Exercise 2.6 - 2.7

2.6 Filtering vectors based on conditions

11)

```
x <- c(80, 39, NA, 51, 51, 11, NA, NA, NA, 100, 80, 70)
```

a).

```
nonMissings <- x[!is.na(x)] # since is.na command gives us whether the value is a missing value or not
nonMissings
```

```
[1]  80  39  51  51  11 100  80  70
```

b).

```
missingsOdd <- x[(x %% 2) != 0]
missingsOdd
```

```
[1] 39 NA 51 51 11 NA NA NA
```

c).

```
odd <- x[!is.na(x) & (x %% 2) != 0] # extracting only the values which are not odd
#ignoring missing values
odd
```

```
[1] 39 51 51 11
```

d).

```
x <- c(80, 39, NA, 51, 51, 11, NA, NA, NA, 100, 80, 70)
notIn <- x[!is.na(x) &!(x %in% 1:50)] # extracting only the values which not in the set 1:50
notIn
```

```
[1]  80  51  51 100  80  70
```

2.7 Modify a vector

12)

```
age <- c(20, 30, 40, 41, 32, 32, 25,  NA, NA, -4, -6, 9999, 10000)
```

a).

```
a <- replace(age, which(age < 0), NA) # assigning NA to negative values
a
```

```
 [1]    20    30    40    41    32    32    25    NA    NA    NA    NA  9999
[13] 10000
```

b).

```
age <- c(20, 30, 40, 41, 32, 32, 25,  NA, NA, -4, -6, 9999, 10000)
age [age < 0] <- 0 # assigning zero for all negative values

valid <- age[age %in% 1:100] # extracting the valid responses

mean(valid, rm.na=TRUE)
```

```
[1] 31.42857
```

13)

```
set.seed(17620212)
b <- rnorm(100)
b
```

```
  [1]  0.589528488 -0.662937204  0.238279278  0.183757174 -0.002364399
  [6]  0.289002107  0.258796402  0.982174159  0.378628085  0.015035037
 [11] -1.203312799  1.510436562  0.219368378 -0.642429444 -0.373969124
 [16] -0.239829685 -0.186344734  0.517975563 -1.256355393  1.067433297
 [21]  1.035128935 -1.016002843  0.830122365  0.427420672  0.170429825
 [26] -0.001345883 -1.022893025 -0.908602635  0.502535054  0.315929086
 [31]  1.294309571 -0.303323749 -0.322819573 -1.377566743  2.714915313
 [36] -0.512573266  1.342424819 -0.457104082 -1.593015886 -0.202338403
 [41]  1.079678527  0.456102666  1.504041202  0.378318229 -0.289765109
 [46]  1.019989890  0.665591385 -1.076213455  0.272375584  0.545493842
 [51]  0.052391342 -0.402364688  0.152662598 -1.486745812  0.102018231
 [56] -0.024357072  0.068276667  0.075642814  0.379600455 -0.988308679
 [61]  0.701330674 -0.491165150  1.494498791 -1.773934043 -0.460454009
 [66]  0.752256616  0.039189614 -0.939203562 -0.419716046  0.084067026
```

```
 [71] -1.081303093  0.780827145  0.207575277  0.733234796 -0.660969465
 [76]  1.649316796  0.491550464 -0.864054075 -0.919522275 -0.727913488
 [81]  1.197400462 -1.645388340  1.704924934 -1.650667045  0.377823148
 [86]  1.436377659 -1.143144414  0.789086285  1.049357974  2.163786809
 [91]  1.626306920  1.317779758  1.647449733  0.588881226 -0.177613835
 [96]  0.081191404  0.093051240  1.202918954 -1.783424334  0.725816313
```

a).

```
b[1:5] <- 1 # changing the first five values to 1
b
```

```
  [1]  1.000000000  1.000000000  1.000000000  1.000000000  1.000000000
  [6]  0.289002107  0.258796402  0.982174159  0.378628085  0.015035037
 [11] -1.203312799  1.510436562  0.219368378 -0.642429444 -0.373969124
 [16] -0.239829685 -0.186344734  0.517975563 -1.256355393  1.067433297
 [21]  1.035128935 -1.016002843  0.830122365  0.427420672  0.170429825
 [26] -0.001345883 -1.022893025 -0.908602635  0.502535054  0.315929086
 [31]  1.294309571 -0.303323749 -0.322819573 -1.377566743  2.714915313
 [36] -0.512573266  1.342424819 -0.457104082 -1.593015886 -0.202338403
 [41]  1.079678527  0.456102666  1.504041202  0.378318229 -0.289765109
 [46]  1.019989890  0.665591385 -1.076213455  0.272375584  0.545493842
 [51]  0.052391342 -0.402364688  0.152662598 -1.486745812  0.102018231
 [56] -0.024357072  0.068276667  0.075642814  0.379600455 -0.988308679
 [61]  0.701330674 -0.491165150  1.494498791 -1.773934043 -0.460454009
 [66]  0.752256616  0.039189614 -0.939203562 -0.419716046  0.084067026
 [71] -1.081303093  0.780827145  0.207575277  0.733234796 -0.660969465
 [76]  1.649316796  0.491550464 -0.864054075 -0.919522275 -0.727913488
 [81]  1.197400462 -1.645388340  1.704924934 -1.650667045  0.377823148
 [86]  1.436377659 -1.143144414  0.789086285  1.049357974  2.163786809
 [91]  1.626306920  1.317779758  1.647449733  0.588881226 -0.177613835
 [96]  0.081191404  0.093051240  1.202918954 -1.783424334  0.725816313
```

