R컴퓨팅

13강

R 그래프 II

정보통계학과 장영재 교수

1. 다중 산점도와 matplot, matpoints, matline 함수
2. 산점도 행렬
3. curve 함수
4. 별그림과 레이더 그림 - stars 함수
5. 삼차원 그림 - persp 함수
6. 등고선 그림과 contour 함수
7. 등고선 그림과 filled.contour 함수

1 다중 산점도와 matplot,matpoints,matline 함수

1 다중 산점도와 matplot,matpoints,matline 함수

- matplot 함수는 주어진 하나의 x 벡터값에 대해서 두 개 이상의 y벡터에 대한 산점도를 그리는 함수

```
matplot(x, y, type = "p", lty = 1:5, lwd = 1, lend =
par("lend"),
pch = NULL, col = 1:6, cex = NULL, bg = NA,
xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
add = FALSE, ...)
```

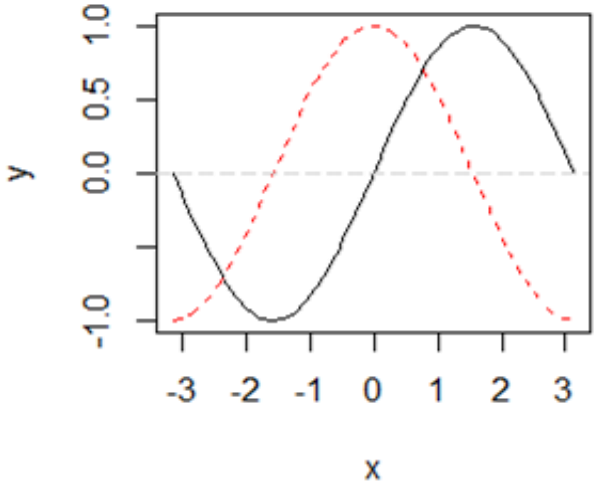
1 다중 산점도와 matplot,matpoints,matline 함수

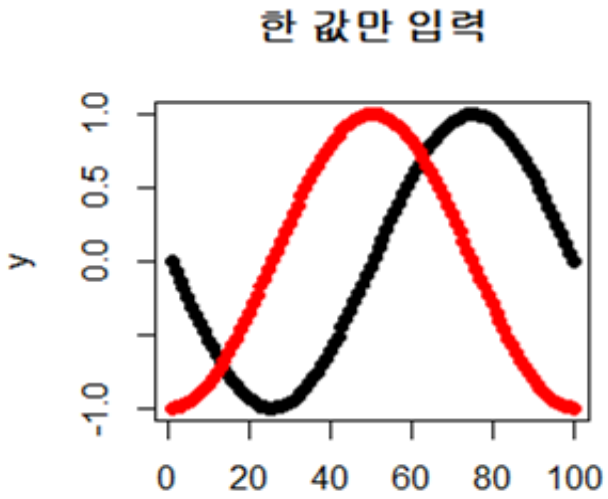
보기 13-1: 산점도와 회귀직선을 arrow 및 segmemt 함수를 사용하기

```
> matplot.test <- function() {
+ par(mfrow=c(1,2))
+ x <- seq(-pi,pi, length=100);   # -π부터 π사이의 100개의 값을 벡터로
+ y <- cbind(sin(x), cos(x))   # x에서 sin 및 cos 함수의 값 계산
+ matplot(x, y, type="l")   # x에서 sin과 cos함수 그리기
+ abline(h=0, lty=5, col="lightgray") # x축 그리기
+ matplot(y, type="p", pch=16, main="한 값만 입력")
+ # y만 입력된 경우 x는 (0, 100)
+ }        # end function
```

1 다중 산점도와 matplot,matpoints,matline 함수

※ matplot 함수를 사용한 겹쳐 그리기





2 산점도 행렬

2 산점도 행렬

- matplot 함수는 주어진 하나의 x 벡터값에 대해서 두 개 이상의 y벡터에 대한 산점도를 그리는 함수

```
pairs(formula, data = NULL, ..., subset, na.action =
stats::na.pass)
```

