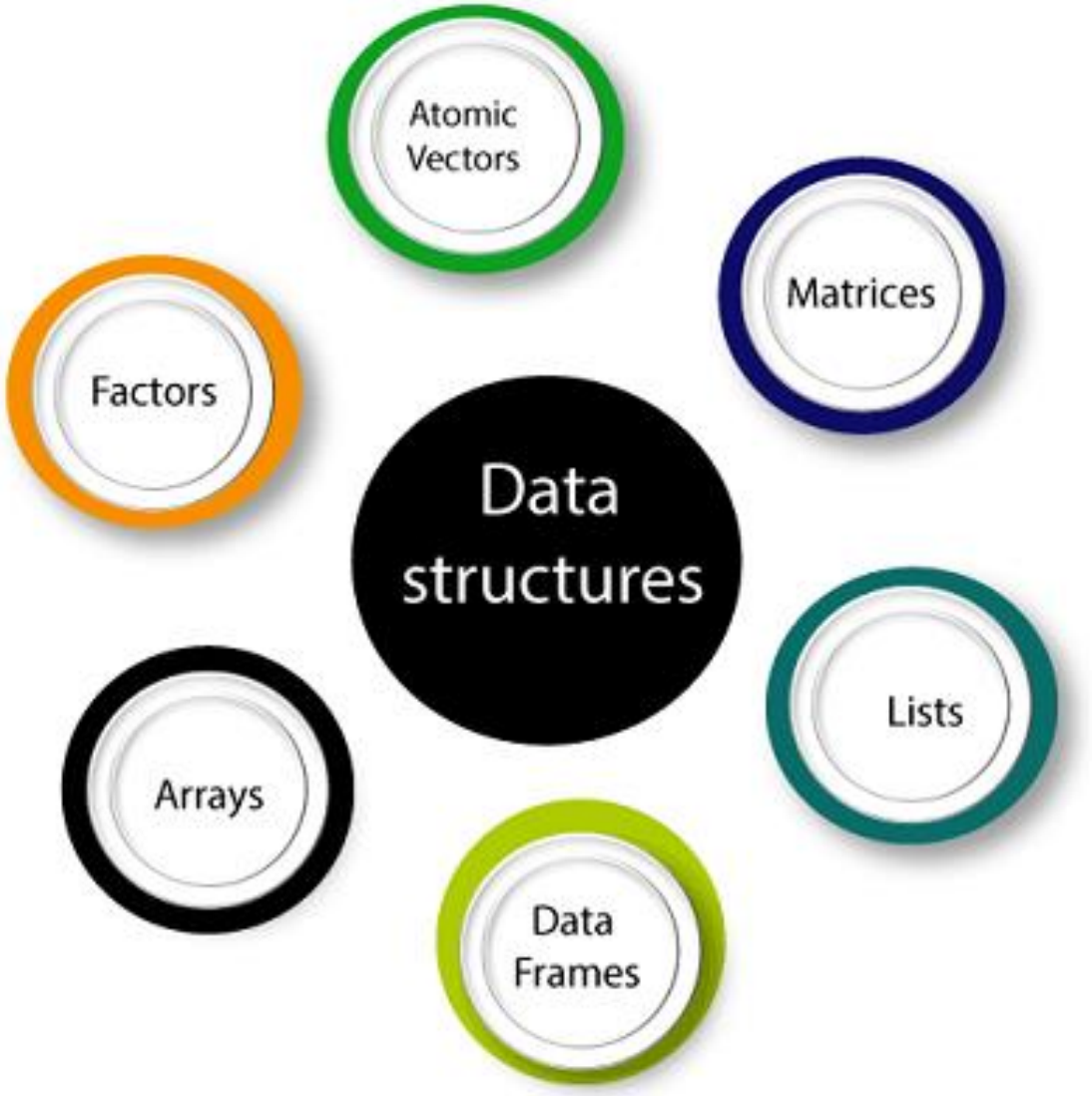
