2.R

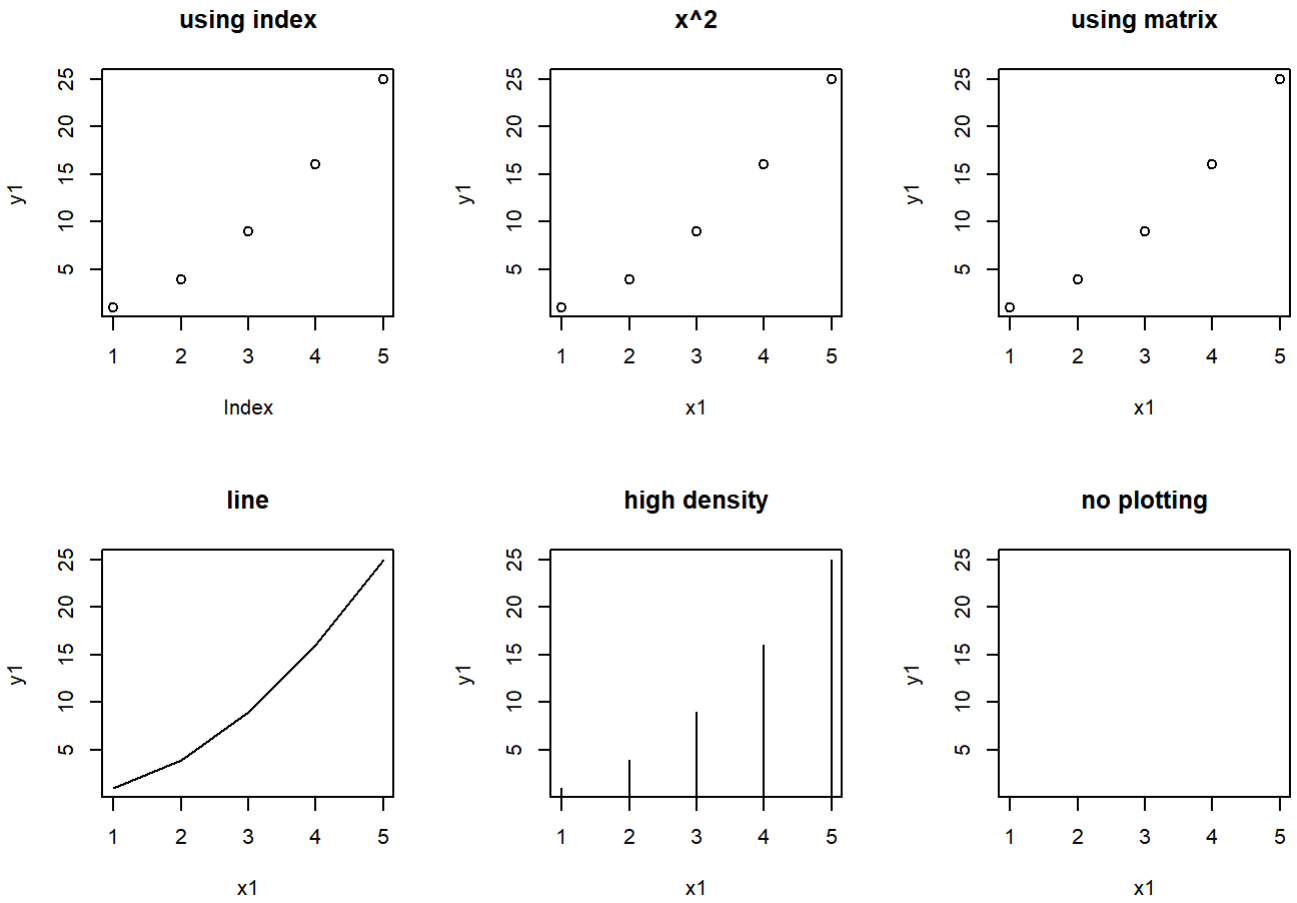
limdr

Sun Aug 26 22:39:11 2018

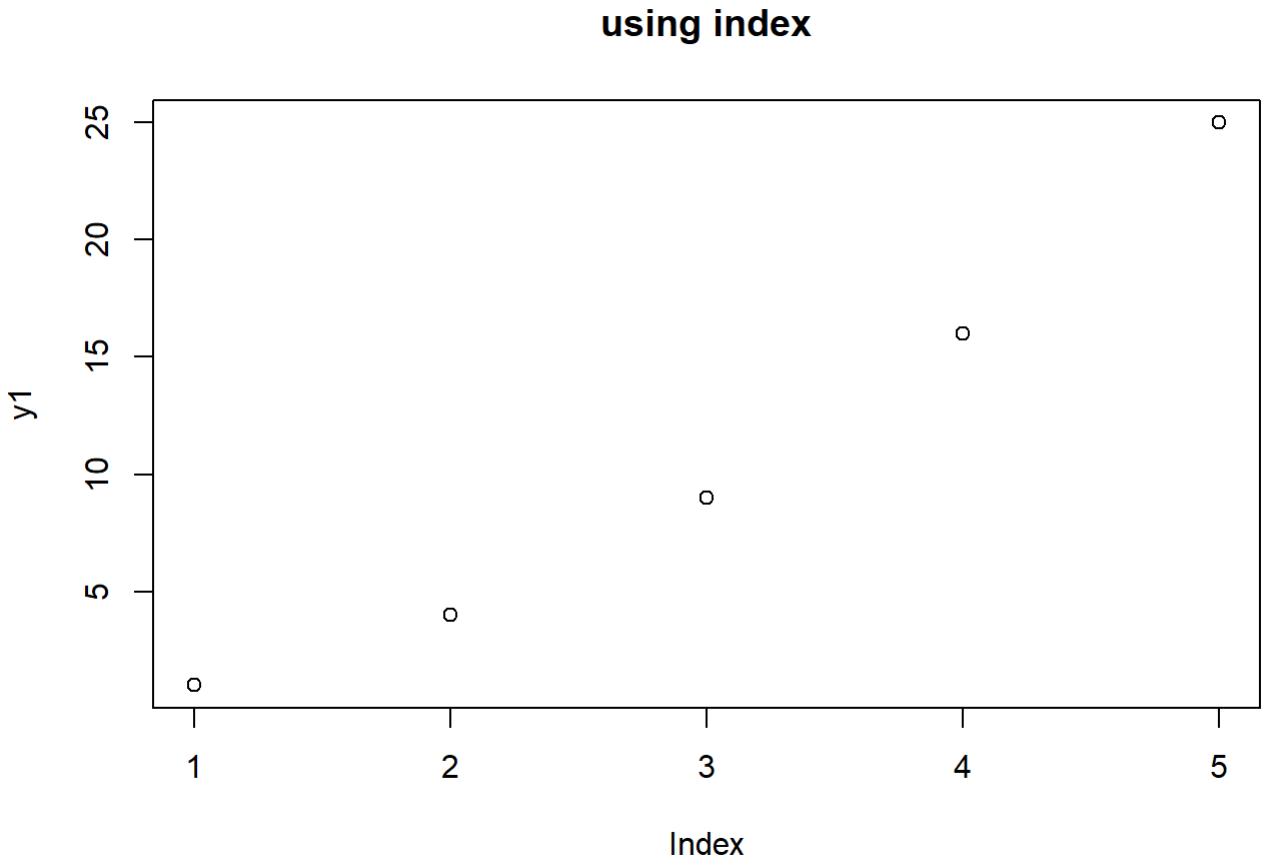
```
## 2장 R 그래픽스의 기초

# 2.2 고수준 그래픽 함수

# 2.2.1 plot() 함수

# x-좌표를 위한 벡터
x1 <- 1:5
# y-좌표를 위한 벡터
y1 <- x1^2
# 벡터 생성
z1 <- 5:1
# 행렬 생성
(mat1 <- cbind(x1, y1, z1))
```

```
##      x1 y1 z1
## [1,]  1  1  5
## [2,]  2  4  4
## [3,]  3  9  3
## [4,]  4 16  2
## [5,]  5 25  1
```

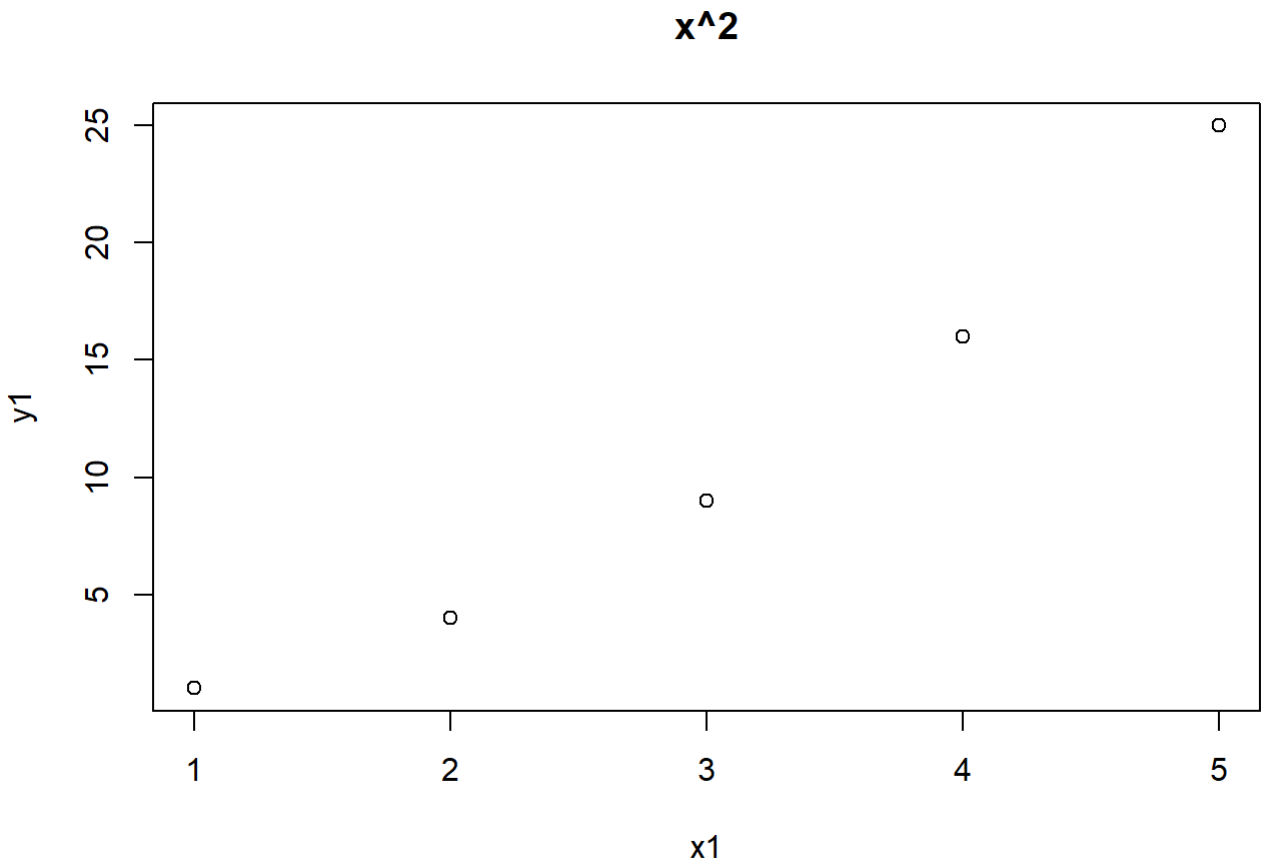
```
# 그래픽 윈도우의 화면 분할 (2행 3열)
op <- par(no.readonly = TRUE)


par(mfrow=c(2, 3)) #par함수를 이용하여 2행 3열 그림판 생성

# 일변량 그래프
plot(y1, main="using index")
# 이변량 그래프
plot(x=x1, y=y1, main="x^2")
# 이변량 그래프 (행렬)
plot(mat1, main="using matrix")
plot(x1, y1, type="l", main="line")
plot(x1, y1, type="h", main="high density")
plot(x1, y1, type="n", main="no plotting")
```



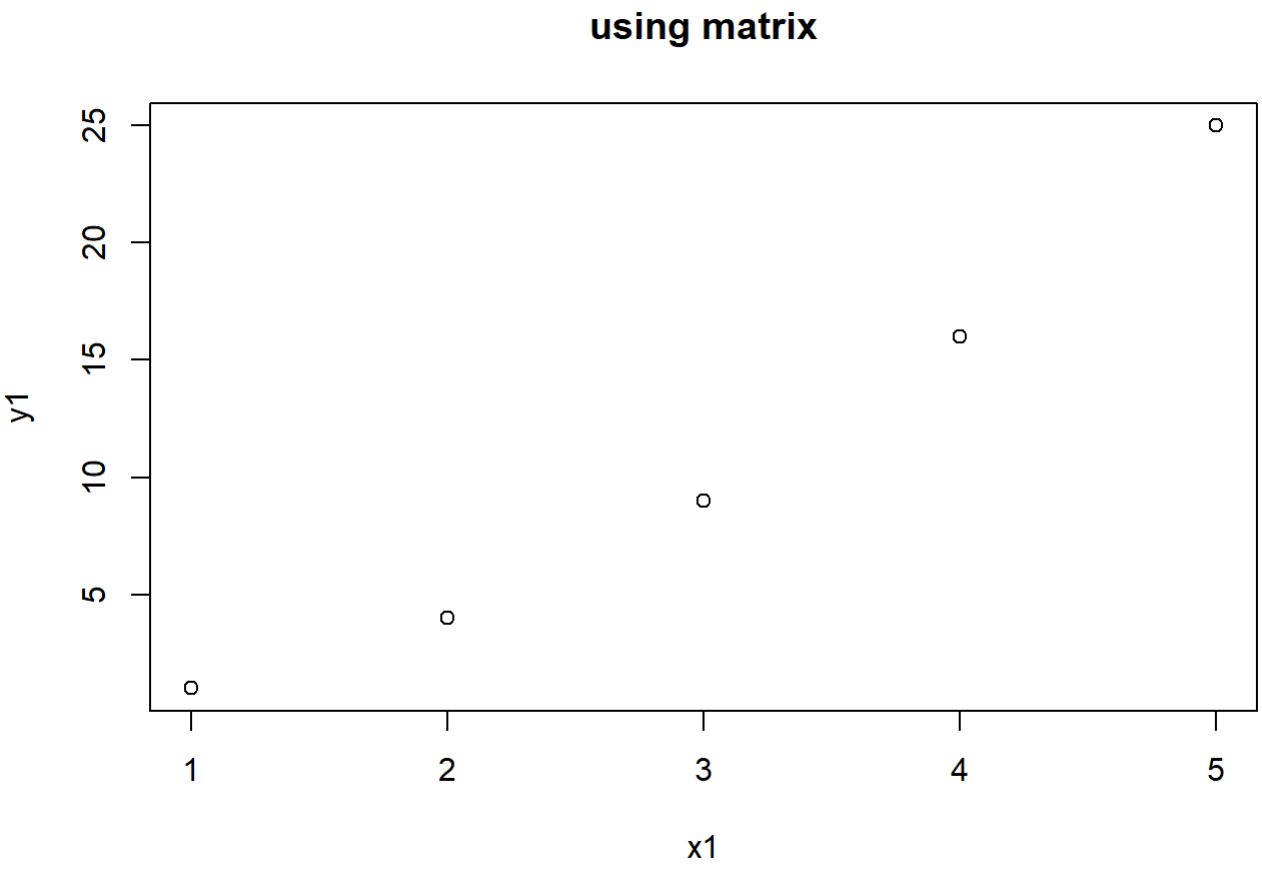

```
# 그래픽 윈도우의 화면 병합 (1행 1열)
par(op)
# 일변량 그래프
plot(y1, main="using index")
```



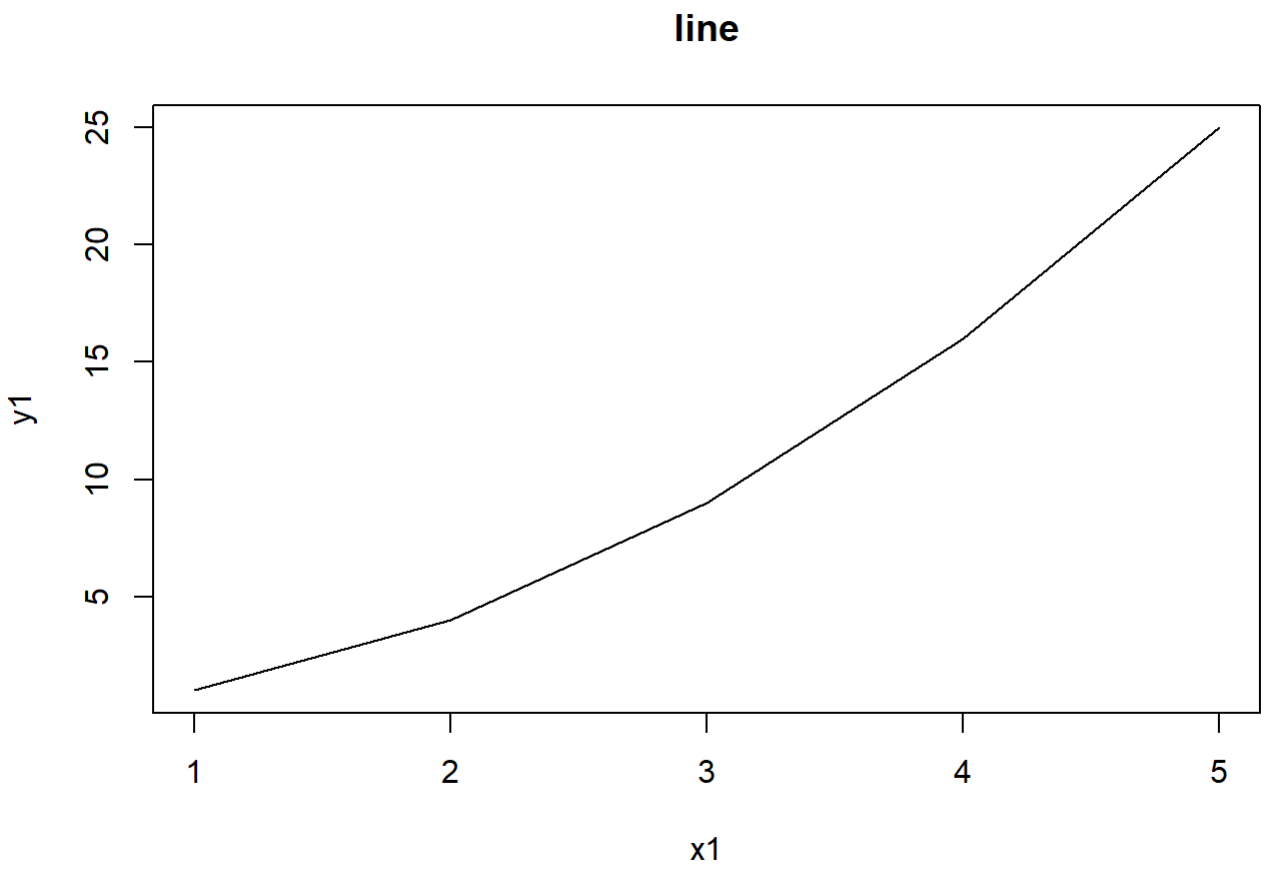

```
# 이변량 그래프
plot(x=x1, y=y1, main="x^2")
```



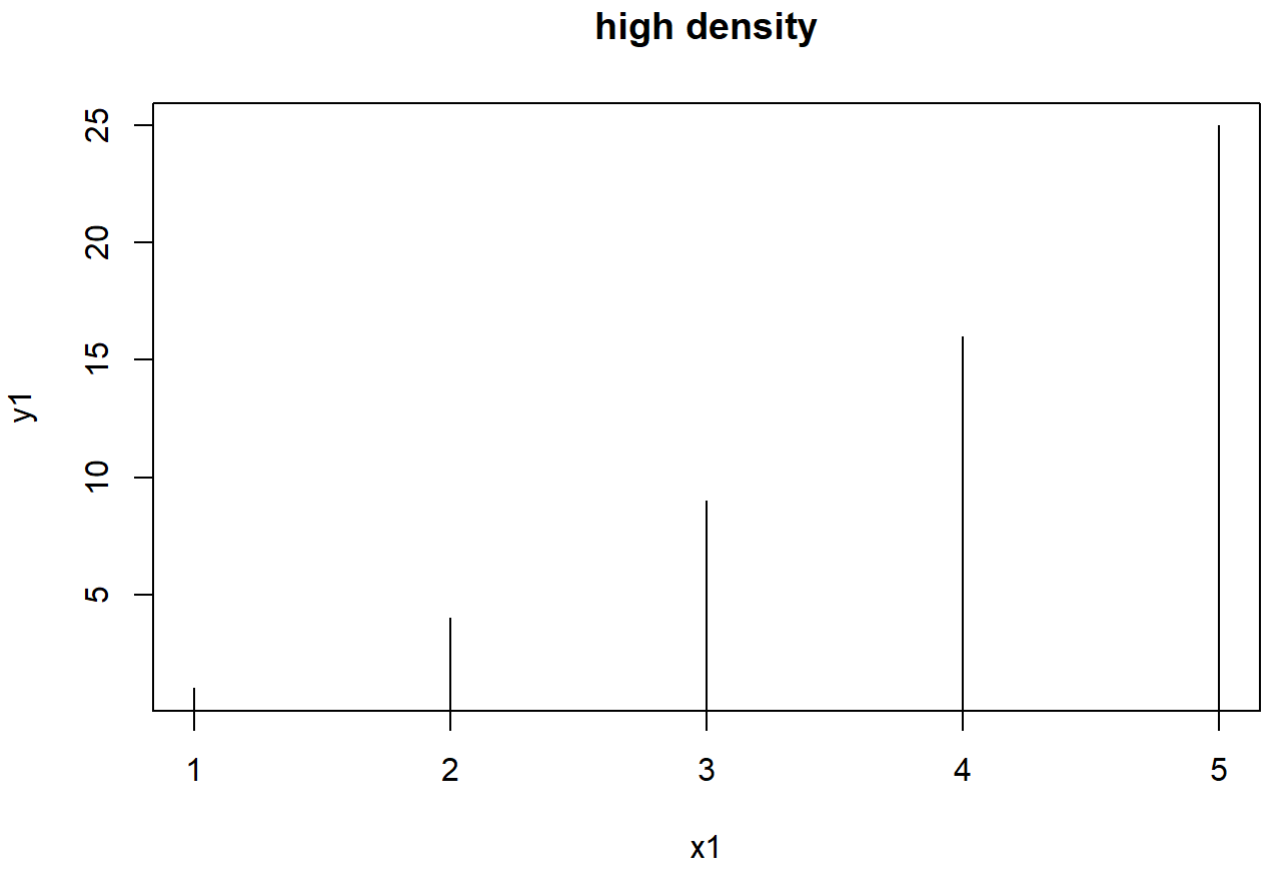
```
# 이변량 그래프 (행렬)
plot(mat1, main="using matrix")
```