또는

```
pairs(formula, data = NULL, ..., subset, na.action =
stats::na.pass)
```

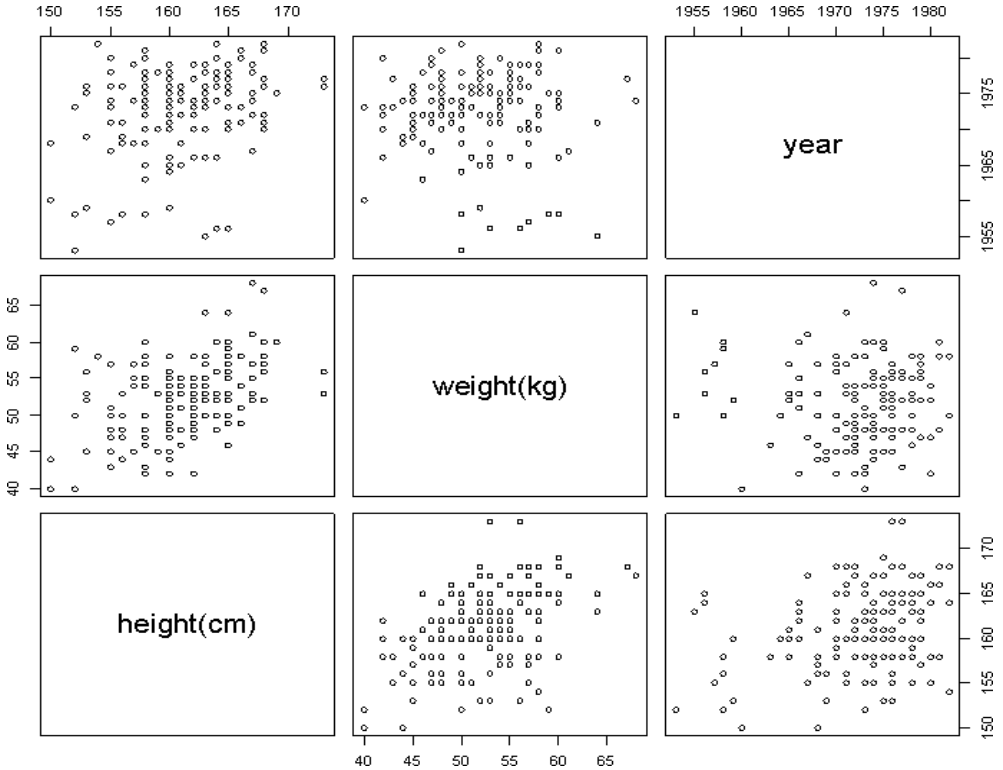
2 산점도 행렬

- 보기 13-2: 키 몸무게 자료에 BMI에서 키, 몸무게, 출생연도에 대한 모든 가능한 산점도를 그리기(pairs(~height+weight+year, data=BMI)이용)

```
> pairs.test <- function() {
+    pairs(~height+weight+year, data=BMI, subset=gender == "F",
+        labels=c( "height(cm)",  "weight(kg)",  "year"),  row1attop=F )
+ }              # end function
```

2 산점도 행렬

※ 산점도 행렬



3 curve 함수

3 curve 함수

- 함수를 그리기 위해서 plot 함수의 매개변수 type에 "l"을 설정하여 그릴 수 있지만 좀더 간단하게 그릴 수 있는 함수인 curve 함수를 R-언어에서 제공

```
curve(expr, from = NULL, to = NULL, n = 101, add =
FALSE, type = "l", xname = "x", xlab = xname, ylab =
NULL, xlim = NULL, ...)
```

