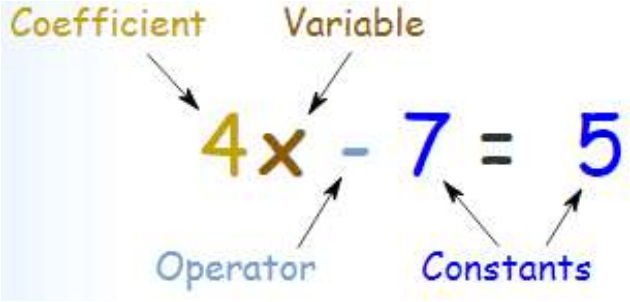
R Programming

Understanding data types, reading data in R, data manipulation techniques

Variables in R

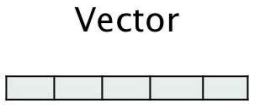
- A variable provides us with named storage that our programs can manipulate.
- A valid variable name consists of letters, numbers and the dot or underline characters.
- The variable name starts with a letter or the dot not followed by a number.
- Variable Assignment
  - The variables can be assigned values using leftward (<-), rightward (->) and equal to (=) operator.
  - The values of the variables can be printed using print() or cat() function.
  - The cat() function combines multiple items into a continuous print output.

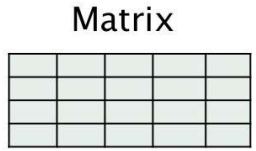


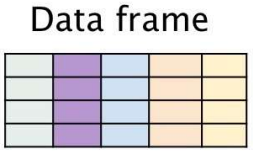
Variables in R

- # Assignment using equal operator.
- var.1 = c(0,1,2,3)
- # Assignment using leftward operator.
- var.2 <- c("learn","R")
- # Assignment using rightward operator.
- c(TRUE,1) -> var.3
- print(var.1)
- cat ("var.1 is ", var.1 ,"\n")
- cat ("var.2 is ", var.2 ,"\n")
- cat ("var.3 is ", var.3 ,"\n")

Variables	Example
integer	100
numeric	0.05
character	“hello”
logical	TRUE
factor	“Green”





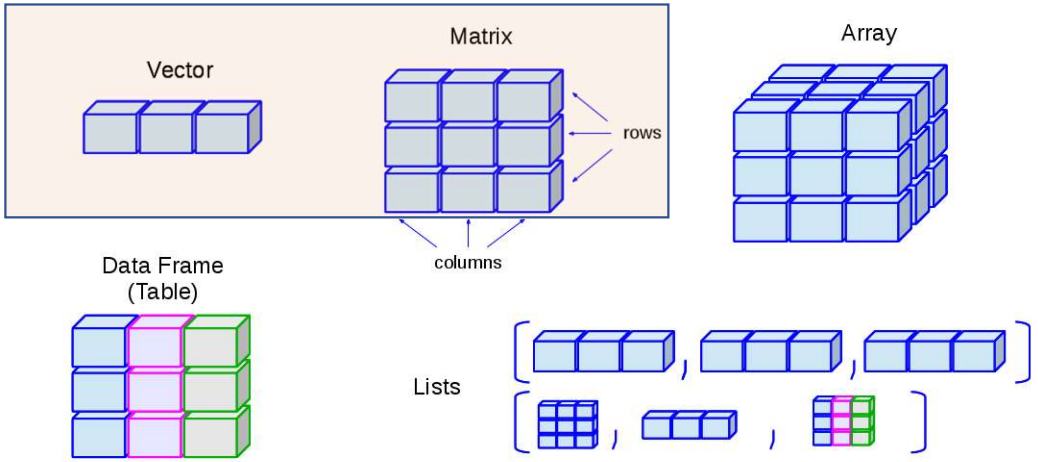


- When we execute the above code, it produces the following result –
  - [1] 0 1 2 3
  - var.1 is 0 1 2 3
  - var.2 is learn R
  - var.3 is 1 1

Data Structures in R

- **Vectors**
  - Most Simplest structure in R
  - If data has only one dimension, like a set of digits, then vectors can be used to represent it.

