbers as numeric objects (i.e. double precision real numbers) e.g 2.00, 4.058,5.06 e.t.c
- If you explicitly want an integer, you need to specify the `L` suffix
- Ex: Entering `1` gives you a numeric object; entering `1L` explicitly gives you an integer.
- For instance, a vector X which contains 2L,3L will be treated as Integer(Integers are whole number) while a vector which contains 2,3,4 will be treated as numeric

Attributes

R objects can have attributes

- names, dimnames
- dimensions (e.g. matrices, arrays)
- class
- length

Attributes of an object can be accessed using the `attributes()` function.

Entering Input

The <- symbol is the assignment operator used in R.

```
> x <- 14
> print(x) ## explicit printing
[1] 14
> x        ## auto printing
[1] 14
> msg <- "hello"
```

The [1] indicates that x is a vector and 14 is the first element.

Basic Operations in R

R is built for analytics, Hence you can do basic operations in R.

```
>x<- 2 + 3
>y<- (5*11)/4 - 7
>s<-7^3
> x
[1] 5
> y
[1] 6.75
> s
[1] 343
> x*y
[1] 33.75
```

Sequence in R

```
> x <- 1:20
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
[16] 16 17 18 19 20
>length(x)
[1] 20
```

The : operator is used to create integer sequences.

We can also use the function seq in to generate integer sequences

```
x<-seq(1, 9, by = 2) ##Integers starting from 1 and
                     ##ending at 9 with a difference of 2
[1] 1 3 5 7 9
```

Creating Vectors

The `c()` function can be used to create vectors of objects.

```
> x <- c(1, 4, 6.5, 4.5) ## numeric
> x <- c(TRUE, FALSE)    ## logical
> x <- c(T, F)           ## logical
> x <- c("c", "d", "e")  ## character
> x <- 19:40             ## integer
> x <- c(1+0i, 2+4i)     ## complex
```

Objects Mixing in R

In R, objects that belong to different class can be mixed

```
> y <- c(16, "a")    ## character
> y <- c(FALSE, 2)     ## numeric
> y <- c("a", TRUE)   ## character
```

There is always *coercion* when objects are mixed in R. For instance, object mixed with a character becomes a character class.

Logical objects mixed with a numeric object becomes a numeric object.

Explicit Coercion in R

Objects can be explicitly coerced from one class to another using the `as.*` functions, if available.

```
> x <- 0:6
> class(x)
[1] "integer"
> as.numeric(x)
[1] 0 1 2 3 4 5 6
> as.logical(x)
[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
> as.character(x)
[1] "0" "1" "2" "3" "4" "5" "6"
```

Matrices

Matrices in R are created through the function *matrix* or by combining multiple vectors through rows or columns.

```
> x <- matrix(nrow = 2, ncol = 3)
> x
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
> dim(x)
[1] 2 3
> attributes(x)
$dim
[1] 2 3
```

Matrices (cont'd)

Matrices can be constructed by *column-wise* or by *row-wise*. By default, R construct matrices by Column-wise, but if you want the matrix to be constructed by row-wise, you need to inform R.

```
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

> m <- matrix(1:6, nrow = 2, ncol = 3, byrow=TRUE)
> m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Matrices (cont'd)

We can also create matrix through vector by adding the dimension attribute.

```
> x <- c(1,2,3,4,5,6,7,8,9,10)
> x
[1]  1  2  3  4  5  6  7  8  9 10
> dim(x) <- c(2, 5)
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

cbind and rbind

We can also create matrices in R through *column-binding* or *row-binding* with `cbind()` and `rbind()`.

```
> x <- 4:6
> y <- 1:3
> cbind(x, y)
      x y
[1,] 4 1
[2,] 5 2
[3,] 6 3
> rbind(x, y)
  [,1] [,2] [,3]
x    4    5    6
y    1    2    3
```

Naming Matrices

we can name the rows and columns of matrices in R using `colnmames` or `rownames`.

```
> m <- matrix(1:4, nrow = 2, ncol = 2)
> colnames(m) <- c("a", "b")
>rownames(m) <- c("c", "d")
> m
  c d
a 1 3
b 2 4
```

Task 1

- Create a vector X containing values 1:4 and another vector Y containing values 5:8.
- Use the cbind function to turn it to matrix called M.
- Also use the rbind function to turn it to another matrix call Q.
- Use the colnames and rownames function to name the matrices.