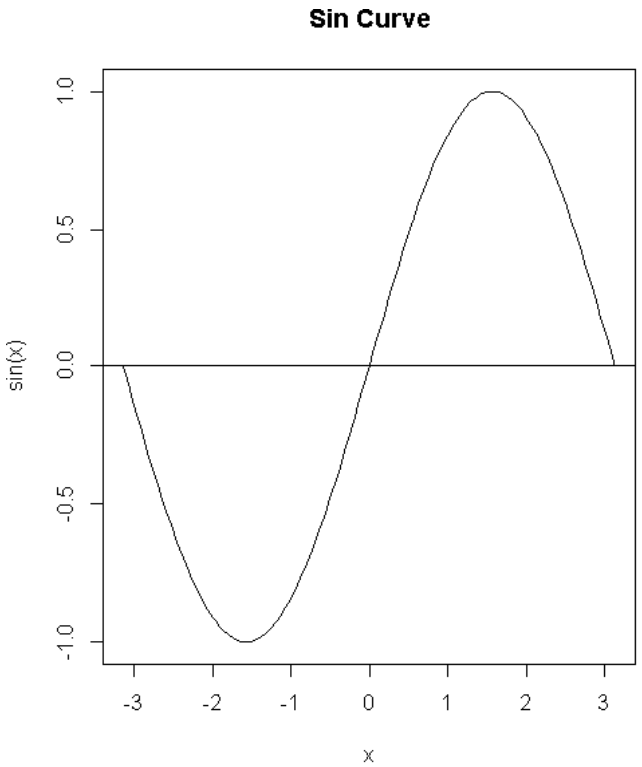
3 curve 함수

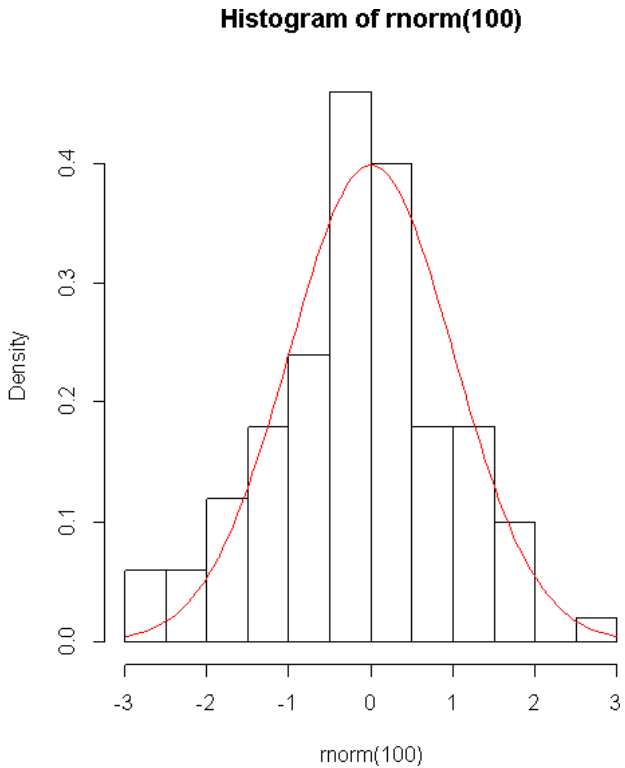
보기 13-3: curve 함수로 함수 그리기 및 기존그림에 추가하기

```
> curve.test <- function() {
+    par(mfrow=c(1,2))
+    curve(sin(x), xlim=c(-pi, pi), main="Sin Curve")
+    abline(h=0)
+    hist(rnorm(100), freq=F) # 히스토그램을 먼저 그리고
+    curve(dnorm(y), from=-3, to=3, add=T, col="red", xnam="y")
+                  # curve 함수 추가
+ }                      # end function
```

3 curve 함수

※ curve 함수를 사용한 그림





4 별그림과 레이더 그림 - stars 함수

4 별그림과 레이더 그림 - stars 함수

- 별그림(star plot 또는 star diagram)은 행렬 등의 형식으로 주어진 자료의 각 행의 값의 크기를 별모양으로 만들어 각 행의 원소값의 크기가 어느 정도인지 그림으로 비교할 수 있도록 제시

```
stars(x, full = TRUE, scale = TRUE, radius = TRUE,
labels = dimnames(x)[[1]], locations = NULL, nrow =
NULL, ncol = NULL, len = 1, key.loc = NULL, key.labels
= dimnames(x)[[2]], key.xpd = TRUE, xlim = NULL, ylim
= NULL, flip.labels = NULL, raw.segments = FALSE,
col.segments = 1:n.seg, col.stars = NA, col.lines = NA,
axes = FALSE, main = NULL, sub = NULL, xlab = "",
ylab = "", cex = 0.8, lwd = 0.25, lty = par("lty"), add =
FALSE, ...)
```

