Computation

Sim, Min Kyu, Ph.D., `mksim@seoultech.ac.kr`

서울과학기술대학교
SEOULTECH SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

The first half of our course

The second half of our course

About

- This lecture note covers how to use statistcal software, R, to deal with computations in linear algebra.
- Installation and basic guides for R can be found at: https://github.com/aceMKSim/teaching/blob/master/Data%20Visualization/Lecture%20Notes/L01_installR.pdf
- More interested in R and data visualization?
  - You can access my lecture note for a graduate course, ‘Data Visualization’ at https://github.com/aceMKSim/teaching/tree/master/Data%20Visualization
  - You can request an invitation link to a superb online course platform, www.datacamp.com, where courses for R and Python are available free through my invitation.

The first half of our course

Matrix Input

- Matrix input is done by 1) listing all elements by `c()` function, 2) telling the exact number of row by `nrow=`, and 3) whether you want to list elements by row or by column (`byrow`).

```
A1 <- matrix(c(1,-3,0,2,-1,5,1,2,3),
            nrow = 3, byrow = FALSE)
b1 <- matrix(c(0,1,-1),
            nrow=3, byrow=FALSE)
A2 <- matrix(c(4,2,2,-1,1,-1,6,6,8),
            nrow=3, byrow=FALSE)
A3 <- matrix(c(1,0,-1,-7,0,7,0,1,4,6,-2,2),
            nrow=3, byrow=FALSE)
b3 <- matrix(c(5,-3,7),
            nrow=3, byrow=FALSE)
# Following to check for correct input
A1
```

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

```
b1
```

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

```
A2
```

```
##      [,1] [,2] [,3]
## [1,]    4   -1    6
## [2,]    2    1    6
## [3,]    2   -1    8
```

```
A3
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1   -7    0    6
## [2,]    0    0    1   -2
## [3,]   -1    7    4    2
```

```r
b3
```

```
##      [,1]
## [1,]    5
## [2,]   -3
## [3,]    7
```

- Concatenating function `cbind()` allows to construct a augumented matrix.

```r
Aug3 <- cbind(A3, b3)
Aug3
```

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

Reduced Echelon Form

- Package (often called as `library`) provides additional functionality.
- In the first ever to use a package, it must be installed by following syntax.

```
install.packages("pracma")
```

- In your computation session, you must use following syntax to activate the package.

```
library(pracma)
```

- The package `pracma` offers `rref()` that gives the reduced echelon form.

```
library(pracma)
A1
```

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

```
rref(A1)
```

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

```r
A2
```

```
##      [,1] [,2] [,3]
## [1,]    4   -1    6
## [2,]    2    1    6
## [3,]    2   -1    8
```

```r
rref(A2)
```

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

```r
A3
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1   -7    0    6
## [2,]    0    0    1   -2
## [3,]   -1    7    4    2
```

```r
rref(A3)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1   -7    0    0
## [2,]    0    0    1    0
## [3,]    0    0    0    1
```

```r
Aug3
```

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

```r
rref(Aug3)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1   -7    0    0 -4.0
## [2,]    0    0    1    0  0.0
## [3,]    0    0    0    1  1.5
```

Solving $Ax = b$

```r
A1
```

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

```r
b1
```

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

```r
x <- solve(A1, b1)
x
```

```
##      [,1]
## [1,]  0.6
## [2,] -0.8
## [3,]  1.0
```

- `%*%` is an operator for matrix multiplication.

```r
# Sanity check
A1 %*% x
```

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

```r
b1
```

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

Inverting

```r
A1
```

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

```r
inv_A1 <- solve(A1)
inv_A1
```

```
##      [,1] [,2] [,3]
## [1,]  1.3  0.1 -0.5
## [2,] -0.9 -0.3  0.5
## [3,]  1.5  0.5 -0.5
```

```r
# Sanity check
A1 %*% inv_A1
```

