Chapter 4 Visualization of Multivariate Data

Lecturer: Zhao Jianhua

Department of Statistics
Yunnan University of Finance and Economics
Outline

4.1 Introduction

4.2 Panel Displays

4.3 Surface Plots and 3D Scatter Plots
   4.3.1 Surface plots
   4.3.2 Three-dimensional scatterplot

4.4 Contour Plots

4.5 Other 2D Representations of Data
   4.5.1 Andrews Curves
   4.5.2 Parallel Coordinate Plots

4.6 Other Approaches to Data Visualization
Introduction

Visualization of multivariate data is related to exploratory data analysis (EDA).

- The term ‘exploratory’ is in contrast to ‘confirmatory’, which could describe hypothesis testing.
- It was important to do the exploratory work before hypothesis testing, to learn what are the appropriate questions to ask, and the most appropriate methods to answer them.
- With multivariate data, we may also be interested in dimension reduction or finding structure or groups in the data.

In this chapter, we focus on methods for visualizing multivariate data. Several graphics functions are used, including R graphics package, lattice and MASS, rggobi interface to GGobi and rgl package for interactive 3D visualization. Table 1.4 lists some basic graphics functions. Table 4.1 lists more.
<table>
<thead>
<tr>
<th>Method</th>
<th>in(graphics)</th>
<th>in(package)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D scatterplot</td>
<td></td>
<td>cloud (lattice)</td>
</tr>
<tr>
<td>Matrix of scatterplots</td>
<td>pairs</td>
<td>splom (lattice)</td>
</tr>
<tr>
<td>Bivariate density surface</td>
<td>persp</td>
<td>wireframe (lattice)</td>
</tr>
<tr>
<td>Contour plot</td>
<td>contour, image</td>
<td>contourplot (lattice)</td>
</tr>
<tr>
<td></td>
<td>contourLines</td>
<td>contour (MASS)</td>
</tr>
<tr>
<td></td>
<td>filled.contour</td>
<td>levelplot (lattice)</td>
</tr>
<tr>
<td>Parallel coord. plot</td>
<td></td>
<td>parallel (lattice)</td>
</tr>
<tr>
<td>Star plot</td>
<td>stars</td>
<td>parcoord (MASS)</td>
</tr>
<tr>
<td>Segment plot</td>
<td>stars</td>
<td></td>
</tr>
<tr>
<td>Interactive 3D graphics</td>
<td></td>
<td>(rggobi), (rgl)</td>
</tr>
</tbody>
</table>
Panel Displays

Panel display: an array of two-dimensional graphical summaries of pairs of variables in a multivariate dataset. For example, a scatterplot matrix displays the scatterplots for all pairs of variables in an array. `pairs`: produce a scatterplot matrix, as shown in Figures 4.1 and 4.2 in Example 4.1, and Figure 3.7. An example of three-dimensional plots is Figure 4.5.

**Example 4.1 (Scatterplot matrix)**

Compare the four variables in the iris data for the species virginica, in a scatterplot matrix.

```r
# virginica data in first 4 columns of the last 50 obs.
pairs(iris[101:150, 1:4])
```

The variable names will appear along the diagonal. The `pairs` function takes an optional argument `diag.panel`, which is a function that determines what is displayed along the diagonal.
To obtain a graph with estimated density curves along the diagonal, supply the name of a function to plot the densities. The following `panel.d` plot the densities.

```r
panel.d <- function(x, ...) {
  usr <- par("usr")
  on.exit(par(usr))
  par(usr = c(usr[1:2], 0, .5))
  lines(density(x))
}
```

In `panel.d`, the graphics parameter `usr` specifies the extremes of the user coordinates of the plotting region. Before plotting, apply `scale` to standardize each of the one-dimensional samples.

```r
x <- scale(iris[101:150, 1:4])
r <- range(x)
pairs(x, diag.panel = panel.d, xlim = r, ylim = r)
```

