ggplot2 Version of Figures in Lattice: Multivariate Data Visualization with R

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Introduction **1**
1.1 Figure 1.1 . . . 1
1.2 Figure 1.2 . . . 2
1.3 Figure 1.3 . . . 2

2 A Technical Overview of lattice **4**
2.1 Figure 2.1 . . . 4
2.2 Figure 2.2 . . . 5
2.3 Figure 2.3 . . . 5
2.4 Figure 2.4 . . . 6
2.5 Figure 2.5 . . . 7
2.6 Figure 2.6 . . . 8
2.7 Figure 2.7 . . . 9
2.8 Figure 2.8 . . . 10
2.9 Figure 2.9 . . . 10

3 Visualizing Univariate Distributions **12**
3.1 Figure 3.1 . . . 12
3.2 Figure 3.2 . . . 13
3.3 Figure 3.3 . . . 13
3.4 Figure 3.4 . . . 14
3.5 Figure 3.5 . . . 15
3.6 Figure 3.6 . . . 16
3.7 Figure 3.7 . . . 16
3.8 Figure 3.8 . . . 17
3.9 Figure 3.9 . . . 17
3.10 Figure 3.10 . . . 18
3.11 Figure 3.11 . . . 18
3.12 Figure 3.12 . . . 19
3.13 Figure 3.13 . . . 19
3.14 Figure 3.14 . . . 20
3.15 Figure 3.15 . . . 20
3.16 Figure 3.16 . . . 21
3.17 Figure 3.17 . . . 21

4 Displaying Multiway Tables **23**

4.1 Figure 4.1 . . . 23
4.2 Figure 4.2 . . . 24
4.3 Figure 4.3 . . . 24
4.4 Figure 4.4 . . . 25
4.5 Figure 4.5 . . . 26
4.6 Figure 4.6 . . . 27
4.7 Figure 4.7 . . . 27
4.8 Figure 4.8 . . . 28
4.9 Figure 4.9 . . . 29

5 Scatter Plots and Extensions **30**

5.1 Figure 5.1 . . . 30
5.2 Figure 5.2 . . . 31
5.3 Figure 5.3 . . . 31
5.4 Figure 5.4 . . . 32
5.5 Figure 5.5 . . . 33
5.6 Figure 5.6 . . . 34
5.7 Figure 5.7 . . . 34
5.8 Figure 5.8 . . . 35
5.9 Figure 5.9 . . . 36
5.10 Figure 5.10 . . . 36
5.11 Figure 5.11 . . . 37
5.12 Figure 5.12 . . . 38
5.13 Figure 5.13 . . . 38
5.14 Figure 5.14 . . . 39
5.15 Figure 5.15 - Scatter Plot Matrix . . . 39
5.16 Figure 5.16 . . . 40
5.17 Figure 5.17 . . . 40
5.18 Figure 5.18 . . . 41
5.19 Figure 5.19 . . . 41

6 Trivariate Displays **43**

6.1 Figure 6.1 . . . 43
6.2 Figure 6.2 . . . 44
6.3 Figure 6.3 . . . 44
6.4 Figure 6.4 . . . 45
6.5 Figure 6.5 . . . 46
6.6 Figure 6.6 . . . 47

6.7 Figure 6.7 . . . 47
6.8 Figure 6.8 . . . 47
6.9 Figure 6.9 . . . 48
6.10 Figure 6.10 . . . 49
6.11 Figure 6.11 . . . 50
6.12 Figure 6.12 . . . 51
6.13 Figure 6.13 . . . 51
6.14 Figure 6.14 . . . 52
6.15 Figure 6.15 . . . 53
6.16 Figure 6.16 . . . 53
6.17 Figure 6.17 . . . 54
6.18 Figure 6.18 . . . 54
6.19 Figure 6.19 . . . 55

7 Graphical Parameters and Other Settings 57
7.1 Figure 7.1 . . . 57
7.2 Figure 7.2 . . . 58
7.3 Figure 7.3 . . . 58
7.4 Figure 7.4 . . . 59

8 Plot Coordinates and Axis Annotation 60
8.1 Figure 8.1 . . . 60
8.2 Figure 8.2 . . . 61
8.3 Figure 8.3 . . . 62
8.4 Figure 8.4 . . . 63
8.5 Figure 8.5 . . . 64
8.6 Figure 8.6 . . . 65

9 Labels and Legends 67
9.1 Figure 9.1 . . . 67
9.2 Figure 9.2 . . . 68
9.3 Figure 9.3 . . . 68

10 Data Manipulation and Related Topics 70
10.1 Figure 10.1 . . . 70
10.2 Figure 10.2 . . . 71
10.3 Figure 10.3 . . . 71
10.4 Figure 10.4 . . . 72
10.5 Figure 10.5 . . . 72
10.6 Figure 10.6 . . . 73

10.7 Figure 10.7 . . . 74
10.8 Figure 10.8 . . . 74
10.9 Figure 10.9 . . . 75
10.10 Figure 10.10 . . . 75
10.11 Figure 10.11 . . . 76
10.12 Figure 10.12 . . . 77
10.13 Figure 10.13 . . . 78
10.14 Figure 10.14 . . . 79
10.15 Figure 10.15 . . . 79
10.16 Figure 10.16 . . . 80
10.17 Figure 10.17 . . . 81
10.18 Figure 10.18 . . . 83
10.19 Figure 10.19 . . . 84
10.20 Figure 10.20 . . . 85
10.21 Figure 10.21 . . . 86
10.22 Figure 10.22 . . . 87
10.23 Figure 10.23 . . . 88
10.24 Figure 10.24 . . . 89
10.25 Figure 10.25 . . . 90

11 Manipulating the "trellis" object **91**
11.1 Figure 11.1 . . . 91
11.2 Figure 11.2 . . . 92
11.3 Figure 11.3 . . . 93
11.4 Figure 11.4 . . . 93
11.5 Figure 11.5 . . . 94
11.6 Figure 11.6 . . . 95

12 Advanced Panel Functions **96**
12.1 Figure 13.1 . . . 96
12.2 Figure 13.2 . . . 97
12.3 Figure 13.3 . . . 98
12.4 Figure 13.4 . . . 99
12.5 Figure 13.5 . . . 100
12.6 Figure 13.6 . . . 101
12.7 Figure 13.7 . . . 102
12.8 Figure 13.8 . . . 103
12.9 Figure 13.9 . . . 104
12.10 Figure 13.10 . . . 105

13 New Trellis Displays **106**

13.1 Figure 14.1 106

13.2 Figure 14.2 107

13.3 Figure 14.3 108

13.4 Figure 14.4 108

13.5 Figure 14.5 109

The data visualization package lattice is part of the base R distribution, and like ggplot2 is built on Grid graphics engine. Deepayan Sarkar's (the developer of lattice) book Lattice: Multivariate Data Visualization with R gives a detailed overview of how the package works. All the figures and code used to produce them is also available on the book website.

In order to give those interested an option to compare graphs produced by ggplot2 and lattice, I will attempt to recreate the book's lattice graphs in ggplot2. There are 14 chapters in the book, so this means that there would be at least 13 more posts on the subject.

The output of both packages can be tweaked so that the graphs would look similar if not the same, however for the purposes of comparison, the standard settings (at least in ggplot2) are used when possible. The code used to create the images is in separate paragraphs, allowing easy comparison, and by clicking on the thumbnail, a bigger image file is also available.

Chapter 1

Introduction

TOPICS COVERED:

- Basic usage; high level functions
- Conditioning
- Superposition (a.k.a. grouping)
- "trellis" objects

1.1 Figure 1.1

```
> library(lattice)
> library(ggplot2)
> data(Chem97, package = "mlmRev")
```

lattice

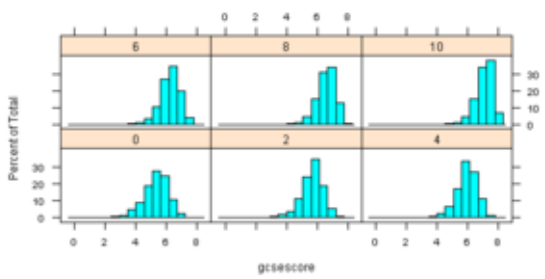
```
> pl <- histogram(~gcsescore | factor(score), data = Chem97)
> print(pl)
```

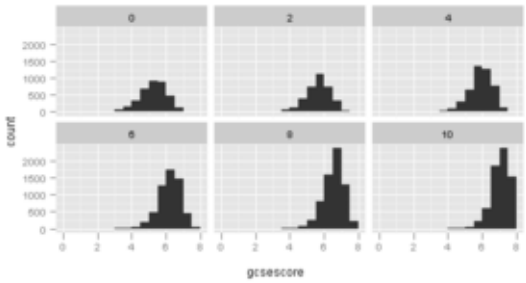
ggplot2

```
> pg <- ggplot(Chem97, aes(gcsescore)) + geom_histogram(binwidth = 0.5) +
+     facet_wrap(~score)
> print(pg)
```

Note
ggplot2 uses counts, not percentages by default.

Note
ggplot2 plots the facets starting from top-left, lattice starts from bottom-left.





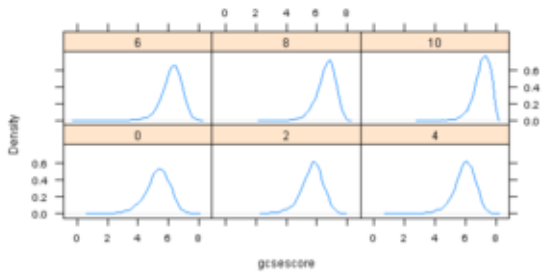
1.2 Figure 1.2

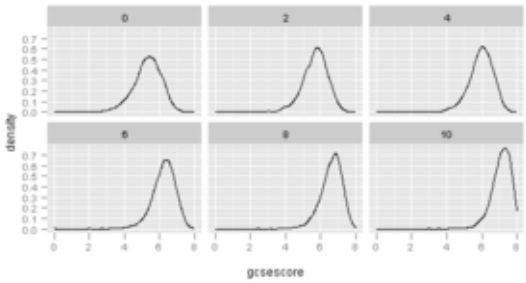
lattice

```
> pl <- densityplot(~gcsescore | factor(score), data = Chem97,
+     plot.points = FALSE, ref = TRUE)
> print(pl)
```

ggplot2

```
> pg <- ggplot(Chem97, aes(gcsescore)) + stat_density(geom = "path",
+     position = "identity") + facet_wrap(~score)
> print(pg)
```





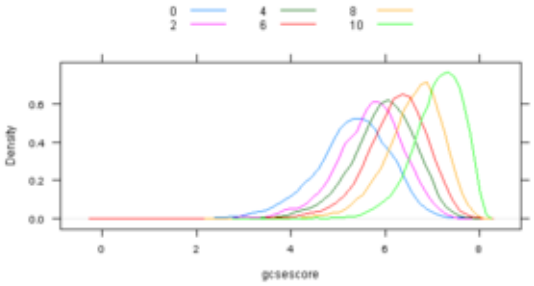
1.3 Figure 1.3

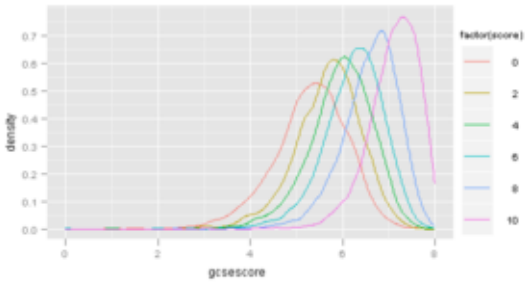
lattice

```
> pl <- densityplot(~gcsescore, data = Chem97, groups = score,
+     plot.points = FALSE, ref = TRUE, auto.key = list(columns = 3))
> print(pl)
```

ggplot2

```
> pg <- ggplot(Chem97, aes(gcsescore)) + stat_density(geom = "path",
+     position = "identity", aes(colour = factor(score)))
> print(pg)
```


0
2
4
6
8
10
Density
gcsescore


density
gcsescore
factor(score)
0
2
4
6
8
10

Chapter 2

A Technical Overview of lattice

TOPICS COVERED:

- The formula interface
- object dimensions and physical layout
- Annotation
- Scales and Axes
- Panel functions

2.1 Figure 2.1

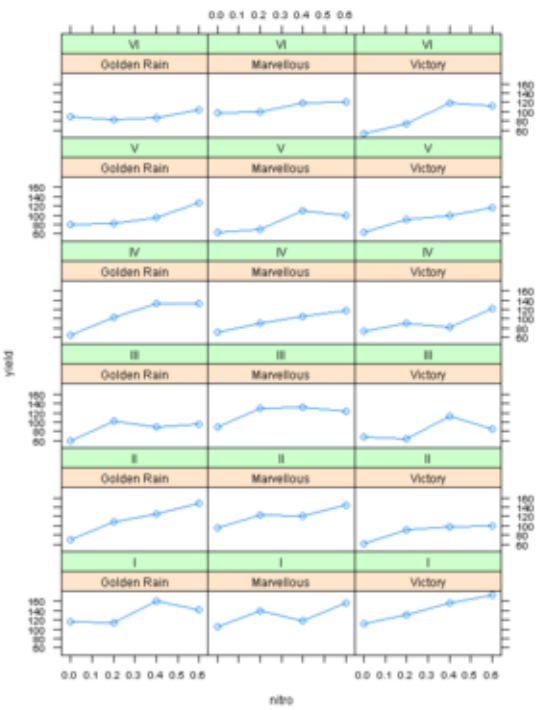
```
> library(lattice)
> library(ggplot2)
> data(Oats, package = "MEMSS")
```

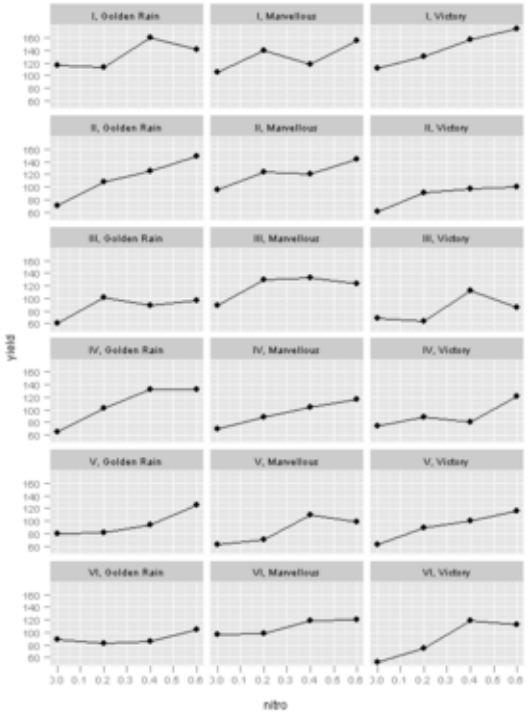
lattice

```
> tp1.oats <- xyplot(yield ~ nitro | Variety + Block, data = Oats,
+     type = "o")
> print(tp1.oats)
```

ggplot2

```
> pg.oats <- ggplot(Oats, aes(nitro, yield)) + geom_line() + geom_point() +
+     facet_wrap(~Block + Variety, ncol = 3)
> print(pg.oats)
```





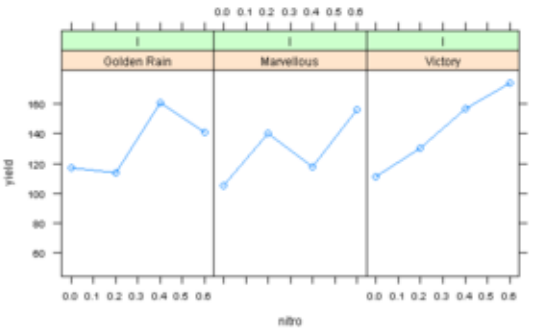
2.2 Figure 2.2

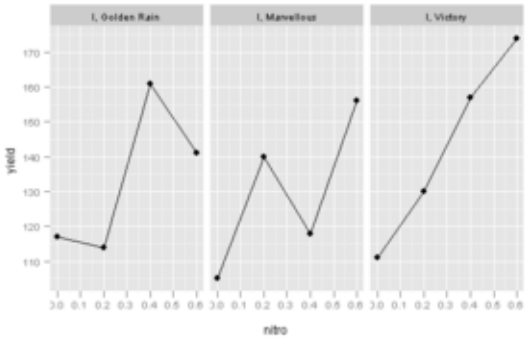
lattice

```
> print(tp1.oats[, 1])
```

ggplot2

```
> pg <- pg.oats %+% subset(Oats, Block == "I")
> print(pg)
```





2.3 Figure 2.3

lattice

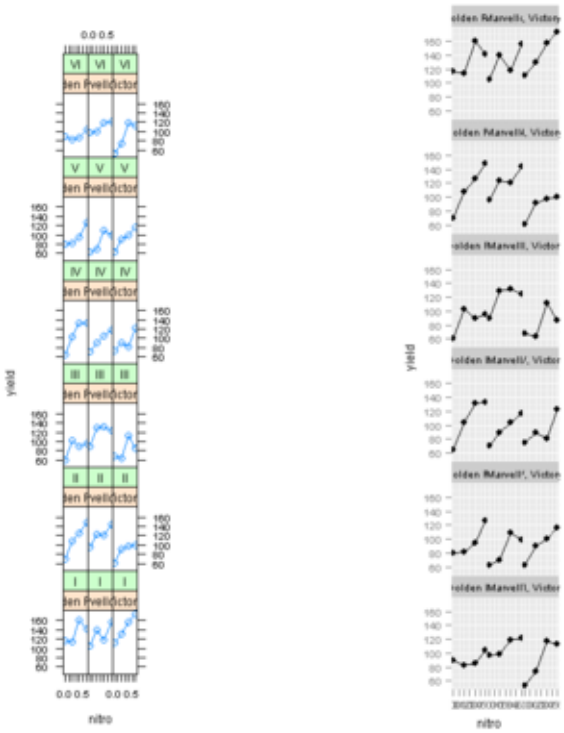
```
> pl <- update(tp1.oats, aspect = "xy")
> print(pl)
```

ggplot2

```
> pg <- pg.oats + opts(panel.margin = unit(0, "lines"))
> print(pg)
```

Note
Currently it is not possible to manipulate the facet aspect ratio. A workaround is to tweak the output image dimensions when saving the output graph to a file.

