# SC-MD_Sept-19

Srijan Kundu

2022-09-19

```
library(matlib)
```

## Dataframe

A dataframe is a collection of vectors (may be of similar or different nature), but of equal length.

```
rm(list=ls())
```

Define some variables:

```
ht = c(1.53,1.62,1.49,1.68,1.55,1.79,1.64)
wt = c(45,82,42,55,41,72,68)
gn = c("F", "F", "M", "M", "F", "M", "M")
```

Storing the above data in a single data-structure.

```
data = data.frame(ht, wt, gn)
is.data.frame(data)
```

```
## [1] TRUE
```

```
typeof(data)
```

```
## [1] "list"
```

```
summary(data)
```

```
##        ht              wt             gn
##  Min.   :1.490   Min.   :41.00   Length:7
##  1st Qu.:1.540   1st Qu.:43.50   Class :character
##  Median :1.620   Median :55.00   Mode  :character
##  Mean   :1.614   Mean   :57.86
##  3rd Qu.:1.660   3rd Qu.:70.00
##  Max.   :1.790   Max.   :82.00
```

Data of any variable can be extracted using the accessor operator, using the `$` sign.

```
data$ht
```

```
## [1] 1.53 1.62 1.49 1.68 1.55 1.79 1.64
```

```
data$gn
```

```
## [1] "F" "F" "M" "M" "F" "M" "M"
```

To obtain the data on any particular variable for any particular unit, we run `data[unit_number, variable_number]`

```
data[1, ]
```

```
##     ht wt gn
## 1 1.53 45  F
```

```
data$ht[1]
```

```
## [1] 1.53
```

```
data[3, 3]
```

```
## [1] "M"
```

```
data[, 3]
```

```
## [1] "F" "F" "M" "M" "F" "M" "M"
```

Obtain the data for all persons who satisfy the criteria: wt>65

```
data$wt>65
```

```
## [1] FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE
```

```
data[data$wt>65, ]
```

```
##     ht wt gn
## 2 1.62 82  F
## 6 1.79 72  M
## 7 1.64 68  M
```

Obtain data for all people who satisfy the criteria weight $> 60$ and height $> 1.64$.

```
data[data$wt>60 & data$ht > 1.64, ]
```

```
##     ht wt gn
## 6 1.79 72  M
```

```
data[data$wt>60 | data$ht > 1.64, ]
```

```
##     ht wt gn
## 2 1.62 82  F
## 4 1.68 55  M
## 6 1.79 72  M
## 7 1.64 68  M
```

Obtain the data for males.

```
data[data$gn == "M", ]
```

```
##     ht wt gn
## 3 1.49 42  M
## 4 1.68 55  M
## 6 1.79 72  M
## 7 1.64 68  M
```

Obtain the data on height for people with weight $> 65$.

```
data[data$wt>65, 1]
```

```
## [1] 1.62 1.79 1.64
```

The function `str` is very useful for finding more about the structure of a dataframe.

```
str(data)
```

```
## 'data.frame':    7 obs. of  3 variables:
##  $ ht: num  1.53 1.62 1.49 1.68 1.55 1.79 1.64
##  $ wt: num  45 82 42 55 41 72 68
##  $ gn: chr  "F" "F" "M" "M" ...
```

```
names(data)
```

```
## [1] "ht" "wt" "gn"
```

Sort the datafame wrt height.

```
order(data$ht)
```

```
## [1] 3 1 5 2 7 4 6
```

```
data[order(data$ht), ]
```

```
##     ht wt gn
## 3 1.49 42  M
## 1 1.53 45  F
## 5 1.55 41  F
## 2 1.62 82  F
## 7 1.64 68  M
## 4 1.68 55  M
## 6 1.79 72  M
```

The function `order` takes a vector as input, and returns the vector of indices that sorts out the input order. Calculate the mean of height from this dataframe.

```
mean(data$ht)
```

```
## [1] 1.614286
```

To apply a function on each and every variable of a dataframe, we use the function `sapply(data, function)`.

```
sapply(data, mean)
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
##        ht        wt        gn
##  1.614286 57.857143        NA
```

```
sapply(data$wt, sum)
```

```
## [1] 45 82 42 55 41 72 68
```