b).

```
length(b) # length of the vector b
```

```
[1] 100
```

```
b[96:100] <- 0 # changing last five values to 0
b
```

```
  [1]  1.000000000  1.000000000  1.000000000  1.000000000  1.000000000
  [6]  0.289002107  0.258796402  0.982174159  0.378628085  0.015035037
 [11] -1.203312799  1.510436562  0.219368378 -0.642429444 -0.373969124
 [16] -0.239829685 -0.186344734  0.517975563 -1.256355393  1.067433297
 [21]  1.035128935 -1.016002843  0.830122365  0.427420672  0.170429825
 [26] -0.001345883 -1.022893025 -0.908602635  0.502535054  0.315929086
 [31]  1.294309571 -0.303323749 -0.322819573 -1.377566743  2.714915313
```

```
 [36] -0.512573266  1.342424819 -0.457104082 -1.593015886 -0.202338403
 [41]  1.079678527  0.456102666  1.504041202  0.378318229 -0.289765109
 [46]  1.019989890  0.665591385 -1.076213455  0.272375584  0.545493842
 [51]  0.052391342 -0.402364688  0.152662598 -1.486745812  0.102018231
 [56] -0.024357072  0.068276667  0.075642814  0.379600455 -0.988308679
 [61]  0.701330674 -0.491165150  1.494498791 -1.773934043 -0.460454009
 [66]  0.752256616  0.039189614 -0.939203562 -0.419716046  0.084067026
 [71] -1.081303093  0.780827145  0.207575277  0.733234796 -0.660969465
 [76]  1.649316796  0.491550464 -0.864054075 -0.919522275 -0.727913488
 [81]  1.197400462 -1.645388340  1.704924934 -1.650667045  0.377823148
 [86]  1.436377659 -1.143144414  0.789086285  1.049357974  2.163786809
 [91]  1.626306920  1.317779758  1.647449733  0.588881226 -0.177613835
 [96]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
```

c).

```
b[b > 0.5] <- 1 # assigning 1 to values grater than 0.5
b
```

```
  [1]  1.000000000  1.000000000  1.000000000  1.000000000  1.000000000
  [6]  0.289002107  0.258796402  1.000000000  0.378628085  0.015035037
 [11] -1.203312799  1.000000000  0.219368378 -0.642429444 -0.373969124
 [16] -0.239829685 -0.186344734  1.000000000 -1.256355393  1.000000000
 [21]  1.000000000 -1.016002843  1.000000000  0.427420672  0.170429825
 [26] -0.001345883 -1.022893025 -0.908602635  1.000000000  0.315929086
 [31]  1.000000000 -0.303323749 -0.322819573 -1.377566743  1.000000000
 [36] -0.512573266  1.000000000 -0.457104082 -1.593015886 -0.202338403
 [41]  1.000000000  0.456102666  1.000000000  0.378318229 -0.289765109
 [46]  1.000000000  1.000000000 -1.076213455  0.272375584  1.000000000
 [51]  0.052391342 -0.402364688  0.152662598 -1.486745812  0.102018231
 [56] -0.024357072  0.068276667  0.075642814  0.379600455 -0.988308679
 [61]  1.000000000 -0.491165150  1.000000000 -1.773934043 -0.460454009
 [66]  1.000000000  0.039189614 -0.939203562 -0.419716046  0.084067026
 [71] -1.081303093  1.000000000  0.207575277  1.000000000 -0.660969465
 [76]  1.000000000  0.491550464 -0.864054075 -0.919522275 -0.727913488
 [81]  1.000000000 -1.645388340  1.000000000 -1.650667045  0.377823148
 [86]  1.000000000 -1.143144414  1.000000000  1.000000000  1.000000000
 [91]  1.000000000  1.000000000  1.000000000  1.000000000 -0.177613835
 [96]  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
```

```
b[b < 0.5] <- 0 # assigning 0 to values less than 0.5
b
```

```
  [1] 1 1 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1
 [38] 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1
 [75] 0 1 0 0 0 0 1 0 1 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0
```

d).

```
b <- ifelse(b == 0, "MALE", "FEMALE")
b
```

```
  [1] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "MALE"   "MALE"   "FEMALE"
  [9] "MALE"   "MALE"   "MALE"   "FEMALE" "MALE"   "MALE"   "MALE"   "MALE"
 [17] "MALE"   "FEMALE" "MALE"   "FEMALE" "FEMALE" "MALE"   "FEMALE" "MALE"
 [25] "MALE"   "MALE"   "MALE"   "MALE"   "FEMALE" "MALE"   "FEMALE" "MALE"
 [33] "MALE"   "MALE"   "FEMALE" "MALE"   "FEMALE" "MALE"   "MALE"   "MALE"
 [41] "FEMALE" "MALE"   "FEMALE" "MALE"   "MALE"   "FEMALE" "FEMALE" "MALE"
 [49] "MALE"   "FEMALE" "MALE"   "MALE"   "MALE"   "MALE"   "MALE"   "MALE"
 [57] "MALE"   "MALE"   "MALE"   "MALE"   "FEMALE" "MALE"   "FEMALE" "MALE"
 [65] "MALE"   "FEMALE" "MALE"   "MALE"   "MALE"   "MALE"   "MALE"   "FEMALE"
 [73] "MALE"   "FEMALE" "MALE"   "FEMALE" "MALE"   "MALE"   "MALE"   "MALE"
 [81] "FEMALE" "MALE"   "FEMALE" "MALE"   "MALE"   "FEMALE" "MALE"   "FEMALE"
 [89] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "FEMALE" "MALE"   "MALE"
 [97] "MALE"   "MALE"   "MALE"   "MALE"
```

