Unit-1

Defining Data Science and Big data, Benefits and Uses, facets of Data, Data Science Process.

Historyand Overview of R, Getting Started with R, R Nuts and Bolts

short answer questions

1. Which values are used to represent missing values and which functions are used to test for presence of missing values.
2. What are the 4 V’s of Big Data?
3. Explain about explicit coercion
4. Explain about naming objects in R

Long answer questions

1. What are the different facets of data
2. What are the different data structures in R? Explain
3. Explain about built in datatypes in R and R attributes

### **Q) History of R**

**What is R?** R is a computer language used for statistics and making graphs.

**Where Did R Come From?**

* **S Language**: In 1976, a language called S was created at a company called Bell Labs. It was used for analyzing data.
* **Updated S**: In 1988, S was updated and rewritten in a programming language called C.
* **Commercial Version**: S was sold as a product named S-PLUS, which had extra features like graphical interfaces.

**Creation of R:**

* **1991**: Two people named Ross Ihaka and Robert Gentleman created R in New Zealand as a free alternative to S-PLUS.
* **1993**: R was shared with the public for the first time.
* **1995**: R became free software, meaning anyone could use and change it.
* **2000**: The first official version of R (version 1.0.0) was released.

**Q) Why is R Special?**

* **Similar to S**: R looks a lot like S, so people who used S-PLUS could easily learn R.
* **Open Source**: R works on almost any computer, and its code is open for everyone to see and change.
* **Great Graphics**: R is very good at making high-quality graphs and charts.
* **Interactive and Programmable**: You can use R for simple data analysis or for writing more complex programs.

**Community and Development:**

* **Active Community**: Many people around the world use R and help each other online.
* **Frequent Updates**: R is updated regularly to fix bugs and add new features.

**Free to Use:**

* R is free to use and share. You can study its code, improve it, and share your improvements with others

**Q) Explain about implicit and explicit coercion in R**

**Answer: Mixing Objects:** Sometimes, we mix different types of objects in R

Example 1: **y = c(1.7, "a")**

This mixes a number (1.7) and a character ("a"). The result is a character vector because R converts the number to a string.

Example 2:

y = c(TRUE, 2)

This mixes a logical value (TRUE) and a number (2). The result is a numeric vector because TRUE is converted to 1.

R converts all elements in the vector to the same type. This process is called **implicit coercion**.

**Explicit Coercion:** We can also manually convert objects from one type to another using specific functions:

**x = 0:3 (**here x will contain **0 1 2 3)**

1. Converting to Numeric: **as.numeric(x)**
2. Converting to Logical:  
    as.logical(x) # Result: FALSE TRUE TRUE TRUE
3. Converting to Character:  
    as.character(x) # Result: "0" "1" "2" "3"

**Q) different builtin-data types in R**.

• atomic classes: numeric, logical, character, integer, complex

• vectors, lists

• factors

• missing values

• data frames and matrices

All R objects can have attributes that help to describe what is in the object.

| Data type | Example data |
| --- | --- |
| logical | TRUE or FALSE |
| integer, single or double | Floating point real numbers; 3, o.753, 1.e+200 |
| complex | 3 + 1.23i |
| character | strings in quotes (“) or apostrophes (‘) |

**Q) R Attributes**

R objects can have attributes, which are like metadata for the object. These metadata can be very

useful in that they help to describe the object. For example, column names on a data frame help to

tell us what data are contained in each of the columns. Some examples of R object attributes are

• names, dimnames

• dimensions (e.g. matrices, arrays)

• class (e.g. integer, numeric)

• length

• other user-defined attributes/metadata

**Q) What are the different data structures in R? Explain**

**The different data structures are**

1. Vectors: The c() function can be used to create vectors of objects  
Empty vectors can be created with the vector() function.

A vector can only contain objects of the same class.

Example:

x <- c(0.5, 0.6) ## numeric vector

x <- c(TRUE, FALSE) ## logical vector

2. Matrices

Matrices are vectors with a dimension attribute.   
The dimension attribute is itself an integer vector of length 2 (number of rows, number of columns)

Example:

> m <- matrix(nrow = 2, ncol = 3)

> m

[,1] [,2] [,3]

[1,] NA NA NA

[2,] NA NA NA

> dim(m)

[1] 2 3

3. Lists

Lists are a special type of vector that can contain elements of different classes.

Example 1:

x <- list(1, "a")

print(x)

Output:

[[1]]

[1] 1

[[2]]

[1] "a"

Example: 2

empId = c(1, 2, 3, 4)  
empName = c("Debi", "Sandeep", "Subham", "Shiba")  
numberOfEmp = 4  
empList = list(empId, empName, numberOfEmp)

print(empList)

4. Factors are used to represent categorical data and can be unordered or ordered.They are useful to categorize unique values in columns like “TRUE” or “FALSE”, or “MALE” or “FEMALE”, etc..

