Data Mining Practice

Intro

[R intro] (http://youtu.be/TR2bHSJ_eck)

1. Simple manipulations; numbers and vectors

1.1 Vectors and assignment

```
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
x
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```
assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
x
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```
c(10.4, 5.6, 3.1, 6.4, 21.7) <- x
```

```
## Error: 대입의 대상이 비언어적 객체로 확장됩니다
```

```
x
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```
1/x
```

```
## [1] 0.09615 0.17857 0.32258 0.15625 0.04608
```

```
y <- c(x, 0, x)
y
```

```
##  [1] 10.4  5.6  3.1  6.4 21.7  0.0 10.4  5.6  3.1  6.4 21.7
```

1.2 Vector arithmetic

```
v <- 2 * x + y + 1
```

```
## Warning: 두 객체의 길이가 서로 배수관계에 있지 않습니다
```

```
v
```

```
##  [1] 32.2 17.8 10.3 20.2 66.1 21.8 22.6 12.8 16.9 50.8 43.5
```

```
sum((x - mean(x))^2)/(length(x) - 1)
```

```
## [1] 53.85
```

```
sqrt(-17)
```

```
## Warning: NaN이 생성되었습니다
```

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

```
sqrt(-17 + 0)
```

```
## Warning: NaN이 생성되었습니다
```

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

1.3 Generating regular sequences

```
s3 <- seq(-5, 5, by = 0.2)
s3
```

```
##  [1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2 -3.0 -2.8 -2.6 -
2.4
## [15] -2.2 -2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2  0.0  0.2
0.4
## [29]  0.6  0.8  1.0  1.2  1.4  1.6  1.8  2.0  2.2  2.4  2.6  2.8  3.0
3.2
## [43]  3.4  3.6  3.8  4.0  4.2  4.4  4.6  4.8  5.0
```

```
s4 <- seq(length = 51, from = -5, by = 0.2)
s4
```

```
##  [1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2 -3.0 -2.8 -2.6 -
2.4
## [15] -2.2 -2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2  0.0  0.2
0.4
## [29]  0.6  0.8  1.0  1.2  1.4  1.6  1.8  2.0  2.2  2.4  2.6  2.8  3.0
3.2
## [43]  3.4  3.6  3.8  4.0  4.2  4.4  4.6  4.8  5.0
```

```
s5 <- rep(x, times = 5)
s5
```

```
##  [1] 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1
6.4
## [15] 21.7 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1  6.4 21.7
```

```
s6 <- rep(x, each = 5)
s6
```

```
##  [1] 10.4 10.4 10.4 10.4 10.4  5.6  5.6  5.6  5.6  5.6  3.1  3.1  3.1
3.1
## [15]  3.1  6.4  6.4  6.4  6.4  6.4 21.7 21.7 21.7 21.7 21.7
```

1.4 Logical vectors

```
x
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```
temp <- x > 13
temp
```

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

1.5 Missing values

```
z <- c(1:3, NA)
ind <- is.na(z)
z
```

```
## [1]  1  2  3 NA
```

```
ind
```

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

```
0/0
```

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

```
Inf - Inf
```

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

1.6 Character vectors

```
labs <- paste(c("X", "Y"), 1:10, sep = "")
labs
```

```
##  [1] "X1"  "Y2"  "X3"  "Y4"  "X5"  "Y6"  "X7"  "Y8"  "X9"  "Y10"
```

```
c("X1", "Y2", "X3", "Y4", "X5", "Y6", "X7", "Y8", "X9", "Y10")
```

```
##  [1] "X1"  "Y2"  "X3"  "Y4"  "X5"  "Y6"  "X7"  "Y8"  "X9"  "Y10"
```

1.7 Index vectors; selecting and modifying subsets of a data set

1.7.1 A logical vector

```
x <- c(1, 2, 3, NA, 5, 6, 7, NA, 9, -10)
x
```

```
##  [1]   1   2   3  NA   5   6   7  NA   9 -10
```

```
y <- x[!is.na(x)]
y
```

```
## [1]   1   2   3   5   6   7   9 -10
```

```
z <- (x + 1)[(!is.na(x)) & x > 0]
z
```

```
## [1]  2  3  4  6  7  8 10
```

1.7.2. A vector of positive integral quantities

```
x[1:10]
```

```
##  [1]   1   2   3  NA   5   6   7  NA   9 -10
```

```
c("x", "y")[rep(c(1, 2, 2, 1), times = 4)]
```

```
##  [1] "x" "y" "y" "x" "x" "y" "y" "x" "x" "y" "y" "x" "x" "y" "y" "x"
```