Or else this can be done by using the following method as well.

```
# b <- replace(b, which(b==0), "MALE")
# b <- replace(b, which(b==1), " FEMALE")
```

Exercise 3

3.2 Matrices

Question 1:

Use the code below to create the vector uniform.values

```
set.seed(21)
uniform.values <- runif(50) # generating 50 random numbers from standard normal distribution
uniform.values
```

```
 [1] 0.78611493 0.25244560 0.69925230 0.18446075 0.95961383 0.91868340
 [7] 0.10180455 0.17219168 0.98600368 0.84939610 0.66754012 0.93521022
[13] 0.05818433 0.61861583 0.17491846 0.03767539 0.52531317 0.28218425
[19] 0.49904520 0.63382510 0.01139965 0.60785656 0.77559853 0.92397118
[25] 0.29170673 0.78907624 0.56849721 0.77843508 0.71323253 0.66904867
[31] 0.93470991 0.50646019 0.74506019 0.83835263 0.86907475 0.19311168
[37] 0.21633194 0.65042346 0.33516604 0.50765589 0.65283937 0.96557667
[43] 0.51466067 0.06165677 0.15101646 0.63556589 0.10296050 0.77269430
[49] 0.41022537 0.87023337
```

Arrange data in uniform.values according to the following formats:

a). single row matrix

```r
SingleRow<-matrix(uniform.values, nrow = 1) # nrow = 1 since a single row
SingleRow
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
[1,] 0.7861149 0.2524456 0.6992523 0.1844608 0.9596138 0.9186834 0.1018046
          [,8]      [,9]     [,10]     [,11]     [,12]      [,13]     [,14]
[1,] 0.1721917 0.9860037 0.8493961 0.6675401 0.9352102 0.05818433 0.6186158
         [,15]      [,16]     [,17]     [,18]     [,19]     [,20]      [,21]
[1,] 0.1749185 0.03767539 0.5253132 0.2821842 0.4990452 0.6338251 0.01139965
         [,22]     [,23]     [,24]     [,25]     [,26]     [,27]     [,28]
[1,] 0.6078566 0.7755985 0.9239712 0.2917067 0.7890762 0.5684972 0.7784351
         [,29]     [,30]     [,31]     [,32]     [,33]     [,34]     [,35]
[1,] 0.7132325 0.6690487 0.9347099 0.5064602 0.7450602 0.8383526 0.8690747
         [,36]     [,37]     [,38]    [,39]     [,40]     [,41]     [,42]
[1,] 0.1931117 0.2163319 0.6504235 0.335166 0.5076559 0.6528394 0.9655767
         [,43]      [,44]     [,45]     [,46]     [,47]     [,48]     [,49]
[1,] 0.5146607 0.06165677 0.1510165 0.6355659 0.1029605 0.7726943 0.4102254
         [,50]
[1,] 0.8702334
```

b). single column matrix

```
SingleColumn<-matrix(uniform.values, ncol = 1) # ncol = 1 since a single column
SingleColumn
```

```
            [,1]
 [1,] 0.78611493
 [2,] 0.25244560
 [3,] 0.69925230
 [4,] 0.18446075
 [5,] 0.95961383
 [6,] 0.91868340
 [7,] 0.10180455
 [8,] 0.17219168
 [9,] 0.98600368
[10,] 0.84939610
[11,] 0.66754012
[12,] 0.93521022
[13,] 0.05818433
[14,] 0.61861583
[15,] 0.17491846
[16,] 0.03767539
[17,] 0.52531317
[18,] 0.28218425
[19,] 0.49904520
[20,] 0.63382510
[21,] 0.01139965
[22,] 0.60785656
[23,] 0.77559853
[24,] 0.92397118
[25,] 0.29170673
[26,] 0.78907624
[27,] 0.56849721
[28,] 0.77843508
[29,] 0.71323253
[30,] 0.66904867
[31,] 0.93470991
[32,] 0.50646019
[33,] 0.74506019
[34,] 0.83835263
[35,] 0.86907475
[36,] 0.19311168
[37,] 0.21633194
[38,] 0.65042346
[39,] 0.33516604
[40,] 0.50765589
[41,] 0.65283937
[42,] 0.96557667
[43,] 0.51466067
[44,] 0.06165677
[45,] 0.15101646
[46,] 0.63556589
[47,] 0.10296050
[48,] 0.77269430
```

```
[49,] 0.41022537
[50,] 0.87023337
```

```
Column<-matrix(uniform.values) # default
Column
```

```
            [,1]
 [1,] 0.78611493
 [2,] 0.25244560
 [3,] 0.69925230
 [4,] 0.18446075
 [5,] 0.95961383
 [6,] 0.91868340
 [7,] 0.10180455
 [8,] 0.17219168
 [9,] 0.98600368
[10,] 0.84939610
[11,] 0.66754012
[12,] 0.93521022
[13,] 0.05818433
[14,] 0.61861583
[15,] 0.17491846
[16,] 0.03767539
[17,] 0.52531317
[18,] 0.28218425
[19,] 0.49904520
[20,] 0.63382510
[21,] 0.01139965
[22,] 0.60785656
[23,] 0.77559853
[24,] 0.92397118
[25,] 0.29170673
[26,] 0.78907624
[27,] 0.56849721
[28,] 0.77843508
[29,] 0.71323253
[30,] 0.66904867
[31,] 0.93470991
[32,] 0.50646019
[33,] 0.74506019
[34,] 0.83835263
[35,] 0.86907475
[36,] 0.19311168
[37,] 0.21633194
[38,] 0.65042346
[39,] 0.33516604
[40,] 0.50765589
[41,] 0.65283937
[42,] 0.96557667
[43,] 0.51466067
[44,] 0.06165677
[45,] 0.15101646
[46,] 0.63556589
[47,] 0.10296050
```

```
[48,] 0.77269430
[49,] 0.41022537
[50,] 0.87023337
```

