Lab1 - Esther Xuanpei Ouyang

Esther Xuanpei Ouyang

1/22/2018

1) Basic Vector and Matrix manipulations in R

```
x = 1:9

matrix(data = x, nrow = 3, ncol = 3, byrow = FALSE)
```

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

```
matrix(data = x, nrow = 3, ncol = 3, byrow = TRUE)
```

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

```
diag(x = 1, 5, 5)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

```
a1 <- c(2, 3, 6, 7, 10)
a2 <- c(1.88, 2.05, 1.70, 1.60, 1.78)
a3 <- c(80, 90, 70, 50, 75)

A = cbind(a1, a2, a3)
A
```

```
##      a1   a2 a3
## [1,]  2 1.88 80
## [2,]  3 2.05 90
## [3,]  6 1.70 70
## [4,]  7 1.60 50
## [5,] 10 1.78 75
```

```
b1 <- c(1, 4, 5, 8, 9)
b2 <- c(1.22, 1.05, 3.60, 0.40, 2.54)
b3 <- c(20, 40, 30, 80, 100)

B = rbind(b1, b2, b3)
B
```

```
##     [,1]  [,2] [,3] [,4]   [,5]
## b1  1.00  4.00  5.0  8.0   9.00
```

```
## b2  1.22  1.05  3.6  0.4   2.54
## b3 20.00 40.00 30.0 80.0 100.00
```

```
A%*%B
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 1604.294 3209.974 2416.768 6416.752 8022.775
## [2,] 1805.501 3614.153 2722.380 7224.820 9032.207
## [3,] 1408.074 2825.785 2136.120 5648.680 7058.318
## [4,] 1008.952 2029.680 1540.760 4056.640 5067.064
## [5,] 1512.172 3041.869 2306.408 6080.712 7594.521
```

```
B%*%A
```

```
##         a1       a2      a3
## b1  190.00  47.4000  1865.0
## b2   55.39  15.7273   654.6
## b3 1900.00 476.6000 18800.0
```

```
t(A)%*%t(B)
```

```
##        b1       b2      b3
## a1  190.0  55.3900  1900.0
## a2   47.4  15.7273   476.6
## a3 1865.0 654.6000 18800.0
```

```
t(B)%*%t(A)
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 1604.294 1805.501 1408.074 1008.952 1512.172
## [2,] 3209.974 3614.153 2825.785 2029.680 3041.869
## [3,] 2416.768 2722.380 2136.120 1540.760 2306.408
## [4,] 6416.752 7224.820 5648.680 4056.640 6080.712
## [5,] 8022.775 9032.207 7058.318 5067.064 7594.521
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
selected_iris = cbind(iris[,1],iris[,2],iris[,3],iris[,4])
parameters = as.matrix(c(1,2,3,4),4,1)
lincombo = selected_iris%*%parameters
lincombo
```

```
##        [,1]
##   [1,] 17.1
##   [2,] 15.9
##   [3,] 15.8
##   [4,] 16.1
##   [5,] 17.2
##   [6,] 19.9
##   [7,] 16.8
##   [8,] 17.1
```

```
##   [9,] 15.2
##  [10,] 16.0
##  [11,] 18.1
##  [12,] 17.2
##  [13,] 15.4
##  [14,] 14.0
##  [15,] 18.2
##  [16,] 20.6
##  [17,] 18.7
##  [18,] 17.5
##  [19,] 19.6
##  [20,] 18.4
##  [21,] 18.1
##  [22,] 18.6
##  [23,] 15.6
##  [24,] 18.8
##  [25,] 18.1
##  [26,] 16.6
##  [27,] 18.2
##  [28,] 17.5
##  [29,] 17.0
##  [30,] 16.7
##  [31,] 16.6
##  [32,] 18.3
##  [33,] 18.3
##  [34,] 18.9
##  [35,] 16.4
##  [36,] 15.8
##  [37,] 17.2
##  [38,] 16.7
##  [39,] 15.1
##  [40,] 17.2
##  [41,] 17.1
##  [42,] 14.2
##  [43,] 15.5
##  [44,] 19.2
##  [45,] 20.0
##  [46,] 16.2
##  [47,] 18.3
##  [48,] 16.0
##  [49,] 18.0
##  [50,] 16.6
##  [51,] 33.1
##  [52,] 32.3
##  [53,] 33.8
##  [54,] 27.3
##  [55,] 31.9
##  [56,] 30.0
##  [57,] 33.4
##  [58,] 23.6
##  [59,] 31.4
##  [60,] 27.9
##  [61,] 23.5
##  [62,] 30.5
```