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

```r
inv_A1 %*% A1
```

```
##      [,1] [,2]         [,3]
## [1,]    1    0 0.000000e+00
## [2,]    0    1 2.220446e-16
## [3,]    0    0 1.000000e+00
```

Rank

```r
library(Matrix)
A1
```

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

```r
rankMatrix(A1)
```

```
## [1] 3
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 6.661338e-16
```

LU decomposition

```r
library(matrixcalc)
A1
```

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

```r
lu.decomposition(A1)
```

```
## $L
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]   -3    1    0
## [3,]    0    1    1
##
## $U
##      [,1] [,2] [,3]
## [1,]    1    2    1
## [2,]    0    5    5
## [3,]    0    0   -2
```

```r
# Sanity check
A1_L <- lu.decomposition(A1)$L
A1_U <- lu.decomposition(A1)$U
A1_L %*% A1_U
```

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

```r
A1
```

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

Determinants

```r
A1
```

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

```r
det(A1)
```

```
## [1] -10
```

```r
A2
```

```
##      [,1] [,2] [,3]
## [1,]    4   -1    6
## [2,]    2    1    6
## [3,]    2   -1    8
```

```r
det(A2)
```

```
## [1] 36
```

The second half of our course

Eigenvalue and Eigenvector

```r
A2
```

```
##      [,1] [,2] [,3]
## [1,]    4   -1    6
## [2,]    2    1    6
## [3,]    2   -1    8
```

```r
eigen(A2)
```

```
## eigen() decomposition
## $values
## [1] 9 2 2
##
## $vectors
##           [,1]      [,2]       [,3]
## [1,] 0.5773503 0.2915782  0.3377822
## [2,] 0.5773503 0.8863713  0.0000000
## [3,] 0.5773503 0.0505358 -0.1125941
```

```r
eigen(A2)$values
```

```
## [1] 9 2 2
```

```r
eigen(A2)$vectors
```

```
##           [,1]      [,2]       [,3]
## [1,] 0.5773503 0.2915782  0.3377822
## [2,] 0.5773503 0.8863713  0.0000000
## [3,] 0.5773503 0.0505358 -0.1125941
```

```r
lambda1 <- eigen(A2)$values[1]
lambda2 <- eigen(A2)$values[2]
lambda3 <- eigen(A2)$values[3]
v1 <- eigen(A2)$vectors[,1]
v2 <- eigen(A2)$vectors[,2]
v3 <- eigen(A2)$vectors[,3]
```

```r
lambda1
```

```
## [1] 9
```

```r
lambda2
```

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

```r
lambda3
```

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

```r
v1
```

```
## [1] 0.5773503 0.5773503 0.5773503
```

```r
v2
```

```
## [1] 0.2915782 0.8863713 0.0505358
```

```r
v3
```

```
## [1]  0.3377822  0.0000000 -0.1125941
```

Diagonalization

$$A = PDP^{-1}$$

```r
P <- eigen(A2)$vectors
D <- diag(eigen(A2)$values)
```

```r
P
```

```
##           [,1]      [,2]       [,3]
## [1,] 0.5773503 0.2915782  0.3377822
## [2,] 0.5773503 0.8863713  0.0000000
## [3,] 0.5773503 0.0505358 -0.1125941
```

```r
D
```

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

```r
# sanity check
A2
```

```
##      [,1] [,2] [,3]
## [1,]    4   -1    6
## [2,]    2    1    6
## [3,]    2   -1    8
```

```r
P %*% D %*% solve(P)
```

```
##      [,1] [,2] [,3]
## [1,]    4   -1    6
## [2,]    2    1    6
## [3,]    2   -1    8
```

Dataset `mtcars`

- The rest of this lecture note will use the one of the popular built-in datasets in R, `mtcats`, which includes several cars and their fuel efficiency. To analyze the dataset.
- Typing `help(mtcars)` in your console will provide basic description on the dataset.

```
help(mtcars)
```

The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models)

- The dataset can be explored by following syntax, `str()`.

```
str(mtcars)
```

```
## '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 ...
```

Each varible's description is also provided from the help page as follows.

