3.R

limdr

Sun Aug 26 23:06:07 2018

```
# 3장 데이터 시각화 함수

# 3.1.2 벡터라이제이션 : 벡터의 연산 과정인 입력, 연산, 출력에서 벡터를 원소 하나씩 반복적으로 처리하지 않고 통째로 처리하는 기법
x <- 1:10
even.loop <- logical(length(x))
names(even.loop) <- x

for (i in x) {
  if (i %% 2 == 0) {
    even.loop[i] <- TRUE }
  else {
    even.loop[i] <- FALSE
  }
}
even.loop
```

```
##     1     2     3     4     5     6     7     8     9    10
## FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
```

```
even.vectoriz <- x %% 2 == 0
names(even.vectoriz) <- x
even.vectoriz
```

```
##     1     2     3     4     5     6     7     8     9    10
## FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
```

```
if (x %% 2 == 0) "짝수" else "홀수"
```

```
## Warning in if (x%%2 == 0) "짝수" else "홀수": length > 1 이라는 조건이 있
## 고, 첫번째 요소만이 사용될 것입니다
```

```
## [1] "홀수"
```

```
ifelse(x %% 2 == 0, "짝수", "홀수")
```

```
##  [1] "홀수" "짝수" "홀수" "짝수" "홀수" "짝수" "홀수" "짝수" "홀수" "짝수"
```

```
x <- 1:10000000

# 반복 처리의 수행 속도
even <- function(x) {
  z <- logical(length(x))
  for (i in x) {
    if (i %% 2 == 0) {
      z[i] <- TRUE }
    else {
      z[i] <- FALSE
    }
  }
  z
}

runtime.loop <- system.time(z1 <- even(x))
runtime.loop
```

```
##    user  system elapsed
##    2.14    0.02    2.15
```

```
# 벡터라이제이션 처리의 수행 속도
runtime.vec <- system.time(z2 <- x %% 2 == 0)
runtime.vec
```

```
##    user  system elapsed
##    0.22    0.01    0.23
```

```
# 결과의 비교
sum(z1 != z2)
```

```
## [1] 0
```

```
# 3.1.3 리사이클링 툴 : 벡터 연산에서 사용되는 벡터 기리가 다를 경우 짧은 쪽의 벡터를 긴 쪽의 벡터의 길이에 맞춰 재상용하여 처리하는 것.

x <- 1:2
y <- 1:4
z <- 1:3
x + y
```

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

```
x + z
```

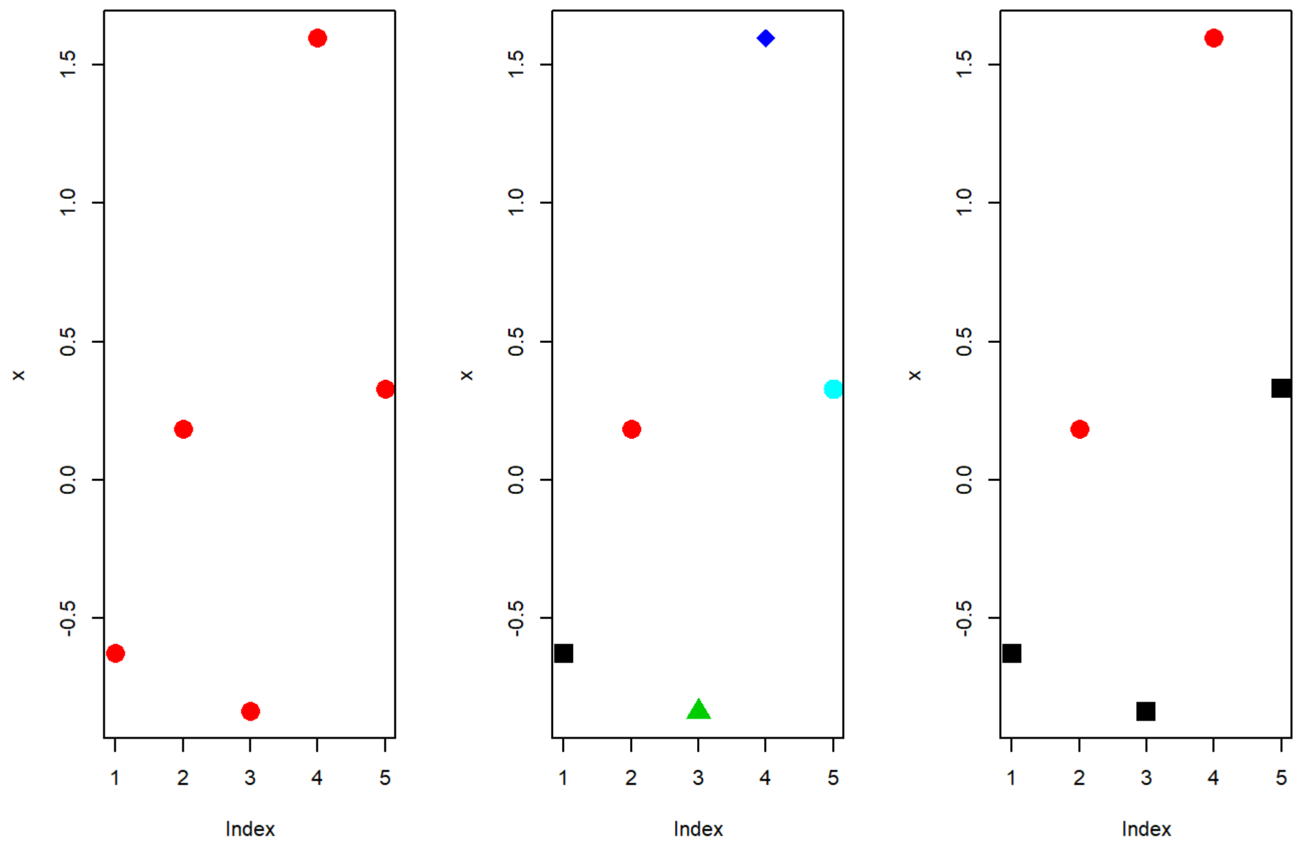
```
## Warning in x + z: 두 객체의 길이가 서로 배수관계에 있지 않습니다
```

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

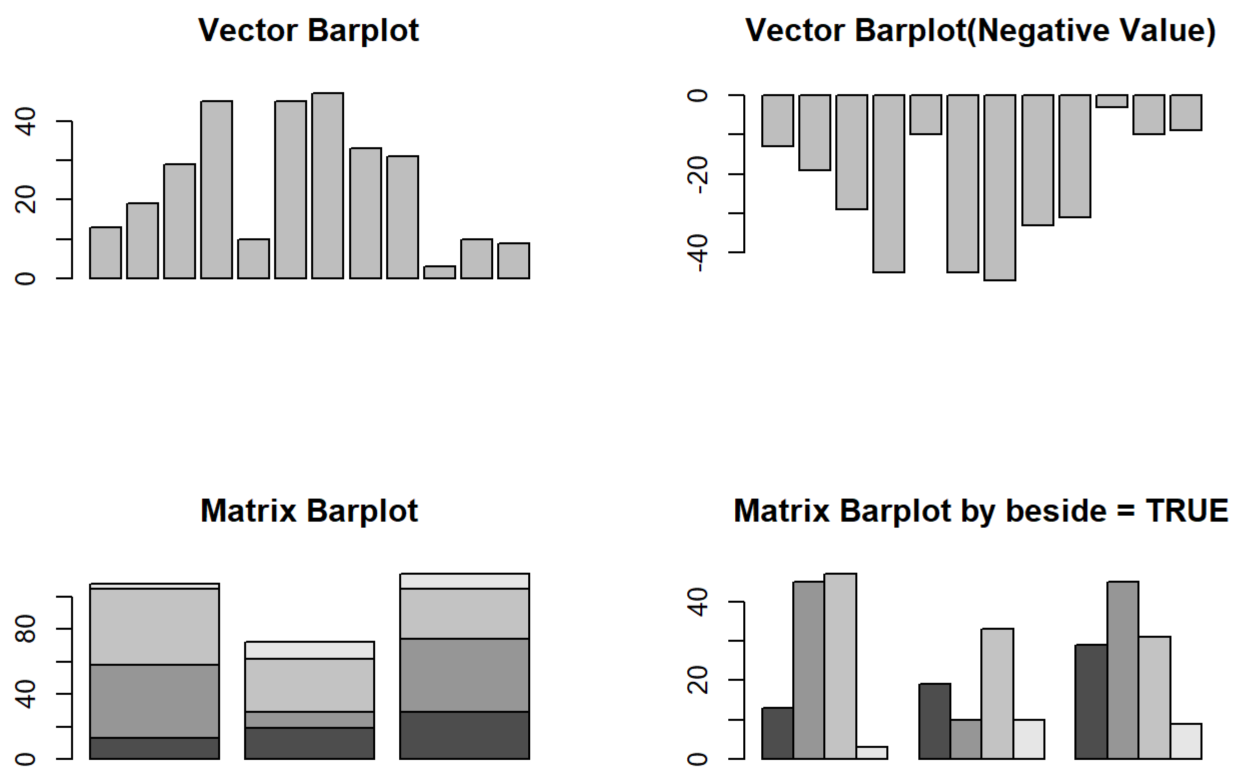
```
op <- par(no.readonly = TRUE)
set.seed(1)
(x <- rnorm(5))
```

```
## [1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078
```

```
par(mfrow = c(1, 3))
plot(x, col="red", pch=16, cex=2)
plot(x, col=1:5, pch=15:19 , cex=2)
plot(x, col=1:2, pch=15:16 , cex=2)
```



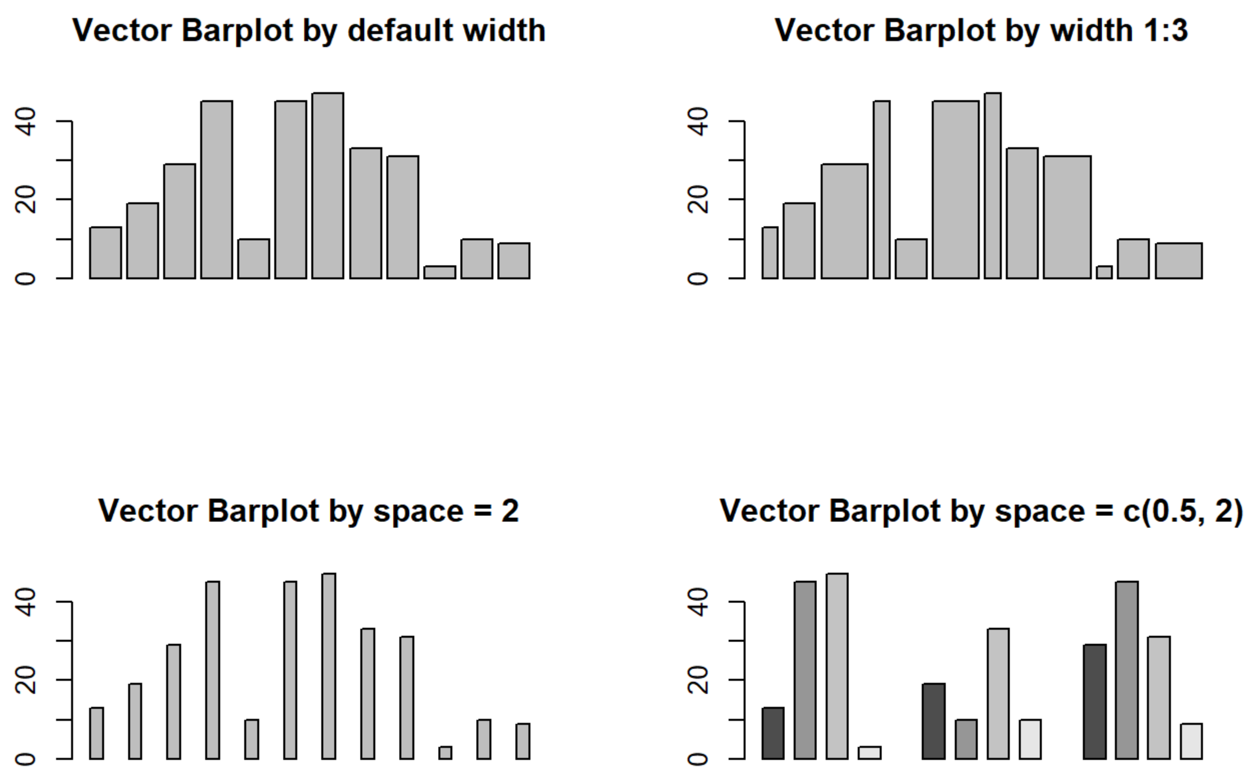
```
par(op)

# 3.2 graphics 패키지
# 3.2.1 barplot() 함수

op <- par(no.readonly = TRUE)
set.seed(1)
bar.x <- round(runif(12) * 50)
set.seed(2)
bar.y <- matrix(bar.x, ncol = 3, byrow = T)
par(mfrow = c(2, 2))
barplot(bar.x)
title(main = "Vector Barplot")
barplot(-bar.x)
title(main = "Vector Barplot(Negative Value)")
barplot(bar.y)
title(main = "Matrix Barplot")
barplot(bar.y, beside = TRUE)
title(main = "Matrix Barplot by beside = TRUE")
```



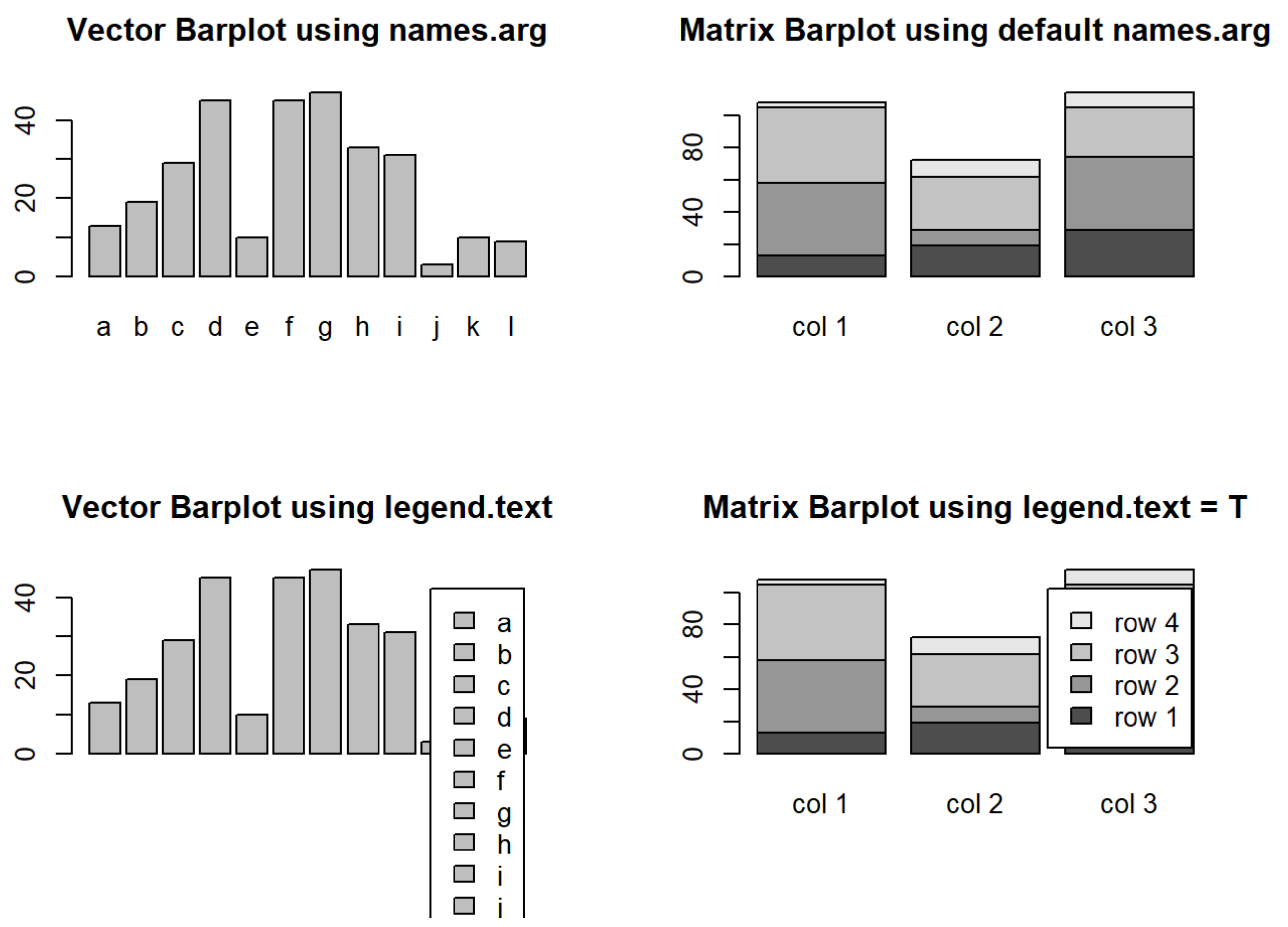
```
par(op)

op <- par(no.readonly = TRUE)
bar.width <- rep(1:3, 4)
bar.width
```

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

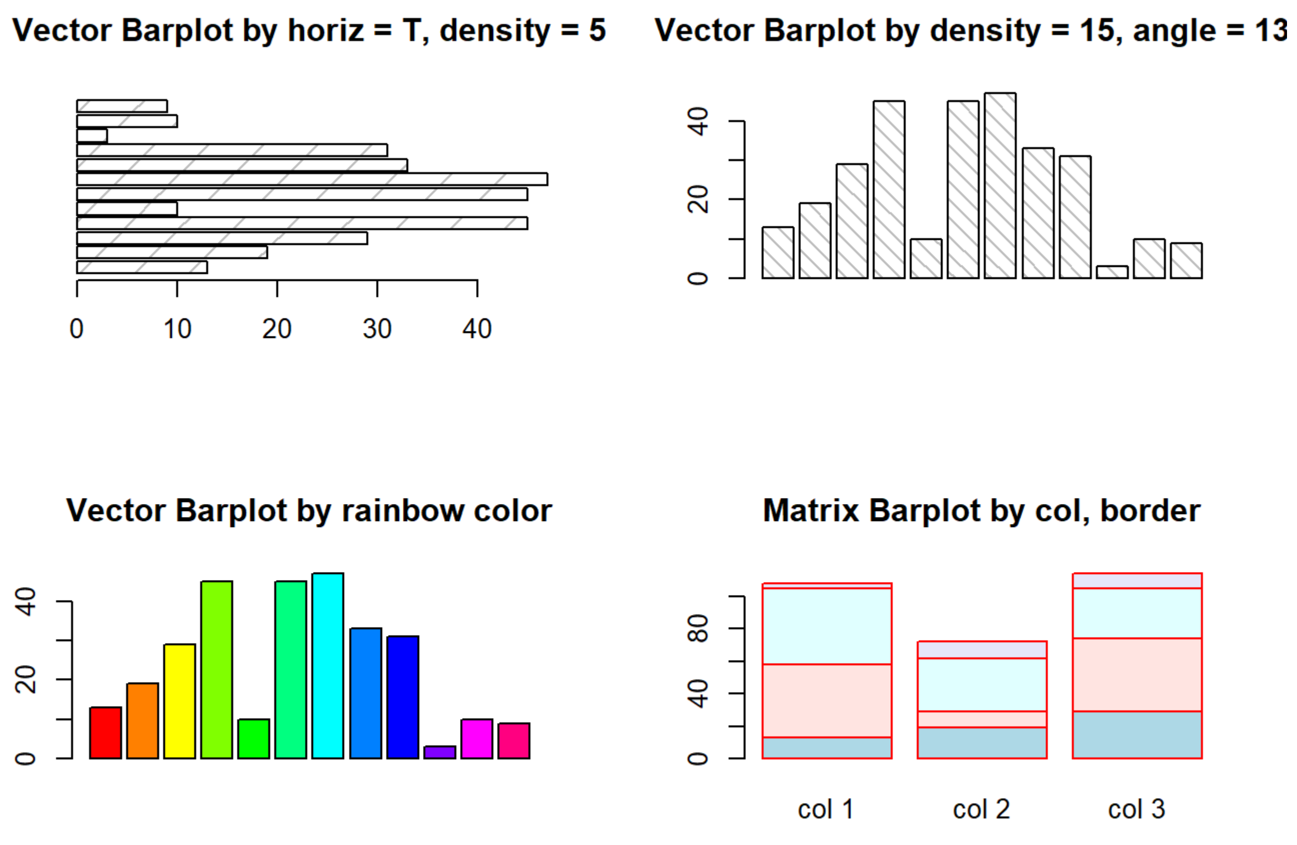
```
par(mfrow = c(2, 2))
barplot(bar.x, width = 1)
title(main = "Vector Barplot by default width")
barplot(bar.x, width = bar.width)
title(main = "Vector Barplot by width 1:3")
barplot(bar.x, space = 2)
title(main = "Vector Barplot by space = 2")
barplot(bar.y, beside = TRUE, space = c(0.5, 2))
title(main="Vector Barplot by space = c(0.5, 2)")
```



```
par(op)

op <- par(no.readonly = TRUE)
rownames(bar.y) <- paste("row", 1:4)
colnames(bar.y) <- paste("col", 1:3)
par(mfrow = c(2, 2))
barplot(bar.x, names.arg = letters[1:length(bar.x)])
title(main = "Vector Barplot using names.arg")
barplot(bar.y)
title(main = "Matrix Barplot using default names.arg")
barplot(bar.x, legend.text = letters[1:length(bar.x)])
title(main = "Vector Barplot using legend.text")
barplot(bar.y, legend.text = T)
title(main = "Matrix Barplot using legend.text = T")
```



```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
barplot(bar.x, horiz = T, density = 5)
title(main = "Vector Barplot by horiz = T, density = 5")
barplot(bar.x, density = 15, angle = 135)
title(main = "Vector Barplot by density = 15, angle = 135")
barplot(bar.x, col = rainbow(length(bar.x)))
title(main = "Vector Barplot by rainbow color")
barplot(bar.y, border = "red",
        col = c("lightblue", "mistyrose","lightcyan", "lavender"))
title(main = "Matrix Barplot by col, border")
```



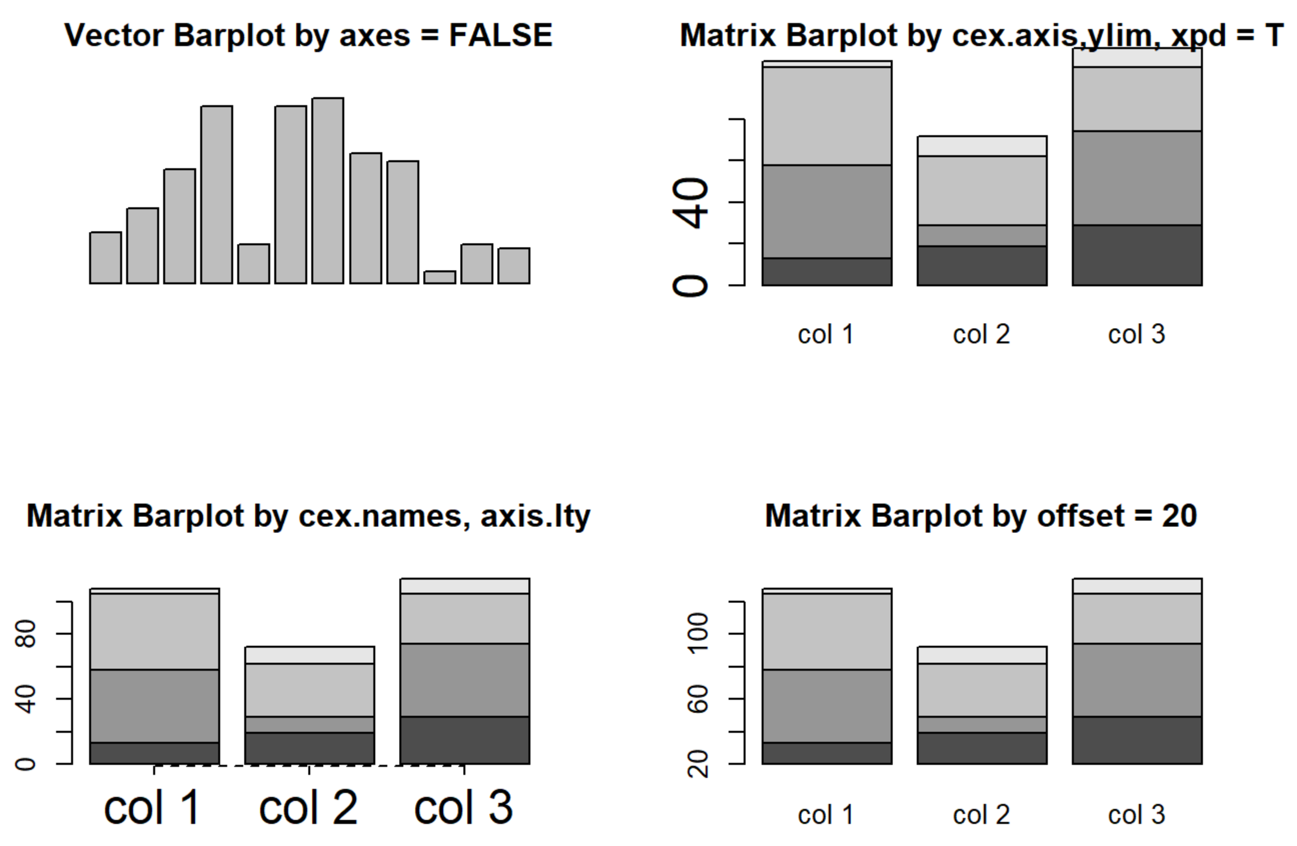
```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
barplot(bar.x, axes = FALSE)
title(main = "Vector Barplot by axes = FALSE")
barplot(bar.y, cex.axis = 1.8, ylim = c(0, 90), xpd = T)
title(main = "Matrix Barplot by cex.axis,ylim, xpd = T")
barplot(bar.y, axisnames = T, cex.names = 1.8, axis.lty = 2)
title(main = "Matrix Barplot by cex.names, axis.lty")
t(barplot(bar.x, plot = F))    # 그래프를 그리지 않는다.
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## [1,]  0.7  1.9  3.1  4.3  5.5  6.7  7.9  9.1 10.3  11.5  12.7  13.9
```

```
barplot(bar.y, plot = F)    # 그래프를 그리지 않는다.
```

```
## [1] 0.7 1.9 3.1
```

```
barplot(bar.y, offset = 20, main = "Matrix Barplot by offset = 20")
```



```
par(op)

# 3.2.2 boxplot() 함수

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
set.seed(1)
norm1 <- round(rnorm(100, 3, 2), digits = 2)
set.seed(2)
norm2 <- round(rnorm(100, 3, 3), digits = 2)
# (1)
boxplot(norm1)
title("boxplot of one vector")
# (2)
boxplot(norm1, norm2)
title("boxplot of two vectors")
list1 = list(data1 = norm1, data2 = norm2, data3 = rnorm(100, 7, 4))
# (3)
boxplot(list1)
title("boxplot of simple list")
dimnames(InsectSprays)
```

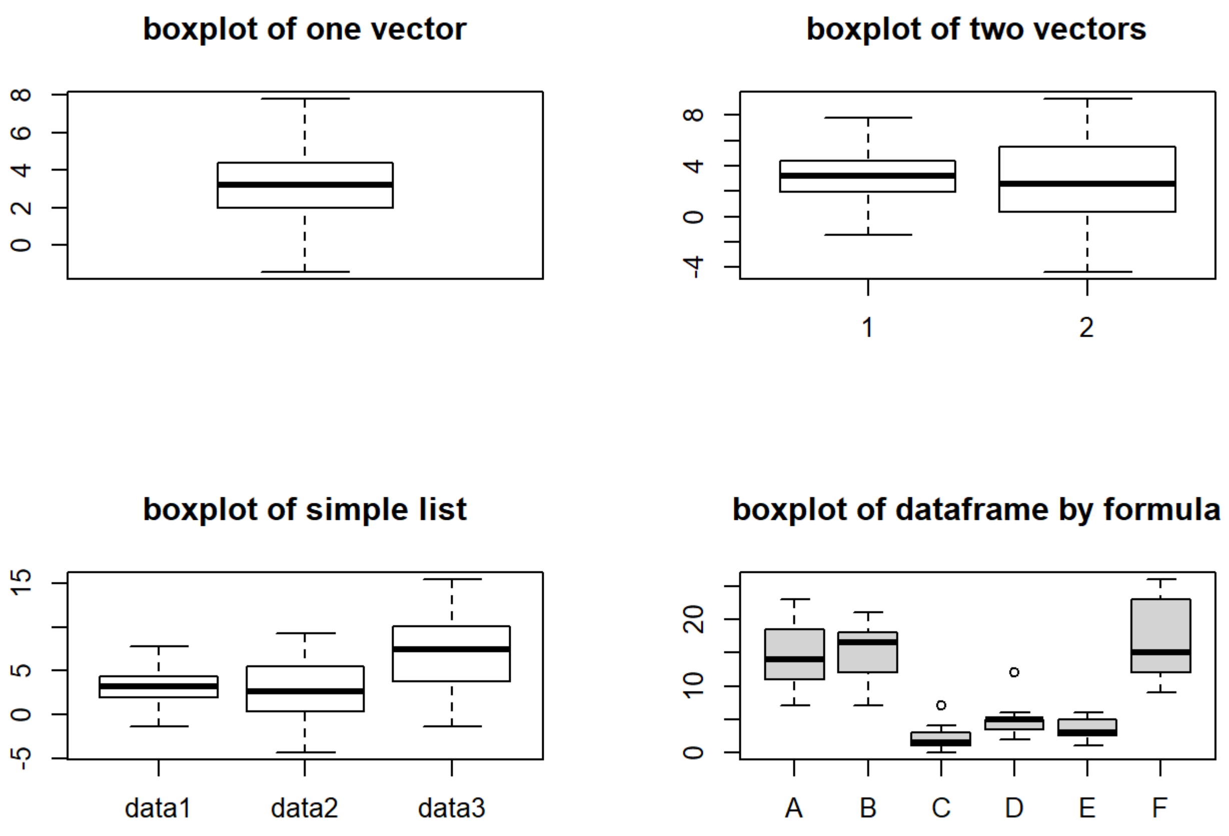
```
## [[1]]
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28"
## [29] "29" "30" "31" "32" "33" "34" "35" "36" "37" "38" "39" "40" "41" "42"
## [43] "43" "44" "45" "46" "47" "48" "49" "50" "51" "52" "53" "54" "55" "56"
## [57] "57" "58" "59" "60" "61" "62" "63" "64" "65" "66" "67" "68" "69" "70"
## [71] "71" "72"
```

```
##
## [[2]]
## [1] "count" "spray"
```

