

Deepayan Sarkar

Lattice

Multivariate Data Visualization with R

June 17, 2008

Springer

Berlin Heidelberg New York

Hong Kong London

Milan Paris Tokyo

To Pinaki Mitra

Preface

The `lattice` package is software that extends the R language and environment for statistical computing (R Development Core Team, 2007) by providing a coherent set of tools to produce statistical graphics with an emphasis on multivariate data. It is modeled on the Trellis suite in S and S-PLUS[®]. From the user's point of view, it is a self-contained system that is largely independent of other graphics facilities in R. This book is about `lattice`, and is primarily intended for (1) both long-time and new R users looking for a powerful system to produce conventional statistical graphics, (2) existing `lattice` users willing to learn a little bit of R programming to gain increased flexibility, and (3) developers who wish to implement new graphical displays building on the infrastructure already available in `lattice`.

Why `lattice`?

Graphics can effectively complement statistical data analysis in various ways. Successful graphics arise from a combination of good design and good implementation. In this day and age, implementation is almost exclusively driven by computers. There is no lack of software tools that allow their users to convert data into graphics; `lattice` is yet another candidate in this ever-widening pool.

What makes `lattice` stand out? A good general-purpose tool should not get in the way of the user, yet it should be flexible enough to enable most tasks (without undue difficulty), whether it be standard, slightly out of the ordinary, or entirely novel. `lattice` tries to meet this standard by being a high-level tool that produces structured graphics, while retaining flexibility by systematically decoupling the various elements of a display; the individual elements have well thought-out defaults, but these can be overridden for detailed control. The end-product is a system that allows the creation of common statistical graphics, often with fairly complex structure, with very simple code. At the same time, it allows various degrees of customization, without requiring undue effort.

What to expect from this book

It is easy to get started with `lattice`, but the transition from seemingly simple to more sophisticated use can be difficult without an appreciation of how the different components and their defaults interact with each other. This appreciation can only come from experience, but it is hoped that this book can ease the transition to some extent.

The book started out as a manual for `lattice`, and was not intended to offer qualitative guidelines about the effective design of statistical graphics. This plan was abandoned quite early on; a static book is not the ideal vehicle for documenting an evolving system, and it is hard to look at and change bits and pieces of a picture without discussing its merits and drawbacks. In the end, this book consists of some comments on graphical design, some interesting (one would hope) examples, and large doses of `lattice` code and wisdom. It is still a book that is primarily about software; the code in the book is at least as important as the pictures. No code is hidden in this book, and if there is one key message that the reader should expect to take away, it is that `lattice` allows the creation of complex displays using relatively little code. This economy may not be appealing to everyone, but it is what I liked most about the Trellis system, and what has driven much of the development of `lattice` beyond the original goal of compatibility with Trellis. The other key idea that is difficult to communicate in function documentation, and one that is addressed in this book, is that of interrelationships between the different components of `lattice`, and how they can be effectively exploited.

What not to expect from this book

This book is not an exhaustive manual for `lattice`. Most functions in `lattice` are described to some extent in this book, but it does not serve as the definitive reference. There are two reasons for this. First, there are many features in `lattice` that are obscure and of very limited use, and do not justify detailed discussion. Second, `lattice` is an evolving system, and any attempt to document it exhaustively is sure to get out of date quickly. All functions in `lattice` come with online documentation, which should be used as the definitive reference.

How to read this book

That depends to a large extent on the reader. Those new to `lattice` should start with Chapter 1 to get a feel for what `lattice` is all about. Chapter 2 gives a more thorough, and sometimes quite technical, overview of the `lattice` model. Intermediate to advanced readers should find this chapter instructive. Beginners are encouraged to go through it as well, but should be prepared to encounter parts they find difficult, and skip them without getting bogged down; things should become clearer after gaining some practical experience.

The rest of Part I describes the various high-level functions in `lattice`. These chapters can be read in any order. Not much is said about the design of these graphics as they are standard, and most of the focus is on the software implementation. The level is basic for the most part; however, a few examples do go into some detail for the sake of taking a discussion to its natural conclusion. Again, beginners should be prepared to skip these parts during a first reading. Part II is more of a reference, going into the nitty-gritty details of `lattice`. A basic understanding of all the chapters is important to get the most out of `lattice`, but is not essential for casual use. These chapters too can be read in any order, for the most part, and the reader should feel free to pick and choose. The final two chapters, in Part III, deal with extensions to `lattice`, and are primarily intended for future developers. Of course, they can still be useful to the casual reader for the examples they provide.