Take a dataframe with 5 individuals with marks: Stat: 92, 87, 55, 84, 90 Math: 87, 65, 32, 99, 47

```
stat = c(92, 87, 55, 84, 90)
math = c(87, 65, 32, 99, 47)
data1 = data.frame(stat, math)
data1
```

```
##   stat math
## 1   92   87
## 2   87   65
## 3   55   32
## 4   84   99
## 5   90   47
```

Obtain mean, variance and sum.

```
sapply(data1, sum)
```

```
## stat math
##  408  330
```

```
sapply(data1, var)
```

```
##  stat  math
## 230.3 762.0
```

```
sapply(data1, mean)
```

```
## stat math
## 81.6 66.0
```

```
summary(data1)
```

```
##       stat           math
##  Min.   :55.0   Min.   :32
##  1st Qu.:84.0   1st Qu.:47
##  Median :87.0   Median :65
##  Mean   :81.6   Mean   :66
##  3rd Qu.:90.0   3rd Qu.:87
##  Max.   :92.0   Max.   :99
```

## Matrix

```
A = matrix(c(1,4,7,2,5,8,3,6,9), nrow = 3, ncol = 3, byrow = T)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
echelon(A, verbose = T, frac = T)
```

```
##
## Initial matrix:
##      [,1] [,2] [,3]
## [1,] 1    4    7
## [2,] 2    5    8
## [3,] 3    6    9
##
## row: 1
##
##  exchange rows 1 and 3
##      [,1] [,2] [,3]
## [1,] 3    6    9
## [2,] 2    5    8
## [3,] 1    4    7
##
##  multiply row 1 by 1/3
##      [,1] [,2] [,3]
## [1,] 1    2    3
## [2,] 2    5    8
```

```
## [3,] 1    4    7
##
##  multiply row 1 by 2 and subtract from row 2
##      [,1] [,2] [,3]
## [1,] 1    2    3
## [2,] 0    1    2
## [3,] 1    4    7
##
##  subtract row 1 from row 3
##      [,1] [,2] [,3]
## [1,] 1    2    3
## [2,] 0    1    2
## [3,] 0    2    4
##
## row: 2
##
##  exchange rows 2 and 3
##      [,1] [,2] [,3]
## [1,] 1    2    3
## [2,] 0    2    4
## [3,] 0    1    2
##
##  multiply row 2 by 1/2
##      [,1] [,2] [,3]
## [1,] 1    2    3
## [2,] 0    1    2
## [3,] 0    1    2
##
##  multiply row 2 by 2 and subtract from row 1
##      [,1] [,2] [,3]
## [1,]  1    0   -1
## [2,]  0    1    2
## [3,]  0    1    2
##
##  subtract row 2 from row 3
##      [,1] [,2] [,3]
## [1,]  1    0   -1
## [2,]  0    1    2
## [3,]  0    0    0
##
## row: 3
```

In matrix, to extract the `(i,j)th` element, we use `A[i][j]`.

```
A[1, 3]
```

```
## [1] 7
```

```
A[, c(1,3)]
```

```
##      [,1] [,2]
## [1,]    1    7
## [2,]    2    8
## [3,]    3    9
```

```
A[, -2]
```

```
##      [,1] [,2]
## [1,]    1    7
## [2,]    2    8
## [3,]    3    9
```

```
A[-2, ]
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    3    6    9
```

Obtain the mean of the first row.

```
mean(A[1, ])
```

```
## [1] 4
```

```
apply(A, 1, mean)
```

```
## [1] 4 5 6
```

```
apply(A, 2, mean)
```

```
## [1] 2 5 8
```

`sapply` is to be applied on dataframe; `apply` is to be applied on array.

Storing the matrix in a dataframe:

```
M = data.frame(A)
is.data.frame(M)
```

```
## [1] TRUE
```

```
is.data.frame(A)
```

```
## [1] FALSE
```

```
M = as.matrix(A)
M
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
B = matrix(c(2,3,1,3,2,6,7,9,0), nrow = 3, byrow = T)
B
```

```
##      [,1] [,2] [,3]
## [1,]    2    3    1
## [2,]    3    2    6
## [3,]    7    9    0
```

