제3강 : 고급자료처리 및 기술통계

금융 통계 및 시계열 분석

TRADE INFORMATIX

2014년 1월 14일

Outline

1 고급 자료 처리
- 자료형 확인 및 변환 (coercion)
- 자료 분할
- Wide/Long 포맷 변환
- 데이터프레임 컬럼 분할
- 데이터프레임 머지
- 데이터프레임 컬럼 변환
- 그룹별로 데이터프레임 컬럼 변환
- apply 명령

2 기술통계
- 요약 통계
- contingency table
- flat (complex) contingency table
- margin table
- propotion table
- stem and leaf plot
- 히스토그램
- 커널 밀도 추정

자료형 확인 및 변환 (coercion)

- 자료형 확인 : `is.xxx()` 명령 사용
  - `is.na, is.nan, is.logical, is.character, is.numeric, is.integer, is.double, is.list, is.matrix, is.array, is.data.frame`
- 자료형 변환 : `as.xxx()` 명령 사용
  - `as.null, as.logical, as.character, as.numeric, as.integer, as.double, as.list, as.matrix, as.array, as.data.frame`
- 자료형 변환 방지 : `I()` 명령 사용
  - `as.data.frame`과 명령 사용시 문자열을 자동으로 factor 로 변환.
  - 이를 막기 위해서는 `stringsAsFactors=FALSE`와 같은 인수를 설정하거나 문자열 자료를 `I()` 명령으로 감싼다.

```
> y <- as.data.frame(list(a=c("a", "b", "c"), b=1:3))
> str(y)

'data.frame': 3 obs. of  2 variables:
 $ a: Factor w/ 3 levels "a","b","c": 1 2 3
 $ b: int  1 2 3

> z <- as.data.frame(I(list(a=c("a", "b", "c"), b=1:3)))
> str(z)

'data.frame': 2 obs. of  1 variable:
 $ I(list(a = c("a", "b", "c"), b = 1:3)):List of 2
  ..$ a: chr  "a" "b" "c"
  ..$ b: int  1 2 3
  ..- attr(*, "class")= chr "AsIs"
```

자료 분할 : split/cut

- ❏ `split(data, index)` : 카테고리 인덱스 벡터에 따라 분할
- ❏ `cut(data)` : 수치값에 따라 분할

```
> x <- 1:10
> y <- c(rep('a', 4), rep('b',3), rep('c', 3))
> split(x, y)

$a
[1] 1 2 3 4

$b
[1] 5 6 7

$c
[1]  8  9 10

> a <- rnorm(20)
> b <- seq(-10, 10, 2.5)
> cut(a, b)

 [1] (0,2.5]   (-2.5,0] (0,2.5]   (0,2.5]   (-2.5,0] (-2.5,0]
 [7] (0,2.5]   (0,2.5]   (2.5,5]   (0,2.5]   (0,2.5]   (-2.5,0]
[13] (-2.5,0] (0,2.5]   (-2.5,0] (0,2.5]   (0,2.5]   (-2.5,0]
[19] (-2.5,0] (-2.5,0]
8 Levels: (-10,-7.5] (-7.5,-5] (-5,-2.5] ... (7.5,10]

> a[cut(a,b)=='(0,2.5]']

 [1] 0.46376512 0.21646754 0.27989934 0.85695848 0.16636867
 [6] 0.88472046 0.02850251 0.79127813 0.97709731 0.79272146
```

Wide/Long 포맷

❏ Wide 포맷 : 카테고리값 컬럼이 별도로 존재

	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

Wide/Long 포맷 (계속)

❏ Long 포맷 : 특정한 컬럼의 팩터/스트링 값이 필드 이름 역할

```
                  id spec value
1          Mazda RX4  mpg  21.0
2      Mazda RX4 Wag  mpg  21.0
3         Datsun 710  mpg  22.8
4     Hornet 4 Drive  mpg  21.4
5  Hornet Sportabout  mpg  18.7
6          Mazda RX4  cyl   6.0
7      Mazda RX4 Wag  cyl   6.0
8         Datsun 710  cyl   4.0
9     Hornet 4 Drive  cyl   6.0
10 Hornet Sportabout  cyl   8.0
11         Mazda RX4 disp 160.0
12     Mazda RX4 Wag disp 160.0
13        Datsun 710 disp 108.0
14    Hornet 4 Drive disp 258.0
15 Hornet Sportabout disp 360.0
16         Mazda RX4   hp 110.0
17     Mazda RX4 Wag   hp 110.0
18        Datsun 710   hp  93.0
19    Hornet 4 Drive   hp 110.0
20 Hornet Sportabout   hp 175.0
```

Wide → Long 포맷 변환

- `melt(data, id.vars, variable.name, value.name)`
  - `data` : 입력 자료
  - `id.vars` : 아이디 컬럼 이름
  - `variable.name` : 필드명 컬럼 이름
  - `value.name` : 필드값 컬럼 이름
- Wide → Long 변환. reshape2 패키지

```
> library(reshape2)
> x <- mtcars[1:5,]
> x$id <- rownames(x)
> y <- melt(x, id.vars="id", variable.name="spec", value.name="value")
> head(y, 15)

                  id spec value
1          Mazda RX4  mpg  21.0
2      Mazda RX4 Wag  mpg  21.0
3         Datsun 710  mpg  22.8
4     Hornet 4 Drive  mpg  21.4
5  Hornet Sportabout  mpg  18.7
6          Mazda RX4  cyl   6.0
7      Mazda RX4 Wag  cyl   6.0
8         Datsun 710  cyl   4.0
9     Hornet 4 Drive  cyl   6.0
10 Hornet Sportabout  cyl   8.0
11         Mazda RX4 disp 160.0
12     Mazda RX4 Wag disp 160.0
13        Datsun 710 disp 108.0
14    Hornet 4 Drive disp 258.0
15 Hornet Sportabout disp 360.0
```