4 별그림과 레이더 그림 - stars 함수

보기 13-4: scale 과 radius의 설정이 True/False일 때 별 그림의 비교

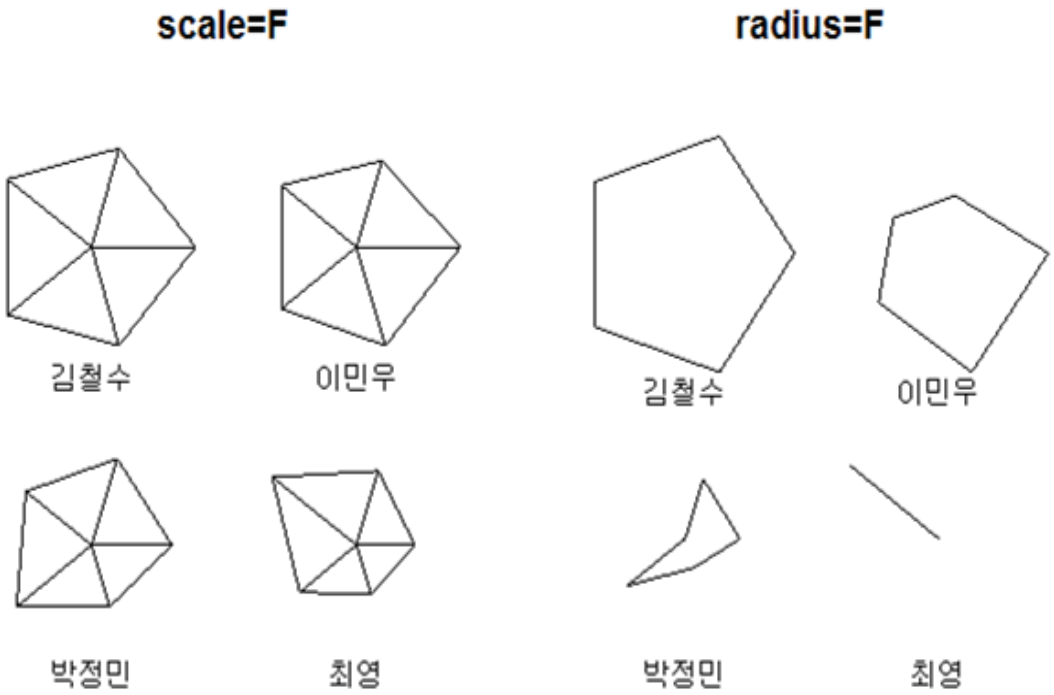
```
> stars.test2 <- function() {
+   X <-matrix( c(90, 80, 90, 90, 80,   90, 70, 80, 80, 80,
+                 70, 70, 70, 80, 50,   50, 60, 90, 60, 40),
+               ncol=5, byrow=T) /100
+   dimnames(X) <- list(c("김철수", "이민우", "박정민", "최영"),
+                       c("통계학", "전산학", "경영학", "영어", "체육"))
+   par(mfrow=c(1,2))
+   stars(X, scale=F, main="scale=F")
+   stars(X, radius=F, main="radius=F")
+ }          # end function
```

4 별그림과 레이더 그림 - stars 함수

```
> X
       통계학 전산학 경영학 영어 체육
김철수     0.9     0.8     0.9  0.9  0.8
이민우     0.9     0.7     0.8  0.8  0.8
박정민     0.7     0.7     0.7  0.8  0.5
최영       0.5     0.6     0.9  0.6  0.4
```