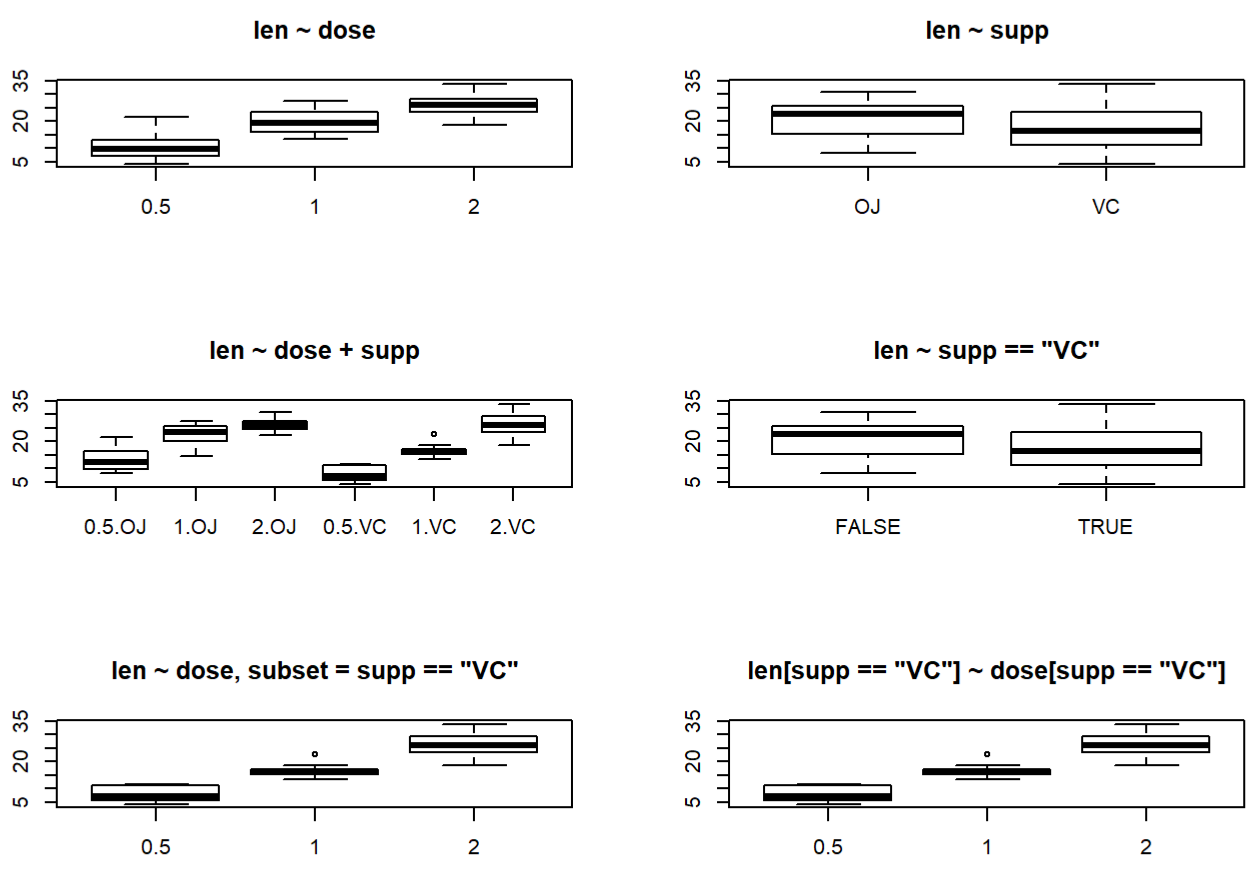
```
dim(InsectSprays)
```

```
## [1] 72  2
```

```
# (4)
boxplot(count ~ spray, data = InsectSprays, col = "lightgray")
title("boxplot of dataframe by formula")
```



```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(3, 2))
boxplot(len ~ dose, data = ToothGrowth)
title("len ~ dose")
boxplot(len ~ supp, data = ToothGrowth)
title("len ~ supp")
boxplot(len ~ dose + supp, data = ToothGrowth)
title("len ~ dose + supp")
boxplot(len ~ supp == "VC", data = ToothGrowth)
title("len ~ supp == \"VC\"")
boxplot(len ~ dose, data = ToothGrowth, subset = supp == "VC")
title("len ~ dose, subset = supp == \"VC\"")
boxplot(len[supp == "VC"] ~ dose[supp == "VC"], data = ToothGrowth)
title("len[supp == \"VC\"] ~ dose[supp == \"VC\"]")
```



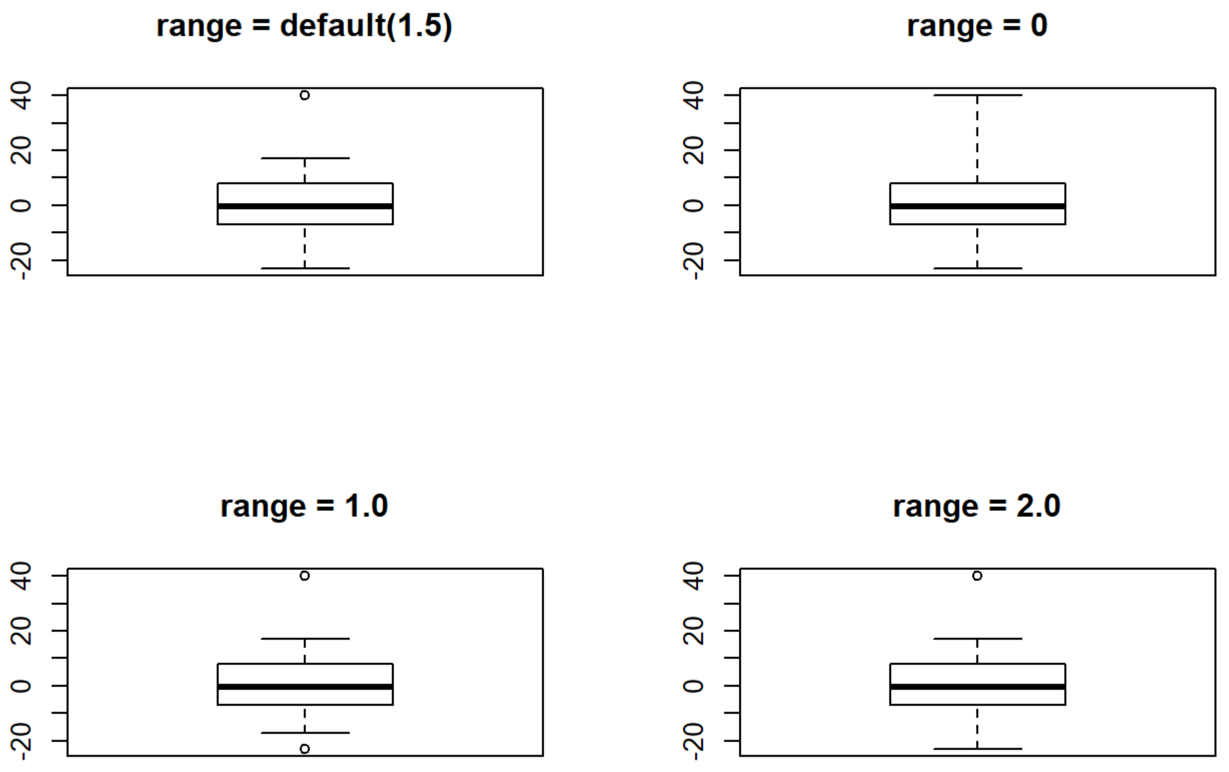
```
par(op)

op <- par(no.readonly = TRUE)
set.seed(3)
z <- round(rnorm(50) * 10)
summary(z)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  -23.00   -7.00   -1.50   -0.66    7.00   17.00
```

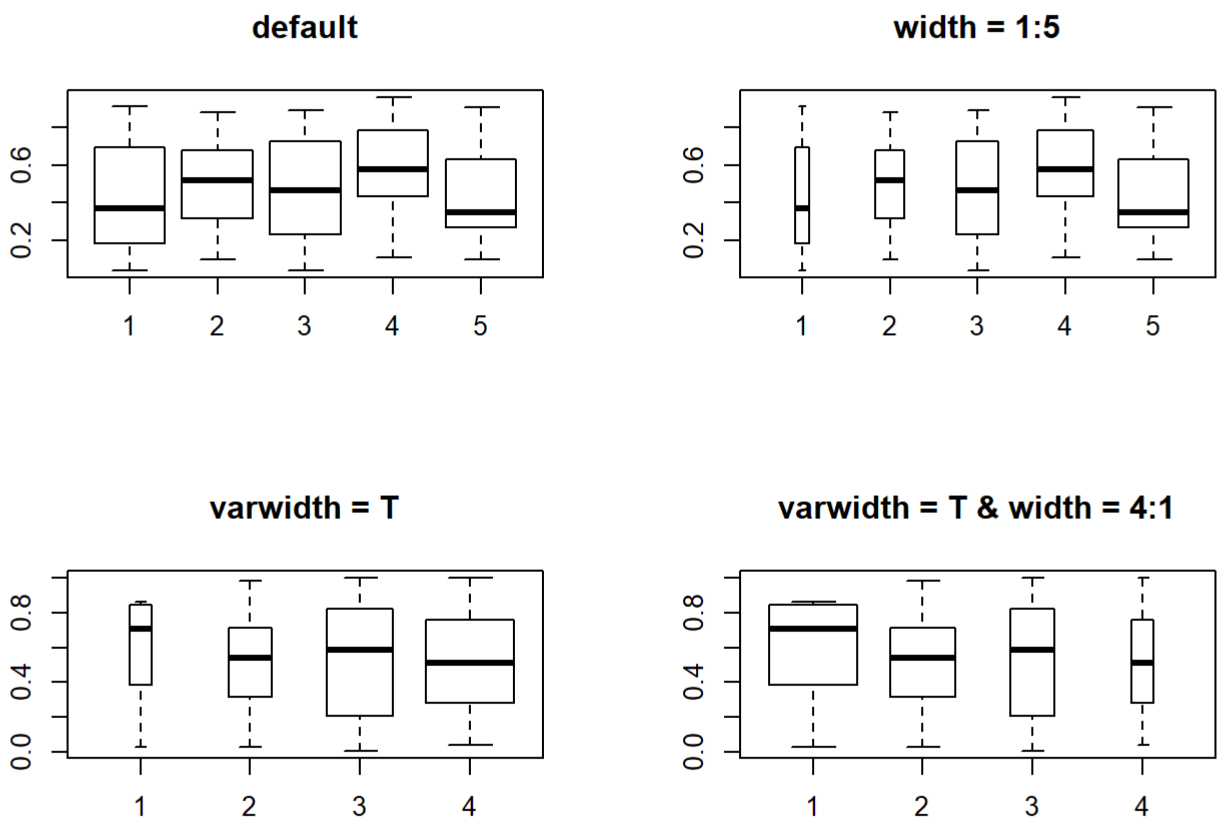
```
z[50] <- 40      # 50번째 데이터를 40으로 치환하여 이상치를 만듦
summary(z)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  -23.00   -7.00   -0.50    0.32    7.75   40.00
```

```
par(mfrow = c(2, 2))
boxplot(z)
title(main="range = default(1.5)")
boxplot(z, range = 0)
title(main="range = 0")
boxplot(z, range = 1.0)
title(main="range = 1.0")
boxplot(z, range = 2.0)
title(main="range = 2.0")
```



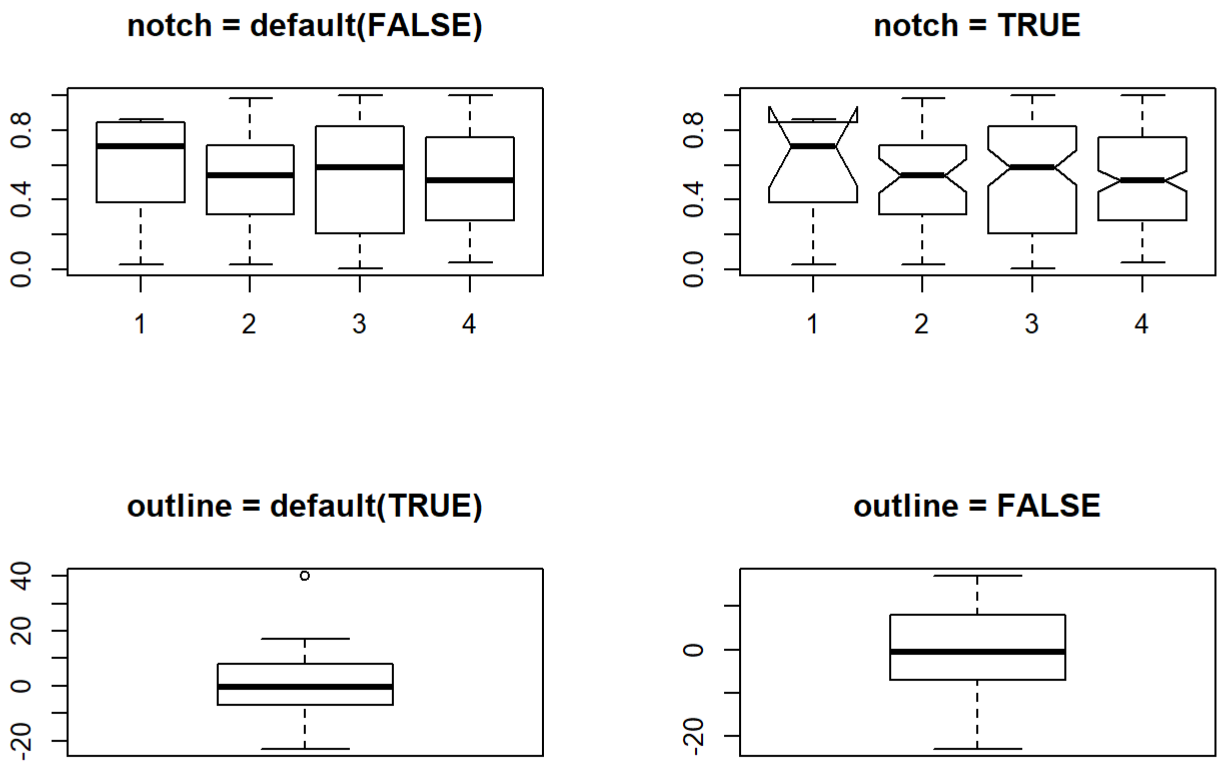
```
par(op)

op <- par(no.readonly = TRUE)
x1 <- runif(20)
x2 <- runif(20)
x3 <- runif(20)
x4 <- runif(20)
x5 <- runif(20)
x <- list(x1, x2, x3, x4, x5)
y1 <- runif(10)
y2 <- runif(40)
y3 <- runif(90)
y4 <- runif(160)
y <- list(y1, y2, y3, y4)
par(mfrow = c(2, 2))
boxplot(x)
title(main = "default")
boxplot(x, width = 1:5)
title(main = "width = 1:5")
boxplot(y, varwidth = T)
title(main = "varwidth = T")
boxplot(y, varwidth = T, width = 4:1)
title(main = "varwidth = T & width = 4:1")
```



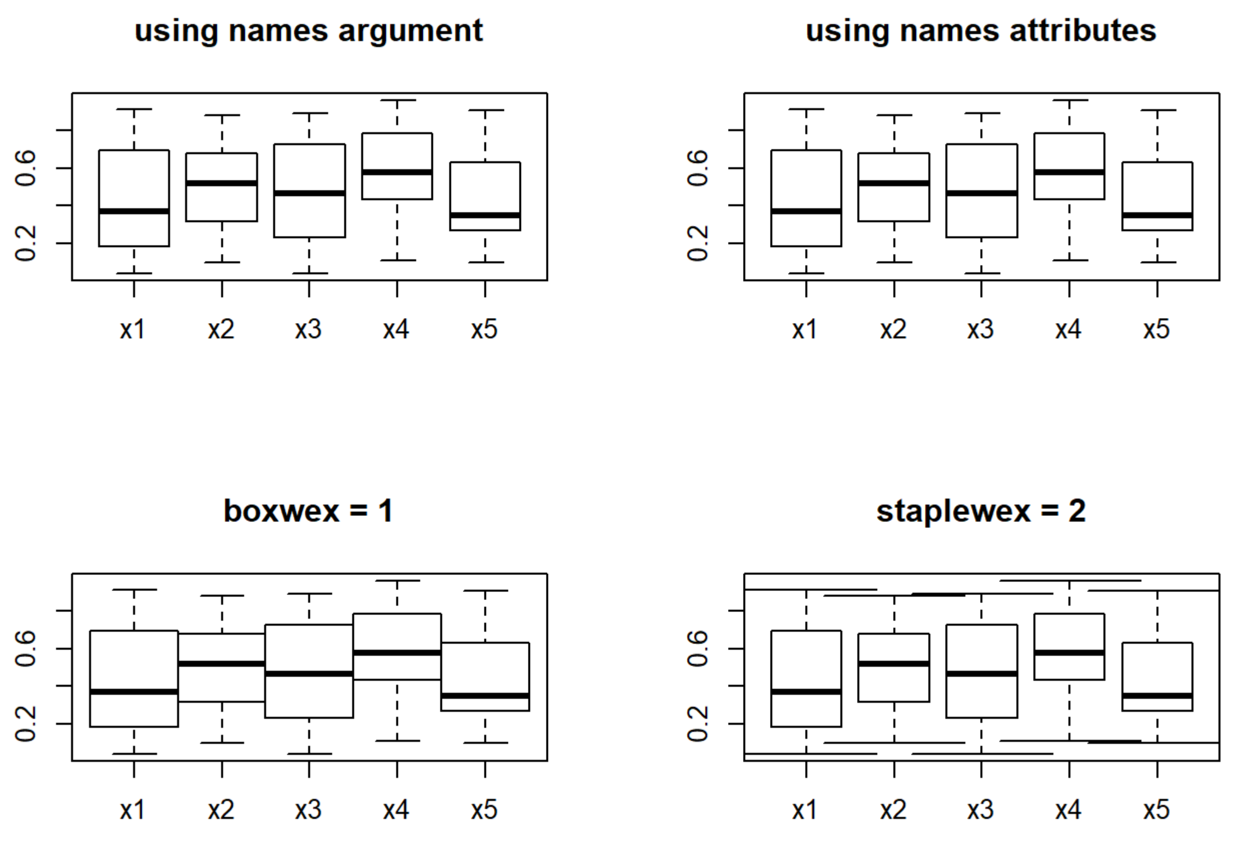
```
op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
boxplot(y)
title(main = "notch = default(FALSE)")
boxplot(y, notch = T, main = "notch = TRUE")
```

```
## Warning in bxp(list(stats = structure(c(0.0328013296239078,
## 0.38725300040096, : some notches went outside hinges ('box'): maybe set
## notch=FALSE
```

```
boxplot(z, main = "outline = default(TRUE)")
boxplot(z, outline = F, main = "outline = FALSE")
```



```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
# names 인수를 사용할 경우
xname <- c("x1", "x2", "x3", "x4", "x5")
boxplot(x, names = xname)
title(main = "using names argument")
# names attributes를 이용할 경우
names(x) <- c("x1", "x2", "x3", "x4", "x5")
boxplot(x)
title(main = "using names attributes")
boxplot(x, boxwex = 1)
title(main = "boxwex = 1")
boxplot(x, staplewex = 2)
title(main = "staplewex = 2")
```



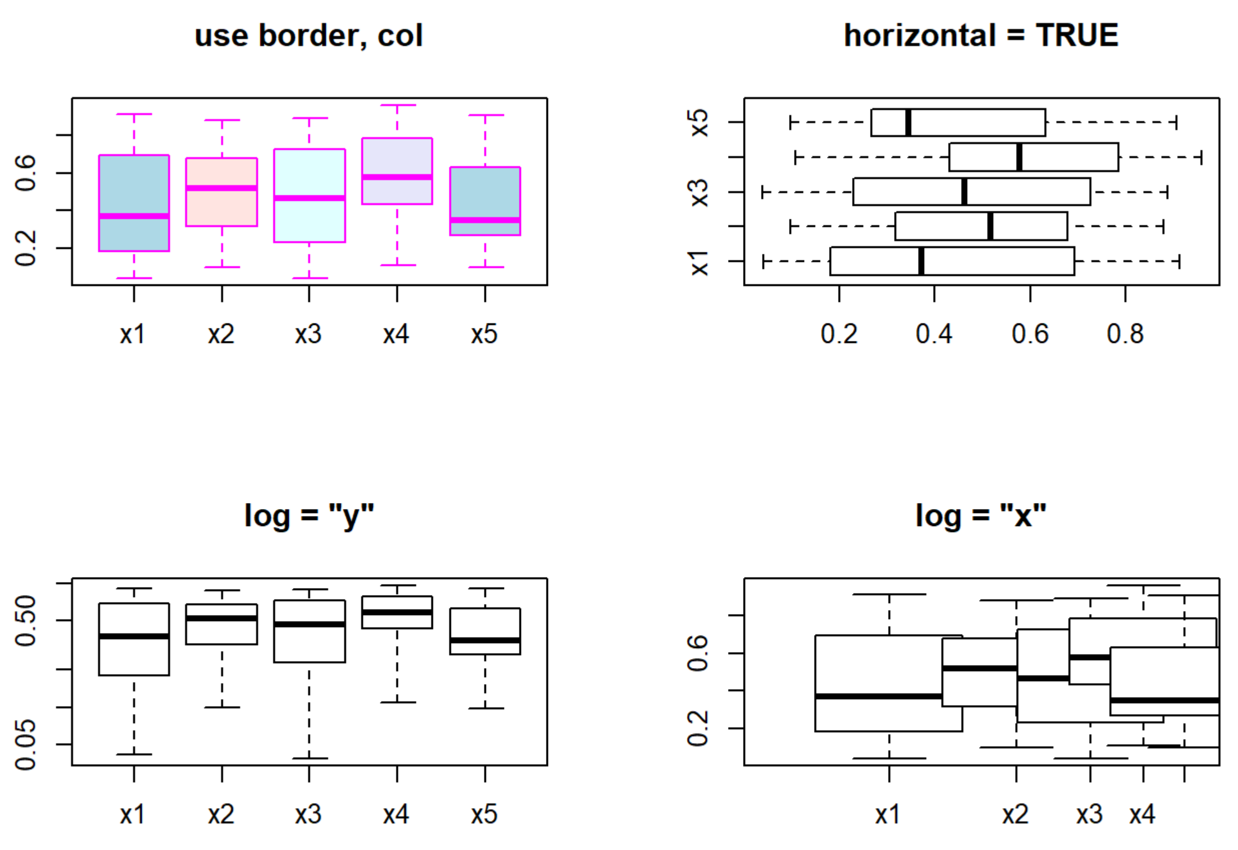
```
par(op)

boxplot(x, plot = FALSE)
```

```
## $stats
##              [,1]       [,2]       [,3]      [,4]       [,5]
## [1,] 0.04117032 0.09869789 0.03852075 0.1087810 0.09831946
## [2,] 0.18077793 0.31802044 0.23061703 0.4298766 0.26834736
## [3,] 0.37115586 0.51577862 0.46259269 0.5765672 0.34477785
## [4,] 0.69219256 0.67721069 0.72621593 0.7847074 0.63193596
## [5,] 0.91342360 0.87987004 0.88821092 0.9586770 0.90653142
##
## $n
## [1] 20 20 20 20 20
##
## $conf
##             [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.1904737 0.3888772 0.2874982 0.4512060 0.2163224
## [2,] 0.5518380 0.6426801 0.6376872 0.7019285 0.4732332
##
## $out
## numeric(0)
##
## $group
## numeric(0)
##
## $names
## [1] "x1" "x2" "x3" "x4" "x5"
```

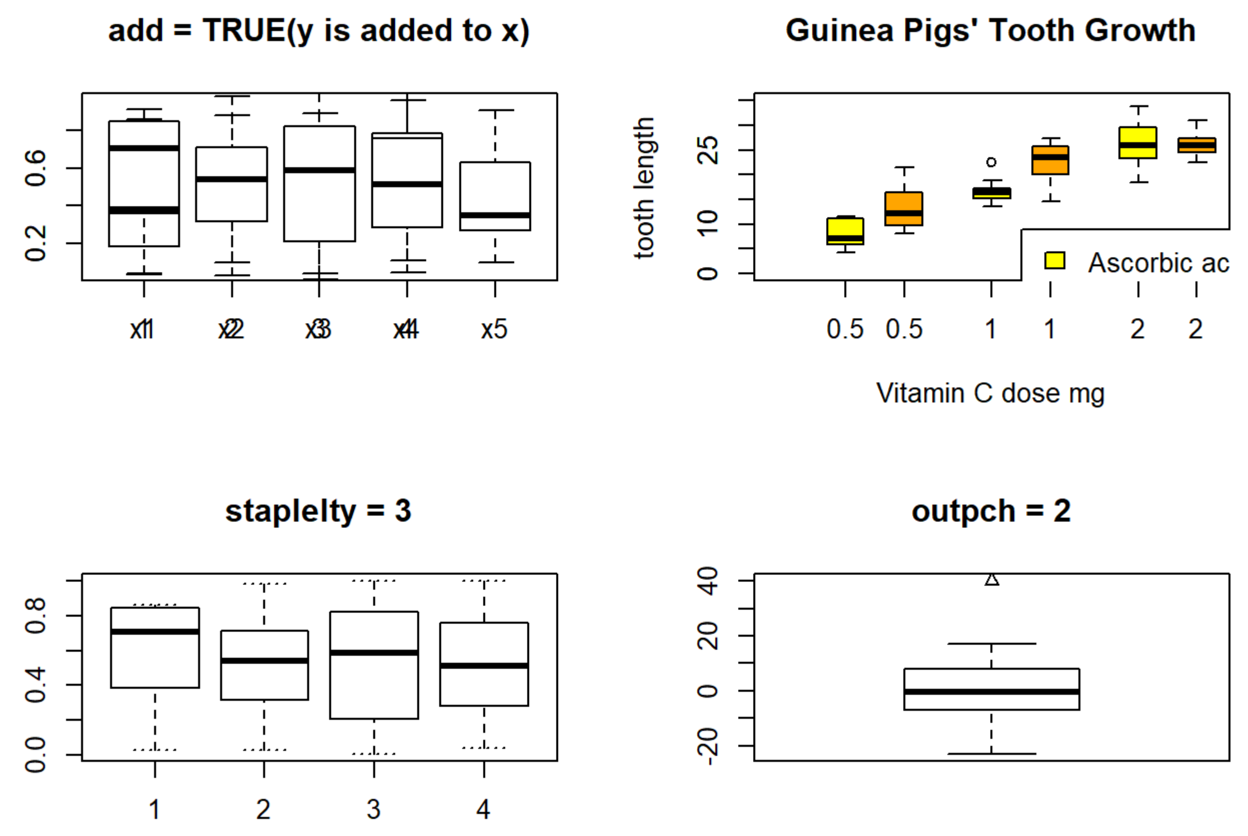
```
boxplot(z, plot = FALSE)
```

```
## $stats
##       [,1]
## [1,] -23.0
## [2,]  -7.0
## [3,]  -0.5
## [4,]   8.0
## [5,]  17.0
##
## $n
## [1] 50
##
## $conf
##           [,1]
## [1,] -3.851686
## [2,]  2.851686
##
## $out
## [1] 40
##
## $group
## [1] 1
##
## $names
## [1] "1"
```

```
op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
boxplot(x, border = "magenta", col = c("lightblue", "mistyrose", "lightcyan", "lavender"))
title(main = "use border, col")
boxplot(x, horizontal = TRUE)
title(main = "horizontal = TRUE")
boxplot(x, log = "y", main = "log = \"y\"")
boxplot(x, log = "x", main = "log = \"x\"")
```



```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
boxplot(x)
boxplot(y, add = TRUE)
title(main = "add = TRUE(y is added to x)")
boxplot(len ~ dose, data = ToothGrowth, boxwex = 0.25, at = 1:3 - 0.2,
        subset = supp == "VC", col = "yellow", main = "Guinea Pigs' Tooth Growth",
        xlab = "Vitamin C dose mg", ylab = "tooth length", ylim = c(0, 35))
boxplot(len ~ dose, data = ToothGrowth, add = TRUE, boxwex = 0.25, at = 1:3 + 0.2,
        subset = supp == "OJ", col = "orange")
legend(2, 9, c("Ascorbic acid", "Orange juice"), fill = c("yellow", "orange"))
boxplot(y, staplelty = 3)
title(main = "staplelty = 3")
boxplot(z, outpch = 2)
title(main = "outpch = 2")
```