Long → Wide 포맷 변환

- `dcast(data, id.columns ~ field.columns, value.var)`
  - `data` : 입력 자료
  - `id.columns` : 아이디 컬럼들 (+로 결합)
  - `field.columns` : 필드 컬럼들 (+로 결합)
  - `value.var` : 값 컬럼
- Long → Wide 변환. reshape2 패키지

```
> library(reshape2)
> library(gcookbook)
> head(plum)

  length      time survival count
1   long   at_once     dead    84
2   long in_spring     dead   156
3  short   at_once     dead   133
4  short in_spring     dead   209
5   long   at_once    alive   156
6   long in_spring    alive    84

> dcast(plum, length + time ~ survival, value.var="count")

  length      time dead alive
1   long   at_once   84   156
2   long in_spring  156    84
3  short   at_once  133   107
4  short in_spring  209    31

> dcast(plum, time ~ length + survival, value.var="count")

       time long_dead long_alive short_dead short_alive
1   at_once        84        156        133         107
2 in_spring       156         84        209          31
```

with/attach/detach를 이용한 데이터프레임 컬럼 분할

- 데이터 프레임의 열 (column/리스트의 원소 등을 접근할 때 데이터 프레임이나 리스트 이름을 쓰지 않도록 하여 코드 가독성 향상
- `with(data, expression)` : express 사용시 data 이름 생략
- `attach(data)` : 워크스페이스에 컬럼 분할하여 로드
- `detach(data)` : 워크스페이스에 로드된 컬럼 삭제

```
> tapply(CO2$conc, CO2$Treatment, table)
> with(CO2, tapply(conc, Treatment, table))
> attach(CO2);tapply(conc, Treatment, table)
```

R 데이터프레임 머지 (merge)

- `merge(x, y, by.x, by.y)`
  - `x, y` : 데이터프레임
  - `by.x, by.y` : 각 데이터프레임에서 join할 컬럼 이름
- Relationship Database 에서의 TABLE JOIN과 같음

```
> (authors <- data.frame(
+   surname = I(c("Tukey", "Venables", "Tierney", "Ripley", "McNeil")),
+   nationality = c("US", "Australia", "US", "UK", "Australia"),
+   deceased = c("yes", rep("no", 4))))

   surname nationality deceased
1    Tukey          US      yes
2 Venables   Australia       no
3  Tierney          US       no
4   Ripley          UK       no
5   McNeil   Australia       no

> (books <- data.frame(
+   name = I(c("Tukey", "Venables", "Tierney", "Ripley", "Ripley", "McNeil", "R Core")),
+   title = c("Exploratory Data Analysis", "Modern Applied Statistics ...",
+             "LISP-STAT", "Spatial Statistics", "Stochastic Simulation",
+             "Interactive Data Analysis", "An Introduction to R"),
+   other.author = c(NA, "Ripley", NA, NA, NA, NA, "Venables & Smith")))

      name                         title     other.author
1    Tukey     Exploratory Data Analysis             <NA>
2 Venables Modern Applied Statistics ...           Ripley
3  Tierney                     LISP-STAT             <NA>
4   Ripley            Spatial Statistics             <NA>
5   Ripley         Stochastic Simulation             <NA>
6   McNeil     Interactive Data Analysis             <NA>
7   R Core           An Introduction to R Venables & Smith
```

R 데이터프레임 머지 (merge) (계속)

```
> merge(authors, books, by.x = "surname", by.y = "name")

   surname nationality deceased
1   McNeil   Australia       no
2   Ripley          UK       no
3   Ripley          UK       no
4  Tierney          US       no
5    Tukey          US      yes
6 Venables   Australia       no
                         title other.author
1     Interactive Data Analysis         <NA>
2            Spatial Statistics         <NA>
3         Stochastic Simulation         <NA>
4                     LISP-STAT         <NA>
5     Exploratory Data Analysis         <NA>
6 Modern Applied Statistics ...       Ripley

> merge(books, authors, by.x = "name", by.y = "surname")

      name                         title other.author
1   McNeil     Interactive Data Analysis         <NA>
2   Ripley             Spatial Statistics         <NA>
3   Ripley          Stochastic Simulation         <NA>
4  Tierney                      LISP-STAT         <NA>
5    Tukey     Exploratory Data Analysis         <NA>
6 Venables Modern Applied Statistics ...       Ripley
  nationality deceased
1   Australia       no
2          UK       no
3          UK       no
4          US       no
5          US      yes
6   Australia       no
```

R 데이터프레임 머지 (merge) (계속 2)

```
> (x <- data.frame(k1=c(NA,NA,3,4,5), k2=c(1,NA,NA,4,5), data=1:5))

  k1 k2 data
1 NA  1    1
2 NA NA    2
3  3 NA    3
4  4  4    4
5  5  5    5

> (y <- data.frame(k1=c(NA,2,NA,4,5), k2=c(NA,NA,3,4,5), data=1:5))

  k1 k2 data
1 NA NA    1
2  2 NA    2
3 NA  3    3
4  4  4    4
5  5  5    5

> merge(x, y, by=c("k1","k2")) # NA's match

  k1 k2 data.x data.y
1  4  4      4      4
2  5  5      5      5
3 NA NA      2      1
```