The pairs plot is displayed in Figure 4.1.
Observation: The length variables are positively correlated, and the width variables appear to be positively correlated. Other structure could be present in the data that is not revealed by the bivariate marginal distributions.
Illustrate the scatterplot matrix function `splom` in `lattice`.

```
library(lattice)
splom(iris[101:150, 1:4]) #plot 1
  #for all 3 at once, in color, plot 2
splom(iris[,1:4], groups = iris$Species)
  #for all 3 at once, black and white, plot 3
splom(~iris[1:4], groups = Species, data = iris,

  col = 1, pch = c(1, 2, 3), cex = c(.5,.5,.5))
```

The last plot (plot 3) is displayed in Figure 4.2. It is displayed here in black and white, but on screen the panel display is easier to interpret when displayed in color (plot 2). Also see the 3D scatterplot of the iris data in Figure 4.5.
Fig. 4.2: Scatterplot matrix comparing four measurements of iris data: setosa (circle), versicolor (triangle), virginica (cross) from Example 4.1.
4.3 Surface Plots and 3D Scatter Plots

- `persp` (graphics) draw perspective plots of surfaces over the plane.
- `demo(persp)`: try running the demo examples for persp.
- 3D methods in the `lattice` graphics package and the `rgl` package.

4.3.1 Surface plots

`expand.grid`: mesh a grid of regularly spaced points in the plane. If we do not need to save the \( x, y \) values, and only need the function values \( \{ z_{ij} = f(x_i, y_j) \} \), the outer function can be used.

**Example 4.2 (Plot bivariate normal density)**

Plot the standard bivariate normal density

\[
f(x, y) = \frac{1}{2\pi} e^{-\frac{1}{2}(x^2+y^2)}, \quad (x, y) \in \mathbb{R}^2
\]
In this example, $z_{ij} = f(x_i, y_j)$ are computed by the outer function.

```r
# the standard BVN density
f <- function(x, y) {
    z <- (1/(2*pi)) * exp(-.5 * (x^2 + y^2))
}
y <- x <- seq(-3, 3, length= 50)
z <- outer(x, y, f) # compute density for all (x,y)
persp(x, y, z) # the default plot
persp(x, y, z, theta = 45, phi = 30, expand = 0.6,
     ltheta = 120, shade = 0.75, ticktype = "detailed",
     xlab = "X", ylab = "Y", zlab = "f(x, y)"
)
```

The second version of the perspective plot is shown in Figure 4.3.

**R note 4.1**

- `outer(x, y, f)` apply the third argument $f$ to the grid of $(x, y)$ values. The returned value is a matrix of function values for every point $(x_i, y_j)$ in the grid.
- For a presentation, adding color (say, `col = "lightblue"`) produces a more attractive plot. The `box` can be suppressed by `box = FALSE`. 
Example 4.3 (Add elements to perspective plot)

Use the viewing transformation returned by the perspective plot of the standard bivariate normal density to add points, lines, and text.

```
# store viewing transformation in M
M = persp(x, y, z, theta = 45, phi = 30, expand = 0.4, box = FALSE)
```

Fig. 4.3: Perspective plot of the standard bivariate normal density in Example 4.2.
The transformation returned by the persp function call is

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>2.357023e-01</td>
<td>-0.1178511</td>
<td>0.2041241</td>
</tr>
<tr>
<td>[2,]</td>
<td>2.357023e-01</td>
<td>0.1178511</td>
<td>-0.2041241</td>
</tr>
<tr>
<td>[3,]</td>
<td>-2.184757e-16</td>
<td>4.3700078</td>
<td>2.5230252</td>
</tr>
<tr>
<td>[4,]</td>
<td>1.732284e-17</td>
<td>-0.3464960</td>
<td>-2.9321004</td>
</tr>
</tbody>
</table>

This transformation $M$ is applied to $(x, y, z, t)$ to project points onto the screen for display in the same coordinate system used to draw the perspective plot.

```r
# add some points along a circle
a <- seq(-pi, pi, pi/16)
newpts <- cbind(cos(a), sin(a))*2; newpts <- cbind(newpts, 0, 1)  # z=0, t=1
N <- newpts %*% M; points(N[,1]/N[,4], N[,2]/N[,4], col=2)

# add lines
x2 <- seq(-3, 3, .1); y2 <- -x2^2 / 3
z2 <- dnorm(x2) * dnorm(y2); N <- cbind(x2, y2, z2, 1) %*% M
lines(N[,1]/N[,4], N[,2]/N[,4], col=4)

# add text
x3 <- c(0, 3.1); y3 <- c(0, -3.1)
z3 <- dnorm(x3) * dnorm(y3) * 1.1; N <- cbind(x3, y3, z3, 1) %*% M
text(N[1,1]/N[1,4], N[1,2]/N[1,4], "f(x,y)")
text(N[2,1]/N[2,4], N[2,2]/N[2,4], bquote(y==(-x^2)/3))
```
The plot with added elements is shown in Fig.4.4 (Note: R provides a function `trans3d` to compute the coordinates above. Here we have shown the calculations.)
Other functions for graphing surfaces

