CUHK RMSC4002 Tutorial 1

Benjamin Chan

September 9, 2018

Basic Principle in R Function

To use a function, you should put arguments in the same order as stated in the function definition if you do not specify the argument names. Or else you need to specify the argument names when you do not provide them in correct order. In fact, you can have a mixed type (only for familiar users).

```r
num <- 3.678

# Function definition: round(x, digits = 0)
# Round the values in x to the specified number of decimal places (default 0).
round(num)                    # Round to default 0 decimal place
```

```
[1] 4
```

```r
round(x = num)                # Same as above
```

```
[1] 4
```

```r
round(num, 2)                 # Round to 2 decimal places
```

```
[1] 3.68
```

```r
round(x = num, digits = 2)    # Same as above
```

```
[1] 3.68
```

```r
round(digits = 2, x = num)    # Note: swapped order of arguments
```

```
[1] 3.68
```

Basic Matrix Operation in R

Read in data

Assume that `RMSC4002_Tutorial_1.Rmd` is located under directory `RMSC4002` → `Tutorial 1` while `fin-ratio.csv` is located under `RMSC4002` → `Dataset`. In fact, you do not need to move any file or change directory in order to read in the data.

```r
# Get your current working directory
getwd()
```

```
[1] "C:/Users/Benjamin Chan/Desktop/Tutorial/CUHK-STAT-or-RMSC-Tutorial-Note/RMSC4002/Tutorial 1"
```

```r
# List the names of files in the named directory
# First single dot (.) means relative to the current directory
dir("./")      # Dataset is not found here
```

```
[1] "RMSC4002_Tutorial_1.html" "RMSC4002_Tutorial_1.md"
[3] "RMSC4002_Tutorial_1.R"    "RMSC4002_Tutorial_1.Rmd"
```

```r
# Second double dots (..) means one directory upward from the current directory
dir("./../")   # Dataset is found here
```

```
[1] "Dataset"    "Tutorial 0" "Tutorial 1"
```

```
# Read in data (a CSV file) under Dataset
# Save it to an object named d
d <- read.csv("./../Dataset/fin-ratio.csv")
names(d)    # Output the variable names
```

```
[1] "EY"    "CFTP"  "ln_MV" "DY"    "BTME"  "DTE"   "HSI"
```

```
head(d)     # Return the first part of data (default: 6 rows)
```

```
       EY    CFTP  ln_MV   DY    BTME     DTE HSI
1 -1.8292 -0.1732 5.5405 0.00  1.0379  0.2466   0
2 -0.0797 -0.0830 6.8272 0.00  0.1275 25.4606   0
3 -2.2360 -0.6897 5.0102 0.00 -0.2959  3.3263   0
4 -1.5406 -4.1667 4.4954 0.00 -2.8571  0.9148   0
5 -0.9006 -0.3872 4.5031 0.00  2.7981  0.0753   0
6  0.1923  4.1667 7.0489 4.73  0.9174  3.2208   0
```

```
str(d)      # Display the structure of an object
```

```
'data.frame':   680 obs. of  7 variables:
 $ EY   : num  -1.8292 -0.0797 -2.236 -1.5406 -0.9006 ...
 $ CFTP : num  -0.173 -0.083 -0.69 -4.167 -0.387 ...
 $ ln_MV: num  5.54 6.83 5.01 4.5 4.5 ...
 $ DY   : num  0 0 0 0 0 ...
 $ BTME : num  1.038 0.128 -0.296 -2.857 2.798 ...
 $ DTE  : num  0.2466 25.4606 3.3263 0.9148 0.0753 ...
 $ HSI  : int  0 0 0 0 0 0 0 0 0 0 ...
```

Manipulate data

The last variable HSI is binary. In this stage, please ignore it.

```
# Extract the first 6 columns in d and save it to an object named x
x <- d[, 1:6]
```