It is important to realize that `lattice` is a complicated piece of software, and it is unrealistic to expect full mastery of it after one reading. The key to “getting it” is practical experience, and the best way to gain that experience is to try out the code. All the code in this book, along with the figures they produce, is available from the supporting Web site

<http://lmdvr.r-forge.r-project.org/>

A critical aspect of graphics that is hard to communicate in a book is its iterative nature; graphics that are presented to an audience is rarely the result of a first attempt. This process is reflected in some of the examples in this book, but many others have silently omitted many intermediate steps. One can get a sense of these missing steps by asking: “What is the purpose of this particular argument?” In other words, trying out variations of the code should be an integral part of the learning process.

The final thing to remember is that all this is the means to an end, namely, producing effective visualizations of data. Software can help, but the ultimate decisions are still the responsibility of the user. For those looking for guidance on how to create effective graphs, the work of Edward R. Tufte, William S. Cleveland, and of course John W. Tukey, are invaluable resources.

Color

Color can be an important factor in the visual impact of a graphic. Most figures in this book are black and white, but a few color plates are also included. Of these, some have the corresponding black and white versions as well, and have been chosen to highlight the impact of color. Others are solely available in color, as their black and white versions are of little or no use. Color versions of all figures are available on the book’s Web site.

Prerequisites

No prior experience with `lattice` is required to read this book, but basic familiarity with R, and in particular the ability to use its online help system, is

assumed. The first chapter of Dalgaard (2002) should suffice for the most part. Relatively advanced concepts such as generic functions and method dispatch are relevant, but can be ignored for casual use (these concepts are briefly introduced where relevant, but not at any deep level). No familiarity with traditional R graphics is presumed. Knowledge of the `grid` package can be beneficial, but is not essential.

Several R packages are used in this book. `lattice` itself should come with all recent installations of R, and it should be sufficient to type

```
> library("lattice")
```

at the R prompt to start using it. Other packages used explicitly (not counting further dependencies) are `grid`, `latticeExtra`, `copula`, `ellipse`, `gridBase`, `flowViz`, `flowCore`, `hexbin`, `locfit`, `logspline`, `mapproj`, `maps`, `MASS`, `MEMSS`, `mlmRev`, and `RColorBrewer`. All of these may not be of interest (some are required just for one or two examples); type

```
> help("install.packages")
```

to learn how to install the packages you need from CRAN.¹ `flowCore`, `flowViz`, and `hexbin` are Bioconductor packages, and may be installed by typing

```
> source("http://bioconductor.org/biocLite.R")
> biocLite(c("flowCore", "flowViz", "hexbin"))
```

A bit of history

The design of S graphics has been heavily influenced by the principles of graph construction laid out in *The Elements of Graphing Data* (Cleveland, 1985). This influence carries over to Trellis graphics, which incorporates further ideas (notably multipanel conditioning and banking) presented in *Visualizing Data* (Cleveland, 1993). Trellis graphics was first implemented in the S system, and has been available in S-PLUS for several years.

The name Trellis refers both to the general ideas underlying the system, as well as the specific implementation in S. The `lattice` package is an independent implementation of Trellis graphics (in the first sense), with an API closely modeled on the one in S. Unlike the S version, which is implemented using traditional graphics, `lattice` uses Paul Murrell's `grid` package, which provides more flexible low-level tools.

Although modeled on it, the `lattice` API is not identical to that of the Trellis suite in S. Some of the differences are due to the choice of `grid` as the underlying engine, but many are intentional. Still, where possible, effort has been made to allow Trellis code written in S to run with minimal modification. Consequently, writings about the original Trellis suite mostly apply to `lattice` as well. This includes the wealth of resources at the Trellis Web site at Bell Labs:

¹ The Comprehensive R Archive Network, <http://cran.r-project.org>

<http://netlib.bell-labs.com/cm/ms/departments/sia/project/trellis/>

However, the converse is not true. `lattice` has been extended beyond the original API in various ways, and is now at a point where it is difficult to partition its feature set into S-compatible ones and additional enhancements. For this reason, this book makes no attempt to distinguish between these, and presents Trellis graphics solely as implemented in the `lattice` package.