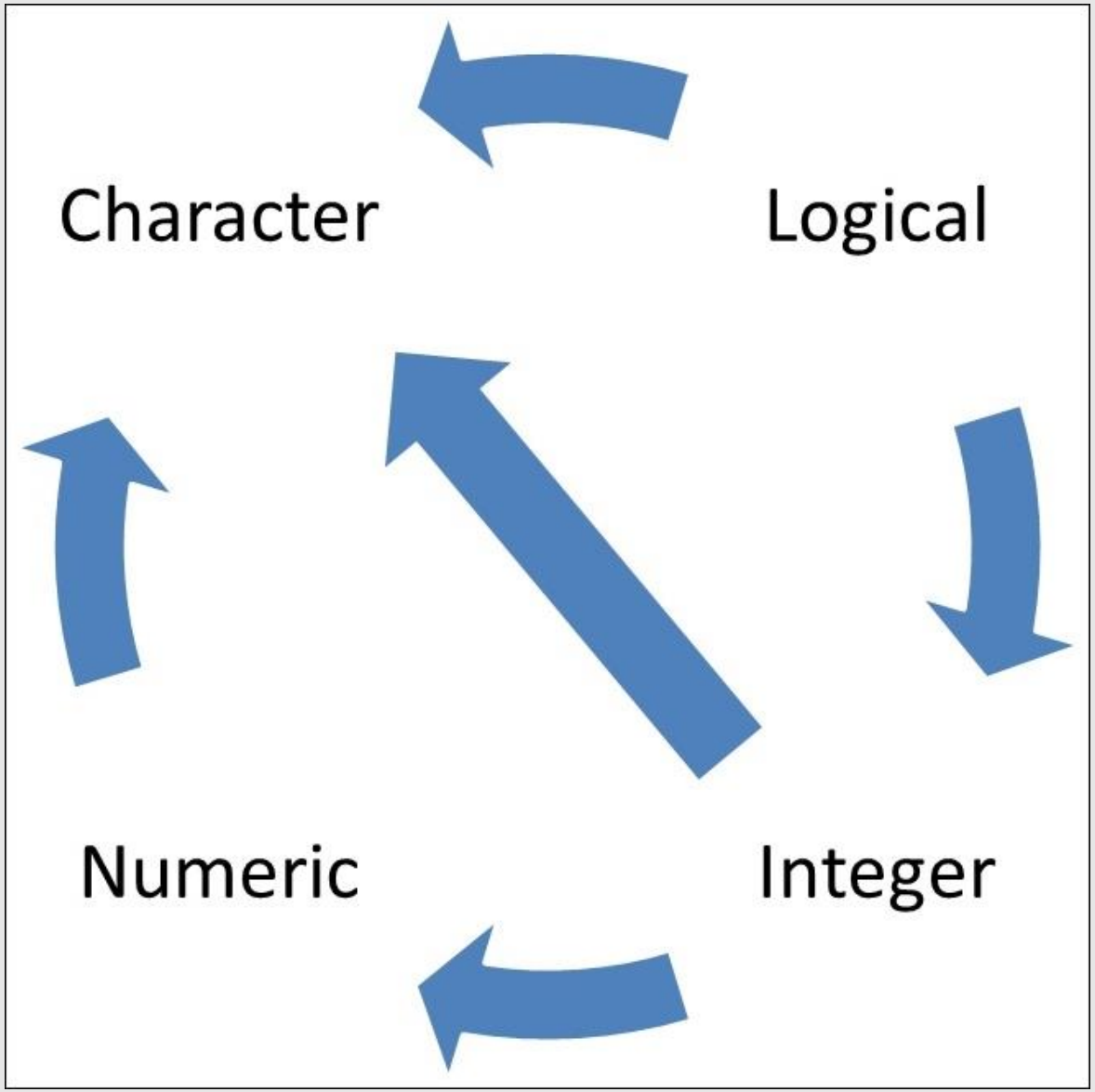
Data structures