Use `apply(X, MARGIN, FUN, ...)` to apply a function `FUN` to `MARGIN` of an array or matrix `X`. Here `MARGIN = 1` means row-wise operation while `MARGIN = 2` means column-wise operation.

```
# Calculate the column means of x and save it to an object named m
# Display object m right after assignment by putting code inside parentheses ()
(m <- apply(x, 2, mean))
```

```
        EY       CFTP      ln_MV         DY       BTME        DTE
-0.6502403 -0.2338956  6.2668068  2.4961735  1.9082626  0.7097322
```

```
# Alternatively
m <- apply(X = x, MARGIN = 2, FUN = mean)    # See Basic Principle in R Function
m
```

```
        EY       CFTP      ln_MV         DY       BTME        DTE
-0.6502403 -0.2338956  6.2668068  2.4961735  1.9082626  0.7097322
```

```
# Calculate the sample covariance matrix of x and save it to an object named S
S <- var(x)
(round(var(x), 3))      # Display only 3 decimal places
```

```
          EY   CFTP   ln_MV     DY    BTME    DTE
```

```
EY     18.498 2.909  1.160  1.920  1.478  0.338
CFTP    2.909 3.693  0.766  1.237  1.823  0.329
ln_MV   1.160 0.766  2.744  0.972 -0.773 -0.074
DY      1.920 1.237  0.972 13.872 -0.258  0.158
BTME    1.478 1.823 -0.773 -0.258 68.308  1.962
DTE     0.338 0.329 -0.074  0.158  1.962 12.993
```

```r
# Calculate the sample correlation matrix of x
round(cor(x), 3)        # Display only 3 decimal places
```

```
         EY  CFTP  ln_MV     DY   BTME    DTE
EY    1.000 0.352  0.163  0.120  0.042  0.022
CFTP  0.352 1.000  0.241  0.173  0.115  0.047
ln_MV 0.163 0.241  1.000  0.158 -0.056 -0.012
DY    0.120 0.173  0.158  1.000 -0.008  0.012
BTME  0.042 0.115 -0.056 -0.008  1.000  0.066
DTE   0.022 0.047 -0.012  0.012  0.066  1.000
```

Note that

$$\mathrm{Corr}(X, X) = 1.$$

Manipulate matrices

```r
options(digits = 4)      # Control display to 4 decimals
det(solve(S))            # Determinant of inverse of S
```

```
[1] 5.706e-07
```

```r
1/det(S)
```

```
[1] 5.706e-07
```

Note that

$$\left|S^{-1}\right| = \frac{1}{|S|}.$$

```r
eig <- eigen(S)          # Save eigenvalues and eigenvectors of S
names(eig)               # Display items in eig
```

```
[1] "values"  "vectors"
```

```r
(eval <- eig$values)     # Save eigenvalues
```

```
[1] 68.487 19.918 13.205 12.899  3.341  2.257
```

```r
(H <- eig$vectors)       # Save matrix of eigenvectors
```

```
          [,1]     [,2]      [,3]      [,4]     [,5]     [,6]
[1,]  0.031107  0.91185  0.364402  0.016136  0.18218 -0.03637
[2,]  0.029477  0.19142 -0.018447  0.005915 -0.81898  0.53980
[3,] -0.010938  0.09104 -0.043829 -0.020125 -0.53213 -0.84030
[4,] -0.003038  0.34621 -0.911976 -0.188393  0.11223  0.01856
[5,]  0.998380 -0.03383 -0.007556 -0.036516  0.01255 -0.02334
[6,]  0.035663  0.05094 -0.182190  0.981058  0.01304 -0.01720
```

```r
# t(x) returns the transpose of x
# %*%: matrix multiplication
round(t(H)%*%H, 3)
```

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

```r
round(H%*%t(H), 3)
```

