Lab 01: Basics of Matrix Algebra in R

Donggyun Kim

1/24/2018

1) Basic Vector and Matrix manipulation in R

A vector **x**

```
x <- 1:9
```

Using **x**, create following matrices

```
matrix(x, nrow = 3, ncol = 3)
```

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

```
matrix(x, nrow = 3, ncol = 3, byrow = 1)
```

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

Create identity matrix

```
diag(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
```

Three vectors, $\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}$

```
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)
```

Use `cbind()` to create a matrix, $A_{5\times3}$

```
A <- cbind(a1, a2, a3)
rownames(A) <- 1:5
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
```

Three vectors, $\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}$

```r
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)
```

Use `rbind()` to create a matrix, $B_{3\times5}$

```r
B <- rbind(b1, b2, b3)
colnames(B) <- 1:5
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
```

Compute matrix products using `%*%` operator

AB

```r
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
```

BA

```r
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
```

$\mathbf{A}^\top\mathbf{B}^\top$

```r
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
```

$\mathbf{B}^\top\mathbf{A}^\top$

```r
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
```

Obtain a linear combination via a matrix multiplication

$$1 \times \text{Sepal.Length} + 2 \times \text{Sepal.Width} + 3 \times \text{Petal.Length} + 4 \times \text{Petal.Width}$$

```r
M_iris <- as.matrix(iris[1:4])
N <- matrix(rep(1:4, nrow(iris)), nrow = 4)
```

```r
M <- M_iris %*% 1:4
rownames(M) <- 1:150
colnames(M) <- paste(1:4, colnames(M_iris), collapse = " + ")
head(M, 10)
```

```
##    1 Sepal.Length + 2 Sepal.Width + 3 Petal.Length + 4 Petal.Width
## 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
```

Write a function `vnorm()` that computes the length of a vector:

$$||\mathbf{v}|| = \sqrt{\mathbf{v}^\top \mathbf{v}}$$

```r
vnorm <- function(v) {
  as.numeric(sqrt(t(v) %*% v))
}
```

Given a vector, $\mathbf{v}$

```r
v <- 1:5
```

Find a unit vector of $\mathbf{v}$

```r
u <- v / vnorm(v)
u
```

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

The length of $\mathbf{u}$ is

```r
vnorm(u)
```

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

Write a function `is_square()` to check whether the provided matrix is a square matrix

```r
is_square <- function(X) {
  if (!is.matrix(X)) {
    stop("The input must be a matrix")
  }

  if (nrow(X) == ncol(X)) {
    TRUE
  } else {
    FALSE
  }
}
```

Test if the function works

```r
D <- diag(4)
D
```

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

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

```r
is_square(D)
```

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

```r
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
```

```r
is_square(A)
```

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

Write a function `mtrace()` to compute the trace of a square matrix

```r
mtrace <- function(X) {
  if (!is.matrix(X) | !is_square(X)) {
    stop("The input must be a suare matrix")
  }

  sum(diag(X))
}
```

Given two square matrices **A** and **B**, verify that `mtrace()` is linear mapping:

```r
A <- matrix(1:9, ncol = 3, nrow = 3)
A
```

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

```r
B <- diag(3)
B
```

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

- $\mathrm{tr}(A + B) = \mathrm{tr}(A) + \mathrm{tr}(B)$

```r
mtrace(A + B)
```

```
## [1] 18
```

```r
mtrace(A) + mtrace(B)
```

```
## [1] 18
```

- $\mathrm{tr}(cA) = c \times \mathrm{tr}(A)$

```
mtrace(3 * A)
```

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

```
3 * mtrace(A)
```

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

Trace of products
Given two matrices $\mathbf{X}$ and $\mathbf{Y}$, verify that

$$\operatorname{tr}(\mathbf{X}^\top\mathbf{Y}) = \operatorname{tr}(\mathbf{X}\mathbf{Y}^\top) = \operatorname{tr}(\mathbf{Y}^\top\mathbf{X}) = \operatorname{tr}(\mathbf{Y}\mathbf{X}^\top)$$

- Prove that trace function is commutative

  $$\operatorname{tr}(\mathbf{AB}) = \operatorname{tr}(\mathbf{BA})$$

  Because $\mathbf{AB}_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}$ for matrices $\mathbf{A_{p\times n}}$ and $\mathbf{B_{n\times p}}$,
  $\operatorname{tr}(\mathbf{AB}) = \sum_{i=1}^{p}\sum_{k=1}^{n} a_{ik}b_{ki}$ and $\operatorname{tr}(\mathbf{BA}) = \sum_{k=1}^{n}\sum_{i=1}^{p} b_{ki}a_{ik}$.