Note
ggplot2 orders facets in the opposite direction compared to lattice.



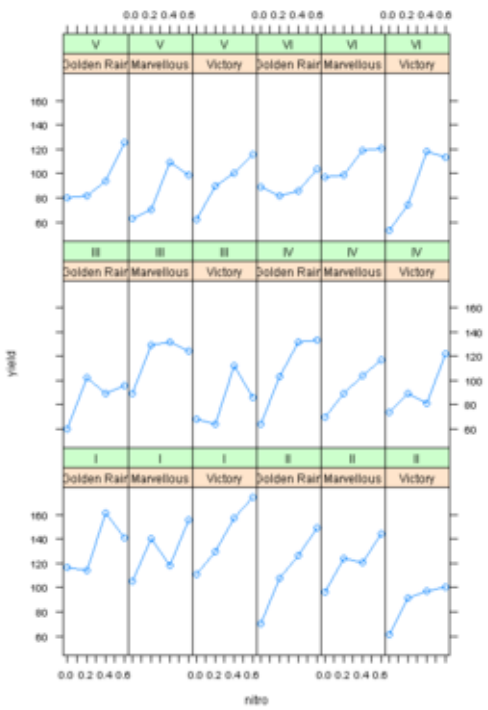
2.4 Figure 2.4

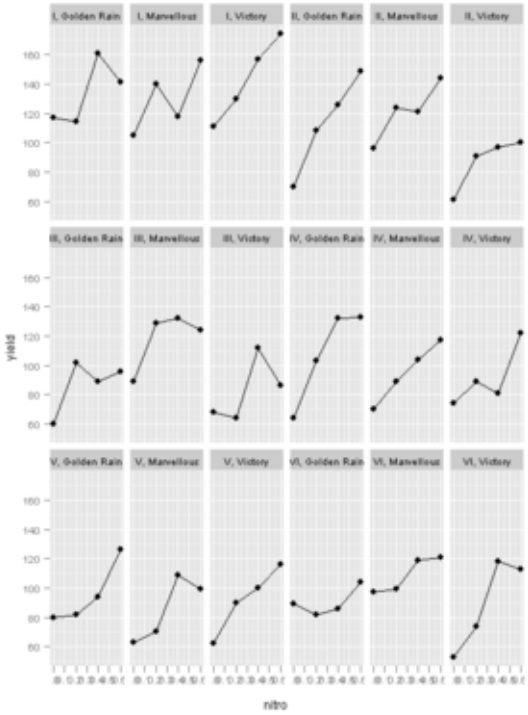
lattice

```
> pl <- update(tp1.oats, aspect = "xy", layout = c(0, 18))
> print(pl)
```

ggplot2

```
> pg <- pg.oats + facet_wrap(~Block + Variety, ncol = 6)
> print(pg)
```





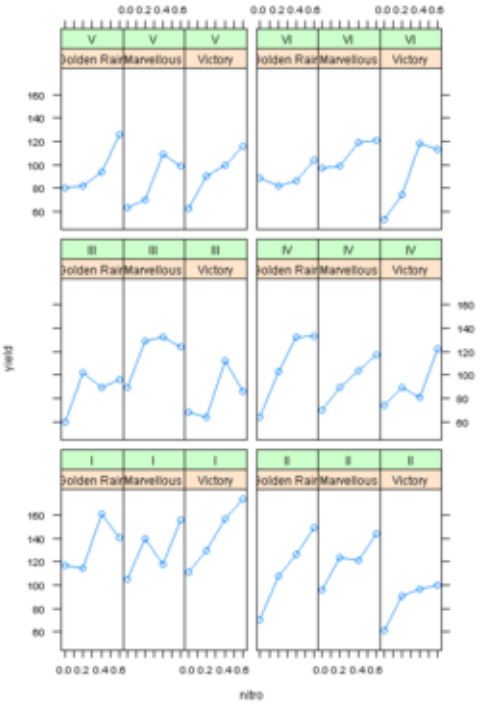
2.5 Figure 2.5

lattice

```
> pl <- update(tp1.oats, aspect = "xy", layout = c(0, 18), between = list(x = c(0,
+     0, 0.5), y = 0.5))
> print(pl)
```

ggplot2

```
Grouping of individual facets not possible in ggplot2.
```



2.6 Figure 2.6

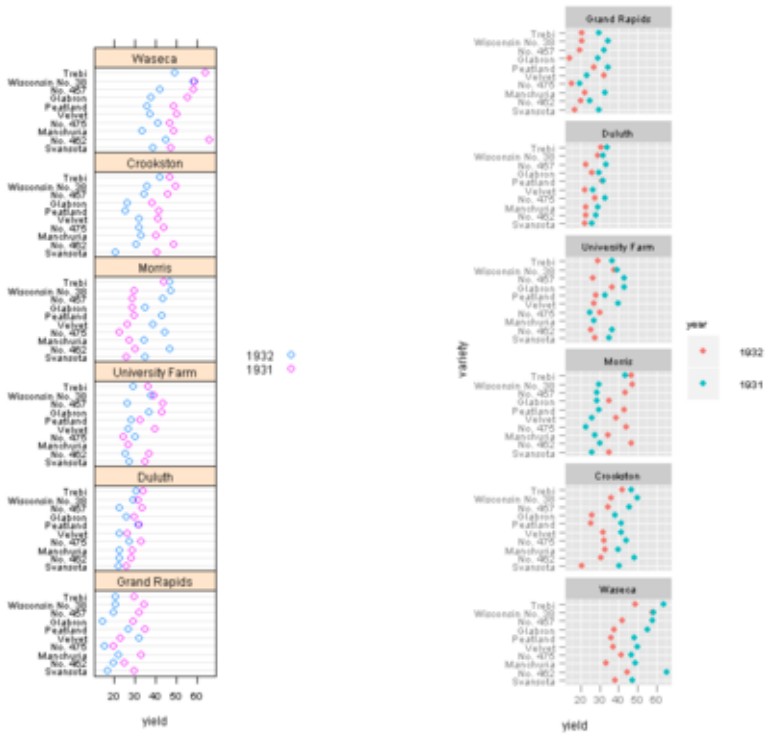
lattice

```
> pl <- dotplot(variety ~ yield | site, barley, layout = c(1, 6),
+     aspect = c(0.7), groups = year, auto.key = list(space = "right"))
> print(pl)
```

ggplot2

```
> pg <- ggplot(barley, aes(yield, variety, colour = year)) + geom_point() +
+     facet_wrap(~site, ncol = 1)
> print(pg)
```

Note
Currently it is not possible to manipulate the facet aspect ratio. A workaround is to tweak the output image dimensions when saving the output graph to a file.



2.7 Figure 2.7

lattice

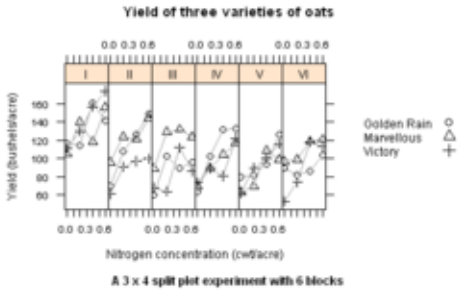
```
> key.variety <- list(space = "right", text = list(levels(Oats$Variety)),
+     points = list(pch = 1:3, col = "black"))
> pl <- xyplot(yield ~ nitro | Block, Oats, aspect = "xy", type = "o",
+     groups = Variety, key = key.variety, lty = 1, pch = 1:3,
+     col.line = "darkgrey", col.symbol = "black", xlab = "Nitrogen concentration (cwt/acre ↩
    )",
+     ylab = "Yield (bushels/acre)", main = "Yield of three varieties of oats",
+     sub = "A 3 x 4 split plot experiment with 6 blocks")
> print(pl)
```

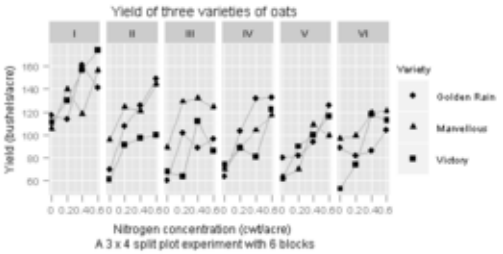
ggplot2

```
> p <- ggplot(Oats, aes(nitro, yield, group = Variety, shape = Variety))
> pg <- p + geom_line(colour = "darkgrey") + geom_point() + facet_grid(~Block) +
+     scale_x_continuous(breaks = seq(0, 0.6, by = 0.2), labels = seq(0,
+         0.6, by = 0.2)) + opts(title = "Yield of three varieties of oats") +
+     labs(x = "Nitrogen concentration (cwt/acre) \n A 3 x 4 split plot experiment with 6  ↩
    blocks",
+         y = "Yield (bushels/acre)")
> print(pg)
```

Note
ggplot2 does not have the subtitle functionality. Nevertheless, very similar result can be achieved by splitting the x-axis label into two rows.

Note
scale_x_continuous() is used to manually set the axis breaks and labels. Otherwise these would be illegible like on Figures 2.3 & 2.4 above.





2.8 Figure 2.8

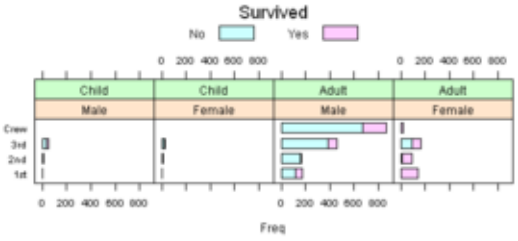
lattice

```
> pl <- barchart(Class ~ Freq | Sex + Age, data = as.data.frame(Titanic),
+     groups = Survived, stack = TRUE, layout = c(4, 1), auto.key = list(title = "Survived ↩
    ",
+         columns = 2))
> print(pl)
```

ggplot2

```
> pg.titanic <- ggplot(as.data.frame(Titanic), aes(Class, Freq,
+     fill = Survived)) + geom_bar(stat = "identity") + facet_wrap(~Age +
+     Sex, nrow = 1) + coord_flip()
> print(pg.titanic)
```

Note
Currently it is not possible to manipulate the facet aspect ratio. A workaround is to tweak the output image dimensions when saving the output graph to a file.



2.9 Figure 2.9

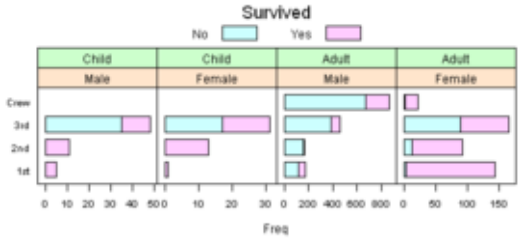
lattice

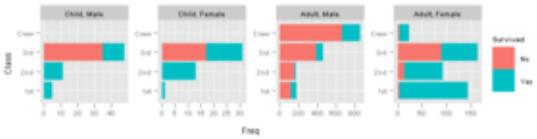
```
> pl <- barchart(Class ~ Freq | Sex + Age, data = as.data.frame(Titanic),
+     groups = Survived, stack = TRUE, layout = c(4, 1), auto.key = list(title = "Survived ↩
    ",
+         columns = 2), scales = list(x = "free"))
> print(pl)
```

ggplot2

```
> pg <- pg.titanic + facet_wrap(~Age + Sex, nrow = 1, scales = "free")
> print(pg)
```

Note
Currently it is not possible to manipulate the facet aspect ratio. A workaround is to tweak the output image dimensions when saving the output graph to a file.





Chapter 3

Visualizing Univariate Distributions

TOPICS COVERED:

- Kernel Density Plot, Histogram
- Theoretical Q-Q plot, Empirical CDF plot
- Two-sample Q-Q plot
- Comparative Box and Whisker plots, Violin plots
- Comparative Strip charts
- Discrete distributions

3.1 Figure 3.1

```
> library(lattice)
> library(ggplot2)
> data(Oats, package = "MEMSS")
```

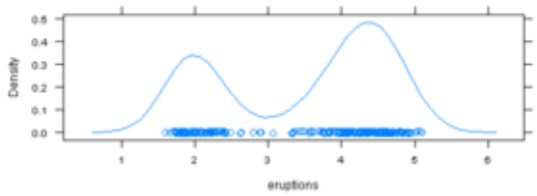
lattice

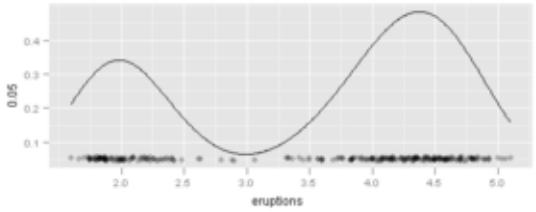
```
> pl <- densityplot(~eruptions, data = faithful)
> print(pl)
```

ggplot2

```
> p <- ggplot(faithful, aes(eruptions))
> pg <- p + stat_density(geom = "path", position = "identity") +
+     geom_point(aes(y = 0.05), position = position_jitter(height = 0.005),
+         alpha = 0.25)
> print(pg)
```

Note
y = 0.05 specifies the position of jitter on y-axis.





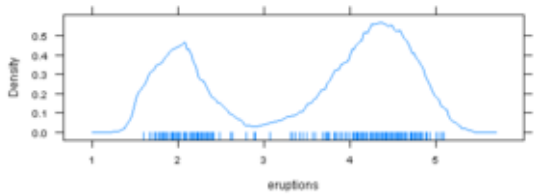
3.2 Figure 3.2

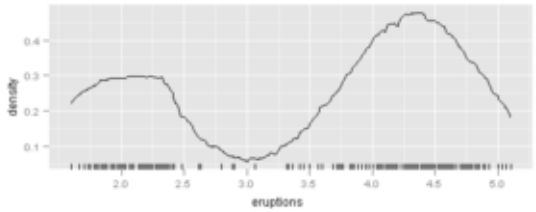
lattice

```
> pl <- densityplot(~eruptions, data = faithful, kernel = "rect",
+     bw = 0.2, plot.points = "rug", n = 200)
> print(pl)
```

ggplot2

```
> pg <- p + stat_density(geom = "path", kernel = "rect", position = "identity",
+     bw = 0.2) + geom_rug()
> print(pg)
```





3.3 Figure 3.3

```
> library("latticeExtra")
> data(gvhd10)
```

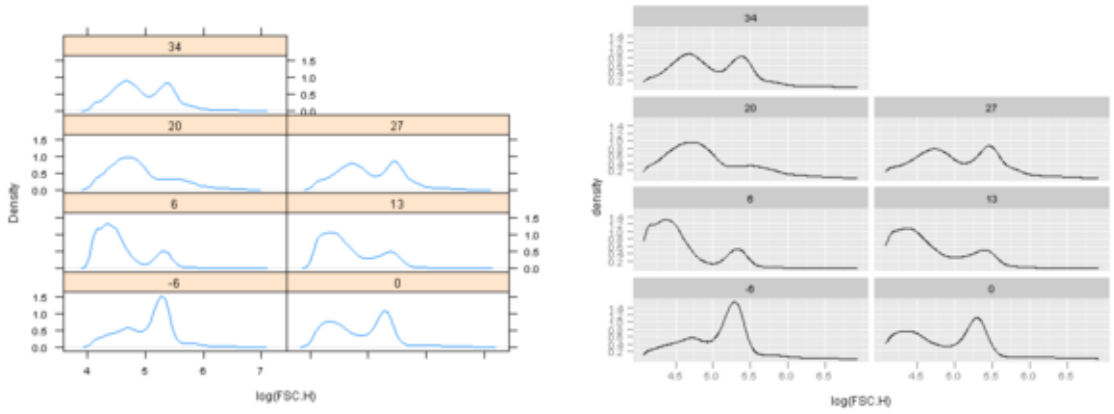
lattice

```
> pl <- densityplot(~log(FSC.H) | Days, data = gvhd10, plot.points = FALSE,
+     ref = TRUE, layout = c(2, 4))
> print(pl)
```

ggplot2

```
> p <- ggplot(gvhd10, aes(log(FSC.H)))
> pg <- p + stat_density(geom = "path", position = "identity") +
+     facet_wrap(~Days, ncol = 2, as.table = FALSE)
> print(pg)
```

Note
as.table = FALSE changes the default orders of the facets.



3.4 Figure 3.4

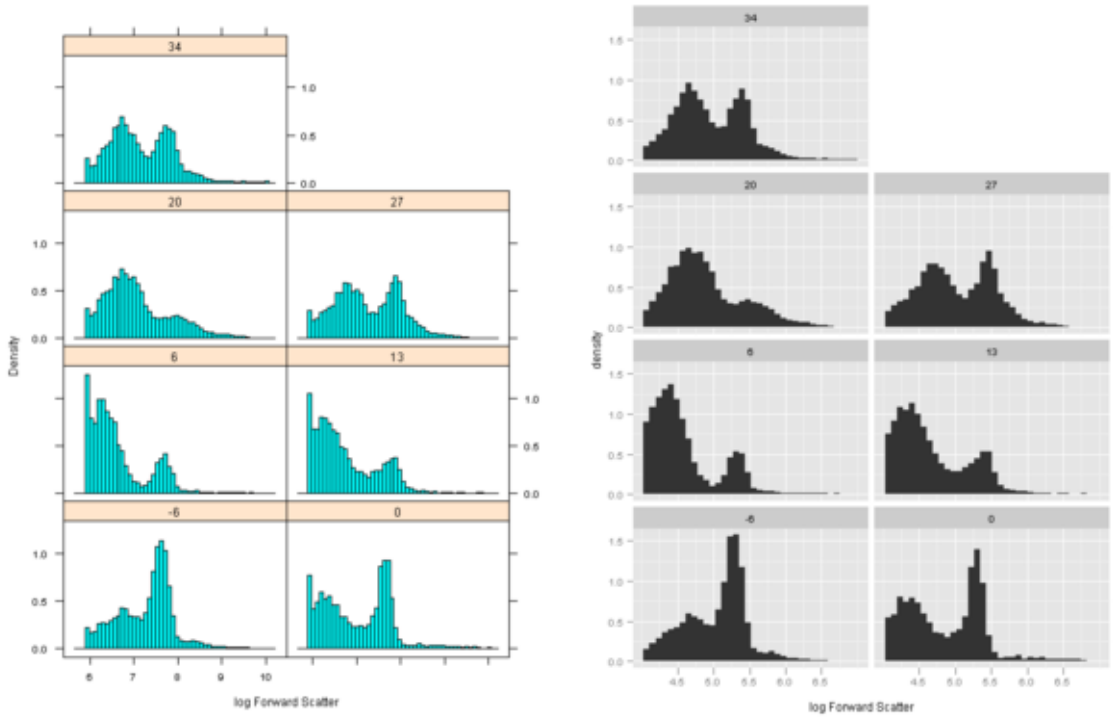
lattice

```
> pl <- histogram(~log2(FSC.H) | Days, gvhd10, xlab = "log Forward Scatter",
+     type = "density", nint = 50, layout = c(2, 4))
> print(pl)
```

ggplot2

```
> pg <- p + geom_histogram(aes(y = ..density..), binwidth = diff(range(log2(gvhd10$FSC.H))) ↩
    /50) +
+     facet_wrap(~Days, ncol = 2, as.table = FALSE) + xlab("log Forward Scatter")
> print(pg)
```

Note
ggplot2 uses binwidth by default, therefore the number of bins needs to be presented in terms of binwidth.