Alok Yadav


Atomic Vectors
Matrices
Factors
Data structures
Lists
Arrays
Data Frames


Character
Logical
Numeric
Integer

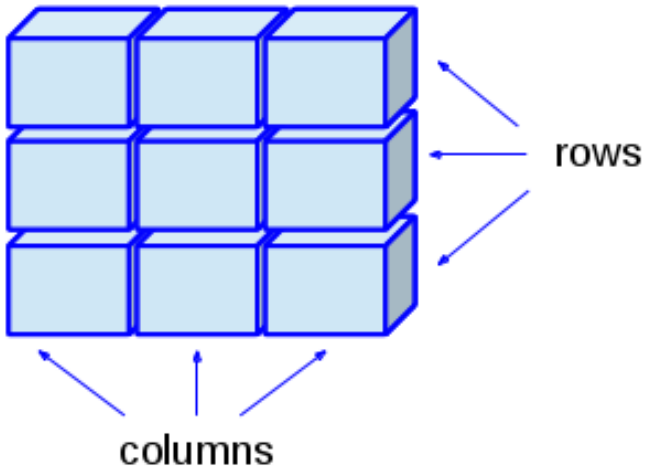
Data Structure

- R's base data structures can be organised by their dimensionality (1d, 2d, or nd) and whether they're homogeneous (all contents must be of the same type) or heterogeneous (the contents can be of different types).
- This gives rise to the five data types most often used in data analysis:
- **Homogeneous:** Atomic vector (1d), Matrix(2d), Array(nd)
- **Heterogeneous:** List, Data frame
- Note that R has no 0-dimensional, or scalar types. Individual numbers or strings, which you might think would be scalars, are actually vectors of length one.
- Given an object, the best way to understand what data structures it's composed of is to use str().
- str() is short for structure and it gives a compact, human readable description of any R data structure.

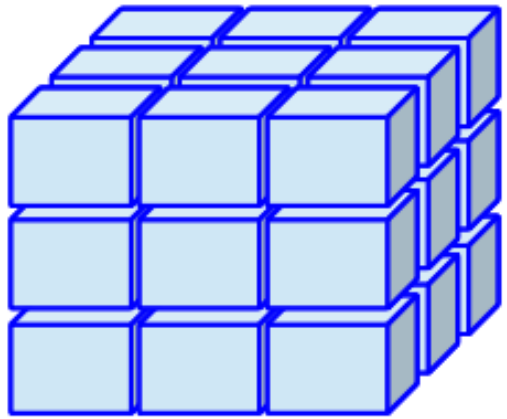
	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	

Vector

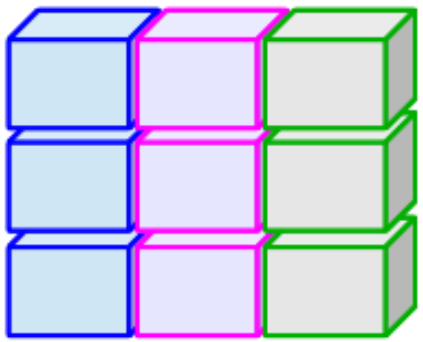
Matrix



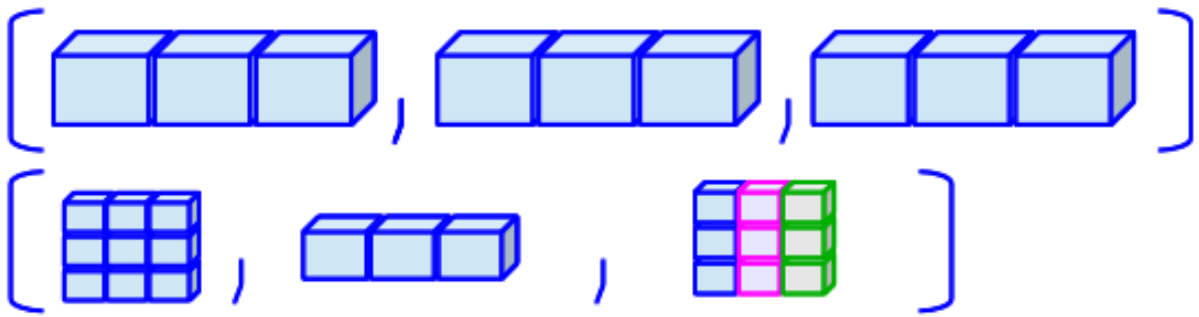
Array



Data Frame
(Table)



Lists



Vectors

- The basic data structure in R is the vector.
- Vectors come in two flavours: atomic vectors and lists.
- They have three common properties:
- **Type**, typeof(), what it is.
- **Length**, length(), how many elements it contains.
- **Attributes**, attributes(), additional arbitrary metadata.
- They differ in the types of their elements: all elements of an atomic vector must be the same type,
- whereas the elements of a list can have different types.