R 데이터프레임 머지 (merge) (계속 3)

```
> merge(x, y, by=c("k1","k2"), incomparables=NA)

  k1 k2 data.x data.y
1  4  4      4      4
2  5  5      5      5
3 NA NA      2      1

> merge(x, y, by="k1") # NA's match, so 6 rows

  k1 k2.x data.x k2.y data.y
1  4    4      4    4      4
2  5    5      5    5      5
3 NA    1      1   NA      1
4 NA    1      1    3      3
5 NA   NA      2   NA      1
6 NA   NA      2    3      3

> merge(x, y, by="k2", incomparables=NA) # 2 rows

  k2 k1.x data.x k1.y data.y
1  4    4      4    4      4
2  5    5      5    5      5
```

transform를 이용한 데이터프레임 컬럼 변환

- transform(data, ...)
- 데이터 프레임의 특정 열값을 수치적으로 변환하거나 새로운 열을 추가

```
> head(airquality)

  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6

> head(transform(airquality, Ozone = -Ozone))

  Ozone Solar.R Wind Temp Month Day
1   -41     190  7.4   67     5   1
2   -36     118  8.0   72     5   2
3   -12     149 12.6   74     5   3
4   -18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6   -28      NA 14.9   66     5   6

> head(transform(airquality, new = -Ozone, Temp = (Temp-32)/1.8))

  Ozone Solar.R Wind     Temp Month Day new
1    41     190  7.4 19.44444     5   1 -41
2    36     118  8.0 22.22222     5   2 -36
3    12     149 12.6 23.33333     5   3 -12
4    18     313 11.5 16.66667     5   4 -18
5    NA      NA 14.3 13.33333     5   5  NA
6    28      NA 14.9 18.88889     5   6 -28
```

그룹별로 데이터프레임 컬럼 변환

- ddply(data, category.column, transform, ...)
  - category.column : 그룹 카테고리 컬럼 지정
- plyr 패키지

```
> library(MASS)
> library(plyr)
> head(cabbages)

  Cult Date HeadWt VitC
1  c39  d16    2.5   51
2  c39  d16    2.2   55
3  c39  d16    3.1   45
4  c39  d16    4.3   42
5  c39  d16    2.5   53
6  c39  d16    4.3   50

> head(transform(cabbages, DevWt=HeadWt-mean(HeadWt)))

  Cult Date HeadWt VitC       DevWt
1  c39  d16    2.5   51 -0.09333333
2  c39  d16    2.2   55 -0.39333333
3  c39  d16    3.1   45  0.50666667
4  c39  d16    4.3   42  1.70666667
5  c39  d16    2.5   53 -0.09333333
6  c39  d16    4.3   50  1.70666667

> head(ddply(cabbages, "Cult", transform, DevWt=HeadWt-mean(HeadWt)))

  Cult Date HeadWt VitC      DevWt
1  c39  d16    2.5   51 -0.4066667
2  c39  d16    2.2   55 -0.7066667
3  c39  d16    3.1   45  0.1933333
4  c39  d16    4.3   42  1.3933333
5  c39  d16    2.5   53 -0.4066667
6  c39  d16    4.3   50  1.3933333
```

apply 명령을 이용한 implicit loop

- 단일 명령어를 반복적으로 수행하는 경우에는 명시적으로 loop를 사용하는 것보다 `apply` 명령어를 사용하는 것이 빠르다.
- `apply` 명령의 종류
  - `apply(X, MARGIN, FUN)` : 행렬/어레이에서 루프 방향을 지정
  - `lapply(X, FUN)` : 벡터/리스트에서 원소별로 루프
  - `sapply(X, FUN)` : `lapply` 결과를 벡터로 변환
  - `vapply(FUN, ...)` : apply 결과의 초기값을 미리 지정
  - `mapply(FUN, ...)` : 복수 개의 벡터/리스트 루프
  - `tapply(X, INDEX, FUN)` : 카테고리별로 루프
  - `by(FUN, ...)` : 데이터프레임 입력
  - `aggregate(FUN, ...)` : 데이터프레임 출력

apply 명령

- `apply(X, MARGIN, FUN)`
  - `X` : 2차원 이상의 행렬/어레이/데이터프레임
  - `MARGIN` : 1이면 행, 2이면 열
  - `FUN` : 적용할 함수
- 2차원 이상의 행렬/어레이/데이터프레임에서 행/열을 지정하여 반복작업

```
> (x <- cbind(x1 = 3, x2 = c(4:1, 2:5)))

     x1 x2
[1,]  3  4
[2,]  3  3
[3,]  3  2
[4,]  3  1
[5,]  3  2
[6,]  3  3
[7,]  3  4
[8,]  3  5

> apply(x, 1, sum)

[1] 7 6 5 4 5 6 7 8

> apply(x, 2, sum)

x1 x2
24 24
```

apply 명령 (계속)

```
> (x <- cbind(x1 = 3, x2 = c(4:1, 2:5)))

     x1 x2
[1,]  3  4
[2,]  3  3
[3,]  3  2
[4,]  3  1
[5,]  3  2
[6,]  3  3
[7,]  3  4
[8,]  3  5

> apply(x, 1, sort)

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

> apply(x, 2, sort)

     x1 x2
[1,]  3  1
[2,]  3  2
[3,]  3  2
[4,]  3  3
[5,]  3  3
[6,]  3  4
[7,]  3  4
[8,]  3  5
```

apply 명령 (계속)

```
> (z <- array(1:24, dim=2:4))

, , 1

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

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12

, , 3

     [,1] [,2] [,3]
[1,]   13   15   17
[2,]   14   16   18

, , 4

     [,1] [,2] [,3]
[1,]   19   21   23
[2,]   20   22   24

> apply(z, 1:2, function(x) max(x))
     [,1] [,2] [,3]
[1,]   19   21   23
[2,]   20   22   24
```

lapply 명령

- `lapply(X, FUN)`
  - `X` : 리스트
  - `FUN` : 적용할 함수. 함수에 적용할 인수는 함수 뒤에
- 리스트의 각 원소에 대해 루프를 생성, 함수 적용. 리스트 자료형 반환

```
> (x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE)))

$a
 [1]  1  2  3  4  5  6  7  8  9 10

$beta
[1]  0.04978707  0.13533528  0.36787944  1.00000000
[5]  2.71828183  7.38905610 20.08553692

$logic
[1]  TRUE FALSE FALSE  TRUE

> lapply(x, mean)

$a
[1] 5.5

$beta
[1] 4.535125

$logic
[1] 0.5
```