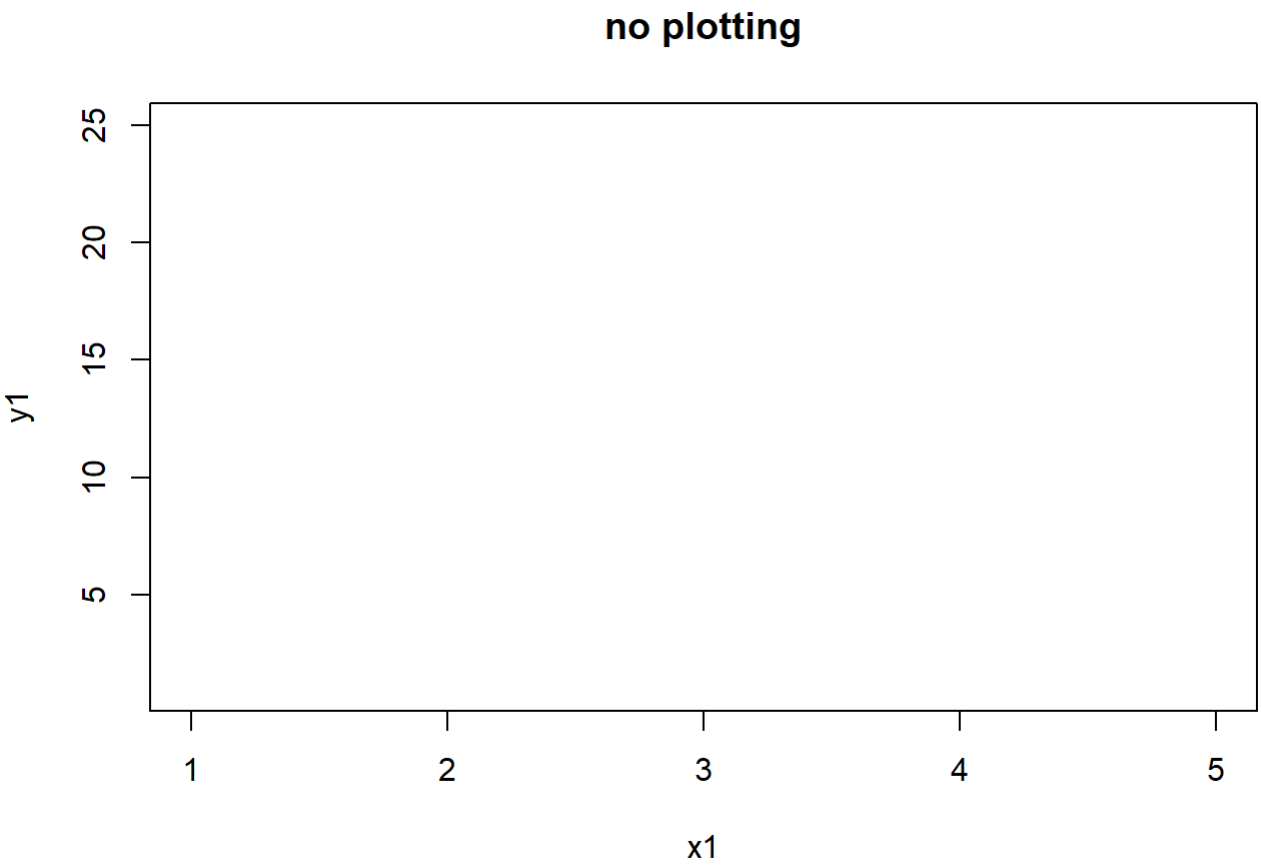

```
plot(x1, y1, type="l", main="line")
```




```
plot(x1, y1, type="h", main="high density")
```



```
plot(x1, y1, type="n", main="no plotting")
```




```
# 2.2 저수준 그래픽 함수

# 2.3.2 점을 찍는 함수

# 2.3.2.1 points() 함수
#       : 플롯 영역의 지정한 좌표 위에 점을 찍는다.

# pch 인수는 점의 모양을 지정.

x <- rep(1:5, rep(5, 5))
x
```

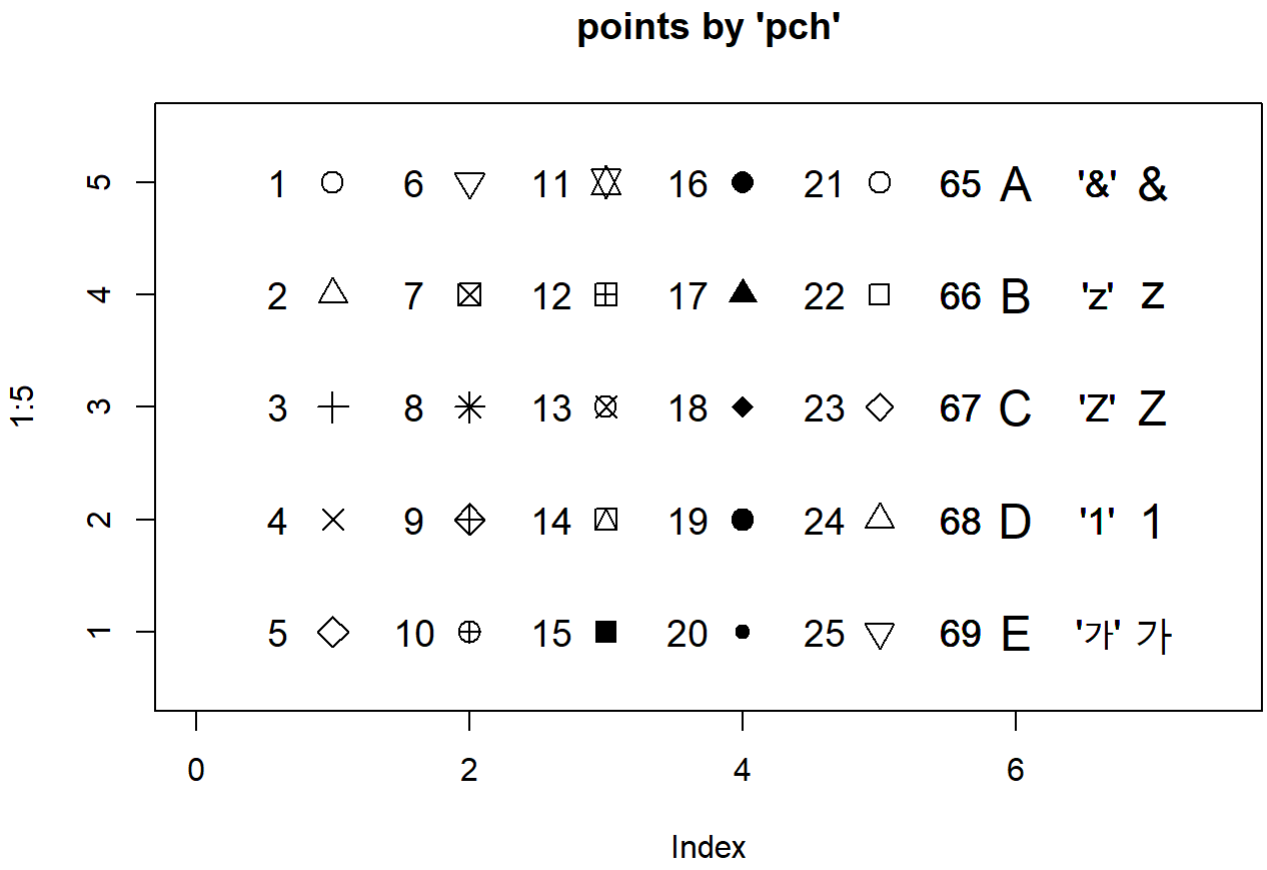
```
##  [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5
```

```
y <- rep(5:1, 5)
y
```

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

```
pchs <- c("&", "z", "Z", "1", "가")
plot(1:5, type = "n", xlim = c(0, 7.5), ylim = c(0.5, 5.5), main = "points by 'pch'")
points(x, y, pch = 1:25, cex = 1.5)
text(x - 0.4, y, labels = as.character(1:25), cex = 1.2)
points(rep(6, 5), 5:1, pch = 65:69, cex = 1.5)
text(rep(6, 5) - 0.4, y, labels = as.character(65:69), cex = 1.2)
```

```
points(rep(7, 5), 5:1, pch = pchs, cex = 1.5)
text(rep(7, 5) - 0.4, y, labels = paste("", pchs, "", sep = ""), cex = 1.2)
```




```
# 2.3.3 선을 그리는 함수
# 2.3.3.1 abline() 함수 : 직교좌표에 직선을 그리는 함수

cars[1:4,]
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
```

```
z <- lm(dist ~ speed, data = cars)
is(z)
```

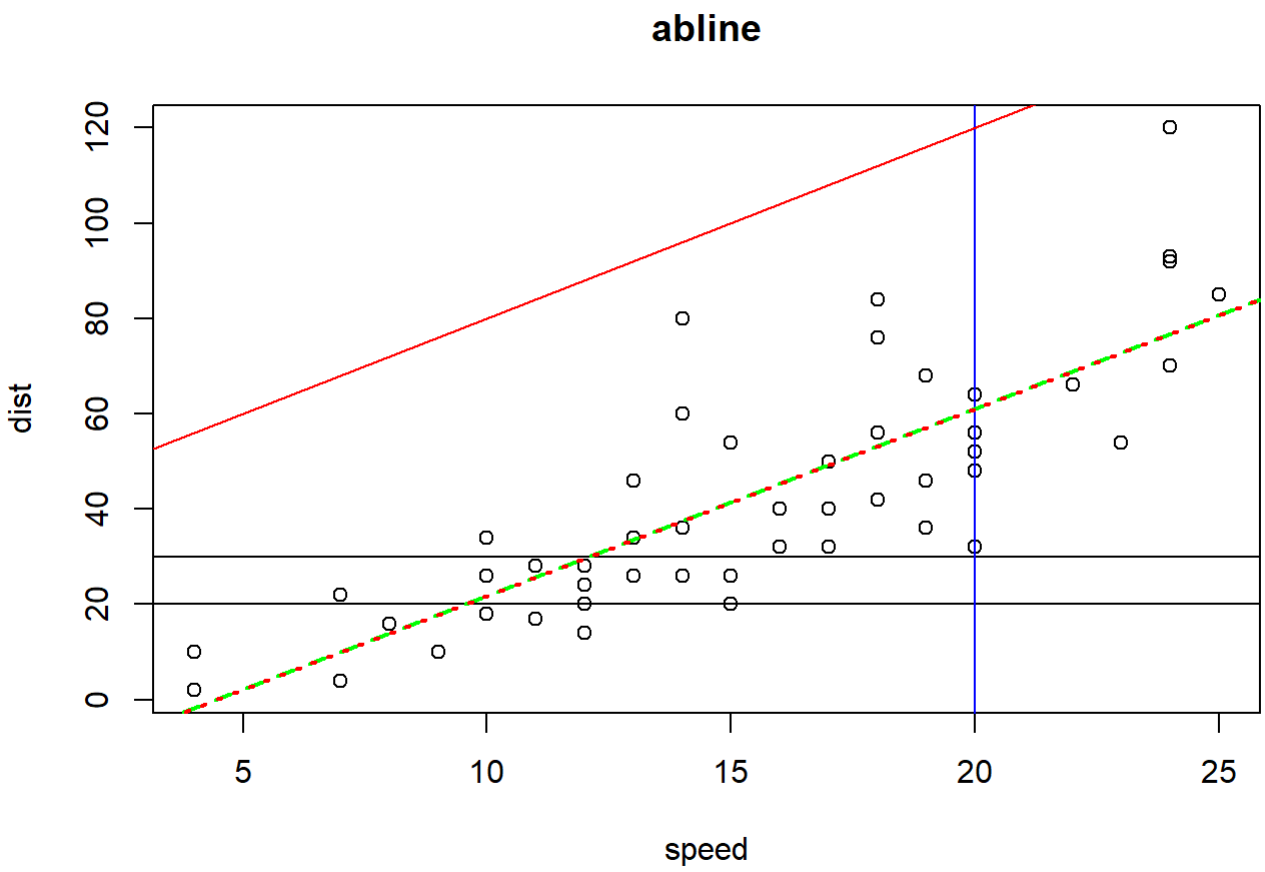
```
## [1] "lm"       "oldClass"
```

```
z$coef
```

```
## (Intercept)       speed
##  -17.579095    3.932409
```

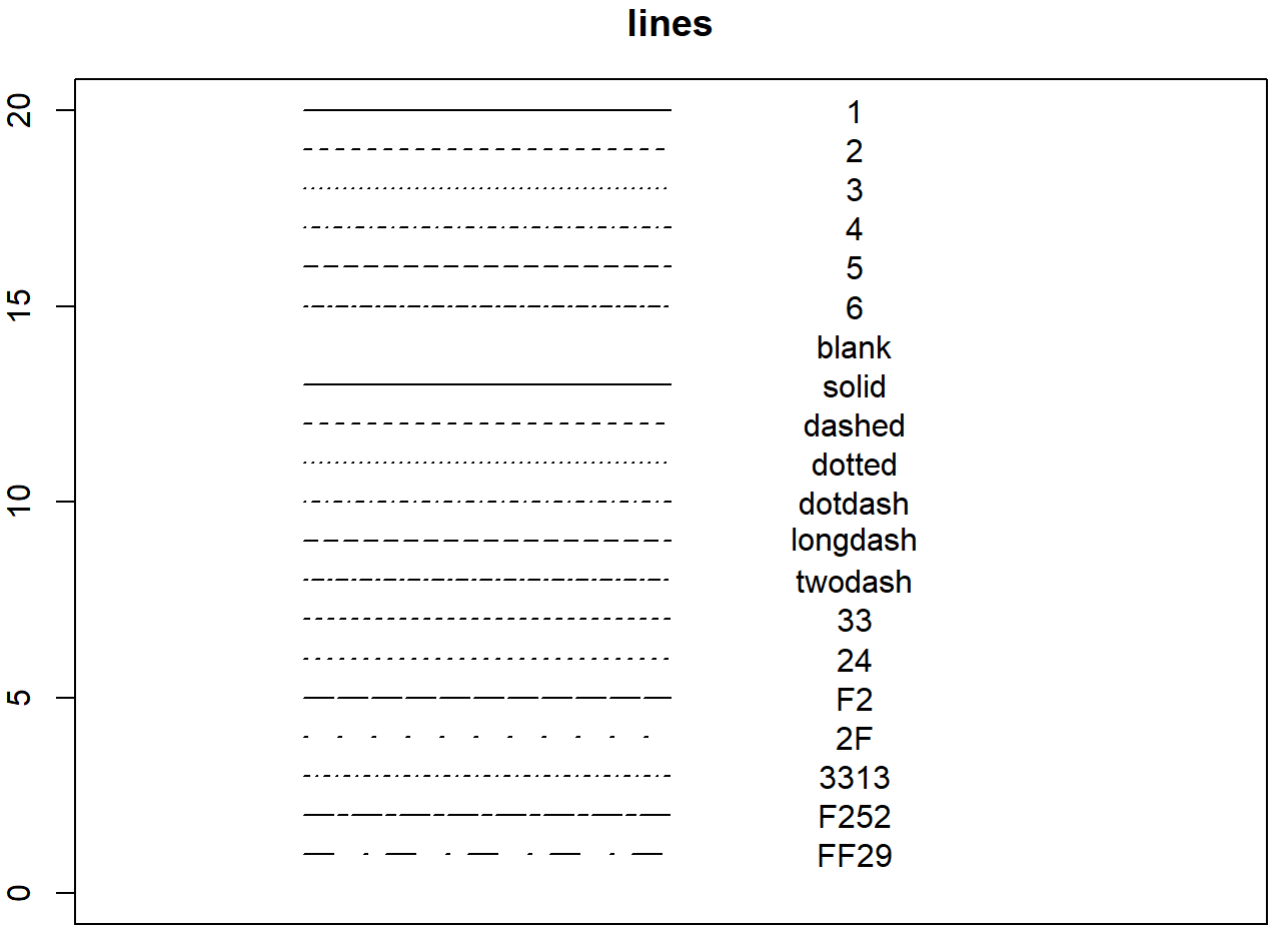
```
plot(cars, main = "abline")
# horizontal
```

```
abline(h = 20)
abline(h = 30)
# vertical
abline(v = 20, col="blue")
# y = a + bx
abline(a = 40, b = 4, col="red")
# reg 인수
abline(z, lty = 2, lwd = 2, col="green")
# coef 인수
abline(z$coef, lty = 3, lwd = 2, col="red")
```



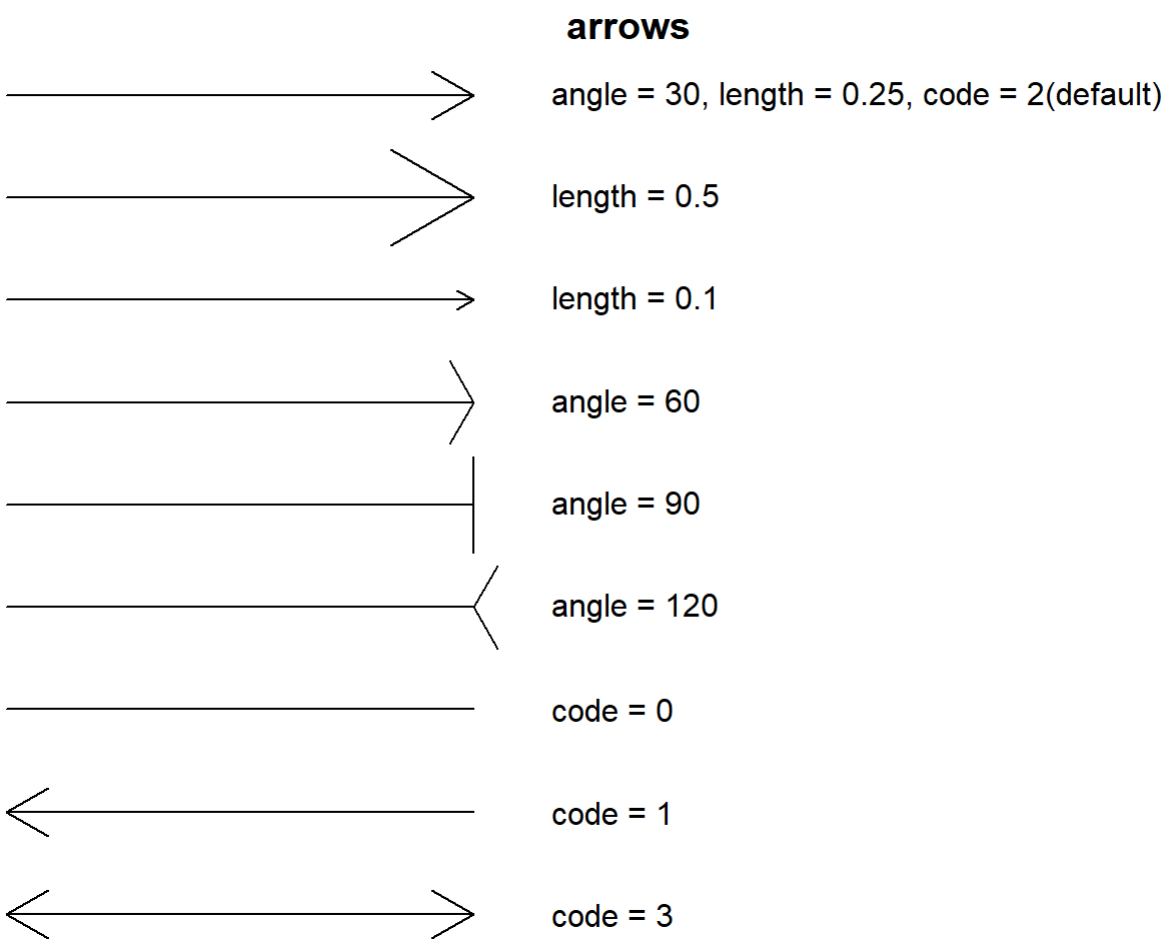

```
# 2.3.3.2 lines() 함수 : 좌표의 점들을 이어서 선을 그리는 함수

# lty 인수 값에 따른 선의 유형 비교
op <- par(no.readonly = TRUE)
par(mar=c(0, 2, 3, 2))
lty1 <- c("blank", "solid", "dashed", "dotted", "dotdash", "longdash","twodash")
lty2 <- c("33", "24", "F2", "2F","3313", "F252","FF29")
plot(0:6, 0:6, type="n", ylim=c(0,20), xlab="", ylab="", main="lines")
lines(c(1, 3), c(20, 20), lty = 1); text(4, 20, "1")
lines(c(1, 3), c(19, 19), lty = 2); text(4, 19, "2")
lines(c(1, 3), c(18, 18), lty = 3); text(4, 18, "3")
lines(c(1, 3), c(17, 17), lty = 4); text(4, 17, "4")
lines(c(1, 3), c(16, 16), lty = 5); text(4, 16, "5")
lines(c(1, 3), c(15, 15), lty = 6); text(4, 15, "6")
lines(c(1, 3), c(14, 14), lty = lty1[1]); text(4, 14, lty1[1])
lines(c(1, 3), c(13, 13), lty = lty1[2]); text(4, 13, lty1[2])
lines(c(1, 3), c(12, 12), lty = lty1[3]); text(4, 12, lty1[3])
lines(c(1, 3), c(11, 11), lty = lty1[4]); text(4, 11, lty1[4])
```

```
lines(c(1, 3), c(10, 10), lty = lty1[5]); text(4, 10, lty1[5])
lines(c(1, 3), c(9, 9), lty = lty1[6]); text(4, 9, lty1[6])
lines(c(1, 3), c(8, 8), lty = lty1[7]); text(4, 8, lty1[7])
lines(c(1, 3), c(7, 7), lty = lty2[1]); text(4, 7, lty2[1])
lines(c(1, 3), c(6, 6), lty = lty2[2]); text(4, 6, lty2[2])
lines(c(1, 3), c(5, 5), lty = lty2[3]); text(4, 5, lty2[3])
lines(c(1, 3), c(4, 4), lty = lty2[4]); text(4, 4, lty2[4])
lines(c(1, 3), c(3, 3), lty = lty2[5]); text(4, 3, lty2[5])
lines(c(1, 3), c(2, 2), lty = lty2[6]); text(4, 2, lty2[6])
lines(c(1, 3), c(1, 1), lty = lty2[7]); text(4, 1, lty2[7])
```

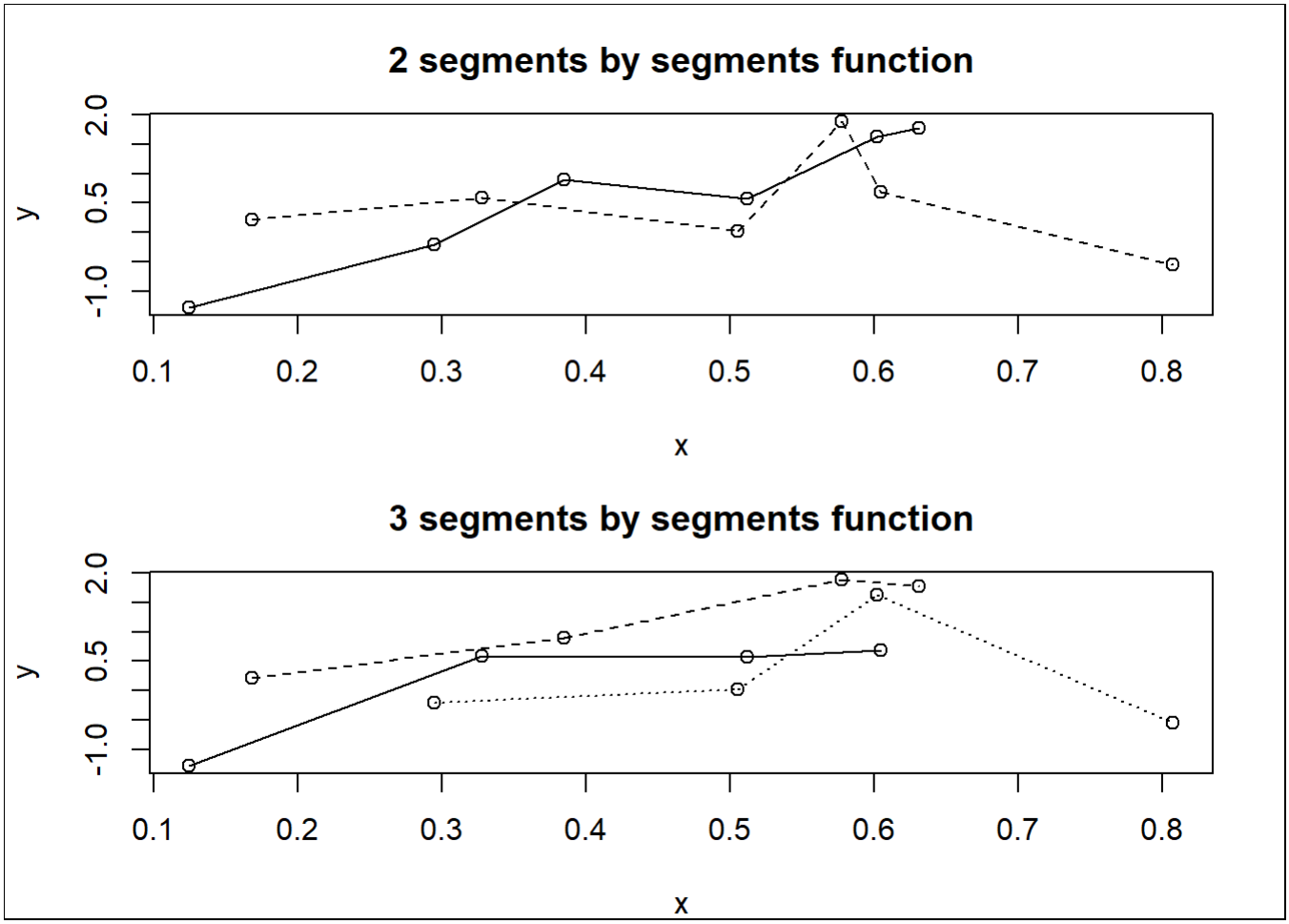



```
par(op)


# 2.3.3.2 arrows() 함수 : 화살표를 그리는 함수

# length, angle, code 인수 값에 따른 선의 유형 비교

op <- par(no.readonly = TRUE)
par(mar=c(0, 0, 2, 0))
plot(1:9, type = "n", axes = FALSE, xlab = "", ylab = "", main = "arrows")
arrows(1, 9, 4, 9, angle = 30, length = 0.25, code = 2)
text(4.5, 9, adj = 0, "angle = 30, length = 0.25, code = 2(default)")
arrows(1, 8, 4, 8, length = 0.5); text(4.5, 8, adj = 0, "length = 0.5")
arrows(1, 7, 4, 7, length = 0.1); text(4.5, 7, adj = 0, "length = 0.1")
arrows(1, 6, 4, 6, angle = 60); text(4.5, 6, adj = 0, "angle = 60")
arrows(1, 5, 4, 5, angle = 90); text(4.5, 5, adj = 0, "angle = 90")
arrows(1, 4, 4, 4, angle = 120); text(4.5, 4, adj = 0, "angle = 120")
arrows(1, 3, 4, 3, code = 0); text(4.5, 3, adj = 0, "code = 0")
arrows(1, 2, 4, 2, code = 1); text(4.5, 2, adj = 0, "code = 1")
```