Vectors

- Contains a sequence of items of the same type.
- This is most basic structure. Items of a vector can be accessed using [].
- Function length can be called to know the number of items.
- NB: is.vector() does not test if an object is a vector.
- Instead it returns TRUE only if the object is a vector with no attributes apart from names.
- Use is.atomic(x) || is.list(x) to test if an object is actually a vector.

Atomic vectors

- There are four common types of atomic vectors
- logical, integer, double (often called numeric), and character.
- There are two rare types that I will not discuss further: complex and raw.
- Atomic vectors are usually created with c(), short for combine:

- https://blog.usejournal.com/understand-basic-to-advance-data-structure-used-in-r-to-use-efficiently-b51b7fd8aff0

- list: Represented as a vector but can contain items of different types. Different columns can contain different lengths. Items of a list can be accessed using [[]]. This is a recursive data type: lists can contain other lists.

Matrix

Syntax:

Matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames= NULL)

Array

- In arrays, data is stored in the form of matrices, rows, and columns.
- We can use the matrix level, row index, and column index to access the matrix elements.
- R arrays are the data objects which can store data in more than two dimensions.
- An array is created using the **array() function**.
- We can use vectors as input and create an array using the below-mentioned values in the dim parameter.

R Array Syntax

```
1.  Array_NAME <- array(data, dim = (row_Size, column_Size,
    matrices, dimnames)
```

- **data** – Data is an input vector that is given to the array.
- **matrices** – Array in R consists of multi-dimensional matrices.
- **row_Size** – row_Size describes the number of row elements that an array can store.
- **column_Size** – Number of column elements that can be stored in an array.
- **dimnames** – Used to change the default names of rows and columns to the user's preference.

Arguments in Array

The array function in R can be written as:

```
1.  array(data = NA, dim = length(data), dimname = NULL)
```

- **data** is a vector that provides data to fill the array.
- **dim** attribute provides maximum indices in each dimension

- How to Create Array in R
- Now, we will create an R array of two 3×3 matrices each with 3 rows and 3 columns.
- **# Create two vectors of different lengths.**
- vector1 <- **c**(2,9,3)
- vector2 <- **c**(10,16,17,13,11,15)
- **# Take these vectors as input to the array.**
- result <- **array**(**c**(vector1,vector2),dim = **c**(3,3,2))

Different Operations on Rows and Columns

1. Naming Columns And Rows

We can give names to the rows, columns, and matrices in the array by using the dimnames parameter.

```
# Create two vectors of different lengths.
vector1 <- c(2,9,6)
vector2 <- c(10,15,13,16,11,12)
column.names <- c("COL1","COL2","COL3")
row.names <- c("ROW1","ROW2","ROW3")
matrix.names <- c("Matrix1","Matrix2")
# Take these vectors as input to the array.
result <- array(c(vector1,vector2),dim = c(3,3,2),dimnames = list(row.names,column.names,
matrix.names))
print(result)
```

We will be using the **apply()function** for calculations in an array in R.

Syntax

```
apply(x, margin, fun)
```

Following is the description of the parameters used –

- x is an array.
- a margin is the name of the dataset used.
- fun is the function to be applied to the elements of the array.

Dataframe

- A data frame is an array.
- Unlike an array, the data we store in the columns of the data frame can be of various types.
- It means one column might be a numeric variable, another might be a factor, and a third might be a character variable.
- All columns have to be of the same length.

Characteristics of R Data Frame

- The column names should be non-empty.
- The row names should be unique.
- The data frame can hold the data which can be a numeric, character or of factor type.
- Each column should contain the same number of data items.

- We can create a data frame by passing the variable a,b,c,d into the data.frame() function.
- We can name the columns with name() and simply specify the name of the variables.

- **Arguments**:
- **df**: It can be a matrix to convert as a data frame or a collection of variables to join
- **stringsAsFactors**: Convert string to factor by default

```
data.frame(df, stringsAsFactors = TRUE)
```