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

If H is an orthogonal matrix, then

$$H'H = HH' = I.$$

```r
h1 <- H[, 1]               # Extract first column of H (first eigenvector) to h1
eval[1]*h1                 # Compute lambda1*h1  (displayed as row vector)
```

```
[1]  2.1304  2.0188 -0.7491 -0.2080 68.3765  2.4425
```

```r
as.vector(S%*%h1)          # Compute S*h1        (displayed as row vector)
```

```
[1]  2.1304  2.0188 -0.7491 -0.2080 68.3765  2.4425
```

Note that

$$Sh_1 = \lambda_1 h_1.$$

```r
round(t(H)%*%S%*%H, 3)
```

```
      [,1]  [,2]  [,3] [,4]  [,5]  [,6]
[1,] 68.49  0.00  0.00  0.0 0.000 0.000
[2,]  0.00 19.92  0.00  0.0 0.000 0.000
[3,]  0.00  0.00 13.21  0.0 0.000 0.000
[4,]  0.00  0.00  0.00 12.9 0.000 0.000
[5,]  0.00  0.00  0.00  0.0 3.341 0.000
[6,]  0.00  0.00  0.00  0.0 0.000 2.257
```

Note that

$$H'SH = D = \mathrm{diag}(\lambda_1, ..., \lambda_6).$$

```r
D <- diag(eval)              # Form diagonal matrix D
H%*%D%*%t(H)
```

```
       [,1]   [,2]     [,3]    [,4]    [,5]     [,6]
[1,] 18.498 2.9090  1.16019  1.9204  1.4781  0.33795
[2,]  2.909 3.6931  0.76630  1.2371  1.8228  0.32879
[3,]  1.160 0.7663  2.74394  0.9721 -0.7734 -0.07413
[4,]  1.920 1.2371  0.97207 13.8716 -0.2575  0.15815
[5,]  1.478 1.8228 -0.77342 -0.2575 68.3082  1.96177
[6,]  0.338 0.3288 -0.07413  0.1582  1.9618 12.99291
```

Note that

$$HDH' = S.$$

```r
sqrt(D)
```

```
      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]
[1,] 8.276 0.000 0.000 0.000 0.000 0.000
[2,] 0.000 4.463 0.000 0.000 0.000 0.000
[3,] 0.000 0.000 3.634 0.000 0.000 0.000
[4,] 0.000 0.000 0.000 3.591 0.000 0.000
[5,] 0.000 0.000 0.000 0.000 1.828 0.000
[6,] 0.000 0.000 0.000 0.000 0.000 1.502
```

```r
(rS <- H%*%sqrt(D)%*%t(H))
```

```
        [,1]   [,2]     [,3]    [,4]     [,5]     [,6]
[1,] 4.26492 0.4603  0.17720  0.2259  0.11267  0.03736
[2,] 0.46026 1.8359  0.19270  0.1992  0.17666  0.05180
[3,] 0.17720 0.1927  1.62490  0.1672 -0.08301 -0.01540
[4,] 0.22591 0.1992  0.16722  3.7083 -0.02570  0.02000
[5,] 0.11267 0.1767 -0.08301 -0.0257  8.26013  0.16422
[6,] 0.03736 0.0518 -0.01540  0.0200  0.16422  3.60017
```

```r
rS%*%rS
```

```
       [,1]   [,2]     [,3]    [,4]    [,5]     [,6]
[1,] 18.498 2.9090  1.16019  1.9204  1.4781  0.33795
[2,]  2.909 3.6931  0.76630  1.2371  1.8228  0.32879
[3,]  1.160 0.7663  2.74394  0.9721 -0.7734 -0.07413
[4,]  1.920 1.2371  0.97207 13.8716 -0.2575  0.15815
[5,]  1.478 1.8228 -0.77342 -0.2575 68.3082  1.96177
[6,]  0.338 0.3288 -0.07413  0.1582  1.9618 12.99291
```

Note that

$$HD^{1/2}H' = S^{1/2}$$

and

$$S^{1/2}S^{1/2} = S.$$