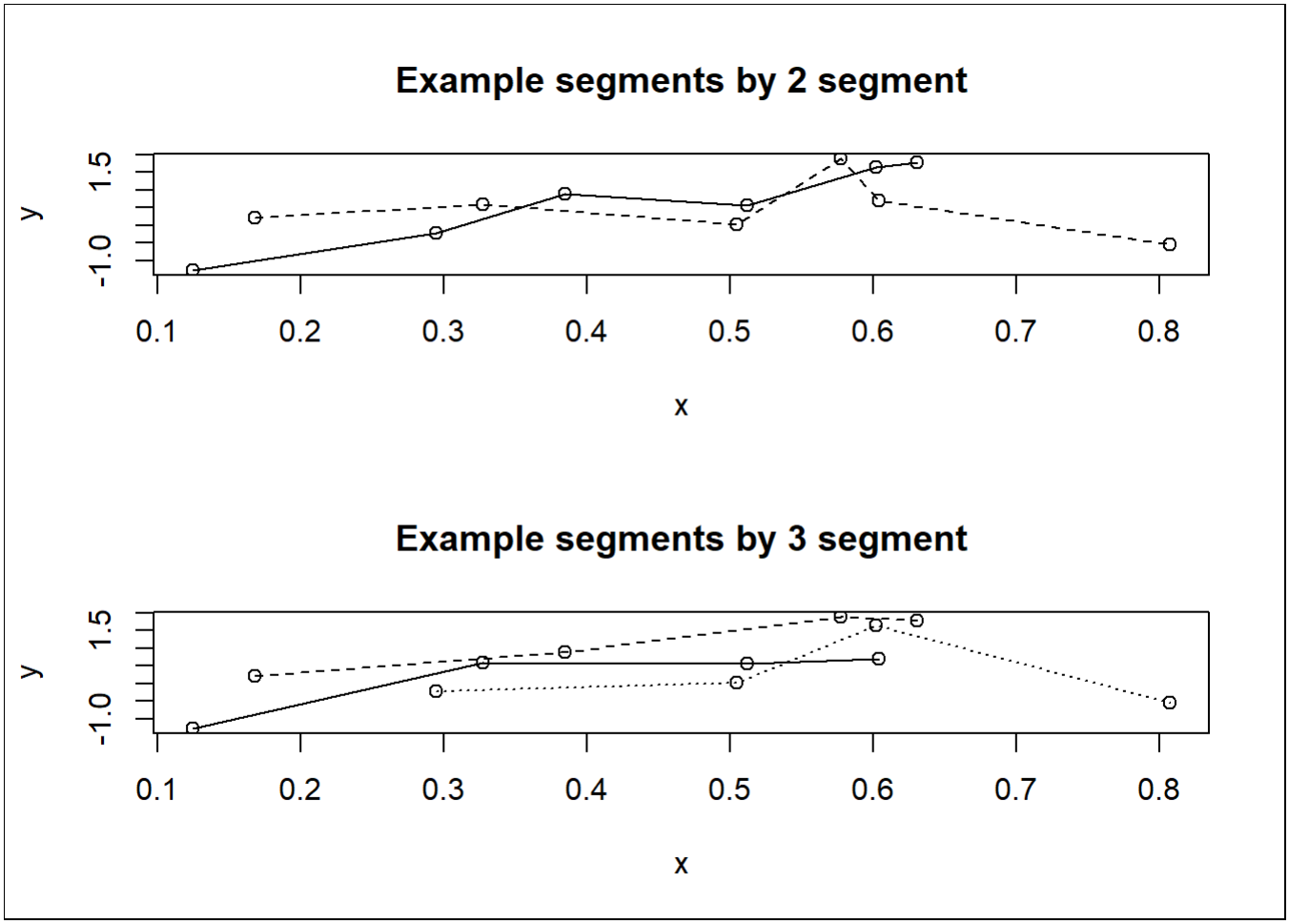
```
arrows(1, 1, 4, 1, code = 3); text(4.5, 1, adj = 0, "code = 3")
```



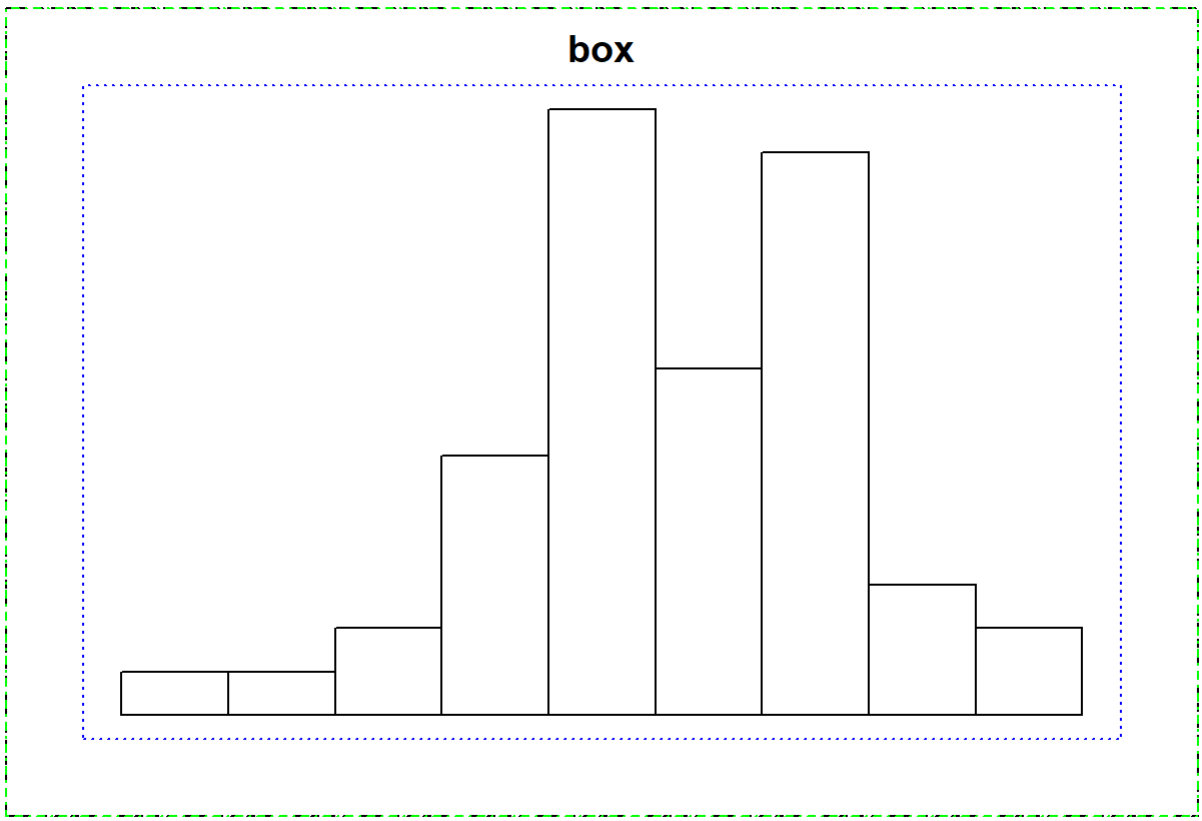

```
par(op)


# 2.3.3.4 segments() 함수 : 좌표점들의 그룹을 만든 후 각각의 그룹별로 꺾은선을 그리는 함수

op <- par(no.readonly = TRUE)
par(mar=c(4, 4, 3, 2), mfrow = c(2, 1))
set.seed(3)
x <- runif(12)
set.seed(4)
y <- rnorm(12)
i <- order(x); x <- x[i]; y <- y[i]
plot(x, y, main = "2 segments by segments function")
s <- seq(length(x) - 1)
segments(x[s], y[s], x[s + 2], y[s + 2], lty = 1:2)
plot(x, y, main = "3 segments by segments function")
s <- seq(length(x) - 2)
segments(x[s], y[s], x[s + 3], y[s + 3], lty = 1:3)
box(which = "outer")
```



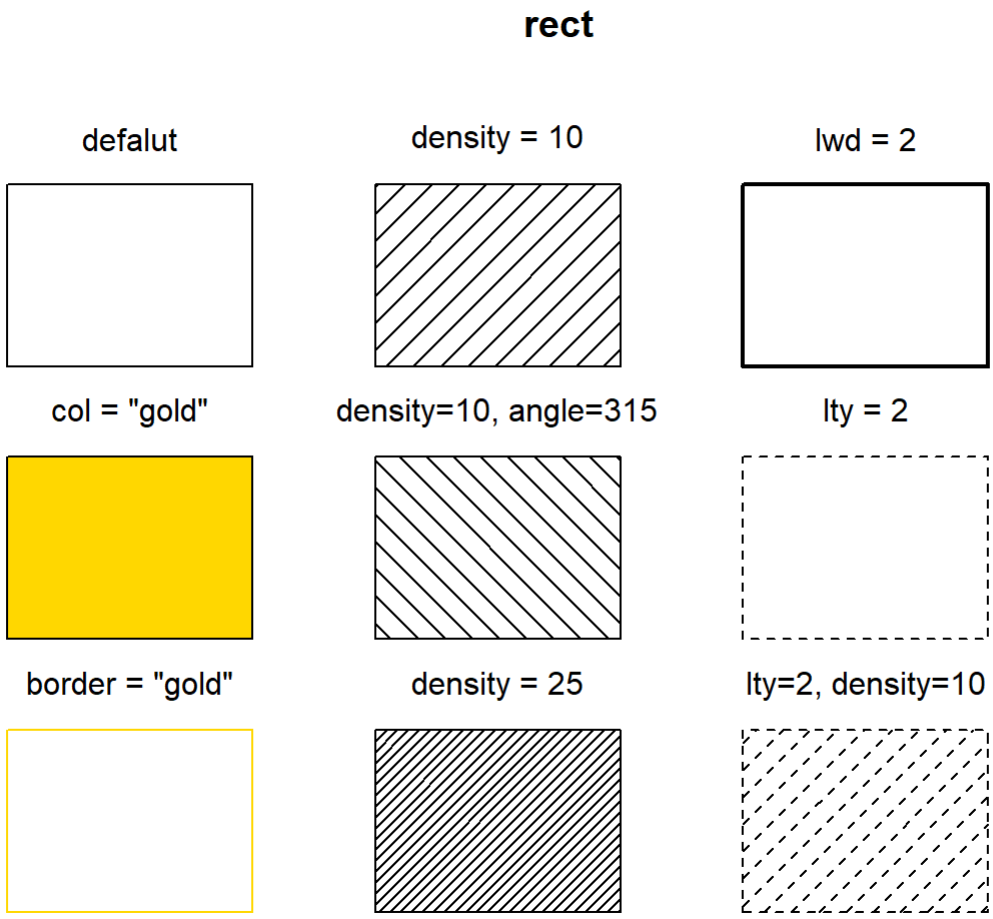

```
par(op)

# lines 함수 사용한 경우
par(mfrow = c(2, 1))
plot(x, y, main = "Example segments by 2 segment")
lines(x[seq(1, 12, 2)], y[seq(1, 12, 2)], lty = 1)
lines(x[seq(2, 12, 2)], y[seq(2, 12, 2)], lty = 2)
plot(x, y, main = "Example segments by 3 segment")
lines(x[seq(1, 12, 3)], y[seq(1, 12, 3)], lty = 1)
lines(x[seq(2, 12, 3)], y[seq(2, 12, 3)], lty = 2)
lines(x[seq(3, 12, 3)], y[seq(3, 12, 3)], lty = 3)
box(which = "outer")
```



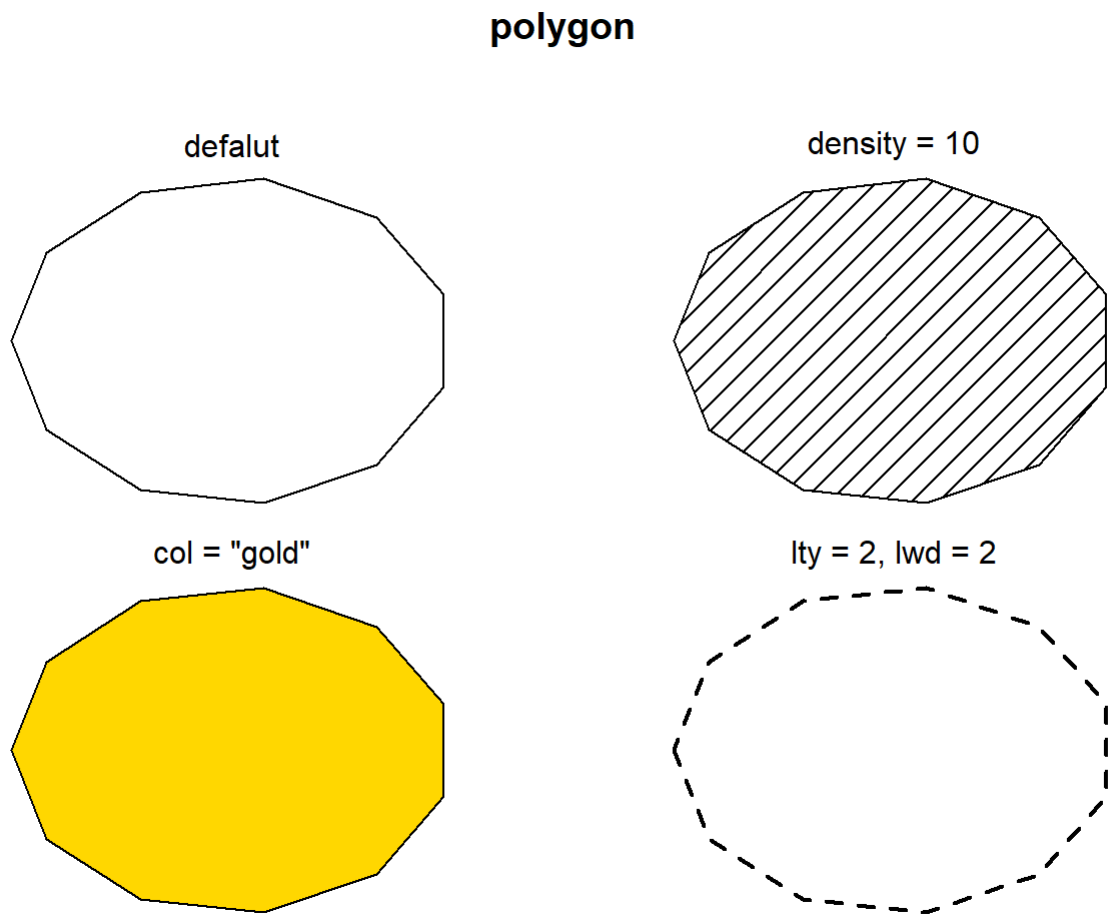

```
par(mfrow=c(1, 1))

# 2.3.4 면을 그리는 함수
# 2.3.4.1 box() 함수 : 현재의 그래픽 장치의 특정 영역에 사각형 상자를 그리는 함수

# box() 함수를 이용한 "outer", "inner", "plot", "figure" 영역 표
op <- par(no.readonly = TRUE)
# margin & outer margin
par(mar = c(2, 2, 2, 2), oma = c(2, 2, 2, 2))
set.seed(1)
hist(rnorm(50), axes = F, xlab = "", ylab = "", main = "box")
# 영역의 종류
whichs <- c("outer", "inner", "plot", "figure")
box(which = whichs[1], lty = 1, lwd = 1.2, col = "red")
box(which = whichs[2], lty = 2, lwd = 1.2, col = "black")
box(which = whichs[3], lty = 3, lwd = 1.2, col = "blue")
box(which = whichs[4], lty = 4, lwd = 1.2, col = "green")
```



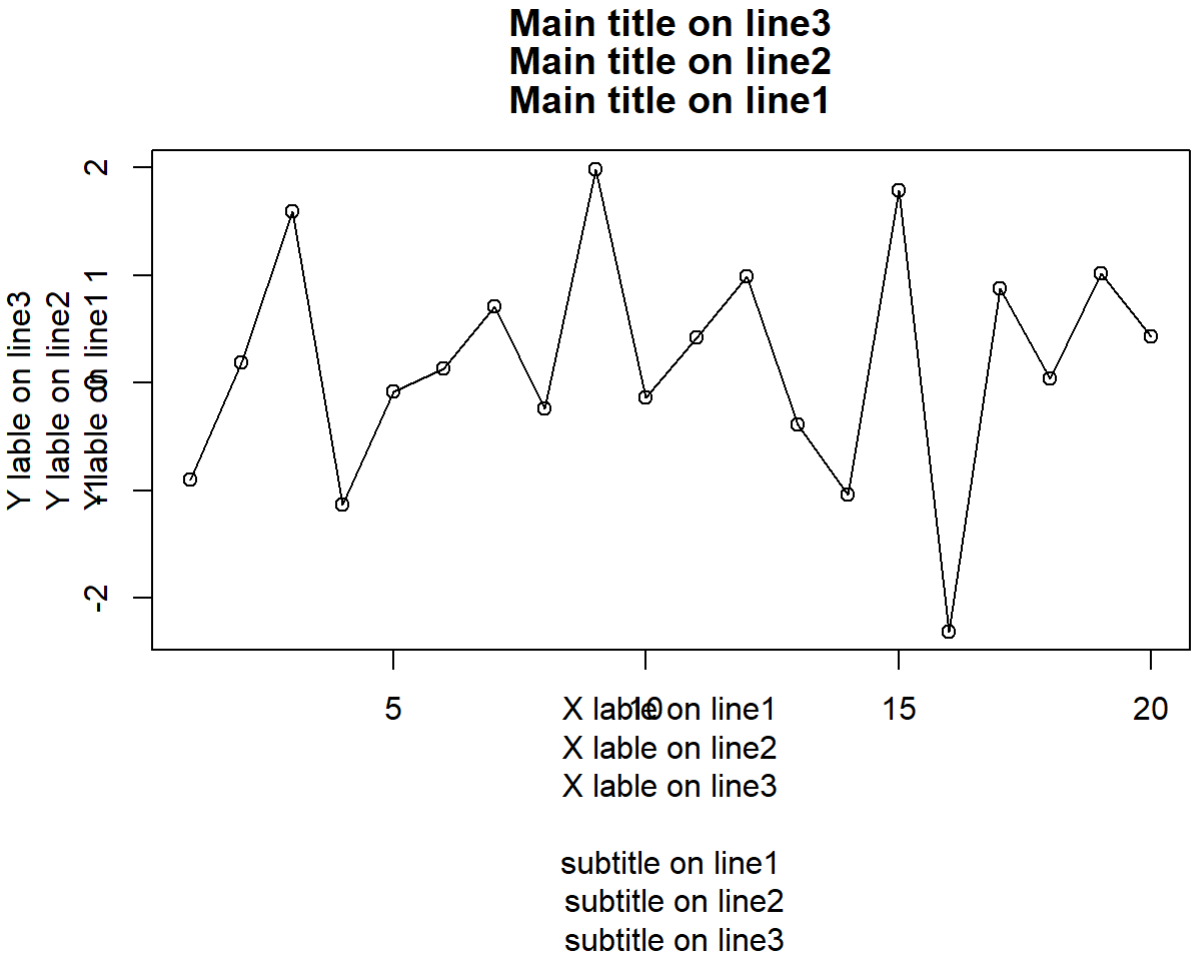

```
# legend(locator(1), legend = whichs, lwd = 1.2, lty = 1:4,
    # col = c("red", "black", "blue", "green"))
par(op)

# 2.3.4.2 rect() 함수 : 플롯 영역 내부의 좌표상에 사각형의 도형을 그리는 함수

op <- par(no.readonly = TRUE)
par(mar = c(0, 2, 2, 2))
plot(1:10, type = "n", main = "rect", xlab = "", ylab = "", axes = F)
rect(xleft = 1, ybottom = 7, xright = 3, ytop = 9)
text(2, 9.5, adj = 0.5, "defalut")
rect(1, 4, 3, 6, col = "gold")
text(2, 6.5, adj = 0.5, "col = \"gold\"")
rect(1, 1, 3, 3, border = "gold")
text(2, 3.5, adj = 0.5, "border = \"gold\"")
rect(4, 7, 6, 9, density = 10)
text(5, 9.5, adj = 0.5, "density = 10")
rect(4, 4, 6, 6, density = 10, angle = 315)
text(5, 6.5, adj = 0.5, "density=10, angle=315")
rect(4, 1, 6, 3, density = 25)
text(5, 3.5, adj = 0.5, "density = 25")
rect(7, 7, 9, 9, lwd = 2)
text(8, 9.5, adj = 0.5, "lwd = 2")
rect(7, 4, 9, 6, lty = 2)
text(8, 6.5, adj = 0.5, "lty = 2")
rect(7, 1, 9, 3, lty = 2, density = 10)
text(8, 3.5, adj = 0.5, "lty=2, density=10")
```




```
par(op)

# 2.3.4.3 polygon() 함수 : 좌표 점들을 이어서 다각형을 그리는 함수

op <- par(no.readonly = TRUE)
par(mar = c(0, 2, 2, 2))
# 원 모양을 만들기 위해 theta를 구함
theta <- seq(-pi, pi, length = 12)
x <- cos(theta)
y <- sin(theta)
plot(1:6, type = "n", main = "polygon", xlab = "", ylab = "", axes = F)
# 좌표 이동을 위한 작업
x1 <- x + 2
y1 <- y + 4.5
polygon(x1, y1)
x2 <- x + 2
y2 <- y + 2
polygon(x2, y2, col = "gold")
x3 <- x + 5
y3 <- y + 4.5
polygon(x3, y3, density = 10)
x4 <- x + 5
y4 <- y + 2
polygon(x4, y4, lty = 2, lwd = 2)
text(2, 5.7, adj = 0.5, "defalut")
text(2, 3.2, adj = 0.5, "col = \"gold\"")
text(5, 5.7, adj = 0.5, "density = 10")
text(5, 3.2, adj = 0.5, "lty = 2, lwd = 2")
```