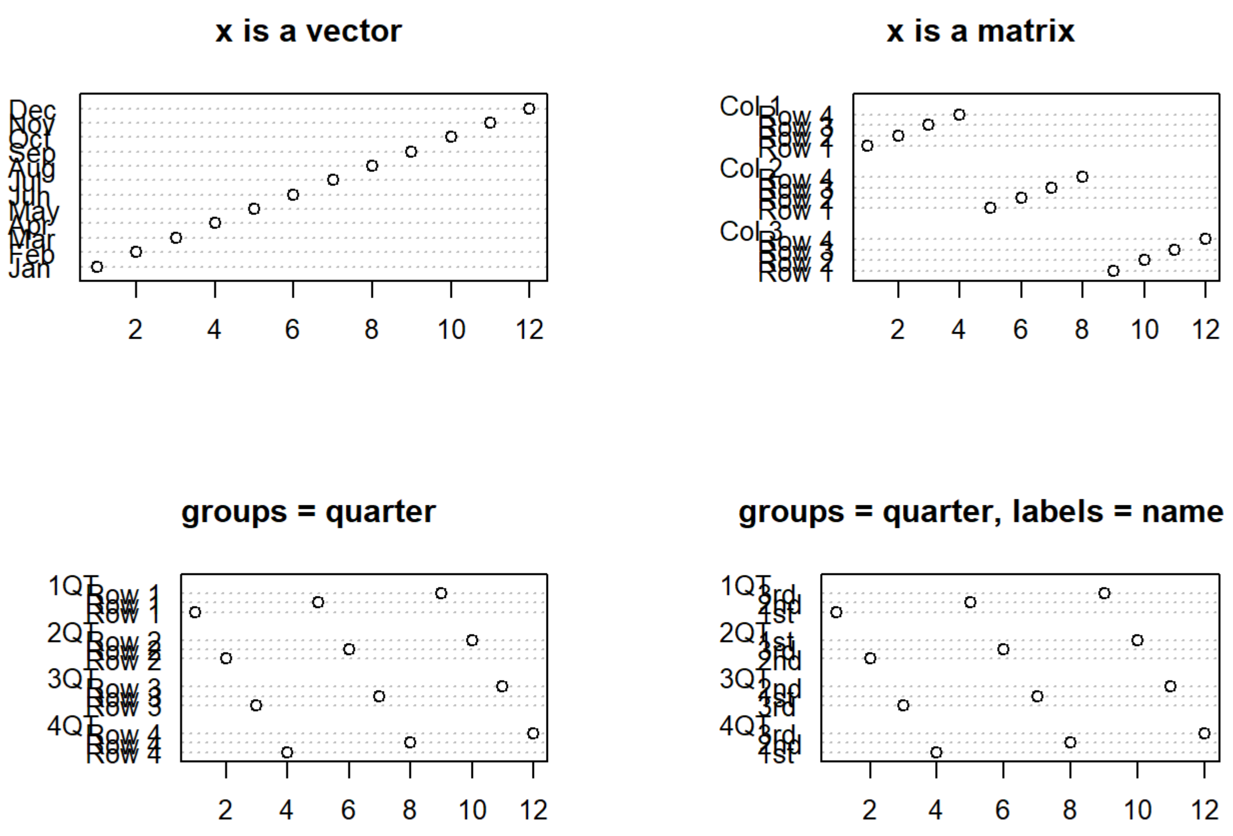


```
par(op)


# 3.2.3 dotchart() 함수

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
month <- matrix(1:12, ncol = 3)
rownames(month) <- paste("Row", 1:4)
colnames(month) <- paste("Col", 1:3)
# (1) 벡터
dotchart(as.vector(month), label = month.abb)
title(main = "x is a vector")
# (2) 행렬
dotchart(month)
title(main = "x is a matrix")
# (3) group
quarter.name <- c("1QT", "2QT", "3QT", "4QT")
quarter <- factor(row(month), label = quarter.name)
quarter
```

```
##  [1] 1QT 2QT 3QT 4QT 1QT 2QT 3QT 4QT 1QT 2QT 3QT 4QT
## Levels: 1QT 2QT 3QT 4QT
```

```
dotchart(month, groups = quarter)
title(main = "groups = quarter")
# (4) groups, labels
name <- c("1st", "2nd", "3rd")
```

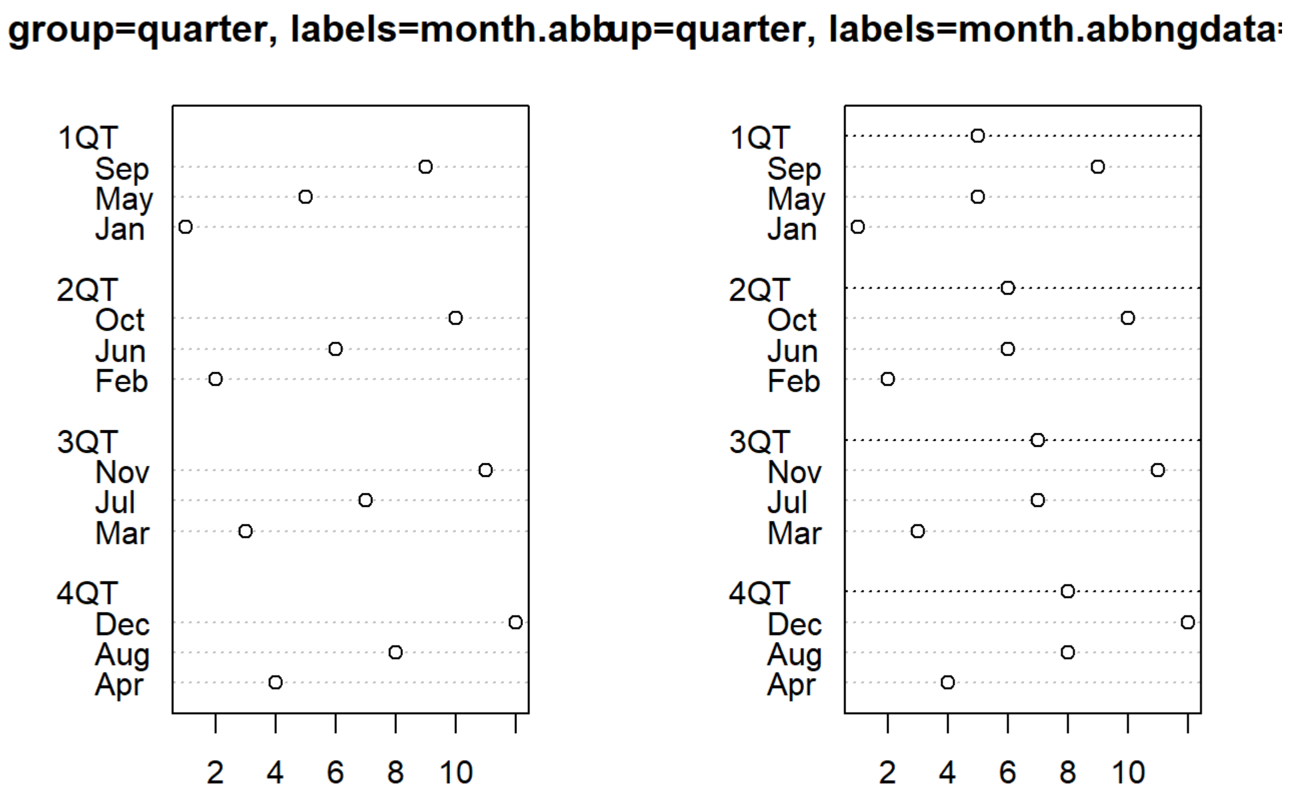
```
dotchart(month, groups = quarter, labels = name)
title(main = "groups = quarter, labels = name")
```



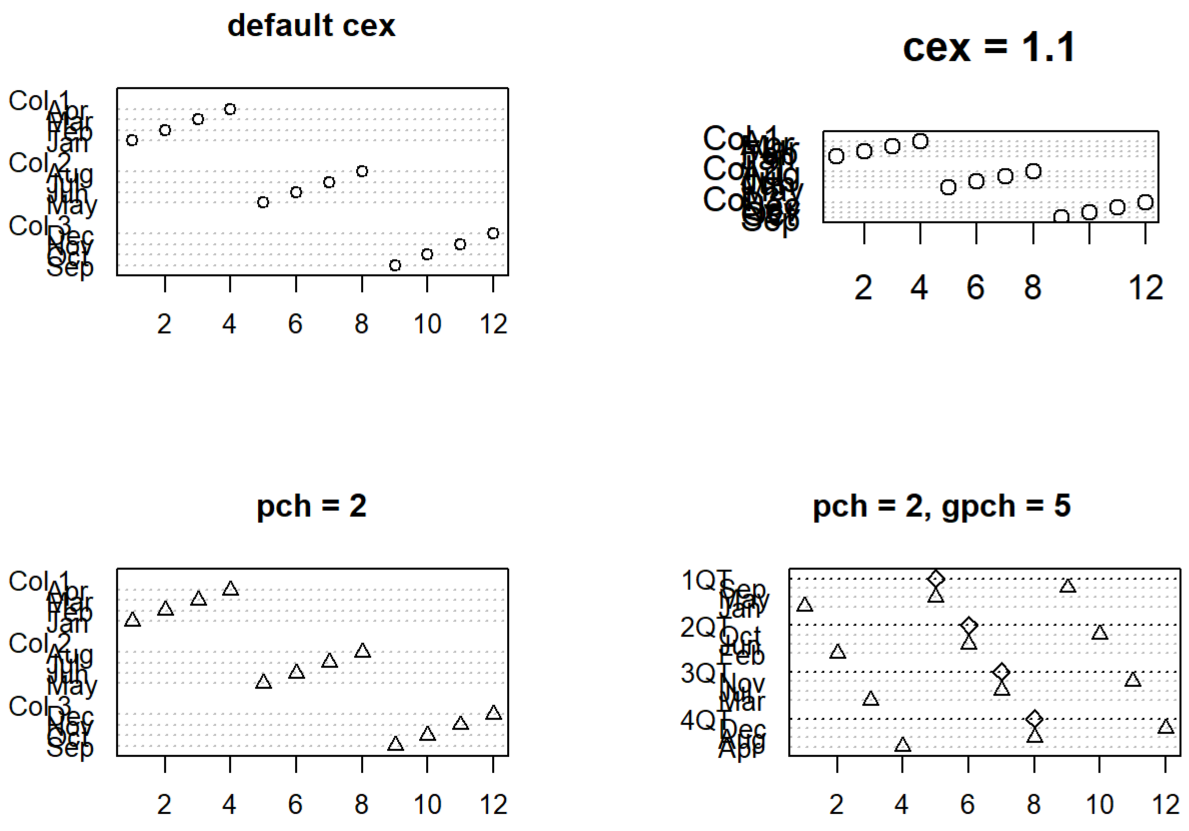
```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(1, 2))
dotchart(month, group = quarter, labels = month.abb)
title(main = "group=quarter, labels=month.abb")
gmean <- tapply(month, quarter, mean)
gmean
```

```
## 1QT 2QT 3QT 4QT
##   5   6   7   8
```

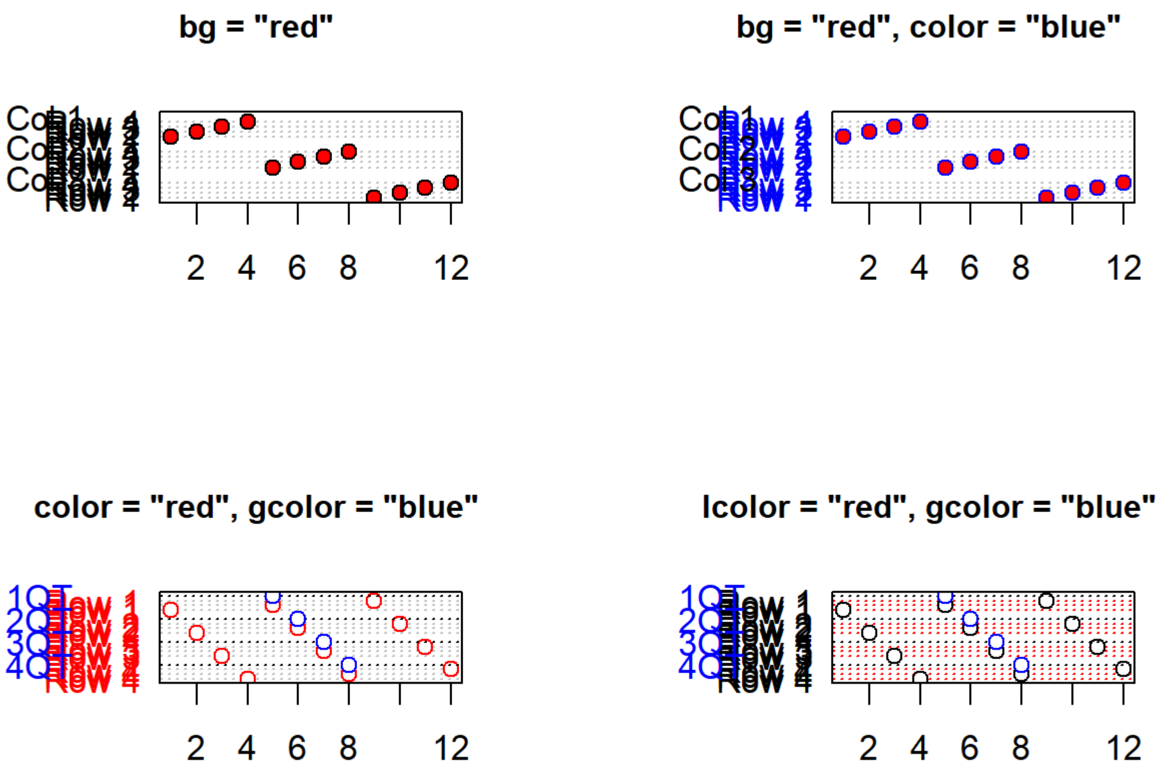
```
dotchart(month, group = quarter, labels = month.abb, gdata = gmean)
title(main = "group=quarter, labels=month.abbngdata=gmean")
```



```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
dotchart(month, labels = month.abb, main = "default cex")
dotchart(month, labels = month.abb, cex = 1.1, main = "cex = 1.1")
dotchart(month, labels = month.abb, pch = 2, main = "pch = 2")
dotchart(month, labels = month.abb, groups = quarter, pch = 2, gpch = 5, gdata = gmean)
title(main = "pch = 2, gpch = 5")
```



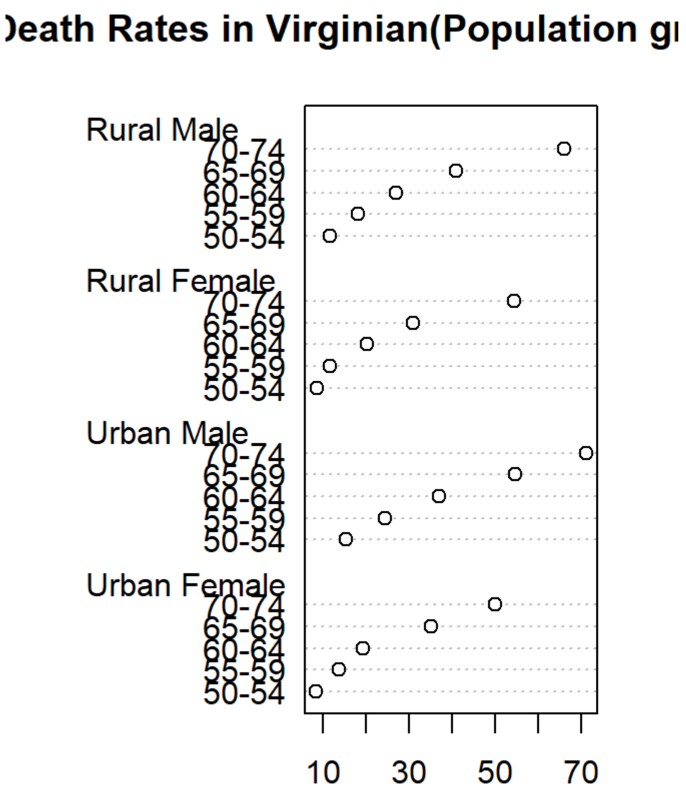
```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
dotchart(month, cex = 1.1, bg = "red")
title(main = "bg = \"red\"")
dotchart(month, cex = 1.1, bg = "red", color = "blue")
title(main = "bg = \"red\", color = \"blue\"")
dotchart(month, cex = 1.1, color = "red", gcolor = "blue", groups = quarter,
         gdata = gmean)
title(main = "color = \"red\", gcolor = \"blue\"")
dotchart(month, cex = 1.1, lcolor = "red", gcolor = "blue", groups = quarter,
         gdata = gmean)
title(main = "lcolor = \"red\", gcolor = \"blue\"")
```

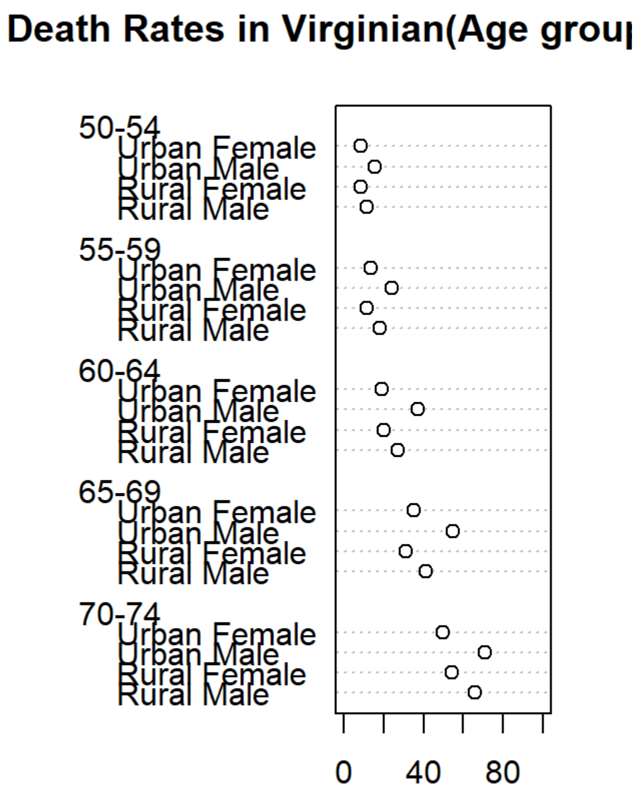


```
par(op)

op <- par(no.readonly = TRUE)
VADeaths
```

```
##       Rural Male Rural Female Urban Male Urban Female
## 50-54       11.7          8.7       15.4          8.4
## 55-59       18.1         11.7       24.3         13.6
## 60-64       26.9         20.3       37.0         19.3
## 65-69       41.0         30.9       54.6         35.1
## 70-74       66.0         54.3       71.1         50.0
```

```
par(mfrow = c(1, 2))
dotchart(VADeaths)
title(main = "Death Rates in Virginian(Population group)")
dotchart(t(VADeaths), xlim = c(0, 100))
title(main = "Death Rates in Virginian(Age group)")
```





```
par(op)


# 3.2.4 hist() 함수

pretty(0:1)
```

```
## [1] 0.0 0.2 0.4 0.6 0.8 1.0
```

```
pretty(0:1, 2)
```

```
## [1] 0.0 0.5 1.0
```

```
pretty(0:1, 1)
```

```
## [1] 0 1
```

```
pretty(c(.1, .98), 3)
```

```
## [1] 0.0 0.5 1.0
```

```
pretty(c(.1, 1.05), 3)
```

```
## [1] 0.0 0.5 1.0 1.5
```

```
set.seed(7)
hist.data <- rnorm(100, 3, 2)
hist.data <- round(hist.data, digits = 2)
summary(hist.data)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -0.570   1.880   3.210   3.278   4.442   8.430
```

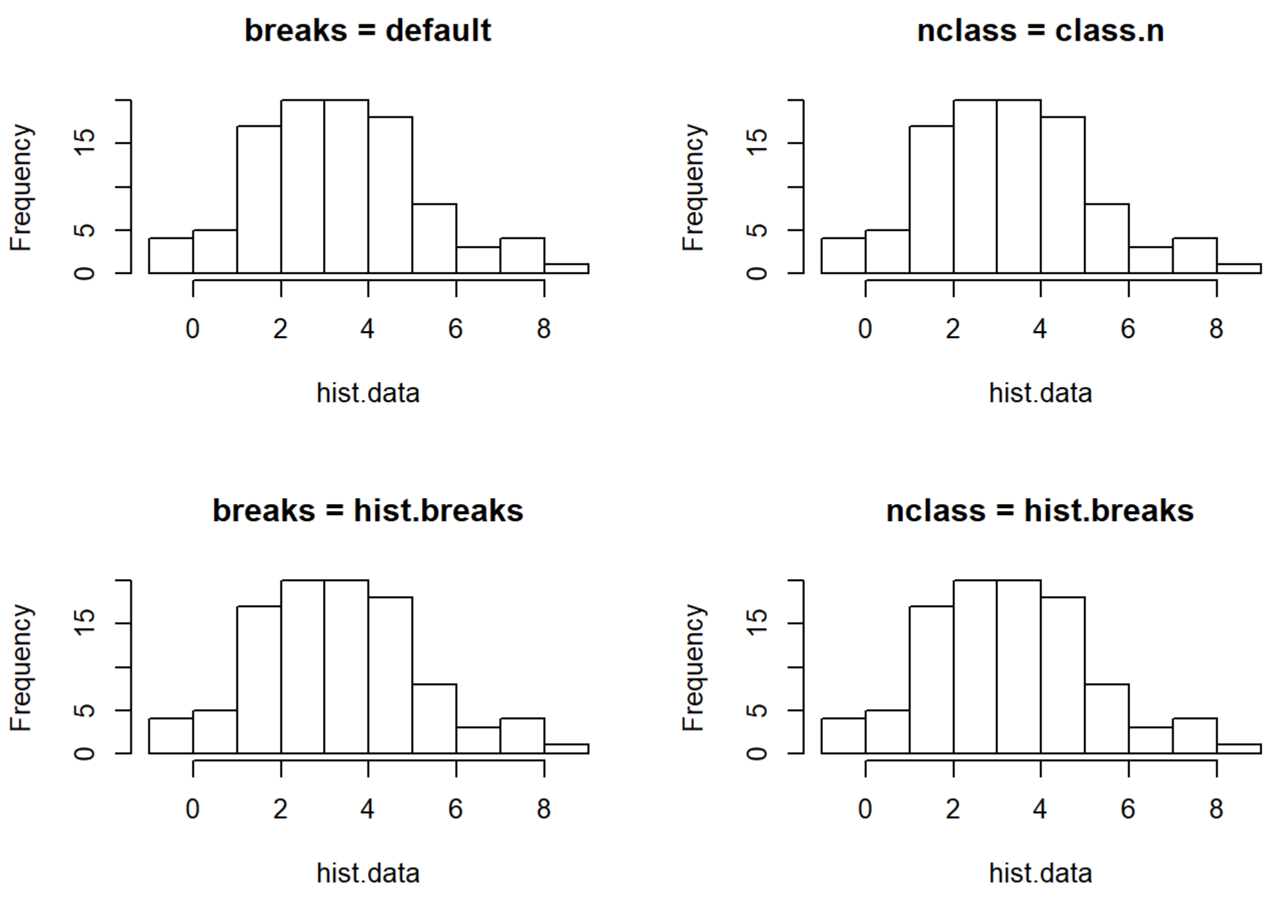
```
# Sturges 공식으로 구해진 계급의 수
class.n <- ceiling(log(length(hist.data), base = 2) +1 )
class.n
```

```
## [1] 8
```

```
# pretty 함수로 구한 breaks
hist.breaks <- pretty(hist.data, class.n)
hist.breaks
```

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

```
par(mfrow = c(2, 2))
hist(hist.data, main = "breaks = default")
hist(hist.data, breaks = class.n, main = "nclass = class.n")
hist(hist.data, breaks = hist.breaks, main = "breaks = hist.breaks")
hist(hist.data, nclass = hist.breaks, main = "nclass = hist.breaks")
```



```
# 도수분포표 계산
freq <- integer(length(hist.breaks) - 1)
for (i in seq(freq)) {
    freq[i] <- sum(hist.breaks[i] < hist.data & hist.data <= hist.breaks[i + 1])
  }
freq
```

```
##  [1]  4  5 17 20 20 18  8  3  4  1
```

```
op <- par(no.readonly = TRUE)
nclass.Sturges(hist.data)
```

```
## [1] 8
```

```
nclass.scott(hist.data)
```

```
## [1] 7
```

```
nclass.FD(hist.data)
```

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

```
pretty(hist.data, nclass.Sturges(hist.data))
```

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

```
pretty(hist.data, nclass.scott(hist.data))
```

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

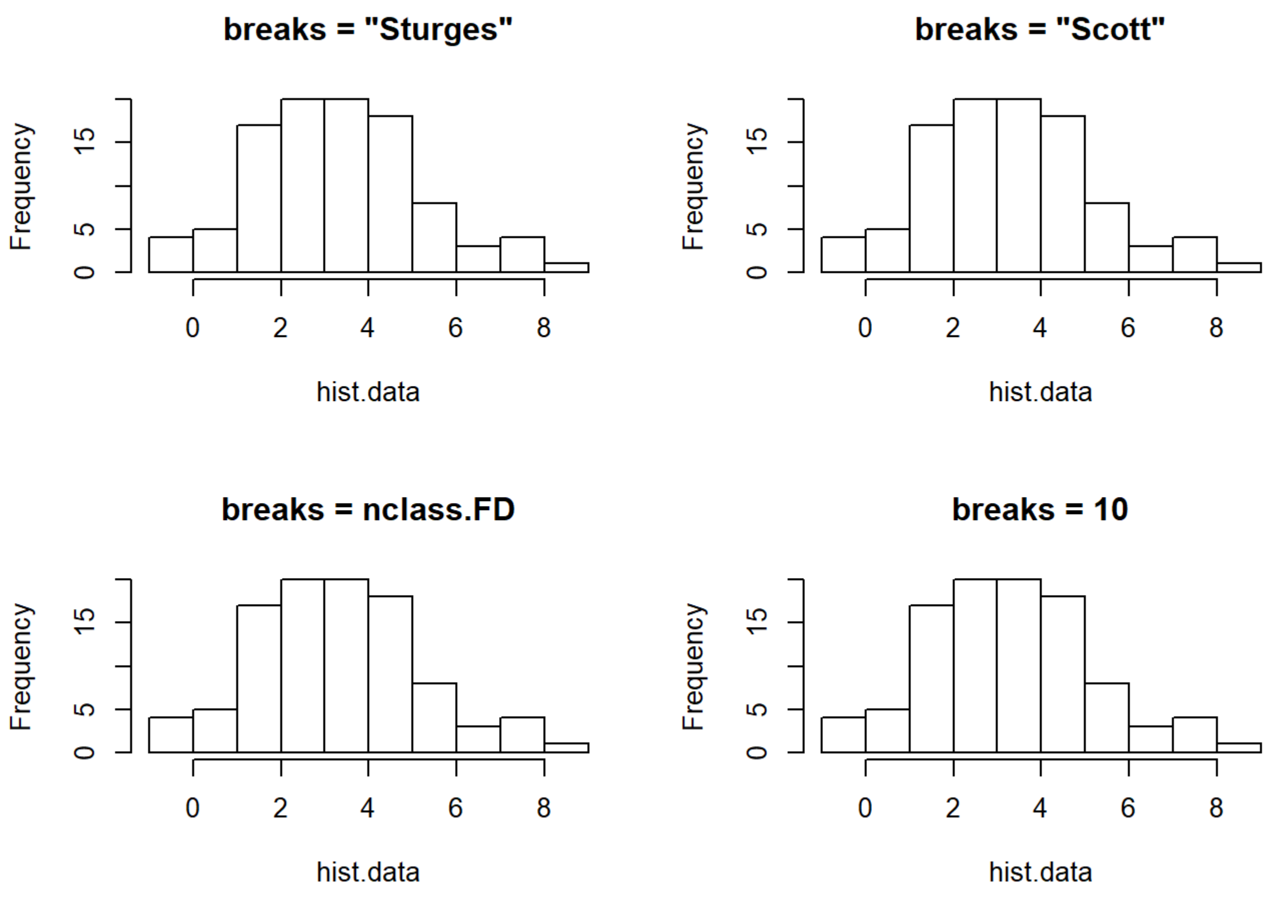
```
pretty(hist.data, nclass.FD(hist.data))
```

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

```
pretty(hist.data, 10)
```

