An Introduction to R
Lattice Graphics

Deepayan Sarkar

Indian Statistical Institute, Delhi

October 2011
R graphics

- R has two largely independent graphics subsystems
  - Traditional graphics
    - Available in R from the beginning
    - Rich collection of tools
    - Not very flexible
  - Grid graphics
    - Relatively recent (2000)
    - Low-level tool, highly flexible
- Grid forms the basis of two high-level graphics systems:
  - lattice: based on Trellis graphics (Cleveland)
  - ggplot2: inspired by “Grammar of Graphics” (Wilkinson)
The lattice package

- Trellis graphics for R (originally developed in S)
- Powerful high-level data visualization system
- Provides common statistical graphics with conditioning
  - Emphasis on multivariate data
  - Sufficient for typical graphics needs
  - Flexible enough to handle most nonstandard requirements
- Traditional user interface:
  - Collection of high-level functions: \texttt{xyplot}, \texttt{dotplot}, etc.
  - Interface based on formula and data source
## High-level functions in lattice

<table>
<thead>
<tr>
<th>Function</th>
<th>Default Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>histogram()</td>
<td>Histogram</td>
</tr>
<tr>
<td>densityplot()</td>
<td>Kernel Density Plot</td>
</tr>
<tr>
<td>qqmath()</td>
<td>Theoretical Quantile Plot</td>
</tr>
<tr>
<td>qq()</td>
<td>Two-sample Quantile Plot</td>
</tr>
<tr>
<td>stripplot()</td>
<td>Stripchart (Comparative 1-D Scatter Plots)</td>
</tr>
<tr>
<td>bwplot()</td>
<td>Comparative Box-and-Whisker Plots</td>
</tr>
<tr>
<td>barchart()</td>
<td>Bar Plot</td>
</tr>
<tr>
<td>dotplot()</td>
<td>Cleveland Dot Plot</td>
</tr>
<tr>
<td>xyplot()</td>
<td>Scatter Plot</td>
</tr>
<tr>
<td>splom()</td>
<td>Scatter-Plot Matrix</td>
</tr>
<tr>
<td>contourplot()</td>
<td>Contour Plot of Surfaces</td>
</tr>
<tr>
<td>levelplot()</td>
<td>False Color Level Plot of Surfaces</td>
</tr>
<tr>
<td>wireframe()</td>
<td>Three-dimensional Perspective Plot of Surfaces</td>
</tr>
<tr>
<td>cloud()</td>
<td>Three-dimensional Scatter Plot</td>
</tr>
<tr>
<td>parallel()</td>
<td>Parallel Coordinates Plot</td>
</tr>
</tbody>
</table>
The Chem97 dataset

1997 A-level Chemistry examination in Britain

```r
data(Chem97, package = "mlmRev")
head(Chem97[c("score", "gender", "gcsescore")])
```

<table>
<thead>
<tr>
<th>score</th>
<th>gender</th>
<th>gcsescore</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>6.625</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>7.625</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>7.250</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>7.500</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>6.444</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>7.750</td>
</tr>
</tbody>
</table>
> histogram(~ gcsescore, data = Chem97)
> histogram(~ gcsescore | factor(score), data = Chem97)
> densityplot(~ gcsescore | factor(score), Chem97,
plot.points = FALSE,
groups = gender, auto.key = TRUE)
Trellis Philosophy: Part I

- Display specified in terms of
  - Type of display (histogram, densityplot, etc.)
  - Variables with specific roles

- Typical roles for variables
  - Primary variables: used for the main graphical display
  - Conditioning variables: used to divide into subgroups and juxtapose (multipanel conditioning)
  - Grouping variable: divide into subgroups and superpose

- Primary interface: high-level functions
  - Each function corresponds to a display type
  - Specification of roles depends on display type
    - Usually specified through the formula and the \texttt{groups} argument
```r
> qqmath(~ gcsescore | factor(score), Chem97,
groups = gender, auto.key = list(columns = 2),
fun = ppoints(100),
type = c("p", "g"), aspect = "xy")
```
> qq(gender ~ gcse_score | factor(score), Chem97,
  f.value = ppoints(100), type = c("p", "g"),
  aspect = 1)
> bwplot(factor(score) ~ gcsescore | gender, Chem97)
> bwplot(gcsescore ~ gender | factor(score), Chem97, layout = c(6, 1))
> stripplot(depth ~ factor(mag), data = quakes, jitter.data = TRUE, alpha = 0.6)
The **VADeaths** dataset