```
##  [63,] 26.4
##  [64,] 31.6
##  [65,] 27.4
##  [66,] 31.7
##  [67,] 31.1
##  [68,] 27.5
##  [69,] 30.1
##  [70,] 26.7
##  [71,] 33.9
##  [72,] 28.9
##  [73,] 32.0
##  [74,] 30.6
##  [75,] 30.3
##  [76,] 31.4
##  [77,] 32.4
##  [78,] 34.5
##  [79,] 31.3
##  [80,] 25.4
##  [81,] 26.1
##  [82,] 25.4
##  [83,] 27.7
##  [84,] 33.1
##  [85,] 30.9
##  [86,] 32.7
##  [87,] 33.0
##  [88,] 29.3
##  [89,] 29.1
##  [90,] 27.7
##  [91,] 28.7
##  [92,] 31.5
##  [93,] 27.8
##  [94,] 23.5
##  [95,] 28.8
##  [96,] 29.1
##  [97,] 29.3
##  [98,] 30.1
##  [99,] 23.5
## [100,] 28.8
## [101,] 40.9
## [102,] 34.1
## [103,] 39.2
## [104,] 36.1
## [105,] 38.7
## [106,] 41.8
## [107,] 30.2
## [108,] 39.2
## [109,] 36.3
## [110,] 42.7
## [111,] 36.2
## [112,] 35.3
## [113,] 37.7
## [114,] 33.7
## [115,] 36.3
## [116,] 37.9
```

```
## [117,] 36.2
## [118,] 44.2
## [119,] 42.8
## [120,] 31.4
## [121,] 39.6
## [122,] 33.9
## [123,] 41.4
## [124,] 33.6
## [125,] 38.8
## [126,] 38.8
## [127,] 33.4
## [128,] 34.0
## [129,] 37.2
## [130,] 37.0
## [131,] 38.9
## [132,] 42.7
## [133,] 37.6
## [134,] 33.2
## [135,] 33.7
## [136,] 41.2
## [137,] 39.5
## [138,] 36.3
## [139,] 33.6
## [140,] 37.7
## [141,] 39.3
## [142,] 37.6
## [143,] 34.1
## [144,] 40.1
## [145,] 40.4
## [146,] 37.5
## [147,] 33.9
## [148,] 36.1
## [149,] 38.4
## [150,] 34.4
```

```
v = 1:5
vnorm = function(x) {
  return(sqrt(t(x)%*%x))
}
vnorm(v)
```

```
##          [,1]
## [1,] 7.416198
```

```
unit_u = v / vnorm(v)
unit_u
```

```
## [1] 0.1348400 0.2696799 0.4045199 0.5393599 0.6741999
```

```
is_square = function(x) {
  dimen = dim(x)
  return (dimen[1] == dimen[2])
}

x = 1:8
test1 = matrix(x, 2, 4)
```

```
test1
```

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

```
is_square(test1)
```

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

```
y = 1:9
test2 = matrix(y, 3, 3)
test2
```

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

```
is_square(test2)
```

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

```
mtrace = function(mat) {
  if (!is_square(mat)) {
    return (0)
  } else {
    dimen = dim(mat)
    I = diag(1, dimen[1], dimen[2])
    return(sum(sum(mat*I)))
  }
}

test1 = matrix(1:9, 3, 3)
test2 = matrix(2:10, 3, 3)
test3 = test1 + test2
# Test 1
mtrace(test1) + mtrace(test2)
```