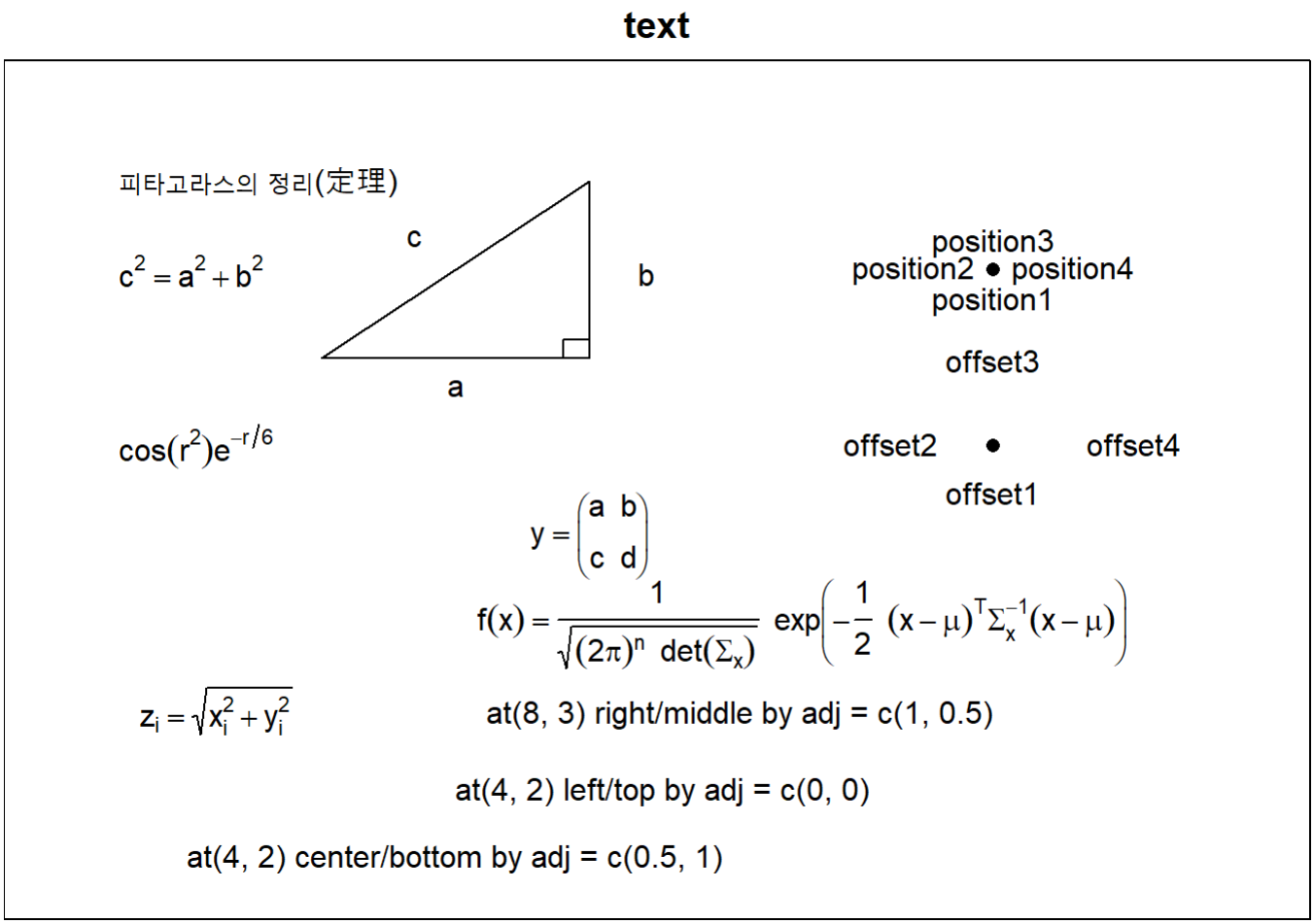



```
par(op)


# 2.3.5 문자를 그리는 함수
# 2.3.5.1 title() 함수 : 플롯에 타이틀 출력

# title() 함수로 그래프의 메인 타이틀, 서브 타이틀, x-축의 이름, y-축의 이름 출력
op <- par(no.readonly = TRUE)
par(mar = c(4, 4, 4, 4), oma = c(4, 0, 0, 0))
set.seed(2)
plot(rnorm(20), type = "o", xlab = "", ylab = "")
title(main = "Main title on line1", line = 1)
title(main = "Main title on line2", line = 2)
title(main = "Main title on line3", line = 3)
title(sub = "subtitle on line1", line = 1, outer = T)
title(sub = " subtitle on line2", line = 2, outer = T)
title(sub = " subtitle on line3", line = 3, outer = T)
title(xlab = "X lable on line1", line = 1)
title(xlab = "X lable on line2", line = 2)
title(xlab = "X lable on line3", line = 3)
title(ylab = "Y lable on line1", line = 1)
title(ylab = "Y lable on line2", line = 2)
title(ylab = "Y lable on line3", line = 3)
```



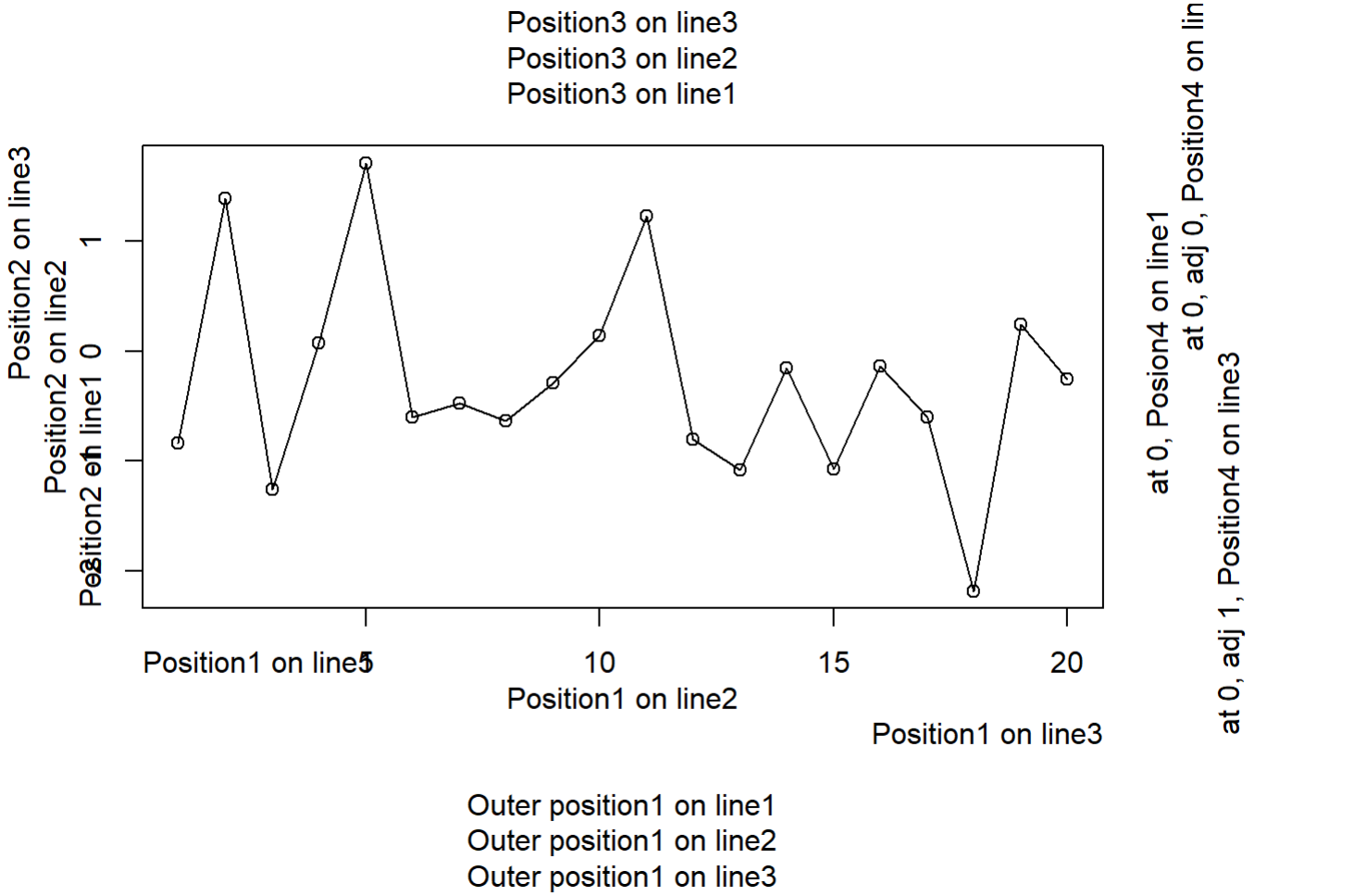

```
par(op)

# 2.3.5.2 text() 함수 : 플롯 영역의 좌표에 문자 출력

op <- par(no.readonly = TRUE)
par(mar = c(0, 0, 2, 0))
plot(1:10, 1:10, type = "n", xlab = "", ylab = "", main = "text")
text(1.5, 9, adj = 0, labels = "피타고라스의 정리(定理)")
polygon(c(5, 3, 5), c(9, 7, 7))
polygon(c(5, 5, 4.8, 4.8), c(7, 7.2, 7.2, 7))
text(3.64, 8.36, adj = 0, labels = "c")
text(3.94, 6.67, adj = 0, labels = "a")
text(5.36, 7.95, adj = 0, labels = "b")
# Example expression labels
text(1.5, 8, adj = 0, labels = expression(c^2 == a^2 + b^2))
text(1.5, 6, adj = 0, labels = expression(cos(r^2) * e^{-r/6}))
text(2, 3, adj = 0.3, labels = expression(z[i] == sqrt(x[i]^2 + y[i]^2)))
text(9, 4, adj = 1, labels = expression(f(x) == frac(1, sqrt((2 *
    pi)^n ~ ~det(Sigma[x])))) ~ ~exp * bgroup("(", -frac(1, 2) ~ ~(x -
    mu)^T * Sigma[x]^-1 * (x - mu), ")")))
text(5, 5, adj = 0.5, labels = expression(y == bgroup("(", atop(a ~ ~b, c ~ ~d), ")")))
# Example position by pos
points(8, 8, pch = 16)
text(8, 8, "position1", pos = 1)
text(8, 8, "position2", pos = 2)
text(8, 8, "position3", pos = 3)
text(8, 8, "position4", pos = 4)
# Example offset
points(8, 6, pch = 16)
text(8, 6, "offset1", pos = 1, offset = 1)
```

```
text(8, 6, "offset2", pos = 2, offset = 1.5)
text(8, 6, "offset3", pos = 3, offset = 2)
text(8, 6, "offset4", pos = 4, offset = 2.5)
# Example adj by adj(x, y)
text(4, 2, "at(4, 2) left/top by adj = c(0, 0)", adj = c(0, 0))
text(4, 1.5, "at(4, 2) center/bottom by adj = c(0.5, 1)", adj = c(0.5, + 1))
text(8, 3, "at(8, 3) right/middle by adj = c(1, 0.5)", adj = c(1, 0.5))
```

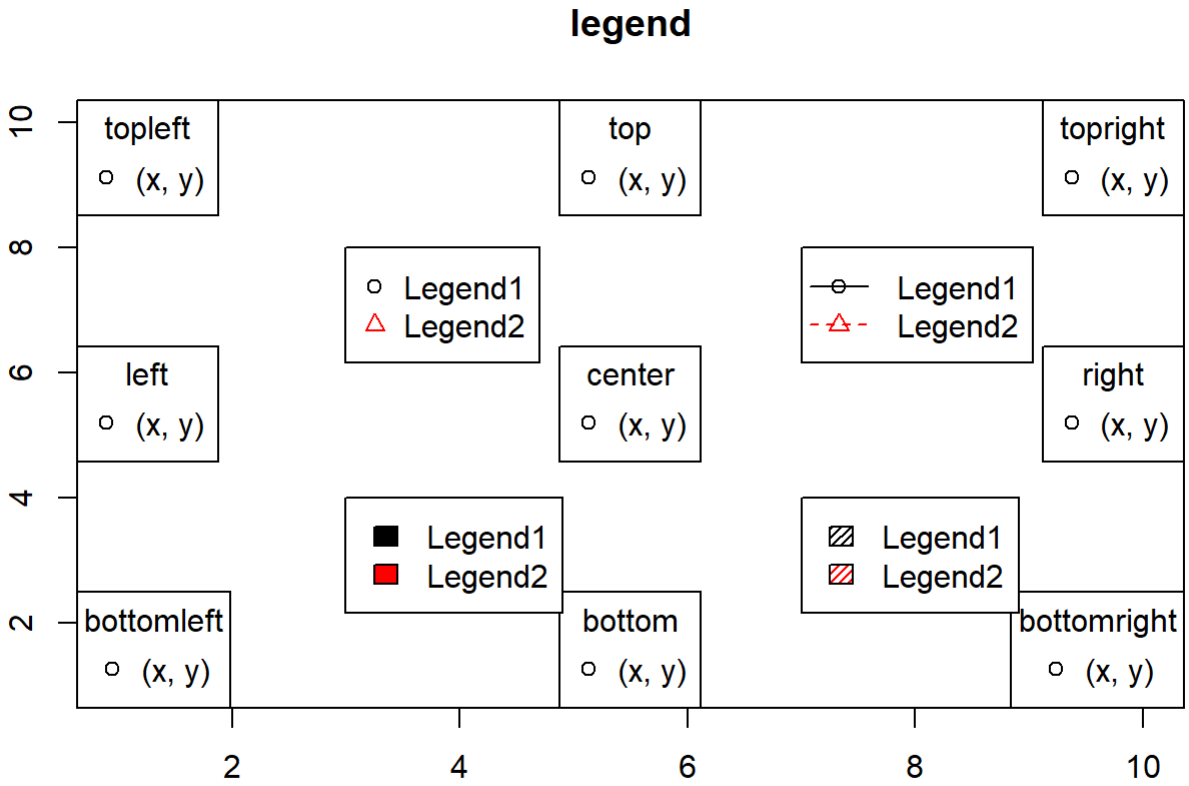



```
par(op)


# 2.3.5.3 mtext() 함수 : 마진이나 바깥 마진에 문자를 출력하는 함수

# mtext() 함수와 title() 함수의 비교
op <- par(no.readonly = TRUE)
par(mar = c(4, 4, 4, 4), oma = c(4, 0, 0, 0))
set.seed(5)
plot(rnorm(20), type = "o", xlab = "", ylab = "")
mtext("Position3 on line1", line = 1)
mtext("Position3 on line2", side = 3, line = 2)
mtext("Position3 on line3", side = 3, line = 3)
mtext("Outer position1 on line1", side = 1, line = 1, outer = T)
mtext("Outer position1 on line2", side = 1, line = 2, outer = T)
mtext("Outer position1 on line3", side = 1, line = 3, outer = T)
mtext("Position1 on line1", side = 1, line = 1, adj = 0)
mtext("Position1 on line2", side = 1, line = 2, adj = 0.5)
mtext("Position1 on line3", side = 1, line = 3, adj = 1)
mtext("Position2 on line1", side = 2, line = 1, adj = 0)
mtext("Position2 on line2", side = 2, line = 2, adj = 0.5)
```

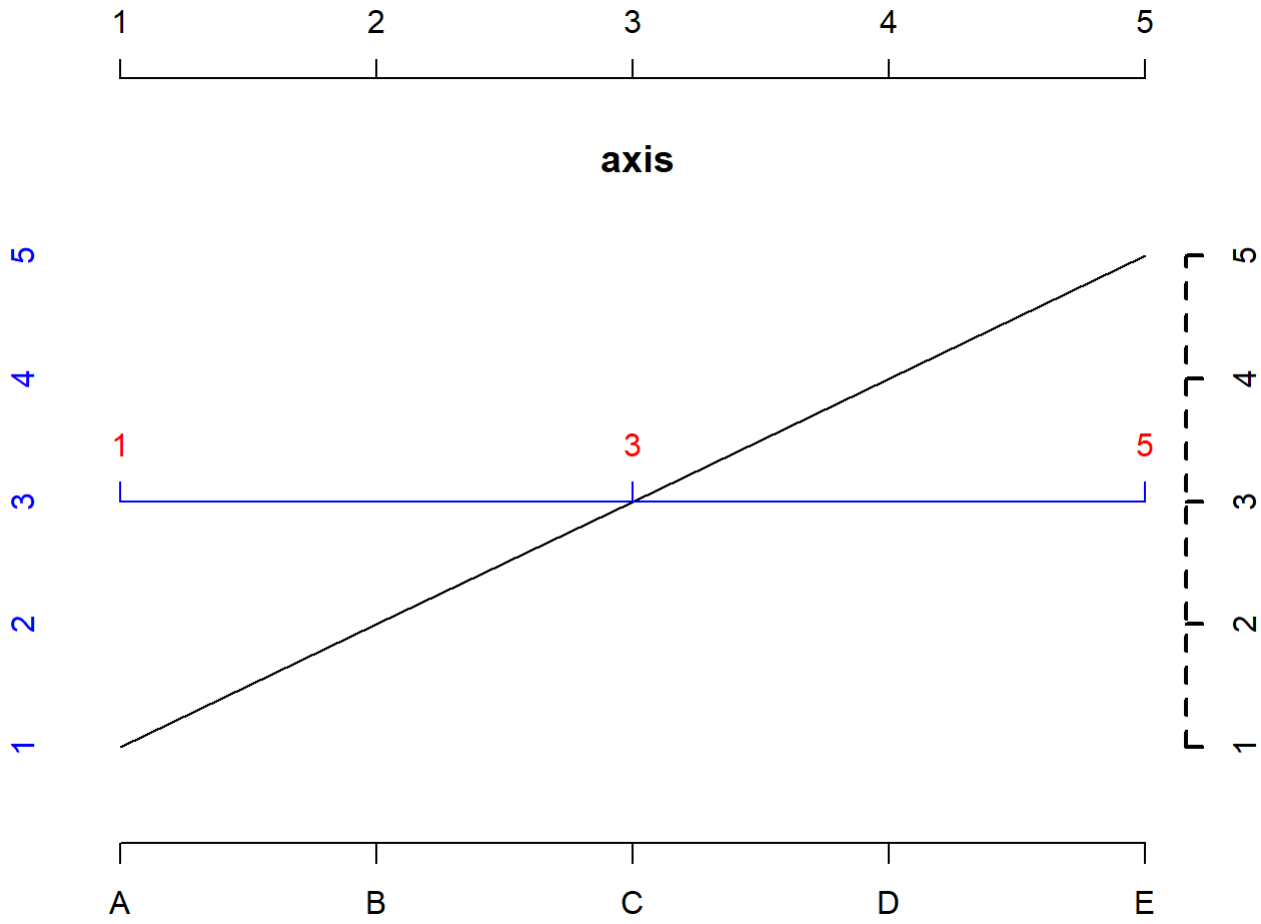
```
mtext("Position2 on line3", side = 2, line = 3, adj = 1)
mtext("at 0, Posion4 on line1", side = 4, line = 1, at = 0)
mtext("at 0, adj 0, Position4 on line2", side = 4, line = 2, at = 0, adj = 0)
mtext("at 0, adj 1, Position4 on line3", side = 4, line = 3, at = 0, adj = 1)
```



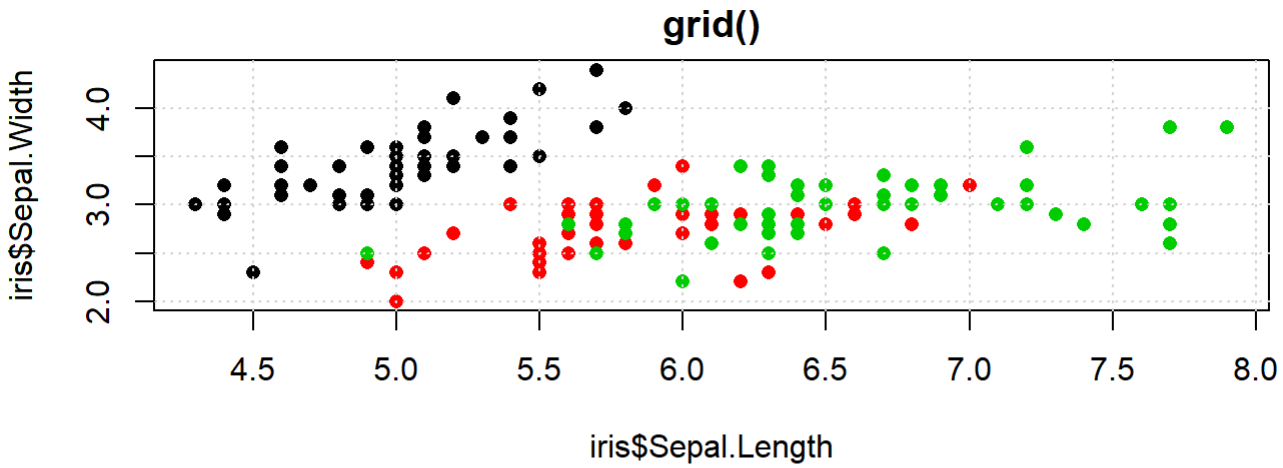

```
par(op)

# 2.3.6 범례를 그리는 함수
# 2.3.6.1 legend() 함수

plot(1:10, type = "n", xlab = "", ylab = "", main = "legend")
legend("bottomright", "(x, y)", pch = 1, title = "bottomright")
legend("bottom", "(x, y)", pch = 1, title = "bottom")
legend("bottomleft", "(x, y)", pch = 1, title = "bottomleft")
legend("left", "(x, y)", pch = 1, title = "left")
legend("topleft", "(x, y)", pch = 1, title = "topleft")
legend("top", "(x, y)", pch = 1, title = "top")
legend("topright", "(x, y)", pch = 1, title = "topright")
legend("right", "(x, y)", pch = 1, title = "right")
legend("center", "(x, y)", pch = 1, title = "center")
legends <- c("Legend1", "Legend2")
legend(3, 8, legend = legends, pch = 1:2, col = 1:2)
legend(7, 8, legend = legends, pch = 1:2, col = 1:2, lty = 1:2)
legend(3, 4, legend = legends, fill = 1:2)
legend(7, 4, legend = legends, fill = 1:2, density = 30)
```

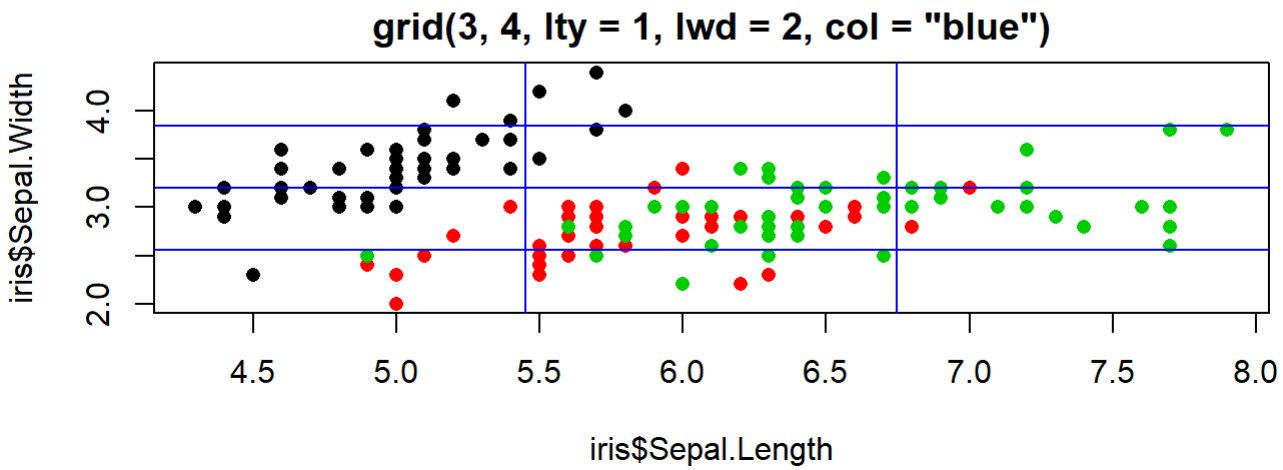



```
# legend(locator(1), legend = "Locator", fill = 1)

# 2.3.7 좌표축을 그리는 함수
# 2.3.7.1 axis() 함수

op <- par(no.readonly = TRUE)
par(oma = c(0, 0, 2, 0))
plot(1:5, type = "l", main = " axis", axes = FALSE, xlab = "", ylab = "")
axis(side = 1, at = 1:5, labels = LETTERS[1:5], line = 2)
# tick = F 이므로 col.axis는 의미 없음
axis(side = 2, tick = F, col.axis = "blue")
axis(side = 3, outer = T)
axis(side = 3, at = c(1, 3, 5), pos = 3, col = "blue", col.axis = "red")
axis(side = 4, lty = 2, lwd = 2)
```