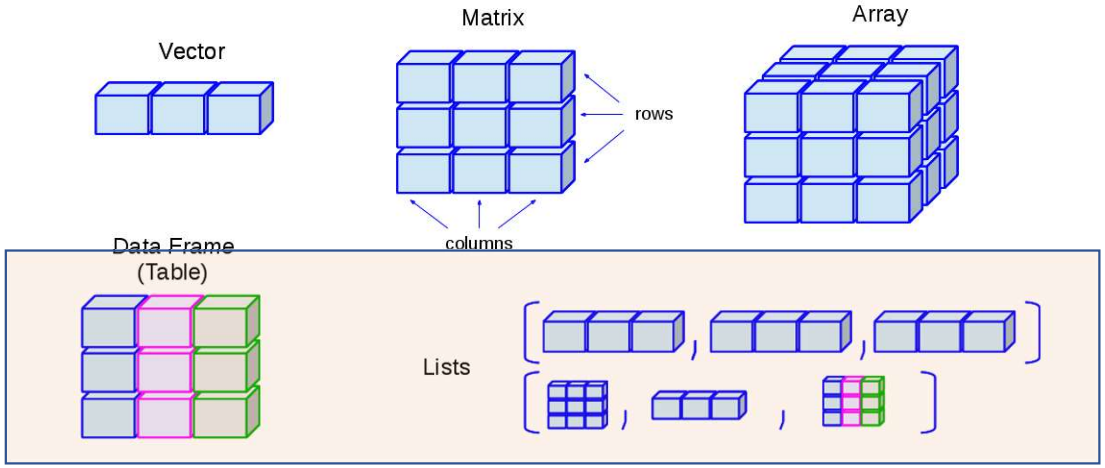
- **Matrices**
  - Used when data is a higher dimensional array
  - But contains only data of a single class Eg : only character or numeric



Data Structures in R

- **Data Frames**
  - It is like a single table with rows and columns of data
  - Contains columns or lists of different data

- **Lists**
  - Used when data cannot be represented by data frames
  - It contain all kinds of other objects, including other lists or data frames
  - Very Flexible



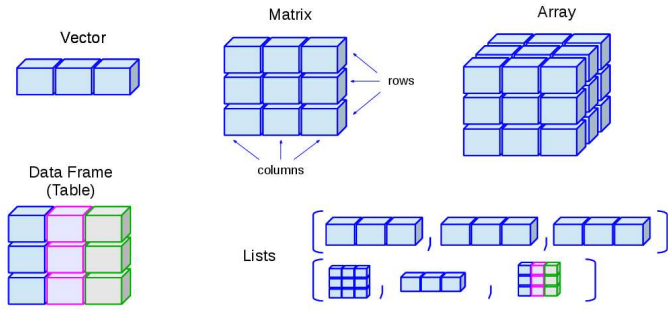
Vectors

- A vector is a sequence of data elements of the same basic type.

- o <- c(1,2,5.3,6,-2,4) # Numeric vector
- p <- c("one","two","three","four","five","six") # Character vector
- q <- c(TRUE,TRUE,FALSE,TRUE,FALSE,TRUE) # Logical vector
- o;p;q
- [1] 1.0 2.0 5.3 6.0 -2.0 4.0
- [1] "one" "two" "three" "four" "five" "six"
- [1] TRUE TRUE FALSE TRUE FALSE