```
## [1] 33
```

```
mtrace(test3)
```

```
## [1] 33
```

```
# Test 2
test4 = 3 * test1
mtrace(test1) * 3
```

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

```
mtrace(test4)
```

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

```
# Test3
X = test1
Y = test2
XtY = t(X)%*%Y
XYt = X%*%t(Y)
```

```
YtX = t(Y)%*%X
YXt = Y%*%t(X)

mtrace(XtY)
```

```
## [1] 330
```

```
mtrace(XYt)
```

```
## [1] 330
```

```
mtrace(YtX)
```

```
## [1] 330
```

```
mtrace(YXt)
```

```
## [1] 330
```

Proof for Trace

1. $tr(A+B) = tr(A) + tr(B)$

Suppose A, B are both square matrix with dimention n x n. Let $C = A + B$, then from the definition of matrix addition, $c_{ij} = a_{ij} + b_{ij}$. By the definition of trace,

$$tr(A) + tr(B) = \sum_{i=1}^{n} a_{ii} + \sum_{i=1}^{n} b_{ii} = \sum_{i=1}^{n} a_{ii} + b_{ii} = \sum_{i=1}^{n} c_i i \text{ (definition of matrix addition)} = tr(C)$$

2. $tr(cA) = c \times tr(A)$, where c is a scalar. Suppose A, B are both square matrix with dimention n x n. Let $D = cA$, then from the definition of scalar matrix multiplication, $d_{ij} = c \times a_{ij}$. By the definition of trace,

$$tr(cA) = \sum_{i=1}^{n} c \times a_{ii} = c \times \sum_{i=1}^{n} a_{ii} \text{ (take c out of the summation form since c is a scalar)} = c \times tr(A)$$

3. $tr(X^T Y) = tr(XY^T) = tr(Y^T X) = tr(YX^T)$ Suppose X, Y are both square matrix with dimention n x n.

2) Transformation and Scaling Operations

```
head(mtcars)
```

```
##                   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
```

```
# Create a matrix M with variables mpg, disp, hp, drat, and wt.
M = mtcars[, c(1,3,4,5,6)]
M
```

```
##                      mpg  disp  hp drat    wt
## Mazda RX4           21.0 160.0 110 3.90 2.620
## Mazda RX4 Wag       21.0 160.0 110 3.90 2.875
## Datsun 710          22.8 108.0  93 3.85 2.320
## Hornet 4 Drive      21.4 258.0 110 3.08 3.215
## Hornet Sportabout   18.7 360.0 175 3.15 3.440
## Valiant             18.1 225.0 105 2.76 3.460
## Duster 360          14.3 360.0 245 3.21 3.570
## Merc 240D           24.4 146.7  62 3.69 3.190
## Merc 230            22.8 140.8  95 3.92 3.150
## Merc 280            19.2 167.6 123 3.92 3.440
## Merc 280C           17.8 167.6 123 3.92 3.440
## Merc 450SE          16.4 275.8 180 3.07 4.070
## Merc 450SL          17.3 275.8 180 3.07 3.730
## Merc 450SLC         15.2 275.8 180 3.07 3.780
## Cadillac Fleetwood  10.4 472.0 205 2.93 5.250
## Lincoln Continental 10.4 460.0 215 3.00 5.424
## Chrysler Imperial   14.7 440.0 230 3.23 5.345
## Fiat 128            32.4  78.7  66 4.08 2.200
## Honda Civic         30.4  75.7  52 4.93 1.615
## Toyota Corolla      33.9  71.1  65 4.22 1.835
## Toyota Corona       21.5 120.1  97 3.70 2.465
## Dodge Challenger    15.5 318.0 150 2.76 3.520
## AMC Javelin         15.2 304.0 150 3.15 3.435
## Camaro Z28          13.3 350.0 245 3.73 3.840
## Pontiac Firebird    19.2 400.0 175 3.08 3.845
## Fiat X1-9           27.3  79.0  66 4.08 1.935
## Porsche 914-2       26.0 120.3  91 4.43 2.140
## Lotus Europa        30.4  95.1 113 3.77 1.513
## Ford Pantera L      15.8 351.0 264 4.22 3.170
## Ferrari Dino        19.7 145.0 175 3.62 2.770
## Maserati Bora       15.0 301.0 335 3.54 3.570
## Volvo 142E          21.4 121.0 109 4.11 2.780
```