Lists

List can be regarded as vector but a special type of vector that can contain elements of different classes.We create a list in R by using the function `list`

```
> m <- list(50, "b", TRUE, 1 + 4i)
> m
[[1]]
[1] 50
[[2]]
[1] "b"
[[3]]
[1] TRUE
[[4]]
[1] 1+4i
```

Names

Lists can also have names.

```
> x <- list(a = 1, b = "c", c = 3)
> x
$a
[1] 1

$b
[1] "c"

$c
[1] 3
```

Factors

Most times, as a data scientist. You will come across data that contains categorical data. R represent categorical data as Factors. They can be unordered or ordered.

In modelling, factors are treated specially by modelling functions like `lm()` and `glm()`

- As a data scientist, it is always good to use factors with labels than using integers because factors are self-describing; having a variable that has values "Male" and "Female" is better than a variable that has values 1 and 2.

Factors

We create factors in R using the function `factor`

```
> x <- factor(c("yes", "yes", "no", "yes", "no"))
> x
[1] yes yes no yes no
Levels: no yes
> table(x)
x
no yes
 2   3
> class(x)
[1] "factor"
```

Missing Values

R denote Missing values by `NA`

It also denote `NaN` for undefined mathematical operations.

- `is.na()` is used to test objects if they are `NA`
- `is.nan()` is used to test for `NaN`

Missing Values

```
> x <- c(1, 2, NA, 10, 3)
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE
> is.nan(x)
[1] FALSE FALSE FALSE FALSE FALSE
> x <- c(1, 2, NaN, NA, 4)
> is.na(x)
[1] FALSE FALSE  TRUE  TRUE FALSE
> is.nan(x)
[1] FALSE FALSE  TRUE FALSE FALSE
```

Data Frames

Data frames are used to store tabular data

- Unlike matrices, data frames can store different classes of objects in each column (just like lists); matrices must have every element be the same class
- Data frames also have a special attribute called `row.names`
- Data frames are usually created by calling `read.table()` or `read.csv()`
- Can be converted to a matrix by calling `data.matrix()`

Data Frames

We create data frame in R using function `data.frame` in R

```
> x <- data.frame(num = 1:4, log = c(T, T, F, F))
> x
  num   log
1   1  TRUE
2   2  TRUE
3   3 FALSE
4   4 FALSE
> nrow(x)
[1] 4
> ncol(x)
[1] 2
```

Names

Vector objects can also have names.

```
> x <- 5:7
> names(x)
NULL
> names(x) <- c("five", "six", "seven")
> x
five six seven
   1   2     3
> names(x)
[1] "five"  "six"  "seven"
```

Subsetting

Subsetting is an important component of R. It allows you to be able to extract any element from a dataframe.

There are a number of operators that can be used to extract subsets of R objects. - `[` can be used to select one or more than one element in a vector, list, matrix or dataframe

- `[[` is used to extract elements of a list or a data frame; it can only be used to extract a single element and the class of the returned object will not necessarily be a list or data frame
- `$` is used to extract elements of a list or data frame by name; semantics are similar to that of `[[`.

Subsetting

```
> x <- c("a", "b", "c", "d")
> x[1]
[1] "a"
> x[2]
[1] "b"
> x[1:4]
[1] "a" "b" "c" "d"
> x[x > "b"]
[1] "c" "d"
```

Subsetting a Matrix

Matrices in R are usually subsetted using the indices type (i,j)

```
> x <- matrix(1:6, 2, 3)
> x[1, 2]
[1] 3
> x[2, 1]
[1] 2
```

Indices can also be missing.

```
> x[1, ]
[1] 1 3 5
> x[, 2]
[1] 3 4
```

Subsetting a Matrix

By default, when a single element of a matrix is retrieved, it is returned as a vector of length 1 rather than a 1×1 matrix. This behavior can be turned off by setting `drop = FALSE`.

```
> x <- matrix(1:6, 2, 3)
> x[1, 2]
[1] 3
> x[1, 2, drop = FALSE]
     [,1]
[1,] 3

> x[1, , drop = FALSE]
     [,1] [,2] [,3]
[1,]    1    3    5
```

Subsetting a Matrix

We can also subset a range of rows and columns from a matrix

```
>  x <- matrix(1:12, 3, 4)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> x[2:3,2:4]
     [,1] [,2] [,3]
[1,]    5    8   11
[2,]    6    9   12
```

Task 2

- Create a matrix Y with 4 rows and 4 columns.
- Fill it with integers starting from number 1 to 16
- Assign any colnames.
- Extract 5th element
- Extract column range 3:4 and row range 1:3

The End of Session 1

Any Questions ?