  $$\operatorname{tr}(\mathbf{AB}) = \sum_{i=1}^{p}\sum_{k=1}^{n} a_{ik}b_{ki} = \sum_{i=1}^{p}\sum_{k=1}^{n} b_{ki}a_{ik} = \sum_{k=1}^{n}\sum_{i=1}^{p} b_{ki}a_{ik} = \operatorname{tr}(\mathbf{BA})$$

- Prove that the trace of a square mtrix, $\mathbf{A}_{n\times n}$ equals the trace of transpose of the square matrix

  $$\operatorname{tr}(\mathbf{A}) = \operatorname{tr}(\mathbf{A}^\top)$$

  Because $\mathbf{A}_{ij} = a_{ij}$ and $\mathbf{A}^\top{}_{ij} = a_{ji}$,
  $\operatorname{tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}$ and $\operatorname{tr}(\mathbf{A}^\top) = \sum_{j=1}^{n} a_{jj}$

  $$\operatorname{tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii} = \sum_{j=1}^{n} a_{jj} = \operatorname{tr}(\mathbf{A}^\top)$$

It is proved that $\operatorname{tr}(\mathbf{X}\mathbf{Y}^\top) = \operatorname{tr}(\mathbf{Y}^\top\mathbf{X})$

Let $\mathbf{P} = \mathbf{X}^\top\mathbf{Y}$. This implies $\mathbf{P}^\top = (\mathbf{X}^\top\mathbf{Y})^\top = \mathbf{Y}^\top\mathbf{X}$
Also, $\operatorname{tr}(\mathbf{P}) = \operatorname{tr}(\mathbf{X}^\top\mathbf{Y}) = \operatorname{tr}(\mathbf{Y}^\top\mathbf{X}) = \operatorname{tr}(\mathbf{P}^\top)$

Using two properties of trace function above, it is verified that

$$\operatorname{tr}(\mathbf{X}^\top\mathbf{Y}) = \operatorname{tr}(\mathbf{X}\mathbf{Y}^\top) = \operatorname{tr}(\mathbf{Y}^\top\mathbf{X}) = \operatorname{tr}(\mathbf{Y}\mathbf{X}^\top)$$

---

2) Transformation and Scaling Operations

Create a Matrix $\mathbf{M}$

```
M <- mtcars[c("mpg", "disp", "hp", "drat", "wt")]
head(M, n = 10)
```

```
##                    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
```

Compute the vector containing the means of the columns in **M**

```r
apply(M, 2, mean)
```

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

Create a matrix **Mc**, mean-centered data

```r
Mc <- scale(M, center = TRUE, scale = FALSE)
head(Mc, n = 10)
```

```
##                         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
```

Test if **Mc** is mean-centered

```r
colMeans(Mc)
```

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

Use `sweep()` to create a mean-centered matrix of **M**

```r
A <- sweep(M, 2, colMeans(M), "-")
head(A, 10)
```

```
##                         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
```

```r
head(A == Mc, 10)  # compare the result with Mc
```

```
##                     mpg disp   hp drat   wt
## Mazda RX4          TRUE TRUE TRUE TRUE TRUE
## Mazda RX4 Wag      TRUE TRUE TRUE TRUE TRUE
## Datsun 710         TRUE TRUE TRUE TRUE TRUE
## Hornet 4 Drive     TRUE TRUE TRUE TRUE TRUE
## Hornet Sportabout TRUE TRUE TRUE TRUE TRUE
## Valiant            TRUE TRUE TRUE TRUE TRUE
## Duster 360         TRUE TRUE TRUE TRUE TRUE
## Merc 240D          TRUE TRUE TRUE TRUE TRUE
## Merc 230           TRUE TRUE TRUE TRUE TRUE
## Merc 280           TRUE TRUE TRUE TRUE TRUE
```

Compute a vector of column maxima from **M**

```r
maxcol_M <- apply(M, 2, max)
maxcol_M
```

```
##     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

```r
sweep(M, 2, maxcol_M, "/")
```

```
##                           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 $\mathbf{M}$ scaled such that they have minimum = 0 and maximum = 1

```r
maxcol_M
```