```
# Use apply() to compute the vector containing the means of the columns in M
apply(M, 2, mean)
```

```
##        mpg       disp         hp       drat         wt
##  20.090625 230.721875 146.687500   3.596563   3.217250
```

```
# Compute a matrix Mc of mean-centered data applying the function scale() on M (do NOT use the argument
Mc = scale(M, scale = FALSE)
Mc
```

```
##                       mpg        disp       hp       drat       wt
## Mazda RX4           0.909375  -70.721875 -36.6875  0.3034375 -0.59725
## Mazda RX4 Wag       0.909375  -70.721875 -36.6875  0.3034375 -0.34225
## Datsun 710          2.709375 -122.721875 -53.6875  0.2534375 -0.89725
## Hornet 4 Drive      1.309375   27.278125 -36.6875 -0.5165625 -0.00225
## Hornet Sportabout  -1.390625  129.278125  28.3125 -0.4465625  0.22275
## Valiant            -1.990625   -5.721875 -41.6875 -0.8365625  0.24275
## Duster 360         -5.790625  129.278125  98.3125 -0.3865625  0.35275
## Merc 240D           4.309375  -84.021875 -84.6875  0.0934375 -0.02725
## Merc 230            2.709375  -89.921875 -51.6875  0.3234375 -0.06725
## Merc 280           -0.890625  -63.121875 -23.6875  0.3234375  0.22275
## Merc 280C          -2.290625  -63.121875 -23.6875  0.3234375  0.22275
```

```
## Merc 450SE          -3.690625   45.078125  33.3125 -0.5265625  0.85275
## Merc 450SL          -2.790625   45.078125  33.3125 -0.5265625  0.51275
## Merc 450SLC         -4.890625   45.078125  33.3125 -0.5265625  0.56275
## Cadillac Fleetwood  -9.690625  241.278125  58.3125 -0.6665625  2.03275
## Lincoln Continental -9.690625  229.278125  68.3125 -0.5965625  2.20675
## Chrysler Imperial   -5.390625  209.278125  83.3125 -0.3665625  2.12775
## Fiat 128            12.309375 -152.021875 -80.6875  0.4834375 -1.01725
## Honda Civic         10.309375 -155.021875 -94.6875  1.3334375 -1.60225
## Toyota Corolla      13.809375 -159.621875 -81.6875  0.6234375 -1.38225
## Toyota Corona        1.409375 -110.621875 -49.6875  0.1034375 -0.75225
## Dodge Challenger    -4.590625   87.278125   3.3125 -0.8365625  0.30275
## AMC Javelin         -4.890625   73.278125   3.3125 -0.4465625  0.21775
## Camaro Z28          -6.790625  119.278125  98.3125  0.1334375  0.62275
## Pontiac Firebird    -0.890625  169.278125  28.3125 -0.5165625  0.62775
## Fiat X1-9            7.209375 -151.721875 -80.6875  0.4834375 -1.28225
## Porsche 914-2        5.909375 -110.421875 -55.6875  0.8334375 -1.07725
## Lotus Europa        10.309375 -135.621875 -33.6875  0.1734375 -1.70425
## Ford Pantera L      -4.290625  120.278125 117.3125  0.6234375 -0.04725
## Ferrari Dino        -0.390625  -85.721875  28.3125  0.0234375 -0.44725
## Maserati Bora       -5.090625   70.278125 188.3125 -0.0565625  0.35275
## Volvo 142E           1.309375 -109.721875 -37.6875  0.5134375 -0.43725
## attr(,"scaled:center")
##        mpg       disp         hp       drat         wt
##  20.090625 230.721875 146.687500   3.596563   3.217250
```