1.7.3. A vector of negative integral quantities.

```
y <- x[-(1:5)]
y
```

```
## [1]   6   7  NA   9 -10
```

1.7.4. A vector of character strings

```
fruit <- c(5, 10, 1, 20)
names(fruit) <- c("orange", "banana", "apple", "peach")
fruit
```

```
## orange banana  apple  peach
##      5     10      1     20
```

```
lunch <- fruit[c("apple", "orange")]
lunch
```

```
##  apple orange
##      1      5
```

```
# x[is.na(x)] <- 0
```

```
# y[y < 0] <- -y[y < 0]
```

```
# y <- abs(y)
```

2. Objects, their modes and attributes

2.1 Intrinsic attributes: mode and length

```
z <- 0:9
digits <- as.character(z)
digits
```

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

```
d <- as.integer(digits)
d
```

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

2.2 Changing the length of an object

```
e <- numeric()
e
```

```
## numeric(0)
```

```
e[3] <- 17
e
```

```
## [1] NA NA 17
```

```
alpha <- 1:10
alpha <- alpha[2 * 1:5]
alpha
```

```
## [1]  2  4  6  8 10
```

```
length(alpha) <- 3
alpha
```

```
## [1] 2 4 6
```

2.3 Getting and setting attributes

```
attr(z, "dim") <- c(1, 10)
```

2.4 The class of an object

```
mtcars
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
```

```
## 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
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## 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
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
class(mtcars)
```

```
## [1] "data.frame"
```

```
# unclass(mtcars)
```

3. Ordered and unordered factors

3.1 A specific example

```
state <- c("tas", "sa", "qld", "nsw", "nsw", "nt", "wa", "wa", "qld", "vic
",
    "nsw", "vic", "qld", "qld", "sa", "tas", "sa", "nt", "wa", "vic", "qld
",
    "nsw", "nsw", "wa", "sa", "act", "nsw", "vic", "vic", "act")
state
```

```
##  [1] "tas" "sa"  "qld" "nsw" "nsw" "nt"  "wa"  "wa"  "qld" "vic" "nsw"
## [12] "vic" "qld" "qld" "sa"  "tas" "sa"  "nt"  "wa"  "vic" "qld" "nsw"
## [23] "nsw" "wa"  "sa"  "act" "nsw" "vic" "vic" "act"
```

```
statef <- factor(state)
statef
```

```
##  [1] tas sa  qld nsw nsw nt  wa  wa  qld vic nsw vic qld qld sa  tas sa
```

```
## [18] nt  wa  vic qld nsw nsw wa  sa  act nsw vic vic act
## Levels: act nsw nt qld sa tas vic wa
```

```
levels(statef)
```

```
## [1] "act" "nsw" "nt"  "qld" "sa"  "tas" "vic" "wa"
```

```
unique(statef)
```

```
## [1] tas sa  qld nsw nt  wa  vic act
## Levels: act nsw nt qld sa tas vic wa
```

3.2 The function tapply() and ragged arrays

```
incomes <- c(60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70, 42, 56, 61, 61, 6
1,
    58, 51, 48, 65, 49, 49, 41, 48, 52, 46, 59, 46, 58, 43)

incmeans <- tapply(incomes, statef, mean)
incmeans
```

```
##   act   nsw    nt   qld    sa   tas   vic    wa
## 44.50 57.33 55.50 53.60 55.00 60.50 56.00 52.25
```

```
stderr <- function(x) sqrt(var(x)/length(x))

incester <- tapply(incomes, statef, sum)
incester
```

```
## act nsw  nt qld  sa tas vic  wa
##  89 344 111 268 220 121 280 209
```

```
incster <- tapply(incomes, statef, stderr)
incster
```

```
##   act   nsw    nt   qld    sa   tas   vic    wa
## 1.500 4.310 4.500 4.106 2.739 0.500 5.244 2.658
```

```
# breif intro to 'apply' in R

# base::apply Apply Functions Over Array Margins base::by Apply a Function
# to a Data Frame Split by Factors base::eapply Apply a Function Over Valu
es
# in an Environment base::lapply Apply a Function over a List or Vector
# base::mapply Apply a Function to Multiple List or Vector Arguments
# base::rapply Recursively Apply a Function to a List base::tapply Apply a
# Function Over a Ragged Array
```

4. Lists and data frames

4.1 Lists

```
Lst <- list(name = "Fred", wife = "Mary", no.children = 3, child.ages = c
(4,
    7, 9))
Lst
```

```
## $name
## [1] "Fred"
##
## $wife
## [1] "Mary"
##
## $no.children
## [1] 3
##
## $child.ages
## [1] 4 7 9
```

```
Lst[[1]]
```

```
## [1] "Fred"
```

```
Lst[[2]]
```

```
## [1] "Mary"
```

```
Lst[[3]]
```

```
## [1] 3
```

```
Lst[[4]][1]
```

```
## [1] 4
```

```
length(Lst)
```

```
## [1] 4
```

```
Lst$name
```

```
## [1] "Fred"
```

```
# same as Lst[[1]]
```