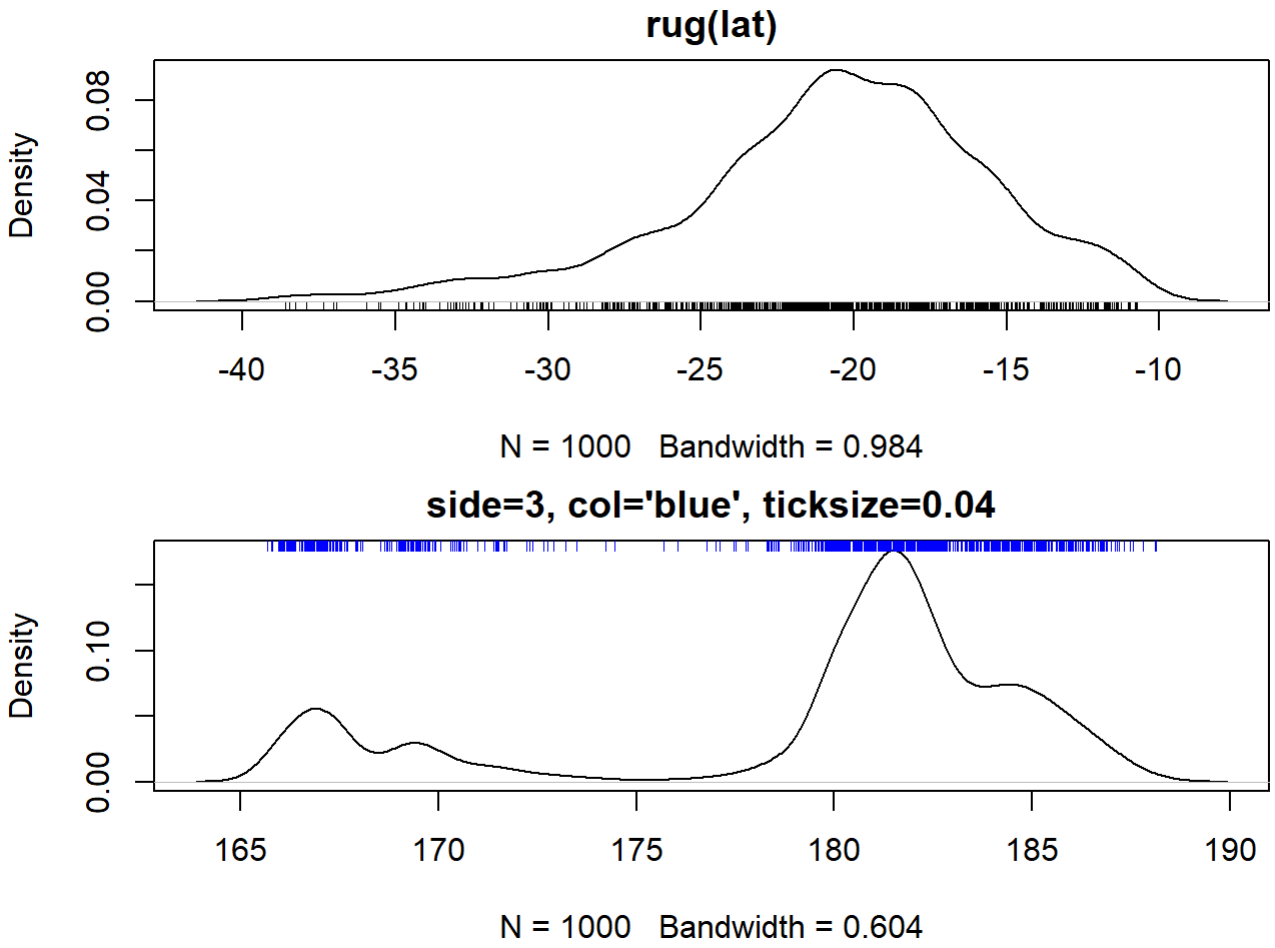



```
par(op)


# 2.3.8 기타 저수준 그래픽 함수
# 2.3.8.1 grid() 함수 : 좌표 평면에 격자를 그리는 함수
op <- par(no.readonly = TRUE)
par(mar = c(4, 4, 2, 2), mfrow = c(2, 1))
plot(iris$Sepal.Length, iris$Sepal.Width, pch = 16, col = as.integer(iris$Species))
# (1)
grid()
title("grid()")
plot(iris$Sepal.Length, iris$Sepal.Width, pch = 16, col = as.integer(iris$Species))
# (2)
grid(3, 4, lty = 1, lwd = 1.2, col = "blue")
title("grid(3, 4, lty = 1, lwd = 2, col = \"blue\")")
```

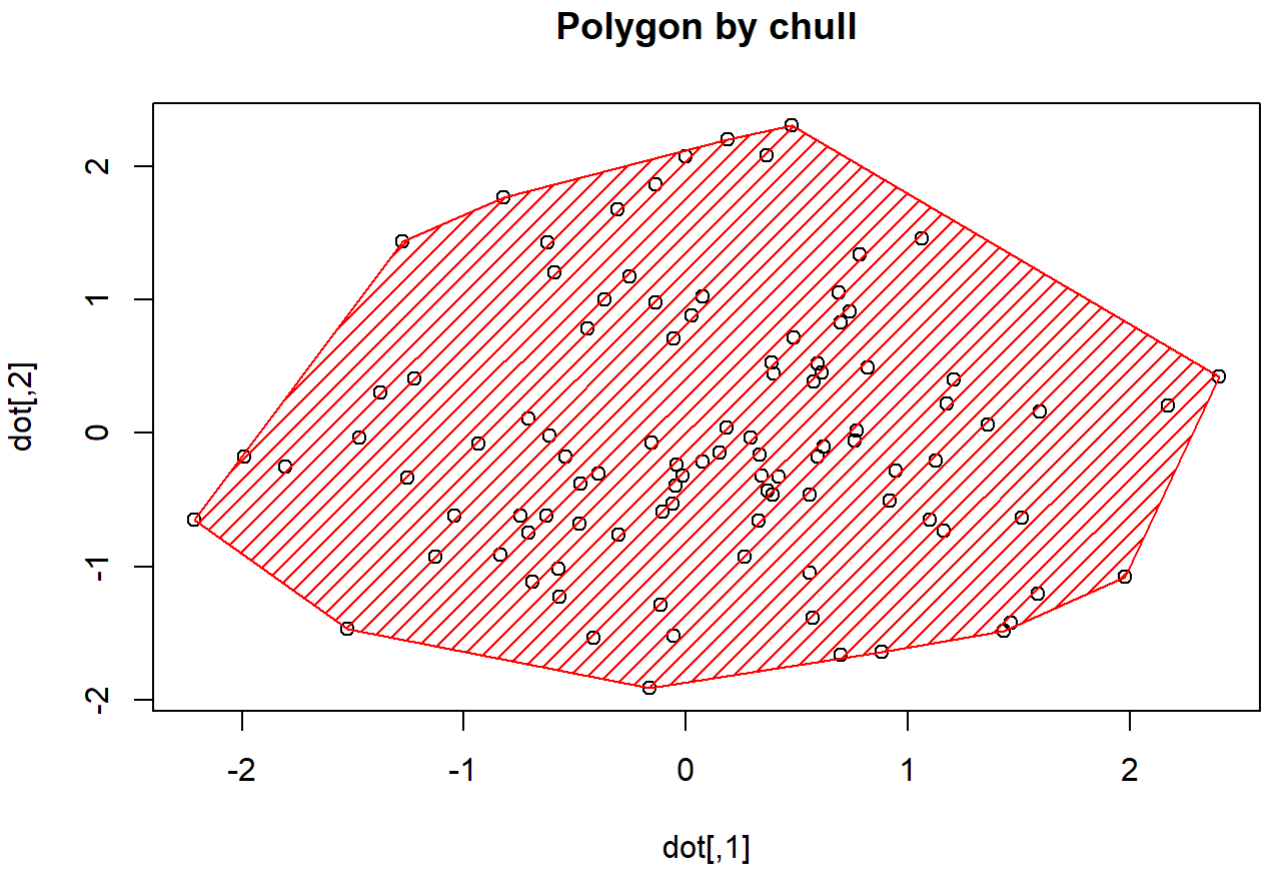





```
par(op)

# 2.3.8.2 rug() 함수 : 그래프에 일차원 정보인 러그(rug)를 추가하는 함수. 러그란 데이터들을 일차원의 좌표축에 표시하는 방법

op <- par(no.readonly = TRUE)
par(mar = c(4, 4, 2, 2), mfrow = c(2, 1))
plot(density(quakes$lat), main = "rug(lat)")
# (1)
rug(quakes$lat)
plot(density(quakes$long), main = "side=3, col='blue', ticksize=0.04")
# (2)
rug(quakes$long, side = 3, col = "blue", ticksize = 0.04)
```



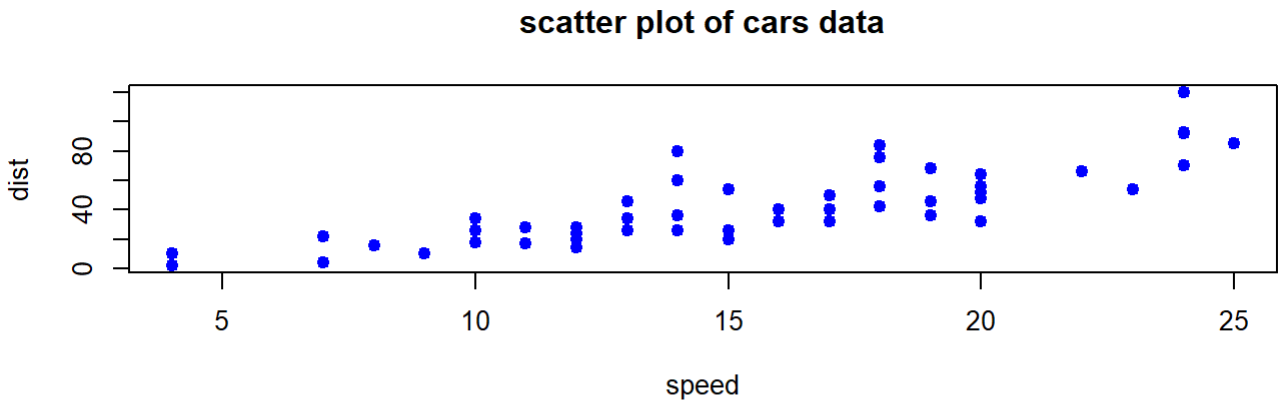

```
par(op)

# 2.3.8.3 chull() 함수 : 모든 좌표 점을 포함한 볼록한(convex) 테두리(hull) 좌표 점들의 원소를 구하는 함수. 직접적으로 플롯을 그리지는 않고 플롯을 그리는 정보로 이용.

set.seed(1)
dot <- matrix(rnorm(200), ncol = 2)
plot(dot)
chull.data <- chull(dot)
polygon(dot[chull.data, ], angle = 45, density = 15, col = "red")
title(main = "Polygon by chull")
```

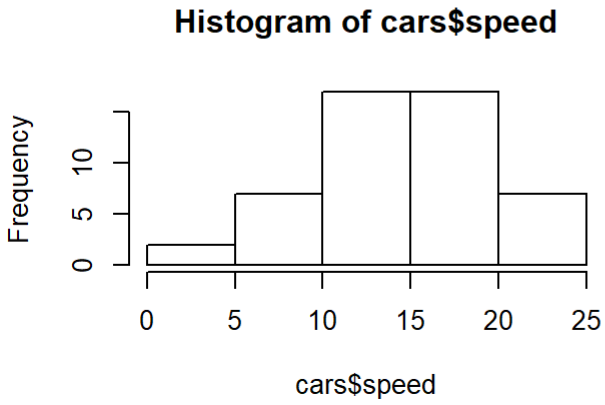


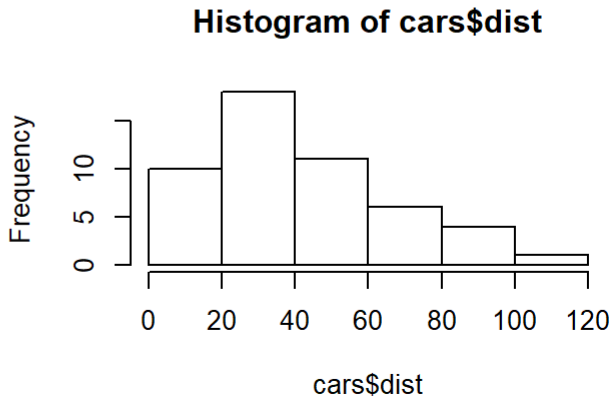

```
# 2.4 par() 함수
# 2.4.1 mfrow, mfcol 인수와 유사함수 : 그림 영역을 분할해서 플롯 영역 생성하고 배치의 순서를 설정. 일반적으로 한화면에 여러 플롯을 그려 비교하는 경우 유용
# 기본 값은 mforw=c(1,1), mfcol=c(1,1)으로 모두 한 화면에 하나의 플롯을 그리도록 설
# 예를 들면 mfrow=c(3,2)의 경우 3행 2열의 6개 플롯 영역을 만듬

# 2.4.1.1 layout() 함수 : mfrow, mfcol 인수와 유사한 기능. 대칭이 아닌 모양으로도 분할이 가

(m <- matrix(c(1, 1, 2, 3), ncol = 2, byrow = T))
```

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

```
layout(mat = m)
plot(cars, main = "scatter plot of cars data", pch = 19, col = 4)
hist(cars$speed)
hist(cars$dist)
```

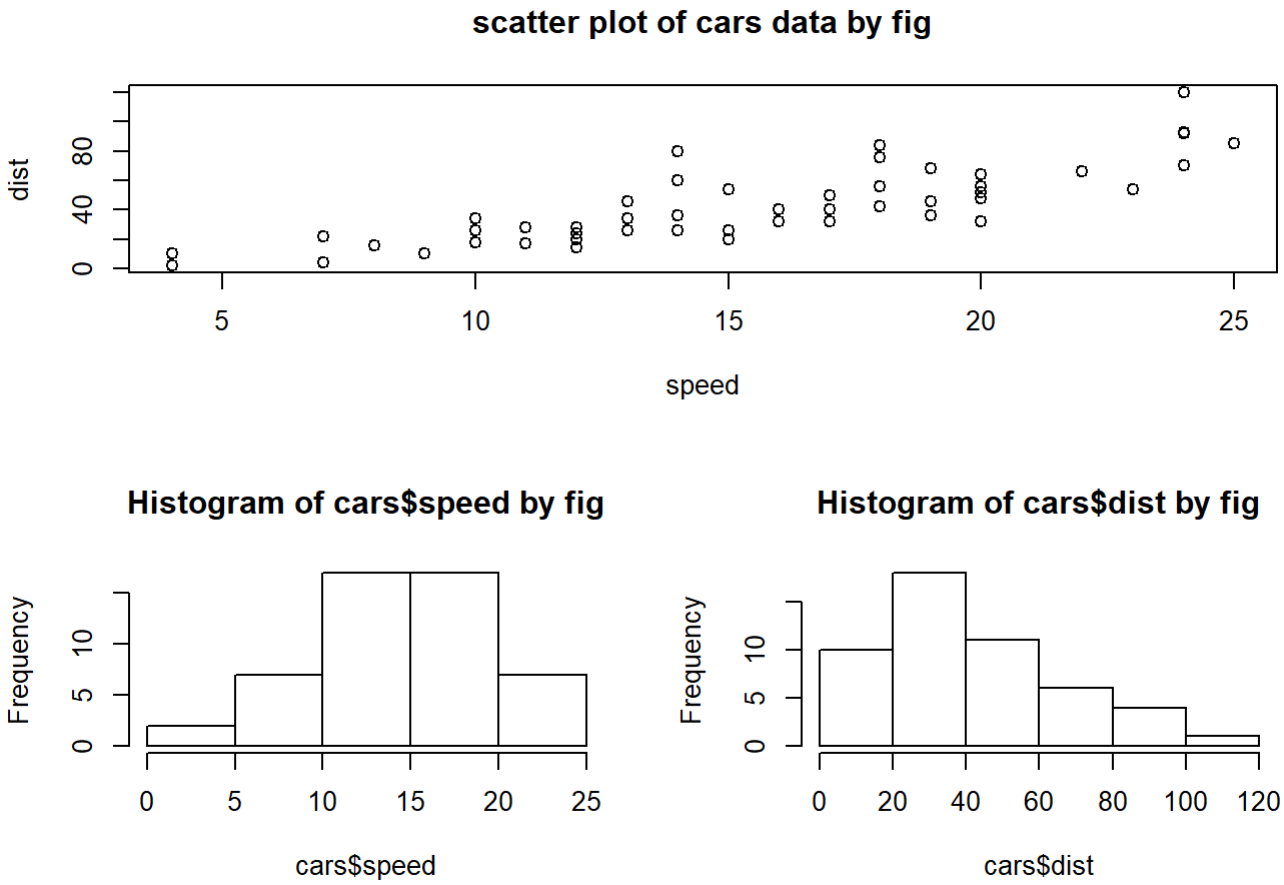






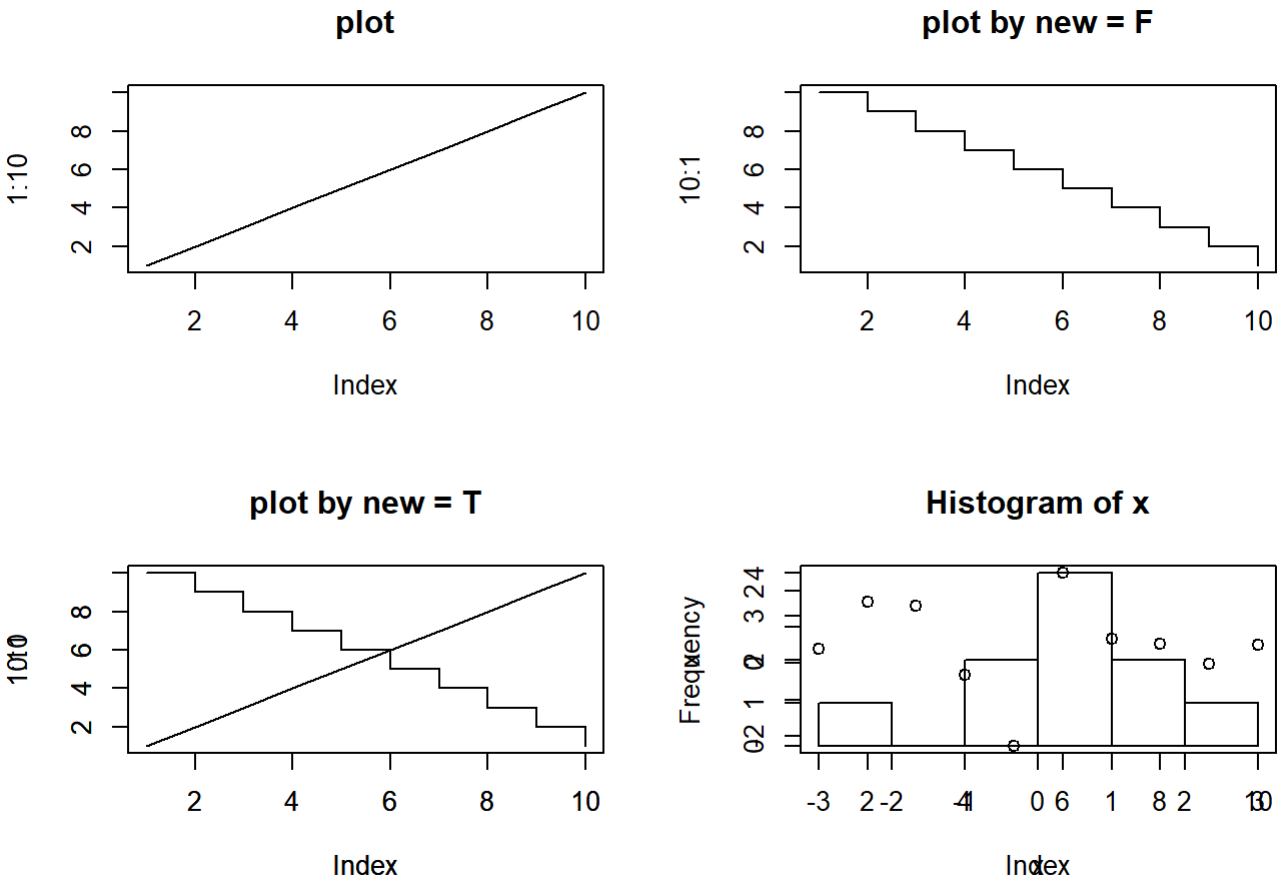

```
# 2.4.1.2 layout.show() 함수 : layout() 함수를 이용해서 분할한 그래픽 화면의 모양을 파악할 수 있도록 해당 구역의 테두리에 선을 그림

# 2.4.1.3 split.screen() 함수 : laayout() 함수처럼 그래픽 장치를 여러 개의 화면으로 분할하는 함수. layout() 함수보다 더 부가적인 기능을 수행하는 몇개의 함수를 제공
# split.screen() : 화면을 분할
# screen() : 분할된 화면을 지정
# erase.screen() : 지정된 화면의 플롯을 지움
# close.screen() : 화면 분할 작업을 마침

# op <- par(no.readonly = TRUE)
## 바탕색을 흰색으로 지정
# par(bg = "white")
## 상하 2개로 화면분할
# split.screen(fig = c(2, 1))
## 2번(아래) 화면을 좌우 두 개로 분할
# split.screen(c(1, 2), screen = 2)
## 3번(아래 왼쪽) 화면을 지정
# screen(n = 3)
# hist(cars$speed)
## 1번(위쪽) 화면을 지정
# screen(1)
# plot(cars, main = "scatter plot of cars data by split.screen")
## 4번(아래 오른쪽) 화면을 지정
# screen(4)
# hist(cars$dist)
## 1번 화면을 바탕색으로 칠함(지움)
# erase.screen(n = 1)
## 다시 1번 화면(위쪽)을 지정
# screen(1)
```

```
# plot(cars, main = "scatter plot of cars data by split.screen", pch = 19,
#       col = "blue")
# # 화면 분할 정의를 마침
# close.screen(all = TRUE)
# par(op)

# 2.4.2 fig 인수 : 그래픽 장치의 영역에서 그림 영역의 크기와 위치를 설정

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
par(fig = c(0, 1, 0.5, 1))
plot(cars, main = "scatter plot of cars data by fig")
par(fig = c(0, 0.5, 0, 0.5), new = T)
hist(cars$speed, main = "Histogram of cars$speed by fig")
par(fig = c(0.5, 1, 0, 0.5), new = T)
hist(cars$dist, main = "Histogram of cars$dist by fig")
```

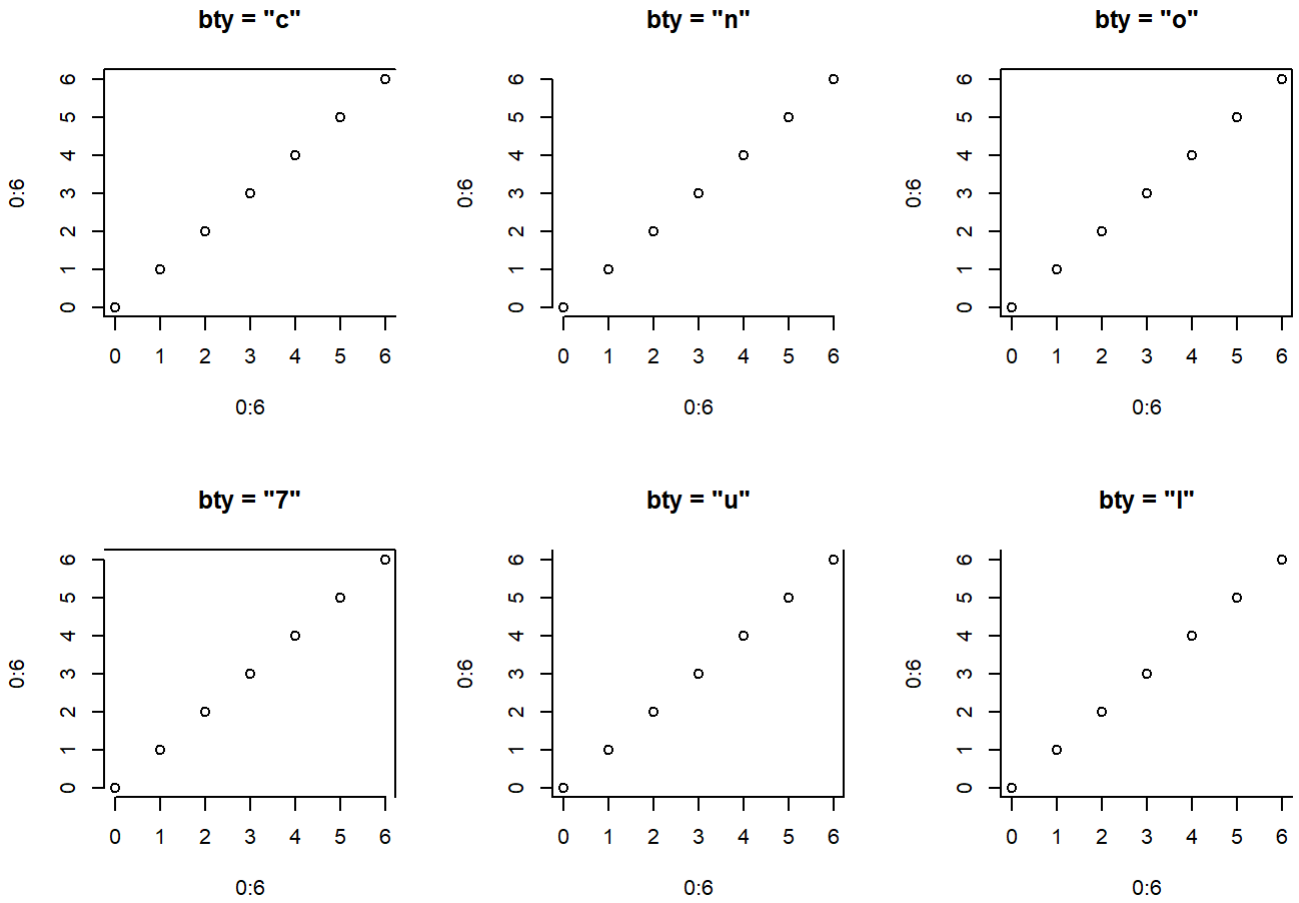



```
par(op)

# 2.4.3 new 인수 : 고수준 그래픽 함수가 호출될 때 그림 영역의 내용을 초기화하는 여부를 설정

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
plot(1:10, type = "l", main = "plot")
par(new = F)
plot(10:1, type = "s", main = "plot by new = F")
plot(1:10, type = "l")
par(new = T)
plot(10:1, type = "s", main = "plot by new = T")
```