```
# Confirm that variables in Mc are mean-centered by calculating the vector of column-means
apply(Mc, 2, mean)
```

```
##           mpg          disp            hp          drat            wt
##  4.440892e-16 -1.199041e-14  0.000000e+00 -1.526557e-16  3.469447e-17
```

```
# Use the function sweep() to mean-center M by sweeping out the vector of column means.
M.mean = apply(M, 2, mean)
M.mean
```

```
##        mpg       disp         hp       drat         wt
##  20.090625 230.721875 146.687500   3.596563   3.217250
```

```
Msweep = sweep(M, 2, M.mean)
Msweep
```

```
##                          mpg        disp       hp       drat       wt
## Mazda RX4            0.909375  -70.721875 -36.6875  0.3034375 -0.59725
## Mazda RX4 Wag        0.909375  -70.721875 -36.6875  0.3034375 -0.34225
## Datsun 710           2.709375 -122.721875 -53.6875  0.2534375 -0.89725
## Hornet 4 Drive       1.309375   27.278125 -36.6875 -0.5165625 -0.00225
## Hornet Sportabout   -1.390625  129.278125  28.3125 -0.4465625  0.22275
## Valiant             -1.990625   -5.721875 -41.6875 -0.8365625  0.24275
## Duster 360          -5.790625  129.278125  98.3125 -0.3865625  0.35275
## Merc 240D            4.309375  -84.021875 -84.6875  0.0934375 -0.02725
## Merc 230             2.709375  -89.921875 -51.6875  0.3234375 -0.06725
## Merc 280            -0.890625  -63.121875 -23.6875  0.3234375  0.22275
## Merc 280C           -2.290625  -63.121875 -23.6875  0.3234375  0.22275
## Merc 450SE          -3.690625   45.078125  33.3125 -0.5265625  0.85275
## Merc 450SL          -2.790625   45.078125  33.3125 -0.5265625  0.51275
## Merc 450SLC         -4.890625   45.078125  33.3125 -0.5265625  0.56275
```

```
## Cadillac Fleetwood  -9.690625  241.278125  58.3125 -0.6665625  2.03275
## Lincoln Continental -9.690625  229.278125  68.3125 -0.5965625  2.20675
## Chrysler Imperial   -5.390625  209.278125  83.3125 -0.3665625  2.12775
## Fiat 128            12.309375 -152.021875 -80.6875  0.4834375 -1.01725
## Honda Civic         10.309375 -155.021875 -94.6875  1.3334375 -1.60225
## Toyota Corolla      13.809375 -159.621875 -81.6875  0.6234375 -1.38225
## Toyota Corona        1.409375 -110.621875 -49.6875  0.1034375 -0.75225
## Dodge Challenger    -4.590625   87.278125   3.3125 -0.8365625  0.30275
## AMC Javelin         -4.890625   73.278125   3.3125 -0.4465625  0.21775
## Camaro Z28          -6.790625  119.278125  98.3125  0.1334375  0.62275
## Pontiac Firebird    -0.890625  169.278125  28.3125 -0.5165625  0.62775
## Fiat X1-9            7.209375 -151.721875 -80.6875  0.4834375 -1.28225
## Porsche 914-2        5.909375 -110.421875 -55.6875  0.8334375 -1.07725
## Lotus Europa        10.309375 -135.621875 -33.6875  0.1734375 -1.70425
## Ford Pantera L      -4.290625  120.278125 117.3125  0.6234375 -0.04725
## Ferrari Dino        -0.390625  -85.721875  28.3125  0.0234375 -0.44725
## Maserati Bora       -5.090625   70.278125 188.3125 -0.0565625  0.35275
## Volvo 142E           1.309375 -109.721875 -37.6875  0.5134375 -0.43725
```

```
# Compare this result with Mc (you should get the same values).
all(Mc == Msweep)
```

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

```
# Compute a vector of column maxima from M.
Mmax = apply(M, 2, max)
Mmax
```

```
##     mpg    disp      hp    drat      wt
##  33.900 472.000 335.000   4.930   5.424
```