3.5 Figure 3.5

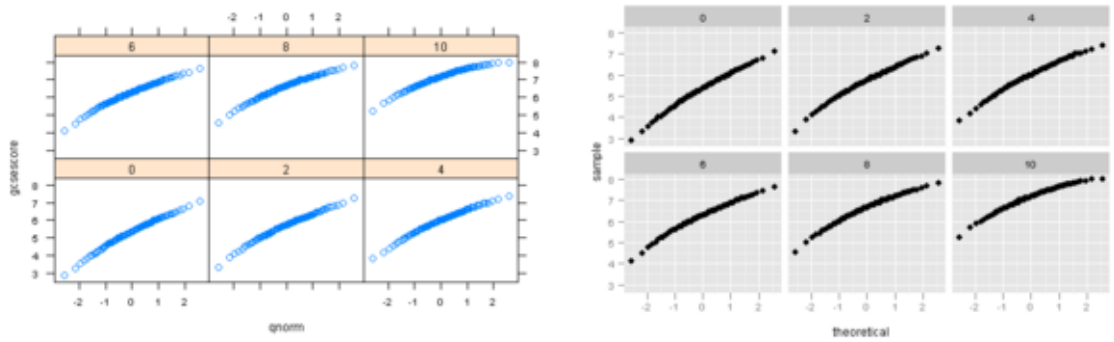
```
> data(Chem97, package = "mlmRev")
```

lattice

```
> pl <- qqmath(~gcsescore | factor(score), data = Chem97, f.value = ppoints(100))
> print(pl)
```

ggplot2

```
> p <- ggplot(Chem97)
> pg <- p + geom_point(aes(sample = gcsescore), stat = "qq", quantiles = ppoints(100)) +
+     facet_wrap(~score)
> print(pg)
```



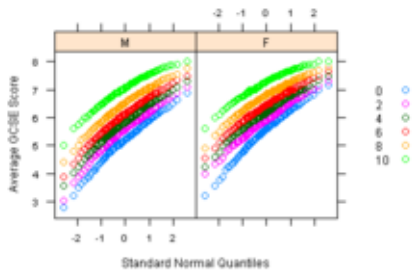
3.6 Figure 3.6

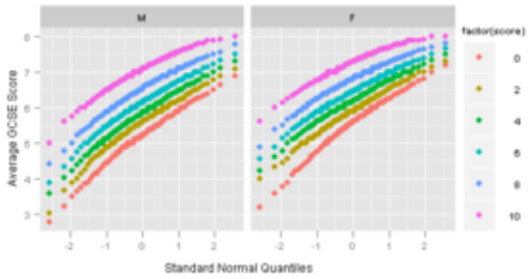
lattice

```
> pl <- qqmath(~gcsescore | gender, Chem97, groups = score, aspect = "xy",
+     f.value = ppoints(100), auto.key = list(space = "right"),
+     xlab = "Standard Normal Quantiles", ylab = "Average GCSE Score")
> print(pl)
```

ggplot2

```
> pg <- p + geom_point(aes(sample = gcsescore, colour = factor(score)),
+     stat = "qq", quantiles = ppoints(100)) + facet_grid(~gender) +
+     opts(aspect.ratio = 1) + scale_x_continuous("Standard Normal Quantiles") +
+     scale_y_continuous("Average GCSE Score")
> print(pg)
```





3.7 Figure 3.7

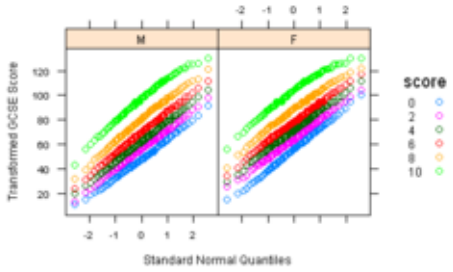
lattice

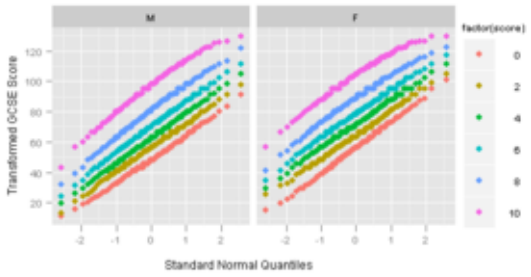
```
> Chem97.mod <- transform(Chem97, gcsescore.trans = gcsescore^2.34)
```

```
> pl <- qqmath(~gcsescore.trans | gender, Chem97.mod, groups = score,
+     f.value = ppoints(100), aspect = "xy", auto.key = list(space = "right",
+         title = "score"), xlab = "Standard Normal Quantiles",
+     ylab = "Transformed GCSE Score")
> print(pl)
```

ggplot2

```
> pg <- p + geom_point(aes(sample = gcsescore^2.34, colour = factor(score)),
+     stat = "qq", quantiles = ppoints(100)) + facet_grid(~gender) +
+     opts(aspect.ratio = 1) + scale_x_continuous("Standard Normal Quantiles") +
+     scale_y_continuous("Transformed GCSE Score")
> print(pg)
```





3.8 Figure 3.8

```
> library("latticeExtra")
```

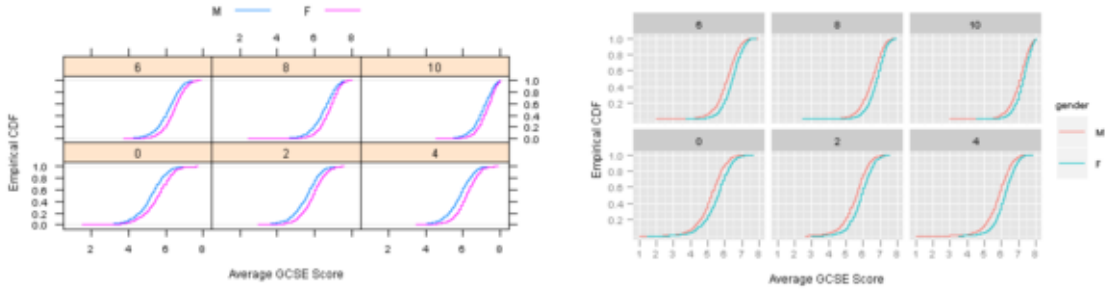
lattice

```
> pl <- ecdfplot(~gcsescore | factor(score), data = Chem97, groups = gender,
+     auto.key = list(columns = 2), subset = gcsescore > 0, xlab = "Average GCSE Score")
> print(pl)
```

ggplot2

```
> Chem97.ecdf <- ddply(Chem97, .(score, gender), transform, ecdf = ecdf(gcsescore)( ↩
    gcsescore))
```

```
> p <- ggplot(Chem97.ecdf, aes(gcsescore, ecdf, colour = gender))
> pg <- p + geom_step(subset = .(gcsescore > 0)) + facet_wrap(~score,
+     as.table = F) + xlab("Average GCSE Score") + ylab("Empirical CDF")
> print(pg)
```



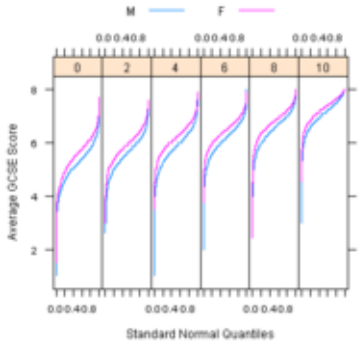
3.9 Figure 3.9

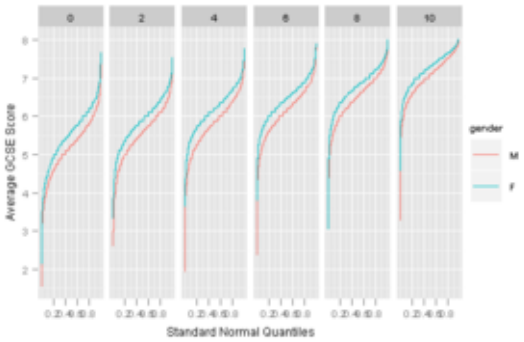
lattice

```
> pl <- qqmath(~gcsescore | factor(score), data = Chem97, groups = gender,
+     auto.key = list(points = FALSE, lines = TRUE, columns = 2),
+     subset = gcsescore > 0, type = "l", distribution = qunif,
+     prepanel = prepanel.qqmathline, aspect = "xy", xlab = "Standard Normal Quantiles",
+     ylab = "Average GCSE Score")
> print(pl)
```

ggplot2

```
> p <- ggplot(Chem97, aes(sample = gcsescore, colour = gender))
> pg <- p + geom_path(subset = .(gcsescore > 0), stat = "qq", distribution = qunif) +
+     facet_grid(~score) + scale_x_continuous("Standard Normal Quantiles") +
+     scale_y_continuous("Average GCSE Score")
> print(pg)
```





3.10 Figure 3.10

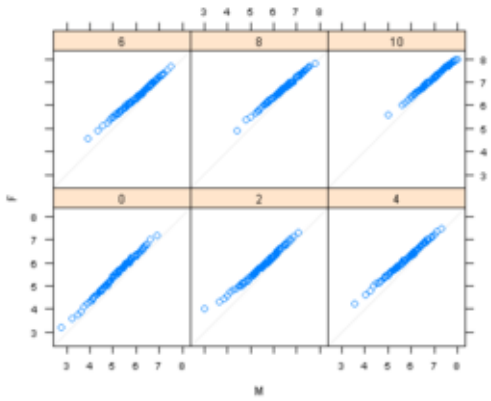
lattice

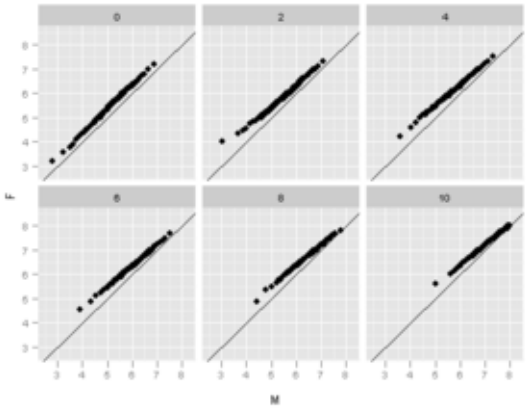
```
> pl <- qq(gender ~ gcsescore | factor(score), Chem97, f.value = ppoints(100),
+     aspect = 1)
> print(pl)
```

ggplot2

```
> q <- function(x, probs = ppoints(100)) {
+     data.frame(q = probs, value = quantile(x, probs))
+ }
> Chem97.q <- ddply(Chem97, c("gender", "score"), function(df) q(df$gcsescore))
> Chem97.df <- recast(Chem97.q, score + q ~ gender, id.var = 1:3)
```

```
> pg <- ggplot(Chem97.df) + geom_point(aes(M, F)) + geom_abline() +
+     facet_wrap(~score) + coord_equal()
> print(pg)
```





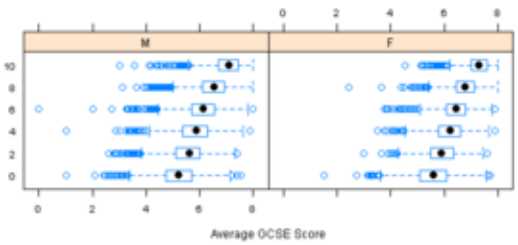
3.11 Figure 3.11

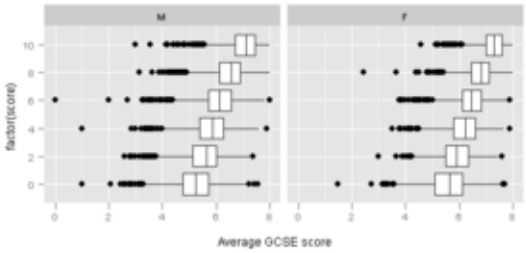
lattice

```
> pl <- bwplot(factor(score) ~ gcsescore | gender, data = Chem97,
+     xlab = "Average GCSE Score")
> print(pl)
```

ggplot2

```
> pg <- ggplot(Chem97, aes(factor(score), gcsescore)) + geom_boxplot() +
+     coord_flip() + ylab("Average GCSE score") + facet_wrap(~gender)
> print(pg)
```





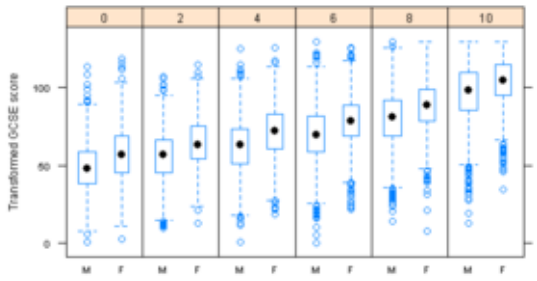
3.12 Figure 3.12

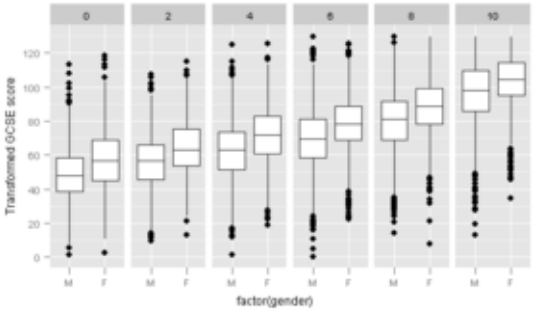
lattice

```
> pl <- bwplot(gcsescore^2.34 ~ gender | factor(score), Chem97,
+     varwidth = TRUE, layout = c(6, 1), ylab = "Transformed GCSE score")
> print(pl)
```

ggplot2

```
> p <- ggplot(Chem97, aes(factor(gender), gcsescore^2.34))
> pg <- p <- ggplot(Chem97, aes(factor(gender), gcsescore^2.34)) +
+     geom_boxplot() + facet_grid(~score) + ylab("Transformed GCSE score")
> print(pg)
```





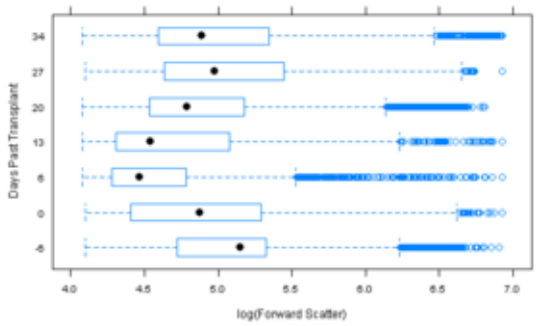
3.13 Figure 3.13

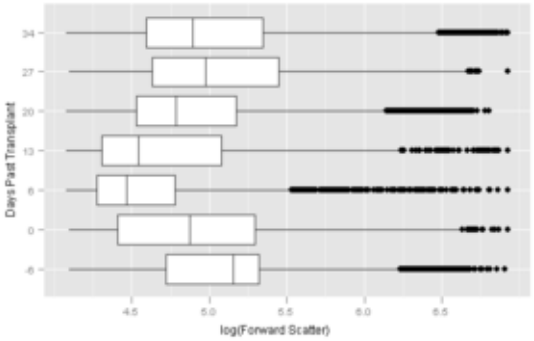
lattice

```
> pl <- bwplot(Days ~ log(FSC.H), data = gvhd10, xlab = "log(Forward Scatter)",
+     ylab = "Days Past Transplant")
> print(pl)
```

ggplot2

```
> p <- ggplot(gvhd10, aes(factor(Days), log(FSC.H)))
> pg <- p + geom_boxplot() + coord_flip() + labs(y = "log(Forward Scatter)",
+     x = "Days Past Transplant")
> print(pg)
```





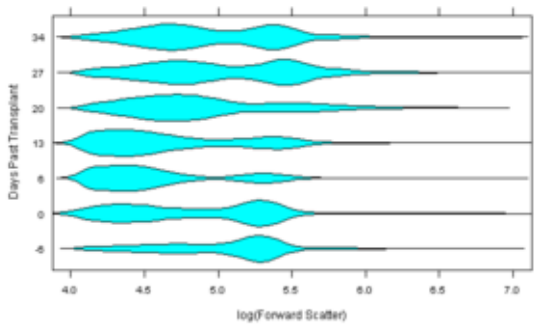
3.14 Figure 3.14

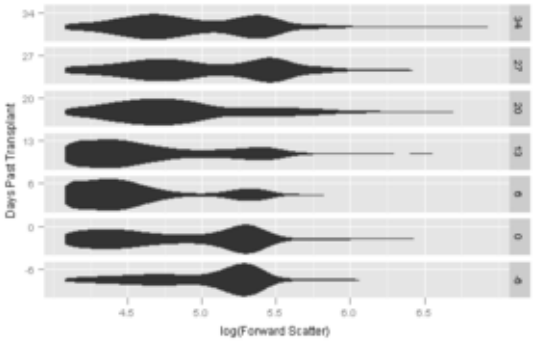
lattice

```
> pl <- bwplot(Days ~ log(FSC.H), gvhd10, panel = panel.violin,
+     box.ratio = 3, xlab = "log(Forward Scatter)", ylab = "Days Past Transplant")
> print(pl)
```

ggplot2

```
> p <- ggplot(gvhd10, aes(log(FSC.H), Days))
> pg <- p + geom_ribbon(aes(ymax = ..density.., ymin = -..density..),
+     stat = "density") + facet_grid(Days ~ ., as.table = F, scales = "free_y") +
+     labs(x = "log(Forward Scatter)", y = "Days Past Transplant")
> print(pg)
```