lapply 명령 (계속)

```
> (x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE)))

$a
 [1]  1  2  3  4  5  6  7  8  9 10

$beta
[1]  0.04978707  0.13533528  0.36787944  1.00000000
[5]  2.71828183  7.38905610 20.08553692

$logic
[1]  TRUE FALSE FALSE  TRUE

> lapply(x, quantile, probs = 1:3/4)

$a
 25%  50%  75%
3.25 5.50 7.75

$beta
      25%       50%       75%
0.2516074 1.0000000 5.0536690

$logic
25% 50% 75%
0.0 0.5 1.0
```

sapply 명령

- `sapply(X, FUN)`
  - `lapply`의 결과를 리스트가 아닌 행렬이나 벡터로 단순화(simplify)하여 반환

```
> (x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE)))

$a
 [1]  1  2  3  4  5  6  7  8  9 10

$beta
[1]  0.04978707  0.13533528  0.36787944  1.00000000
[5]  2.71828183  7.38905610 20.08553692

$logic
[1]  TRUE FALSE FALSE  TRUE

> sapply(x, quantile, probs = 1:3/4)

       a      beta logic
25% 3.25 0.2516074   0.0
50% 5.50 1.0000000   0.5
75% 7.75 5.0536690   1.0
```

vapply 명령

- `vapply(X, FUN, FUN.VALUE)`
  - `X` : 벡터/리스트
  - `FUN` : 적용할 함수
  - `FUN.VALUE` : apply 결과의 초기값
- apply 결과의 자료형, 포맷이나 초기값을 미리 정해놓음

```
> (x <- sapply(3:9, seq))
[[1]]
[1] 1 2 3

[[2]]
[1] 1 2 3 4

[[3]]
[1] 1 2 3 4 5

[[4]]
[1] 1 2 3 4 5 6

[[5]]
[1] 1 2 3 4 5 6 7

[[6]]
[1] 1 2 3 4 5 6 7 8

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

```
> sapply(x, fivenum)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]  1.0  1.0    1  1.0  1.0  1.0    1
[2,]  1.5  1.5    2  2.0  2.5  2.5    3
[3,]  2.0  2.5    3  3.5  4.0  4.5    5
[4,]  2.5  3.5    4  5.0  5.5  6.5    7
[5,]  3.0  4.0    5  6.0  7.0  8.0    9

> vapply(x, fivenum,
+   c(Min. = 0,
+     "1st Qu." = 0,
+     Median = 0,
+     "3rd Qu." = 0,
+     Max. = 0))
        [,1] [,2] [,3] [,4] [,5] [,6] [,7]
Min.     1.0  1.0    1  1.0  1.0  1.0    1
1st Qu.  1.5  1.5    2  2.0  2.5  2.5    3
Median   2.0  2.5    3  3.5  4.0  4.5    5
3rd Qu.  2.5  3.5    4  5.0  5.5  6.5    7
Max.     3.0  4.0    5  6.0  7.0  8.0    9
```

mapply 명령

- `mapply(FUN, ...)`
  - `FUN` : 적용할 함수
  - `...` : 복수개의 벡터/리스트
- 복수개의 벡터/리스트의 n번째 원소들끼리 묶은뒤 루프 생성

```
> '+'(1,2)

[1] 3

> rbind(11:14, 1:4)

     [,1] [,2] [,3] [,4]
[1,]   11   12   13   14
[2,]    1    2    3    4

> mapply('+', 11:14, 1:4)

[1] 12 14 16 18

> word <- function(C,k) paste(rep(C,k), collapse = '')
> utils::str(mapply(word, LETTERS[1:6], 6:1, SIMPLIFY = FALSE))

List of 6
 $ A: chr "AAAAAA"
 $ B: chr "BBBBB"
 $ C: chr "CCCC"
 $ D: chr "DDD"
 $ E: chr "EE"
 $ F: chr "F"
```

tapply 명령

- `tapply(X, INDEX, FUN)`
  - `X` : 자료 벡터
  - `INDEX` : 인덱스 벡터
  - `FUN` : 적용할 함수
- 인덱스 벡터값이 같은 것끼리 묶은 뒤 루프 생성, 함수 적용

```
> n <- 17
> fac <- factor(rep(1:3, length=n), levels = 1:5)
> rbind(1:n, fac)

    [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
       1    2    3    4    5    6    7    8    9    10
fac    1    2    3    1    2    3    1    2    3     1
    [,11] [,12] [,13] [,14] [,15] [,16] [,17]
       11    12    13    14    15    16    17
fac     2     3     1     2     3     1     2

> tapply(1:n, fac, sum)

 1  2  3  4  5
51 57 45 NA NA

> (1:n)[fac==1]

[1]  1  4  7 10 13 16

> sum((1:n)[fac==1])

[1] 51
```

by 명령

- `by(df, INDEX, FUN)`
  - 리스트 자료형 반환
  - `X` : 데이터프레임
  - `INDEX` : 인덱스 벡터/데이터프레임
  - `FUN` : 적용할 함수
- 입력자료가 데이터프레임이거나 함수가 데이터프레임을 요구하는 경우

```
> head(CO2)

  Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4
3   Qn1 Quebec nonchilled  250   34.8
4   Qn1 Quebec nonchilled  350   37.2
5   Qn1 Quebec nonchilled  500   35.3
6   Qn1 Quebec nonchilled  675   39.2

> tapply(CO2[,4], CO2[,3], summary)

$nonchilled
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     95     175     350     435     675    1000

$chilled
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     95     175     350     435     675    1000
```

by 명령 (계속)