```
# Use sweep() to scale the columns of M by dividing by the column maxima.
sweep(M, 2, Mmax,"/")
```

```
##                           mpg      disp        hp      drat        wt
## Mazda RX4           0.6194690 0.3389831 0.3283582 0.7910751 0.4830383
## Mazda RX4 Wag       0.6194690 0.3389831 0.3283582 0.7910751 0.5300516
## Datsun 710          0.6725664 0.2288136 0.2776119 0.7809331 0.4277286
## Hornet 4 Drive      0.6312684 0.5466102 0.3283582 0.6247465 0.5927360
## Hornet Sportabout   0.5516224 0.7627119 0.5223881 0.6389452 0.6342183
## Valiant             0.5339233 0.4766949 0.3134328 0.5598377 0.6379056
## Duster 360          0.4218289 0.7627119 0.7313433 0.6511156 0.6581858
## Merc 240D           0.7197640 0.3108051 0.1850746 0.7484787 0.5881268
## Merc 230            0.6725664 0.2983051 0.2835821 0.7951318 0.5807522
## Merc 280            0.5663717 0.3550847 0.3671642 0.7951318 0.6342183
## Merc 280C           0.5250737 0.3550847 0.3671642 0.7951318 0.6342183
## Merc 450SE          0.4837758 0.5843220 0.5373134 0.6227181 0.7503687
## Merc 450SL          0.5103245 0.5843220 0.5373134 0.6227181 0.6876844
## Merc 450SLC         0.4483776 0.5843220 0.5373134 0.6227181 0.6969027
## Cadillac Fleetwood  0.3067847 1.0000000 0.6119403 0.5943205 0.9679204
## Lincoln Continental 0.3067847 0.9745763 0.6417910 0.6085193 1.0000000
## Chrysler Imperial   0.4336283 0.9322034 0.6865672 0.6551724 0.9854351
## Fiat 128            0.9557522 0.1667373 0.1970149 0.8275862 0.4056047
## Honda Civic         0.8967552 0.1603814 0.1552239 1.0000000 0.2977507
## Toyota Corolla      1.0000000 0.1506356 0.1940299 0.8559838 0.3383112
## Toyota Corona       0.6342183 0.2544492 0.2895522 0.7505071 0.4544617
```

```
## Dodge Challenger    0.4572271 0.6737288 0.4477612 0.5598377 0.6489676
## AMC Javelin         0.4483776 0.6440678 0.4477612 0.6389452 0.6332965
## Camaro Z28          0.3923304 0.7415254 0.7313433 0.7565923 0.7079646
## Pontiac Firebird    0.5663717 0.8474576 0.5223881 0.6247465 0.7088864
## Fiat X1-9           0.8053097 0.1673729 0.1970149 0.8275862 0.3567478
## Porsche 914-2       0.7669617 0.2548729 0.2716418 0.8985801 0.3945428
## Lotus Europa        0.8967552 0.2014831 0.3373134 0.7647059 0.2789454
## Ford Pantera L      0.4660767 0.7436441 0.7880597 0.8559838 0.5844395
## Ferrari Dino        0.5811209 0.3072034 0.5223881 0.7342799 0.5106932
## Maserati Bora       0.4424779 0.6377119 1.0000000 0.7180527 0.6581858
## Volvo 142E          0.6312684 0.2563559 0.3253731 0.8336714 0.5125369
```

```
# Compute a matrix in which all columns of M are scaled such that they have minimum = 0 and maximum = 1
Mmax = apply(M, 2, max)
Mmin = apply(M, 2, min)
Mrange = Mmax - Mmin
scaled_M = scale(M , center = Mmin, scale = Mrange)
scaled_M
```