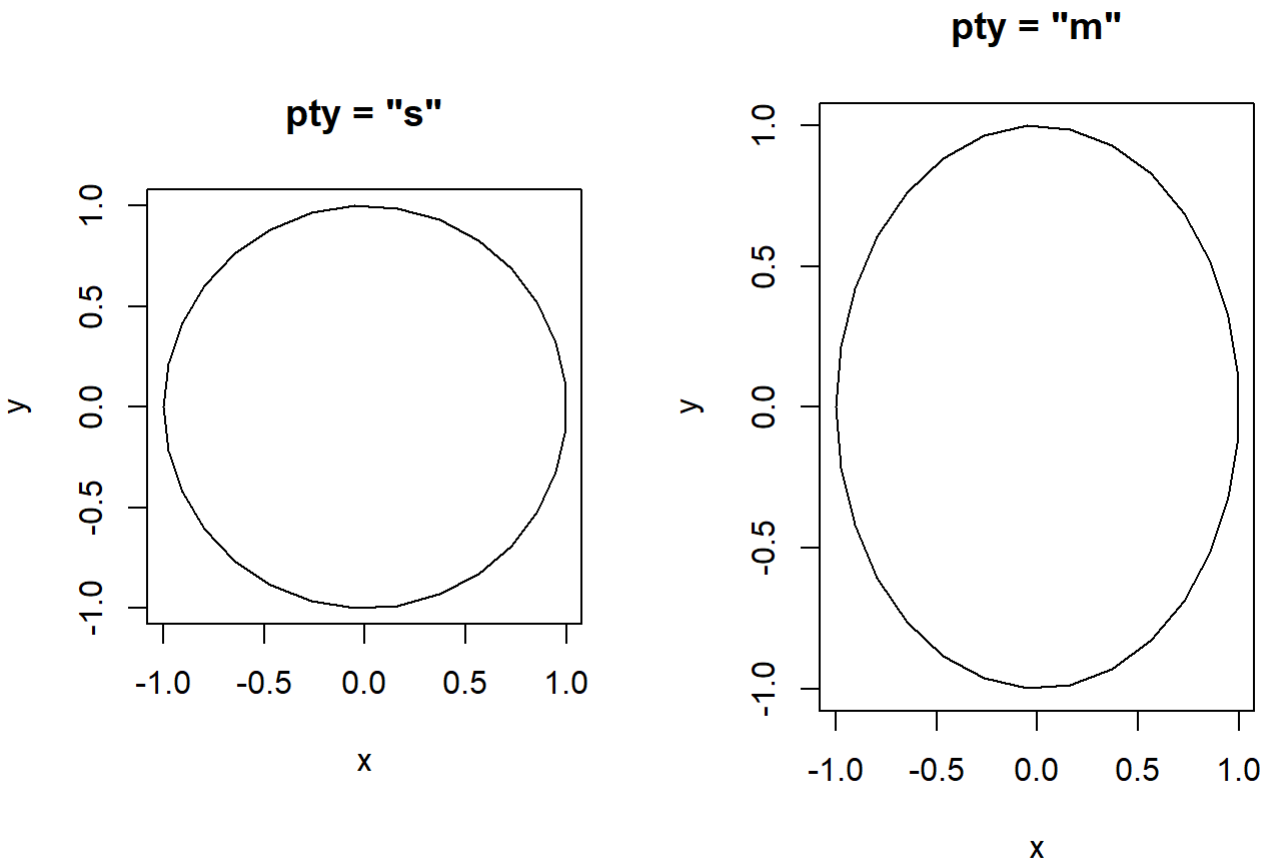
```
x <- rnorm(10)
plot(x)
par(new = T)
hist(x)
```




```
par(op)

# 2.4.4 bty 인수 : bty(box type) 인수는 플롯 영역을 둘러싼 상자의 모양을 설정

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 3), bty = "l")
# C모양(1, 2, 3 영역)
plot(0:6, 0:6, bty = "c", main = " bty = \"c\" ")
# 출력하지 않음
plot(0:6, 0:6, bty = "n", main = " bty = \"n\" ")
# O모양(1, 2, 3, 4 영역)
plot(0:6, 0:6, bty = "o", main = " bty = \"o\" ")
# 7모양(3, 4 영역)
plot(0:6, 0:6, bty = "7", main = " bty = \"7\" ")
# U모양(1, 2, 4 영역)
plot(0:6, 0:6, bty = "u", main = " bty = \"u\" ")
# L영역(1, 2 영역)
plot(0:6, 0:6, main = " bty = \"l\" ")
```



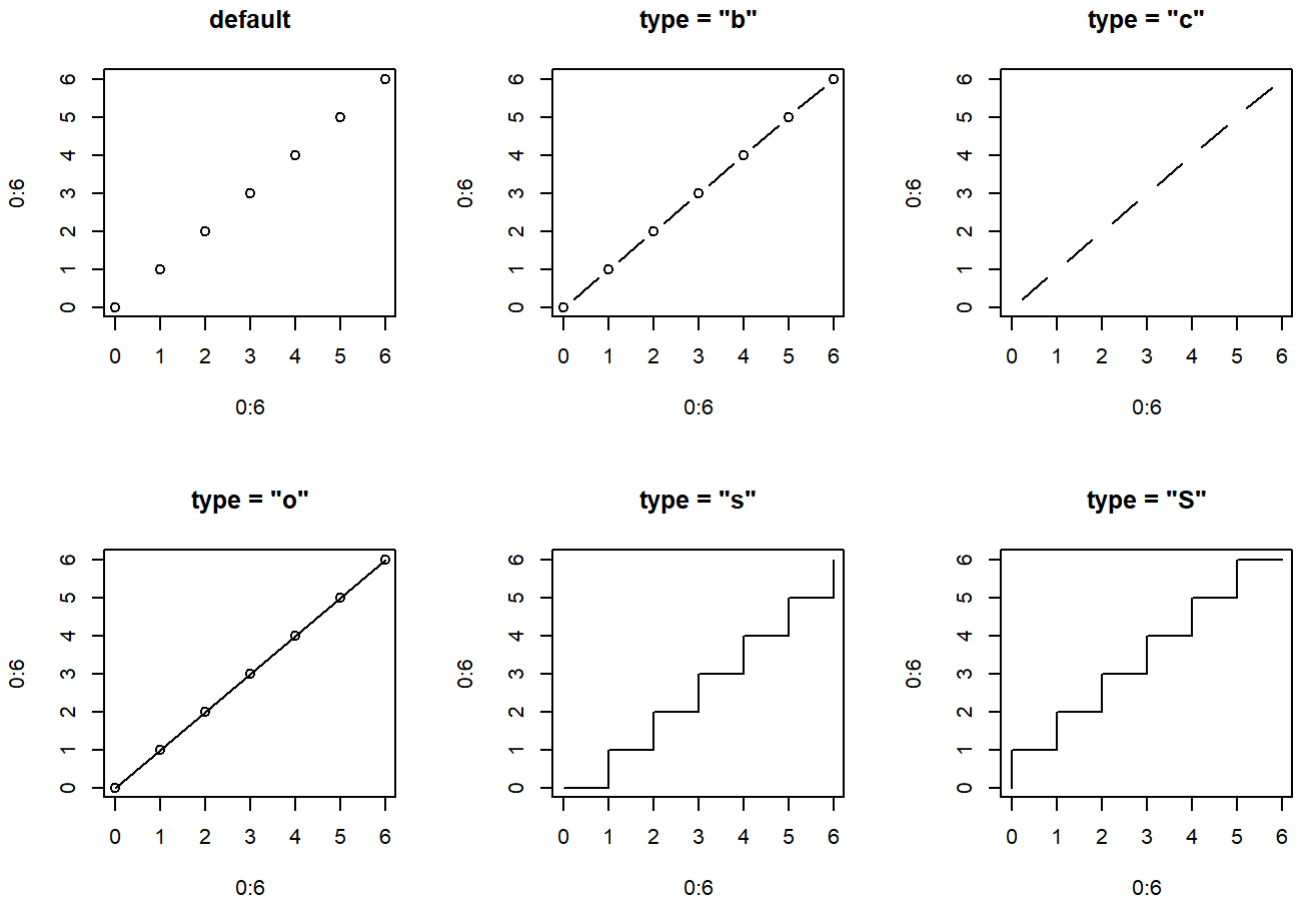

```
par(op)


# 2.4.5 pty 인수 : pty(plot type) 인수는 플롯 영역의 형태를 지정.

op <- par(no.readonly = TRUE)
theta <- seq(-pi, pi, length = 30)
x <- cos(theta)
y <- sin(theta)
par(mfrow = c(1, 2), pty = "s", bty = "o")
plot(x, y, type = "l", main = "pty = \"s\"")
par(pty = "m")
plot(x, y, type = "l", main = "pty = \"m\"")
```




```
par(op)

# 2.4.6 type 인수 : 좌표 영역에 데이터가 표현되는 형태를 설정하는 인수

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 3), type = "n")
```

```
## Warning in par(mfrow = c(2, 3), type = "n"): 그래픽 파라미터 "type"는 필요
## 하지 않습니다
```

```
plot(0:6, 0:6, main = "default")
plot(0:6, 0:6, type = "b", main = "type = \"b\"")
plot(0:6, 0:6, type = "c", main = "type = \"c\"")
plot(0:6, 0:6, type = "o", main = "type = \"o\"")
plot(0:6, 0:6, type = "s", main = "type = \"s\"")
plot(0:6, 0:6, type = "S", main = "type = \"S\"")
```




```
par(op)


# 2.4.7 pch 인수 : 점으로 표시될 문자를 지정

par("pch")
```

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

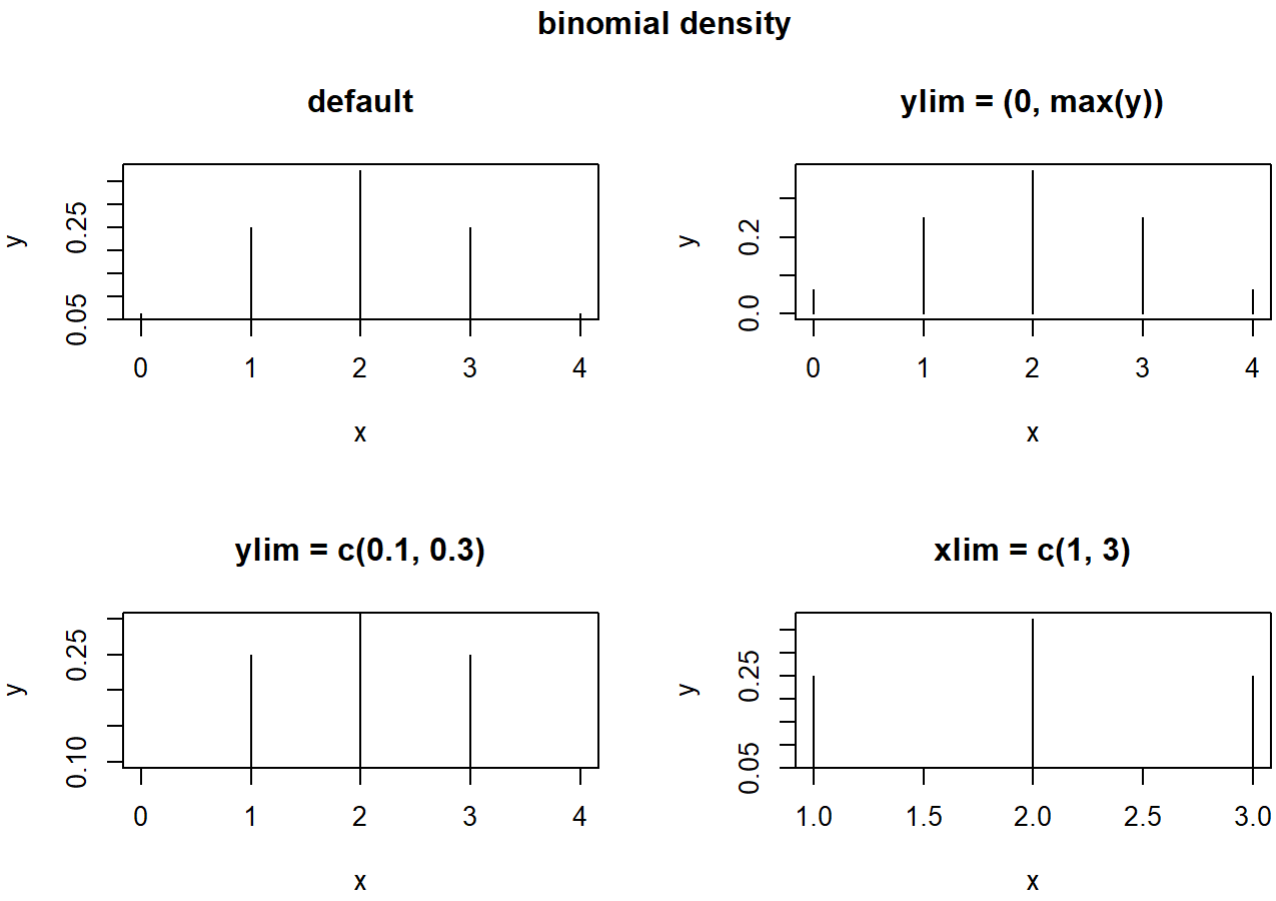
```
# 2.4.8 lty 인수 : 선의 종류를 지정

par("lty")
```

```
## [1] "solid"
```

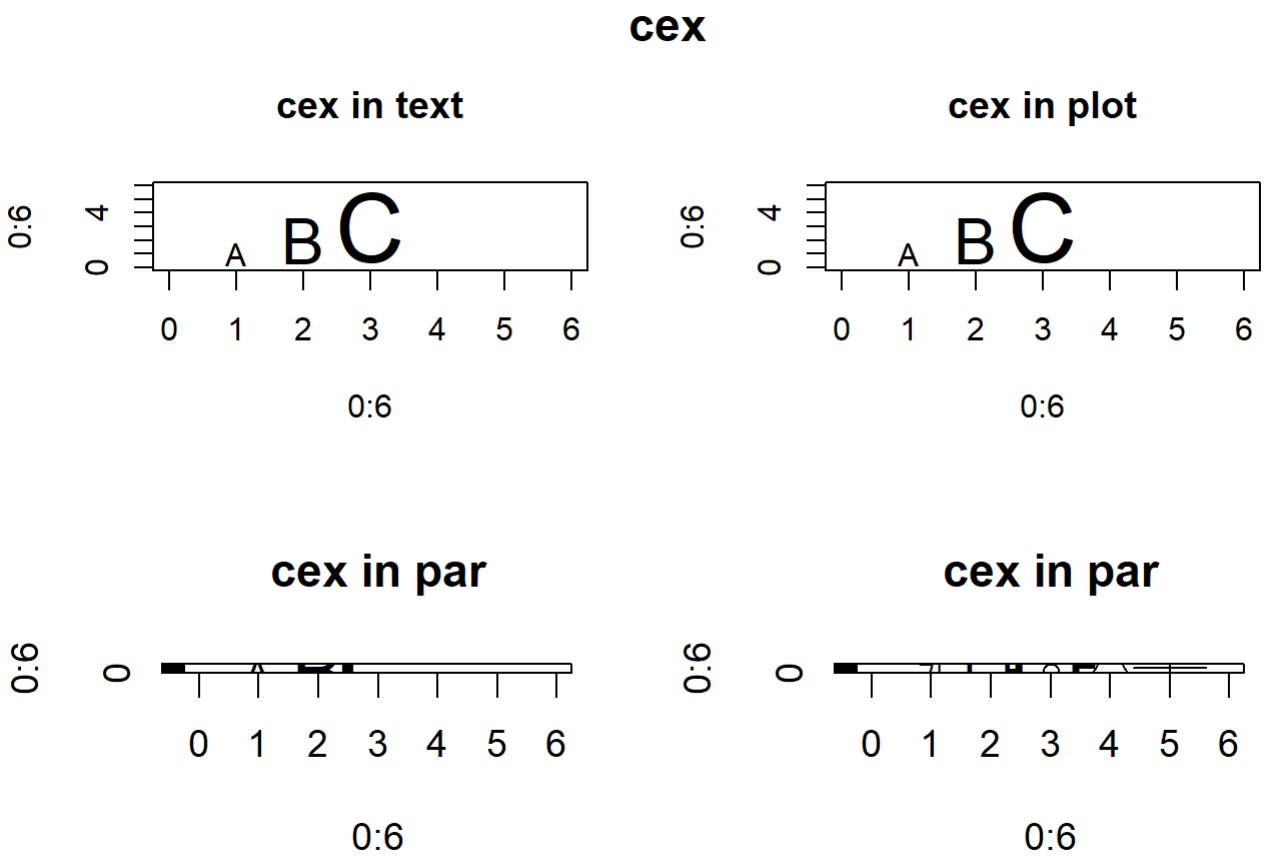
```
# 2.4.9 xlab, ylab 인수 : x-축과 y-축의 라벨 지정

# 2.4.10 xlim, ylim 인수 : x-축과 y-축의 범위를 지정

op <- par(no.readonly = TRUE)
x <- 0:4
set.seed(7)
(y <- dbinom(x, size = 4, prob = 0.5))
```

```
## [1] 0.0625 0.2500 0.3750 0.2500 0.0625
```

```
par(oma = c(0, 0, 2, 0), mfrow = c(2, 2))
plot(x, y, type = "h", main = "default")
plot(x, y, type = "h", ylim = c(0, max(y)), main = "ylim = (0, max(y))")
plot(x, y, type = "h", ylim = c(0.1, 0.3), main = "ylim = c(0.1, 0.3)")
plot(x, y, type = "h", xlim = c(1, 3), main = "xlim = c(1, 3)")
title(main = "binomial density", line = 0, outer = T)
```



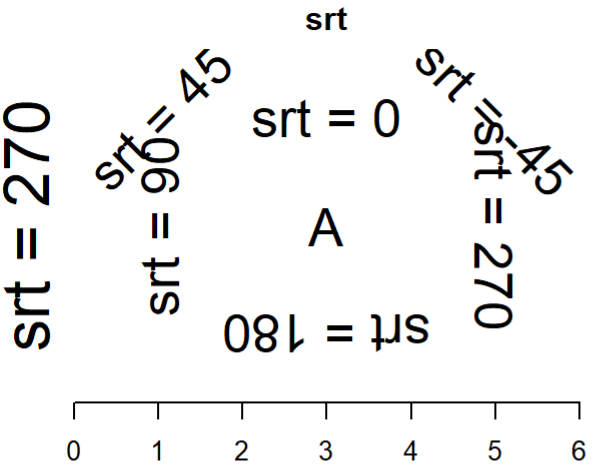

```
par(op)

# 2.4.11 col 인수 : 색상의 선택을 지원하는 인수

# 2.4.12 cex 인수 : 문자나 점의 크기를 설정
op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2), oma = c(0, 0, 2, 0), cex = 1)
plot(0:6, 0:6, type = "n", main = "cex in text")
text(1:3, 1:3, labels = LETTERS[1:3], cex = 1:3)
plot(0:6, 0:6, type = "n", cex = 2, main = "cex in plot")
text(1:3, 1:3, labels = LETTERS[1:3], cex = 1:3)
par(cex = 1.2)
plot(0:6, 0:6, type = "n", main = "cex in par")
text(1:3, 1:3, labels = LETTERS[1:3], cex = 1:3)
plot(0:6, 0:6, type = "n", main = "cex in par")
text(1:3, 1:3, labels = c("가", "나", "다"), cex = 1:3)
points(3:5, 1:3, pch = 1:3, cex = 1:3)
title(main = "cex", line = 0, outer = T)
```



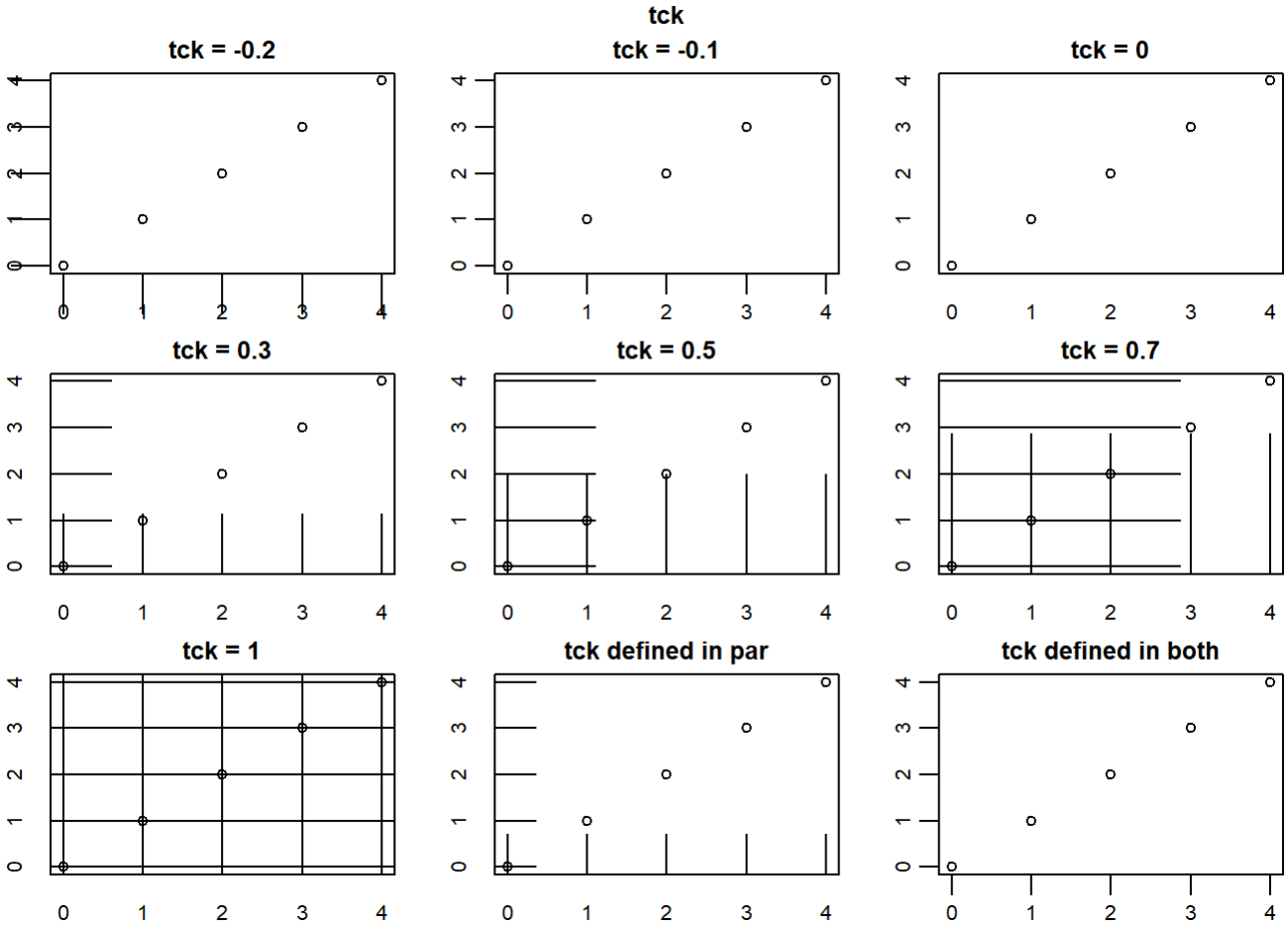

```
par(op)

# 2.4.13 srt 인수 : 문자열을 회전하여 출력할 때 사용

par("srt")
```