Row-augmentation of two matrices can be done by using the function `rbind`, but the columns of the matrics has to be same. Similarly, column augmentation can be done using the function `cbind`, but the number of rows of the two matrics must be same

```
rbind(A, B)
```

```
##      [,1] [,2] [,3]
```

```
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
## [4,]    2    3    1
## [5,]    3    2    6
## [6,]    7    9    0
```

```
cbind(A,B)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    4    7    2    3    1
## [2,]    2    5    8    3    2    6
## [3,]    3    6    9    7    9    0
```

```
A+2
```

```
##      [,1] [,2] [,3]
## [1,]    3    6    9
## [2,]    4    7   10
## [3,]    5    8   11
```

```
A/B
```

```
##           [,1]      [,2]     [,3]
## [1,] 0.5000000 1.3333333 7.000000
## [2,] 0.6666667 2.5000000 1.333333
## [3,] 0.4285714 0.6666667      Inf
```

Sum of all elements of a matrix:

```
sum(A)
```

```
## [1] 45
```

Make a new matrix whose (i, j)th element is the sin of the (i,j)th element of A.

```
sin(A)
```

```
##           [,1]       [,2]      [,3]
## [1,] 0.8414710 -0.7568025 0.6569866
## [2,] 0.9092974 -0.9589243 0.9893582
## [3,] 0.1411200 -0.2794155 0.4121185
```

Element wise operation is possible in R.

## Matrix Algebra

Matrix addition

```
A+B
```

```
##      [,1] [,2] [,3]
## [1,]    3    7    8
## [2,]    5    7   14
## [3,]   10   15    9
```

Matrix multiplication

```
A%*%B
```

```
##      [,1] [,2] [,3]
## [1,]   63   74   25
```

```
## [2,]   75   88   32
## [3,]   87  102   39
```

Transpose of a matrix:

```
t(A)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

Determinant of a matrix:

```
det(B)
```

```
## [1] 31
```

Inverse of matrix B:

```
solve(B)
```

```
##            [,1]        [,2]       [,3]
## [1,] -1.7419355  0.29032258  0.5161290
## [2,]  1.3548387 -0.22580645 -0.2903226
## [3,]  0.4193548  0.09677419 -0.1612903
```

Eigen values and eigen vecotrs:

```
eigen(B)
```

```
## eigen() decomposition
## $values
## [1] 10.5405517 -6.0548191 -0.4857325
##
## $vectors
##            [,1]       [,2]       [,3]
## [1,] -0.3017688  0.1369153 -0.7849877
## [2,] -0.6167521 -0.6240430  0.6028421
## [3,] -0.7270161  0.7693013  0.1427432
```

```
#This gives orthonormal eigen vectors
```

Solving a system of linear equations:

```
b = c(1,3,5)
solve(B, b)
```

```
## [1]  1.70967742 -0.77419355 -0.09677419
```

```
A = matrix(1:4,nrow = 2)
B = c(11, 2, 3)
A+B
```

```
## Warning in A + B: longer object length is not a multiple of shorter object
## length
```

```
##      [,1] [,2]
## [1,]   12    6
## [2,]    4   15
```

```
library(pracma)
```

```
##
## Attaching package: 'pracma'
## The following objects are masked from 'package:matlib':
##
##     angle, inv
R(A)
## [1] 2
rank(A)
## [1] 1 2 3 4
Rank(A)
## [1] 2
library(Matrix)
##
## Attaching package: 'Matrix'
## The following objects are masked from 'package:pracma':
##
##     expm, lu, tril, triu
rankMatrix(A)
## [1] 2
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 4.440892e-16
```

## List

```
u = 1:5
v = 1:3
list1 = list(u, v)
list1
## [[1]]
## [1] 1 2 3 4 5
##
## [[2]]
## [1] 1 2 3
list1[[1]]
## [1] 1 2 3 4 5
list1[[1]][2]
## [1] 2
list2 = list(a = u, b = v)
list2
```

```
## $a
## [1] 1 2 3 4 5
##
## $b
## [1] 1 2 3
```

Mean of first list:

```
mean(list2[[1]])
```

```
## [1] 3
```

```
lapply(list1, mean)
```

```
## [[1]]
## [1] 3
##
## [[2]]
## [1] 2
```

```
unlist(list1)
```

```
## [1] 1 2 3 4 5 1 2 3
```

Give name to the columns:

```
colnames(A) = c("col1", "col2")
A
```

```
##      col1 col2
## [1,]    1    3
## [2,]    2    4
```

```
rownames(A) = c("row1", "row2")
colnames(A) = month.abb[1:2]
A
```

```
##      Jan Feb
## row1   1   3
## row2   2   4
```