Caveats and alternatives

No system is perfect for all uses, and `lattice` is no exception. Trellis is a “high-level” paradigm by design, and `lattice` imposes considerable structure on the displays it creates. `lattice` allows a lot of wiggle room within these constraints while retaining its stylistic consistency and simple user interface, but this is not always enough. Fortunately, R provides excellent facilities for creating new displays from scratch, especially using the `grid` package. `lattice` itself is implemented using `grid`, and can benefit from the use of low-level facilities provided by it. Even for high-level graphics, R provides various alternatives. The traditional graphics system includes many high-level tools, which although not as proficient in dealing with multivariate data, often provide a richer set of options. Murrell (2005) gives a comprehensive overview of both traditional R graphics and `grid` graphics (as well as a brief introduction to `lattice`). The `vcd` package, inspired by Friendly (2000), provides many useful tools for categorical data, often with Trellis-style conditioning. Another high-level alternative is Hadley Wickham’s `ggplot2` (formerly `ggplot`) package, modeled on the approach of Wilkinson (1999), which is philosophically rather different from the Trellis approach. Like `lattice`, `vcd` and `ggplot` are also implemented using `grid`.

One thing R currently has virtually no support for is interactive graphics. Fortunately, some R packages provide interfaces to external systems that are better, notably `rgl` (OpenGL) and `rggobi` (GGobi). The `playwith` package written by Felix Andrews provides a modicum of interactivity within the R graphics framework, and works well with displays produced by `lattice`.

Acknowledgements

Ross Ihaka and Paul Murrell are the primary architects of the graphics infrastructure in R that `lattice` builds upon; `lattice` would have been particularly difficult to write without Paul’s `grid` package. The `lattice` API is modeled on the original implementation of Trellis graphics in S. Implementing software is nowhere near as difficult as designing it, and the success of `lattice` owes much to the insight of the original designers. R is the wonderful platform that it is due to the efforts of its developer team and its vibrant user community, whose feedback has also driven much of the development of `lattice` beyond its

original goals. My own introduction to R was eased by encouragement and guidance from Doug Bates and Saikat DebRoy.

Thanks to John Kimmel and Doug Bates for convincing me to start writing this book, and to Robert Gentleman for providing a wonderful environment in which to finish it. Several friends and colleagues commented on drafts of the book and the supporting Web site. Feedback from Springer's anonymous reviewers, and in particular John Maindonald, has been extremely valuable. On a technical note, this book was written on a number of different computers running Debian[®] GNU/Linux[®], using the combination of Sweave and L^AT_EX as a document preparation system. The use of Emacs+ESS as the editing environment has improved productivity considerably.

Seattle, WA
December 2007

Deepayan Sarkar

Contents

Preface	vii
1 Introduction	1
1.1 Multipanel conditioning	2
1.1.1 A histogram for every group	2
1.1.2 The Trellis call	3
1.1.3 Kernel density plots	4
1.2 Superposition	5
1.3 The “ <i>trellis</i> ” object	6
1.3.1 The missing Trellis display	7
1.3.2 Arranging multiple Trellis plots	7
1.4 Looking ahead	7

Part I Basics

2 A Technical Overview of <i>lattice</i>	13
2.1 Basic usage	13
2.1.1 The Trellis formula	13
2.1.2 The <code>data</code> argument	14
2.1.3 Conditioning	14
2.1.4 Shingles	15
2.2 Dimension and physical layout	16
2.2.1 Aspect ratio	19
2.2.2 Layout	20
2.2.3 Fine-tuning the layout: <code>between</code> and <code>skip</code>	24
2.3 Grouped displays	24
2.4 Annotation: Captions, labels, and legends	26
2.4.1 More on legends	26
2.5 Graphing the data	28
2.5.1 Scales and axes	28

2.5.2	The panel function	30
2.5.3	The panel function demystified	31
2.6	Return value	33
3	Visualizing Univariate Distributions	35
3.1	Density Plot	35
3.2	Large datasets	37
3.3	Histograms	39
3.4	Normal Q–Q plots	40
3.4.1	Normality and the Box–Cox transformation	42
3.4.2	Other theoretical Q–Q plots	43
3.5	The empirical CDF	44
3.6	Two-sample Q–Q plots	44
3.7	Box-and-whisker plots	47
3.7.1	Violin plots	47
3.8	Strip plots	50
3.9	Coercion rules	52
3.10	Discrete distributions	53
3.11	A note on the formula interface	54
4	Displaying Multiway Tables	55
4.1	Cleveland dot plot	55
4.2	Bar chart	57
4.2.1	Manipulating order	61
4.2.2	Bar charts and discrete distributions	63
4.3	Visualizing categorical data	65
5	Scatter Plots and Extensions	67
5.1	The standard scatter plot	67
5.2	Advanced indexing using <code>subscripts</code>	71
5.3	Variants using the <code>type</code> argument	75
5.3.1	Superposition and <code>type</code>	79
5.4	Scatter-plot variants for large data	82
5.5	Scatter-plot matrix	84
5.5.1	Interacting with scatter-plot matrices	86
5.6	Parallel coordinates plot	87
6	Trivariate Displays	91
6.1	Three-dimensional scatter plots	91
6.1.1	Dynamic manipulation versus stereo viewing	95
6.1.2	Variants and panel functions	96
6.2	Surfaces and two-way tables	98
6.2.1	Data preparation	99
6.2.2	Visualizing surfaces	102
6.2.3	Visualizing discrete array data	105