```
> by(CO2[,-3], CO2[,3], summary)
CO2[, 3]: nonchilled
     Plant            Type         conc          uptake
 Qn1    :7   Quebec     :21   Min.   :  95   Min.   :10.60
 Qn2    :7   Mississippi:21   1st Qu.: 175   1st Qu.:26.48
 Qn3    :7                    Median : 350   Median :31.30
 Mn3    :7                    Mean   : 435   Mean   :30.64
 Mn2    :7                    3rd Qu.: 675   3rd Qu.:38.70
 Mn1    :7                    Max.   :1000   Max.   :45.50
 (Other):0
------------------------------------------------------------
CO2[, 3]: chilled
     Plant            Type         conc          uptake
 Qc1    :7   Quebec     :21   Min.   :  95   Min.   : 7.70
 Qc3    :7   Mississippi:21   1st Qu.: 175   1st Qu.:14.53
 Qc2    :7                    Median : 350   Median :19.70
 Mc2    :7                    Mean   : 435   Mean   :23.78
 Mc3    :7                    3rd Qu.: 675   3rd Qu.:34.90
 Mc1    :7                    Max.   :1000   Max.   :42.40
 (Other):0
```

aggregate 명령

- `aggregate(x, INDEX, FUN)`
  - `X` : 입력 자료
  - `INDEX` : 인덱스 벡터
  - `FUN` : 적용할 함수
- `tapply`와 유사. 결과가 데이터프레임 자료형으로 반환

```
> class(state.x77)
[1] "matrix"
> head(state.x77)
           Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
California      21198   5114        1.1    71.71   10.3    62.6    20 156361
Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
> aggregate(state.x77, list(Region = state.region), mean)
         Region Population   Income Illiteracy Life Exp    Murder  HS Grad    Frost      Area
1     Northeast   5495.111 4570.222   1.000000 71.26444  4.722222 53.96667 132.7778  18141.00
2         South   4208.125 4011.938   1.737500 69.70625 10.581250 44.34375  64.6250  54605.12
3 North Central   4803.000 4611.083   0.700000 71.76667  5.275000 54.51667 138.8333  62652.00
4          West   2915.308 4702.615   1.023077 71.23462  7.215385 62.00000 102.1538 134463.00
```

기술통계 (descriptive statistics)

- 기술통계 (descriptive statistics)
  - 샘플에 대한 확률적 모형을 가정하지 않고 순수하게 샘플 집합 자체의 특성만을 기술 (describe)
- 추리통계 (inferential statistics)
  - 모집단 혹은 확률 분포 모형에 대한 정보를 이끌어내는 통계학
- 기술 방법
  - 요약 통계 (summary statistics)
  - 분할표 (table)
  - 그래픽 (histogram, density plot)

요약통계

- 샘플 전체/부분을 나열하는 경우
  - `print()`, `head()`, `tail()` 명령 사용
- 샘플의 특성을 요약하는 경우
  - `summary()`, `quantile()`, `fivenum()` 명령 사용
  - 자료의 열(column) 별로 최소값(`min()`), 1분위수, 중간값(`median()`), 평균(`mean()`), 3분위수, 최대값(`max()`) 계산
  - `quantile()`, `fivenum()` 은 `summary()`와 달리 벡터입력만을 받을 수 있음
  - `quantile()`은 (0,0.25,0.5,0.75,1) quantile 출력 `fivenum()`은 quartile 출력. 값이 다를 수 있음

```
> head(CO2, n=2)

  Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4

> tail(CO2, n=1)

   Plant        Type Treatment conc uptake
84   Mc3 Mississippi   chilled 1000   19.9

> summary(CO2)

     Plant              Type         Treatment       conc          uptake
 Qn1    : 7   Quebec      :42   nonchilled:42   Min.   :  95   Min.   : 7.70
 Qn2    : 7   Mississippi:42   chilled   :42   1st Qu.: 175   1st Qu.:17.90
 Qn3    : 7                                    Median : 350   Median :28.30
 Qc1    : 7                                    Mean   : 435   Mean   :27.21
 Qc3    : 7                                    3rd Qu.: 675   3rd Qu.:37.12
 Qc2    : 7                                    Max.   :1000   Max.   :45.50
 (Other):42
```

분위수 (Quantile)

- q-분위수의 개념은 전체 샘플 집합을 q개의 같은 갯수의 서로 다른 부분 집합으로 분할했을 때 경계가 되는 수
- q-분위수의 갯수는 q-1개
- 유한개의 샘플집합에서의 일반적 정의 : k번째 q-분위수는 그 보다 큰 샘플의 수가 전체의 k/q 비율 이하가 되도록 하는 가장 큰 수
- 확률분포가 존재하는 경우 : 갯수가 아닌 확률이 같아지도록 분할
- `quantile(x, prob, type)` 명령으로 구함
  - `x` : 샘플 집합
  - `probe` : 경계값 벡터 (0부터 1사이 실수 벡터)
  - `type` : 분위수 정의/계산 알고리즘 선택 (9가지)

```
> quantile(1:10, c(0, 0.5, 1))

  0%  50% 100%
 1.0   5.5  10.0

> quantile(rnorm(1000), seq(0, 1, 0.1))

         0%         10%         20%         30%         40%
-2.94198542 -1.22552214 -0.80555999 -0.53989336 -0.29434825
        50%         60%         70%         80%         90%
-0.02943144  0.22127934  0.49455713  0.89417602  1.32292765
       100%
 2.73074651
```

개별 기술통계 명령

- ❏ `length(x)` : 샘플 갯수
- ❏ `max(x, na.rm=FALSE)` : 최대값
- ❏ `min(x, na.rm=FALSE)` : 최소값
- ❏ `which.min(x)` : 최대값 위치
- ❏ `which.max(x)` : 최소값 위치
- ❏ `mean(x, na.rm=FALSE)` : 산술 평균
- ❏ `median(x, na.rm=FALSE)` : 중간값
- ❏ `sum(x, na.rm=FALSE)` : 합계
- ❏ `sd(x, na.rm=FALSE)` : 표준편차 (standar deviation)
- ❏ `var(x, na.rm=FALSE)` : 분산 (variance)
- ❏ `mad(x, na.rm=FALSE)` : 중간값절대변차 (median absolute deviation)[1]

$$\text{MAD} = \text{median}_i \left(|X_i - \text{median}_j(X_j)| \right) \tag{1}$$

분할표 (contingency table)

- 가능한 값의 수가 유한한 카테고리 자료(categorical data)에 대해 동일한 값이 나오는 빈도를 조사한 표
- 실수값을 가지는 자료의 경우에는 `cut()` 명령어 등으로 분할하여 조사하거나 stem/leaf plot, histogram 등을 사용
- `table(x, ...)` 명령 사용
  - 단일 벡터 입력의 경우에는 1차원 자료 출력
  - 2개의 벡터를 입력한 경우에는 2차원 자료 출력
  - n개의 벡터를 입력한 경우에는 n차원 자료 출력
  - 출력 자료형은 table 클래스