- [, 1] `mpg`, Miles/(US) gallon
- [, 2] `cyl`, Number of cylinders
- [, 3] `disp`, Displacement (cu.in.)
- [, 4] `hp`, Gross horsepower
- [, 5] `drat`, Rear axle ratio
- [, 6] `wt`, Weight (1000 lbs)
- [, 7] `qsec`, 1/4 mile time
- [, 8] `vs`, Engine (0 = V-shaped, 1 = straight)
- [, 9] `am`, Transmission (0 = automatic, 1 = manual)
- [,10] `gear`, Number of forward gears
- [,11] `carb`, Number of carburetors
- The first variable `mpg` refers to the fuel efficiency, since it measures how many miles the car can move with a gallon of its fuel. The other 10 variables should affect the `mpg`. That is, we call `mpg` as a dependent variable and the other variables as independent variables.

Linear regression

- For simplicity, the rest of this will use only the first six independent variables from now on: `cyl, disp, hp, drat, wt, qsec`
- We will use
  - `disp` as an independent variable (i.e. 'X' variable)
  - `mpg` as dependent variable (i.e. 'Y' variable).
  - Following codes will reveal a few observed pairs of (X, Y).
  - i.e. the function `head()` reveals the first six observations of the dataset.

```
head(mtcars[,c("disp", "mpg")])
```

```
##                   disp  mpg
## Mazda RX4          160 21.0
## Mazda RX4 Wag      160 21.0
## Datsun 710         108 22.8
## Hornet 4 Drive     258 21.4
## Hornet Sportabout  360 18.7
## Valiant            225 18.1
```

- Based on the textbook (Ch.6, Example 4),
  - we shall construct a A matrix where
    - the first column is one-vector (generated by `rep (1, length(mtcars$disp))`)
    - the second column is the observed quantity of `disp` (generated by `mtcars$disp`).
  - The matrix (or, a vector) b is nothing but the quantities of `mtcars$mpg`.
- This can be done with following codes. (The function `matrix()` is another way to create matrix.)

```
A <- matrix(cbind(rep(1,length(mtcars$disp)), mtcars$disp), ncol = 2)
b <- matrix(mtcars$mpg, ncol = 1)
```

- To make sure A and b are correctly assigned, following codes reveal its beginning part.

```
head(A)
```

```
##      [,1] [,2]
## [1,]    1  160
## [2,]    1  160
## [3,]    1  108
## [4,]    1  258
## [5,]    1  360
## [6,]    1  225
```

```
head(b)
```

```
##      [,1]
## [1,] 21.0
## [2,] 21.0
## [3,] 22.8
## [4,] 21.4
## [5,] 18.7
## [6,] 18.1
```

- Using A and b, the least-squares solution is obtained by solving

$$A^t Ax = A^t b$$

- Assign $(A^t A)^{-1} A^t b$ to the variable x

```
x <- solve(t(A) %*% A) %*% t(A) %*% b
x
```

```
##             [,1]
## [1,] 29.59985476
## [2,] -0.04121512
```

```
x[1] # alpha
```

```
## [1] 29.59985
```

```
x[2] # beta
```

```
## [1] -0.04121512
```