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

```
par(mfrow = c(2, 2))
hist(hist.data, breaks = "Sturges", main = "breaks = \"Sturges\"")
hist(hist.data, breaks = "Scott", main = "breaks = \"Scott\"")
hist(hist.data, breaks = nclass.FD, main = "breaks = nclass.FD")
hist(hist.data, breaks = 10, main = "breaks = 10")
```



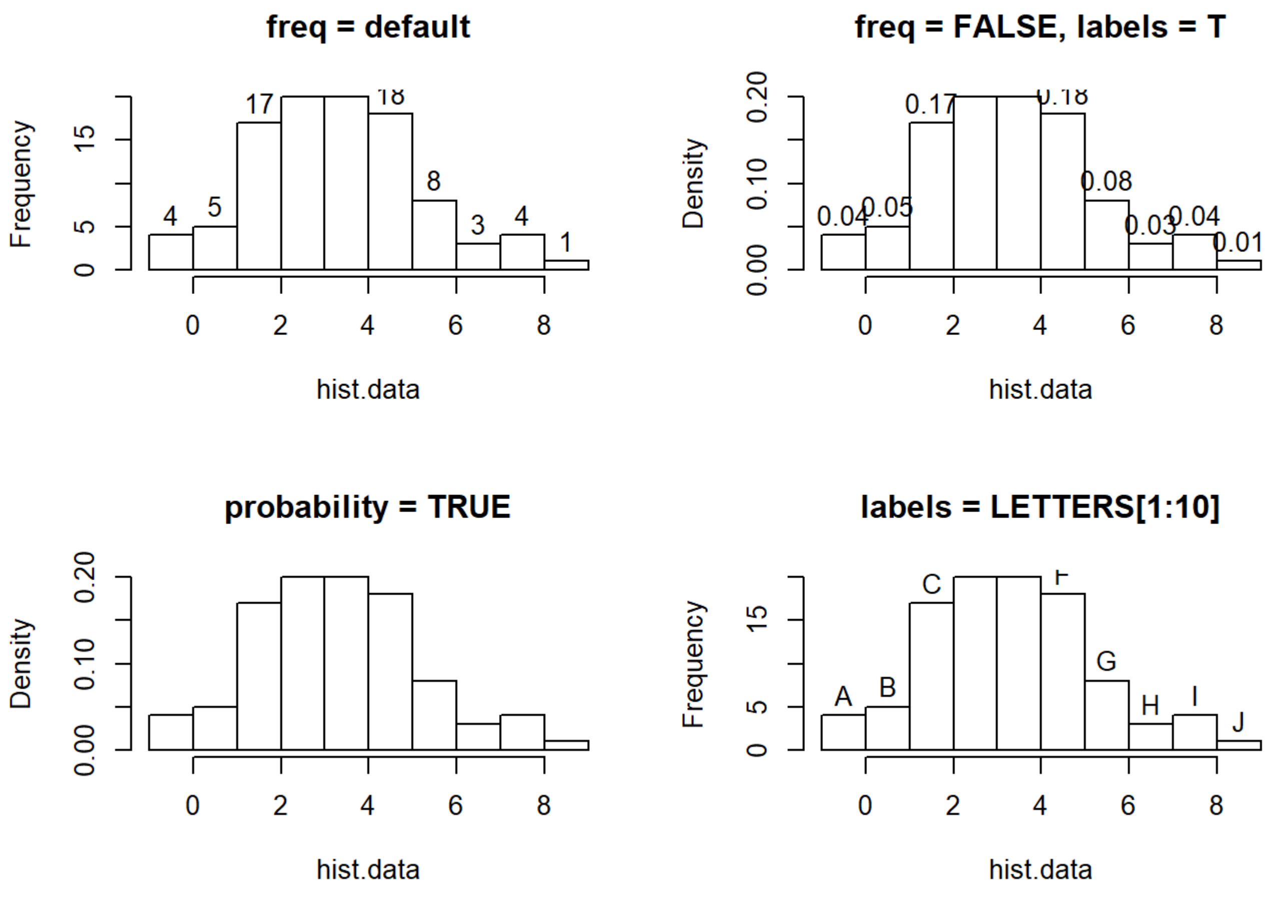
```
par(op)

op <- par(no.readonly = TRUE)
hist.interval <- cut(hist.data, breaks = hist.breaks)
hist.interval
```

```
##   [1] (7,8]  (0,1]  (1,2]  (2,3]  (1,2]  (1,2]  (4,5]  (2,3]  (3,4]  (7,8]
##  [11] (3,4]  (8,9]  (7,8]  (3,4]  (6,7]  (3,4]  (1,2]  (2,3]  (2,3]  (4,5]
##  [21] (4,5]  (4,5]  (5,6]  (0,1]  (5,6]  (3,4]  (4,5]  (4,5]  (1,2]  (2,3]
##  [31] (1,2]  (4,5]  (3,4]  (2,3]  (2,3]  (1,2]  (4,5]  (0,1]  (2,3]  (3,4]
##  [41] (5,6]  (1,2]  (2,3]  (0,1]  (2,3]  (2,3]  (5,6]  (4,5]  (3,4]  (4,5]
##  [51] (2,3]  (2,3]  (3,4]  (6,7]  (4,5]  (3,4]  (-1,0] (3,4]  (3,4]  (1,2]
##  [61] (3,4]  (3,4]  (4,5]  (4,5]  (3,4]  (5,6]  (4,5]  (5,6]  (5,6]  (4,5]
##  [71] (3,4]  (1,2]  (1,2]  (1,2]  (-1,0] (0,1]  (1,2]  (1,2]  (2,3]  (7,8]
##  [81] (3,4]  (3,4]  (3,4]  (2,3]  (1,2]  (2,3]  (2,3]  (2,3]  (-1,0] (4,5]
##  [91] (6,7]  (1,2]  (4,5]  (3,4]  (4,5]  (1,2]  (5,6]  (-1,0] (2,3]  (2,3]
## 10 Levels: (-1,0] (0,1] (1,2] (2,3] (3,4] (4,5] (5,6] (6,7] ... (8,9]
```

```
table(hist.interval)
```

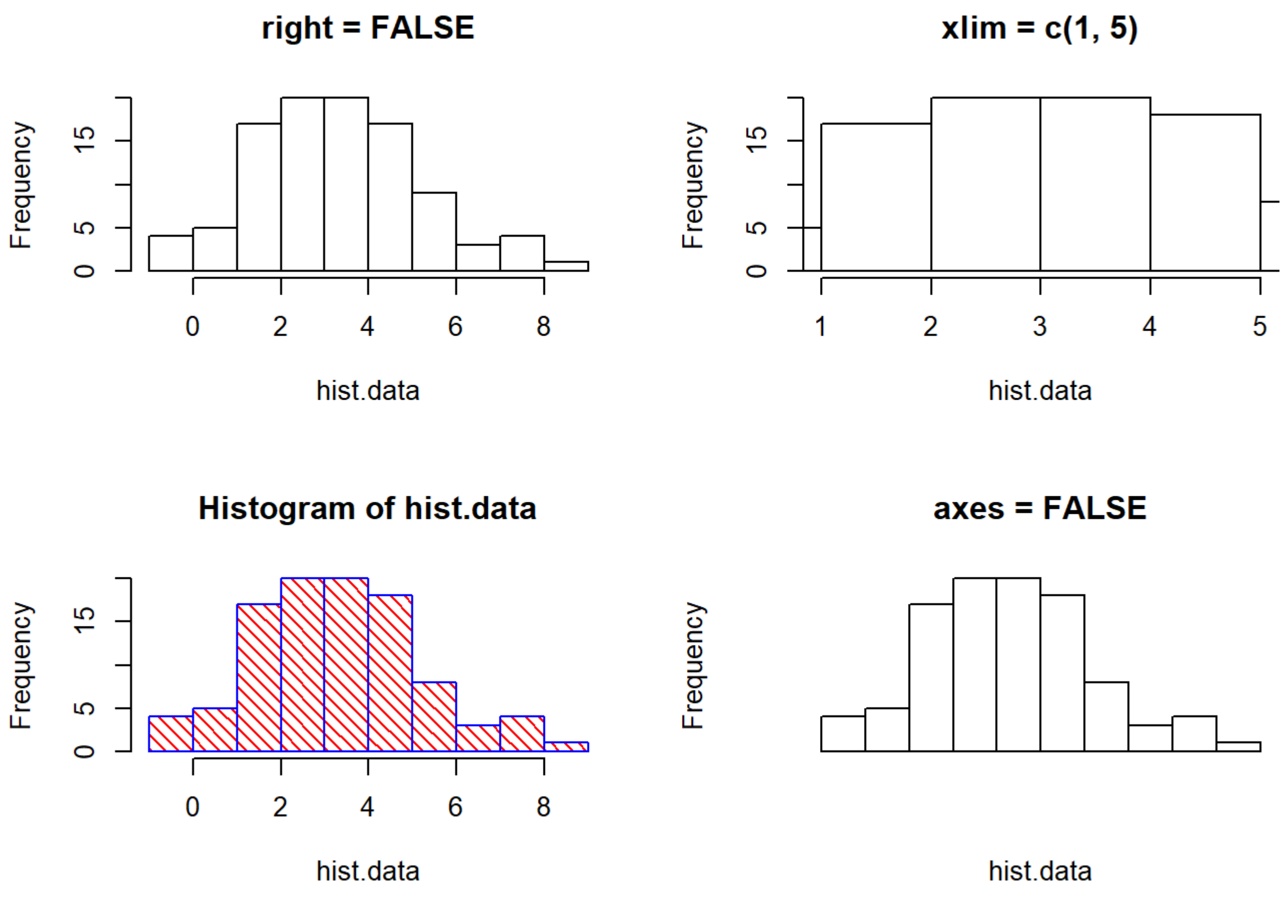
```
## hist.interval
## (-1,0]  (0,1]  (1,2]  (2,3]  (3,4]  (4,5]  (5,6]  (6,7]  (7,8]  (8,9]
##      4      5     17     20     20     18      8      3      4      1
```

```
par(mfrow = c(2, 2))
hist(hist.data, labels = T, main = "freq = default")
hist(hist.data, freq = F, labels = T, main = "freq = FALSE, labels = T")
hist(hist.data, probability = TRUE, main = "probability = TRUE")
hist(hist.data, labels = LETTERS[1:10], main = "labels = LETTERS[1:10]")
```



```
par(op)
```

```
op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
hist(hist.data, right = FALSE, main = "right = FALSE")
hist(hist.data, xlim = c(1, 5), main = "xlim = c(1, 5)")
hist(hist.data, density = 20, col = "red", angle = 135, border = "blue")
hist(hist.data, axes = FALSE, main = "axes = FALSE")
```



```
par(op)

hist(hist.data, plot = FALSE)
```

```
## $breaks
##  [1] -1  0  1  2  3  4  5  6  7  8  9
##
## $counts
##  [1]  4  5 17 20 20 18  8  3  4  1
##
## $density
##  [1] 0.04 0.05 0.17 0.20 0.20 0.18 0.08 0.03 0.04 0.01
##
## $mids
##  [1] -0.5  0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5
##
## $xname
## [1] "hist.data"
##
## $equidist
## [1] TRUE
##
```

```
## attr(,"class")
## [1] "histogram"
```

```
# 3.2.5 pie() 함수

x <- 1:5
x <- c(0, cumsum(x) / sum(x))
x
```

```
## [1] 0.00000000 0.06666667 0.20000000 0.40000000 0.66666667 1.00000000
```

```
dx <- diff(x)
dx
```

```
## [1] 0.06666667 0.13333333 0.20000000 0.26666667 0.33333333
```

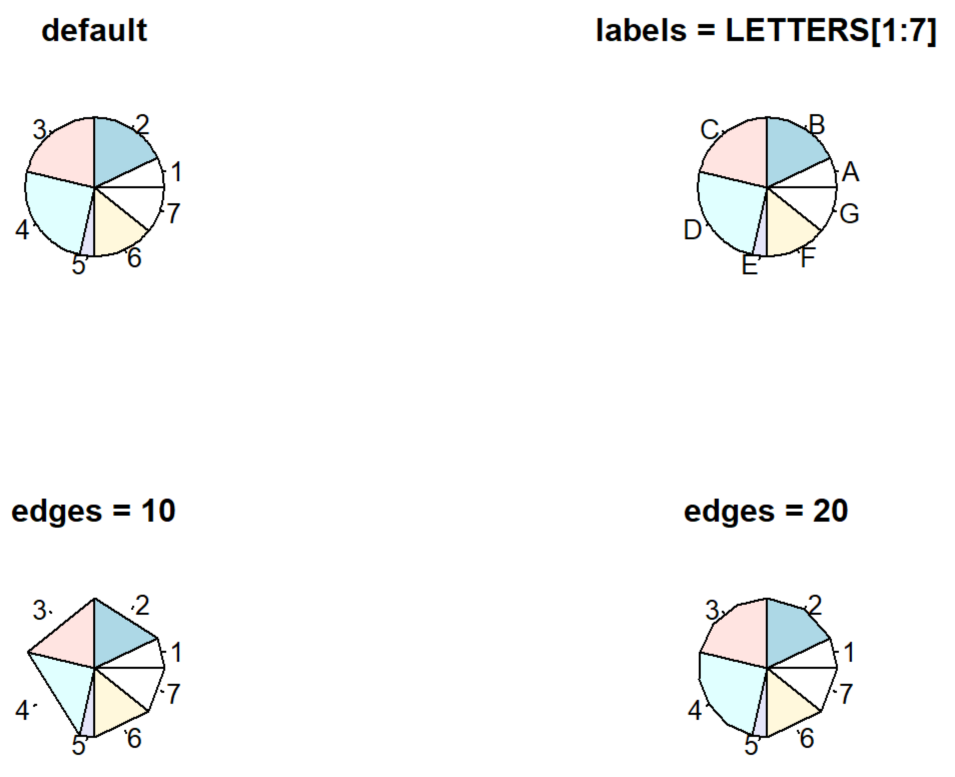
```
sum(dx)
```

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

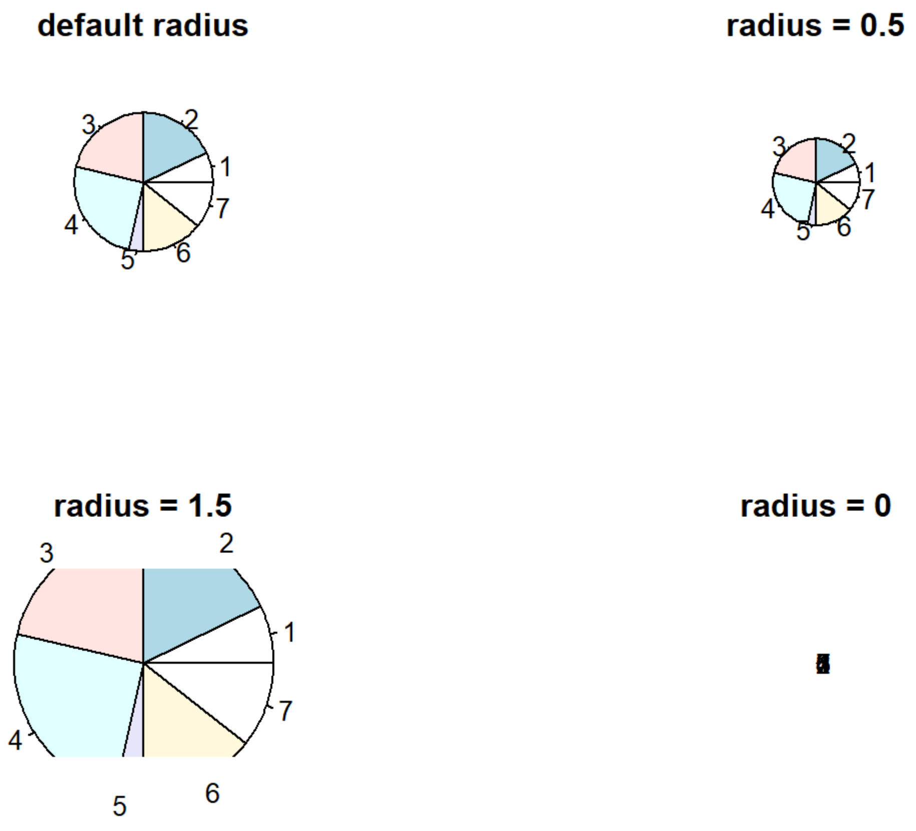
```
op <- par(no.readonly = TRUE)
set.seed(5)
pie.data <- sample(7)
pie.data
```

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

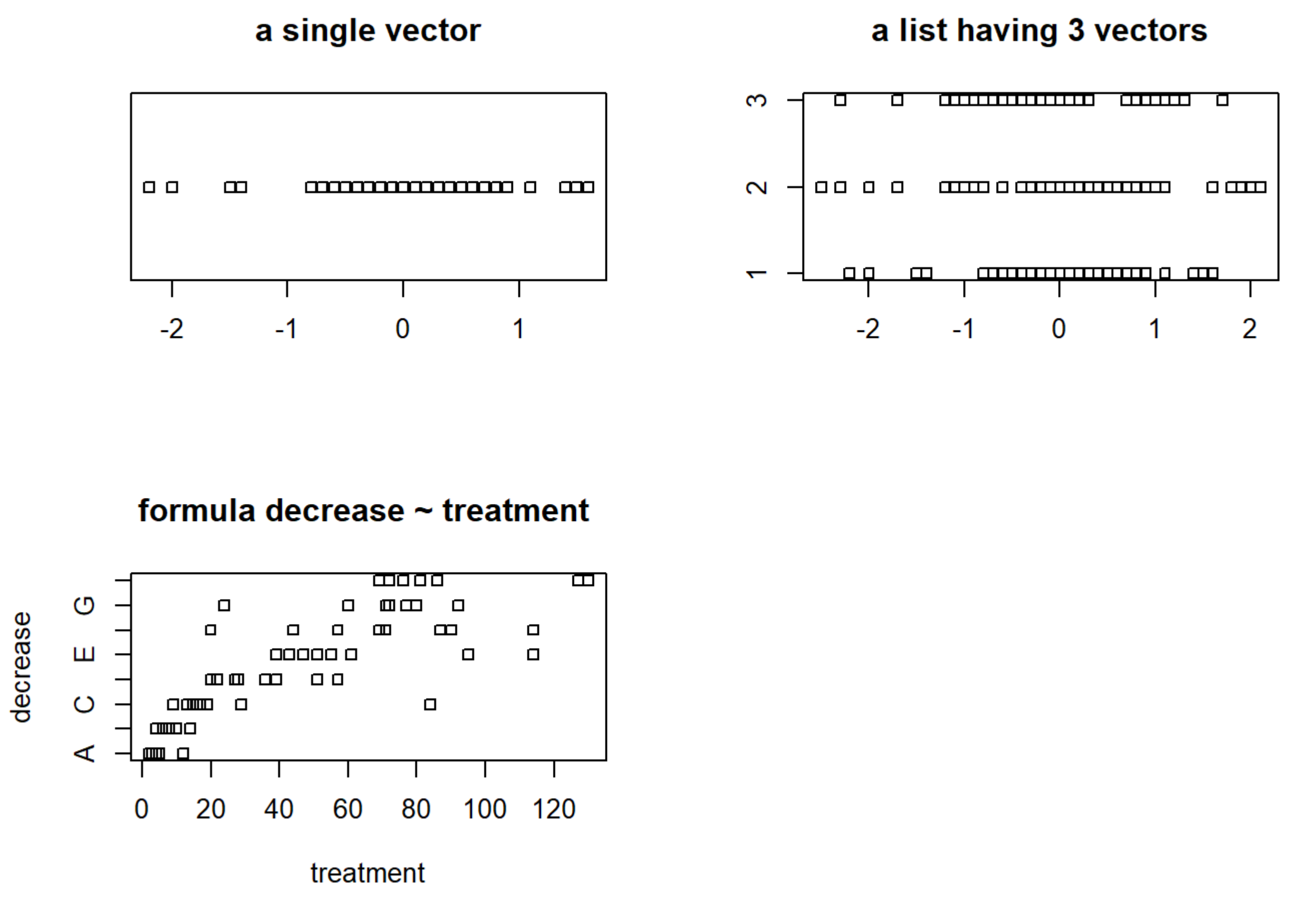
```
par(mfrow = c(2, 2))
pie(pie.data, main = "default")
pie(pie.data, labels = LETTERS[1:7], main = "labels = LETTERS[1:7]")
pie(pie.data, edges = 10, main = "edges = 10")
pie(pie.data, edges = 20, main = "edges = 20")
```



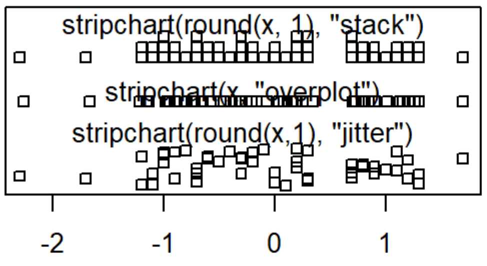
```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
pie(pie.data, main = "default radius")
pie(pie.data, radius = 0.5, main = "radius = 0.5")
pie(pie.data, radius = 1.5, main = "radius = 1.5")
pie(pie.data, radius = 0, main = "radius = 0")
```



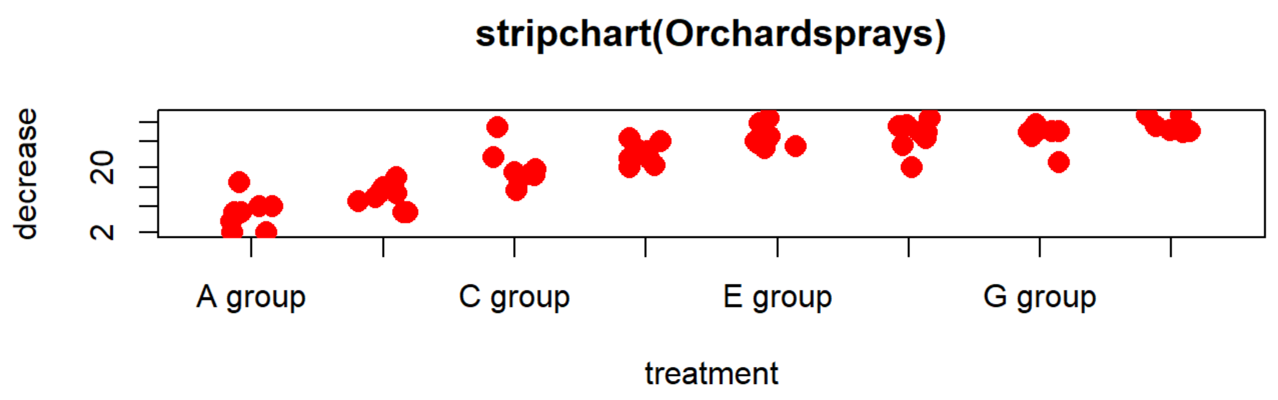
```
par(op)


# 3.2.6 stripchart() 함수

op <- par(no.readonly = TRUE)
set.seed(1)
x <- round(rnorm(50), 1)
set.seed(2)
y <- round(rnorm(50), 1)
set.seed(3)
z <- round(rnorm(50), 1)
strip.data <- list(x, y, z)
par(mfrow = c(2, 2))
stripchart(x, main = "a single vector")              # 벡터
stripchart(strip.data, main = "a list having 3 vectors")  # 리스트
with(OrchardSprays, stripchart(decrease ~ treatment,      # formula
     main = "formula decrease ~ treatment ", xlab = "treatment", ylab = "decrease"))
par(op)
```

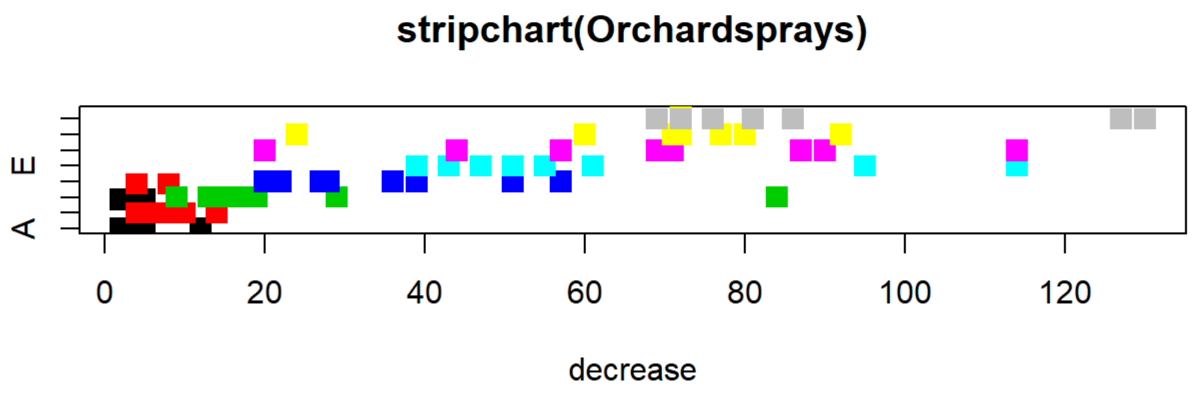


```
set.seed(3)
x <- rnorm(50)
xr <- round(x, 1)
stripchart(x)
m <- mean(par("usr")[1:2])
text(m, 1.04, "stripchart(x, \"overplot\")")
stripchart(xr, method = "stack", add = TRUE, at = 1.2)
text(m, 1.35, "stripchart(round(x, 1), \"stack\")")
stripchart(xr, method = "jitter", add = TRUE, at = 0.7)
text(m, 0.85, "stripchart(round(x,1), \"jitter\")")

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 1))
```

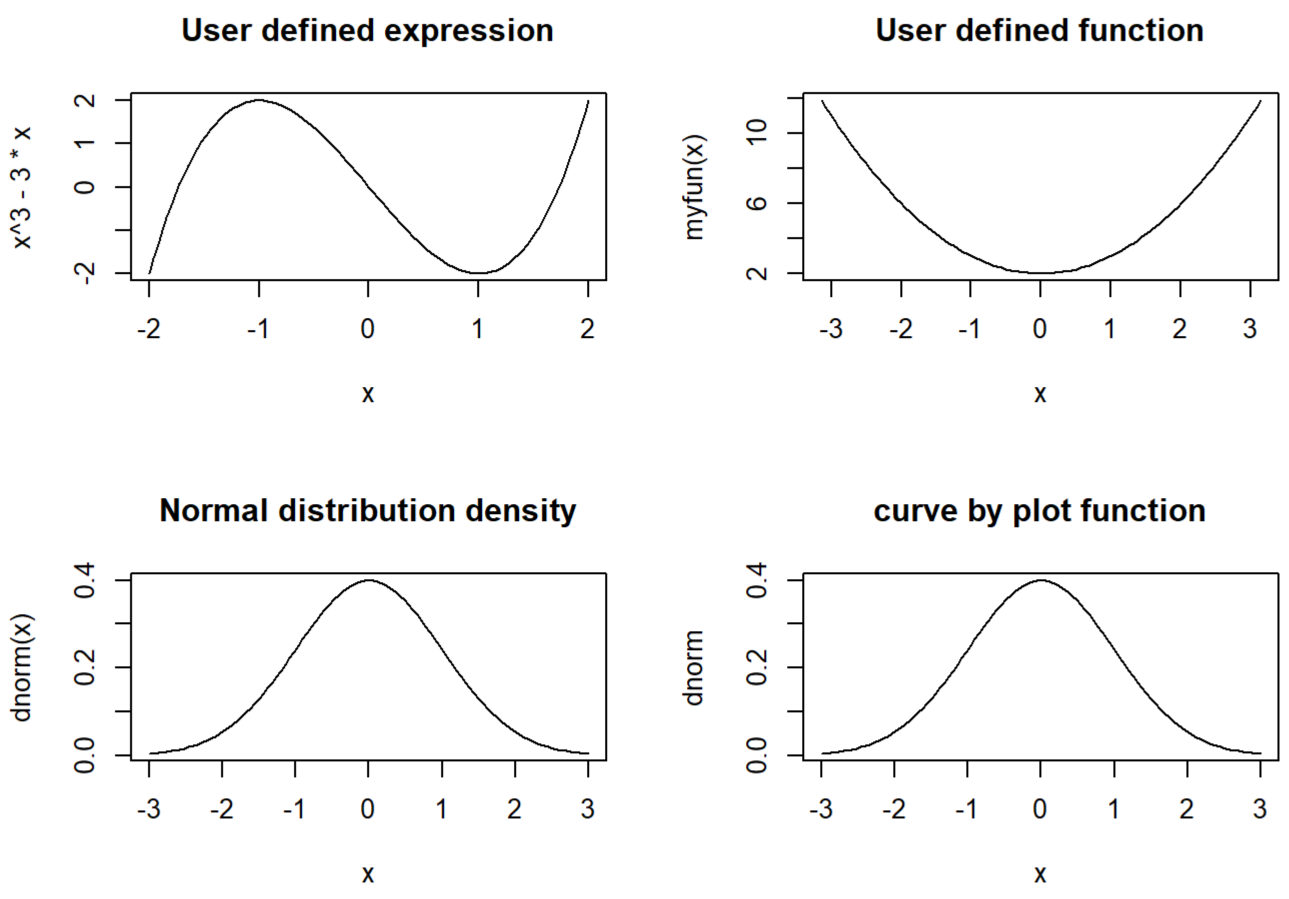


```
with(OrchardSprays, stripchart(decrease ~ treatment, method = "jitter",
      jitter = 0.2, col = "red", pch = 16, cex = 1.5, vertical = TRUE, log = "y",
      main="stripchart(Orchardsprays)", xlab = "treatment", ylab = "decrease",
      group.names = paste(LETTERS[1:8], "group")))
with(OrchardSprays, stripchart(decrease ~ treatment, method = "stack",
      offset = 1/2, col = 1:8, pch = 15, cex = 1.5, main = "stripchart(Orchardsprays)"))
```