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

```
op <- par(no.readonly = TRUE)
par(mar = c(2, 2, 2, 2))
plot(0:6, 0:6, type = "n", axes = F, xlab = "", ylab = "")
text(3, 5, "srt = 0", srt = 0, cex = 2)
text(1, 3, "srt = 90", srt = 90, cex = 2)
text(3, 1, "srt = 180", srt = 180, cex = 2)
text(5, 3, "srt = 270", srt = 270, cex = 2)
text(5, 5, "srt = -45", srt = -45, cex = 2)
text(1, 5, "srt = 45", srt = 45, cex = 2)
points(3, 3, pch = "A", srt = 45, cex = 2)
title("srt", srt = 45)
mtext(side = 2, "srt = 270", srt = 270, cex = 2)
axis(side = 1, srt = 45)
par(op)
```



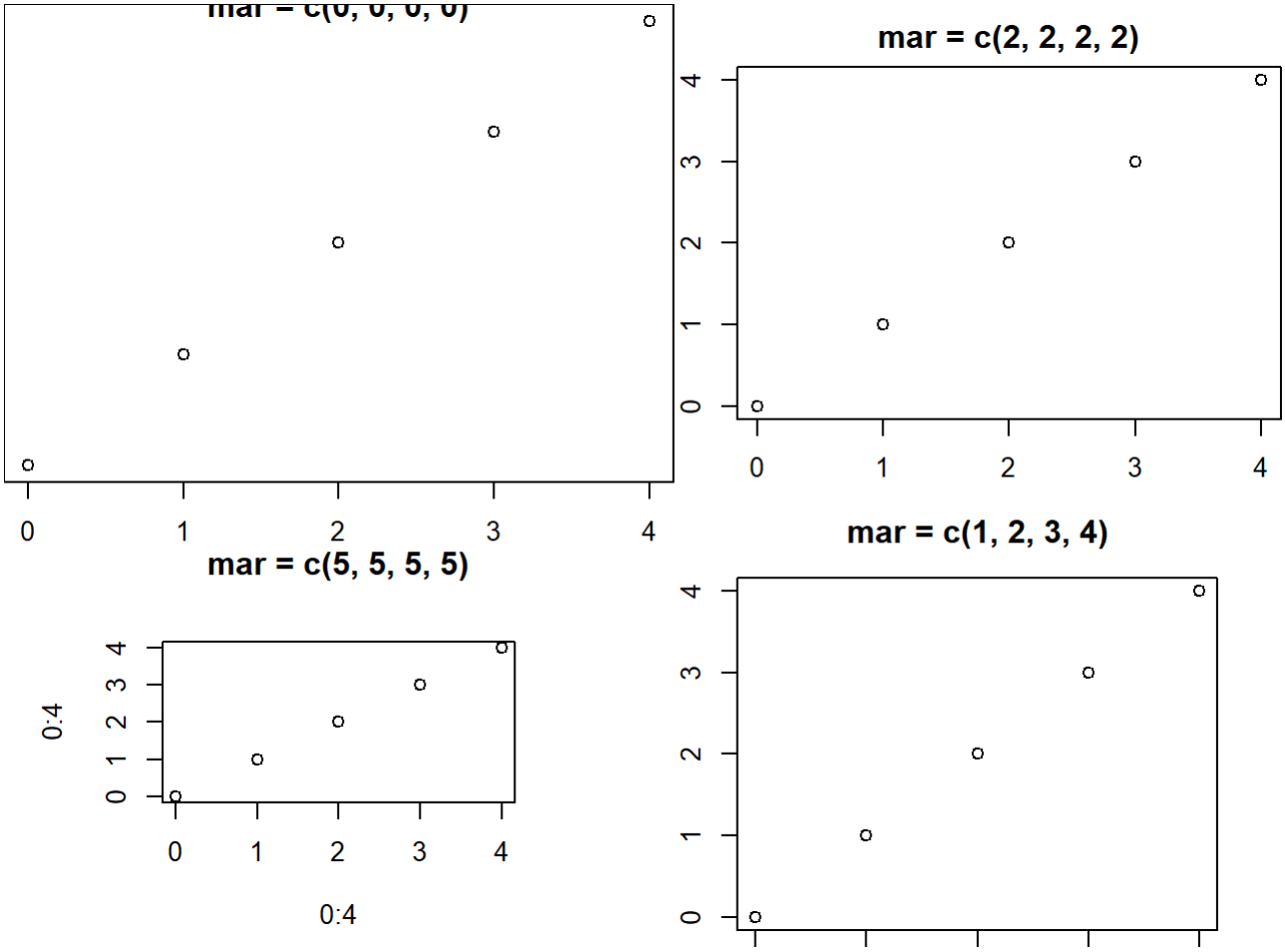

```
# 2.4.14 tck 인수 : 좌표 눈금선인 틱(ticks)의 길이 지정

op <- par(no.readonly = TRUE)
par(mfrow = c(3, 3), oma = c(0, 0, 2, 0), mar = c(2, 2, 2, 2))
plot(0:4, 0:4, tck = -0.2, main = "tck = -0.2")
plot(0:4, 0:4, tck = -0.1, main = "tck = -0.1")
plot(0:4, 0:4, tck = 0, main = "tck = 0")
plot(0:4, 0:4, tck = 0.3, main = "tck = 0.3")
plot(0:4, 0:4, tck = 0.5, main = "tck = 0.5")
plot(0:4, 0:4, tck = 0.7, main = "tck = 0.7")
plot(0:4, 0:4, tck = 1, main = "tck = 1")
par(tck = 0.2)
plot(0:4, 0:4, main = "tck defined in par")
plot(0:4, 0:4, tck = -0.1, main = "tck defined in both")
title(main = "tck", line = 0, outer = T)
```



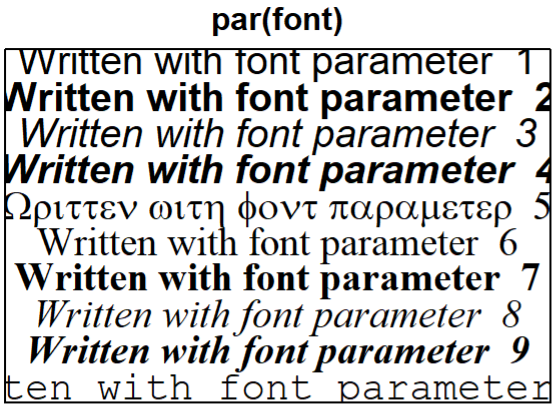

```
par(op)

# 2.4.15 tcl 인수 : 좌표 눈금선의 길이 지정. tck가 플롯 영역의 크기를 기준으로 계산되지만, tcl은 cex=1일 때 문자의 길이를 tcl =1로 정하여 계산

# 2.4.16 mar 인수 : 플롯 영역의 마진을 설

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

```
## [1] 5.1 4.1 4.1 2.1
```

```
# Figure 1
par(mar = c(0, 0, 0, 0))
plot(0:4, 0:4)
title("mar = c(0, 0, 0, 0)")
# Figure 2
par(mar = c(2, 2, 2, 2))
plot(0:4, 0:4, main = "mar = c(2, 2, 2, 2)")
# Figure 3
par(mar = c(5, 5, 5, 5))
plot(0:4, 0:4, main = "mar = c(5, 5, 5, 5)")
# Figure 4
par(mar = c(1, 2, 3, 4))
plot(0:4, 0:4, main = "mar = c(1, 2, 3, 4)")
```



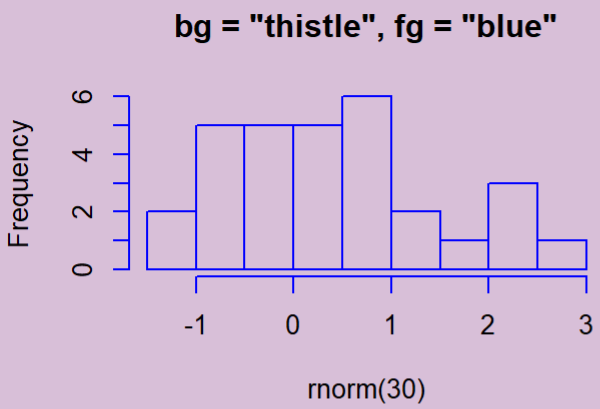

```
par(op)

# 2.4.17 oma 인수 : 바깥 마진의 크기 서정

# 2.4.18 family, font 인수 : 그래픽 장치에서 출력되는 문자의 폰트 종류 설정

op <- par(no.readonly = TRUE)
par(mar = c(2, 2, 2, 2))
plot(1:10, type = "n", main = "par(font)", axes = FALSE, ylab = "",
   xlab = "")
lab <- "Written with font parameter "
for (i in 1:10) {
   par(font = i)
   text(5.5, 11 - i, labels = paste(lab, i), adj = 0.5, cex = 1.5)
 }
box()
par(op)
```



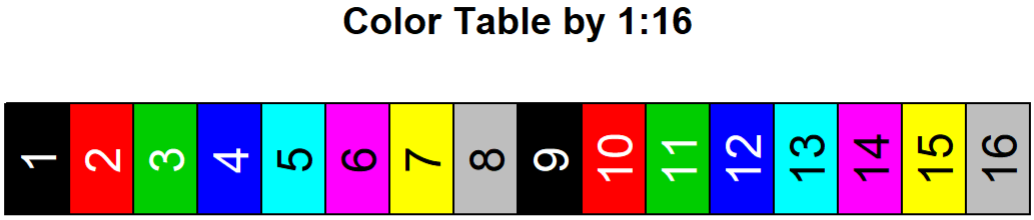

```
# 2.4.19 fg, bg 인수 : 그래픽 장치의 전경색 및 배경색 지

op <- par(no.readonly = TRUE)
# 기본값 조회
unlist(par("fg", "bg"))
```

```
##       fg       bg
## "black" "white"
```

```
par(bg = "thistle", fg = "blue")
hist(rnorm(30), main = "bg = \"thistle\", fg = \"blue\"")
par(op)
```




```
# 2.5 색상 표현하기

# 2.5.2 색상 상수로 표현

col.table <- function(cols, main=NULL, fg=NULL) {
  n <- length(cols)
  plot(seq(n), rep(1, n), xlim = c(0, n), ylim = c(0, 1), type = "n", xlab = "", ylab = "", axes = F)

  main.txt <- if(is.null(main)) paste("Color Table by", deparse(substitute(cols)))
  else main
  title(main=main.txt)

  fg <- if(is.null(fg)) unlist(lapply(cols, function(x)
    ifelse(mean(col2rgb(x)) > 127 | toupper(x) %in% c("WHITE", "#FFFFFF"), "black", "white")))
  else rep(fg, n)

  for(i in 1:n) {
    polygon(c(i - 1, i - 1, i, i), c(0.05, 1, 1, 0.05), col = cols[i])
    text(mean(c(i - 1, i)), 0.52, labels = cols[i], srt=90, adj=0.5, col=fg[i], cex=1.5)
  }
}
op <- par(no.readonly = TRUE)
par(mfrow=c(2,1))
col.table(1:16)
col.table(5:20)
```




```
par(op)


# 2.5.3 색상 이름으로 표현
# 색상 이름을 사용하는 몇 가지의 그래프
cols <- colors()
length(cols)
```

```
## [1] 657
```

```
cols[1:5]
```

```
## [1] "white"         "aliceblue"     "antiquewhite"  "antiquewhite1"
## [5] "antiquewhite2"
```

```
grep("sky", cols, value=TRUE)
```

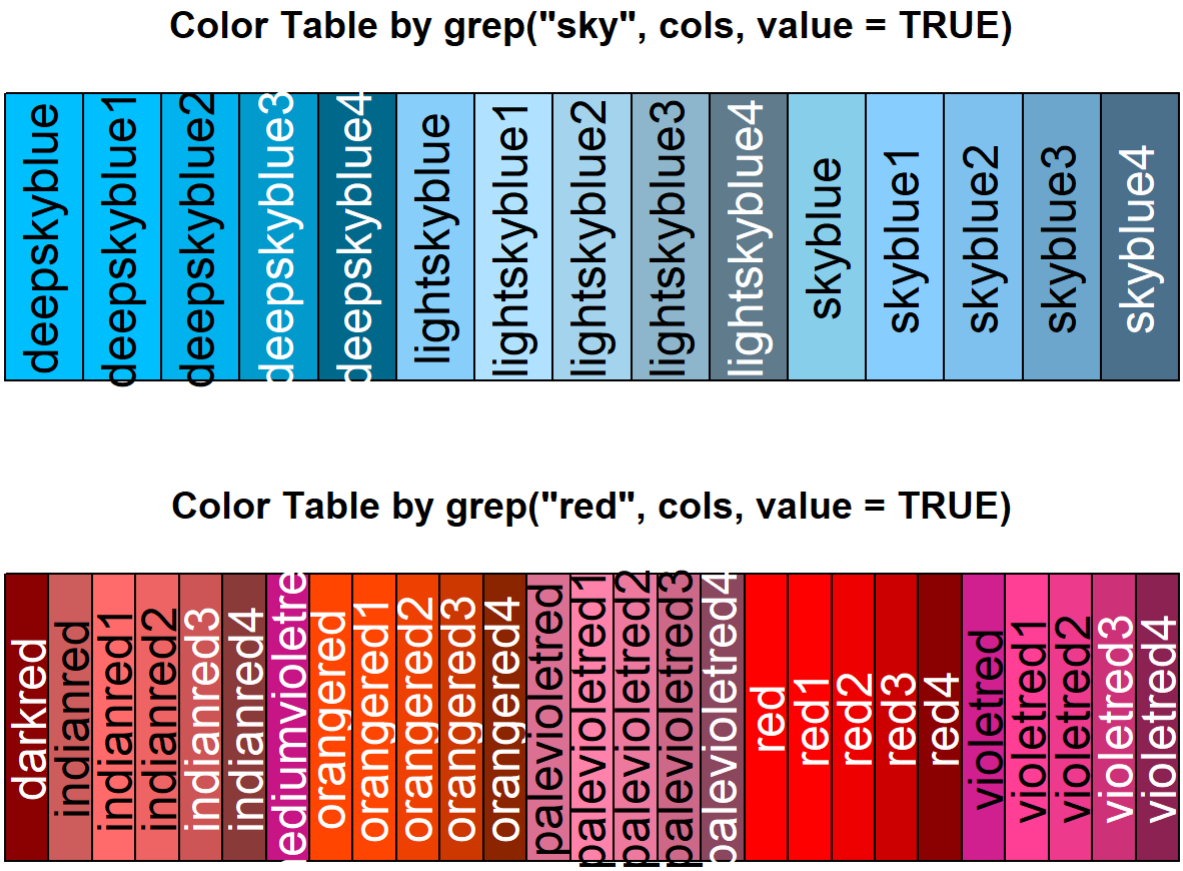
```
##  [1] "deepskyblue"   "deepskyblue1"  "deepskyblue2"  "deepskyblue3"
##  [5] "deepskyblue4"  "lightskyblue"  "lightskyblue1" "lightskyblue2"
##  [9] "lightskyblue3" "lightskyblue4" "skyblue"       "skyblue1"
## [13] "skyblue2"      "skyblue3"      "skyblue4"
```

```
col2rgb(grep("sky", cols, value=TRUE))
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
```

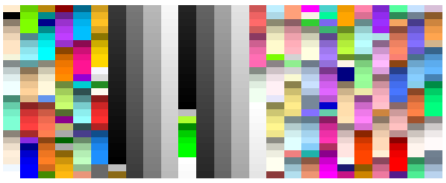
```
## red      0    0    0    0    0  135  176  164  141    96  135  135  126
## green  191  191  178  154  104  206  226  211  182  123  206  206  192
## blue   255  255  238  205  139  250  255  238  205  139  235  255  238
##       [,14] [,15]
## red     108    74
## green   166   112
## blue    205   139
```

```
op <- par(no.readonly = TRUE)
par(mfrow=c(2, 1), mar=c(1, 1, 3, 1))
col.table(grep("sky", cols, value=TRUE))
col.table(grep("red", cols, value=TRUE))
```

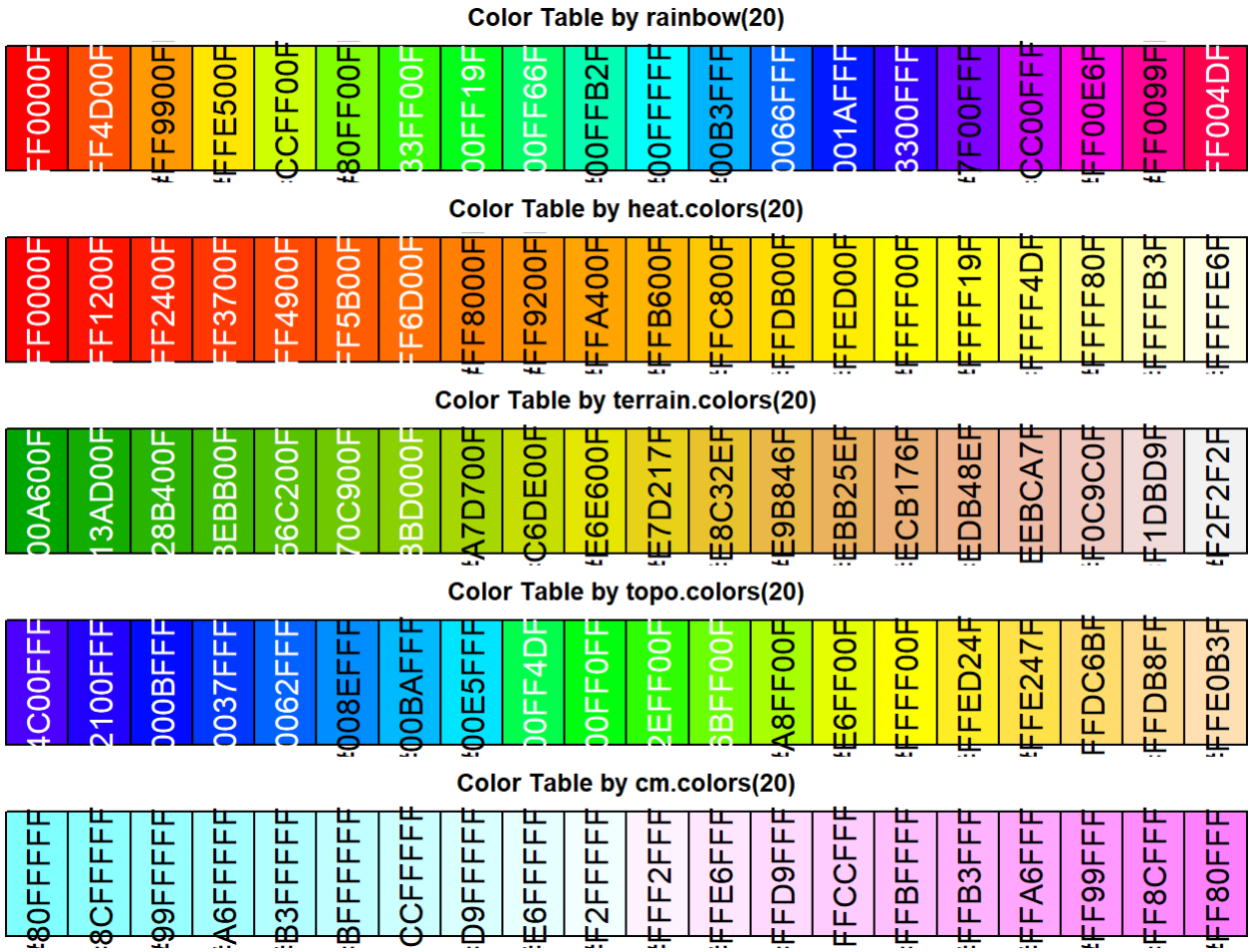



```
par(op)


# 657 개의 색상 이름을 모두 사용한 색상
col.map <- function(cols=colors()) {
   n <- length(cols)
   cnt <- floor(sqrt(n))
   plot.new()
   plot.window(xlim=c(0, cnt), ylim=c(0, cnt))

   for (i in 1:cnt) for (j in 1:cnt)
      rect(i-1, j-1, i, j, col=cols[(i-1)*cnt +j], border=NA)
}
col.map(colors())

# 2.5.4 RGB 색상으로 표현하는 방법
```

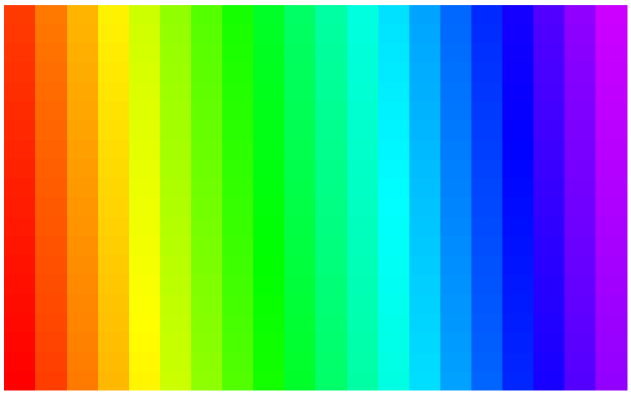
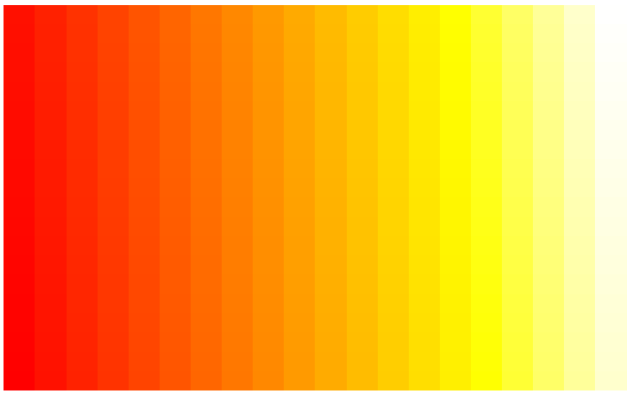
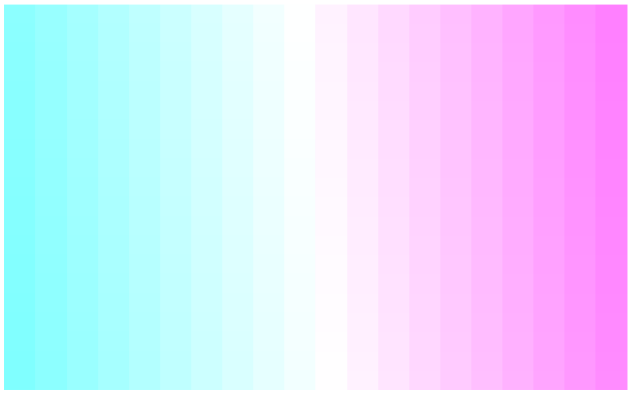
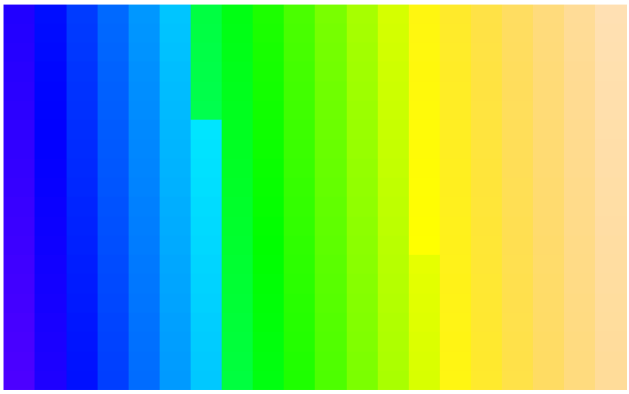
```
seqs <- seq(0, 255, length = 15)
hexs <- toupper(as.character.hexmode(seqs))
red <- paste("#", hexs, "0000", sep = "")
green <- paste("#00", hexs, "00", sep = "")
blue <- paste("#0000", hexs, sep = "")
mix1 <- paste("#", hexs, hexs, hexs, sep = "")
mix2 <- paste("#", rev(hexs), hexs, rev(hexs), sep = "")