Use `wireframe(lattice)` to display a surface plot of the bivariate normal density similar to Figure 4.3.

**Example 4.4 (Surface plot using `wireframe(lattice)`)**

`wireframe` requires a formula $z \sim x \times y$, where $z = f(x, y)$ is the surface to be plotted. $x, y$ and $z$ must have the same number of rows. Generate matrix of $(x, y)$ coordinates by `expand.grid`.

```r
library(lattice)
x <- y <- seq(-3, 3, length = 50)
xy <- expand.grid(x, y)
z <- (1/(2*pi)) * exp(-.5 * (xy[,1]^2 + xy[,2]^2))
wireframe(z ~ xy[,1] * xy[,2])
```

An interactive 3D display is provided by the graphics package `rgl`. One of the examples in the demo of `rgl` package shows a bivariate normal density.

```r
library(rgl)
demo(bivar) # or demo(rgl) to see more
```
4.3.2 Three-dimensional scatterplot

cloud (lattice) function produces 3D scatterplots, which could explore whether there are groups or clusters in the data. To apply cloud, provide a formula \( z \sim x \ast y \), where \( z = f(x, y) \) is the surface.

**Example 4.5 (3D scatterplot)**

Use cloud to display a 3D scatterplot of the iris data. There are three species of iris and each is measured on four variables. The following code produces a 3D scatterplot of sepal length, sepal width, and petal length (similar to (3) in Figure 4.5).

```r
library(lattice)
attach(iris)
# basic 3 color plot with arrows along axes
print(cloud(Petal.Length ~ Sepal.Length * Sepal.Width, data = iris, groups = Species))
```

The iris data has four variables, so there are four subsets of three variables to graph. To see all four plots on the screen, use the more and split options. The split arguments determine the location of the plot within the panel display.
print(cloud(Sepal.Length ~ Petal.Length * Petal.Width, 
  data = iris, groups = Species, main = "1", pch=1:3, 
  scales = list(draw = FALSE), zlab = "SL", 
  screen = list(z = 30, x = -75, y = 0)),
  split = c(1, 1, 2, 2), more = TRUE)

print(cloud(Sepal.Width ~ Petal.Length * Petal.Width, 
  data = iris, groups = Species, main = "2", pch=1:3, 
  scales = list(draw = FALSE), zlab = "SW", 
  screen = list(z = 30, x = -75, y = 0)),
  split = c(2, 1, 2, 2), more = TRUE)

print(cloud(Petal.Length ~ Sepal.Length * Sepal.Width, 
  data = iris, groups = Species, main = "3", pch=1:3, 
  scales = list(draw = FALSE), zlab = "PL", 
  screen = list(z = 30, x = -55, y = 0)),
  split = c(1, 2, 2, 2), more = TRUE)

print(cloud(Petal.Width ~ Sepal.Length * Sepal.Width, 
  data = iris, groups = Species, main = "4", pch=1:3, 
  scales = list(draw = FALSE), zlab = "PW", 
  screen = list(z = 30, x = -55, y = 0)),
  split = c(2, 2, 2, 2))

detach(iris)
Fig.4.5: 3D scatterplots of iris data produced by cloud (lattice) in Example 4.5, with each species represented by a different plotting character.
Observation: three species of iris are separated into groups or clusters, which is evident in these plots. One might follow up with cluster analysis or principal components analysis to analyze the apparent structure in the data.