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

```r
mincol_M <- apply(M, 2, min)
mincol_M
```

```
##    mpg   disp     hp   drat     wt
## 10.400 71.100 52.000  2.760  1.513
```

```r
B <- sweep(M, 2, mincol_M, "-")
B <- sweep(B, 2, maxcol_M - mincol_M, "/")
head(B, 10)
```

```
##                         mpg      disp         hp      drat        wt
## Mazda RX4         0.4510638 0.2217511 0.20494700 0.5253456 0.2830478
## Mazda RX4 Wag     0.4510638 0.2217511 0.20494700 0.5253456 0.3482485
## Datsun 710        0.5276596 0.0920429 0.14487633 0.5023041 0.2063411
## Hornet 4 Drive    0.4680851 0.4662010 0.20494700 0.1474654 0.4351828
## Hornet Sportabout 0.3531915 0.7206286 0.43462898 0.1797235 0.4927129
## Valiant           0.3276596 0.3838863 0.18727915 0.0000000 0.4978266
## Duster 360        0.1659574 0.7206286 0.68197880 0.2073733 0.5259524
## Merc 240D         0.5957447 0.1885757 0.03533569 0.4285714 0.4287906
## Merc 230          0.5276596 0.1738588 0.15194346 0.5345622 0.4185630
## Merc 280          0.3744681 0.2407084 0.25088339 0.5345622 0.4927129
```

```r
summary(B)
```

```
##       mpg              disp               hp               drat
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2138   1st Qu.:0.1240   1st Qu.:0.1572   1st Qu.:0.1475
##  Median :0.3745   Median :0.3123   Median :0.2509   Median :0.4309
##  Mean   :0.4124   Mean   :0.3982   Mean   :0.3346   Mean   :0.3855
##  3rd Qu.:0.5277   3rd Qu.:0.6358   3rd Qu.:0.4523   3rd Qu.:0.5346
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##        wt
##  Min.   :0.0000
##  1st Qu.:0.2731
##  Median :0.4633
##  Mean   :0.4358
##  3rd Qu.:0.5362
##  Max.   :1.0000
```

Compute the sample covariance matrix of $\mathbf{M}$

```r
n <- nrow(Mc)
crossprod(Mc) / (n - 1)
```

```
##              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
```

```r
cov(M) # compare the result with cov(M)
```

```
##               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
```

Compute the correlation matrix of **M**

```r
Msd <- scale(M, center = TRUE, scale = TRUE)  # create a standard matirx of M

n <- nrow(Msd)
crossprod(Msd) / (n - 1)
```

```
##             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
```

```r
cor(M)  # compare the result with cor(M)
```

```
##             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
```

Write a function `dummify()` that takes a factor or a character vector, and which returns a matrix with dummy indicators.

```r
dummify <- function(x, all = TRUE) {
  if (!is.character(x) & !is.factor(x)) {
    stop("The input must be a character or factor vector")
  } else if (is.character(x)) {
    x <- as.factor(x)  # convert x into factor if x is a character vector
  } else if (!is.logical(all)) {
    stop("The argument 'all' must be a logical value")
  }




  level <- attributes(x)$levels  # create a vector of levels of x
  n <- length(level)  # number of elements in level

  if (all == FALSE) {
    level <- level[-n]
    n <- n - 1
  }

  M <- matrix((numeric(length(x) * n)), ncol = n)
```

```r
  for (i in 1:n) {
    M[, i] <- as.numeric(x == level[i])
  }

  colnames(M) <- level
  M
}
```

Test `dummify()` works

```r
cyl <- factor(mtcars$cyl)

CYL1 <- dummify(cyl, all = TRUE)
head(CYL1, 10)
```

```
##       4 6 8
##  [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
```

```r
CYL2 <- dummify(cyl, all = FALSE)
head(CYL2, 10)
```

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

Write a function `corsstable()` that takes two factors, and which returns a cross-table between those factors

```r
crosstable <- function(x, y) {
  if(!is.factor(x) | !is.factor(y)) {
    stop("The input must be two factor vectors")
  }

  crossprod(dummify(x), dummify(y))
}
```

Test `crostable()` works

```r
cyl
```

```
##  [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
```

```
## Levels: 4 6 8
```

```r
gear <- factor(mtcars$gear)
gear
```

```
##  [1] 4 4 4 3 3 3 3 4 4 4 4 3 3 3 3 3 3 4 4 4 3 3 3 3 3 4 5 5 5 5 5 4
## Levels: 3 4 5
```

```r
crosstable(cyl, gear)
```

```
##    3 4 5
## 4  1 8 2
## 6  2 4 1
## 8 12 0 2
```