# 2.5.5 색상 팔레트

op <- par(no.readonly = TRUE)
par(mfrow = c(5, 1), mar = c(0, 0, 2, 0))
```



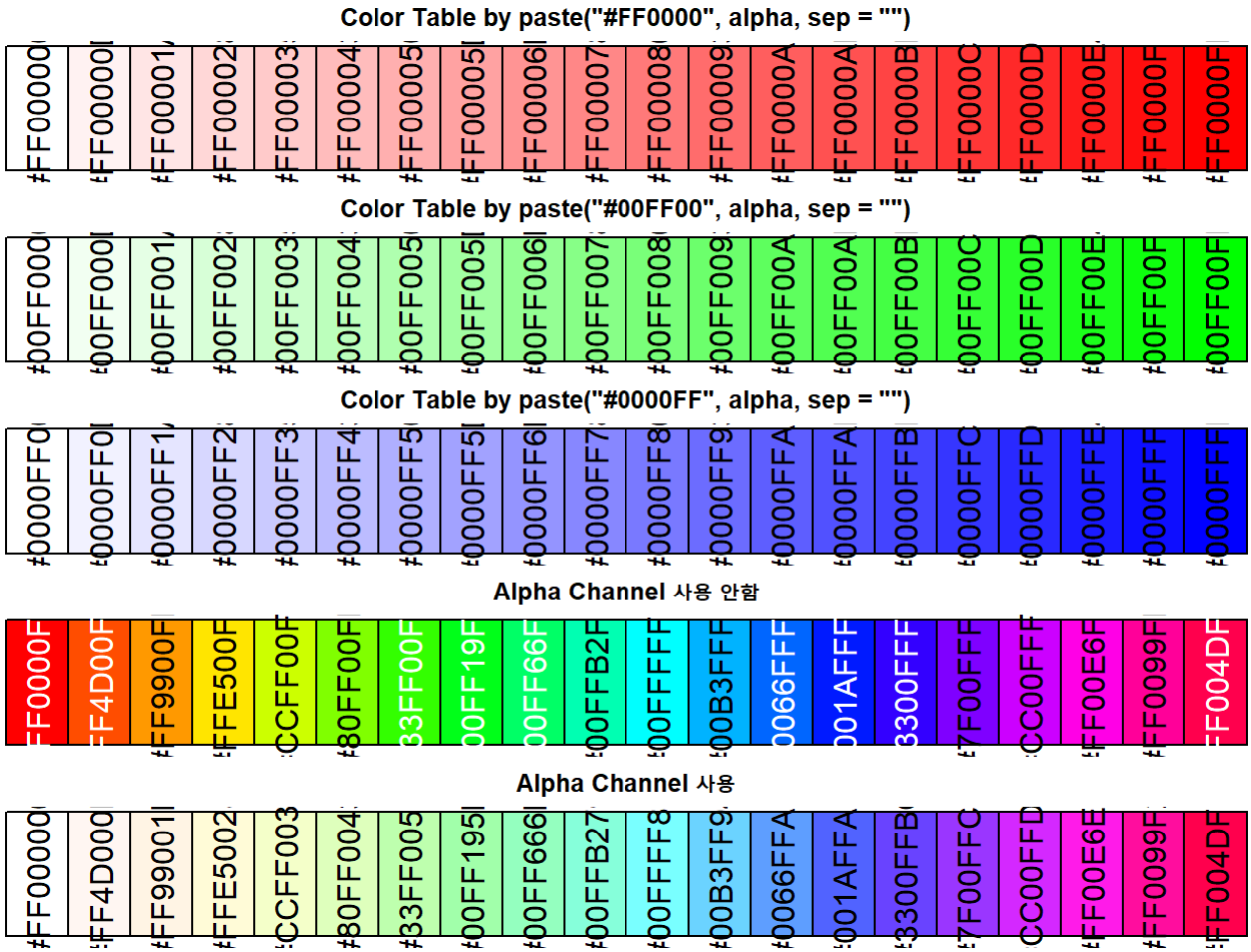
```
col.table(rainbow(20))
col.table(heat.colors(20))
col.table(terrain.colors(20))
col.table(topo.colors(20))
col.table(cm.colors(20))
```



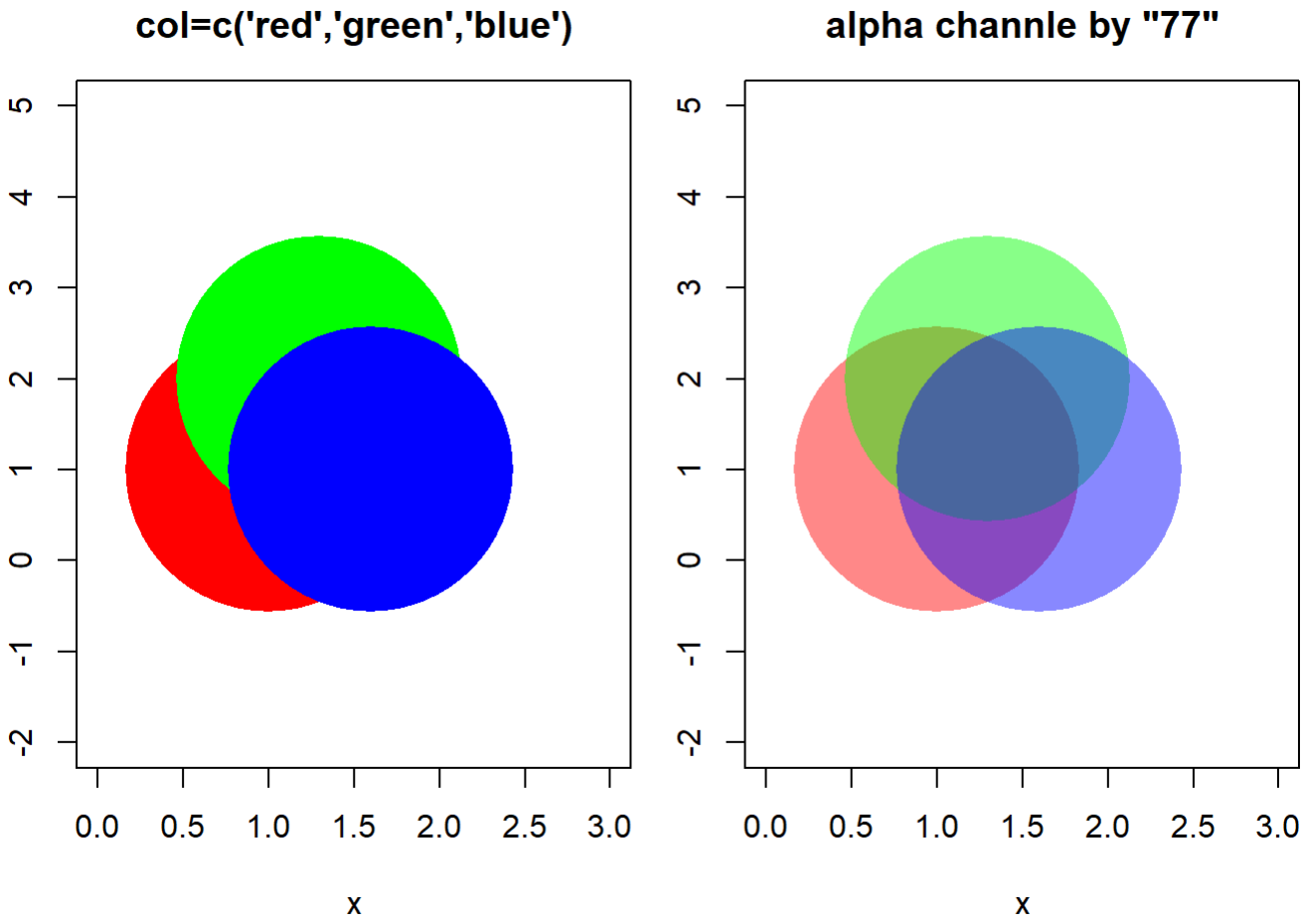

```
par(op)

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2), mar = c(0, 0, 2, 0))
col.map(rainbow(400, start = 0, end = 0.8))
col.map(heat.colors(400))
col.map(cm.colors(400))
col.map(topo.colors(400))
```

```
par(op)

# 2.5.6 알파 채널 : 색상에서 중요한 요소 중 하나. 색상의 투명도를 지정하는 값.

seqs <- seq(0, 255, length = 20)
alpha <- toupper(as.character.hexmode(seqs))
op <- par(no.readonly = TRUE)
par(mfrow = c(5, 1), mar = c(0, 0, 2, 0))
col.table(paste("#FF0000", alpha, sep = ""), fg = 1)
col.table(paste("#00FF00", alpha, sep = ""), fg = 1)
col.table(paste("#0000FF", alpha, sep = ""), fg = 1)
col.table(rainbow(20), main = "Alpha Channel 사용 안함")
col.table(rainbow(20, alpha = seq(0, 1, length = 20)),
          main = "Alpha Channel 사용", fg=1)
```




```
par(op)

op <- par(no.readonly = TRUE)
x <- c(1, 1.3, 1.6)
y <- c(1, 2, 1)
par(mar = c(4, 2, 3, 1), mfrow = c(1, 2))
plot(x, y, pch = 16, cex = 20, col = c("red", "green", "blue"),
     xlim = c(0,3), ylim = c(-2, 5))
title(main = "col=c('red','green','blue')")
plot(x, y, pch = 16, cex = 20, col = c("#FF000077", "#00FF0077", "#0000FF77"),
     xlim = c(0, 3), ylim = c(-2, 5))
title(main = "alpha channle by \"77\"")
```

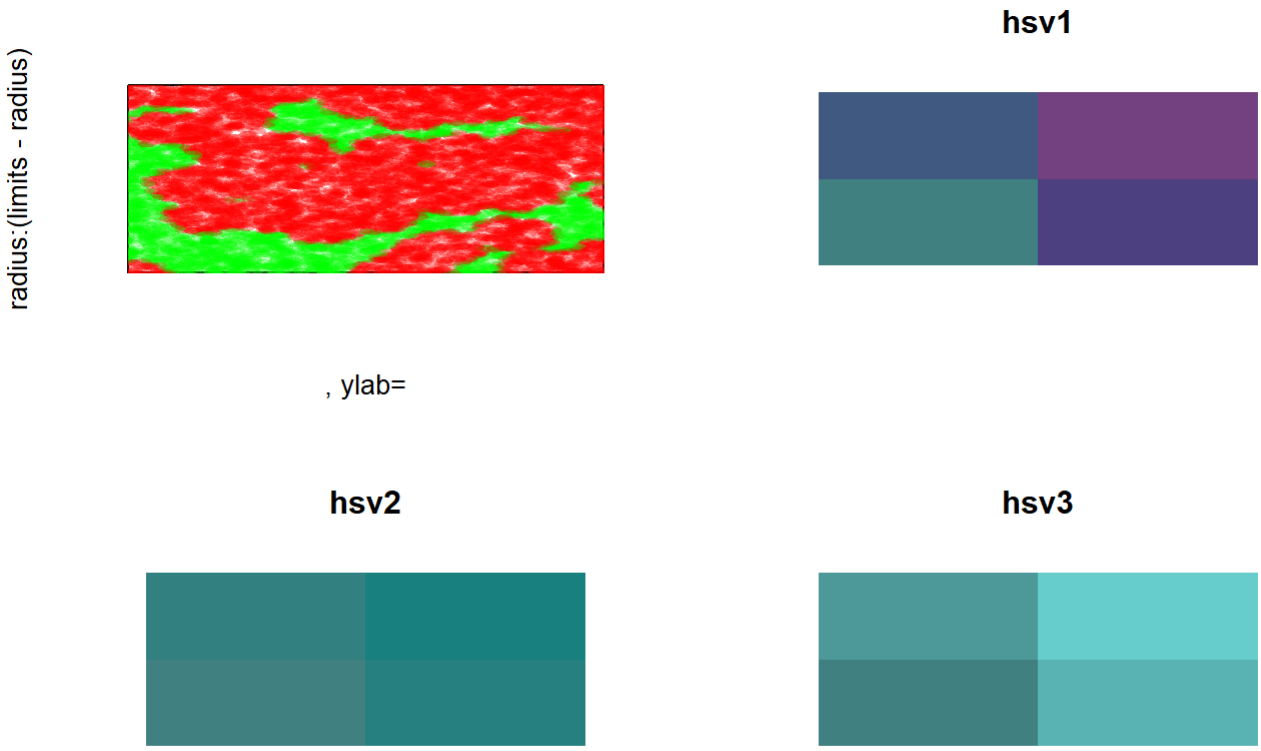



```
par(op)

play.circle <- function(circle.counts=100, limits=3, radius=0.2, densitys=1) {
   circle <- function (x, y, r=1, col=1) {
      angle <- (0:360)*pi/180
      pos.x <- r*cos(angle) + x
      pos.y <- r*sin(angle) + y
      lines(pos.x, pos.y, col=col)
   }

   leaf <- function(limits, xs, ys, radius, r=1, alpha="55") {
      isin <- function(x, y) {
         any(sqrt((xs-x)^2+(ys-y)^2) <= radius)
      }
      x <- runif(1, 0, limits)
      y <- runif(1, 0, limits)
      angle <- (0:360)*pi/180
      pos.x <- r*cos(angle) + x
      pos.y <- r*sin(angle) + y
      polygon(pos.x, pos.y, col=paste(ifelse(isin(x,y), "#FF0000", "#00FF00"),
            alpha, sep=""), border=NA)
   }

   xs <- runif(n=circle.counts, min=0, max=limits)
   ys <- runif(n=circle.counts, min=0, max=limits)
   plot(radius:(limits-radius), radius:(limits-radius), type='n', axes=F, xlab='', ylab='')
   box()
   for (i in 1:circle.counts) {
      circle(xs[i], ys[i], r=radius, col="#FF000011")
```

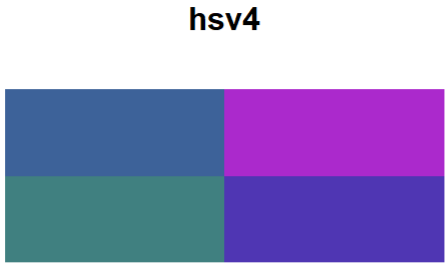
```
  }
  for (i in 1:(circle.counts^2*densitys)) {
    leaf(limits, xs, ys, radius, r=radius/5)
  }
}
play.circle()


# 2.5.7 색상 관련 함수들

# 2.5.7.1 hsv() 함수 : 색의 세가지 속성인 색상, 채도, 명도를 사용해서 색상지정. 이외에 알파 채널 값 지정 가능

hsv(0.5, 0.5, 0.5)
```

```
## [1] "#408080"
```

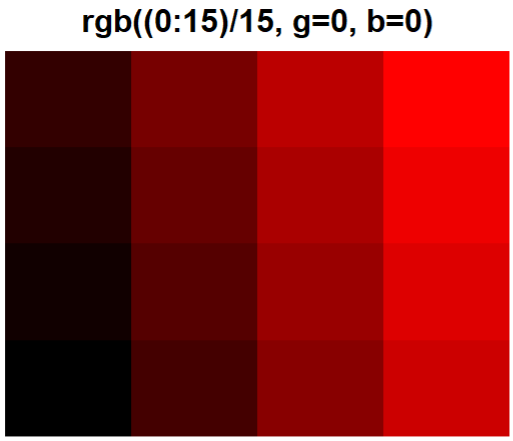
```
hsv1 <- c(hsv(0.5, 0.5, 0.5), hsv(0.6, 0.5, 0.5), hsv(0.7, 0.5, 0.5),
  hsv(0.8, 0.5, 0.5))
hsv2 <- c(hsv(0.5, 0.5, 0.5), hsv(0.5, 0.6, 0.5), hsv(0.5, 0.7, 0.5),
  hsv(0.5, 0.8, 0.5))
hsv3 <- c(hsv(0.5, 0.5, 0.5), hsv(0.5, 0.5, 0.6), hsv(0.5, 0.5, 0.7),
  hsv(0.5, 0.5, 0.8))
hsv4 <- c(hsv(0.5, 0.5, 0.5), hsv(0.6, 0.6, 0.6), hsv(0.7, 0.7, 0.7),
  hsv(0.8, 0.8, 0.8))
op <- par(no.readonly = TRUE)
col.map(hsv1)
title("hsv1")
col.map(hsv2)
title("hsv2")
col.map(hsv3)
title("hsv3")
```

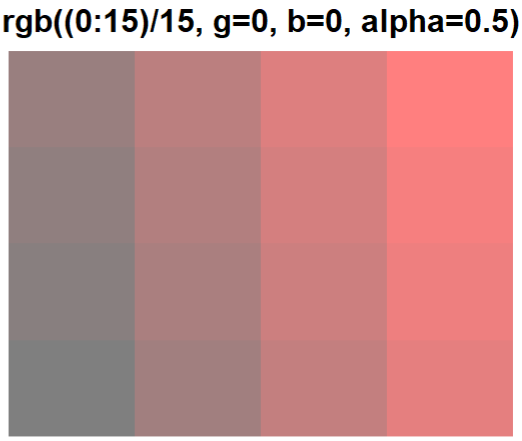


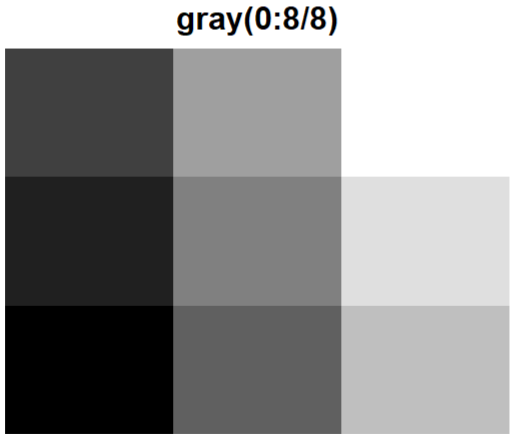

```
col.map(hsv4)
title("hsv4")
par(op)
```

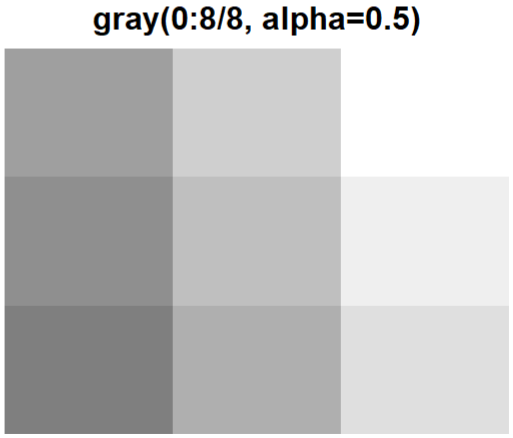



```
# 2.5.7.2  gray() 함수 : 회색 계열이 생상 생성


reds1 <- rgb((0:15)/15, g = 0, b = 0)
reds2 <- rgb((0:15)/15, g = 0, b = 0, alpha = 0.5)
gray1 <- gray(0:8/8)
gray2 <- gray(0:8/8, alpha = 0.5)
op <- par(no.readonly = TRUE)
par(mfrow = c(2, 2), mar = c(1, 3, 1, 1))
col.map(reds1)
title("rgb((0:15)/15, g=0, b=0)")
col.map(reds2)
title("rgb((0:15)/15, g=0, b=0, alpha=0.5)")
col.map(gray1)
title("gray(0:8/8)")
col.map(gray2)
title("gray(0:8/8, alpha=0.5)")
```

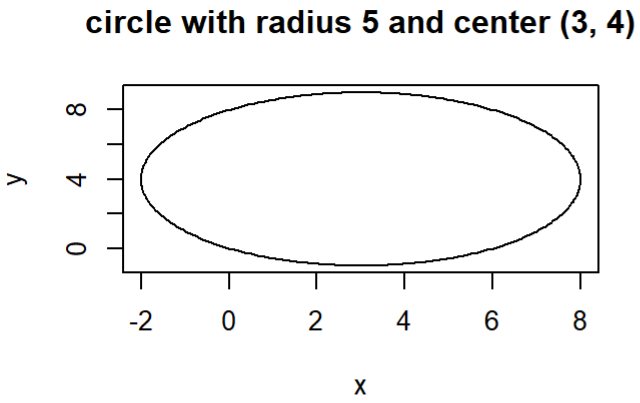




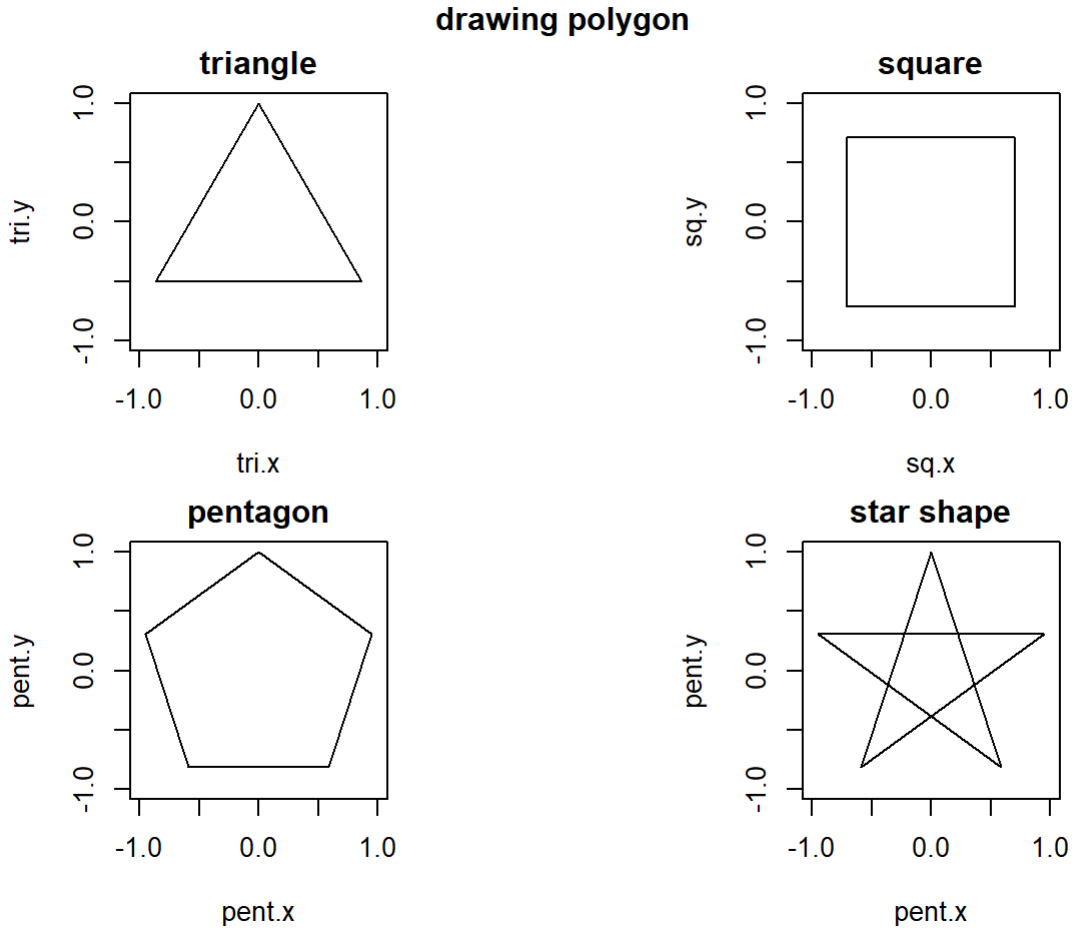





```
par(op)


# 2.6 R을 이용한 도형 그리기

# 2.6.1 원 그리기

op <- par(no.readonly = TRUE)
# 방법 1
angle <- (0:360) * pi/180
# 방법 2
angle <- seq(-pi, pi, length = 361)
x <- 3 + 5 * cos(angle)
y <- 4 + 5 * sin(angle)
plot(x, y, type = "l", main = "circle with radius 5 and center (3, 4)")
par(op)
```




```r
# 2.6.2 다각형 그리기

op <- par(no.readonly = TRUE)
par(oma = c(0, 0, 2, 0), mar = c(4, 2, 2, 0), mfrow = c(2, 2), pty = "s")
# triangle
theta <- seq(pi/2, pi/2 + 2 * pi, by = 2 * pi/3)
tri.x <- cos(theta)
tri.y <- sin(theta)
plot(tri.x, tri.y, type = "l", xlim = c(-1, 1), ylim = c(-1, 1), main = "triangle")
# square
theta <- seq(pi/4, pi/4 + 2 * pi, by = 2 * pi/4)
sq.x <- cos(theta)
sq.y <- sin(theta)
plot(sq.x, sq.y, type = "l", xlim = c(-1, 1), ylim = c(-1, 1), main = "square")
# pentagon
theta <- seq(pi/2, pi/2 + 2 * pi, by = 2 * pi/5)
pent.x <- cos(theta)
pent.y <- sin(theta)
plot(pent.x, pent.y, type = "l", xlim = c(-1, 1), ylim = c(-1, 1),
     main = "pentagon")
# star
s <- seq(length(pent.x))
# line을 순서를 지정하기 위한 벡터
s <- c(s[s%%2 == 1], s[s%%2 == 0])
plot(pent.x, pent.y, type = "n", xlim = c(-1, 1), ylim = c(-1, 1), main = "star shape")
lines(pent.x[s], pent.y[s])
# main title
title(main = "drawing polygon", line = 0, outer = T)
```


drawing polygon
triangle
tri.y
tri.x
square
sq.y
sq.x
pentagon
pent.y
pent.x
star shape
pent.y
pent.x
1.0
0.0
-1.0

```
par(op)
```