4 별그림과 레이더 그림 - stars 함수

※ stars 함수에서 scale과 radius 속성 설정에 따른 차이



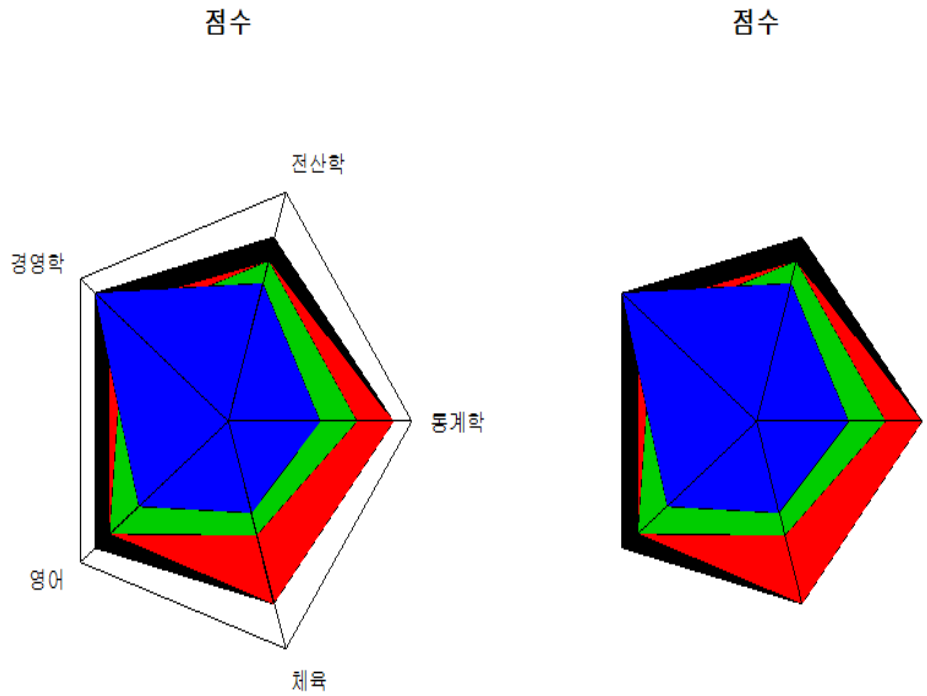
4 별그림과 레이더 그림 - stars 함수

- 보기 13-5: locations를 같은 위치로 설정하여 모든 별 그림이 같은 위치에 있도록 하여 생성한 레이다 그림의 예

```
> stars.test3 <- function() {
+   X <-matrix( c(90, 80, 90, 90, 80,   90, 70, 80, 80, 80,
+                 70, 70, 70, 80, 50,   50, 60, 90, 60, 40),
+               ncol=5, byrow=T) /100
+    dimnames(X) <- list(c("김철수", "이민우", "박정민", "최영"),
+                        c("통계학", "전산학", "경영학", "영어", "체육"))
+   par(mfrow=c(1,2))
+   stars(X, locations = c(0, 0), scale = FALSE, radius  =  T,
+       col.stars = 1:10, key.loc = c(0, 0), main = "점수",
+ key.xpd=T,  xlim=c(-1.2,1.2))
+    stars(X, locations = c(0, 0), scale = FALSE, radius  =  T,
+        col.stars = 1:10, xlim=c(-1.2,1.2), main = "점수")
+ } # end function
```

4 별그림과 레이더 그림 - stars 함수

※ col.stars와 key.loc 속성의 설정에 따른 stars 함수의 결과



4 별그림과 레이더 그림 - stars 함수

- 보기 13-6: 별이 아닌 원으로 표시하기 및 각 별그림의 위치 및 key.loc의 위치 수동 설정 (draw.segments 값에 True를 설정)