c). matrix 5 × 10

```
matrixCol1<-matrix(uniform.values, nrow = 5, ncol = 10) # Matrix is filled by column (default)
matrixCol1
```

```
          [,1]      [,2]       [,3]       [,4]       [,5]      [,6]      [,7]
[1,] 0.7861149 0.9186834 0.66754012 0.03767539 0.01139965 0.7890762 0.9347099
[2,] 0.2524456 0.1018046 0.93521022 0.52531317 0.60785656 0.5684972 0.5064602
[3,] 0.6992523 0.1721917 0.05818433 0.28218425 0.77559853 0.7784351 0.7450602
[4,] 0.1844608 0.9860037 0.61861583 0.49904520 0.92397118 0.7132325 0.8383526
[5,] 0.9596138 0.8493961 0.17491846 0.63382510 0.29170673 0.6690487 0.8690747
          [,8]       [,9]     [,10]
[1,] 0.1931117 0.65283937 0.6355659
[2,] 0.2163319 0.96557667 0.1029605
[3,] 0.6504235 0.51466067 0.7726943
[4,] 0.3351660 0.06165677 0.4102254
[5,] 0.5076559 0.15101646 0.8702334
```

```
# or else
# matrixCol1<-matrix(uniform.values, nrow = 5, ncol = 10, byrow = FALSE)
```

```
matrixRow1<-matrix(uniform.values, nrow = 5, ncol = 10, byrow = TRUE) # Matrix is filled by row
matrixRow1
```

```
           [,1]      [,2]       [,3]       [,4]      [,5]       [,6]      [,7]
[1,] 0.78611493 0.2524456 0.69925230 0.18446075 0.9596138 0.91868340 0.1018046
[2,] 0.66754012 0.9352102 0.05818433 0.61861583 0.1749185 0.03767539 0.5253132
[3,] 0.01139965 0.6078566 0.77559853 0.92397118 0.2917067 0.78907624 0.5684972
[4,] 0.93470991 0.5064602 0.74506019 0.83835263 0.8690747 0.19311168 0.2163319
[5,] 0.65283937 0.9655767 0.51466067 0.06165677 0.1510165 0.63556589 0.1029605
          [,8]      [,9]     [,10]
[1,] 0.1721917 0.9860037 0.8493961
[2,] 0.2821842 0.4990452 0.6338251
[3,] 0.7784351 0.7132325 0.6690487
[4,] 0.6504235 0.3351660 0.5076559
[5,] 0.7726943 0.4102254 0.8702334
```

d). matrix 10 × 5

```
matrixCol2<-matrix(uniform.values, nrow = 10, ncol = 5) # Matrix is filled by column (default)
matrixCol2
```

```
           [,1]       [,2]       [,3]      [,4]       [,5]
 [1,] 0.7861149 0.66754012 0.01139965 0.9347099 0.65283937
 [2,] 0.2524456 0.93521022 0.60785656 0.5064602 0.96557667
 [3,] 0.6992523 0.05818433 0.77559853 0.7450602 0.51466067
 [4,] 0.1844608 0.61861583 0.92397118 0.8383526 0.06165677
 [5,] 0.9596138 0.17491846 0.29170673 0.8690747 0.15101646
```

```
 [6,] 0.9186834 0.03767539 0.78907624 0.1931117 0.63556589
 [7,] 0.1018046 0.52531317 0.56849721 0.2163319 0.10296050
 [8,] 0.1721917 0.28218425 0.77843508 0.6504235 0.77269430
 [9,] 0.9860037 0.49904520 0.71323253 0.3351660 0.41022537
[10,] 0.8493961 0.63382510 0.66904867 0.5076559 0.87023337
```

```
# or else
# matrixCol2<-matrix(uniform.values, nrow = 10, ncol = 5, byrow = FALSE)
```

```
matrixRow2<-matrix(uniform.values, nrow = 10, ncol = 5, byrow = TRUE) # Matrix is filled by row
matrixRow2
```

```
           [,1]      [,2]       [,3]       [,4]      [,5]
 [1,] 0.78611493 0.2524456 0.69925230 0.18446075 0.9596138
 [2,] 0.91868340 0.1018046 0.17219168 0.98600368 0.8493961
 [3,] 0.66754012 0.9352102 0.05818433 0.61861583 0.1749185
 [4,] 0.03767539 0.5253132 0.28218425 0.49904520 0.6338251
 [5,] 0.01139965 0.6078566 0.77559853 0.92397118 0.2917067
 [6,] 0.78907624 0.5684972 0.77843508 0.71323253 0.6690487
 [7,] 0.93470991 0.5064602 0.74506019 0.83835263 0.8690747
 [8,] 0.19311168 0.2163319 0.65042346 0.33516604 0.5076559
 [9,] 0.65283937 0.9655767 0.51466067 0.06165677 0.1510165
[10,] 0.63556589 0.1029605 0.77269430 0.41022537 0.8702334
```

Question 2:

Write the code to output the following matrix.

```
rnames <- c("a", "b")
matrix_with_names1 <- matrix(uniform.values, nrow=2, dimnames= list(rnames))
# filled by column
matrix_with_names1
```

```
       [,1]      [,2]      [,3]      [,4]      [,5]      [,6]       [,7]
a 0.7861149 0.6992523 0.9596138 0.1018046 0.9860037 0.6675401 0.05818433
b 0.2524456 0.1844608 0.9186834 0.1721917 0.8493961 0.9352102 0.61861583
        [,8]      [,9]     [,10]      [,11]     [,12]     [,13]     [,14]
a 0.17491846 0.5253132 0.4990452 0.01139965 0.7755985 0.2917067 0.5684972
b 0.03767539 0.2821842 0.6338251 0.60785656 0.9239712 0.7890762 0.7784351
      [,15]     [,16]     [,17]     [,18]     [,19]     [,20]     [,21]
a 0.7132325 0.9347099 0.7450602 0.8690747 0.2163319 0.3351660 0.6528394
b 0.6690487 0.5064602 0.8383526 0.1931117 0.6504235 0.5076559 0.9655767
       [,22]     [,23]     [,24]     [,25]
a 0.51466067 0.1510165 0.1029605 0.4102254
b 0.06165677 0.6355659 0.7726943 0.8702334
```

```
rnames <- c("a", "b")
matrix_with_names2 <- matrix(uniform.values, nrow=2, dimnames= list(rnames), byrow = TRUE)
# filled by raw
matrix_with_names2
```

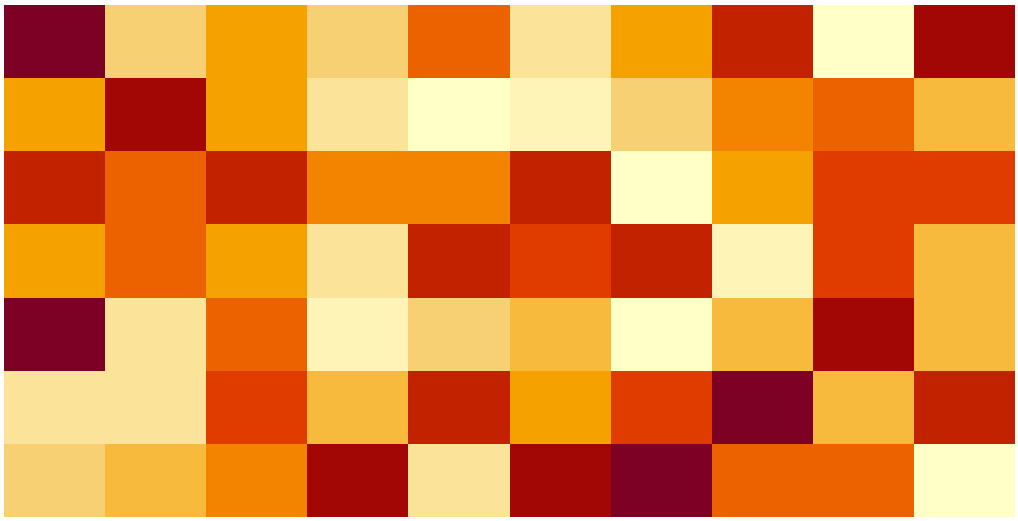
```
       [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
a 0.7861149 0.2524456 0.6992523 0.1844608 0.9596138 0.9186834 0.1018046
b 0.7890762 0.5684972 0.7784351 0.7132325 0.6690487 0.9347099 0.5064602
       [,8]      [,9]     [,10]     [,11]     [,12]      [,13]     [,14]
a 0.1721917 0.9860037 0.8493961 0.6675401 0.9352102 0.05818433 0.6186158
b 0.7450602 0.8383526 0.8690747 0.1931117 0.2163319 0.65042346 0.3351660
      [,15]      [,16]     [,17]     [,18]      [,19]     [,20]      [,21]
a 0.1749185 0.03767539 0.5253132 0.2821842 0.49904520 0.6338251 0.01139965
b 0.5076559 0.65283937 0.9655767 0.5146607 0.06165677 0.1510165 0.63556589
      [,22]     [,23]     [,24]     [,25]
a 0.6078566 0.7755985 0.9239712 0.2917067
b 0.1029605 0.7726943 0.4102254 0.8702334
```

Section 3.2 - Question 3

```
set.seed(1)
values <- runif(70)
m <- matrix(values, 10, 7)
m
```

```
            [,1]      [,2]       [,3]      [,4]      [,5]       [,6]       [,7]
 [1,] 0.26550866 0.2059746 0.93470523 0.4820801 0.8209463 0.47761962 0.91287592
 [2,] 0.37212390 0.1765568 0.21214252 0.5995658 0.6470602 0.86120948 0.29360337
 [3,] 0.57285336 0.6870228 0.65167377 0.4935413 0.7829328 0.43809711 0.45906573
 [4,] 0.90820779 0.3841037 0.12555510 0.1862176 0.5530363 0.24479728 0.33239467
 [5,] 0.20168193 0.7698414 0.26722067 0.8273733 0.5297196 0.07067905 0.65087047
 [6,] 0.89838968 0.4976992 0.38611409 0.6684667 0.7893562 0.09946616 0.25801678
 [7,] 0.94467527 0.7176185 0.01339033 0.7942399 0.0233312 0.31627171 0.47854525
 [8,] 0.66079779 0.9919061 0.38238796 0.1079436 0.4772301 0.51863426 0.76631067
 [9,] 0.62911404 0.3800352 0.86969085 0.7237109 0.7323137 0.66200508 0.08424691
[10,] 0.06178627 0.7774452 0.34034900 0.4112744 0.6927316 0.40683019 0.87532133
```

```
image(m, useRaster=TRUE, axes=FALSE)
```



Look at this visualization carefully. The **x axis shows the number of rows** and the **y axis shows the number of columns**. This commands fills the colour according to the value of the cells in the matrix. The color intensity is high for the higher values and the color intensity is low for the lower values.

image() command starts to fill the colours from bottom left cell and finishes at top right cell. Therefore, if needed one can rotate the image of the matrix by 90 degrees clockwise direction in order to match with the original matrix.