```
##                           mpg       disp         hp       drat         wt
## Mazda RX4           0.4510638 0.22175106 0.20494700 0.52534562 0.28304781
## Mazda RX4 Wag       0.4510638 0.22175106 0.20494700 0.52534562 0.34824853
## Datsun 710          0.5276596 0.09204290 0.14487633 0.50230415 0.20634109
## Hornet 4 Drive      0.4680851 0.46620105 0.20494700 0.14746544 0.43518282
## Hornet Sportabout   0.3531915 0.72062859 0.43462898 0.17972350 0.49271286
## Valiant             0.3276596 0.38388626 0.18727915 0.00000000 0.49782664
## Duster 360          0.1659574 0.72062859 0.68197880 0.20737327 0.52595244
## Merc 240D           0.5957447 0.18857570 0.03533569 0.42857143 0.42879059
## Merc 230            0.5276596 0.17385882 0.15194346 0.53456221 0.41856303
## Merc 280            0.3744681 0.24070841 0.25088339 0.53456221 0.49271286
## Merc 280C           0.3148936 0.24070841 0.25088339 0.53456221 0.49271286
## Merc 450SE          0.2553191 0.51060115 0.45229682 0.14285714 0.65379698
## Merc 450SL          0.2936170 0.51060115 0.45229682 0.14285714 0.56686269
## Merc 450SLC         0.2042553 0.51060115 0.45229682 0.14285714 0.57964715
## Cadillac Fleetwood  0.0000000 1.00000000 0.54063604 0.07834101 0.95551010
## Lincoln Continental 0.0000000 0.97006735 0.57597173 0.11059908 1.00000000
## Chrysler Imperial   0.1829787 0.92017960 0.62897527 0.21658986 0.97980056
## Fiat 128            0.9361702 0.01895735 0.04946996 0.60829493 0.17565840
## Honda Civic         0.8510638 0.01147418 0.00000000 1.00000000 0.02608029
## Toyota Corolla      1.0000000 0.00000000 0.04593640 0.67281106 0.08233188
## Toyota Corona       0.4723404 0.12222499 0.15901060 0.43317972 0.24341601
## Dodge Challenger    0.2170213 0.61586431 0.34628975 0.00000000 0.51316799
## AMC Javelin         0.2042553 0.58094288 0.34628975 0.17972350 0.49143442
## Camaro Z28          0.1234043 0.69568471 0.68197880 0.44700461 0.59498849
## Pontiac Firebird    0.3744681 0.82040409 0.43462898 0.14746544 0.59626694
## Fiat X1-9           0.7191489 0.01970566 0.04946996 0.60829493 0.10790079
## Porsche 914-2       0.6638298 0.12272387 0.13780919 0.76958525 0.16031705
## Lotus Europa        0.8510638 0.05986530 0.21554770 0.46543779 0.00000000
## Ford Pantera L      0.2297872 0.69817910 0.74911661 0.67281106 0.42367681
## Ferrari Dino        0.3957447 0.18433525 0.43462898 0.39631336 0.32140118
## Maserati Bora       0.1957447 0.57345972 1.00000000 0.35944700 0.52595244
## Volvo 142E          0.4680851 0.12446994 0.20141343 0.62211982 0.32395807
## attr(,"scaled:center")
##    mpg   disp     hp   drat     wt
## 10.400 71.100 52.000  2.760  1.513
```

```
## attr(,"scaled:scale")
##     mpg    disp      hp    drat      wt
##  23.500 400.900 283.000   2.170   3.911
```

```
apply(scaled_M, 2, min)
```

```
##  mpg disp   hp drat   wt
##    0    0    0    0    0
```

```
apply(scaled_M, 2, max)
```

```
##  mpg disp   hp drat   wt
##    1    1    1    1    1
```

```
# Without using the function cov(), compute the sample covariance matrix of the variables in M: mpg, di
n = dim(mtcars)[1]
Mcov = (t(Mc)%*%Mc)/(n-1)
Mcov
```

```
##              mpg        disp         hp        drat          wt
## mpg    36.324103  -633.09721 -320.73206   2.1950635  -5.1166847
## disp -633.097208 15360.79983 6721.15867 -47.0640192 107.6842040
## hp   -320.732056  6721.15867 4700.86694 -16.4511089  44.1926613
## drat    2.195064   -47.06402  -16.45111   0.2858814  -0.3727207
## wt     -5.116685   107.68420   44.19266  -0.3727207   0.9573790
```