```
> x <- rpois(100,5)
> x
  [1]  7  7  6  6  3  4  9  5  4  5  6  2  8  9  4  2  4  7
 [19]  7  4  3  7  5  8  6  7 10  5  5  4  4  4  8  9  7  4
 [37]  1  4  5  2  4  4  7  7  8  2  1  4  8  6 11  3  3  5
 [55]  4  4  4  9  5  5 11  5  3  5  4  9  3  5  0  1  8  4
 [73]  4  2  4  4  4  6  5  5  4  3  3  6  4  7  4  5  3  3
 [91]  4  6  6  8  6  7  3  3  4  1

> table(x)

x
 0  1  2  3  4  5  6  7  8  9 10 11
 1  4  5 12 27 15 10 11  7  5  1  2
```

분할표 (contingency table) 계속 1

```
> head(data.frame(state.division, state.region), n=3)

      state.division state.region
1 East South Central        South
2            Pacific         West
3           Mountain         West

> table(state.division)

state.division
       New England    Middle Atlantic     South Atlantic
                 6                  3                  8
East South Central West South Central East North Central
                 4                  4                  5
West North Central           Mountain            Pacific
                 7                  8                  5

> table(state.region)

state.region
    Northeast         South North Central          West
            9            16            12            13

> table(state.division, state.region)

                    state.region
state.division       Northeast South North Central West
  New England                6     0             0    0
  Middle Atlantic            3     0             0    0
  South Atlantic             0     8             0    0
  East South Central         0     4             0    0
  West South Central         0     4             0    0
  East North Central         0     0             5    0
  West North Central         0     0             7    0
  Mountain                   0     0             0    8
  Pacific                    0     0             0    5
```

분할표 (contingency table) 계속 2

```
> head(airquality)

  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6

> attach(airquality)
> quantile(Temp)

  0%  25%  50%  75% 100%
  56   72   79   85   97

> head(cut(Temp, quantile(Temp)))

[1] (56,72] (56,72] (72,79] (56,72] <NA>    (56,72]
Levels: (56,72] (72,79] (79,85] (85,97]

> table(cut(Temp, quantile(Temp)), Month)

         Month
           5  6  7  8  9
  (56,72] 24  3  0  1 10
  (72,79]  5 15  2  9 10
  (79,85]  1  7 19  7  5
  (85,97]  0  5 10 14  5
```

분할표 (contingency table) 계속 3

```
> head(CO2)

  Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4
3   Qn1 Quebec nonchilled  250   34.8
4   Qn1 Quebec nonchilled  350   37.2
5   Qn1 Quebec nonchilled  500   35.3
6   Qn1 Quebec nonchilled  675   39.2

> with(CO2, table(conc, Type, Treatment))

, , Treatment = nonchilled

      Type
conc   Quebec Mississippi
  95        3           3
  175       3           3
  250       3           3
  350       3           3
  500       3           3
  675       3           3
  1000      3           3

, , Treatment = chilled

      Type
conc   Quebec Mississippi
  95        3           3
  175       3           3
  250       3           3
  350       3           3
  500       3           3
  675       3           3
  1000      3           3
```

xtabs 명령

❑ `xtabs(formula, data)` 명령은 벡터 자료가 아닌 formula를 사용

```
> with(CO2, xtabs(~ conc + Type + Treatment))
, , Treatment = nonchilled

      Type
conc   Quebec Mississippi
  95        3           3
  175       3           3
  250       3           3
  350       3           3
  500       3           3
  675       3           3
  1000      3           3

, , Treatment = chilled

      Type
conc   Quebec Mississippi
  95        3           3
  175       3           3
  250       3           3
  350       3           3
  500       3           3
  675       3           3
  1000      3           3
```

Flat (complex) contingency table

- 3차원 이상의 table에 대해 어레이가 아닌 복합 컬럼 인덱스 사용
- `ftable(x, ...)` 명령 사용

```
> with(CO2, ftable(conc, Type, Treatment))
                Treatment nonchilled chilled
conc Type
95   Quebec                         3       3
     Mississippi                    3       3
175  Quebec                         3       3
     Mississippi                    3       3
250  Quebec                         3       3
     Mississippi                    3       3
350  Quebec                         3       3
     Mississippi                    3       3
500  Quebec                         3       3
     Mississippi                    3       3
675  Quebec                         3       3
     Mississippi                    3       3
1000 Quebec                         3       3
     Mississippi                    3       3
```

margin table

- 축방향의 합 (marginal sum) 을 구함
- `margin.table(x, margin)` 명령 사용
  - `margin=1`은 행방향. `margin=2`은 열방향

```
> m <- matrix(1:4,2)
> m
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> margin.table(m,1)
[1] 4 6
> margin.table(m,2)
[1] 3 7
```

propotion table

- 값의 갯수가 아니라 marginal sum에 대한 비율 계산
- `prop.table(x, ...)` 명령 사용