R note 4.2

- The screen option sets the orientation of the axes. Setting `draw = FALSE` suppresses arrows and tick marks on the axes.
- To split the screen into \( n \) rows and \( m \) columns, and put the plot into position \((r, c)\), set `split` equal to the vector \((r, c, n, m)\).
- One unusual feature of cloud is that unlike most graphics functions in R, cloud does not plot a panel figure unless we print it.
4.4 Contour Plots

- A contour plot represents a 3D surface \((x, y, f(x, y))\) in the plane by projecting the level curves \(f(x, y) = c\) for selected constants \(c\).
- The functions `contour` (graphics) and `contourplot` (lattice) produce contour plots.
- The functions `filled.contour` in the graphics package and `levelplot` function in the lattice package produce filled contour plots. Both contour and contourplot label the contours by default.
- A variation of this type of plot is `image` (graphics), which uses color to identify contour levels.
Example 4.6 (Contour plot)

volcano data: an 87 by 61 matrix containing topographic information for the Maunga Whau volcano.

```r
# contour plot with labels
contour(volcano, asp = 1, labcex = 1)

# another version from lattice package
library(lattice)
contourplot(volcano) # similar to above
```

A 3D view of the volcano surface is provided in the examples of the `persp` function. Type `example(persp)`. An interactive 3D view of the volcano appears in the examples.

```r
library(rgl)
example(rgl)
```

For another 3D view of the volcano data, with shading to indicate contour levels, see the first example in the `wireframe` help file.
Fig. 4.6: Contour plot and levelplot of volcano data in Examples 4.6 and 4.7.
Example 4.7 (Filled contour plots)

A contour plot with a 3D effect could be displayed in 2D by overlaying the contour lines on a color map corresponding to the height. The `image` function in the graphics package provides the color background for the plot. The plot produced below is similar to Figure 4.6(a), with the background of the plot in terrain colors.

```r
image(volcano, col=terrain.colors(100), axes=FALSE)
contour(volcano, levels=seq(100,200,by=10), add=TRUE)
```

Using `image` without `contour` produces essentially the same type of plot as `filled.contour` (graphics) and `levelplot` (lattice). The contours of `filled.contour` and `levelplot` are identified by a legend rather than superimposing the contour lines.
Compare the plot produced by image with the following two plots.

```
filled.contour(volcano, color=terrain.colors, asp=1)
levelplot(volcano, scales = list(draw = FALSE),
xlab = "", ylab = "")
```

The plot produced by `levelplot` is shown in Figure 4.6(b).

- A limitation of 2D scatterplots is that for large data sets, there are often regions where data is very dense, and regions where data is quite sparse. In this case, the 2D scatterplot does not reveal much information about the bivariate density.

- Another approach is to produce a 2D or flat histogram, with the density estimate in each bin represented by an appropriate color.
Example 4.8 (2D histogram)

Simulated bivariate normal data is displayed in a flat histogram with hexagonal bins. `hexbin` in package `hexbin` produces a basic version of this plot in grayscale.

```
library(hexbin)
x <- matrix(rnorm(4000), 2000, 2)
plot(hexbin(x[,1], x[,2]))
```

Fig.4.7: Flat density histogram of bivariate normal data with hexagonal bins produced by hexbin in Example 4.8.
• Compare Figure 4.7 with Figure 10.11 on page 308. Note that the darker colors correspond to the regions where the density is highest, and colors are increasingly lighter along radial lines extending from the mode near the origin. The plot exhibits approximately circular symmetry, consistent with the standard bivariate normal density.

The bivariate histogram can also be displayed in 2D using a color palette, such as `heat.colors` or `terrain.colors`, to represent the density for each bin. A similar type of plot is implemented in the `gplots` package. The plot (not shown) resulting from the following code is similar to Figure 4.7, but with color and square bins.

```r
library(gplots)
hist2d(x, nbins = 30, 
col = c("white", rev(terrain.colors(30))))
```
4.5 Other 2D Representations of Data

Andrews curves, parallel coordinate plots, and various iconographic displays such as segment plots and star plots.