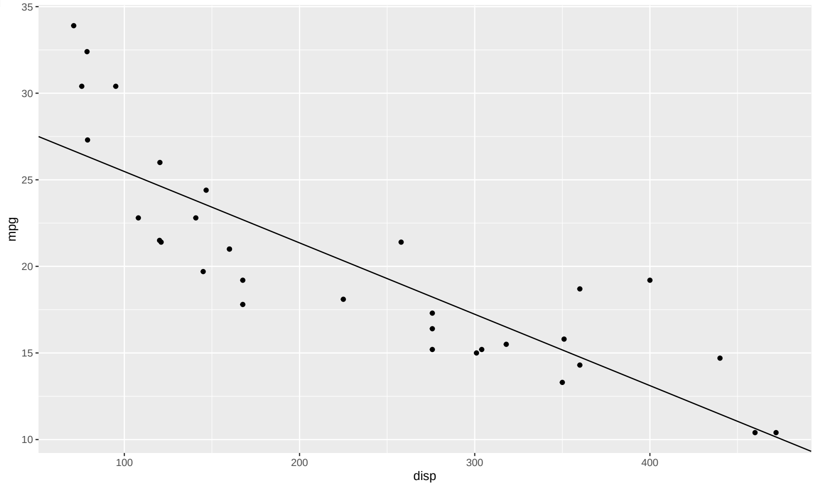
- As a statistical software, R offers linear regression function called `lm()`.
- Basic usage is given below. Confirm that the results match the numbers in the previous page.

```r
lm(mtcars$mpg ~ mtcars$disp)
```

```
##
## Call:
## lm(formula = mtcars$mpg ~ mtcars$disp)
##
## Coefficients:
## (Intercept)  mtcars$disp
##    29.59985     -0.04122
```

- Following is visualizion of original data and regression line.

```
library(ggplot2)
ggplot(mtcars, aes(x=disp, y=mpg)) +
  geom_point() +
  geom_abline(intercept = x[1], slope = x[2])
```



Principal Component Analysis

Preparation

- PCA (Principal Component Analysis) is to identify common underlying components in dataset.
- In our set of six independent variables, i.e. `mtcars[,c("cyl", "disp", "hp", "drat", "wt", "qsec")]`, the six variables are naturally highly correlated to each other.
- For example, larger engine size (`disp`) is likely to be positively correlated to its horse power (`hp`) and its weight (`wt`).
- Let X be the dataset of the independent variables.

```
X <- as.matrix(mtcars[,c("cyl", "disp", "hp", "drat", "wt", "qsec")])
head(X)
```

```
##                   cyl disp  hp drat    wt  qsec
## Mazda RX4           6  160 110 3.90 2.620 16.46
## Mazda RX4 Wag       6  160 110 3.90 2.875 17.02
## Datsun 710          4  108  93 3.85 2.320 18.61
## Hornet 4 Drive      6  258 110 3.08 3.215 19.44
## Hornet Sportabout   8  360 175 3.15 3.440 17.02
## Valiant             6  225 105 2.76 3.460 20.22
```

Correlation matrix

- To explore its correlation structure, the following code generates its correlation matrix.

```r
cor(X)
```

```
##             cyl       disp         hp        drat         wt        qsec
## cyl   1.0000000  0.9020329  0.8324475 -0.69993811  0.7824958 -0.59124207
## disp  0.9020329  1.0000000  0.7909486 -0.71021393  0.8879799 -0.43369788
## hp    0.8324475  0.7909486  1.0000000 -0.44875912  0.6587479 -0.70822339
## drat -0.6999381 -0.7102139 -0.4487591  1.00000000 -0.7124406  0.09120476
## wt    0.7824958  0.8879799  0.6587479 -0.71244065  1.0000000 -0.17471588
## qsec -0.5912421 -0.4336979 -0.7082234  0.09120476 -0.1747159  1.00000000
```

Assumption of PCA

- PCA assumes that, each observation can be tersely expressed without using all of the six variables.
- For example, since `cyl` and `disp` are highly correlated, and one can find a nearly linear combination between the two.
- Or, it would mean that each observation may be characterized by a single variable, which is a linear combination of `cyl` and `disp`.

Dimension reduction

- Since the original two variables, `cyl` and `disp`, can be used to generate a single variable through their linear combination without necessarily hurting represent-ability, now the originally two variables are reduced to a single variable.
- Thus, PCA is one of the *dimension reduction* techniques, and maybe useful especially if
  - *i)* variables are highly correlated to each other and/or
  - *ii)* there are too many variables to consider.