```
> m <- matrix(1:4,2)
> m
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> prop.table(m,1)
          [,1]      [,2]
[1,] 0.2500000 0.7500000
[2,] 0.3333333 0.6666667
> prop.table(m,2)
          [,1]      [,2]
[1,] 0.3333333 0.4285714
[2,] 0.6666667 0.5714286
```

stem and leaf plot

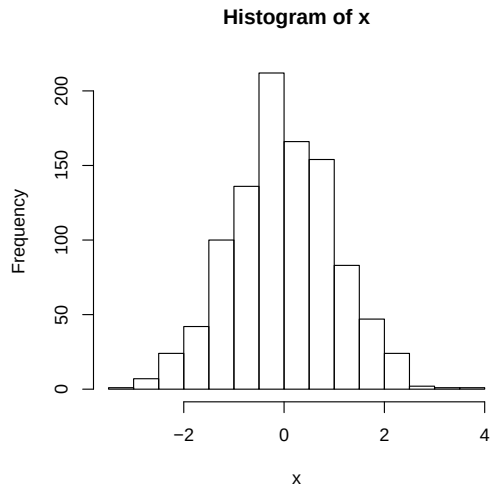
- ❏ `stem(data)` 명령 사용
- ❏ 숫자 character를 이용한 일종의 histogram
- ❏ 빈도 분포와 개별 샘플을 동시에 알 수 있음
- ❏ 기준 자릿수 이상은 중복 표시

```
> set.seed(1)
> x1 <- round(runif(20) * 10)
> x1
 [1]  3  4  6  9  2  9  9  7  6  1  2  2  7  4  8  5  7 10  4  8
> stem(x1)

  The decimal point is at the |

   0 | 0
   2 | 0000
   4 | 0000
   6 | 00000
   8 | 00000
  10 | 0
> x2 <- round(runif(20, 0, 0.3) * 10, 1)
> x2
 [1] 2.8 0.6 2.0 0.4 0.8 1.2 0.0 1.1 2.6 1.0 1.4 1.8 1.5 0.6 2.5 2.0 2.4 0.3 2.2 1.2
> stem(x2)

  The decimal point is at the |

  0 | 034668
  1 | 0122458
  2 | 0024568
```

히스토그램 작성

- `hist` 명령 사용
- 샘플 값을 특정 구간(bucket)들로 나누어 각 구간의 상대적인 샘플 발생 빈도를 계산