> 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

> ## See the underlying representation of factor

> unclass(x)

[1] 2 2 1 2 1

attr(,"levels")

[1] "no" "yes"

5. Dataframes: Dataframes are used to store the tabular data.

Example  
Name = c("Amiya", "Raj", "Asish")

Language = c("R", "Python", "Java")

df = data.frame(Name, Language)

print(df)

Name Language

1 Amiya R

2 Raj Python

3 Asish Java

### **Q) Handling Missing Values in R**

**Understanding Missing Values:**

* **NA**: This represents a missing value in R.
* **NaN**: This stands for "Not a Number" and is used for undefined mathematical operations.

**Testing for Missing Values:**

* Use is.na() to check if elements in an object are NA.
* Use is.nan() to check if elements in an object are NaN.

**Classes of NA:**

* NA values can belong to different classes such as integer, character, etc.
* A NaN value is also considered as NA, but not all NA values are NaN.

**Example 1: Creating a Vector with NA Values:**

x <- c(1, NA)

is.na(x)

Output: FALSE TRUE

is.nan(x)

Output: FALSE FALSE

**Example 2: Creating a Vector with Both NA and NaN Values:**

x <- c(1, NaN, NA)

is.na(x)

Output: FALSE TRUE TRUE

is.nan(x)

Output: FALSE TRUE FALSE

**Q) Explain about naming objects in R**

Use names() to set names for vectors and data frame columns.

x <- 1:2

Assigning names to vector x

names(x) = c("vja", "gnt")

print(x)

Output: vja gnt

1 2

Use names() for lists.

x = list("vja" = 1, gnt = 2)

print(x)

Output:

$`vja`

1

$gnt

2

Use dimnames(), colnames(), and rownames() for matrices.

m <- matrix(1:4, nrow = 2, ncol = 2)

dimnames(m) <- list(c("a", "b"), c("c", "d"))

print(m)

Output:

c d

a 1 3

b 2 4

colnames(m) <- c("h", "f")

rownames(m) <- c("x", "z")

print(m)

Output:

h f

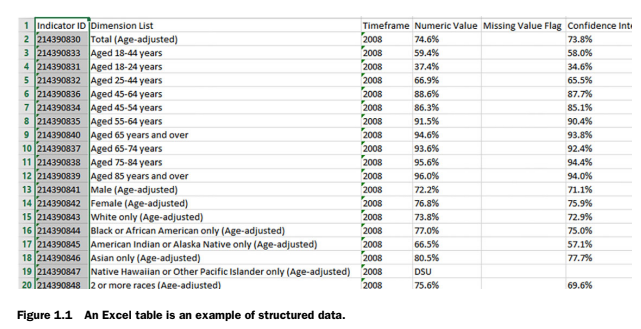
x 1 3

z 2 4

**Q) What are the different facets of data**

### **Structured Data**

Structured data is organized and stored in a fixed way. For example, data in tables or Excel files. It’s easy to manage and query using SQL. Structured data helps in understanding and organizing information easily.





### **Unstructured Data**

Unstructured data doesn’t fit neatly into tables or predefined models. Emails are an example. They have parts like the sender and body, but are difficult to organize and analyze. This type of data needs more effort to make sense of.

### **Natural Language**

Natural language data is a special type of unstructured data. It’s hard to process because it involves understanding languages. An example is a human-written email. Analyzing this data requires advanced techniques and understanding of language.

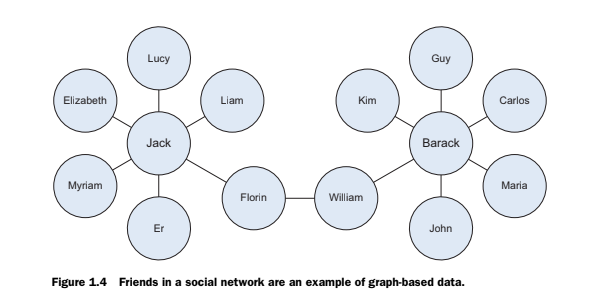
### **Machine-Generated Data**

This data is created automatically by machines without human input. Examples are server logs and sensor data. This type of data is growing fast and needs tools that can handle large volumes quickly.



### **Graph-Based Data**

Graph-based data shows relationships between different objects. Social media connections are a good example. This data is stored in graph databases and helps understand how different entities are connected.



### **Audio, Image, and Video**

These data types include sounds, pictures, and videos. Computers find it hard to recognize and understand objects in these data types. For example, analyzing a video of a sports game to track ball movements.

### **Streaming Data**