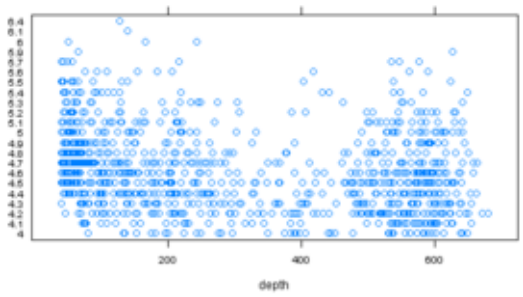
3.15 Figure 3.15

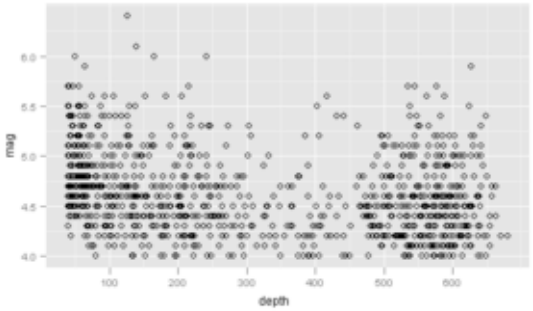
lattice

```
> pl <- stripplot(factor(mag) ~ depth, quakes)
> print(pl)
```

ggplot2

```
> pg <- ggplot(quakes) + geom_point(aes(depth, mag), shape = 1)
> print(pg)
```





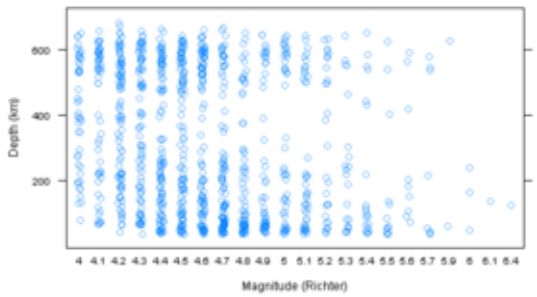
3.16 Figure 3.16

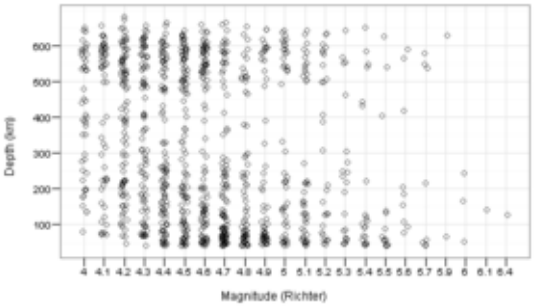
lattice

```
> pl <- stripplot(depth ~ factor(mag), quakes, jitter.data = TRUE,
+     alpha = 0.6, xlab = "Magnitude (Richter)", ylab = "Depth (km)")
> print(pl)
```

ggplot2

```
> p <- ggplot(quakes, aes(factor(mag), depth))
> pg <- p + geom_point(position = position_jitter(width = 0.15),
+     alpha = 0.6, shape = 1) + theme_bw() + xlab("Magnitude (Richter)") +
+     ylab("Depth (km)")
> print(pg)
```





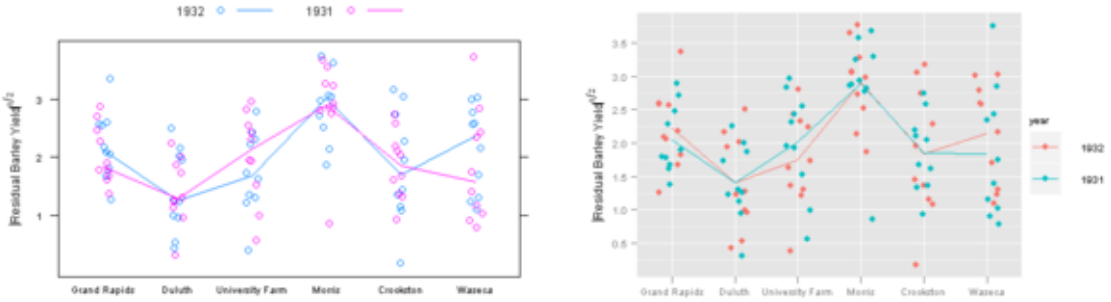
3.17 Figure 3.17

lattice

```
> pl <- stripplot(sqrt(abs(residuals(lm(yield ~ variety + year +
+     site)))) ~ site, data = barley, groups = year, jitter.data = TRUE,
+     auto.key = list(points = TRUE, lines = TRUE, columns = 2),
+     type = c("p", "a"), fun = median, ylab = expression(abs("Residual Barley Yield")^{
+         1/2
+     }))
> print(pl)
```

ggplot2

```
> p <- ggplot(barley, aes(site, sqrt(abs(residuals(lm(yield ~ variety +
+     year + site))))), colour = year, group = year))
> pg <- p + geom_jitter(position = position_jitter(width = 0.2)) +
+     geom_line(stat = "summary", fun.y = "mean") + labs(x = "",
+     y = expression(abs("Residual Barley Yield")^{
+         1/2
+     }))
> print(pg)
```



Chapter 4

Displaying Multiway Tables

TOPICS COVERED:

- Cleveland dot plot
- Bar chart
- Reordering factor levels

4.1 Figure 4.1

```
> library(lattice)
> library(ggplot2)
```

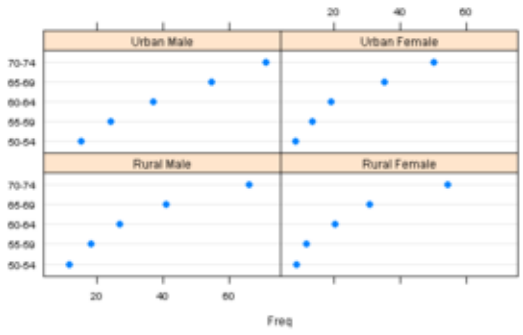
```
> data(VADeaths)
```

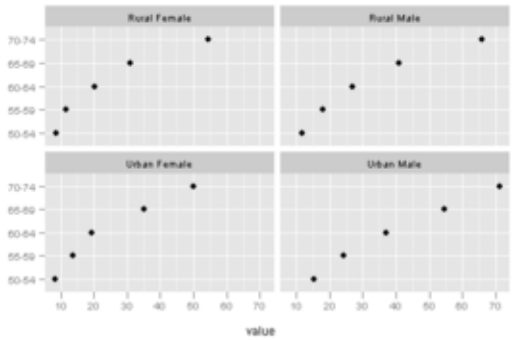
lattice

```
> pl <- dotplot(VADeaths, groups = FALSE)
> print(pl)
```

ggplot2

```
> pg <- ggplot(melt(VADeaths), aes(value, X1)) + geom_point() +
+     facet_wrap(~X2) + ylab("")
> print(pg)
```





4.2 Figure 4.2

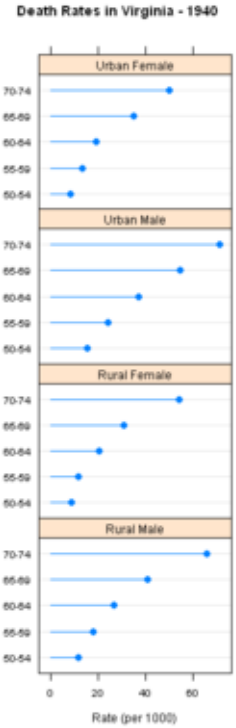
lattice

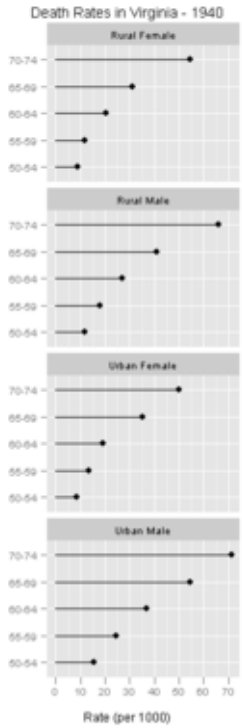
```
> pl <- dotplot(VADeaths, groups = FALSE, layout = c(1, 4), aspect = 0.7,
+     origin = 0, type = c("p", "h"), main = "Death Rates in Virginia - 1940",
+     xlab = "Rate (per 1000)")
> print(pl)
```

ggplot2

```
> p <- ggplot(melt(VADeaths), aes(x = 0, xend = value, y = X1,
+     yend = X1))
> pg <- p + geom_point(aes(value, X1)) + geom_segment() + facet_wrap(~X2,
+     ncol = 1) + labs(x = "Rate (per 1000)", y = "") + opts(title = "Death Rates in  ↩
    Virginia - 1940")
> print(pg)
```

Note
When using facet_wrap() it is not possible to manipulate the aspect ratio of facets. A workaround is to tweak the output image dimensions when saving the output graph to a file.





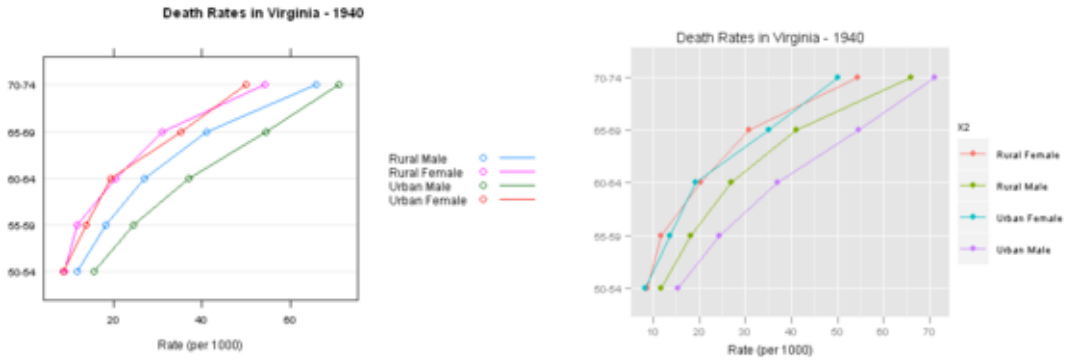
4.3 Figure 4.3

lattice

```
> pl <- dotplot(VADeaths, type = "o", auto.key = list(lines = TRUE,
+     space = "right"), main = "Death Rates in Virginia - 1940",
+     xlab = "Rate (per 1000)")
> print(pl)
```

ggplot2

```
> p <- ggplot(melt(VADeaths), aes(value, X1, colour = X2, group = X2))
> pg <- p + geom_point() + geom_line() + xlab("Rate (per 1000)") +
+     ylab("") + opts(title = "Death Rates in Virginia - 1940")
> print(pg)
```



4.4 Figure 4.4

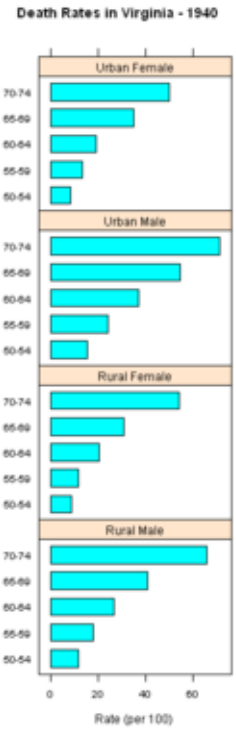
lattice

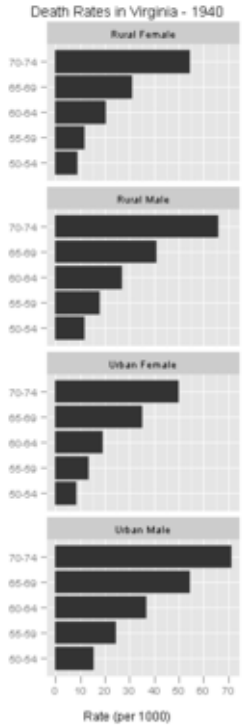
```
> pl <- barchart(VADeaths, groups = FALSE, layout = c(1, 4), aspect = 0.7,
+     reference = FALSE, main = "Death Rates in Virginia - 1940",
+     xlab = "Rate (per 100)")
> print(pl)
```

ggplot2

```
> p <- ggplot(melt(VADeaths), aes(X1, value))
> pg <- p + geom_bar(stat = "identity") + facet_wrap(~X2, ncol = 1) +
+     coord_flip() + xlab("") + ylab("Rate (per 1000)") + opts(title = "Death Rates in  ↩
    Virginia - 1940")
> print(pg)
```

Note
When using facet_wrap() it is not possible to manipulate the aspect ratio of facets. A workaround is to tweak the output image dimensions when saving the output graph to a file.





4.5 Figure 4.5

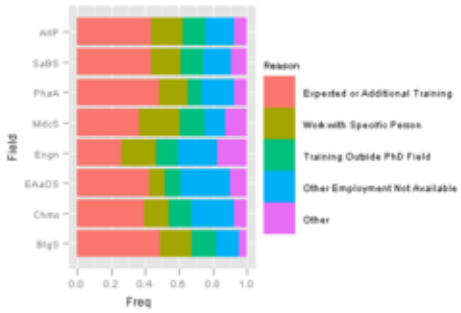
```
> data(postdoc, package = "latticeExtra")
```

lattice

```
> pl <- barchart(prop.table(postdoc, margin = 1), xlab = "Proportion",
+     auto.key = list(adj = 1))
> print(pl)
```

ggplot2

```
> pg <- ggplot(as.data.frame(postdoc), aes(Field, Freq, fill = Reason)) +
+     geom_bar(position = "fill") + coord_flip() + scale_x_discrete(formatter = "abbreviate ↩
    ")
> print(pg)
```



4.6 Figure 4.6

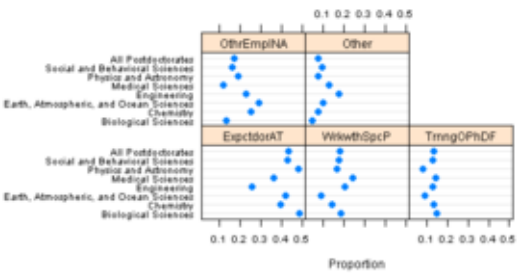
lattice

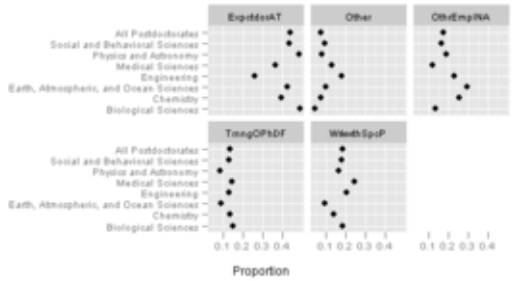
```
> pl <- dotplot(prop.table(postdoc, margin = 1), groups = FALSE,
+     xlab = "Proportion", par.strip.text = list(abbreviate = TRUE,
+         minlength = 10))
> print(pl)
```

ggplot2

```
> postdoc.df <- as.data.frame(prop.table(postdoc, margin = 1),
+     stringsAsFactors = FALSE)
> postdoc.df$Reason <- abbreviate(postdoc.df$Reason, minlength = 10)
```

```
> pg <- ggplot(postdoc.df, aes(Freq, Field)) + geom_point() + facet_wrap(~Reason) +
+     xlab("Proportion") + ylab("")
> print(pg)
```





4.7 Figure 4.7

lattice

```
> pl <- dotplot(prop.table(postdoc, margin = 1), groups = FALSE,
+     index.cond = function(x, y) median(x), xlab = "Proportion",
+     layout = c(1, 5), aspect = 0.6, scales = list(y = list(relation = "free",
+         rot = 0)), prepanel = function(x, y) {
+         list(ylim = levels(reorder(y, x)))
+     }, panel = function(x, y, ...) {
+         panel.dotplot(x, reorder(y, x), ...)
+     })
> print(pl)
```

ggplot2

```
Sorting each facets separately is not possible in ggplot2.
```

4.8 Figure 4.8

```
> data(Chem97, package = "mlmRev")
```

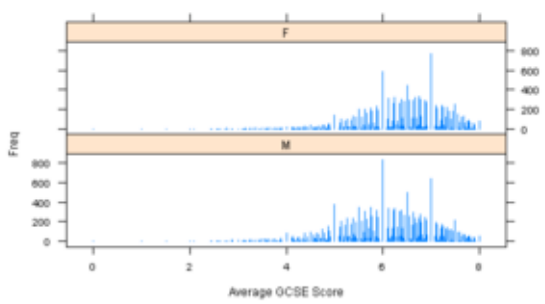
lattice

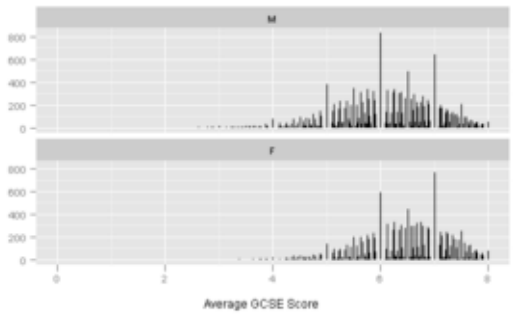
```
> gcsescore.tab <- xtabs(~gcsescore + gender, Chem97)
> gcsescore.df <- as.data.frame(gcsescore.tab)
> gcsescore.df$gcsescore <- as.numeric(as.character(gcsescore.df$gcsescore))
```

```
> pl <- xyplot(Freq ~ gcsescore | gender, data = gcsescore.df,
+     type = "h", layout = c(1, 2), xlab = "Average GCSE Score")
> print(pl)
```

ggplot2

```
> pg <- ggplot(Chem97, aes(gcsescore)) + geom_linerange(aes(ymin = 0,
+     ymax = ..count..), stat = "bin", binwidth = 0.005) + facet_wrap(~gender,
+     ncol = 1) + xlab("Average GCSE Score") + ylab("")
> print(pg)
```