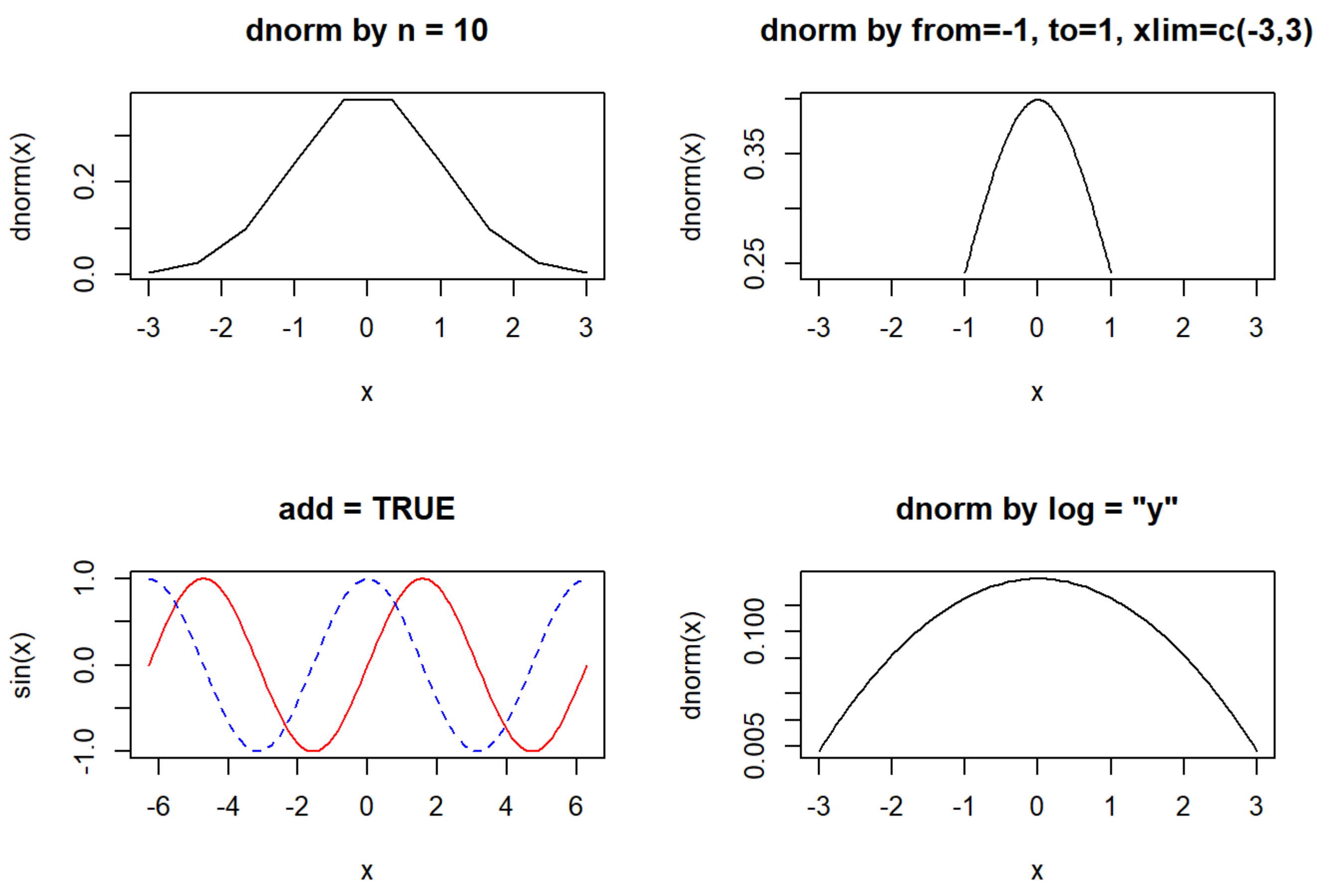




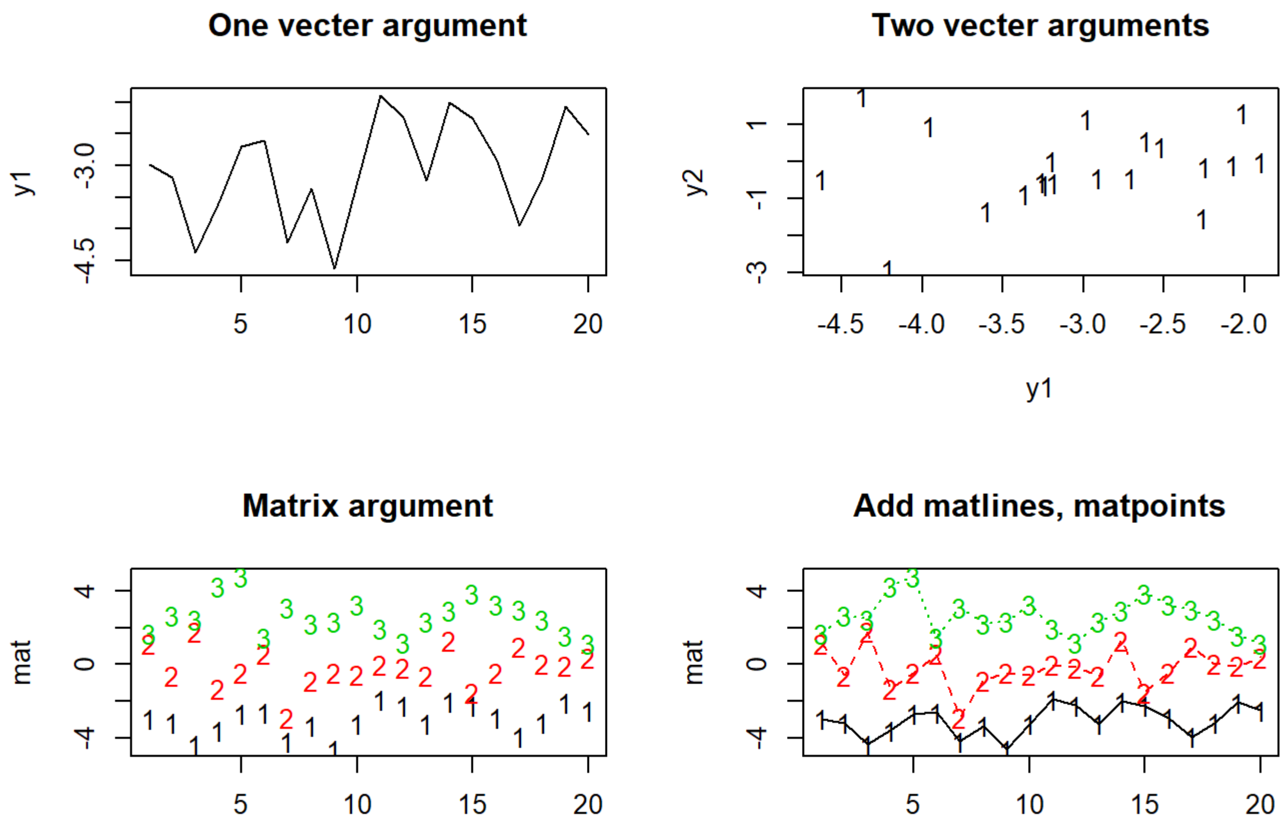
```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
# (1) Expression
curve(x ^ 3 - 3 * x, -2, 2)
title(main = "User defined expression")
myfun <- function(x) x ^ 2 + 2
# (2) User Function
curve(myfun, -pi, pi)
title(main = "User defined function")
# (3) R Function
curve(dnorm, from = -3, to = 3)
title(main = "Normal distribution density")
# (4) plot Function
plot(dnorm, from = -3, to = 3)
title(main = "curve by plot function")
```



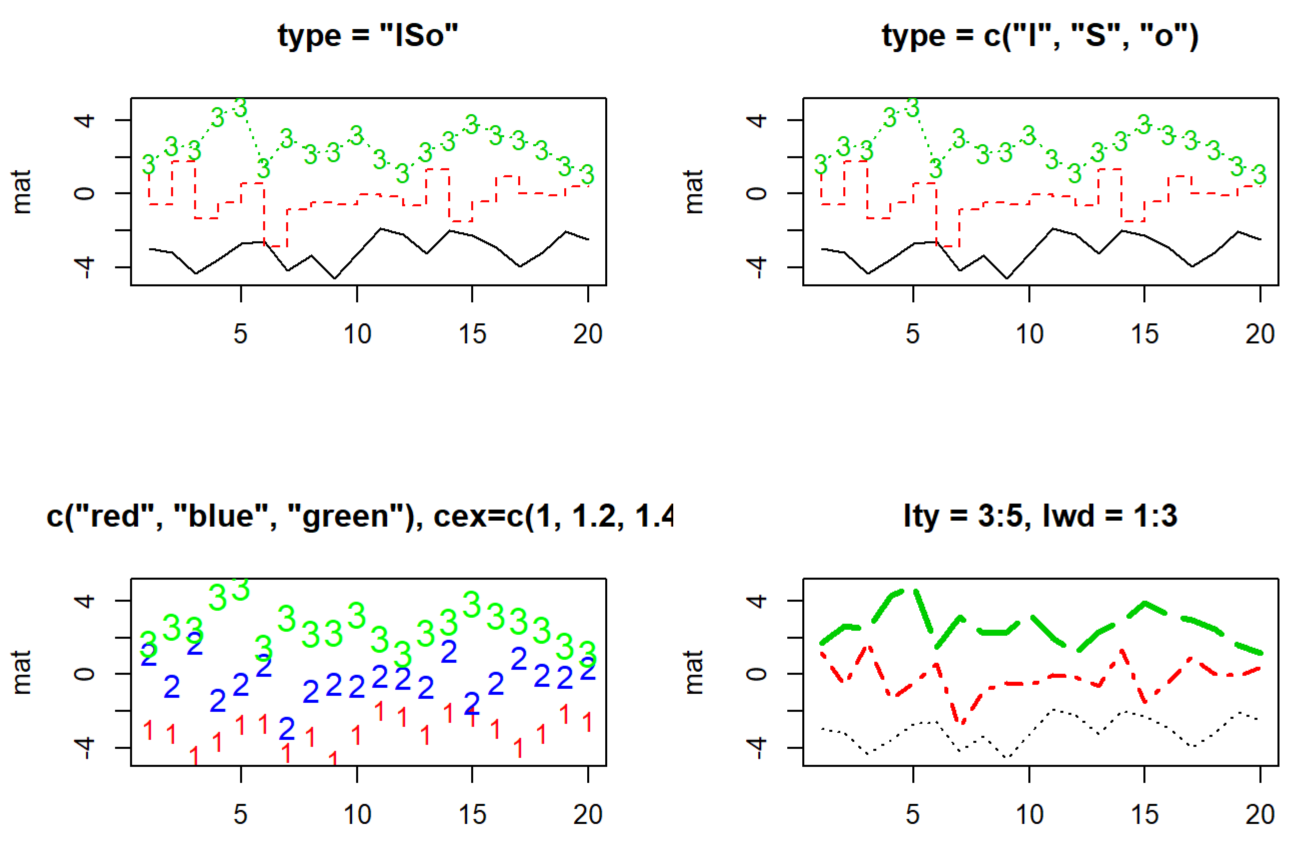
```
par(op)



op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
curve(dnorm, from = -3, to = 3, n = 10)
title(main = "dnorm by n = 10")
curve(dnorm, from = -1, to = 1, xlim = c(-3, 3))
title(main = "dnorm by from=-1, to=1, xlim=c(-3,3)")
curve(sin, from = -2 * pi, to = 2 * pi, lty = 1, col = "red")
curve(cos, from = -2 * pi, to = 2 * pi, lty = 2, col = "blue", add = T)
title(main = "add = TRUE")
curve(dnorm, from = -3, to = 3, log = "y")
title(main = "dnorm by log = \"y\"")
```



```
par(op)


# 3.2.8 matplot(), matpoints(), matlines() 함수

op <- par(no.readonly = TRUE)
set.seed(10)
y1 <- rnorm(20, mean = -3, sd = 1)
set.seed(20)
y2 <- rnorm(20, mean = 0, sd = 1)
set.seed(30)
y3 <- rnorm(20, mean = 3, sd = 1)
mat <- cbind(y1, y2, y3)
par(mfrow = c(2, 2))
matplot(y1, type = "l", main = "One vecter argument")
matplot(y1, y2, main = "Two vecter arguments")
matplot(mat, main = "Matrix argument")
matplot(mat, type = "n", main = "Add matlines, matpoints")
matlines(mat)
matpoints(mat)
```



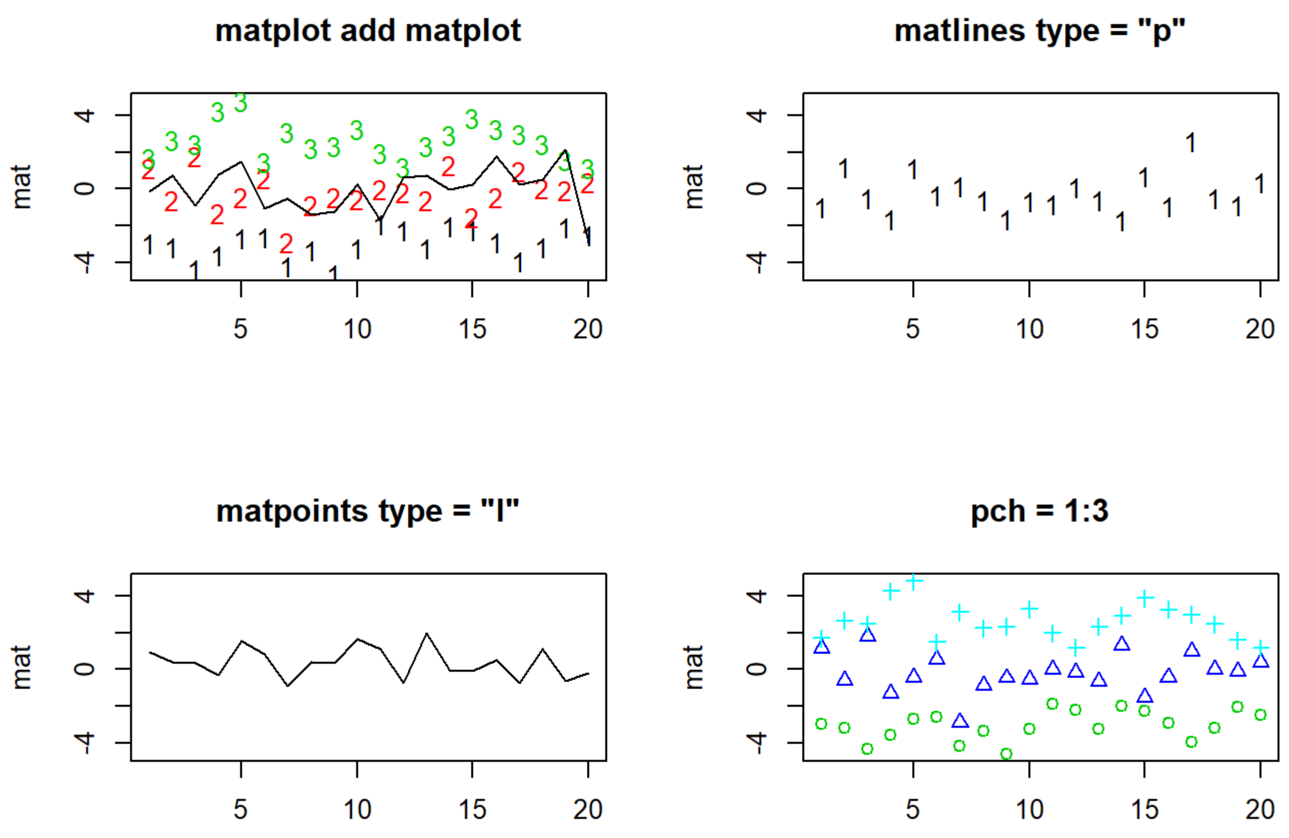
```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
matplot(mat, type = "lSo", main = "type = \"lSo\"")
matplot(mat, type = c("l","S","o"), main = "type = c(\"l\", \"S\", \"o\")")
matplot(mat, col = c("red","blue","green"), cex = c(1, 1.2, 1.4))
title(main = "c(\"red\", \"blue\", \"green\"), cex=c(1, 1.2, 1.4)")
matplot(mat, type = "l", lty = 3:5, lwd = 1:3)
title(main = "lty = 3:5, lwd = 1:3")
```



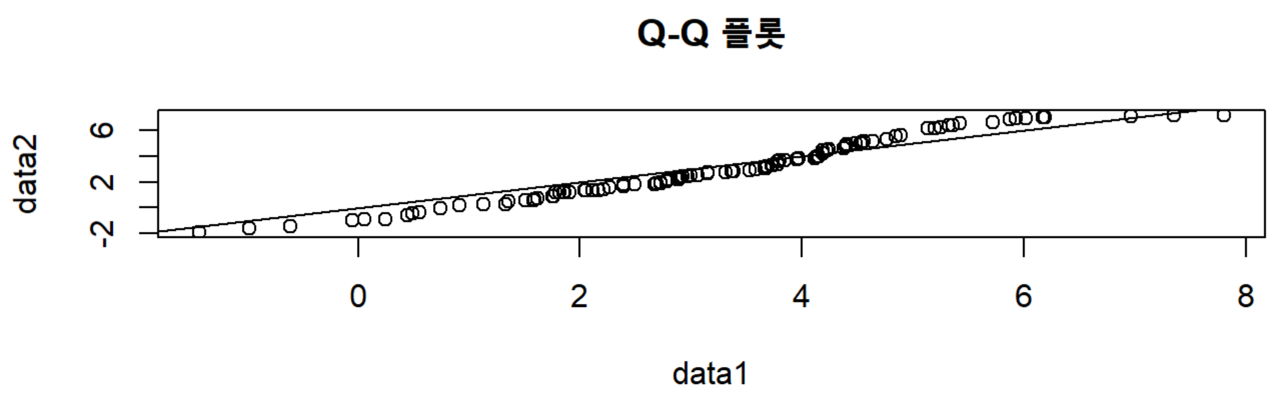
```
par(op)

# 3.2.9 qqnrom(), qqline(), qqplot() 함수

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
matplot(mat)
matplot(rnorm(20), type = "l", add = TRUE)
title(main = "matplot add matplot")
matplot(mat, type = "n")
matlines(rnorm(20), type = "p")
title(main = "matlines type = \"p\"")
matplot(mat, type = "n")
matpoints(rnorm(20), type = "l")
title(main = "matpoints type = \"l\"")
matplot(mat, pch = 1:3, col = 3:5, verbose = TRUE) # matplot 정보가 출력됨
```

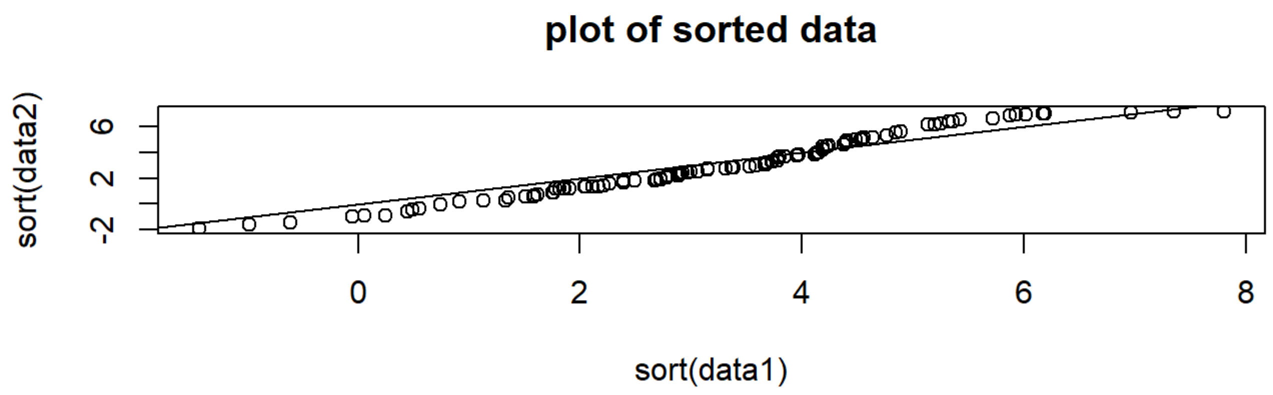
```
## matplot: doing 3 plots with  col= ("3" "4" "5") pch= ("1" "2" "3") ...
```

```
title(main = "pch = 1:3")
```



```
par(op)

op <- par(no.readonly = TRUE)
set.seed(1)
data1 <- round(rnorm(100, 3, 2), 2)
set.seed(2)
data2 <- round(rnorm(100, 3, 2), 2)
par(mfrow = c(2, 1))
qqplot(data1, data2, main = "Q-Q 플롯")
abline(0,1)
plot(sort(data1), sort(data2), main = "plot of sorted data")
abline(0,1)
```





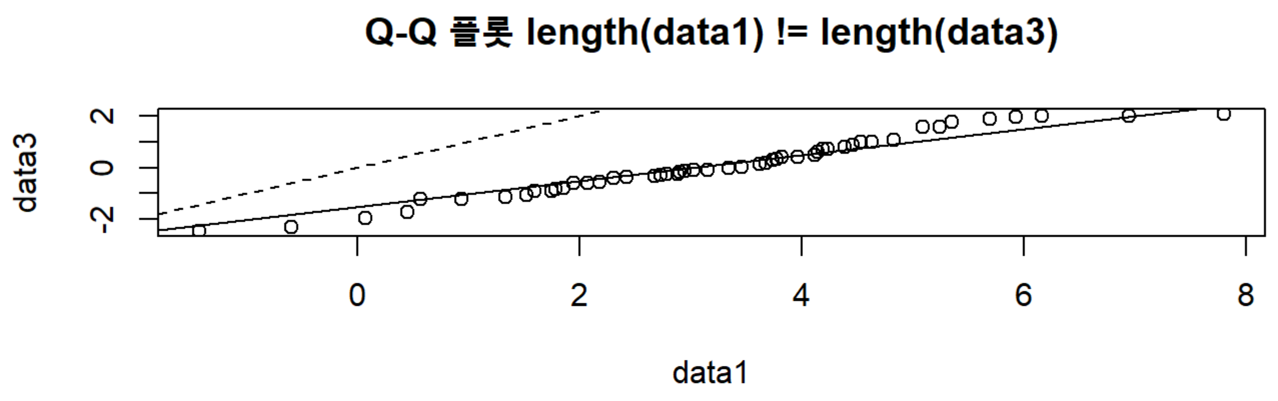
```
par(op)

x <- round(rnorm(10), 2)
y <- round(rnorm(10), 2)
qqplot(x, y, plot.it = FALSE)
.Last.value
```

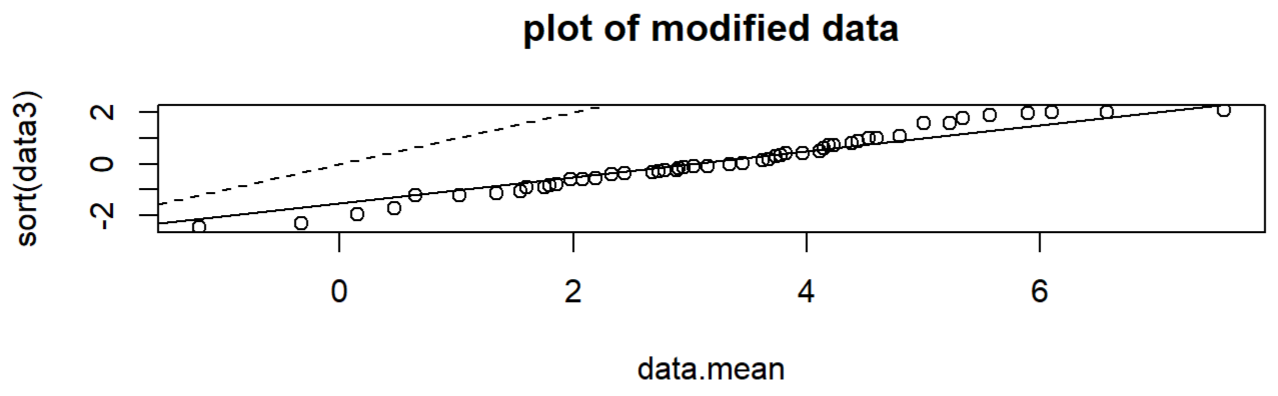
```
## $help_type
## NULL
```

```
qq <- qqplot(x, y, plot.it = FALSE)
qq
```

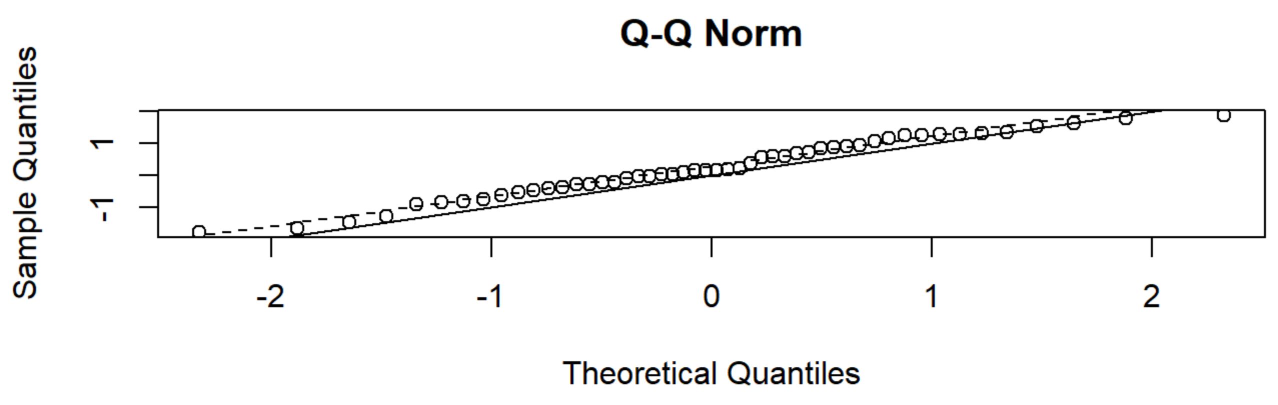
```
## $x
##  [1] -0.86 -0.75 -0.42 -0.35 -0.31  0.26  0.94  1.07  2.01  2.05
##
## $y
##  [1] -1.03 -0.78 -0.25  0.11  0.46  0.47  0.56  1.15  1.23  1.36
```