Part a

This is how we asign the values 0 and 1 to the values in the cell.

```
m[ m < 0.5] <- 0 #Assigning 0 to the values less than 0.5
m[m >= 0.5] <- 1 #Assigning 1 to the values which are greater or equal to 0.5
```

The matrix will be as follows.

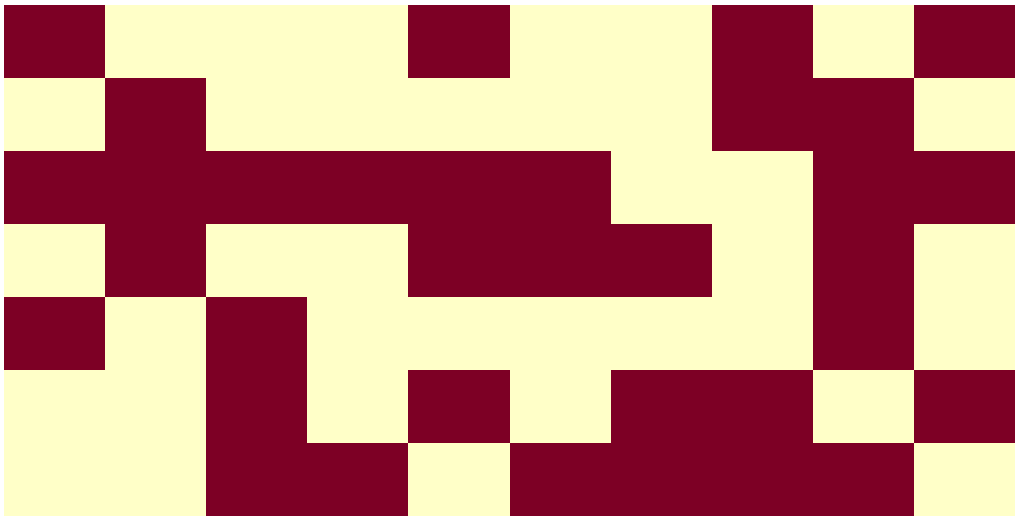
```
m
```

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

```
 [6,]    1    0    0    1    1    0    0
 [7,]    1    1    0    1    0    0    0
 [8,]    1    1    0    0    0    1    1
 [9,]    1    0    1    1    1    1    0
[10,]    0    1    0    0    1    0    1
```

Then we can visualize it as shown below.

```
image(m, useRaster=TRUE, axes=FALSE)
```



We can see that for the cells with 1 a higher intense colour is filled and for the cells with 0 a lower intense colour is filled.

Part b

We can use the sequence command to get the values in the matrix. And then we can fill those values by the columns in order to get the needed matrix. We can generate a sequence starting with 10 and finishing with 100 with increments of 10.

```
values<-seq(10,1000,by=10)
ymat<-matrix(values,nrow = 10,ncol = 10)
ymat
```

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

```
 [1,]   10  110  210  310  410  510  610  710  810   910
 [2,]   20  120  220  320  420  520  620  720  820   920
 [3,]   30  130  230  330  430  530  630  730  830   930
 [4,]   40  140  240  340  440  540  640  740  840   940
 [5,]   50  150  250  350  450  550  650  750  850   950
 [6,]   60  160  260  360  460  560  660  760  860   960
 [7,]   70  170  270  370  470  570  670  770  870   970
 [8,]   80  180  280  380  480  580  680  780  880   980
 [9,]   90  190  290  390  490  590  690  790  890   990
[10,]  100  200  300  400  500  600  700  800  900  1000
```

Now we need to convert all the values in the **even numbered indexes to 0**. Also, we need to convert all the values in the **odd numbered indexes to 1**. We convert the values in the matrix as shown below.

```
ymat[(1:length(ymat))%%2==0]<-0
ymat[(1:length(ymat))%%2==1]<-1
ymat
```

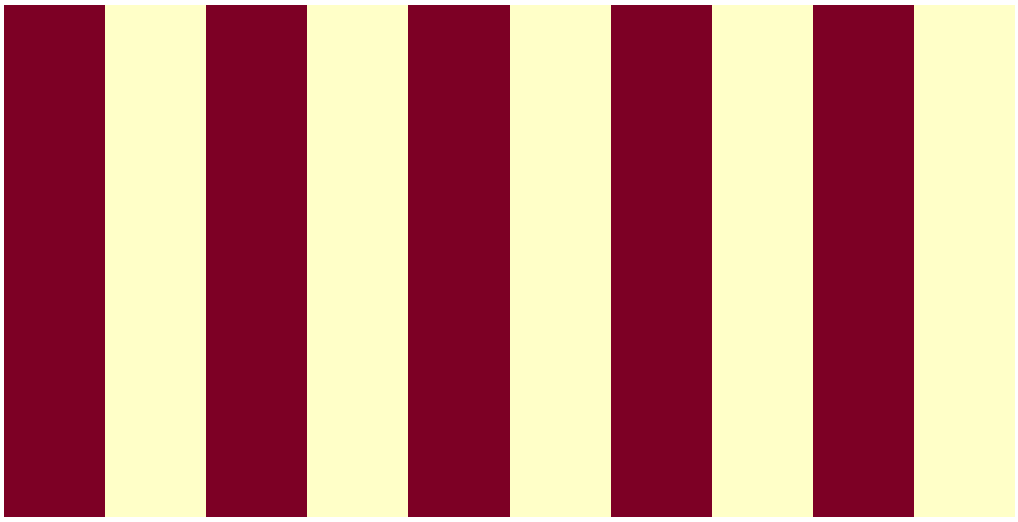
```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    1    1    1    1    1    1    1    1     1
 [2,]    0    0    0    0    0    0    0    0    0     0
 [3,]    1    1    1    1    1    1    1    1    1     1
 [4,]    0    0    0    0    0    0    0    0    0     0
 [5,]    1    1    1    1    1    1    1    1    1     1
 [6,]    0    0    0    0    0    0    0    0    0     0
 [7,]    1    1    1    1    1    1    1    1    1     1
 [8,]    0    0    0    0    0    0    0    0    0     0
 [9,]    1    1    1    1    1    1    1    1    1     1
[10,]    0    0    0    0    0    0    0    0    0     0
```