4.5.1 Andrews Curves

If $X_1, \ldots, X_n \in \mathbb{R}^d$, one approach to visualizing the data in two dimensions is to map each of the sample data vectors onto a real valued function. Andrews Curves map each sample observation $x_i = (x_{i1}, \ldots, x_{id})$ to the function

$$f_i(t) = \frac{x_{i1}}{\sqrt{2}} + x_{i2} \sin t + x_{i3} \cos t + x_{i4} \sin 2t + x_{i5} \sin 2t + \ldots$$

$$= \frac{x_{i1}}{\sqrt{2}} + \sum_{1 \leq k \leq d/2} x_{i,2k} \sin kt + \sum_{1 \leq k \leq d/2} x_{i,2k+1} \cos kt, -\pi \leq t \leq \pi.$$

Thus, each observation is represented by its projection onto a set of orthogonal basis functions $\{2^{-1/2}, \{\sin kt\}_{k=1}^{\infty}, \{\cos kt\}_{k=1}^{\infty}\}$. Notice that differences between measurements are amplified more in the lower frequency terms, so that the representation depends on the order of the variables or features.
Example 4.9 (Andrews curves)

- Measurements of leaves for two types of leaf architecture are represented by Andrews curves (leafshape17 in DAAG package). Three measurements (leaf length, petiole, and leaf width) correspond to points in $\mathbb{R}^3$.

- To plot the curves, define a function to compute $f_i(t)$ for arbitrary points $x_i$ in $\mathbb{R}^3$ and $-\pi \leq t \leq \pi$. Evaluate the function along the interval $[-\pi, \pi]$ for each sample point $x_i$.

```r
library(DAAG)
attach(leafshape17)
f <- function(a, v) {
  # Andrews curve f(a) for a data vector v in R^3
  v[1]/sqrt(2) + v[2]*sin(a) + v[3]*cos(a)}
  # scale data to range [-1, 1]
x <- cbind(bladelen, petiole, bladewid)
n <- nrow(x)
mins <- apply(x, 2, min)  # column minimums
maxs <- apply(x, 2, max)  # column maximums
```
\[ r \leftarrow \text{maxs} - \text{mins} \quad \text{#column ranges} \]
\[ y \leftarrow \text{sweep}(x, 2, \text{mins}) \quad \text{#subtract column mins} \]
\[ y \leftarrow \text{sweep}(y, 2, r, "/") \quad \text{#divide by range} \]
\[ x \leftarrow 2 \times y - 1 \quad \text{#now has range \([-1, 1]\)} \]

#set up plot window, but plot nothing yet
plot(0, 0, xlim = c(-pi, pi), ylim = c(-3,3),
xlab = "t", ylab = "Andrews Curves",
main = "", type = "n")

#now add the Andrews curves for each observation
#line type corresponds to leaf architecture
#0=orthotropic, 1=plagiotropic
a \leftarrow \text{seq}(-pi, pi, \text{len}=101)
dim(a) \leftarrow \text{length}(a)
for (i in 1:n) {
g \leftarrow \text{arch}[i] + 1
y \leftarrow \text{apply}(a, \text{MARGIN} = 1, \text{FUN} = f, v = x[i,])
\text{lines}(a, y, \text{lty} = g)
}
\text{legend}(3, c("Orthotrophic", "Plagiotrophic"), \text{lty} = 1:2)
\text{detach(leafshape17)}
The plot reveals similarities within plagiotropic and orthotropic leaf architecture groups, and differences between these groups. In general, this type of plot may reveal possible clustering of data.

**R note 4.4**

To identify the curves by color, replace `lty` with `col` parameters in the lines and legend statements.
In Example 4.9 the sweep operator is applied to subtract the column minimums above. The syntax is

\[ \text{\texttt{sweep}(x, \text{MARGIN}, \text{STATS}, \text{FUN}="-", \ldots)} \]

By default, the statistic is subtracted but other operations are possible. Here

\[ y \leftarrow \text{\texttt{sweep}}(x, 2, \text{mins}) \texttt{#subtract column mins} \]
\[ y \leftarrow \text{\texttt{sweep}}(y, 2, r, "/") \texttt{#divide by range} \]

sweeps out (subtracts) the minimum of each columns (margin = 2). Then the ranges of each of the three columns (in r) are swept out; that is, each column is divided by its range.
4.5.2 Parallel Coordinate Plots