```
> x <- rnorm(1000)
> hist(x)
```



- 설정

```
> hist(x, breaks='st')  # Sturges algorithm
> hist(x, breaks='sc')  # Scott algorithm
> hist(x, breaks='fr')  # Freedman-Diaconis
> hist(x, breaks=c(-2,0,2))
> hist(x, freq=FALSE)  # densities, not frequencies
```

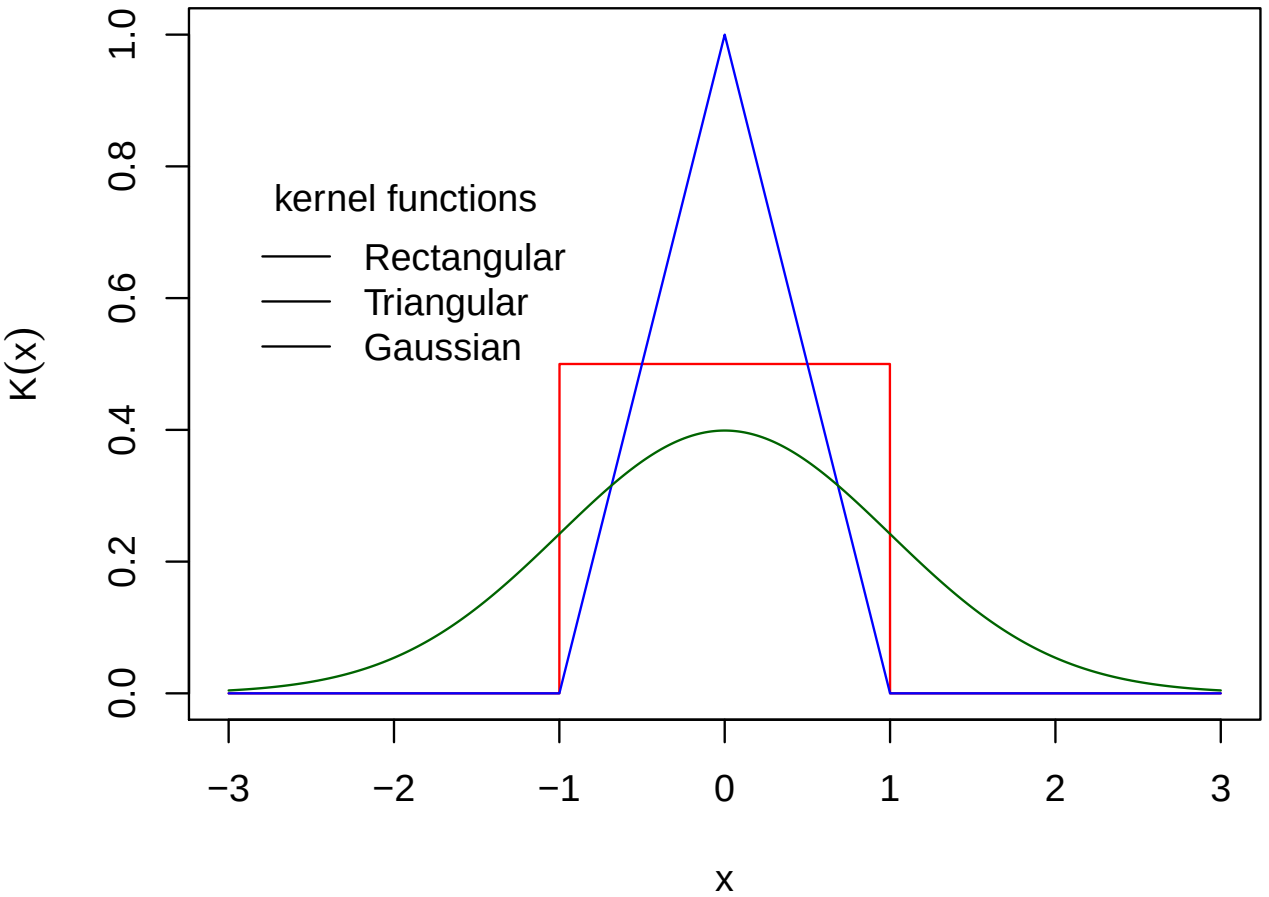
커널 밀도 (Kenel Density) 추정

- `density` 명령 사용
- 히스토그램과 달리 연속적인 밀도 추정이 가능
- 확률 분포가 특정한 커널 함수의 합으로 이루어져 있다고 가정
- 커널 함수는 폭 (bandwidth) 을 조정가능하며 전체 발생빈도를 가장 적절하게 근사화할 수 있는 폭을 계산
  - x : 확률변수의 값
  - x_j : j 번째 샘플의 값
  - n : 샘플 갯수
  - b : 폭
  - $K(\cdot)$: 커널 함수
  - $f(x)$: 추정된 확률 분포 함수

$$f(x) = \frac{1}{nb} \sum_{j=1}^{n} K\left(\frac{x - x_j}{b}\right)$$

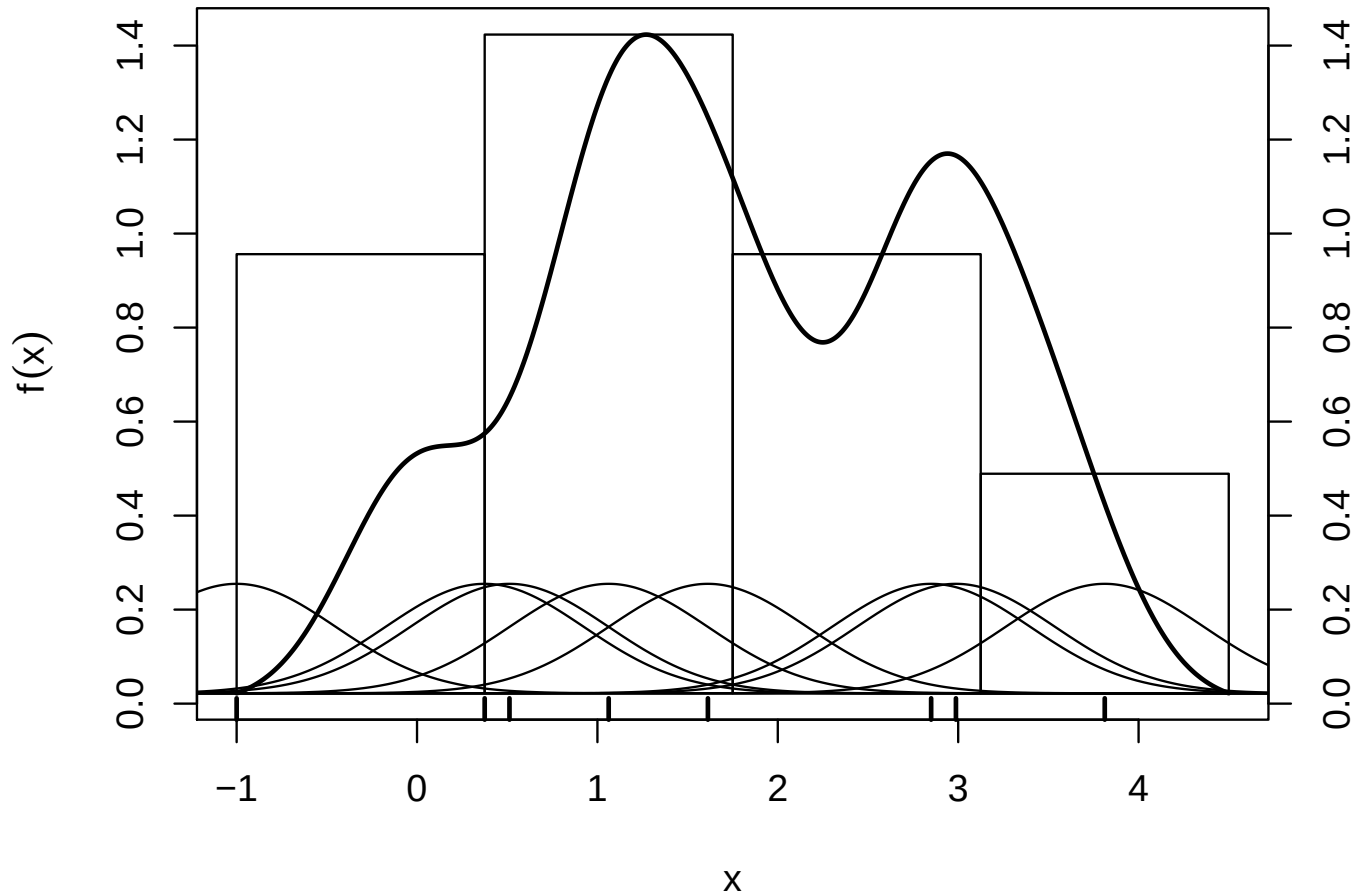
커널 함수의 종류

- ❏ 가우시안 (Gaussian) : 가장 일반적으로 사용됨
- ❏ 사각형 (rectangular)
- ❏ 삼각형 (triangular)
- ❏ Epanechnikov : 최적 커널



커널 밀도(Kenel Density) 추정 예제

```
> x <- c(0, 1, 1.1, 1.5, 1.9, 2.8, 2.9, 3.5)
```



커널 폭 계산 알고리즘

- 최적 커널 폭 : 분포의 오차 최소화
  - 평균 적분 오차 (Mean Interaged Square Error)

$$MISE = E\left[\int \left|\hat{f}(x;b) - f(x)\right|^2 dx\right]$$

  - asymptotic expansion으로 최소 MISE에 대한 b값을 구하면

$$b^*_{MISE} = \left[\frac{\int K^2}{n\int (f'')^2 (\int x^2 K)^2}\right]^{1/5}$$

- 가우시안 커널에 대한 계산 결과
  - `width="nrd0"` (추천 파라미터)

$$b = 1.06 \min(\sigma, R/1.34) n^{-1/5}$$

  - `width="sj"` (R의 디폴트 파라미터)

$$b = 0.9 \min(\sigma, R/1.34) n^{-1/5}$$