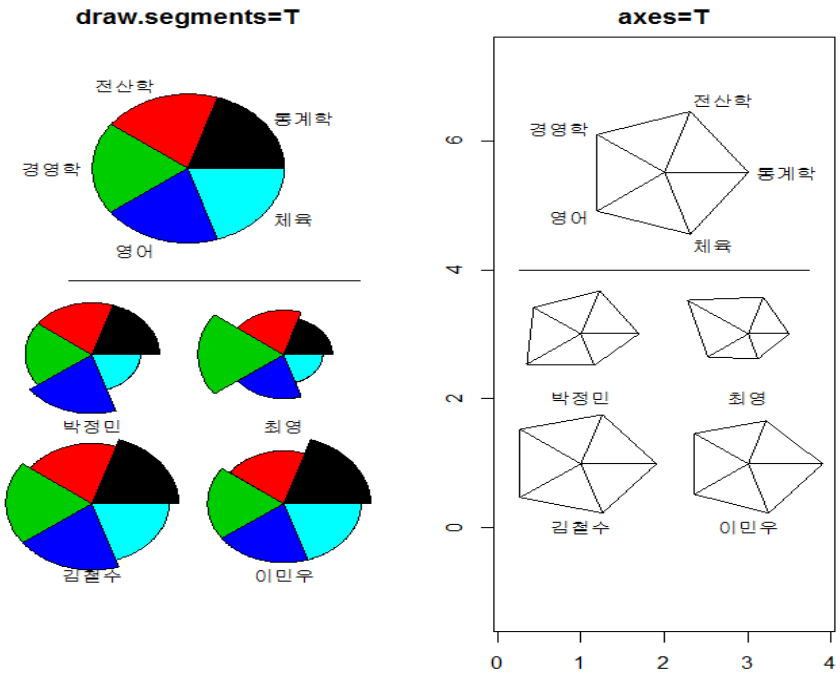
```
> stars.test4 <- function() {
+   X <-matrix( c(90, 80, 90, 90, 80,   90, 70, 80, 80, 80,
+                 70, 70, 70, 80, 50,   50, 60, 90, 60, 40),
+               ncol=5, byrow=T) /100
+   dimnames(X) <- list(c("김철수", "이민우", "박정민", "최영"),
+                       c("통계학", "전산학", "경영학", "영어", "체육"))
+   par(mfrow=c(1,2))
+   myloc <- rbind(c(1,1), c(3,1), c(1,3), c(3,3))
```

4 별그림과 레이더 그림 - stars 함수

```
+ stars(X, draw.segments=T, location = myloc, key.loc =
+ c(2, 5.5), scale=F, main="draw.segments=T", ylim=c(0,6))
+ abline(h=4)            # 분리선 그리기
+     stars(X, axes=T, location=myloc, key.loc = c(2, 5.5),
+ scale=F, ylim=c(0,6), main="axes=T")
+     abline(h=4)                # 분리선 그리기
+ }                    # end function
```

4 별그림과 레이더 그림 - stars 함수

※ draw.segments, axes, location 및 key.loc를 다르게 설정한 경우에 따른 stars 함수의 결과



5 삼차원 그림 - persp 함수

5 삼차원 그림 - persp 함수

- persp 함수는 평면의 좌표 (x,y)에서 높이가 z인 삼차원 함수를 그림으로 표현하는 함수

```
persp(x = seq(0, 1, length.out = nrow(z)), y = seq(0, 1,
length.out = ncol(z)), z, xlab = NULL, ylab = NULL, zlab
= NULL, main = NULL, sub = NULL, theta = 0, phi = 15,
col = "white", border = NULL, box = TRUE, scale =
TRUE ...)
```

- x, y: 평면에서의 x와 y의 좌표값. 이들은 오름차순으로 정렬되어 있어야 하며 기본값은 0부터 1사이의 값을 z의 행의 수와 열의 수만큼 같은 간격으로 나눈 값
- z: (x[i], y[j])에서의 높이가 z[i,j]에 저장된 행렬

5 삼차원 그림 - persp 함수

- 보기 13-7: 상관계수 r을 매개변수로 받아서(기본값 0) x, y의 범위가 모두 (-3, 3)사이에서의 이변량 정규분포함수

$$f(x,y) = \frac{1}{2\pi\sqrt{(1-r^{2)}}} exp\left\{-\frac{(x^2-2rxy+y^2)}{2(1-r^2)}\right\}$$

를 계산하여 그 결과를 z로 얻는 함수

```
> binormalpdf <- function(r=0) {
+ x <- seq(-3,3,length=30)
+ y <- x
+ z <- matrix(0,ncol=length(x),nrow=length(y))
+ for (i in 1:length(x)) {
+ for (j in 1:length(y)) {
```