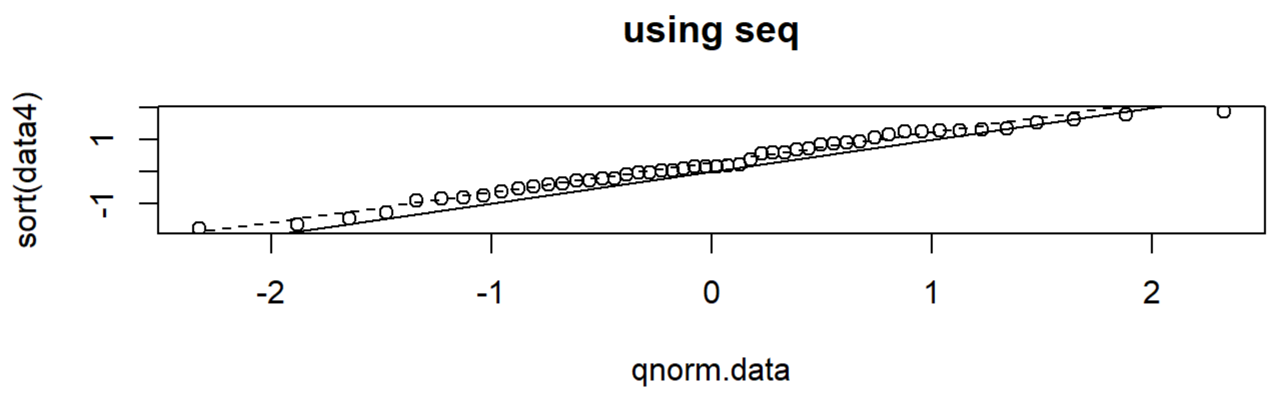
```
op <- par(no.readonly = TRUE)
par(mfrow = c(2,1))
set.seed(1)
data1 <- rnorm(100, 3, 2)
set.seed(2)
data3 <- round(rnorm(50), 2)
qqplot(data1, data3, main = "Q-Q 플롯 length(data1) != length(data3)")
abline(-3/2, 1/2)
abline(0, 1, lty = 2)
seq.odd <- seq(1, 99, 2)
seq.even <- seq(2, 100, 2)
data.odd <- sort(data1)[seq.odd]
data.even <- sort(data1)[seq.even]
data.mean <- (data.odd + data.even ) / 2
plot(data.mean, sort(data3), main = "plot of modified data")
abline(-3/2, 1/2)
abline(0, 1, lty = 2)
```





```
par(op)

set.seed(4)
data4 <- rnorm(50)
seq.norm <- seq(1, 99, 2) / 100   # 분위수를 구하기 위한 시퀀스
qnorm.data <- qnorm(seq.norm)  # 표준 정규분포의 분위수 계산
par(mfrow = c(2,1))
qqnorm(data4, main = "Q-Q Norm")
used.qqnorm <- .Last.value    # qqnorm이 사용한 데이터 추출
abline(0, 1)
qqline(data4, lty = 2)
plot(qnorm.data, sort(data4), main = "using seq")
abline(0, 1)
qqline(data4, lty = 2)
```





```
sort(data4)      # data4를 정렬한 순서통계량
```

```
##  [1] -1.79738202 -1.68804858 -1.48218912 -1.28124663 -0.92802811
##  [6] -0.86214614 -0.82099358 -0.75421121 -0.63754350 -0.54249257
## [11] -0.46589588 -0.40451983 -0.37565514 -0.28344457 -0.28294368
## [16] -0.22740542 -0.21314452 -0.10036844 -0.04513712 -0.04420400
## [21]  0.01571945  0.03435191  0.09884369  0.15346418  0.16516902
## [26]  0.16902677  0.18153538  0.21675486  0.38305734  0.56660450
## [31]  0.59289694  0.59598058  0.68927544  0.72390416  0.86113187
## [36]  0.89114465  0.90983915  0.93409617  1.05193258  1.16502684
## [41]  1.24018084  1.25588403  1.28825688  1.29251234  1.30762236
## [46]  1.34370863  1.54081498  1.63561800  1.77686321  1.89653987
```

```
sort(used.qqnorm$y)      # qqnorm이 사용한 y 좌표값
```

```
## NULL
```

```
qnorm.data      # 표준 정규분포의 분위수
```

```
##  [1] -2.32634787 -1.88079361 -1.64485363 -1.47579103 -1.34075503
##  [6] -1.22652812 -1.12639113 -1.03643339 -0.95416525 -0.87789630
## [11] -0.80642125 -0.73884685 -0.67448975 -0.61281299 -0.55338472
## [16] -0.49585035 -0.43991317 -0.38532047 -0.33185335 -0.27931903
## [21] -0.22754498 -0.17637416 -0.12566135 -0.07526986 -0.02506891
## [26]  0.02506891  0.07526986  0.12566135  0.17637416  0.22754498
## [31]  0.27931903  0.33185335  0.38532047  0.43991317  0.49585035
## [36]  0.55338472  0.61281299  0.67448975  0.73884685  0.80642125
## [41]  0.87789630  0.95416525  1.03643339  1.12639113  1.22652812
## [46]  1.34075503  1.47579103  1.64485363  1.88079361  2.32634787
```

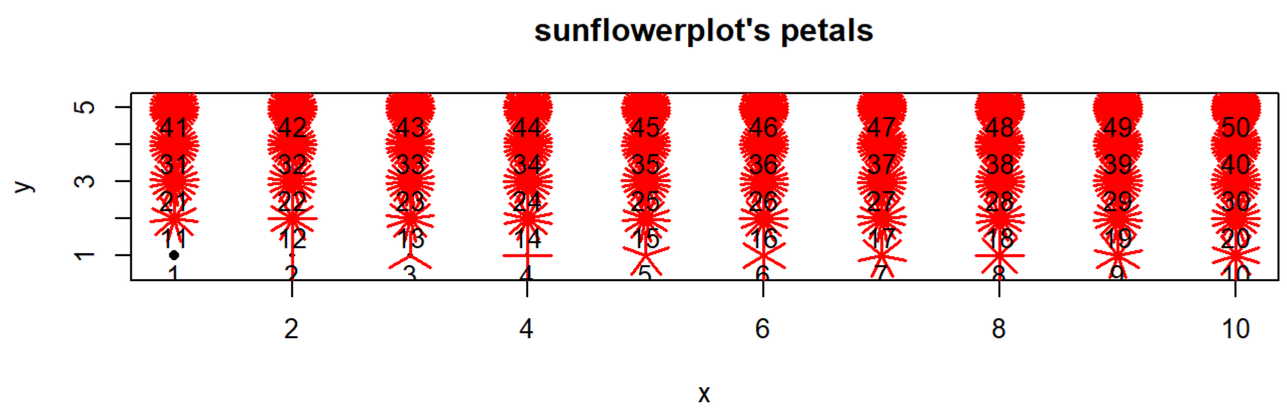
```
sort(used.qqnorm$x)    # qqnorm이 사용한 x 좌표값
```

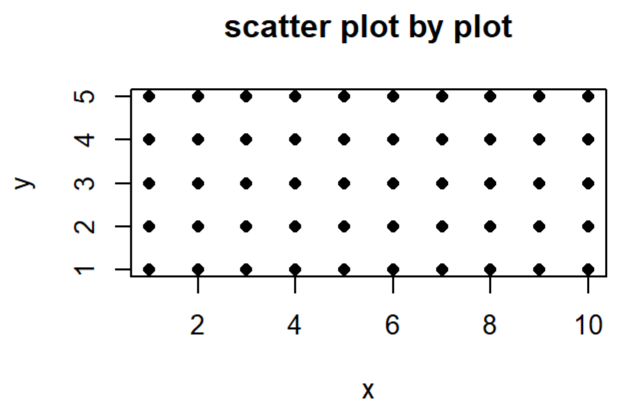
```
## NULL
```

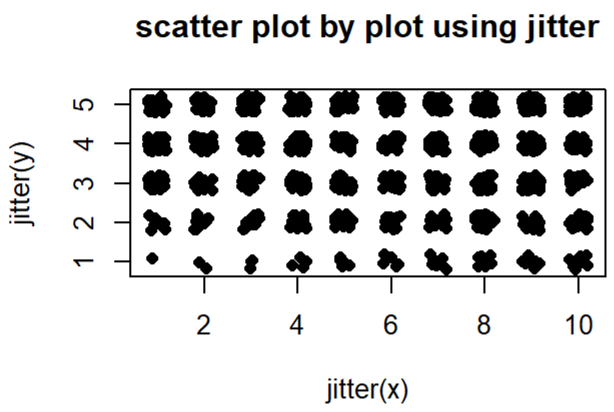
```
# 3.2.10 sunflowerplot() 함수

x <- NULL
y <- NULL
for (i in 1:50) {
    x <- c(x, rep(ifelse(i%%10 == 0, 10, i %%10), i))
    y <- c(y, rep((i-1) %/% 10 + 1, i))
  }
# (1)
t(table(x, y))
```

```
##     x
## y     1  2  3  4  5  6  7  8  9 10
##    1  1  2  3  4  5  6  7  8  9 10
##    2 11 12 13 14 15 16 17 18 19 20
##    3 21 22 23 24 25 26 27 28 29 30
##    4 31 32 33 34 35 36 37 38 39 40
##    5 41 42 43 44 45 46 47 48 49 50
```

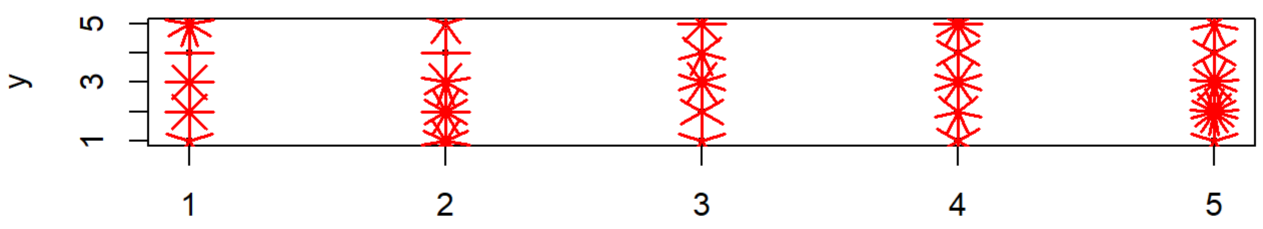
```
layout(matrix(c(1, 1, 2, 3), ncol = 2, byrow = T))
sunflowerplot(x, y, ylim = c(0.5, 5.2))
title(main = "sunflowerplot's petals")
text(x=ifelse(1:50 %% 10==0, 10, 1:50 %% 10), y=((1:50 - 1)%/%10+1) - 0.5,
      as.character(1:50))
plot(x, y, pch = 16)
title(main = "scatter plot by plot")
plot(jitter(x), jitter(y), pch = 16)
title(main = "scatter plot by plot using jitter")
```

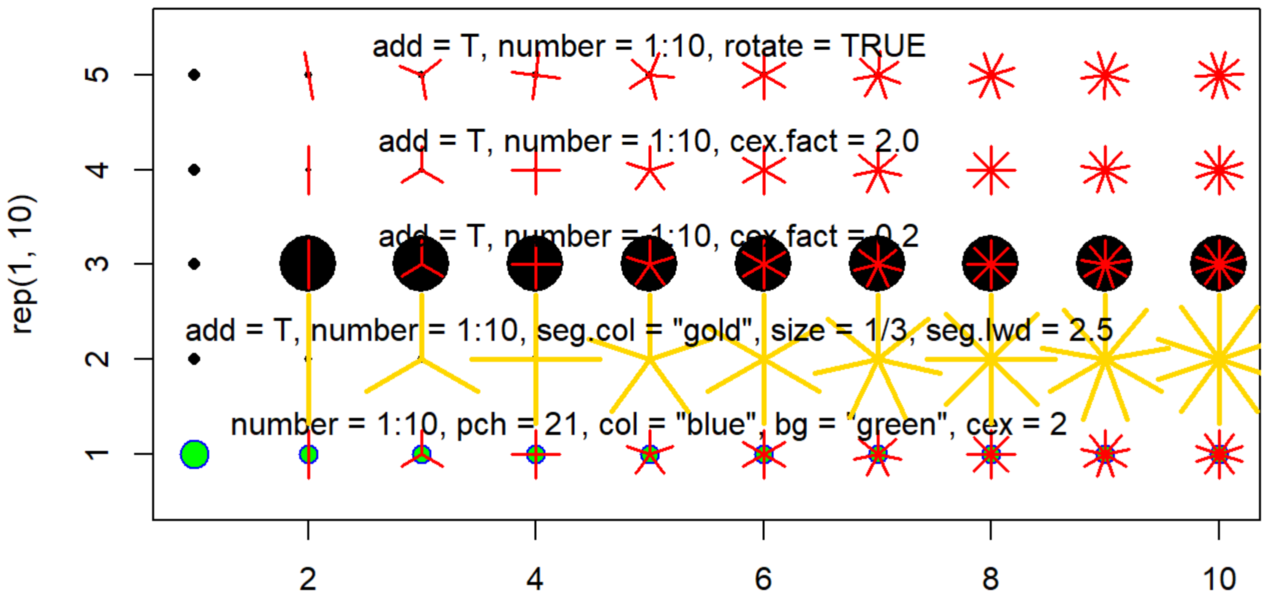




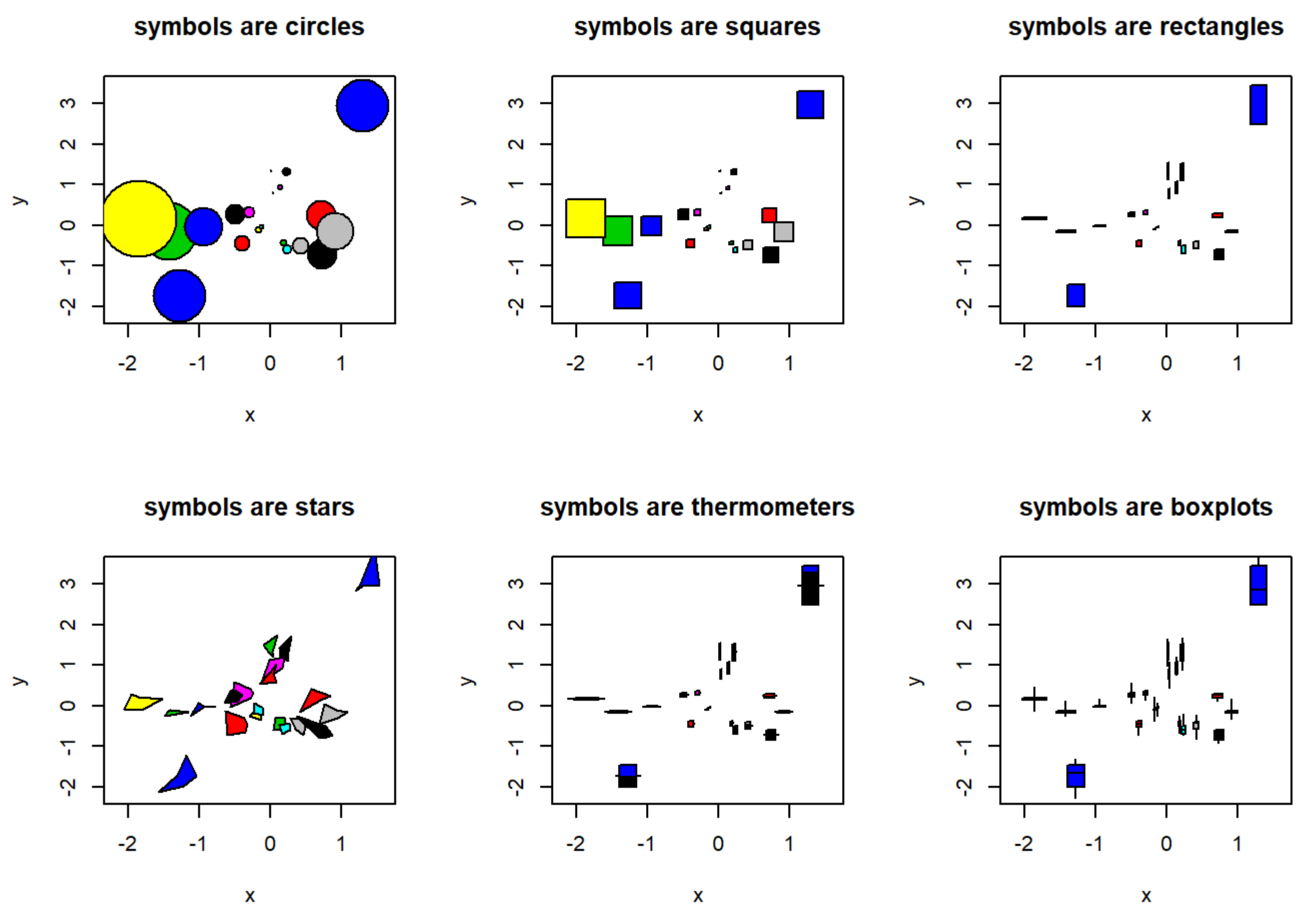


```
op <- par(no.readonly = TRUE)
set.seed(101)
x <- sample(1:5, 200, replace = T)
set.seed(102)
y <- sample(1:5, 200, replace = T)
par(mfrow = c(2, 1), mar = c(2.1, 4.1, 2.1, 2.1), fig = c(0, 1, 0.7, 1))
sunflowerplot(x, y)
par(fig=c(0, 1, 0, 0.7), new = T)
sunflowerplot(1:10, rep(1, 10), ylim = c(0.5, 5.5), number = 1:10,
          pch = 21, col = "blue", bg = "green", cex = 2)
text(5, 1.3, "number = 1:10, pch = 21, col = \"blue\", bg = \"green\", cex = 2", adj = 0.5)
sunflowerplot(1:10, rep(2, 10), add = T, number = 1:10,
          seg.col = "gold", size = 1/3, seg.lwd = 2.5)
text(5, 2.3, "add = T, number = 1:10, seg.col = \"gold\", size = 1/3, seg.lwd = 2.5", adj = 0.5)
sunflowerplot(1:10, rep(3, 10), add = T, number = 1:10, cex.fact = 0.2)
text(5, 3.3, "add = T, number = 1:10, cex.fact = 0.2", adj = 0.5)
sunflowerplot(1:10, rep(4, 10), add = T, number = 1:10, cex.fact = 2.0)
text(5, 4.3, "add = T, number = 1:10, cex.fact = 2.0", adj = 0.5)
sunflowerplot(1:10, rep(5, 10), add = T, number = 1:10, rotate = TRUE)
text(5, 5.3, "add = T, number = 1:10, rotate = TRUE", adj = 0.5)
```

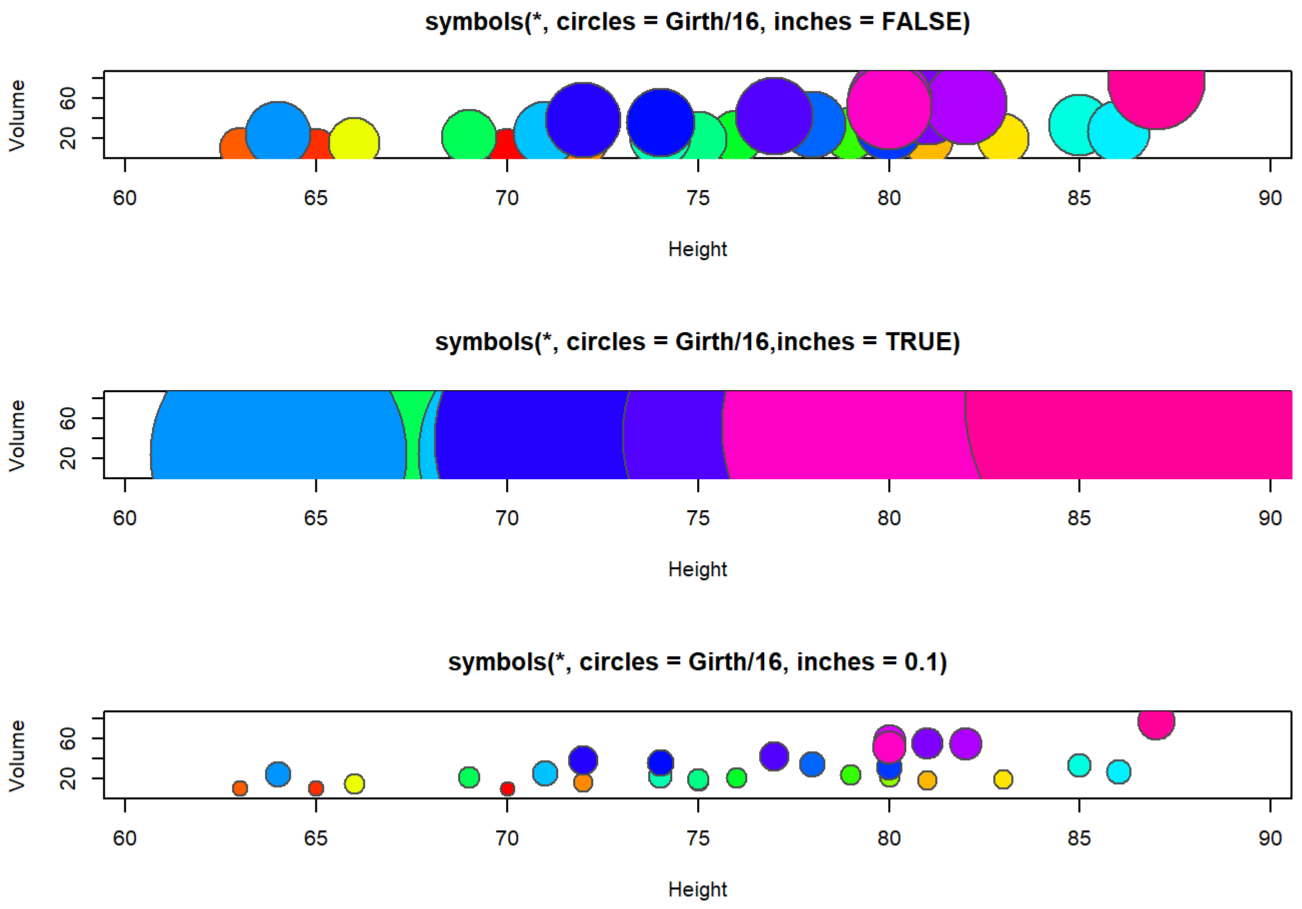




```
par(op)


# 3.2.11 symbols() 함수

op <- par(no.readonly = TRUE)
x <- round(rnorm(20), 2)
y <- round(rnorm(20), 2)
z1 <- abs(round(rnorm(20), 2))
z2 <- abs(round(rnorm(20), 2))
z3 <- round(runif(20), 2)
z4 <- round(runif(20), 2)
z5 <- round(runif(20), 2)
par(mfrow = c(2, 3))
symbols(x, y, circles = abs(x), inches = 0.2, bg = 1:20)
title(main = "symbols are circles")
symbols(x, y, squares = abs(x), inches = 0.2, bg = 1:20)
title(main = "symbols are squares")
symbols(x, y, rectangles = cbind(abs(x), abs(y)), inches = 0.2, bg = 1:20)
title(main = "symbols are rectangles")
symbols(x, y, stars = cbind(abs(x), abs(y), z1, z2, z3), inches = 0.2, bg=1:20)
title(main = "symbols are stars")
symbols(x, y, thermometers = cbind(abs(x), abs(y), z4), inches=0.2, bg=1:20)
title(main = "symbols are thermometers")
symbols(x, y, boxplots = cbind(abs(x), abs(y), z3, z4, z5), inches=0.2, bg=1:20)
title(main = "symbols are boxplots")
```



```
par(op)



op <- par(no.readonly = TRUE)
N <- nrow(trees)
palette(rainbow(N, end = 0.9))
par(mfrow = c(3, 1))
with(trees, {
   symbols(Height, Volume, circles = Girth/16, inches = FALSE, bg = 1:N,
          fg = "gray30", main = "symbols(*, circles = Girth/16, inches = FALSE)")
   symbols(Height, Volume, circles = Girth/16, inches = TRUE, bg = 1:N,
          fg = "gray30", main = "symbols(*, circles = Girth/16,inches = TRUE)")
   symbols(Height, Volume, circles = Girth/16, inches = 0.1, bg = 1:N,
          fg = "gray30", main = "symbols(*, circles = Girth/16, inches = 0.1)")
})
```



```
palette("default")
par(op)


# 3.2.12 assocplot() 함수

x <- margin.table(HairEyeColor, c(1, 2))
x
```

```
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    68   20    15     5
##   Brown   119   84    54    29
##   Red      26   17    14    14
##   Blond     7   94    10    16
```

```
assocplot(x, main = "Relation between hair and eye color")
chisq.test(x)
```

```
##
##  Pearson's Chi-squared test
##
## data:  x
## X-squared = 138.29, df = 9, p-value < 2.2e-16
```

```
op <- par(no.readonly = TRUE)
x <- matrix(rep(120, 4), ncol = 2, dimnames = list(c("row1", "row2"), c("col1", "col2")))
```

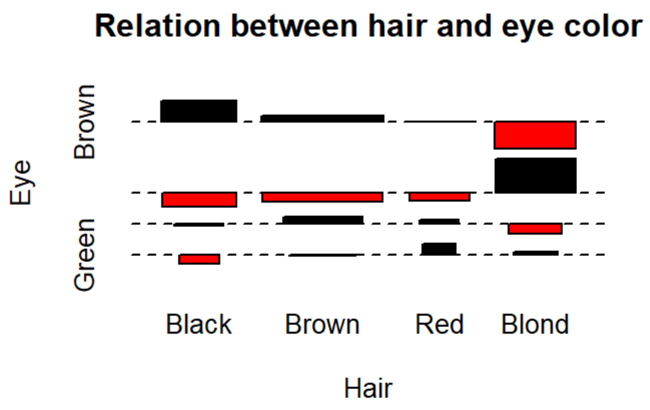
```
x
```

```
##      col1 col2
## row1  120  120
## row2  120  120
```

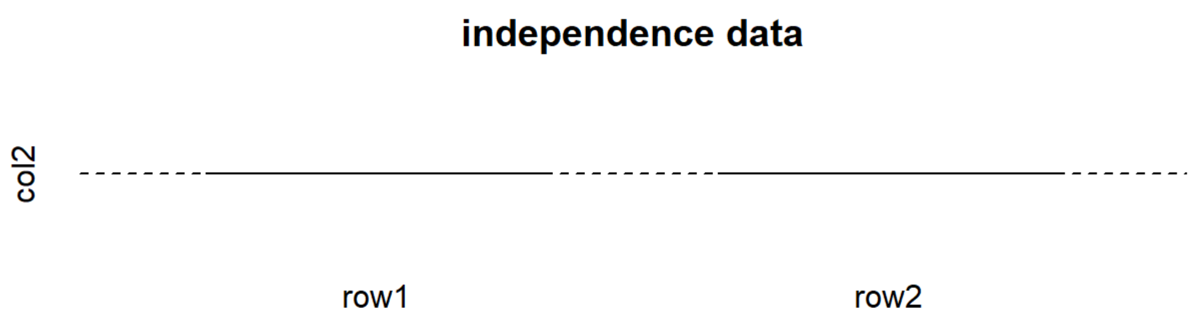
```
y <- matrix(c(120, 120 ,120, 121), ncol = 2, dimnames = list(c("row1", "row2"), c("col1", "col2")))
y
```

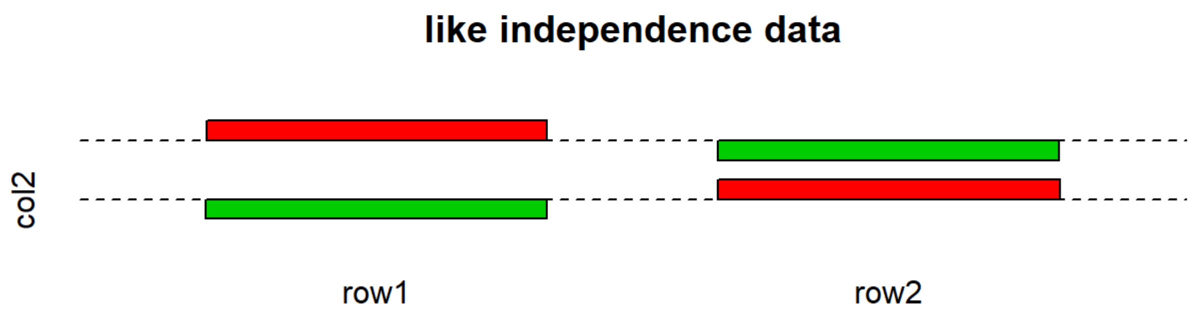
```
##      col1 col2
## row1  120  120
## row2  120  121
```

```
par(mfrow = c(2, 1))
```



```
assocplot(x, col = 2:3, space = 0.5)
title(main = "independence data")
assocplot(y, col = 2:3, space = 0.5)
title(main = "like independence data")
```





```
chisq.test(x)
```

```
##
##  Pearson's Chi-squared test
##
## data:  x
## X-squared = 0, df = 1, p-value = 1
```

```
chisq.test(y)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  y
## X-squared = 0, df = 1, p-value = 1
```

```
par(op)


# 3.2.13 fourfolodplot() 함수

admis <- aperm(UCBAdmissions, c(2, 1, 3))
dimnames(admis)[[2]] <- c("Yes", "No")
names(dimnames(admis)) <- c("Sex", "Admit?", "Department")
ftable(admis)
```

```
##                 Department   A   B   C   D   E   F
## Sex    Admit?
## Male   Yes                 512 353 120 138  53  22
##        No                  313 207 205 279 138 351
## Female Yes                  89  17 202 131  94  24
##        No                   19   8 391 244 299 317
```

```
# (1) 성별 합격여부의 데이터
admis.sex <- margin.table(admis, c(1, 2))
admis.sex
```

```
##           Admit?
## Sex         Yes   No
##   Male     1198 1493
##   Female    557 1278
```

```
# (2) 성별 합격/불합격 비율
prop.table(admis.sex, 1)
```

```
##           Admit?
## Sex              Yes        No
##   Male     0.4451877 0.5548123
##   Female   0.3035422 0.6964578
```

```
# (3)
fourfoldplot(admis.sex)
# (4)
fourfoldplot(admis)
# (5)
fourfoldplot(admis, margin = 2)



prop.table(admis[,,1], 1)
```

```
##           Admit?
## Sex              Yes        No
##   Male     0.6206061 0.3793939
##   Female   0.8240741 0.1759259
```

```
prop.table(admis[,,2], 1)
```

```
##           Admit?
## Sex              Yes        No
##   Male     0.6303571 0.3696429
##   Female   0.6800000 0.3200000
```

```
prop.table(admis[,,3], 1)
```

```
##           Admit?
## Sex              Yes        No
##   Male   0.3692308 0.6307692
##   Female 0.3406408 0.6593592
```

```
prop.table(admis[,,4], 1)
```

```
##         Admit?
## Sex            Yes        No
##   Male   0.3309353 0.6690647
##   Female 0.3493333 0.6506667
```

```
prop.table(admis[,,5], 1)
```

```
##         Admit?
## Sex            Yes        No
##   Male   0.2774869 0.7225131
##   Female 0.2391858 0.7608142
```