Using the length command we get the total number of elements is the matrix. In here it is 100. Then by using that we generate a sequence of numbers from 1 to 100. Then we extract the positions which are even and assign the values in those positions as zero. And we extract the positions which are odd and assign the values in those positions as 1.

Part c

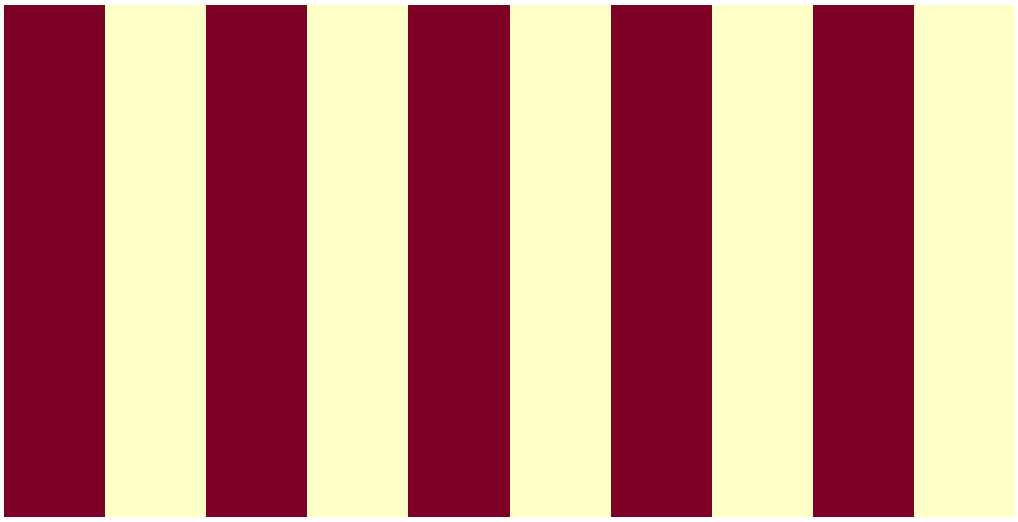
Visualizing the updated ymat.

```
image(ymat, useRaster=TRUE, axes=FALSE)
```



NOTE

```
ymat <- ymat/10
ymat <- ymat %% 2
image(ymat, useRaster=TRUE, axes=FALSE)
```



3.5

```
# Method 1
data(mtcars) # Loads specified data sets, or list the available data sets

# Method 2
data("mtcars") # Loads specified data sets, or list the available data sets

is.data.frame(mtcars) # To check if it is a data frame
```

```
[1] TRUE
```

```
str(mtcars) # Structure of the data set
```

```
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
```

```
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
# The first or last parts of a vector, matrix, table, data frame or function
head(mtcars) # Default it shows 6 rows
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
head(mtcars, 3) # To extract only first 3 rows
```

```
               mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

```
# The first or last parts of a vector, matrix, table, data frame or function
tail(mtcars) # Default it shows 6 rows
```

```
                mpg cyl  disp  hp drat    wt qsec vs am gear carb
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

```
tail(mtcars, 3) # To extract only last 3 rows
```

```
               mpg cyl disp  hp drat   wt qsec vs am gear carb
Ferrari Dino  19.7   6  145 175 3.62 2.77 15.5  0  1    5    6
Maserati Bora 15.0   8  301 335 3.54 3.57 14.6  0  1    5    8
Volvo 142E    21.4   4  121 109 4.11 2.78 18.6  1  1    4    2
```

```
names(mtcars) # To get or set name of an object (i:e: vector, matrix, or data frame)
```

```
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
[11] "carb"
```

```
# To retrieve or set the column names of a matrix-like object (i:e: matrix, or data frame)
colnames(mtcars)
```

```
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
[11] "carb"
```

```
# To get or set the length of vectors (including lists) and factors, and of any other R object
length(mtcars)  # To get number of rows in the data frame
```

```
[1] 11
```

```
dim(mtcars) # To retrieve or set the dimension of an object
```

```
[1] 32 11
```

```
nrow(mtcars) # To get the number of rows
```

```
[1] 32
```

```
ncol(mtcars) # To get the number of columns
```

```
[1] 11
```

2.
3.