6.3	Theoretical surfaces	110
6.3.1	Parameterized surfaces	111
6.4	Choosing a palette for false-color plots	113

Part II Finer Control

7	Graphical Parameters and Other Settings	119
7.1	The parameter system	119
7.1.1	Themes	120
7.1.2	Devices	120
7.1.3	Initializing a graphics device	121
7.1.4	Reading and modifying a theme	122
7.1.5	Usage and alternative forms	125
7.1.6	The <code>par.settings</code> argument	125
7.2	Available graphical parameters	126
7.2.1	Nonstandard settings	129
7.3	Non-graphical options	131
7.3.1	Argument defaults	131
7.4	Making customizations persistent	131
8	Plot Coordinates and Axis Annotation	133
8.1	Packets and the <code>prepanel</code> function	133
8.2	The <code>scales</code> argument	134
8.2.1	Relation	134
8.2.2	Axis annotation: Ticks and labels	135
8.2.3	Defaults	138
8.2.4	Three-dimensional displays: <code>cloud()</code> and <code>wireframe()</code>	139
8.3	Limits and aspect ratio	140
8.3.1	The <code>prepanel</code> function revisited	140
8.3.2	Explicit specification of limits	141
8.3.3	Choosing aspect ratio by banking	143
8.4	Scale components and the <code>axis</code> function	144
8.4.1	Components	144
8.4.2	Axis	148
9	Labels and Legends	151
9.1	Labels	151
9.2	Legends	152
9.2.1	Legends as grid graphical objects	152
9.2.2	The <code>colorkey</code> argument	155
9.2.3	The <code>key</code> argument	156
9.2.4	The problem with settings, and the <code>auto.key</code> argument	158
9.2.5	Dropping unused levels from <code>groups</code>	159
9.2.6	A more complicated example	159

9.2.7	Further control: The <code>legend</code> argument	161
9.3	Page annotation	162
10	Data Manipulation and Related Topics	165
10.1	Nonstandard evaluation	165
10.2	The extended formula interface	166
10.3	Combining data sources with <code>make.groups()</code>	170
10.4	Subsetting	173
10.4.1	Dropping of factor levels	176
10.5	Shingles and related utilities	177
10.5.1	Coercion to factors and shingles	182
10.5.2	Using shingles for axis breaks	183
10.5.3	Cut-and-stack plots	184
10.6	Ordering levels of categorical variables	187
10.7	Controlling the appearance of strips	193
10.8	An Example Revisited	198
11	Manipulating the “<i>trellis</i>” Object	201
11.1	Methods for “ <i>trellis</i> ” objects	201
11.2	The <code>plot()</code> , <code>print()</code> , and <code>summary()</code> methods	202
11.3	The <code>update()</code> method and <code>trellis.last.object()</code>	206
11.4	Tukey mean-difference plot	208
11.5	Specialized manipulations	210
11.6	Manipulating the display	211
12	Interacting with Trellis Displays	215
12.1	The traditional graphics model	215
12.1.1	Interaction	216
12.2	Viewports, <code>trellis.vpname()</code> , and <code>trellis.focus()</code>	216
12.3	Interactive additions	217
12.4	Other uses	223

Part III Extending Trellis Displays

13	Advanced Panel Functions	229
13.1	Preliminaries	229
13.1.1	Building blocks for panel functions	229
13.1.2	Accessor functions	231
13.1.3	Arguments	232
13.2	A toy example: Hypotrochoids and hypocycloids	232
13.3	Some more examples	235
13.3.1	An alternative density estimate	235
13.3.2	A modified box-and-whisker plot	237
13.3.3	Corrgrams as customized level plots	238

13.4	Three-dimensional projections	241
13.5	Maps	242
13.5.1	A simple projection scheme	244
13.5.2	Maps with conditioning	245
14	New Trellis Displays	247
14.1	<i>S</i> ³ methods	248
14.2	<i>S</i> ₄ methods	249
14.3	New functions	251
14.3.1	A complete example: Multipanel pie charts	252
	References	255
	Index	259