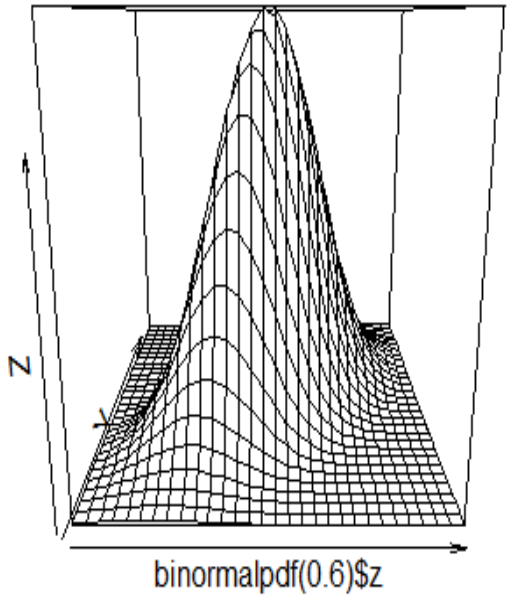
5 삼차원 그림 - persp 함수

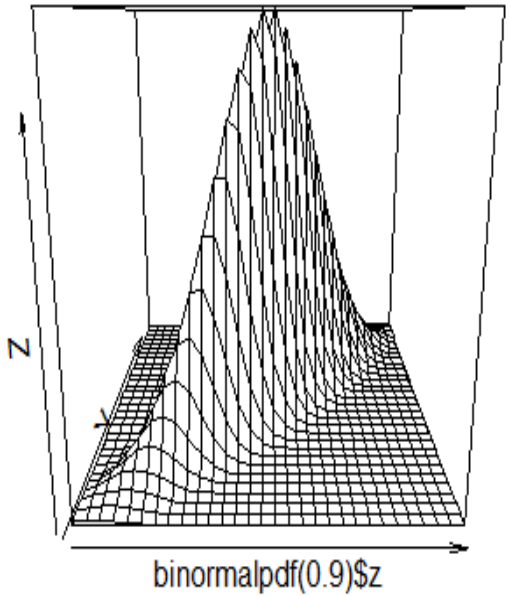
```
+ z[i,j] <- exp(-(x[i]^2-2*r*x[i]*y[j] + y[j]^2)/(2*(1-r^2)))
+ }} # end for
+ z <- z/(2*pi*sqrt(1-r^2))
+ list(x=x, y=y, z=z)
+ }       # end function
> persp.test <- function() {
+ par(mfrow=c(1,2))
+ persp(binormalpdf(0.6)$z)
+ persp(binormalpdf(0.9)$z)
+ }
```

- 이 함수의 결과를 이용하여 이변량 정규분포의 확률밀도함수를 persp.test ()를 사용하여 그린 그림은 다음과 같음

5 삼차원 그림 - persp 함수

※ 상관계수 0.6 및 0.9인 경우 이변량 정규분포의 확률밀도함수–persp 함수





6 등고선 그림과 contour 함수

6 등고선 그림과 contour 함수

```
contour(x = seq(0, 1, length.out = nrow(z)),
y = seq(0, 1, length.out = ncol(z)),
z,
nlevels = 10, levels = pretty(zlim, nlevels),
labels = NULL,
xlim = range(x, finite = TRUE),
ylim = range(y, finite = TRUE),
zlim = range(z, finite = TRUE),
labcex = 0.6, col = par("fg"), ...)
```