3. Write an R code to extract column names and row names.

```
# To retrieve or set the column names of a matrix-like object (i:e: matrix, or data frame)
colnames(mtcars)
```

```
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
[11] "carb"
```

```
# To retrieve or set the row names of a matrix-like object (i:e: matrix, or data frame)
rownames(mtcars)
```

```
 [1] "Mazda RX4"           "Mazda RX4 Wag"       "Datsun 710"
 [4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"
 [7] "Duster 360"          "Merc 240D"           "Merc 230"
[10] "Merc 280"            "Merc 280C"           "Merc 450SE"
[13] "Merc 450SL"          "Merc 450SLC"         "Cadillac Fleetwood"
[16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 128"
[19] "Honda Civic"         "Toyota Corolla"      "Toyota Corona"
[22] "Dodge Challenger"    "AMC Javelin"         "Camaro Z28"
[25] "Pontiac Firebird"    "Fiat X1-9"           "Porsche 914-2"
[28] "Lotus Europa"        "Ford Pantera L"      "Ferrari Dino"
[31] "Maserati Bora"       "Volvo 142E"
```

4. Extract and display the column corresponding to the number of cylinders.

Method 1

```r
No_of_cylinders <- mtcars[,"cyl"] # To extract elements from specific column(s) by name
No_of_cylinders # To display the column
```

```
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

Method 2

```r
No_of_cylinders <- mtcars[,2] # To extract elements from specific row(s) and/or column(s) by index
No_of_cylinders # To display the column
```

```
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

Method 3

```r
No_of_cylinders <- mtcars$cyl # To extract elements from specific column
No_of_cylinders # To display the column
```

```
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

Method 4

```r
# To attach the data frame to the R search path
attach(mtcars)
# This means that the database is searched by R when evaluating a variable,
# so objects in the database can be accessed by simply giving their names.
No_of_cylinders <- cyl # To extract elements from specific column
No_of_cylinders
```

```
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

5. Extract and display the observations of cars with 4 cylinders AND 4 gears.

Method 1

```r
# To return row index which the logical expression is TRUE
id <- which(mtcars$cyl == 4 & mtcars$gear == 4)

# To extract element from specific row
mtcars[id,]
```

```
               mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
```

```
Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Method 2

```
# To return row index which the logical expression is TRUE
id_new <- which(cyl == 4 & gear == 4)

# To extract element from specific row
mtcars[id_new,]
```

```
               mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Method 3

```
# To return row index which the logical expression is TRUE
id_n <- which(mtcars[,2] == 4 & mtcars[,1] == 4)

# To extract element from specific row
mtcars[id_n,]
```

```
 [1] mpg  cyl  disp hp   drat wt   qsec vs   am   gear carb
<0 rows> (or 0-length row.names)
```

6. What is the maximum mpg?

Method 1

```
max(mtcars$mpg) # To get the the maximum value/ highest value from the vector/column
```

```
[1] 33.9
```

```
is.vector(mtcars$mpg) # To check if it is a vector
```

```
[1] TRUE
```

Method 2

```
max(mpg) # To get the the maximum value/ highest value from the vector/column
```

```
[1] 33.9
```

```
is.vector(mpg) # To check if it is a vector
```

```
[1] TRUE
```

Method 3

```
max(mtcars[,"mpg"]) # To get the the maximum value/ highest value from the vector/column
```

```
[1] 33.9
```

```
is.vector(mtcars[,"mpg"]) # To check if it is a vector
```

```
[1] TRUE
```

Method 4

```
max(mtcars[,1]) # To get the the maximum value/ highest value from the vector/column
```

```
[1] 33.9
```

```
is.vector(mtcars[,1]) # To check if it is a vector
```

```
[1] TRUE
```

7. Which car has the maximum mpg?

Method 1

```
Name_of_car <- rownames(mtcars) # To extract row names of the data frame
is.vector(Name_of_car) # To check if it is a vector
```

```
[1] TRUE
```

```
mtcars$car_name <- Name_of_car # Assign vector to a column of the data frame
```

Method 1

```r
# To return row index which has the highest value from the vector/column
which.max(mtcars$mpg)
```

```
[1] 20
```

```r
# To extract element from specific row and column
mtcars[which.max(mtcars$mpg) ,"car_name"]
```

```
[1] "Toyota Corolla"
```

Method 2

```r
# To return row index which has the highest value from the vector/column
which.max(mpg)
```

```
[1] 20
```

```r
# To extract element from specific row and column
mtcars[which.max(mpg) ,"car_name"]
```

```
[1] "Toyota Corolla"
```

Method 3

```r
# To return row index which has the highest value from the vector/column
id_mpg <- which.max(mpg)

# To extract element from specific row and column
mtcars[id_mpg ,"car_name"]
```

```
[1] "Toyota Corolla"
```

Method 4

```r
max(mtcars$mpg) # To get the the maximum value/ highest value from the vector
```

```
[1] 33.9
```

```r
# If equal return as TRUE
# If not return as FALSE
# Check whether each value of the vector and return whether the value is equal to
# the maximum value or not
mtcars$mpg == max(mtcars$mpg)
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
[25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
# To return row index which the logical expression is TRUE
which(mtcars$mpg == max(mtcars$mpg))
```

```
[1] 20
```

```
mtcars[which(mtcars$mpg == max(mtcars$mpg)),"car_name"] # To extract element from specific row
```

```
[1] "Toyota Corolla"
```

8. Compute suitable summary statistics for each column.

```
summary(mtcars) # Descriptive statistics for each field - (min, 1st Q, Median, Mean, 3rd Q, max)
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb         car_name        
 Min.   :0.0000   Min.   :3.000   Min.   :1.000   Length:32         
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000   Class :character  
 Median :0.0000   Median :4.000   Median :2.000   Mode  :character  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812                     
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000                     
 Max.   :1.0000   Max.   :5.000   Max.   :8.000                     
```

```
# Method 1
summary(mtcars$mpg) # Descriptive statistics of a specific field
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  10.40   15.43   19.20   20.09   22.80   33.90 
```

```
# Method 2
summary(mtcars[,1]) # Descriptive statistics of a specific field
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  10.40   15.43   19.20   20.09   22.80   33.90 
```

```r
# Method 3
summary(mpg) # Descriptive statistics of a specific field
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.43   19.20   20.09   22.80   33.90
```