Vectors

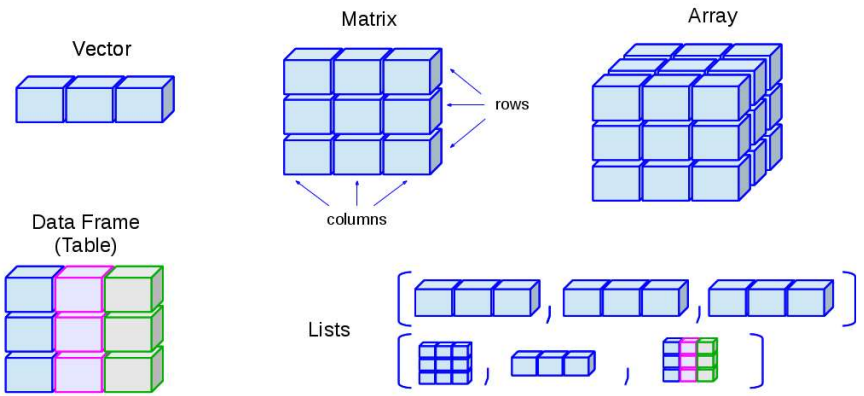
- > o[q] # Logical vector can be used to extract vector components
- [1] 1 2 6 4
- > names(o) <- p # Give each component a name
- > o
- one two three four five six
- 1.0 2.0 5.3 6.0 -2.0 4.0
- > o["three"] # Extract your components by "calling" their names
- three
- 5.3



Matrices

- A matrix is a collection of data elements arranged in a two-dimensional rectangular layout.
- Same as vector, the components in a matrix must be of the same basic type.
- The following is an example of a matrix with 4 rows and 3 columns.
  - t <- matrix(1:12,nrow=4,ncol=3,byrow = FALSE)
  - t

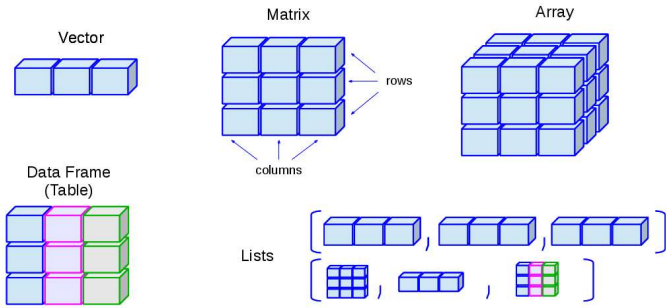
  - [,1] [,2] [,3]
  - [1,] 1 5 9
  - [2,] 2 6 10
  - [3,] 3 7 11
  - [4,] 4 8 12



Matrices

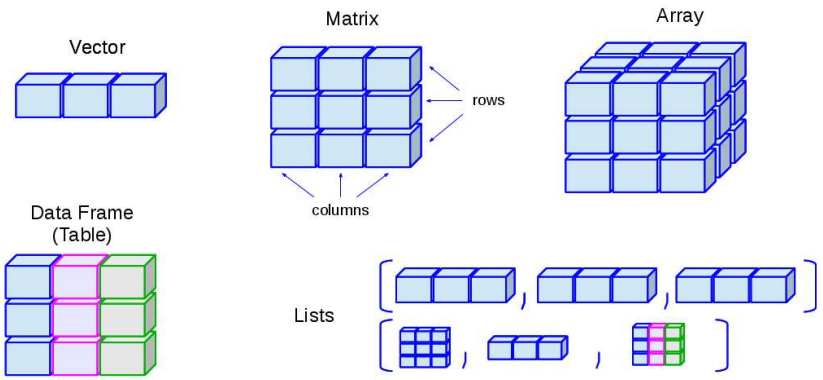
- Similar to vectors, matrices also use [] to reference elements.

- > t[2,3] # component at 2nd row and 3rd column
- [1] 10
- > t[,3] # 3rd column of matrix
- [1] 9 10 11 12
- > t[4,] # 4th row of matrix
- [1] 4 8 12
- > t[2:4,1:3] # rows 2,3,4 of columns 1,2,3
- [,1] [,2] [,3]
- [1,] 2 6 10
- [2,] 3 7 11
- [3,] 4 8 12