- Death rates in Virginia, 1941, among different population subgroups

```r
> VADeaths

<table>
<thead>
<tr>
<th></th>
<th>Rural Male</th>
<th>Rural Female</th>
<th>Urban Male</th>
<th>Urban Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>50-54</td>
<td>11.7</td>
<td>8.7</td>
<td>15.4</td>
<td></td>
</tr>
<tr>
<td>55-59</td>
<td>18.1</td>
<td>11.7</td>
<td>24.3</td>
<td></td>
</tr>
<tr>
<td>60-64</td>
<td>26.9</td>
<td>20.3</td>
<td>37.0</td>
<td></td>
</tr>
<tr>
<td>65-69</td>
<td>41.0</td>
<td>30.9</td>
<td>54.6</td>
<td></td>
</tr>
<tr>
<td>70-74</td>
<td>66.0</td>
<td>54.3</td>
<td>71.1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Urban Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>50-54</td>
</tr>
<tr>
<td>55-59</td>
</tr>
<tr>
<td>60-64</td>
</tr>
<tr>
<td>65-69</td>
</tr>
<tr>
<td>70-74</td>
</tr>
</tbody>
</table>
> barchart(VADeaths, groups = FALSE, layout = c(4, 1))
> dotplot(VADeaths, groups = FALSE, layout = c(4, 1))
> dotplot(VADeaths, type = "o",
        auto.key = list(points = TRUE, lines = TRUE,
                      space = "right"))
> data(Earthquake, package = "nlme")
> xyplot(accel ~ distance, data = Earthquake)
> xyplot(accel ~ distance, data = Earthquake,
    scales = list(log = TRUE),
    type = c("p", "g", "smooth"))
> Depth <- equal.count(quakes$depth, number = 8, overlap = 0.1)

> summary(Depth)

Intervals:

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>max</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>39.5</td>
<td>63.5</td>
<td>138</td>
</tr>
<tr>
<td>2</td>
<td>60.5</td>
<td>102.5</td>
<td>138</td>
</tr>
<tr>
<td>3</td>
<td>97.5</td>
<td>175.5</td>
<td>138</td>
</tr>
<tr>
<td>4</td>
<td>161.5</td>
<td>249.5</td>
<td>142</td>
</tr>
<tr>
<td>5</td>
<td>242.5</td>
<td>460.5</td>
<td>138</td>
</tr>
<tr>
<td>6</td>
<td>421.5</td>
<td>543.5</td>
<td>137</td>
</tr>
<tr>
<td>7</td>
<td>537.5</td>
<td>590.5</td>
<td>140</td>
</tr>
<tr>
<td>8</td>
<td>586.5</td>
<td>680.5</td>
<td>137</td>
</tr>
</tbody>
</table>

Overlap between adjacent intervals:
[1] 16 14 19 15 14 15 15
> xyplot(lat ~ long / Depth, data = quakes)
> cloud(depth ~ lat * long, data = quakes,
    zlim = rev(range(quakes$depth)),
    screen = list(z = 105, x = -70),
    panel.aspect = 0.75)
> cloud(depth ~ lat * long, data = quakes,
  zlim = rev(range(quakes$depth)),
  screen = list(z = 80, x = -70),
  panel.aspect = 0.75)
More high-level functions

• More high-level functions in lattice
  • Won’t discuss, but examples in manual page
• Other Trellis high-level functions can be defined in other packages, e.g.,
  • `ecdfplot()`, `mapplot()` in the `latticeExtra` package
  • `hexbinplot()` in the `hexbin` package
The “trellis” object model

- One important feature of lattice:
  - High-level functions do not actually plot anything
  - They return an object of class “trellis”
  - Display created when such objects are `print()`-ed or `plot()`-ed

- Usually not noticed because of automatic printing rule
- Can be used to arrange multiple plots
- Other uses as well
> dp.uspe <-
  dotplot(t(USPersonalExpenditure),
  groups = FALSE, layout = c(1, 5),
  xlab = "Expenditure (billion dollars)")
> dp.uspe.log <-
  dotplot(t(USPersonalExpenditure),
  groups = FALSE, layout = c(1, 5),
  scales = list(x = list(log = 2)),
  xlab = "Expenditure (billion dollars)")
> plot(dp.uspe, split = c(1, 1, 2, 1))
> plot(dp.uspe.log, split = c(2, 1, 2, 1),
  newpage = FALSE)
Trellis Philosophy: Part I

• Display specified in terms of
  • Type of display (histogram, densityplot, etc.)
  • Variables with specific roles

• Typical roles for variables
  • Primary variables: used for the main graphical display
  • Conditioning variables: used to divide into subgroups and juxtapose (multipanel conditioning)
  • Grouping variable: divide into subgroups and superpose

• Primary interface: high-level functions
  • Each function corresponds to a display type
  • Specification of roles depends on display type
    • Usually specified through the formula and the groups argument
Trellis Philosophy: Part II

• Design goals:
  • Enable effective graphics by encouraging good graphical practice (e.g., Cleveland, 1985)
  • Remove the burden from the user as much as possible by building in good defaults into software

• Some obvious examples:
  • Use as much of the available space as possible
  • Encourage direct comparison by superposition (grouping)
  • Enable comparison when juxtaposing (conditioning):
    • use common axes
    • add common reference objects (such as grids)

• Inevitable departure from traditional R graphics paradigms
Any serious graphics system must also be flexible. Lattice tries to balance flexibility and ease of use using the following model:

- A display is made up of various elements
- Coordinated defaults provide meaningful results, but
- Each element can be controlled independently
- The main elements are:
  - the primary (panel) display
  - axis annotation
  - strip annotation (describing the conditioning process)
  - legends (typically describing the grouping process)
• The full system would take too long to describe
• Online documentation has details; start with \texttt{?Lattice}