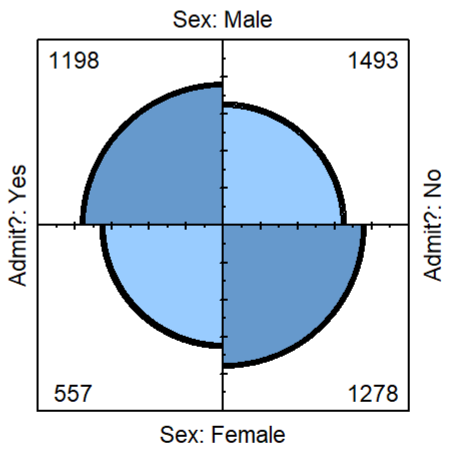
```
prop.table(admis[,,6], 1)
```

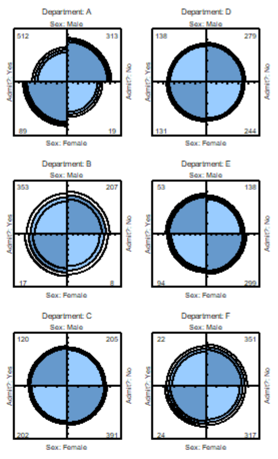
```
##         Admit?
## Sex             Yes         No
##   Male   0.05898123 0.94101877
##   Female 0.07038123 0.92961877
```

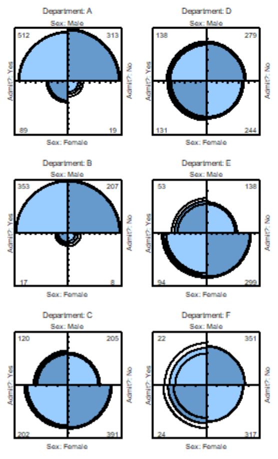
```
# 학부별 합격률
round(prop.table(margin.table(admis, c(2, 3)), 2) * 100, 2)
```

```
##       Department
## Admit?     A     B     C     D     E     F
##    Yes 64.42 63.25 35.08 33.96 25.17  6.44
##    No  35.58 36.75 64.92 66.04 74.83 93.56
```

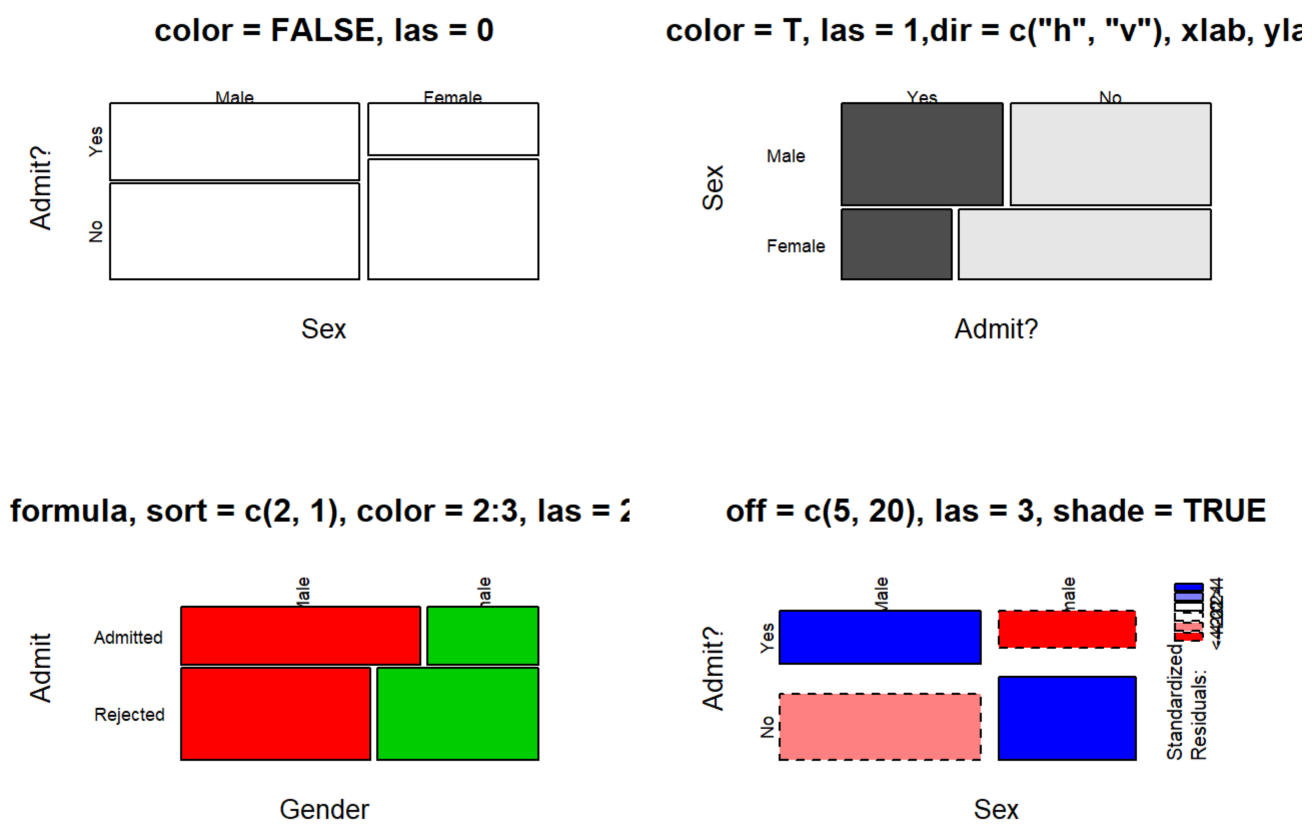
```
# 3.2.14 mosaiocplot() 함수

par(mfrow = c(2, 2))
```



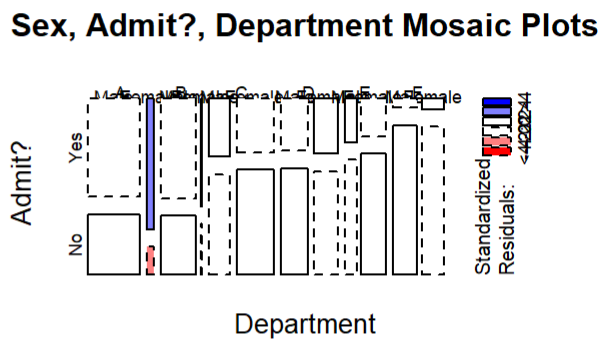


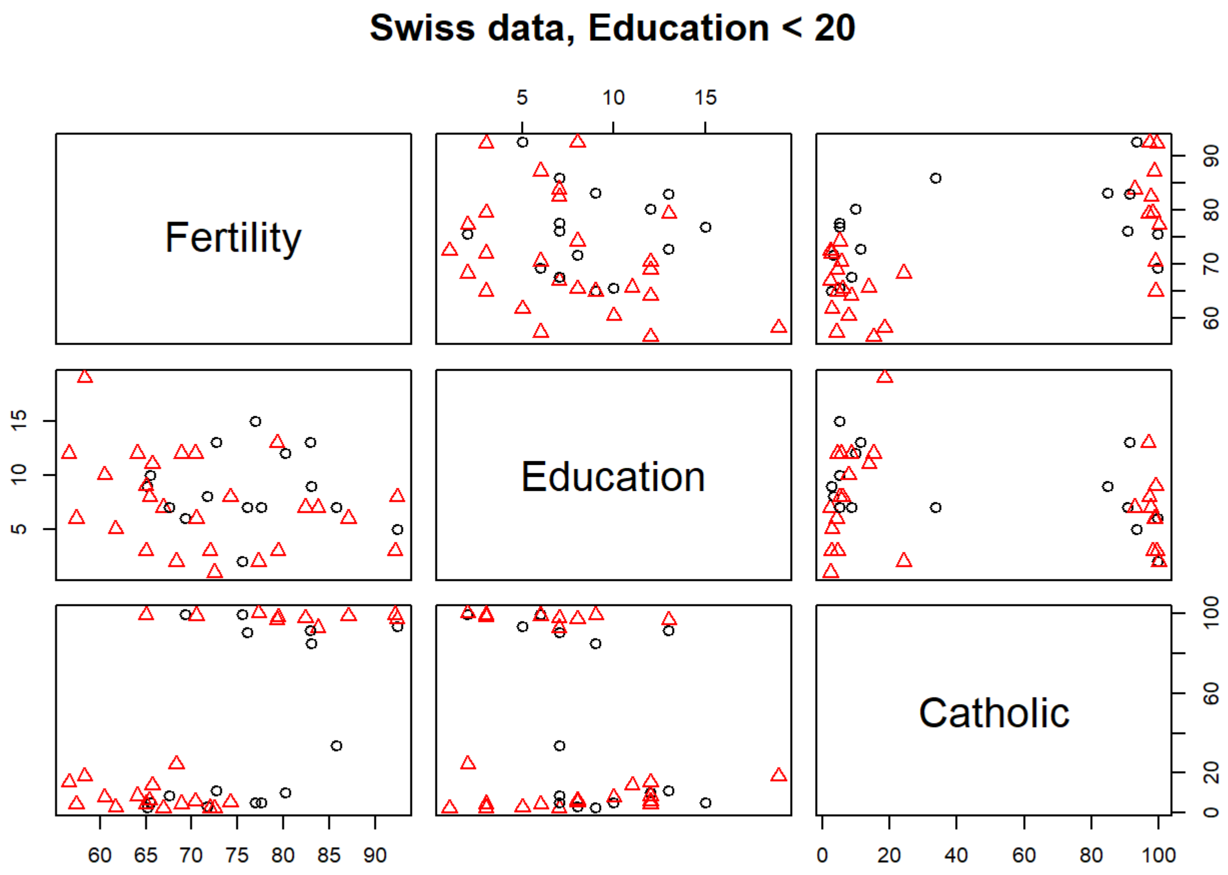


```
mosaicplot(admis.sex, color = FALSE, las = 0, main = "color = FALSE, las = 0")
mosaicplot(admis.sex, color = TRUE, las = 1, dir = c("h", "v"),
        xlab = "Admit?", ylab = "Sex",
        main = "color = T, las = 1,dir = c(\"h\", \"v\"), xlab, ylab")
mosaicplot(~ Gender + Admit, data = UCBAdmissions, sort = c(2, 1),
        color = 2:3, las = 2,
        main = "formula, sort = c(2, 1), color = 2:3, las = 2")
mosaicplot(admis.sex, off = c(5, 20), las = 3, shade = TRUE,
         main = "off = c(5, 20), las = 3, shade = TRUE")
```

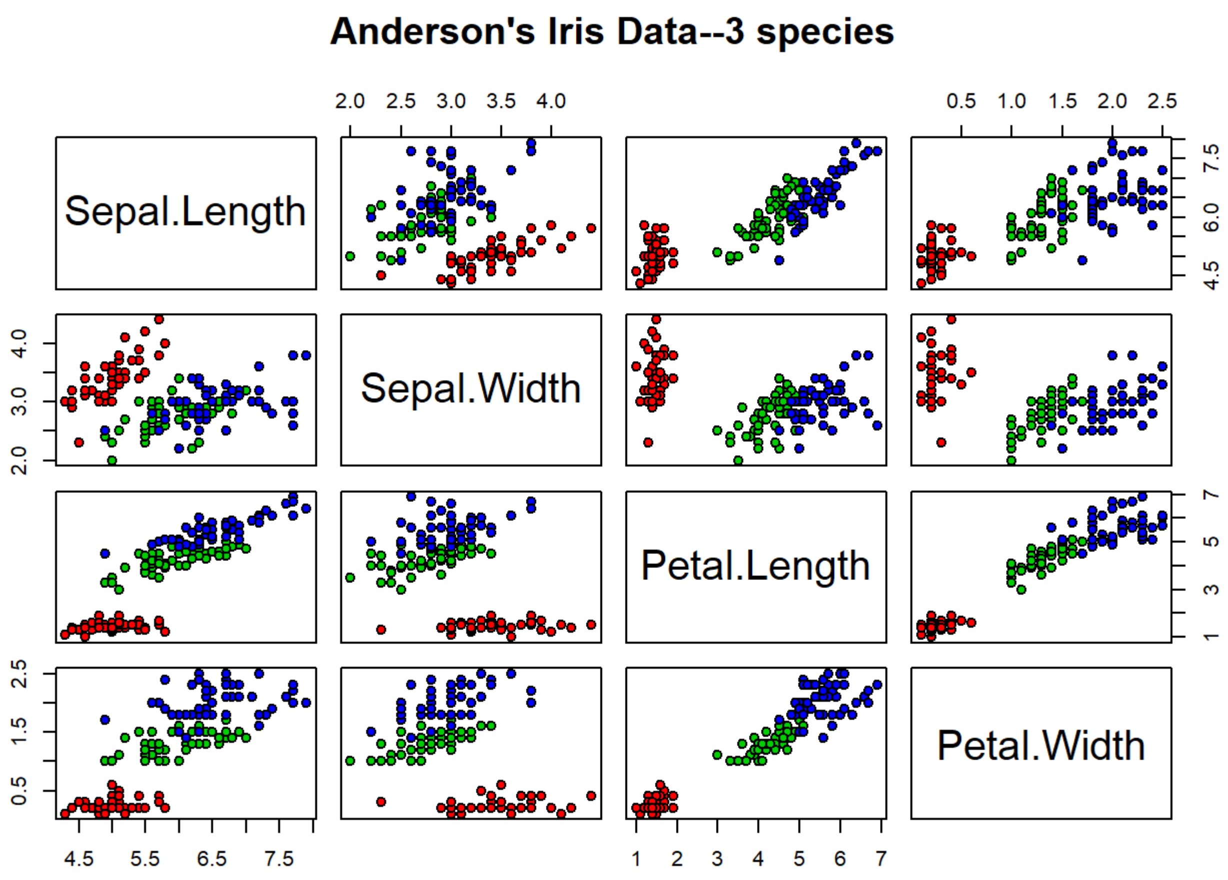


```
par(op)

mosaicplot(admis, sort = c(3, 1, 2), shade = T, margin = list(c(1, 3), c(2, 3)),
    xlab = "Department", main = "Sex, Admit?, Department Mosaic Plots")

# 3.2.15 paris() 함수
pairs(~ Fertility + Education + Catholic, data = swiss,
    subset = Education < 20, main = "Swiss data, Education < 20",
    col = 1 + (swiss$Agriculture > 50), cex = 1.2,
    pch = 1 + (swiss$Agriculture > 50))
```

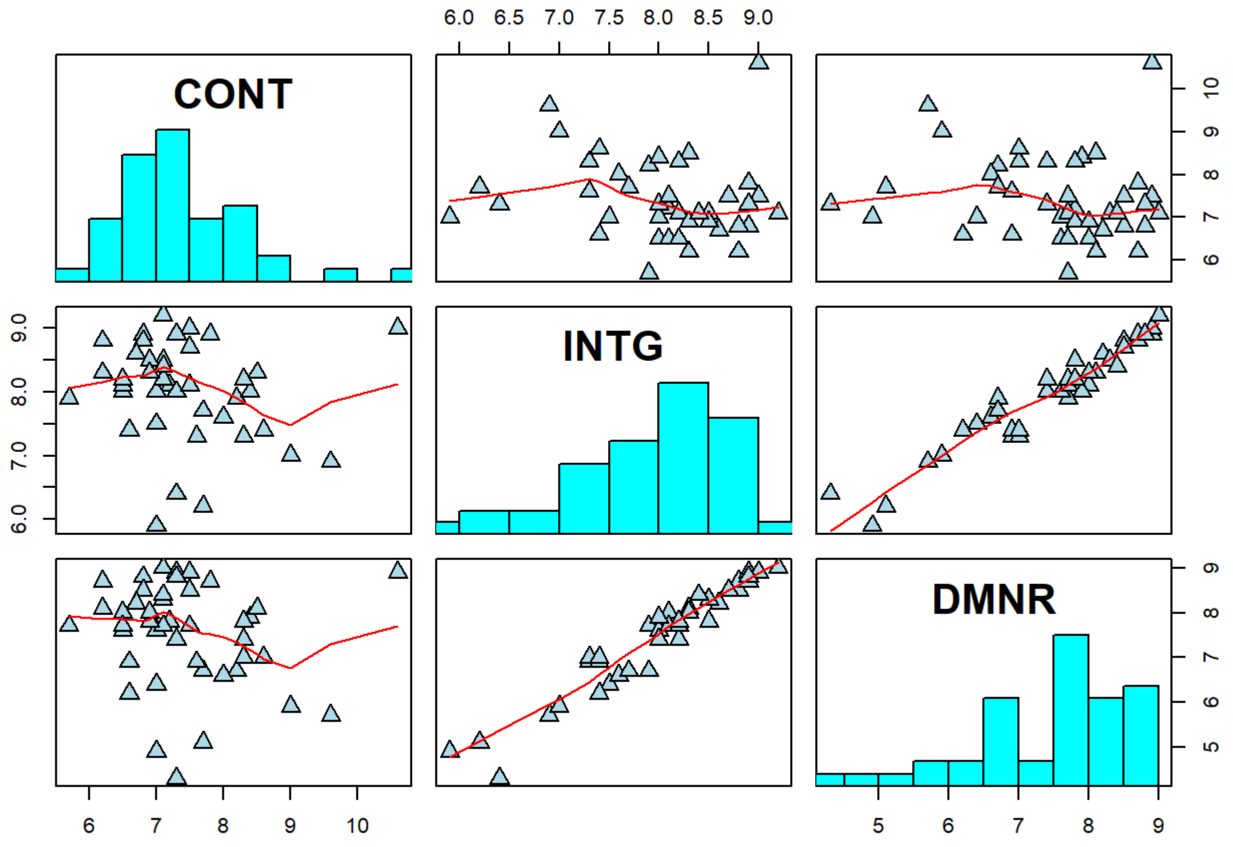




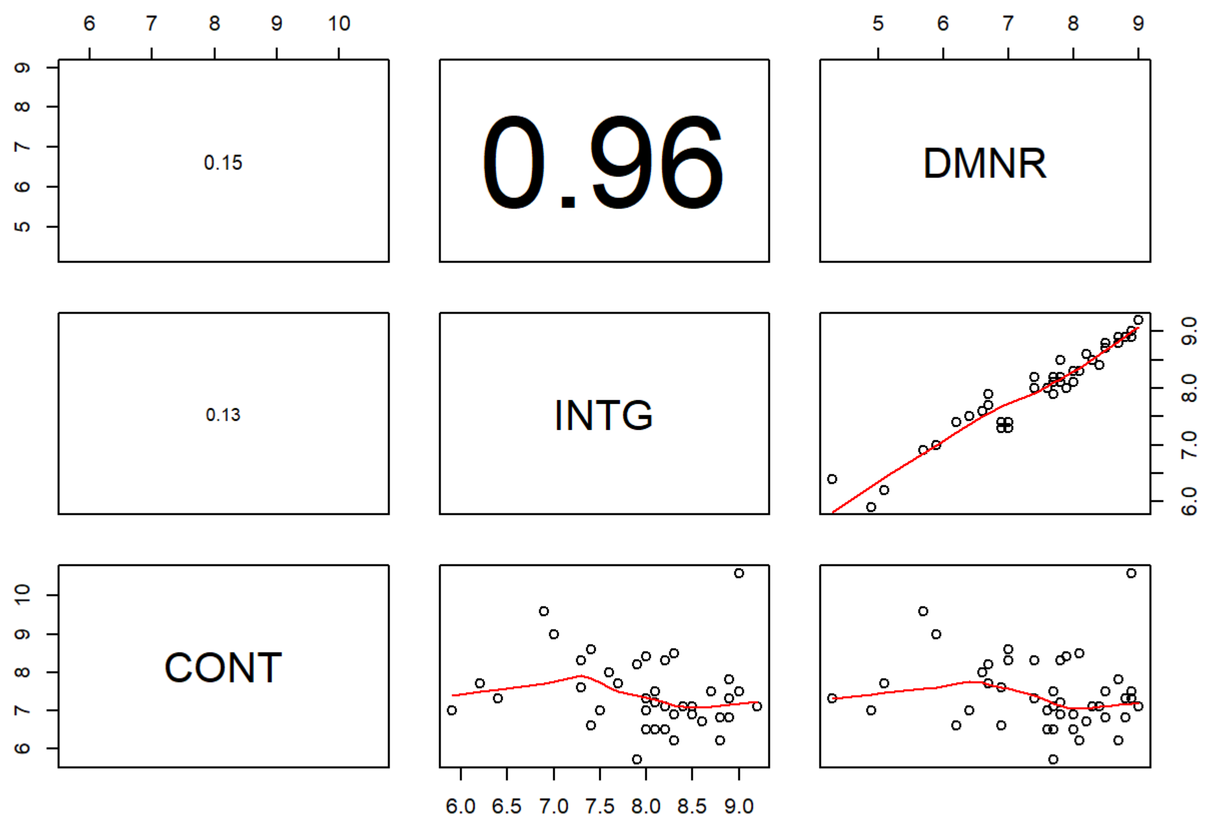
```
pairs(iris[1:4], main = "Anderson's Iris Data--3 species",
   pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)])
```



```
# 대각 패널 함수의 정의
panel.hist <- function(x, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks; nB <- length(breaks)
    y <- h$counts; y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col = "cyan", ...)
}
pairs(USJudgeRatings[1:3], panel = panel.smooth,
    cex = 1.5, pch = 24, bg = "light blue",
    diag.panel = panel.hist, cex.labels = 2, font.labels = 2)
```



```
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor)
{
   usr <- par("usr"); on.exit(par(usr))
   par(usr = c(0, 1, 0, 1))
   r <- abs(cor(x, y))
   txt <- format(c(r, 0.123456789), digits = digits)[1]
   txt <- paste(prefix, txt, sep = "")

   if(missing(cex.cor)) cex <- 0.8 / strwidth(txt)
   text(0.5, 0.5, txt, cex = cex * r)
}
pairs(USJudgeRatings[1:3], row1attop = FALSE, gap = 2,
   lower.panel = panel.smooth, upper.panel = panel.cor )
```



```
# 3.2.16 coplot() 함수

length(quakes$depth)
```

```
## [1] 1000
```

```
summary(quakes$depth)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    40.0    99.0   247.0   311.4   543.0   680.0
```

```
inter <- co.intervals(quakes$depth, number = 4, overlap = 0.1)
inter
```

```
##       [,1]  [,2]
## [1,]  39.5 107.5
## [2,]  96.5 260.5
## [3,] 238.5 544.5
## [4,] 534.5 680.5
```

```
inter[, 2] - inter[, 1]     # 등 간격이 아님
```

```
## [1]  68 164 306 146
```

```
length.inter <- as.numeric(0)
for (i in 1:4)
  length.inter[i] <- length(quakes$depth[inter[i, 1] <= quakes$depth &
                              quakes$depth <= inter[i, 2]])
length.inter            # 도수의 분포가 균일하게 나눔
```

```
## [1] 272 272 271 270
```

```
sum(length.inter)          # 도수의 합이 1000을 넘음
```

```
## [1] 1085
```

```
dim(quakes)
```

```
## [1] 1000    5
```

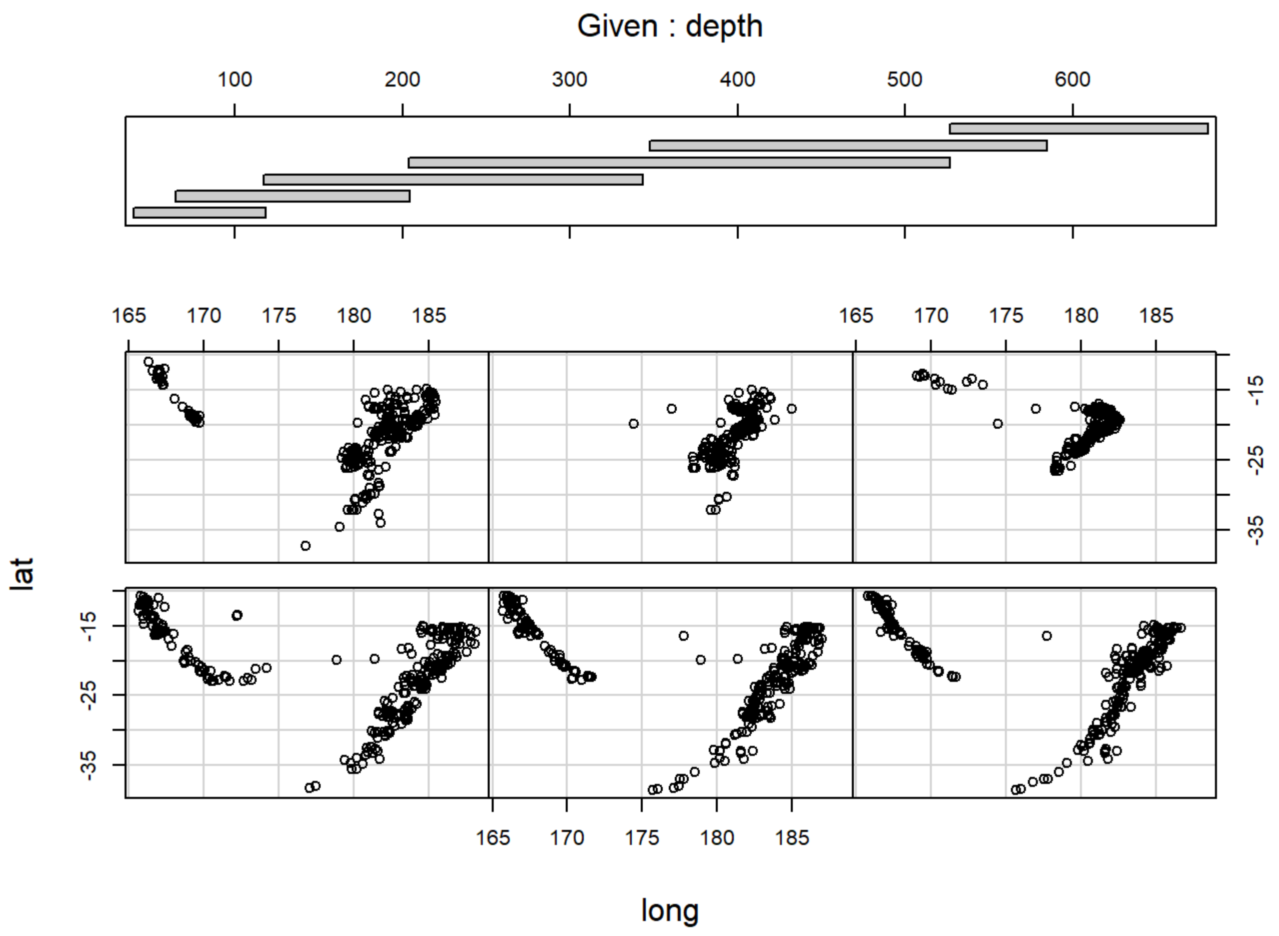
```
is.data.frame(quakes)
```

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

```
names(quakes)
```

```
## [1] "lat"      "long"     "depth"    "mag"      "stations"
```

```
coplot(lat ~ long | depth, data = quakes)
```



```
dim(iris)
```

```
## [1] 150   5
```

```
is.data.frame(iris)
```