Data frames

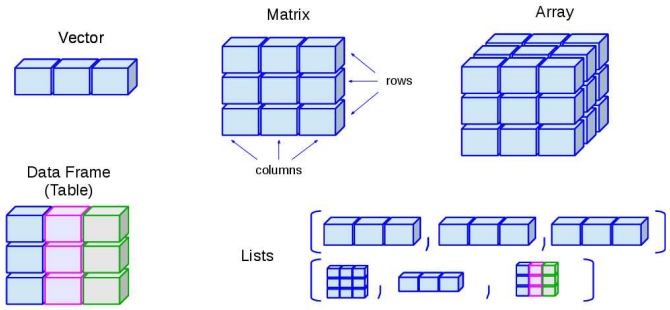
- A data frame is more general than a matrix, in that different columns can have different basic data types.
  - > d <- c(1,2,3,4)
  - > e <- c("red", "white", "red", NA)
  - > f <- c(TRUE,TRUE,TRUE,FALSE)
  - > mydata <- data.frame(d,e,f)
  - > names(mydata) <- c("ID","Color","Passed") # variable names
  - > mydata
  - ID Color Passed
  - 1 1 red TRUE
  - 2 2 white TRUE
  - 3 3 red TRUE
  - 4 4 <NA> FALSE



Data frames

- Extracting components from data frames is somehow similar to what we did for matrices, but after assigning names to each column (variable), it becomes more flexible.

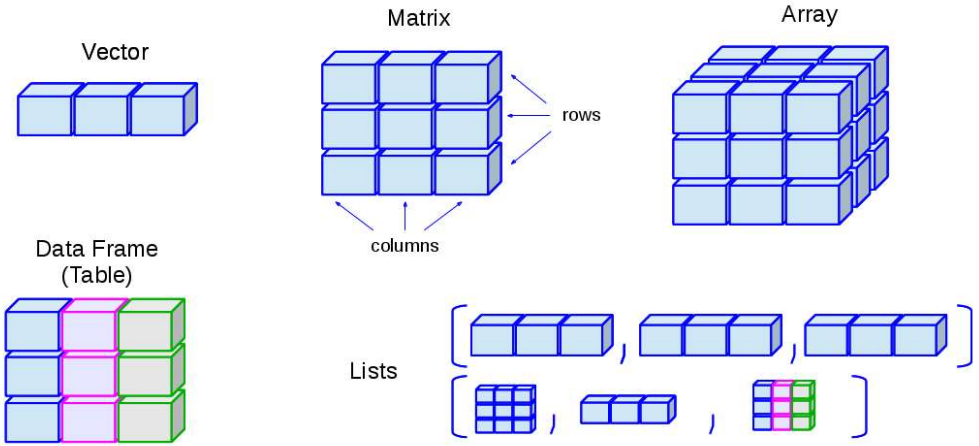
- > mydata$ID # try mydata["ID"] or mydata[1]
- [1] 1 2 3 4
- > mydata$ID[3] # try mydata[3,"ID"] or mydata[3,1]
- [1] 3
- > mydata[1:2,] # first two records
- ID Color Passed
- 1 1 red TRUE
- 2 2 white TRUE



List

- A list is a generic vector containing other objects.
- There is no restriction on data types or length of the components.
  - # a list with a vector, a matrix, a data frame defined earlier and a scalar
  - > p=c("one", "two", "three", "four", "five", "six")
  - > l <-list(vec=p, mat=t, fra=mydata, count=3)
  - > l
  - l$vec
  - [1] "one" "two" "three" "four" "five" "six"

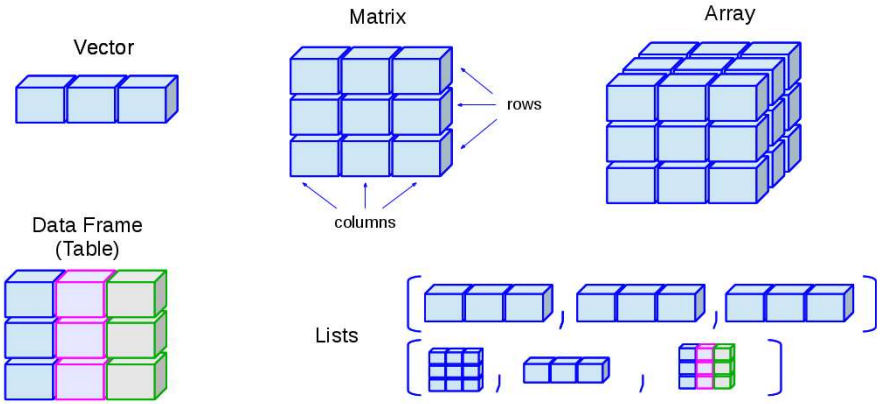
  - l$mat
  - [,1] [,2] [,3]
  - [1,] 1 5 9
  - [2,] 2 6 10
  - [3,] 3 7 11
  - [4,] 4 8 12