4.9 Figure 4.9

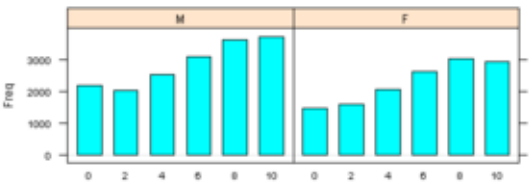
lattice

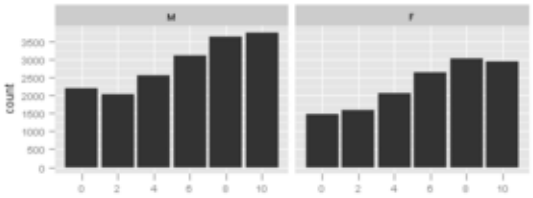
```
> score.tab <- xtabs(~score + gender, Chem97)
> score.df <- as.data.frame(score.tab)
```

```
> pl <- barchart(Freq ~ score | gender, score.df, origin = 0)
> print(pl)
```

ggplot2

```
> pg <- ggplot(Chem97, aes(factor(score))) + geom_bar(aes(y = ..count..),
+     stat = "bin") + facet_grid(. ~ gender) + xlab("")
> print(pg)
```





Chapter 5

Scatter Plots and Extensions

TOPICS COVERED:

- The standard scatter plot
- Using subscripts
- Using the type argument
- Variants for large data
- Scatter plot matrix
- Parallel coordinate plot

5.1 Figure 5.1

```
> library(lattice)
> library(ggplot2)
```

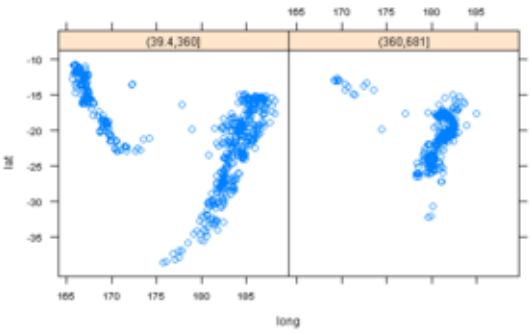
lattice

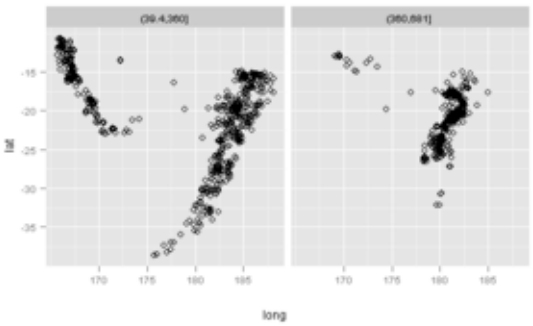
```
> pl <- xyplot(lat ~ long | cut(depth, 2), data = quakes)
> print(pl)
```

ggplot2

```
> quakes$Depth <- with(quakes, cut(depth, 2))
```

```
> pg <- ggplot(quakes, aes(long, lat)) + geom_point(shape = 1) +
+     facet_grid(~Depth) + opts(aspect.ratio = 1)
> print(pg)
```





5.2 Figure 5.2

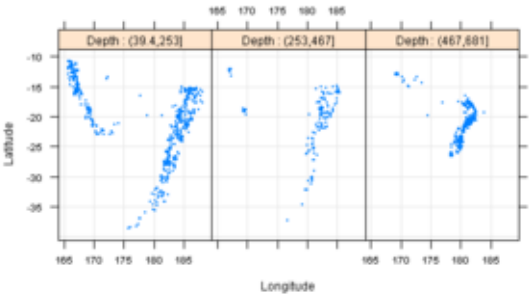
lattice

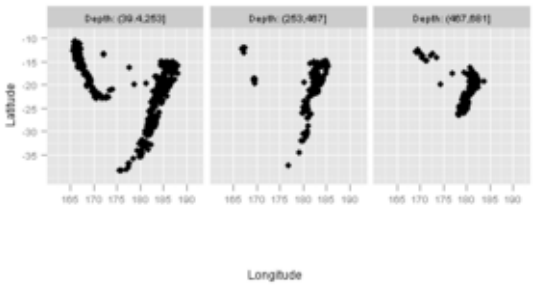
```
> pl <- xyplot(lat ~ long | cut(depth, 3), data = quakes, aspect = "iso",
+     pch = ".", cex = 2, type = c("p", "g"), xlab = "Longitude",
+     ylab = "Latitude", strip = strip.custom(strip.names = TRUE,
+         var.name = "Depth"))
> print(pl)
```

ggplot2

```
> quakes$Depth <- with(quakes, cut(depth, 3))
```

```
> pg <- ggplot(quakes, aes(long, lat)) + geom_point() + facet_grid(~Depth,
+     labeller = label_both) + coord_equal() + labs(x = "Longitude",
+     y = "Latitude")
> print(pg)
```





5.3 Figure 5.3

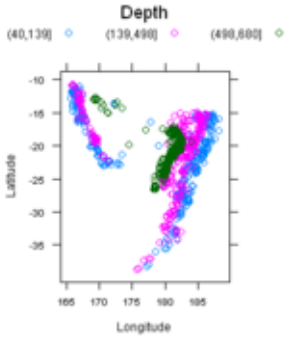
lattice

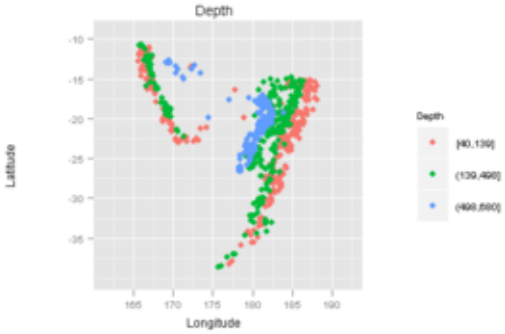
```
> pl <- xyplot(lat ~ long, data = quakes, aspect = "iso", groups = cut(depth,
+     breaks = quantile(depth, ppoints(4, 1))), auto.key = list(columns = 3,
+     title = "Depth"), xlab = "Longitude", ylab = "Latitude")
> print(pl)
```

ggplot2

```
> quakes$Depth <- with(quakes, cut(depth, breaks = quantile(depth,
+     ppoints(4, 1)), include.lowest = TRUE))
```

```
> pg <- ggplot(quakes, aes(long, lat, colour = Depth)) + geom_point() +
+     coord_equal() + labs(x = "Longitude", y = "Latitude") + opts(title = "Depth")
> print(pg)
```





5.4 Figure 5.4

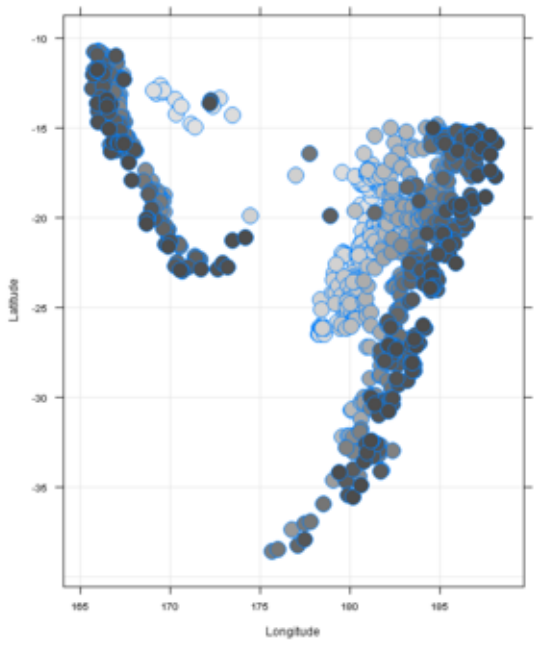
lattice

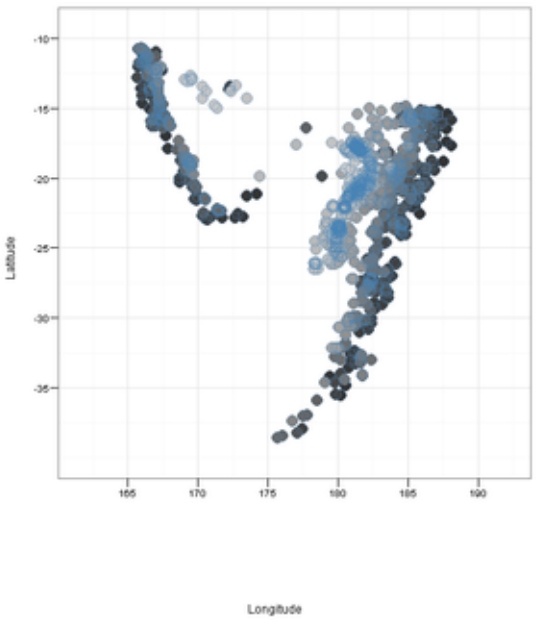
```
> depth.col <- gray.colors(100)[cut(quakes$depth, 100, label = FALSE)]
> depth.ord <- rev(order(quakes$depth))
```

```
> pl <- xyplot(lat ~ long, data = quakes[depth.ord, ], aspect = "iso",
+     type = c("p", "g"), pch = 21, fill = depth.col[depth.ord],
+     cex = 2, xlab = "Longitude", ylab = "Latitude")
> print(pl)
```

ggplot2

```
> pg <- ggplot(quakes, aes(long, lat, colour = factor(cut(quakes$depth,
+     100, label = FALSE)))) + geom_point(size = 4) + geom_point(size = 4,
+     shape = 1, colour = "steelblue", alpha = 0.4) + labs(x = "Longitude",
+     y = "Latitude") + scale_colour_grey() + theme_bw() + opts(legend.position = "none") +
+     coord_equal()
> print(pg)
```





5.5 Figure 5.5

lattice

```
> quakes$Magnitude <- equal.count(quakes$mag, 4)
> quakes$color <- depth.col
> quakes.ordered <- quakes[depth.ord, ]
```

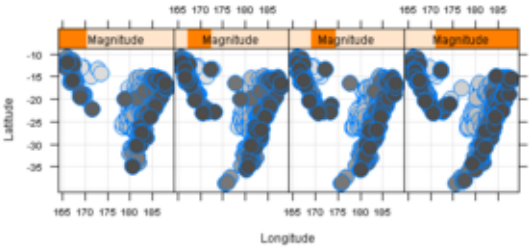
```
> pl <- xyplot(lat ~ long | Magnitude, data = quakes.ordered, aspect = "iso",
+     fill.color = quakes.ordered$color, cex = 2, panel = function(x,
+         y, fill.color, ..., subscripts) {
+         fill <- fill.color[subscripts]
+         panel.grid(h = -1, v = -1)
+         panel.xyplot(x, y, pch = 21, fill = fill, ...)
+     }, xlab = "Longitude", ylab = "Latitude")
> print(pl)
```

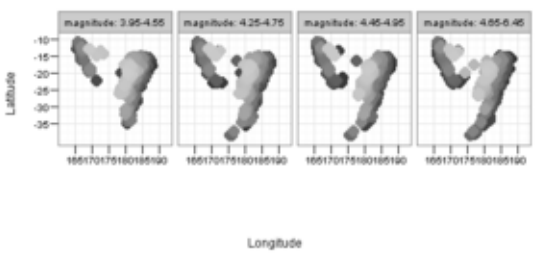
ggplot2

```
> fn <- function(data = quakes$mag, number = 4, ...) {
+     intrv <<- as.data.frame(co.intervals(data, number, ...))
+     mag <- sort(unique(data))
+     intervals <- ldply(mag, function(x) {
+         t(as.numeric(x < intrv$V2 & x > intrv$V1))
+     })
+     tmp <- melt(cbind(mag, intervals), id.var = 1)
+     tmp[tmp$value > 0, 1:2]
+ }
> quakes.ordered <- merge(quakes, fn())
> intrv <- with(intrv, paste(V1, V2, sep = "-"))
> quakes.ordered <- rename(quakes.ordered, c(variable = "magnitude"))
> quakes.ordered$magnitude <- factor(quakes.ordered$magnitude,
+     labels = intrv)
```

```
> pg <- ggplot(quakes.ordered, aes(long, lat, colour = factor(cut(depth,
+     100, label = FALSE)))) + geom_point(size = 4) + facet_grid(~magnitude,
+     labeller = label_both) + scale_colour_grey() + theme_bw() +
+     labs(x = "Longitude", y = "Latitude") + opts(legend.position = "none") +
+     coord_equal()
> print(pg)
```

Note
Custom wrapper function fn() used to break the data into intervals.





5.6 Figure 5.6

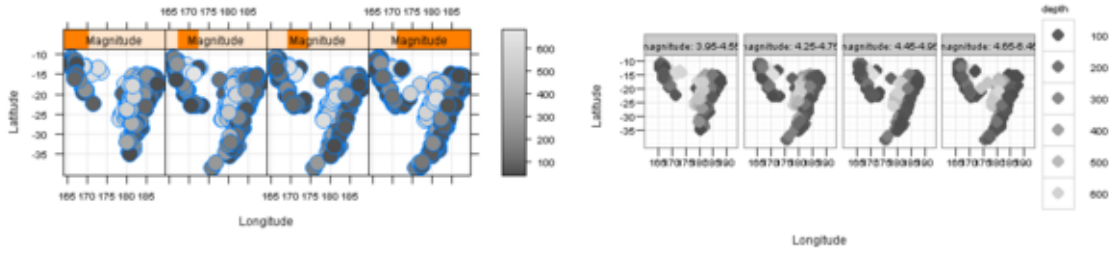
lattice

```
> depth.breaks <- do.breaks(range(quakes.ordered$depth), 50)
> quakes.ordered$color <- level.colors(quakes.ordered$depth, at = depth.breaks,
+     col.regions = gray.colors)
```

```
> pl <- xyplot(lat ~ long | Magnitude, data = quakes.ordered, aspect = "iso",
+     groups = color, cex = 2, panel = function(x, y, groups, ...,
+         subscripts) {
+         fill <- groups[subscripts]
+         panel.grid(h = -1, v = -1)
+         panel.xyplot(x, y, pch = 21, fill = fill, ...)
+     }, legend = list(right = list(fun = draw.colorkey, args = list(key = list(col = gray. ↩
    colors,
+         at = depth.breaks), draw = FALSE))), xlab = "Longitude",
+     ylab = "Latitude")
> print(pl)
```

ggplot2

```
> pg <- ggplot(quakes.ordered, aes(long, lat, colour = depth)) +
+     geom_point(size = 4) + facet_grid(~magnitude, labeller = label_both) +
+     coord_equal() + scale_colour_gradient(low = "grey30", high = "grey90") +
+     labs(x = "Longitude", y = "Latitude") + theme_bw()
> print(pg)
```



5.7 Figure 5.7

lattice

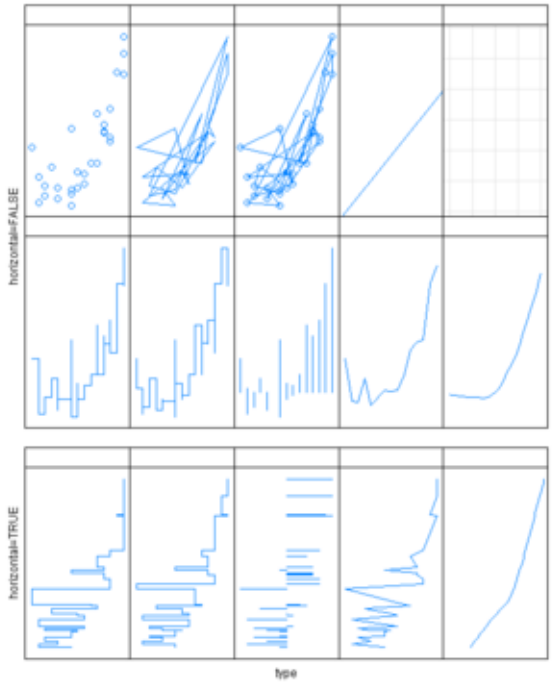
```
> types.plain <- c("p", "l", "o", "r", "g", "s", "S", "h", "a",
+     "smooth")
> types.horiz <- c("s", "S", "h", "a", "smooth")
> horiz <- rep(c(FALSE, TRUE), c(length(types.plain), length(types.horiz)))
> types <- c(types.plain, types.horiz)
> set.seed(2007041)
> x <- sample(seq(-10, 10, length = 15), 30, TRUE)
> y <- x + 0.25 * (x + 1)^2 + rnorm(length(x), sd = 5)
```

```
> pl <- xyplot(y ~ x | gl(1, length(types)), xlab = "type", ylab = list(c("horizontal=TRUE ↩
    ",
+     "horizontal=FALSE"), y = c(1/6, 4/6)), as.table = TRUE, layout = c(5,
+     3), between = list(y = c(0, 1)), strip = function(...) {
+     panel.fill(trellis.par.get("strip.background")$col[1])
```

```
+     type <- types[panel.number()]
+     grid.text(lab = sprintf("\"%s\"", type), x = 0.5, y = 0.5)
+     grid.rect()
+ }, scales = list(alternating = c(0, 2), tck = c(0, 0.7), draw = FALSE),
+     par.settings = list(layout.widths = list(strip.left = c(1,
+         0, 0, 0))), panel = function(...) {
+         type <- types[panel.number()]
+         horizontal <- horiz[panel.number()]
+         panel.xyplot(..., type = type, horizontal = horizontal)
+     })[rep(1, length(types))]
> print(pl)
```

ggplot2

```
No direct support - one would need to draw 15 separate graphs and combine these into one  ↩
    using grid.page()
```



5.8 Figure 5.8

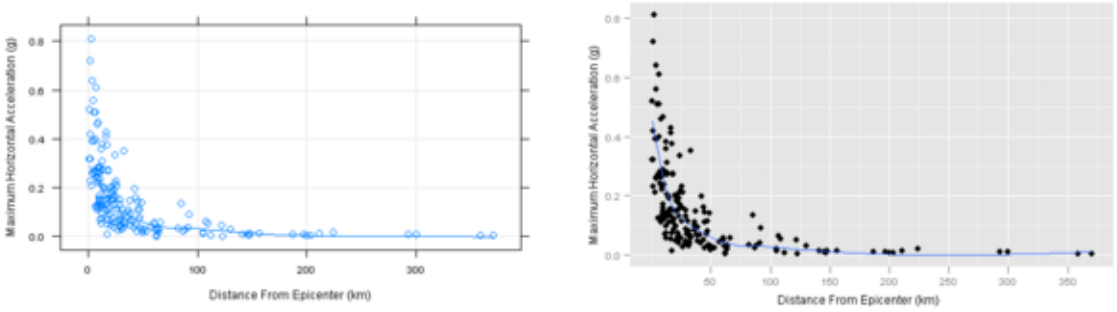
```
> data(Earthquake, package = "MEMSS")
```

lattice

```
> pl <- xyplot(accel ~ distance, data = Earthquake, panel = function(...) {
+     panel.grid(h = -1, v = -1)
+     panel.xyplot(...)
+     panel.loess(...)
+ }, xlab = "Distance From Epicenter (km)", ylab = "Maximum Horizontal Acceleration (g)")
> print(pl)
```

ggplot2

```
> pg <- ggplot(Earthquake, aes(distance, accel)) + geom_point() +
+     geom_smooth(method = "loess", se = FALSE) + xlab("Distance From Epicenter (km)") +
+     ylab("Maximum Horizontal Acceleration (g)")
> print(pg)
```