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

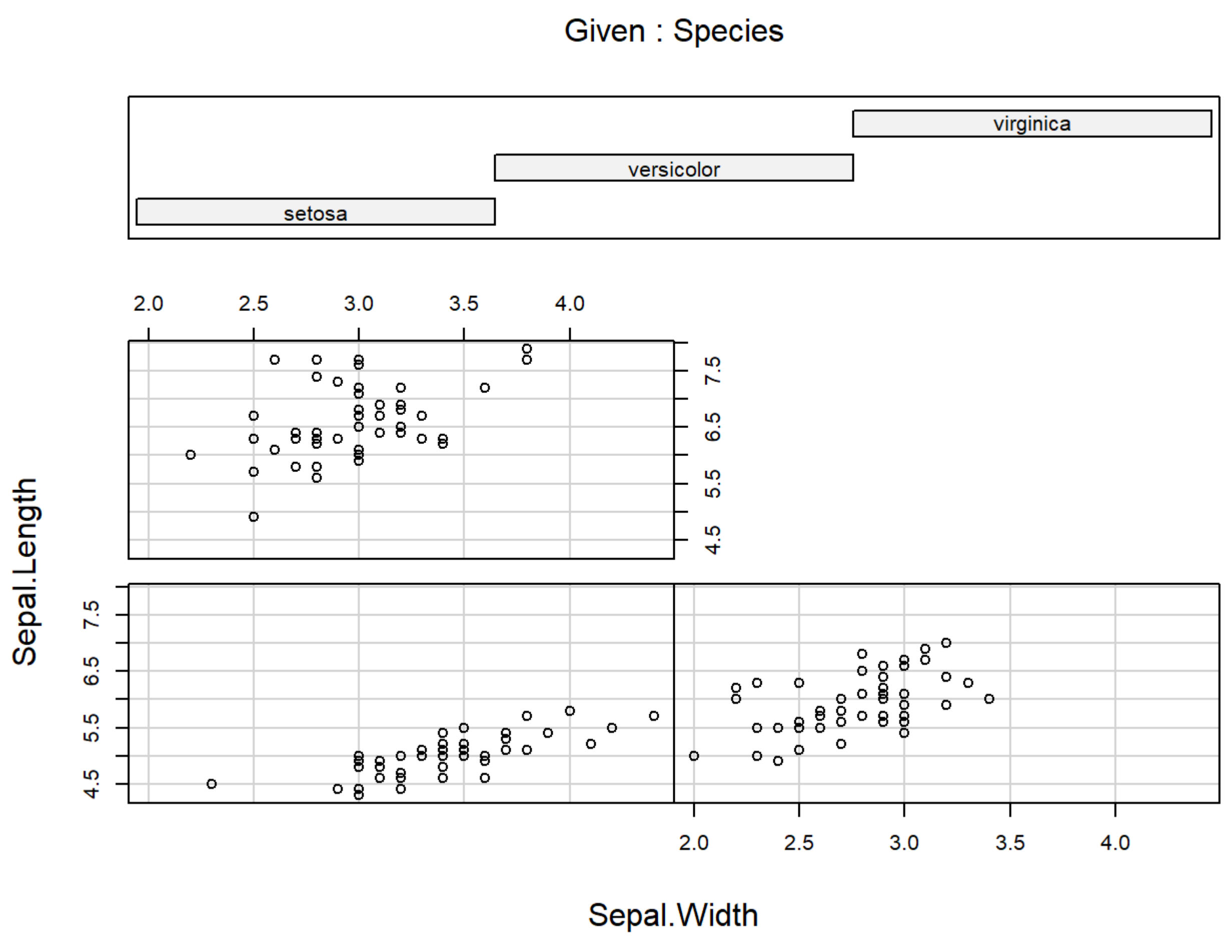
```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

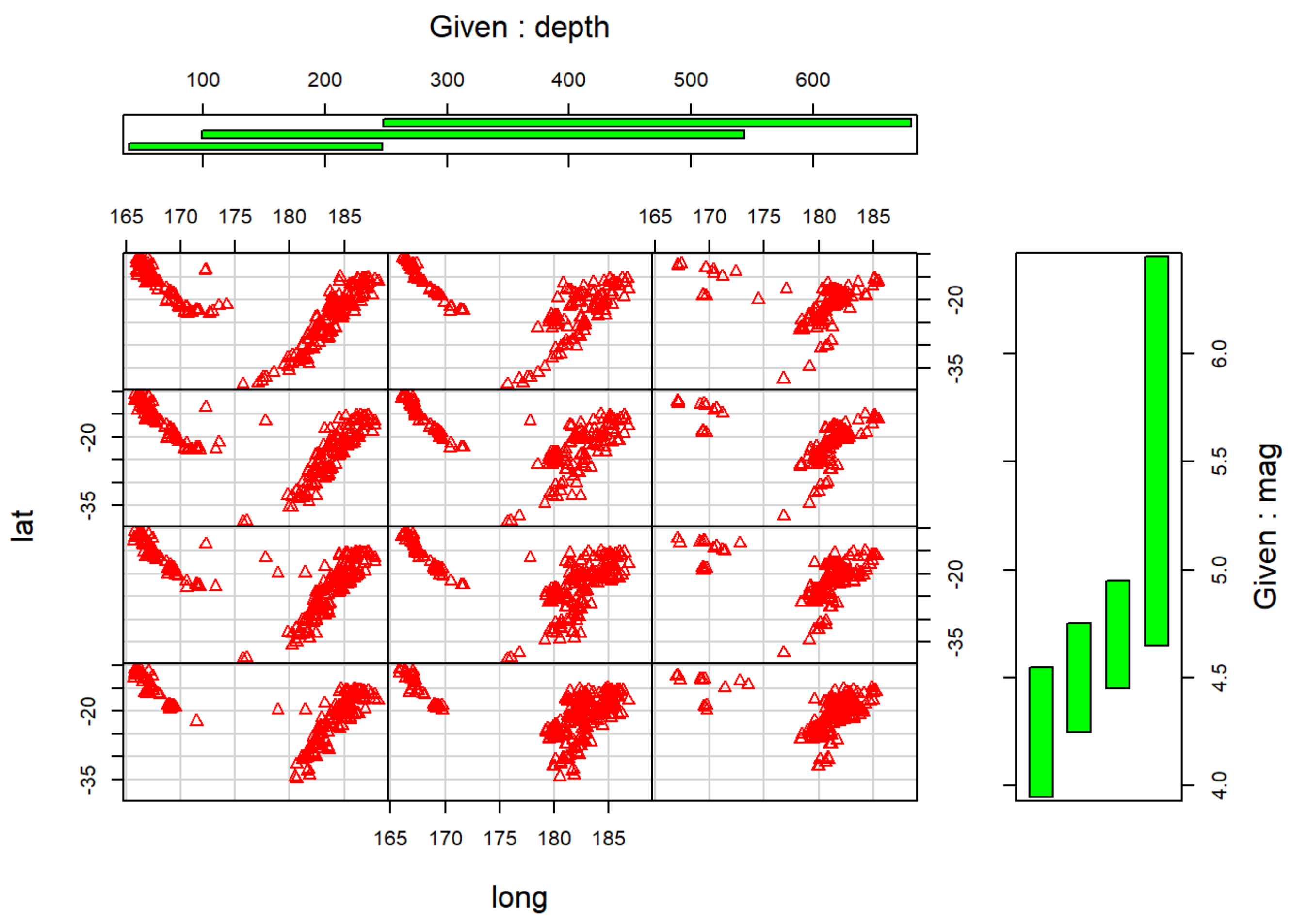
```
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

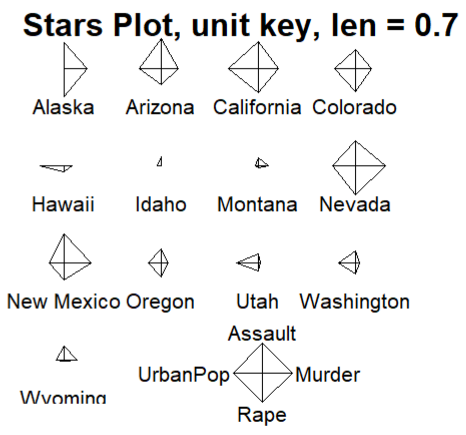
```
coplot(Sepal.Length ~ Sepal.Width | Species, data = iris)
```

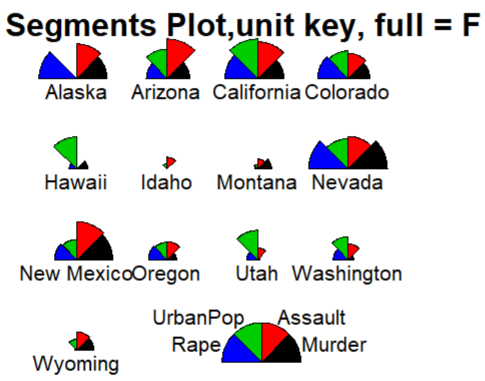


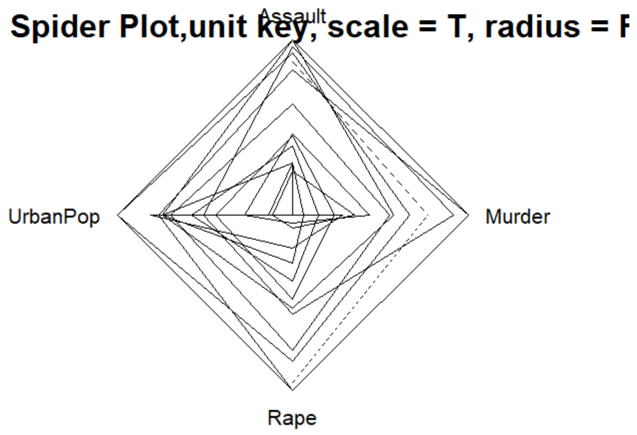
```
formulas <- lat ~ long | depth * mag
coplot(formulas, data = quakes, col = "red", pch = 2,
       number = c(3, 4), bar.bg = c(num = "green", fac = "blue"))
```

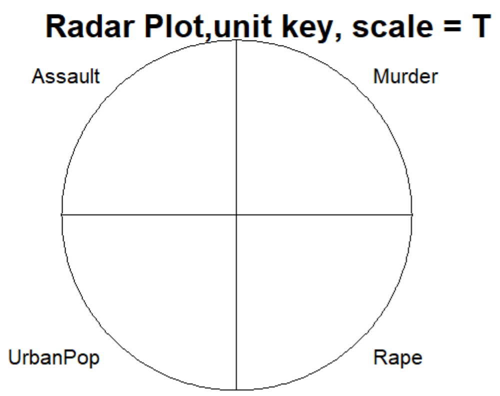


```
# 3.2.17 satrs() 함수

op <- par(no.readonly = TRUE)
WESTarrests <- USArrests[state.region == "West",]                    # (1)
par(mfrow = c(2, 2))
stars(WESTarrests, draw.segments = FALSE, len = 0.7, key.loc = c(7, 2))    # (2)
title(main = "Stars Plot, unit key, len = 0.7")
stars(WESTarrests, draw.segments = TRUE, full = FALSE, key.loc = c(7, 2))   # (3)
title(main = "Segments Plot,unit key, full = F")
stars(WESTarrests, locations = c(0, 0), scale = TRUE, radius = FALSE,
      col.stars = 0, key.loc = c(0, 0))                               # (4)
title(main = "Spider Plot,unit key, scale = T, radius = F")
stars(WESTarrests, locations = 0:1, scale = TRUE, draw.segments = TRUE,
      col.segments = 0, col.stars = 0, key.loc = 0:1)                   # (5)
title(main = "Radar Plot,unit key, scale = T")
```









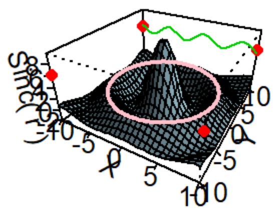
```
par(op)


# 3.2.18 persp() 함수

# (1) sinc 함수를 정의함
x <- seq(-10, 10, length = 30)
y <- x
f <- function(x, y) { r <- sqrt(x ^ 2 + y ^ 2); 10 * sin(r) / r }
z <- outer(x, y, f)
z[is.na(z)] <- 1       # 결측치의 값을 1로 바꾼다.

# (2) sinc 함수를 투시도로 그림
persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue",
      ltheta = 120, shade = 0.75, ticktype = "detailed",
      xlab = "X", ylab = "Y", zlab = "Sinc( r )") -> res
title(main="Perspective Plots with Sinc Function")
round(res, 3)      # persp() 함수의 반환 값
```

```
##        [,1]   [,2]   [,3]   [,4]
## [1,] 0.087 -0.025  0.043 -0.043
## [2,] 0.050  0.043 -0.075  0.075
## [3,] 0.000  0.074  0.042 -0.042
## [4,] 0.000 -0.273 -2.890  3.890
```

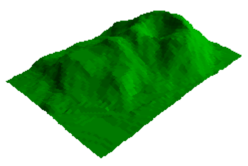
```
# (3) 3차원 좌표로 변환하는 함수
trans3d <- function(x, y, z, pmat) {
  tr <- cbind(x, y, z, 1) %*% pmat
```

```
    list(x = tr[,1] / tr[,4], y = tr[,2] / tr[,4])
}

xE <- c(-10,10); xy <- expand.grid(xE, xE)
points(trans3d(xy[,1], xy[,2], 6, pm = res), col = 2, pch = 16)
lines (trans3d(x, y =10, z = 6 + sin(x), pm = res), col = 3)

phi <- seq(0, 2 * pi, len = 201)
r1 <- 7.725
xr <- r1 * cos(phi)
yr <- r1 * sin(phi)
lines(trans3d(xr, yr, f(xr, yr), res), col = "pink", lwd = 2)



op <- par(no.readonly = TRUE)
z <- 2 * volcano        # Exaggerate the relief
x <- 10 * (1:nrow(z))   # 10 meter spacing (S to N)
y <- 10 * (1:ncol(z))   # 10 meter spacing (E to W)

par(bg = "lavender")
persp(x, y, z, theta = 135, phi = 30, col = "green3", scale = FALSE,
      ltheta = -120, shade = 0.75, border = NA, box = FALSE)
title("Perspective Plots with volcano")
par(op)
```

Perspective Plots with Sinc Function

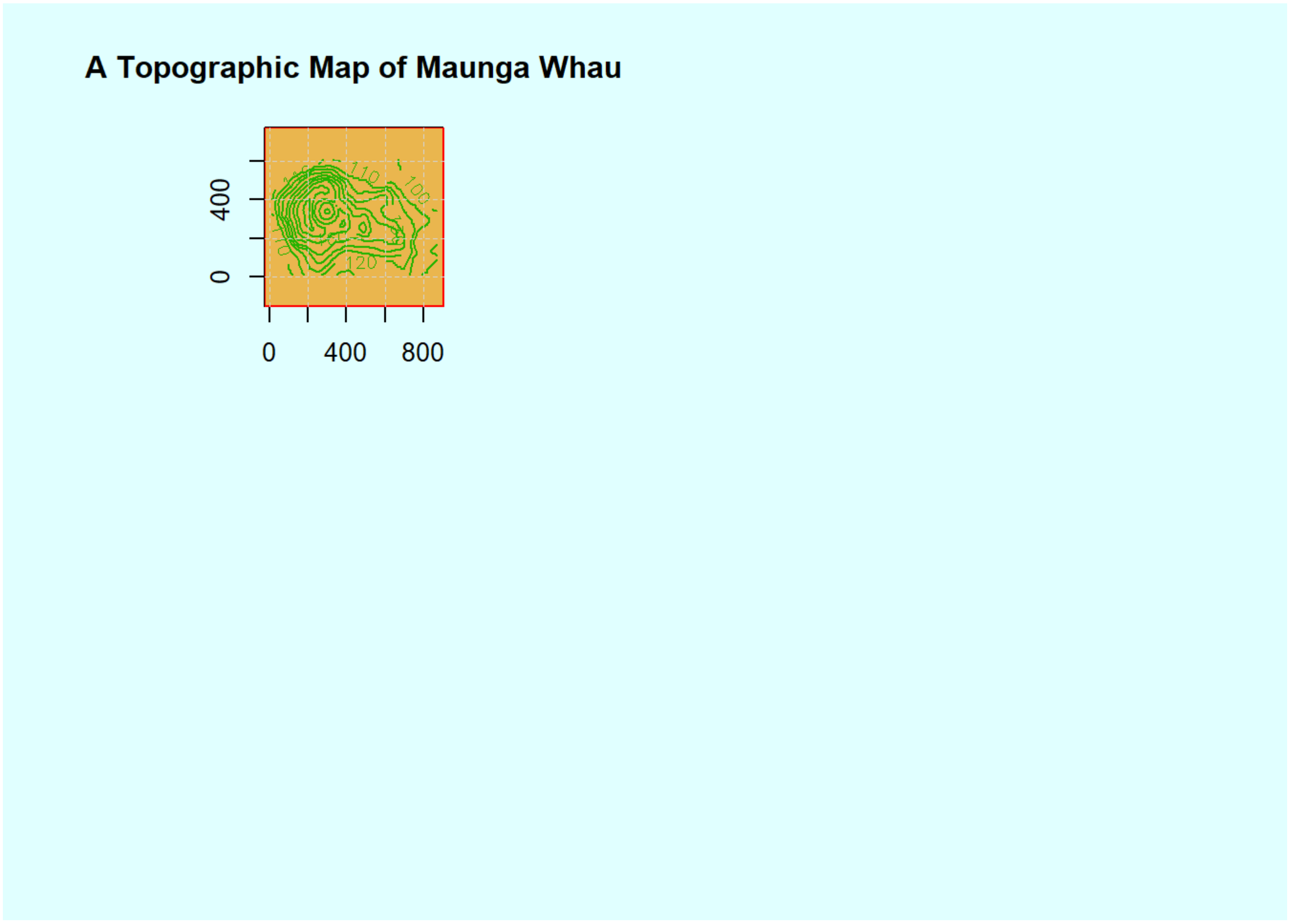


Perspective Plots with volcano



```
# 3.2.19 contour() 함수

op <- par(no.readonly = TRUE)
rx <- range(x <- 10 * 1:nrow(volcano))
```

```
ry <- range(y <- 10 * 1:ncol(volcano))
ry <- ry + c(-1, 1) * (diff(rx) - diff(ry)) / 2
tcol <- terrain.colors(12)
par(pty = "s", bg = "lightcyan")
plot(x = 0, y = 0, type = "n", xlim = rx, ylim = ry, xlab = "", ylab = "")
u <- par("usr")
rect(u[1], u[3], u[2], u[4], col = tcol[8], border = "red")
contour(x, y, volcano, col = tcol[2], lty = "solid", add = TRUE,
        vfont = c("sans serif", "plain"))
title("A Topographic Map of Maunga Whau", font = 4)
abline(h = 200 * 0:4, v = 200 * 0:4, col = "lightgray", lty = 2, lwd = 0.1)
par(op)
```



```
op <- par(no.readonly = TRUE)
line.list <- contourLines(x, y, volcano)  # (1) contourLines 호출
par(pty = "s", bg = "lightcyan")
plot(x = 0, y = 0, type = "n", xlim = rx, ylim = ry, xlab = "", ylab = "")
rect(u[1], u[3], u[2], u[4], col = tcol[8], border = "red")
is.list(line.list)                 # (2) 리스트 여부 확인
```

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

```
length(line.list)                  # (3) 성분의 개수
```

```
## [1] 20
```

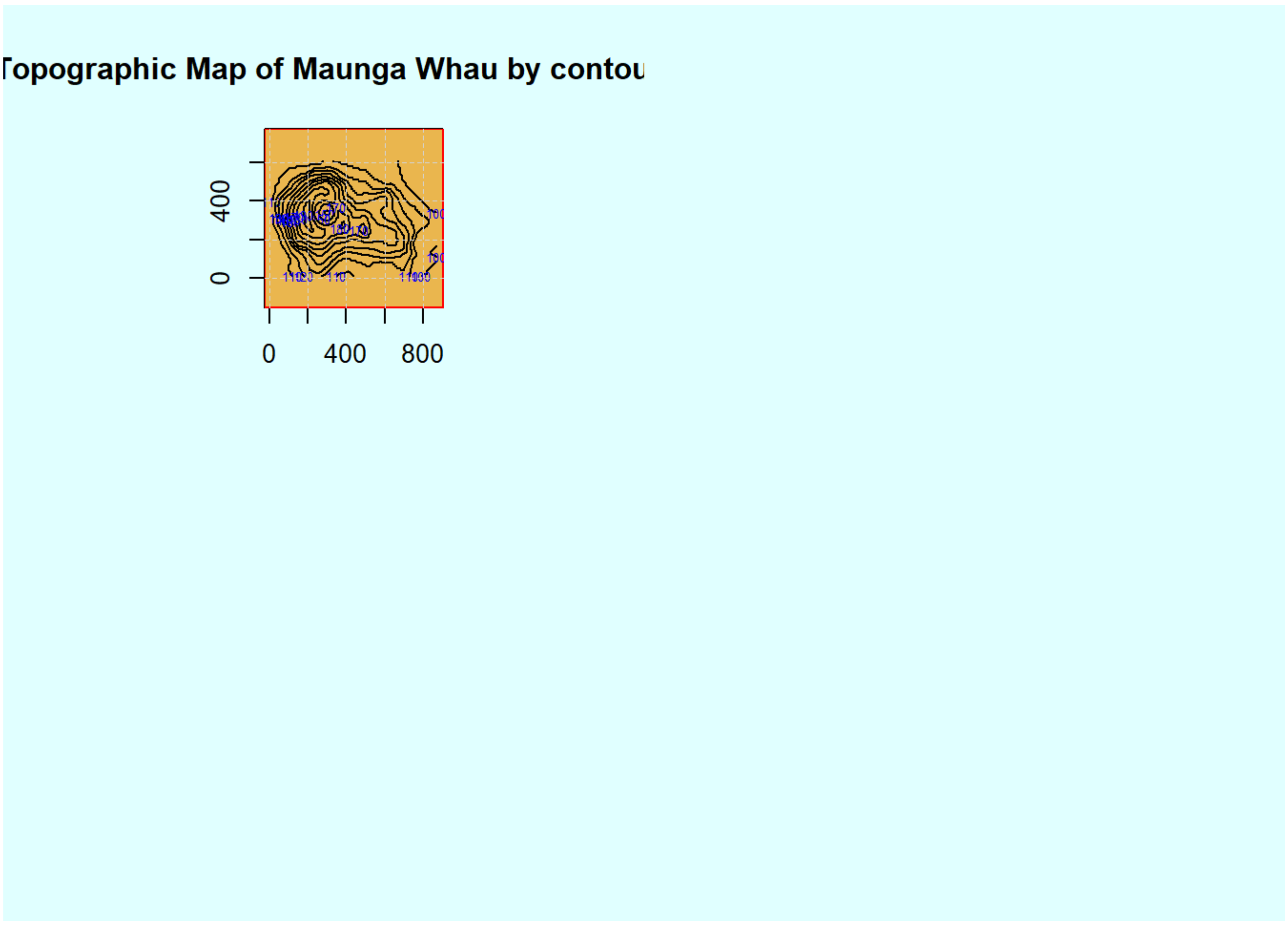
```
names(line.list[[1]])                  # (4) 성분의 이름
```

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

```
line.list[[1]]                    # (5) 첫 번째 성분 출력
```

```
## $level
## [1] 100
## 
## $x
##  [1] 870.0000 860.0000 850.9173 850.0000 840.0000 830.9173 830.0000
##  [8] 820.9173 820.0000 810.9173 810.0000 800.9173 800.0000 790.9173
## [15] 790.0000 780.9173 780.0000 770.4807 770.0000 760.3257 760.0000
## [22] 750.3257 750.0000 740.2463 740.0000 730.2463 730.0000 720.3257
## [29] 720.0000 710.4807 710.4807 710.0000 700.9173 700.0000 690.9173
## [36] 690.9173 690.9173 690.0000 680.9173 680.9173 680.9173 680.9173
## [43] 680.0000 670.9173 670.9173 670.9173 670.9173
## 
## $y
##  [1] 340.9173 340.9173 350.0000 350.9173 350.9173 360.0000 360.9173
##  [8] 370.0000 370.9173 380.0000 380.9173 390.0000 390.9173 400.0000
## [15] 400.9173 410.0000 410.4807 420.0000 420.3257 430.0000 430.3257
## [22] 440.0000 440.2463 450.0000 450.2463 460.0000 460.3257 470.0000
## [29] 470.4807 480.0000 490.0000 490.9173 500.0000 500.9173 510.0000
## [36] 520.0000 530.0000 530.9173 540.0000 550.0000 560.0000 570.0000
## [43] 570.9173 580.0000 590.0000 600.0000 610.0000
```

```
templines <- function(clines) {             # (6) 등고선과 라벨 출력 함수 정의
  lines(clines[[2]], clines[[3]])
  text(clines[[2]][1], clines[[3]][1], clines[[1]][1], cex = 0.5, col = "blue")
}
invisible(lapply(line.list, templines))       # (7) 등고선을 그린다.
title("A Topographic Map of Maunga Whau by contourLines", font=4)
abline(h = 200 * 0:4, v = 200*0:4, col = "lightgray", lty = 2, lwd = 0.1)
par(op)
```



```
# 3.2.20 image() 함수

image(volcano, zlim = c(150, 200), xaxs = "r", yaxs = "r",
      xlab = "West to East", ylab = "South to North")
image(volcano, zlim = c(0, 150), add = T, col = cm.colors(12),
      xlab = "0 to 1", ylab = "0 to 1")
title(main = "image & add image")


x <- 10 * (1:nrow(volcano))
y <- 10 * (1:ncol(volcano))
image(x, y, volcano, col = terrain.colors(100), axes = FALSE)
contour(x, y, volcano, levels = seq(90, 200, by = 5),
        add = TRUE, col = "peru")
axis(1, at = seq(100, 800, by = 100))
axis(2, at = seq(100, 600, by = 100))
box()
title(main = "Maunga Whau Volcano", font.main = 4)


# 3.2.21 filled.contour 함수()

x <- 10 * 1:nrow(volcano)
y <- 10 * 1:ncol(volcano)
filled.contour(x, y, volcano, color = terrain.colors,
               plot.title = title(main = "The Topography of Maunga Whau",
                                  xlab = "Meters North", ylab = "Meters West"),
               plot.axes = { axis(1, seq(100, 800, by = 100))
                             axis(2, seq(100, 600, by = 100)) },
               key.title = title(main = "Heightn(meters)"),
        key.axes = axis(4, seq(90, 190, by = 10)))
```

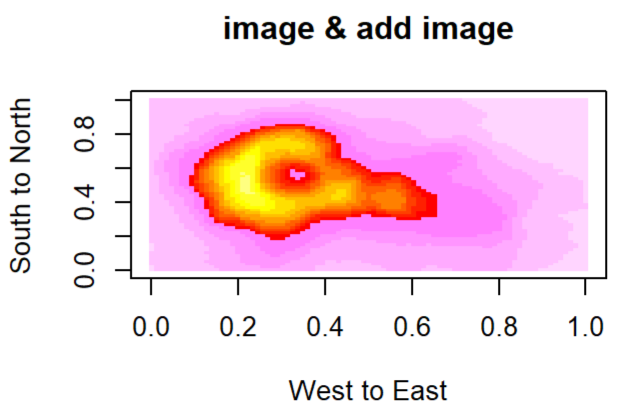
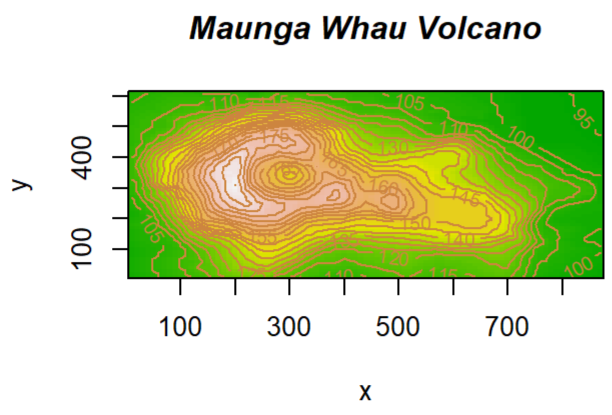
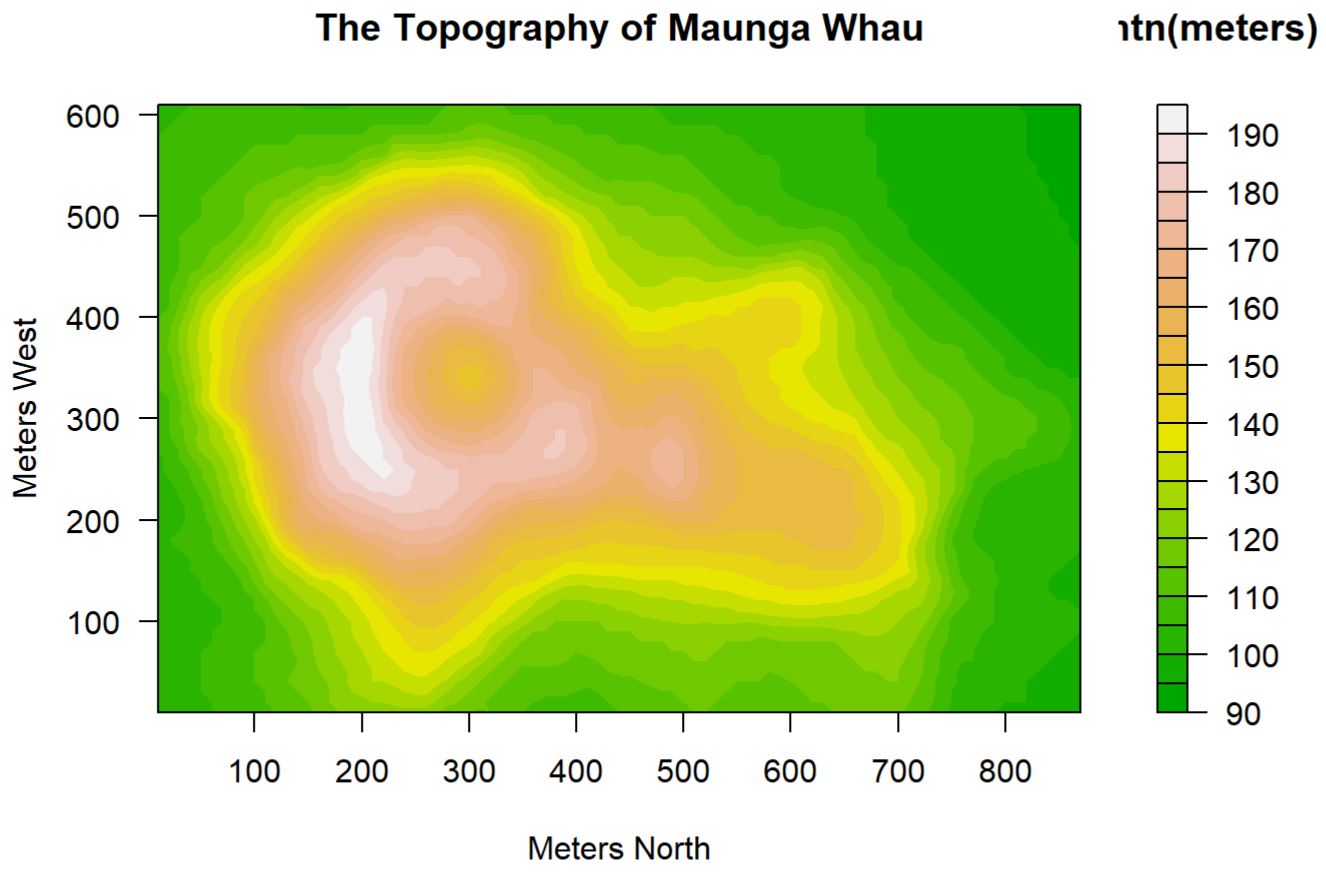


image & add image
South to North
West to East


Maunga Whau Volcano
y
x


The Topography of Maunga Whau
ıtn(meters)
Meters West
Meters North