```r
# Plotting a heatmap: Graphical representation of data to visualize different values.
# To plot a heatmap of a numeric matrix, use image()
# heat.colors(), topo.colors(), etc: create a color vector
# contour(): draw contours, or add to existing plot
# curve(): draw a curve, or add to existing plot

mat = 1:5 %*% t(6:10)
image(mat) # Red means low, white means high

# The orientation of image() is to plot the heatmap according to the following order, in terms of the matrix elements:

# (1,ncol)  (2,ncol) ... (nrow,ncol)
#
# (1,2)     (2,2)    ...  (nrow,2)
# (1,1)     (2,1)    ...  (nrow,1)

a <- seq(0,1, length=5)     # row index
a_mat <- matrix(rep(a,length(a)),length(a),length(a))    # column index
text(a_mat,t(a_mat), mat)

# Color scaled
# The default is to use a red-to-white color scale in image(). But the col argument can take any vector of colors.
# Built-in functions gray.colors(), rainbow(), heat.colors(), topo.colors(), terrain.colors(), cm.colors() all return
# continguous color vectors of given length

phi = dnorm(seq(-2,6,length=50)) + dnorm(seq(-2,6,length=50), mean=4, sd=1)  #dnorm: normal density
plot(phi, type="l")
normal.mat = phi %*% t(phi)
image(normal.mat) # Default is col=heat.colors(12)

image(normal.mat, col=heat.colors(50)) # More colors
?heat.colors
image(normal.mat, col=heat.colors(3)) # less colors

image(normal.mat, col=terrain.colors(12)) # Terrain colors
image(normal.mat, col=topo.colors(12)) # Topological colors

# Drawing contour lines

# To draw contour lines from a numeric matrix, use contour(); to add contours to an existing plot (like, a heatmap),
# use contour() with add=TRUE

contour(normal.mat, col = 1:3)

image(normal.mat, col=terrain.colors(20))
contour(normal.mat, add=TRUE)

# Drawing a curve
# To draw a curve of a function, use curve()

curve(x^3) # Default is to plot between 0 and 1.
curve(x^3, from=-3, to=3, lwd=3, col="red") # More plotting options

n = 100
set.seed(0)
x = sort(runif(n, min=-2, max=2))
y = x^3 + rnorm(n)

plot(x, y)
curve(x^3, lwd=3, col="red", add=TRUE)
z <- loess(y~x)
lines(x,fitted(z), lwd=3, lty=2, col="blue")
legend("bottomright", c("true", "local"), horiz=TRUE, col=c("red", "blue"), lwd=3, lty=c(1,2))
```

```
64
65  x <- matrix(rep(seq(-10, 10, length= 30),30),30,30)
66  y <- t(x)
67  f <- function(x, y) { r <- sqrt(x^2+y^2); return(10 * sin(r)/r) }
68  # f <- function(x, y) { r <- sqrt((100-x^2-y^2)*(100-x^2+-y^2>0)) }
69  z <- f(x,y)
70  # z[is.na(z)] <- 1
71  persp(x[,1], y[1,], z, theta = 50, phi = 0, expand = 2, col = "lightblue",
    xlim=c(-10,10), ylim=c(-10,10), xlab="x",
72        ylab="y", zlab="z")
73  # theta: left, right rotation, phi: top down rotation
74  # expand: expansion factor applied to the z coordinates. Often used with 0 < expand < 1
75  # to shrink the plotting box in the z direction.
76
77  z <- x^3 + y^3
78  surface <- persp(x[,1], y[1,], z, theta = 80, phi = 30, expand = 1, col =
    terrain.colors(30))
79  ?terrain.colors
80
81  n = 200
82  x1 = runif(n, -10, 10)
83  y1 = runif(n, -10, 10)
84  z1 = x1^3 + y1^3 + 10*rnorm(n)
85
86  xy.list = trans3d(x1, y1, z1, surface)
87  points(xy.list, pch=20, col=heat.colors(10))
88
89  # Draw a map
90
91  install.packages("maps")
92  # install.packages("ggmap")
93  # install.packages("DeducerSpatial")
94
95  library(maps)
96  # library(ggmap)
97  data()
98  state.fips
99
100 map("usa", cex=1.5)
101
102 map("county")
103
104 map("state", "GEORGIA")
105 us.cities # data included in maps
106 map.cities(us.cities, country = "GA")
107
108 # Plot the unemployment rate in each county
109
110 unemp
111 county.fips
112
113 # Plot unemployment by country
114 colors = c("slateblue", "slateblue1", "slateblue2", "slateblue3", "slateblue4")
115 head(unemp)
116 head(county.fips)
117
118 unemp$colorBuckets <- as.numeric(cut(unemp$unemp, c(0, 2.5, 5, 7.5, 10, 100))) #
    distretize unempl
119 # match returns a vector of the positions of (first) matches of its first argument in
    its second.
120 a <- c(2,3,5,1,3)
121 b <- 0:4
122 match(a,b)
123 colorsmatched <- unemp$colorBuckets[match(county.fips$fips, unemp$fips)]   # county -
    fips - uncomployment rate
124
125 install.packages("mapproj")
126
127 map("county", col = colors[colorsmatched], fill = T, resolution = 0,
```

```
    lty = 0, projection = "polyconic")    # projection specifies the shape of the map.
    Also try "lagrange" and "aitoff"

# library(mapproj)
# Add border around each State
map("state", col = "white", fill = FALSE, add = TRUE, lty = 1, lwd = 0.2,
    projection = "polyconic")
title("unemployment rate by county, 2009")

leg.txt <- c("<2.5%", "2.5-5%", "5-7.5%", "7.5-10%", ">10%")
legend("bottomleft", leg.txt, horiz = TRUE, fill = colors, cex=0.6)


# Exercise

# Draw contour lines of an upper half of a sphere with a radius of 5 and also make image

x <- matrix(rep(seq(-5,5,length=100),100),100,100,byrow=TRUE)
y <- t(x)

zsq <- (5^2 - x^2 - y^2)*(5^2 - x^2 - y^2 > 0)
z <- sqrt(zsq)
contour(x[1,],y[,1],z, nlevels=10)

image(z)

# Run the following regression model and draw a graph in 3-D.
x <- runif(100, min=-1, max=1)
y <- runif(100, min=-1, max=1)
z <- 1 + x + 2 * y + rnorm(100)
lm(z~x+y)

x1 <- matrix(rep(seq(-1, 1, length= 21),21),21,21,byrow=TRUE)
y1 <- t(x1)

z1 <- 1 + x1 + 2 * y1

surface <- persp(x1[1,], y1[,1], z1, theta = 10, phi = 30, expand = 1, col =
terrain.colors(30))
xy.list = trans3d(x, y, z, surface)

points(xy.list, pch=20, col="black")
```