Real data

```
mkt <- read.table("C:/Users/dox/Desktop/DS&BDA 중급과정-2013/데이터/marketp
rice.csv",
    header = T, sep = ",")
```

```
head(mkt)
```

```
##    P_SEQ M_SEQ          M_NAME A_SEQ           A_NAME     A_UNIT A_PRIC
E
## 1 418416    74 이마트 왕십리점   305 사과(부사, 300g)       1 개      140
## 2 418417    74 이마트 왕십리점   306   배(신고, 600g)       1 개     3933
## 3 418418    74 이마트 왕십리점   125       배추(국산) 1 포기(2kg)     2980
## 4 418419    74 이마트 왕십리점    25              무       1 개     1480
## 5 418420    74 이마트 왕십리점    24            양파 1 망(1.5kg)     4580
## 6 418421    74 이마트 왕십리점    23            상추       100g      720
##   P_YEAR_MONTH     ADD_COL M_TYPE_CODE M_TYPE_NAME M_GU_CODE M_GU_NAME
## 1      2013-03  4 개 5600 원           2     대형마트     200000     성동구
## 2      2013-03 3 개 11800 원           2     대형마트     200000     성동구
## 3      2013-03                        2     대형마트     200000     성동구
## 4      2013-03                        2     대형마트     200000     성동구
## 5      2013-03                        2     대형마트     200000     성동구
## 6      2013-03 150g 1080 원           2     대형마트     200000     성동구
```

```
tail(mkt)
```

```
##       P_SEQ M_SEQ   M_NAME A_SEQ           A_NAME        A_UNIT A_PRIC
E
## 4699 406491   146 방이시장   283      닭고기(육계) 1 마리(1000g)     8000
## 4700 406492   146 방이시장   171        달걀(특란)         10 개     2150
## 4701 406493   146 방이시장   259   조기(냉동,국산)   1 마리(20cm)     1780
## 4702 406494   146 방이시장   152 명태(러시아,냉동)   1 마리(45cm)     5000
## 4703 406495   146 방이시장   256 오징어(냉동,국산)   1 마리(25cm)     3000
## 4704 406496   146 방이시장   266 고등어(생물,국산)   1 마리(30cm)     5000
##      P_YEAR_MONTH      ADD_COL M_TYPE_CODE M_TYPE_NAME M_GU_CODE M_GU_N
AME
## 4699      2013-01       황금닭           1     전통시장     710000     송파
구
## 4700      2013-01     신선특란           1     전통시장     710000     송파
구
## 4701      2013-01 2 마리 3560 원           1     전통시장     710000     송파
구
## 4702      2013-01         행사           1     전통시장     710000     송파
구
## 4703      2013-01         해동           1     전통시장     710000     송파
```

구

```
## 4704      2013-01           대           1     전통시장     710000     송파
```

구

```
str(mkt)
```

```
## 'data.frame':    4704 obs. of  13 variables:
##  $ P_SEQ       : int  418416 418417 418418 418419 418420 418421 418422
418423 418424 418425 ...
##  $ M_SEQ       : int  74 74 74 74 74 74 74 74 74 74 ...
##  $ M_NAME      : Factor w/ 98 levels "2001 아울렛 불광점",..: 68 68 68 68
 68 68 68 68 68 68 ...
##  $ A_SEQ       : int  305 306 125 25 24 23 311 119 58 52 ...
##  $ A_NAME      : Factor w/ 75 levels "고등어","고등어(30cm,국산)",..: 42
31 36 23 56 46 60 74 52 15 ...
##  $ A_UNIT      : Factor w/ 293 levels " (100g)"," (138g)",..: 31 31 261
 31 207 17 31 31 268 268 ...
##  $ A_PRICE     : int  140 3933 2980 1480 4580 720 680 1680 22800 8280
 ...
##  $ P_YEAR_MONTH: Factor w/ 3 levels "2013-01","2013-02",..: 3 3 3 3 3 3
 3 3 3 3 ...
##  $ ADD_COL     : Factor w/ 1470 levels "","  "," 소"," 텃골 영양란",..: 3
28 253 1 1 1 115 1 1 76 37 ...
##  $ M_TYPE_CODE : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ M_TYPE_NAME : Factor w/ 2 levels "대형마트","전통시장": 1 1 1 1 1 1 1
1 1 1 ...
##  $ M_GU_CODE   : int  200000 200000 200000 200000 200000 200000 200000
200000 200000 200000 ...
##  $ M_GU_NAME   : Factor w/ 25 levels "강남구","강동구",..: 16 16 16 16 1
6 16 16 16 16 16 ...
```

```
# save(mkt, file=paste(getwd(), '/savtest.R', sep=''))

#
```