List

- l$fra
- ID Color Passed
- 1 1 red TRUE
- 2 2 white TRUE
- 3 3 red TRUE
- 4 4 <NA> FALSE

- l$count
- [1] 3
- > l$vec # extract components from list
- [1] "one" "two" "three" "four" "five" "six"
- > l$mat[2,3]
- [1] 10
- > lfraColor
- [1] red white red <NA>
- Levels: red white

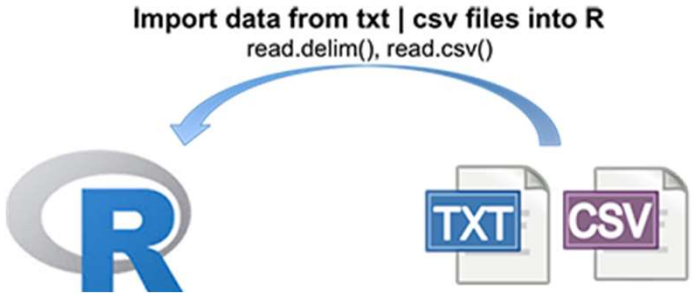


Reading and writing CSV files

- Example of importing CSV data are provided below.
- Read from a Comma Delimited Text File
  - # first row contains variable names
  - # “row.names” assigns the variable id to row names
  - # If we do not specify row.names then it would create another running serial number to it

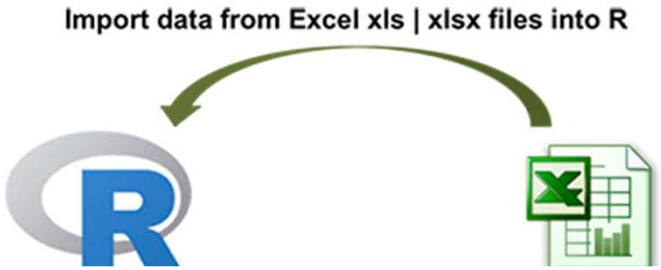
  - mydata <- read.csv("empdata.csv", header=TRUE, row.names="Employee_ID")
  - mydata

- Write to Comma Delimited Text File
  - write.csv(mydata, "MyEmpData.csv")



Importing data from excel

- From Excel
  - You can use the xlsx package to access Excel files. The first row should contain variable/column names.



  - install.packages("xlsx")
  - library(xlsx)
  - write.xlsx(mydata, "EmployeeSales.xlsx", row.names= F)
  - mydata <- read.xlsx("EmployeeSales.xlsx", 1)
  - mydata
  - # read in the worksheet named mysheet
  - mydata <- read.xlsx("EmployeeSales.xlsx", sheetName = "Sheet1")
  - mydata

Selecting rows/observations

- Suppose you have a data frame, df - consisting of three vectors that consist of information such as height, weight, age.
  - df <- data.frame(c(183, 85, 40), c(175, 76, 35), c(178, 79, 38))
  - names(df) <- c("Height", "Weight", "Age")

- # All Rows and All Columns
- df[,]
- # First row and all columns
- df[1,]
- # First two rows and all columns
- df[1:2,]

Selecting rows/observations

- # First and third row and all columns
- df[c(1,3),]
- # First Row and 2nd and third column
- df[1, 2:3]
- # First, Second Row and Second and Third Column
- df[1:2, 2:3]

Order data

- order() returns the element order that results in a sorted vector

```
> students <- c("John", "Alice", "Zeus", "Tim")
> students
[1] "John"  "Alice" "Zeus"  "Tim"
> students[order(students)]
[1] "Alice" "John"  "Tim"   "Zeus"
> order(students)
[1] 2 1 4 3
> students[order(students)]
[1] "Alice" "John"  "Tim"   "Zeus"
```

- Application: Very useful for sorting dataframes


Thank
you !!!