Preparing covariance matrix

- Remind that, from your elementary statistics course, one variable's sample variance calculated as $\frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n-1}$.
- In order words, from original observations, i) subtract its sample mean (a.k.a. *de-mean* process), then ii) take squared value, and then divide it by $n-1$.
- Preparing sample covariance matrix is analgous to that as follows.
- Please attemp to understand the following code.

```
# de-meaning process of $X$
# (for each column, x-mean(x) is applied)
X_demeaned <- apply(X, 2, function(x) x-mean(x))
```

- Remind that, for a matrix A, `t(X)%*%X` is equivalent to squaring it.

```
Cov <- (t(X_demeaned) %*% X_demeaned)/(nrow(X)-1)
```

- Covariance matrix `Cov` is a 6×6 matrix, because six variables are involved here.

```
Cov
```

```
##              cyl        disp         hp        drat          wt        qsec
## cyl    3.1895161   199.66028  101.93145 -0.66836694   1.3673710 -1.88685484
## disp 199.6602823 15360.79983 6721.15867 -47.06401915 107.6842040 -96.05168145
## hp   101.9314516  6721.15867 4700.86694 -16.45110887  44.1926613 -86.77008065
## drat  -0.6683669   -47.06402  -16.45111  0.28588135  -0.3727207  0.08714073
## wt     1.3673710   107.68420   44.19266 -0.37272073   0.9573790 -0.30548161
## qsec  -1.8868548   -96.05168  -86.77008  0.08714073  -0.3054816  3.19316613
```

Positive semi-definite (psd)

- A proper covariance matrix is positive semi-definite, meaning that its eigenvalues are all non-negative. (Just like your familiar variance is always non-negative)
- Find its eigenvalues and see if they are all non-negative numbers.

```r
eigen(Cov)$values
```

```
## [1] 1.861324e+04 1.453791e+03 1.586537e+00 4.584231e-01 1.248216e-01
## [6] 8.738486e-02
```

- All looks good, so we can proceed.

Principal components

- Remind that the six pricinpal components correspond to $v_1, v_2, \ldots, v_6$. It can be retrieved as following.

```
eigen(Cov)$vectors
```

```
##              [,1]         [,2]         [,3]         [,4]         [,5]
## [1,] -0.012042822 -0.003376004  0.225464493  0.913153920  0.116301042
## [2,] -0.900263018  0.435130058  0.004337984 -0.012130860  0.004694063
## [3,] -0.435075671 -0.899991800 -0.026532118 -0.002512279  0.003469759
## [4,]  0.002661502 -0.003901248  0.057414058 -0.323952312 -0.131060362
## [5,] -0.006242691  0.004868276 -0.203604150  0.191625494 -0.958484375
## [6,]  0.006676402  0.025025499 -0.950627144  0.155984033  0.224879700
##              [,6]
## [1,]  0.318799814
## [2,]  0.000566658
## [3,] -0.001271341
## [4,]  0.935178808
## [5,] -0.055408055
## [6,]  0.143997516
```

Explanation power of each principal component.

- Let the eigenvalues of the matrix `Cov` be $\lambda_1, \lambda_2, \ldots, \lambda_6$ in a descending order.
- The amount of overall variation that the first principal component can explain is $\frac{\lambda_1}{\sum_{i=1}^{6} \lambda_i}$. (Check this number is about 92.7%)
- The amount of overall variation that the second principal component can explain is $\frac{\lambda_2}{\sum_{i=1}^{6} \lambda_i}$, and so on.
- In sum, the following is the explanation power of each principal components.

```
exp_power <- eigen(Cov)$values/sum(eigen(Cov)$values)
exp_power
```

```
## [1] 9.274490e-01 7.243857e-02 7.905297e-05 2.284202e-05 6.219529e-06
## [6] 4.354157e-06
```

- Cumulatively, the first n principal components' explanation power is:

```
cumsum(exp_power) # cumulative sum
```

```
## [1] 0.9274490 0.9998875 0.9999666 0.9999894 0.9999956 1.0000000
```

```
"Optimism is the faith that leads to achievement - Hellen Keller"
```