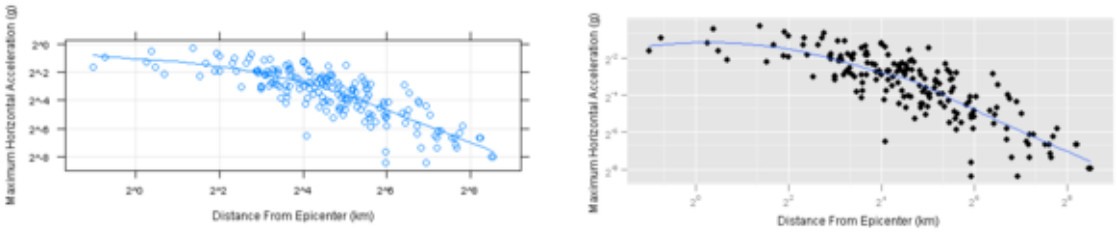
5.9 Figure 5.9

lattice

```
> pl <- xyplot(accel ~ distance, data = Earthquake, type = c("g",
+     "p", "smooth"), scales = list(log = 2), xlab = "Distance From Epicenter (km)",
+     ylab = "Maximum Horizontal Acceleration (g)")
> print(pl)
```

ggplot2

```
> pg <- pg + scale_x_log2() + scale_y_log2()
> print(pg)
```



5.10 Figure 5.10

```
> library(locfit)
```

lattice

```
> Earthquake$Magnitude <- equal.count(Earthquake$Richter, 3, overlap = 0.1)
> coef <- coef(lm(log2(accel) ~ log2(distance), data = Earthquake))
```

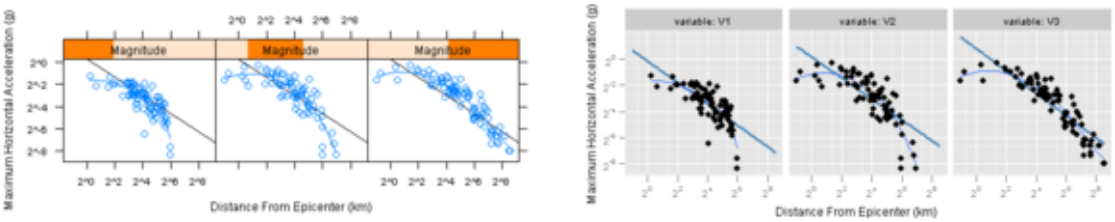
```
> pl <- xyplot(accel ~ distance | Magnitude, data = Earthquake,
+     scales = list(log = 2), col.line = "grey", lwd = 2, panel = function(...) {
+         panel.abline(reg = coef)
+         panel.locfit(...)
+     }, xlab = "Distance From Epicenter (km)", ylab = "Maximum Horizontal Acceleration (g) ↩
    ")
> print(pl)
```

ggplot2

```
> Earthquake2 <- merge(Earthquake, fn(Earthquake$Richter, 3, overlap = 0.1),
+     by.x = "Richter", by.y = "mag", all.x = TRUE)
```

```
> pg <- ggplot(Earthquake2, aes(distance, accel)) + facet_grid(~variable,
+     labeller = label_both) + geom_smooth(method = "lm", se = F,
+     fullrange = T, colour = "steelblue", size = 1) + geom_smooth(method = "locfit",
+     formula = y ~ x, se = F) + geom_point() + scale_x_log2() +
+     scale_y_log2() + xlab("Distance From Epicenter (km)") + ylab("Maximum Horizontal ↩
    Acceleration (g)")
> print(pg)
```

Note
Custom wrapper function fn() used to break the data into intervals. See Figure 5.5.



5.11 Figure 5.11

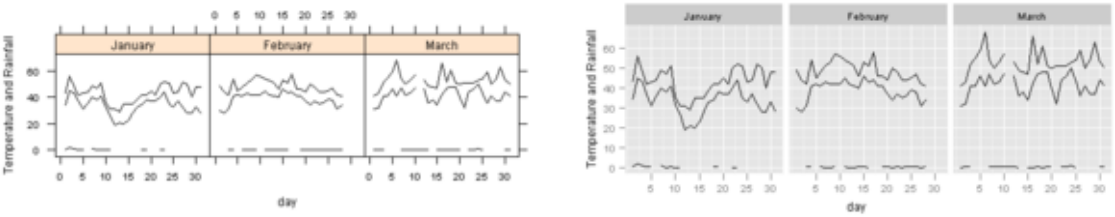
```
> data(SeatacWeather, package = "latticeExtra")
```

lattice

```
> pl <- xyplot(min.temp + max.temp + precip ~ day | month, ylab = "Temperature and Rainfall ↩
    ",
+     data = SeatacWeather, type = "l", lty = 1, col = "black")
> print(pl)
```

ggplot2

```
> p.precip <- ggplot(SeatacWeather, aes(day)) + facet_grid(~month) +
+     geom_line(aes(y = min.temp)) + geom_line(aes(y = max.temp)) +
+     ylab("Temperature and Rainfall")
> pg <- p.precip + geom_line(aes(y = precip))
> print(pg)
```



5.12 Figure 5.12

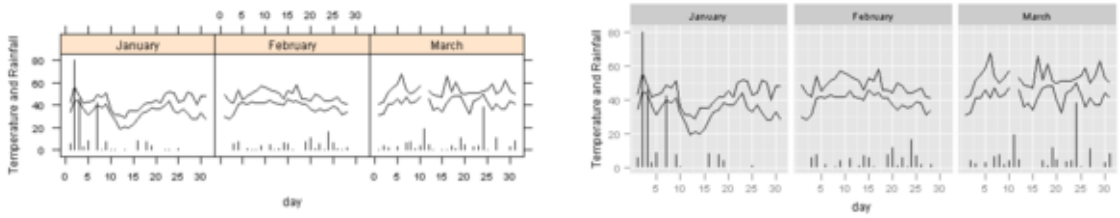
```
> maxp <- max(SeatacWeather$precip, na.rm = TRUE)
```

lattice

```
> pl <- xyplot(min.temp + max.temp + I(80 * precip/maxp) ~ day |
+     month, data = SeatacWeather, lty = 1, col = "black", ylab = "Temperature and Rainfall ↩
    ",
+     type = c("l", "l", "h"), distribute.type = TRUE)
> print(pl)
```

ggplot2

```
> pg <- p.precip + geom_linerange(aes(ymin = 0, ymax = 80 * precip/maxp))
> print(pg)
```



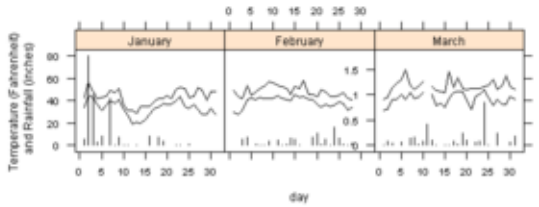
5.13 Figure 5.13

lattice

```
> pl <- update(trellis.last.object(), ylab = "Temperature (Fahrenheit) \n and Rainfall ( ↩
    inches)",
+     panel = function(...) {
+         panel.xyplot(...)
+         if (panel.number() == 2) {
+             at <- pretty(c(0, maxp))
+             panel.axis("right", half = FALSE, at = at * 80/maxp,
+                 labels = at)
+         }
+     })
> print(pl)
```

ggplot2

```
ggplot2 does not support the addition of a secondary axis.
```



5.14 Figure 5.14

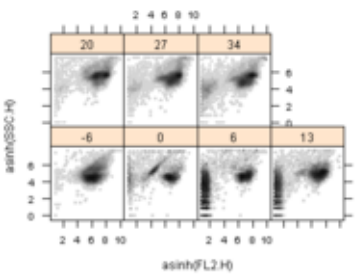
```
> library(hexbin)
> data(gvhd10, package = "latticeExtra")
```

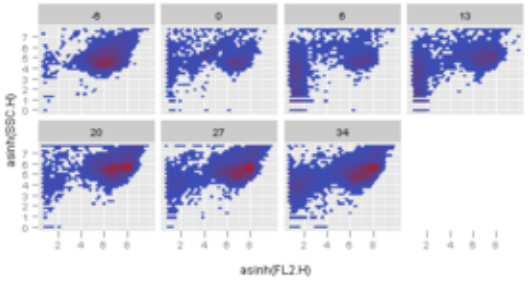
lattice

```
> pl <- xyplot(asinh(SSC.H) ~ asinh(FL2.H) | Days, gvhd10, aspect = 1,
+     panel = panel.hexbinplot, .aspect.ratio = 1, trans = sqrt)
> print(pl)
```

ggplot2

```
> pg <- ggplot(gvhd10, aes(asinh(FL2.H), asinh(SSC.H), fill = sqrt(..count..))) +
+     geom_hex() + facet_wrap(~Days, nrow = 2) + opts(legend.position = "none")
> print(pg)
```





5.15 Figure 5.15 - Scatter Plot Matrix

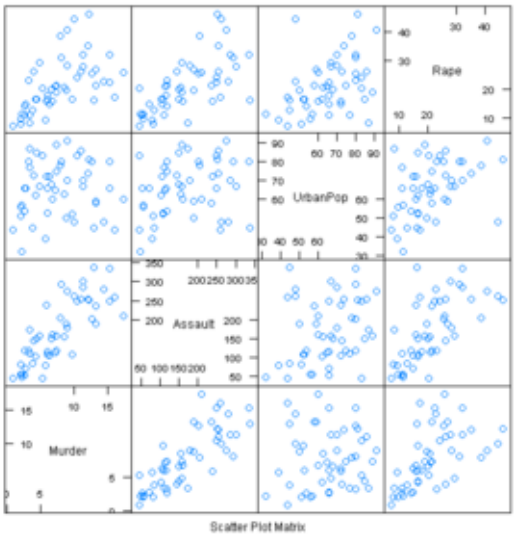
lattice

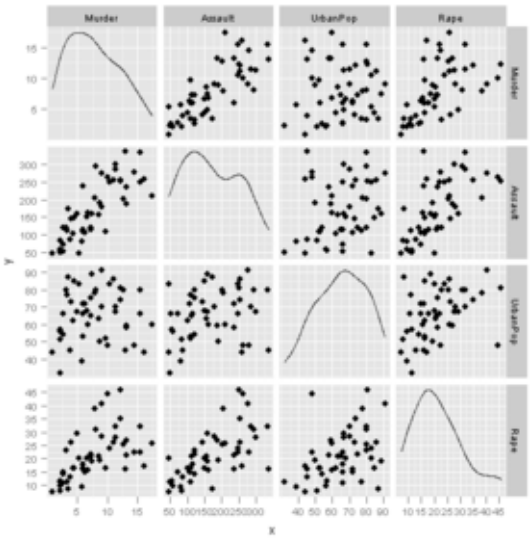
```
> pl <- splom(USArrests)
> print(pl)
```

ggplot2

```
> pg <- plotmatrix(USArrests)
> print(pg)
```

Note
plotmatrix function is still at experimental stage.





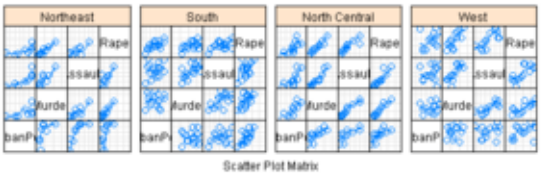
5.16 Figure 5.16

lattice

```
> pl <- splom(~USArrests[c(3, 1, 2, 4)] | state.region, pscales = 0,
+     type = c("g", "p", "smooth"))
> print(pl)
```

ggplot2

```
There is currently no easy way of achieving the same in ggplot2
```



5.17 Figure 5.17

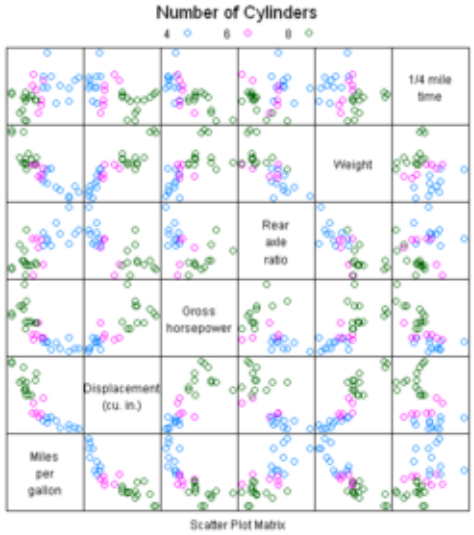
lattice

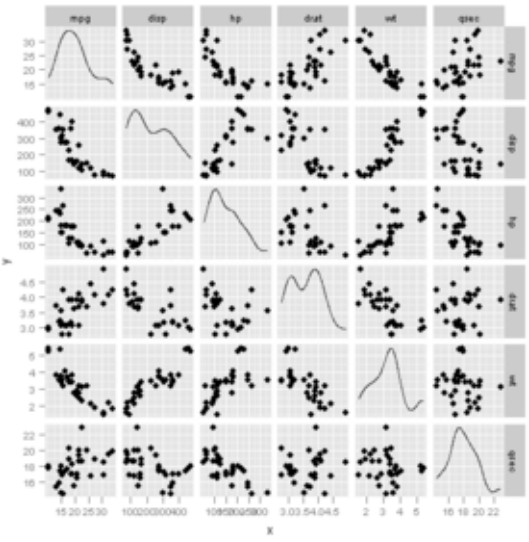
```
> pl <- splom(~data.frame(mpg, disp, hp, drat, wt, qsec), data = mtcars,
+     groups = cyl, pscales = 0, varnames = c("Miles\nper\ngallon",
+         "Displacement\n(cu. in.)", "Gross\nhorsepower", "Rear\naxle\nratio",
+         "Weight", "1/4 mile\ntime"), auto.key = list(columns = 3,
+         title = "Number of Cylinders"))
> print(pl)
```

ggplot2

```
> pg <- plotmatrix(with(mtcars, data.frame(mpg, disp, hp, drat,
+     wt, qsec)))
> print(pg)
```

Note
plotmatrix function is still at experimental stage. Colour mapping is a planned future feature.





5.18 Figure 5.18

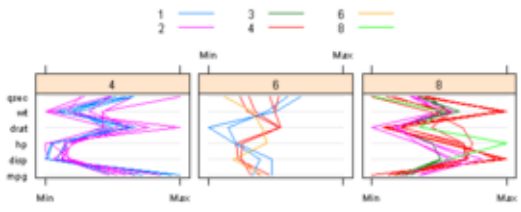
lattice

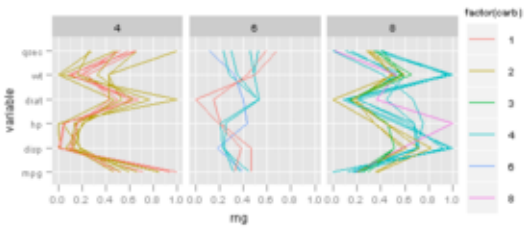
```
> pl <- parallel(~mtcars[c(1, 3, 4, 5, 6, 7)] | factor(cyl), mtcars,
+     groups = carb, key = simpleKey(levels(factor(mtcars$carb)),
+         points = FALSE, lines = TRUE, space = "top", columns = 3),
+     layout = c(3, 1))
> print(pl)
```

ggplot2

```
> mtcars <- namerows(mtcars, col.name = "car")
> df <- melt(mtcars[c(-8:-10)], id.var = c("cyl", "carb", "car"))
> dfm <- ddply(df, .(variable), transform, rng = rescaler(value,
+     type = "range"))
```

```
> pg <- ggplot(dfm, aes(group = car, colour = factor(carb))) +
+     geom_line(aes(variable, rng)) + facet_grid(~cyl) + coord_flip()
> print(pg)
```





5.19 Figure 5.19

lattice

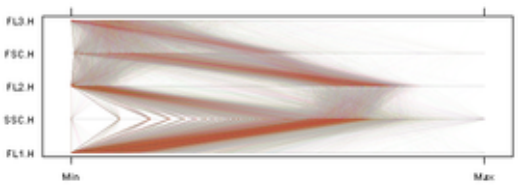
```
> pl <- parallel(~asinh(gvhd10[c(3, 2, 4, 1, 5)]), data = gvhd10,
+     subset = Days == "13", alpha = 0.01, lty = 1)
> print(pl)
```

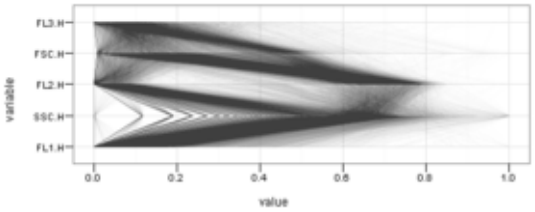
ggplot2

```
> df <- gvhd10[gvhd10$Days == "13", c(1:5)]
> df$id <- seq_along(df[, 1])
> df <- melt(df, id.vars = c("id"))
> df$variable <- factor(df$variable, levels = names(gvhd10)[c(3,
+     2, 4, 1, 5)])
> df <- ddply(df, .(variable), transform, value = rescaler(asinh(value),
+     type = "range"))
```

```
> pg <- ggplot(df, aes(value, variable, group = id)) + geom_path(alpha = 0.01) +
+     theme_bw()
> print(pg)
```

Note
Built-in black and white theme is used, otherwise the thin grey lines would be invisible on a grey background.