- Parallel coordinate plots provide another approach to visualization of multivariate data.
- Rather than represent axes as orthogonal, the parallel coordinate system represents axes as equidistant parallel lines. Usually these lines are horizontal with common origin, scale, and orientation. Then to represent vectors in $\mathbb{R}^d$, the parallel coordinates are simply the coordinates along the $d$ copies of the real line. Each coordinate of a vector is then plotted along its corresponding axis, and the points are joined together with line segments.
- Parallel coordinate plots are implemented by the `parcoord` (MASS) and `parallel` (lattice). `parcoord` display the axes as vertical lines. `parallel` display the axes as horizontal lines.
Example 4.10 (Parallel coordinates)

- Use parallel (lattice) to construct a panel display of parallel coordinate plots for crabs (MASS) data.
- Crab data frame has 5 measurements on each of 200 crabs, from four groups of size 50. The groups are identified by species (blue or orange) and sex.
- The graph is best viewed in color. Here we use black and white, and for readability select only 1/5 of the data.

```r
library(MASS)
library(lattice)
trellis.device(color = FALSE) # black and white display
x <- crabs[seq(5, 200, 5), ] # get every fifth obs.
parallel(∼x[4:8] | sp*sex, x)
```

The resulting parallel coordinate plots are displayed in Figure 4.9(a). The labels along the vertical axis identify each axis corresponding to the five measurements (frontal lobe size, rear width, carapace length, carapace width, body depth). Much of the variability between groups is in overall size.
Fig. 4.9: Parallel coordinate plots in Example 4.10 for a subset of the crabs (MASS) data. (a) Differences between species (B=blue, O=orange) and sex (M, F) are largely obscured by large variation in overall size. (b) After adjusting the measurements for size of individual crabs, differences between groups are evident.
Adjusting the measurements of individual crabs for size may produce more interesting plots. We adjust the measurements by the area of the carapace.

```r
trellis.device(color = FALSE) # black and white display
x <- crabs[seq(5, 200, 5), ] # get every fifth obs.
a <- x$CW * x$CL # area of carapace
x[4:8] <- x[4:8] / sqrt(a) # adjust for size
parallel(~x[4:8] | sp*sex, x)
```

In the resulting plot in Figure 4.9(b), differences in species and sex are much more evident after adjustment than in Figure 4.9(a).
Example 4.11 (Segment plot)

This example uses the subset of crabs (MASS) data from Example 4.10. As in Example 4.10, individual measurements are adjusted for overall size by area of carapace.

```r
#segment plot
library(MASS) # for crabs data
attach(crabs)
x <- crabs[seq(5, 200, 5), ] # get every fifth obs.
x <- subset(x, sex == "M") # keep just the males
a <- x$CW * x$CL # area of carapace
x[4:8] <- x[4:8] / sqrt(a) # adjust for size
# use default color palette or other colors
palette(gray(seq(.4, .95, len = 5))) # use gray scale
# palette(rainbow(6)) # or use color
stars(x[4:8], draw.segments = TRUE,
labels = x$sp, nrow = 4,
ylim = c(-2,10), key.loc = c(3,-1))
# after viewing, restore the default colors
palette("default"); detach(crabs)
```
Fig. 4.10: Segment plot of a subset of the males in the crabs (MASS) data set in Example 4.11. The measurements have been adjusted by overall size of the individual crab. The two species are blue (B) and orange (O).

The observations are labeled by species. The differences between the species (for males) in this sample are quite evident in the plot. The plot suggests, for example, that orange crabs have greater body depth relative to carapace width than blue crabs.
4.6 Other Approaches to Data Visualization

- Asimov’s grand tour [14] is an interactive graphical tool that projects data onto a plane, rotating through all angles to reveal any structure in data. The grand tour is similar to projection pursuit exploratory data analysis (PPEDA) [100].

- Principal components analysis (PCA) similarly uses projections. Dimension is reduced by projecting onto a small number of principal components that collectively explain most of variation.

- Chernoff’s faces [46] are implemented in faces(aplpack) and in faces(TeachingDemos) [254]. Mosaic plots for visualization of categorical data are available in mosaicplot. Package vcd for visualization of categorical data. Functions prcomp and princomp provide PCA.

- Many packages for R fall under the data mining or machine learning umbrella; for a start see nnet, rpart, and randomForest. More packages are described on the Multivariate Task View and Machine Learning Task View on the CRAN web.