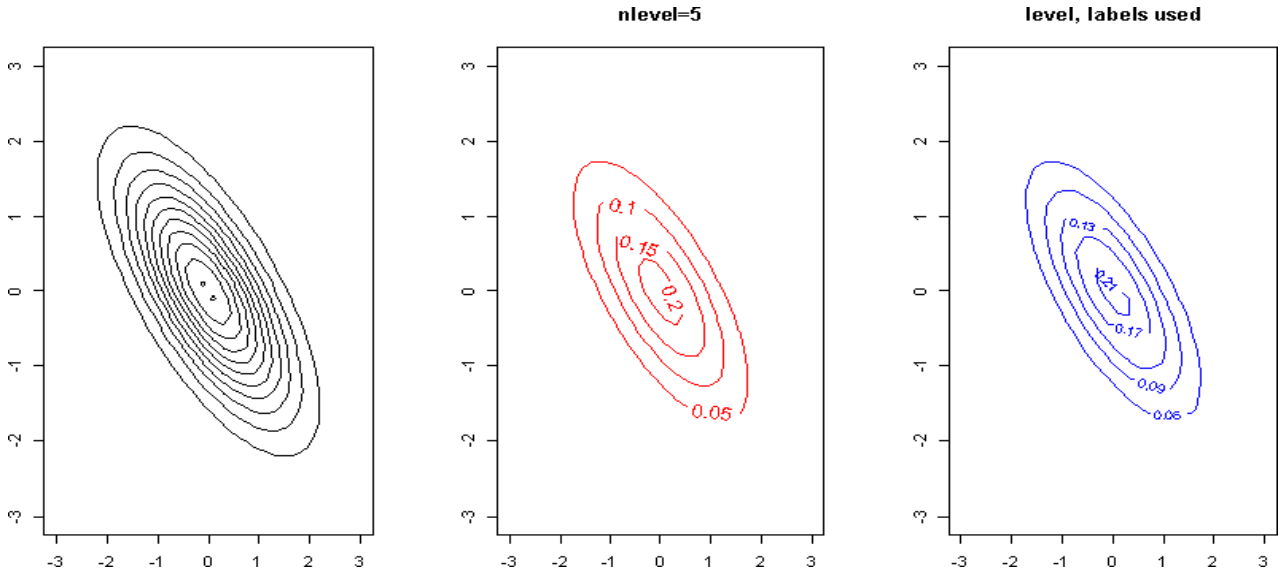
6 등고선 그림과 contour 함수

- 보기 13-8: 이변량 정규분포 함수 binormalpdf 함수를 사용하여 등고선 그리기 (binormalpdf 함수는 persp.test.r에 포함되어 있음)

```
> contour.test <- function() {
+ # NOTE: binormalpdf 함수가 필요함.
+   x <- binormalpdf(.7)$x
+   y <- x
+   z <- binormalpdf(0.7)$z
+   par(mfrow=c(1,3))
+   contour(x, y, z, drawlabels=F)
+   contour(x, y, z, nlevel=5, main="nlevel=5", col=2, labcex=.8)
+   contour(x, y, z, level=seq(0.05, 0.25, by=0.04),
```

6 등고선 그림과 contour 함수

```
+ labels=as.character(seq(0.05, 0.25, by=0.04)),
+      main="level, labels used", col=4)
+ }      # end function
```



7 등고선 그림과 filled.contour 함수

7 등고선 그림과 filled.contour 함수

- filled.contour 함수는 그림영역을 두 개로 나누어 왼쪽은 색을 채운 등고선 그림을, 오른쪽엔 범례를 만드는 방법의 그림을 제공

```
filled.contour(x = seq(0, 1, length.out = nrow(z)),
y = seq(0, 1, length.out = ncol(z)),
z,
col = color.palette(length(levels) – 1), ...)
```

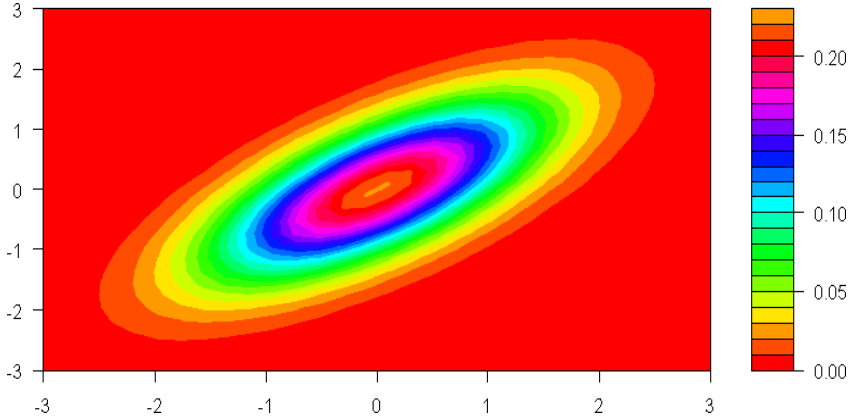
- 보기 13-9: 색으로 채운 등고선 그림

```
〉 contour.test 〈– function() {

〉 filled.contour.test 〈– function() {
+ # NOTE: binormalpdf 함수가 필요함. 이 함수는
persp.test.r에 포함되어 있음
```

7 등고선 그림과 filled.contour 함수

```
+ x <- seq(-3,3,length=30)
+ y <- x
+  filled.contour(x,y, binormalpdf(0.7)$z, col=rainbow(20))
+ }          # end function
> filled.contour.test()
```



R컴퓨팅