```
# Without using the function cor(), compute the correlation matrix of the variables in M: mpg, disp, hp
standardize_M = scale(M) # variance = 1
Mcor = (t(standardize_M)%*%standardize_M)/(n-1)
Mcor
```

```
##             mpg       disp         hp       drat         wt
## mpg   1.0000000 -0.8475514 -0.7761684  0.6811719 -0.8676594
## disp -0.8475514  1.0000000  0.7909486 -0.7102139  0.8879799
## hp   -0.7761684  0.7909486  1.0000000 -0.4487591  0.6587479
## drat  0.6811719 -0.7102139 -0.4487591  1.0000000 -0.7124406
## wt   -0.8676594  0.8879799  0.6587479 -0.7124406  1.0000000
```

```
cyl = factor(mtcars$cyl)

dummify = function(char_vector, all) {
  n_class = nlevels(char_vector)
  if(!all) {
    res = matrix(0,1,n_class-1)
  } else {
    res = matrix(0,1,n_class)
  }
  dummy_mat = diag(1, n_class, n_class)
  if (!all) {
    dummy_mat = dummy_mat[, -1]
  }
  class_level = c(levels(char_vector))
  for (class in char_vector) {
    class_i = match(class, class_level)
    res = rbind(res, dummy_mat[class_i, ])
  }
  return(res[-1,])
}
```

```r
dummify(cyl, all = TRUE)
```

```
##       [,1] [,2] [,3]
##  [1,]    0    1    0
##  [2,]    0    1    0
##  [3,]    1    0    0
##  [4,]    0    1    0
##  [5,]    0    0    1
##  [6,]    0    1    0
##  [7,]    0    0    1
##  [8,]    1    0    0
##  [9,]    1    0    0
## [10,]    0    1    0
## [11,]    0    1    0
## [12,]    0    0    1
## [13,]    0    0    1
## [14,]    0    0    1
## [15,]    0    0    1
## [16,]    0    0    1
## [17,]    0    0    1
## [18,]    1    0    0
## [19,]    1    0    0
## [20,]    1    0    0
## [21,]    1    0    0
## [22,]    0    0    1
## [23,]    0    0    1
## [24,]    0    0    1
## [25,]    0    0    1
## [26,]    1    0    0
## [27,]    1    0    0
## [28,]    1    0    0
## [29,]    0    0    1
## [30,]    0    1    0
## [31,]    0    0    1
## [32,]    1    0    0
```

```r
dummify(cyl, all = FALSE)
```

```
##       [,1] [,2]
##  [1,]    1    0
##  [2,]    1    0
##  [3,]    0    0
##  [4,]    1    0
##  [5,]    0    1
##  [6,]    1    0
##  [7,]    0    1
##  [8,]    0    0
##  [9,]    0    0
## [10,]    1    0
## [11,]    1    0
## [12,]    0    1
## [13,]    0    1
## [14,]    0    1
## [15,]    0    1
```

```
## [16,]    0    1
## [17,]    0    1
## [18,]    0    0
## [19,]    0    0
## [20,]    0    0
## [21,]    0    0
## [22,]    0    1
## [23,]    0    1
## [24,]    0    1
## [25,]    0    1
## [26,]    0    0
## [27,]    0    0
## [28,]    0    0
## [29,]    0    1
## [30,]    1    0
## [31,]    0    1
## [32,]    0    0
```

```
gear = factor(mtcars$gear)

crosstable = function(vec1, vec2) {
  dummy1 = dummify(vec1, all = TRUE)
  dummy2 = dummify(vec2, all = TRUE)
  return(t(dummy1)%*%dummy2)
}

xtb <- crosstable(cyl, gear)
xtb
```

```
##      [,1] [,2] [,3]
## [1,]    1    8    2
## [2,]    2    4    1
## [3,]   12    0    2
```