Chapter 6

Trivariate Displays

TOPICS COVERED:

- Three dimensional scatter plots
- Surfaces and two-way tables
- Level plots and contour plots
- Wireframe rendering
- Parameterized surfaces

6.1 Figure 6.1

```
> library(lattice)
> library(ggplot2)
```

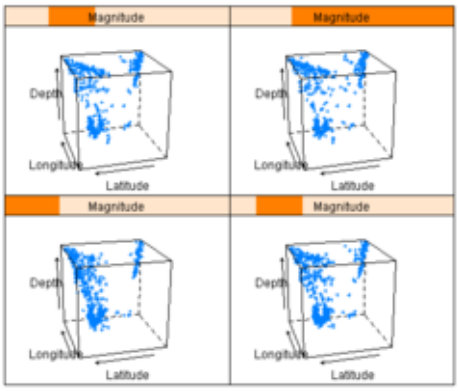
```
> quakes$Magnitude <- equal.count(quakes$mag, 4)
```

lattice

```
> pl <- cloud(depth ~ lat * long | Magnitude, data = quakes, zlim = rev(range(quakes$depth) ↩
    ),
+     screen = list(z = 105, x = -70), panel.aspect = 0.75, xlab = "Longitude",
+     ylab = "Latitude", zlab = "Depth")
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



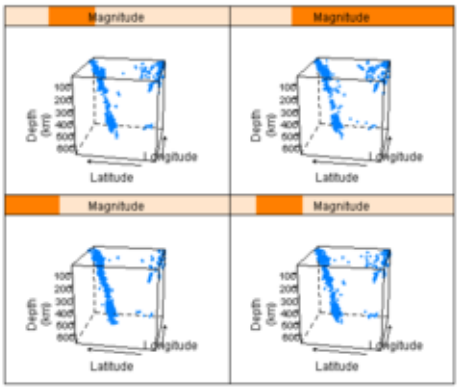
6.2 Figure 6.2

lattice

```
> pl <- cloud(depth ~ lat * long | Magnitude, data = quakes, zlim = rev(range(quakes$depth) ↩
    ),
+     panel.aspect = 0.75, screen = list(z = 80, x = -70), zoom = 0.7,
+     scales = list(z = list(arrows = FALSE, distance = 2)), xlab = "Longitude",
+     ylab = "Latitude", zlab = list("Depth\n(km)", rot = 90))
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



6.3 Figure 6.3

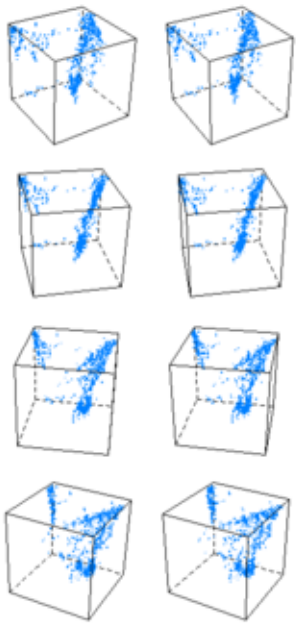
lattice

```
> p <- cloud(depth ~ long + lat, quakes, zlim = c(690, 30), pch = ".",
+     cex = 1.5, zoom = 1, xlab = NULL, ylab = NULL, zlab = NULL,
+     par.settings = list(axis.line = list(col = "transparent")),
+     scales = list(draw = FALSE))
> npanel <- 4
> rotz <- seq(-30, 30, length = npanel)
> roty <- c(3, 0)
```

```
> pl <- update(p[rep(1, 2 * npanel)], layout = c(2, npanel), panel = function(...,
+     screen) {
+     crow <- current.row()
+     ccol <- current.column()
+     panel.cloud(..., screen = list(z = rotz[crow], x = -60, y = roty[ccol]))
+ })
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



6.4 Figure 6.4

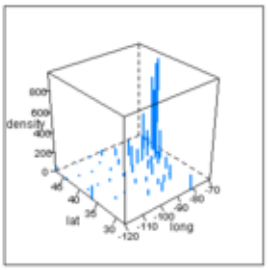
```
> state.info <- data.frame(name = state.name, long = state.center$x,
+     lat = state.center$y, area = state.x77[, "Area"], population = 1000 *
+         state.x77[, "Population"])
> state.info$density <- with(state.info, population/area)
```

lattice

```
> pl <- cloud(density ~ long + lat, state.info, subset = !(name %in%
+     c("Alaska", "Hawaii")), type = "h", lwd = 2, zlim = c(0,
+     max(state.info$density)), scales = list(arrows = FALSE))
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



6.5 Figure 6.5

```
> library("maps")
> state.map <- map("state", plot = FALSE, fill = FALSE)
```

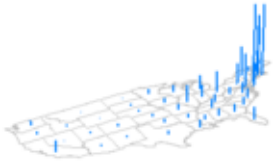
lattice

```
> panel.3dmap <- function(..., rot.mat, distance, xlim, ylim, zlim,
+     xlim.scaled, ylim.scaled, zlim.scaled) {
+     scaled.val <- function(x, original, scaled) {
+         scaled[1] + (x - original[1]) * diff(scaled)/diff(original)
+     }
+     m <- ltransform3dto3d(rbind(scaled.val(state.map$x, xlim,
+         xlim.scaled), scaled.val(state.map$y, ylim, ylim.scaled),
+         zlim.scaled[1]), rot.mat, distance)
+     panel.lines(m[1, ], m[2, ], col = "grey76")
+ }
```

```
> pl <- cloud(density ~ long + lat, state.info, subset = !(name %in%
+     c("Alaska", "Hawaii")), panel.3d.cloud = function(...) {
+     panel.3dmap(...)
+     panel.3dscatter(...)
+ }, type = "h", scales = list(draw = FALSE), zoom = 1.1, xlim = state.map$range[1:2],
+     ylim = state.map$range[3:4], xlab = NULL, ylab = NULL, zlab = NULL,
+     aspect = c(diff(state.map$range[3:4])/diff(state.map$range[1:2]),
+         0.3), panel.aspect = 0.75, lwd = 2, screen = list(z = 30,
+         x = -60), par.settings = list(axis.line = list(col = "transparent"),
+         box.3d = list(col = "transparent", alpha = 0)))
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



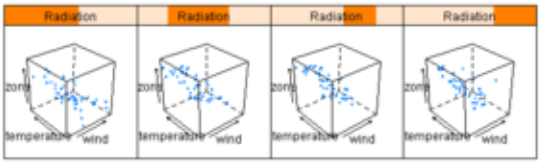
6.6 Figure 6.6

lattice

```
> env <- environmental
> env$ozone <- env$ozone^(1/3)
> env$Radiation <- equal.count(env$radiation, 4)
> pl <- cloud(ozone ~ wind + temperature | Radiation, env)
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



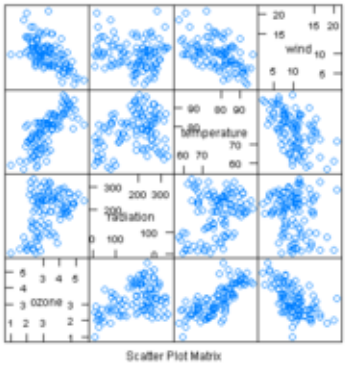
6.7 Figure 6.7

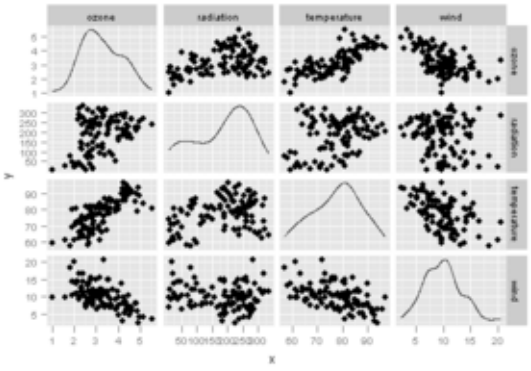
lattice

```
> pl <- splom(env[1:4])
> print(pl)
```

ggplot2

```
> pg <- plotmatrix(env[1:4])
> print(pg)
```





6.8 Figure 6.8

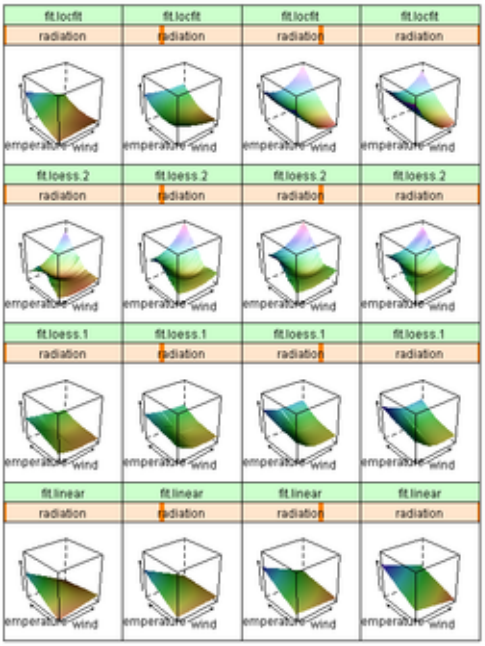
```
> fm1.env <- lm(ozone ~ radiation * temperature * wind, env)
> fm2.env <- loess(ozone ~ wind * temperature * radiation, env,
+     span = 0.75, degree = 1)
> fm3.env <- loess(ozone ~ wind * temperature * radiation, env,
+     parametric = c("radiation", "wind"), span = 0.75, degree = 2)
> library("locfit")
locfit 1.5-4     2007-11-27
> fm4.env <- locfit(ozone ~ wind * temperature * radiation, env)
> w.mesh <- with(env, do.breaks(range(wind), 50))
> t.mesh <- with(env, do.breaks(range(temperature), 50))
> r.mesh <- with(env, do.breaks(range(radiation), 3))
> grid <- expand.grid(wind = w.mesh, temperature = t.mesh, radiation = r.mesh)
> grid[["fit.linear"]] <- predict(fm1.env, newdata = grid)
> grid[["fit.loess.1"]] <- as.vector(predict(fm2.env, newdata = grid))
> grid[["fit.loess.2"]] <- as.vector(predict(fm3.env, newdata = grid))
> grid[["fit.locfit"]] <- predict(fm4.env, newdata = grid)
```

lattice

```
> pl <- wireframe(fit.linear + fit.loess.1 + fit.loess.2 + fit.locfit ~
+     wind * temperature | radiation, grid, outer = TRUE, shade = TRUE,
+     zlab = "")
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



6.9 Figure 6.9

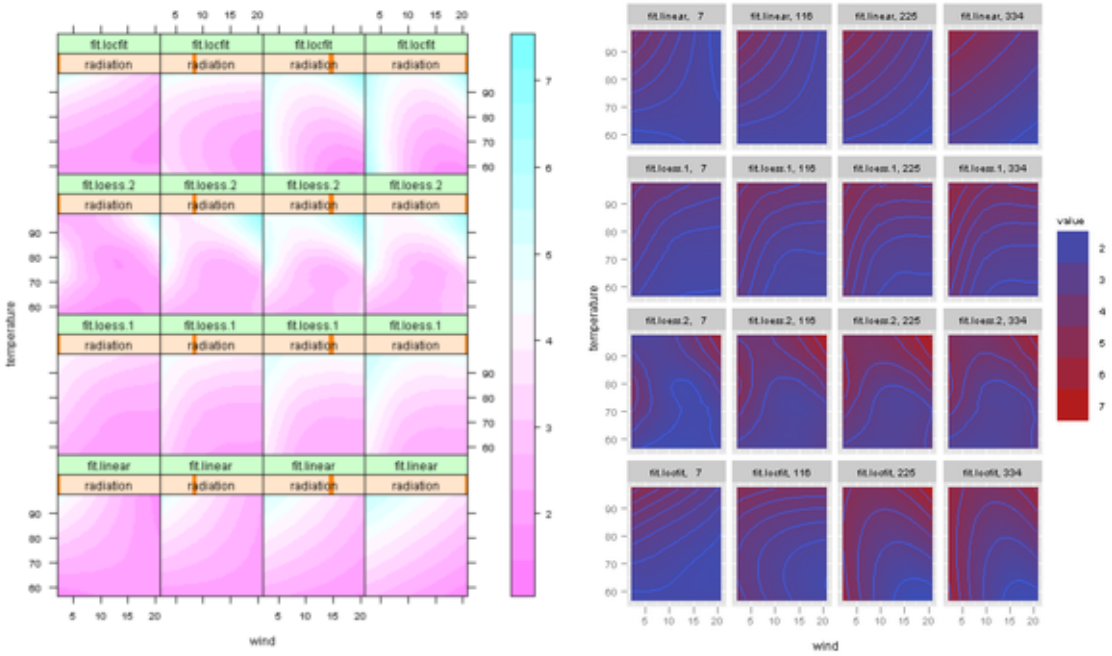
lattice

```
> pl <- levelplot(fit.linear + fit.loess.1 + fit.loess.2 + fit.locfit ~
+     wind * temperature | radiation, data = grid)
> print(pl)
```

ggplot2

```
> grid.m <- melt(grid, id.vars = 1:3)
```

```
> pg <- ggplot(grid.m, aes(wind, temperature, z = value, fill = value)) +
+     facet_wrap(~variable + radiation) + geom_tile() + geom_contour()
> print(pg)
```



6.10 Figure 6.10

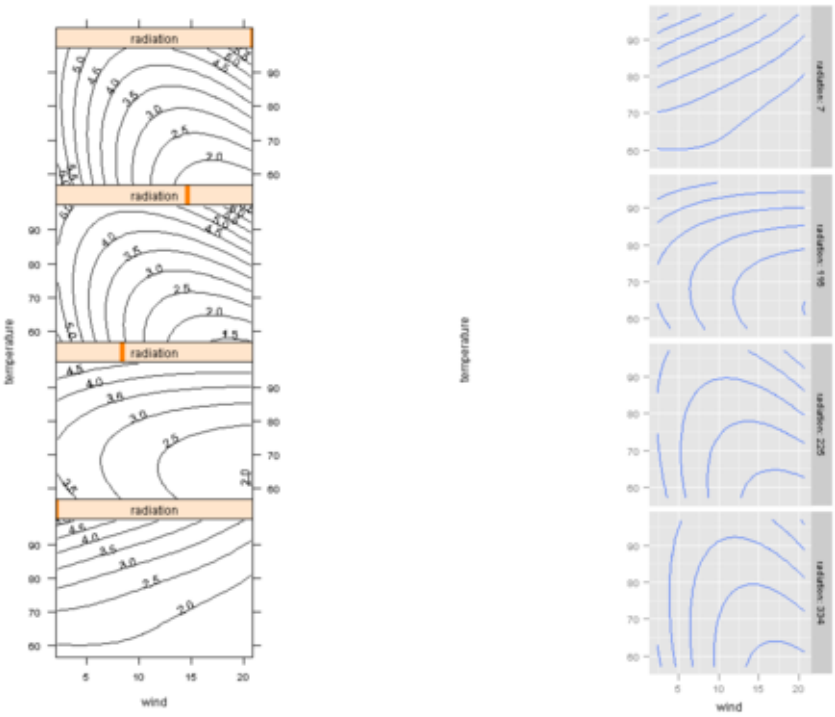
lattice

```
> pl <- contourplot(fit.locfit ~ wind * temperature | radiation,
+     data = grid, aspect = 0.7, layout = c(1, 4), cuts = 15, label.style = "align")
> print(pl)
```

ggplot2

```
> pg <- ggplot(grid[, c(1:3, 7)], aes(wind, temperature, z = fit.locfit)) +
+     geom_contour() + facet_grid(radiation ~ ., labeller = label_both) +
+     opts(aspect.ratio = 1)
> print(pg)
```

Note
Contour labeling not easily accomplished.



6.11 Figure 6.11

lattice

```
> plot(levelplot(volcano), split = c(1, 1, 1, 3), more = TRUE)
> plot(contourplot(volcano, cuts = 20, label = FALSE), split = c(1,
+     2, 1, 3), more = TRUE)
> plot(wireframe(volcano, panel.aspect = 0.7, zoom = 1, lwd = 0.01),
+     split = c(1, 3, 1, 3), more = FALSE)
```

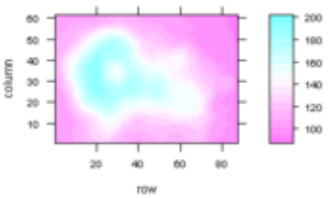
ggplot2

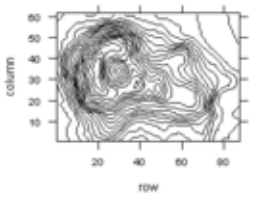
```
> library(ggextra)
```

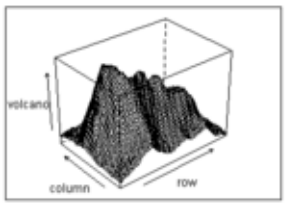
Note
To install this package directly within R type: `install.packages("ggextra", repos="http://R-Forge.R-project.org")`

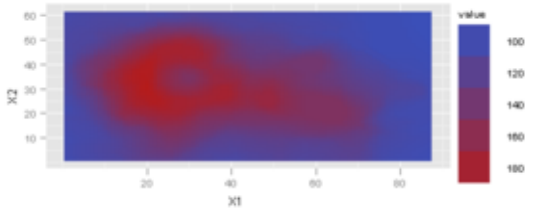
```
> p <- ggplot(melt(volcano), aes(x = X1, y = X2, z = value, fill = value))
> p1 <- p + geom_tile()
> p2 <- p + geom_contour(bins = 20)
> print(arrange(p1, p2, ncol = 1))
[1] 3
```

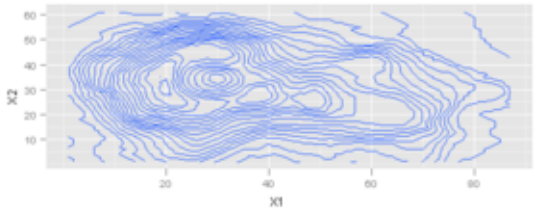
Note
ggplot2 currently does not support true 3d surfaces.











6.12 Figure 6.12

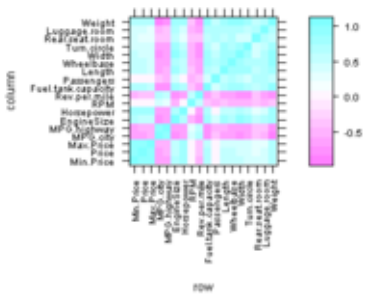
```
> data(Cars93, package = "MASS")
> cor.Cars93 <- cor(Cars93[, !sapply(Cars93, is.factor)], use = "pair")
```

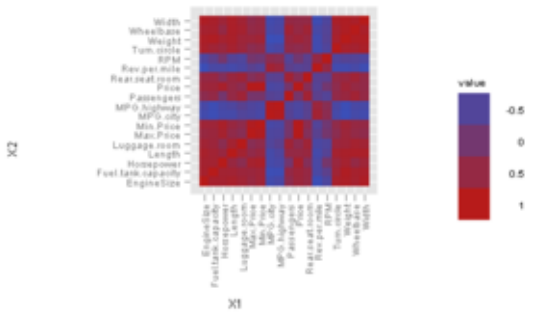
lattice