CLT Example

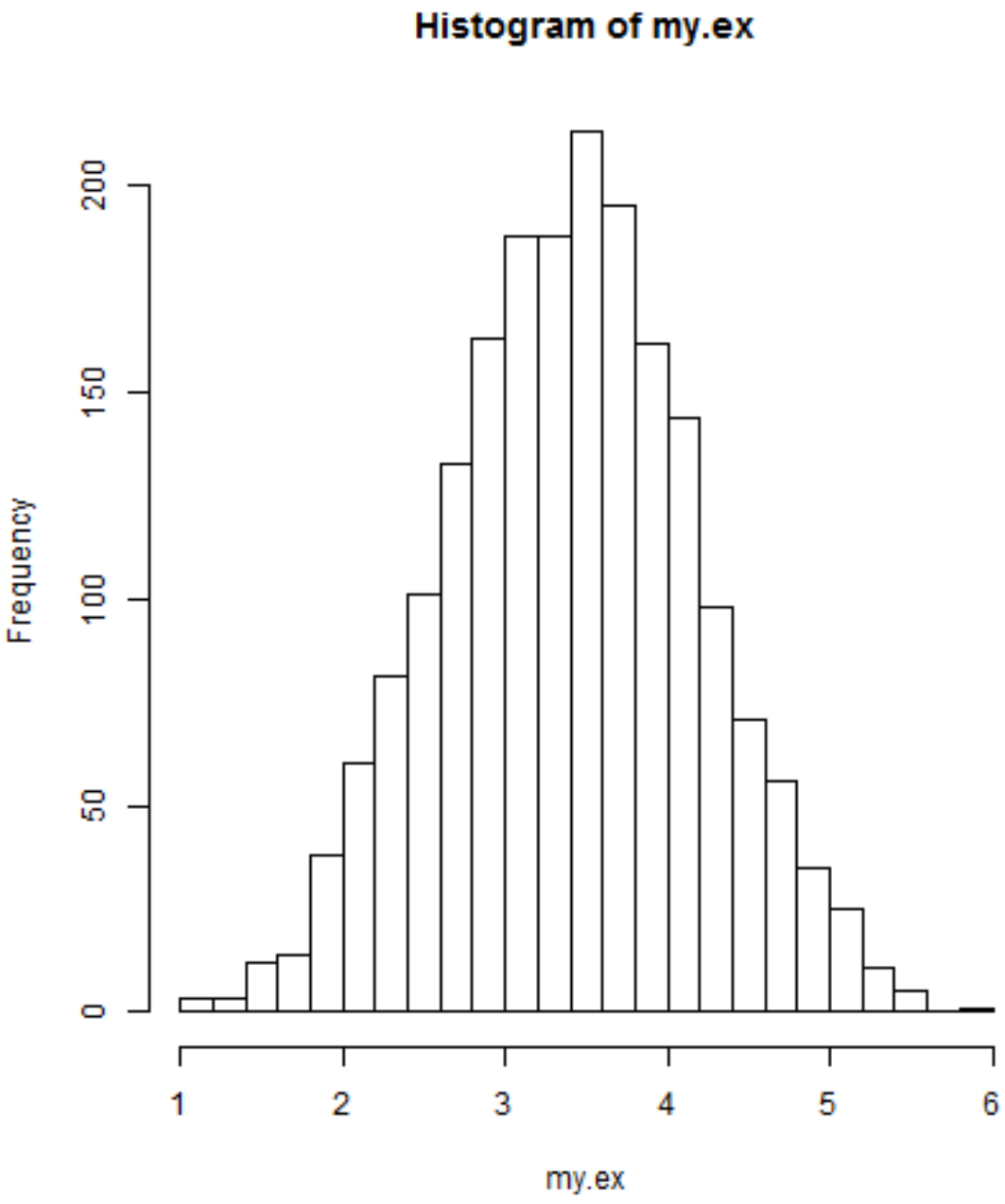
```
# dice
my.clt <- function(no.throw = 5, no.rep = 500) {
    exp.res <- matrix(sample(1:6, no.throw * no.rep, replace = TRUE), ncol
 = no.throw,
        byrow = T)
    return(apply(exp.res, 1, mean))
}

my.ex <- my.clt(5, 2000)
head(my.ex)
```

```
## [1] 2.8 3.2 3.0 4.0 3.2 5.2
```

```r
tail(my.ex)
```

```
## [1] 3.6 3.8 4.0 2.4 4.2 2.8
```

```r
hist(my.ex, nclass = 20)
```



plot of chunk unnamed-chunk-19