```
> pl <- levelplot(cor.Cars93, scales = list(x = list(rot = 90)))
> print(pl)
```

ggplot2

```
> pg <- ggplot(melt(cor.Cars93), aes(X1, X2, fill = value)) + geom_tile() +
+     opts(axis.text.x = theme_text(lineheight = 0.9, colour = "grey50",
+         hjust = 1, angle = 90)) + opts(aspect.ratio = 1)
> print(pg)
```





6.13 Figure 6.13

```
> ord <- order.dendrogram(as.dendrogram(hclust(dist(cor.Cars93))))
```

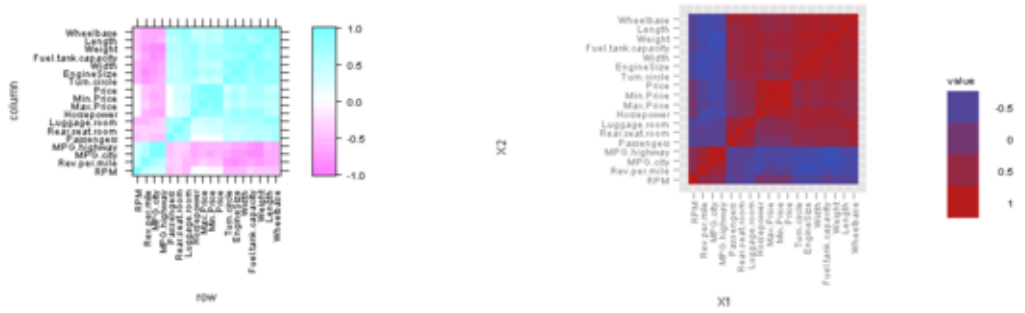
lattice

```
> pl <- levelplot(cor.Cars93[ord, ord], at = do.breaks(c(-1.01,
+     1.01), 20), scales = list(x = list(rot = 90)))
> print(pl)
```

ggplot2

```
> lvls <- rownames(cor.Cars93)[ord]
> cor.Cars93.m <- melt(cor.Cars93)
> cor.Cars93.m$X1 <- factor(cor.Cars93.m$X1, levels = lvls)
> cor.Cars93.m$X2 <- factor(cor.Cars93.m$X2, levels = lvls)
```

```
> pg <- pg %+% cor.Cars93.m
> print(pg)
```



6.14 Figure 6.14

```
> data(Chem97, package = "mlmRev")
> Chem97$gcd <- with(Chem97, cut(gcsescore, breaks = quantile(gcsescore,
+     ppoints(11, a = 1))))
> ChemTab <- xtabs(~score + gcd + gender, Chem97)
> ChemTabDf <- as.data.frame.table(ChemTab)
```

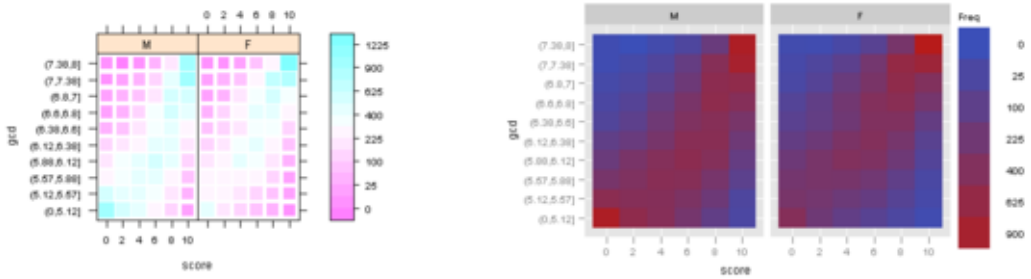
lattice

```
> tick.at <- pretty(range(sqrt(ChemTabDf$Freq)))
```

```
> pl <- levelplot(sqrt(Freq) ~ score * gcd | gender, ChemTabDf,
+     shrink = c(0.7, 1), colorkey = list(labels = list(at = tick.at,
+         labels = tick.at^2)), aspect = "iso")
> print(pl)
```

ggplot2

```
> pg <- ggplot(ChemTabDf, aes(score, gcd, fill = Freq)) + facet_grid(~gender) +
+     geom_tile() + scale_fill_gradient(trans = "sqrt")
> print(pg)
```



6.15 Figure 6.15

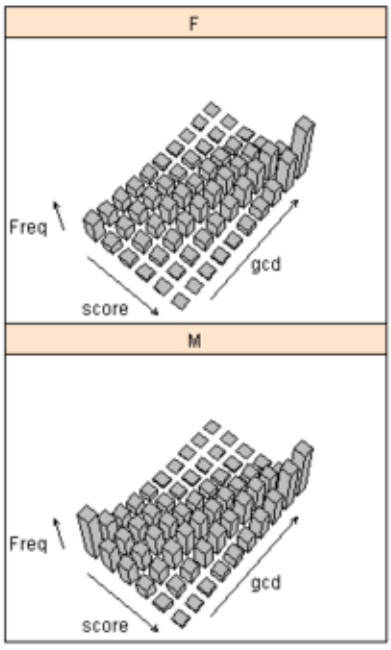
```
> library("latticeExtra")
```

lattice

```
> pl <- cloud(Freq ~ score * gcd | gender, data = ChemTabDf, screen = list(z = -40,
+     x = -25), zoom = 1.1, col.facet = "grey", xbase = 0.6, ybase = 0.6,
+     par.settings = list(box.3d = list(col = "transparent")),
+     aspect = c(1.5, 0.75), panel.aspect = 0.75, panel.3d.cloud = panel.3dbars)
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



6.16 Figure 6.16

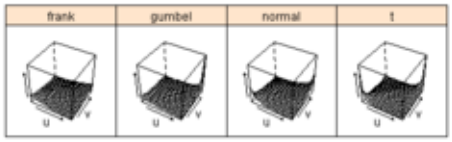
```
> library("copula")
> grid <- expand.grid(u = do.breaks(c(0.01, 0.99), 25), v = do.breaks(c(0.01,
+     0.99), 25))
> grid$frank <- with(grid, dcopula(frankCopula(2), cbind(u, v)))
> grid$gumbel <- with(grid, dcopula(gumbelCopula(1.2), cbind(u,
+     v)))
> grid$normal <- with(grid, dcopula(normalCopula(0.4), cbind(u,
+     v)))
> grid$t <- with(grid, dcopula(tCopula(0.4), cbind(u, v)))
```

lattice

```
> pl <- wireframe(frank + gumbel + normal + t ~ u * v, grid, outer = TRUE,
+     zlab = "", screen = list(z = -30, x = -50), lwd = 0.01)
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



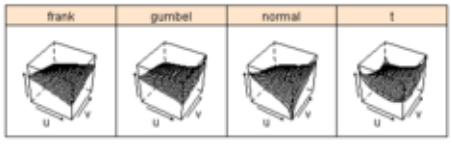
6.17 Figure 6.17

lattice

```
> pl <- wireframe(frank + gumbel + normal + t ~ u * v, grid, outer = TRUE,
+     zlab = "", screen = list(z = -30, x = -50), scales = list(z = list(log = TRUE)),
+     lwd = 0.01)
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



6.18 Figure 6.18

```
> kx <- function(u, v) cos(u) * (r + cos(u/2) * sin(t * v) - sin(u/2) *
+     sin(2 * t * v))
> ky <- function(u, v) sin(u) * (r + cos(u/2) * sin(t * v) - sin(u/2) *
+     sin(2 * t * v))
> kz <- function(u, v) sin(u/2) * sin(t * v) + cos(u/2) * sin(t *
+     v)
```

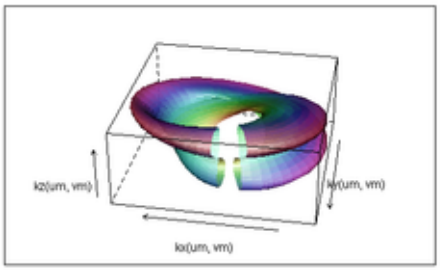
```
> n <- 50
> u <- seq(0.3, 1.25, length = n) * 2 * pi
> v <- seq(0, 1, length = n) * 2 * pi
> um <- matrix(u, length(u), length(u))
> vm <- matrix(v, length(v), length(v), byrow = TRUE)
> r <- 2
> t <- 1
```

lattice

```
> pl <- wireframe(kz(um, vm) ~ kx(um, vm) + ky(um, vm), shade = TRUE,
+     screen = list(z = 170, x = -60), alpha = 0.75, panel.aspect = 0.6,
+     aspect = c(1, 0.4))
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



6.19 Figure 6.19

```
> data(USAge.df, package = "latticeExtra")
> library("RColorBrewer")
```

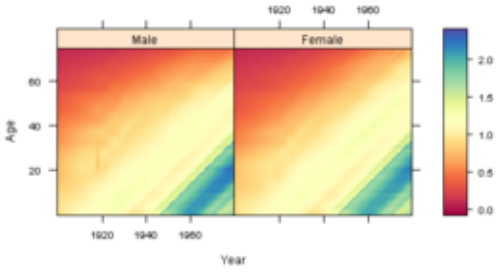
lattice

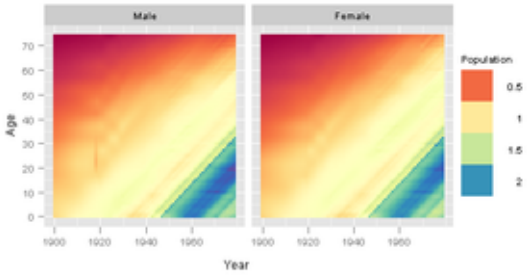
```
> brewer.div <- colorRampPalette(brewer.pal(11, "Spectral"), interpolate = "spline")
```

```
> pl <- levelplot(Population ~ Year * Age | Sex, data = USAge.df,
+     cuts = 199, col.regions = brewer.div(200), aspect = "iso")
> print(pl)
```

ggplot2

```
> pg <- ggplot(USAge.df, aes(Year, Age, fill = Population)) + facet_grid(~Sex) +
+     geom_tile() + scale_fill_gradientn("Population", colours = brewer.div(200)) +
+     opts(aspect.ratio = 1)
> print(pg)
```


1920
1940
1960
Male
Female
Age
60
40
20
1920
1940
1960
Year
2.0
1.5
1.0
0.5
0.0


Male
Female
Age
70
60
50
40
30
20
10
0
1900
1920
1940
1960
Year
Population
0.5
1
1.5
2

Chapter 7

Graphical Parameters and Other Settings

TOPICS COVERED:

- The graphical parameter system
- Themes, devices
- Initializing graphics devices
- Querying and modifying parameters
- Available parameters
- Non-graphical options
- Making customizations persistent

7.1 Figure 7.1

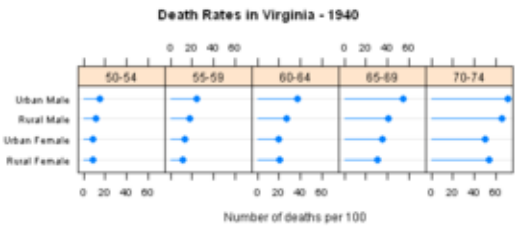
```
> library(lattice)
> library(ggplot2)
```

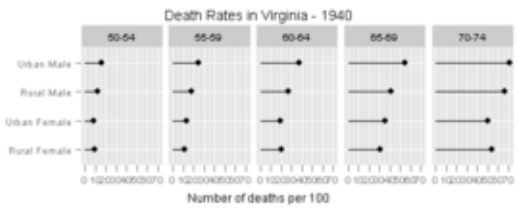
lattice

```
> pl <- dotplot(reorder(Var2, Freq) ~ Freq | Var1, data = as.data.frame.table(VADeaths),
+     origin = 0, type = c("p", "h"), main = "Death Rates in Virginia - 1940",
+     xlab = "Number of deaths per 100")
> print(pl)
```

ggplot2

```
> pg <- ggplot(as.data.frame.table(VADeaths), aes(reorder(Var2,
+     Freq), Freq)) + geom_point() + geom_linerange(aes(ymin = 0,
+     ymax = Freq)) + facet_grid(~Var1) + ylab("Number of deaths per 100") +
+     xlab("") + opts(title = "Death Rates in Virginia - 1940") +
+     coord_flip()
> print(pg)
```





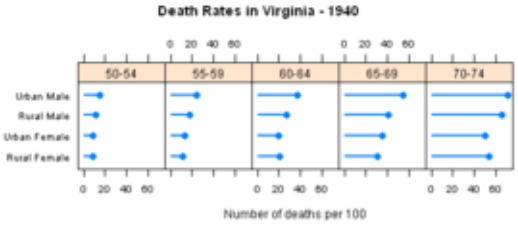
7.2 Figure 7.2

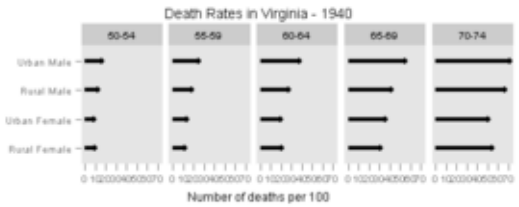
lattice

```
> dot.line.settings <- trellis.par.get("dot.line")
> dot.line.settings$col <- "transparent"
> trellis.par.set("dot.line", dot.line.settings)
> plot.line.settings <- trellis.par.get("plot.line")
> plot.line.settings$lwd <- 2
> trellis.par.set("plot.line", plot.line.settings)
> print(trellis.last.object())
```

ggplot2

```
> pg <- pg + geom_linerange(aes(ymin = 0, ymax = Freq), size = 1.5) +
+     opts(panel.grid.major = theme_blank(), panel.grid.minor = theme_blank())
> print(pg)
```





7.3 Figure 7.3

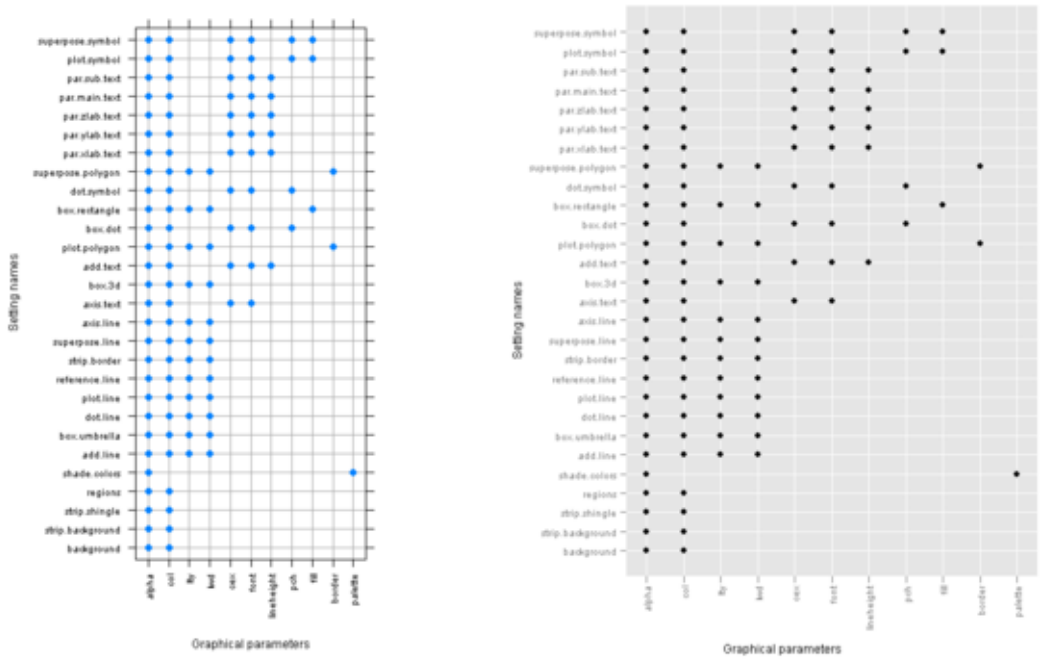
```
> tp <- trellis.par.get()
> unusual <- c("grid.pars", "fontsize", "clip", "axis.components",
+     "layout.heights", "layout.widths")
> for (u in unusual) tp[[u]] <- NULL
> names.tp <- lapply(tp, names)
> unames <- sort(unique(unlist(names.tp)))
> ans <- matrix(0, nrow = length(names.tp), ncol = length(unames))
> rownames(ans) <- names(names.tp)
> colnames(ans) <- unames
> for (i in seq(along = names.tp)) ans[i, ] <- as.numeric(unames %in%
+     names.tp[[i]])
> ans <- ans[, order(-colSums(ans))]
> ans <- ans[order(rowSums(ans)), ]
> ans[ans == 0] <- NA
```

lattice

```
> pl <- levelplot(t(ans), colorkey = FALSE, scales = list(x = list(rot = 90)),
+     panel = function(x, y, z, ...) {
+         panel.abline(v = unique(as.numeric(x)), h = unique(as.numeric(y)),
+             col = "darkgrey")
+         panel.xyplot(x, y, pch = 16 * z, ...)
+     }, xlab = "Graphical parameters", ylab = "Setting names")
> print(pl)
```

ggplot2

```
> pg <- ggplot(subset(as.data.frame.table(ans), Freq == 1), aes(Var2,
+     Var1)) + geom_point() + xlab("Graphical parameters") + ylab("Setting names") +
+     opts(axis.text.x = theme_text(angle = 90, hjust = 1, colour = "grey50"))
> print(pg)
```



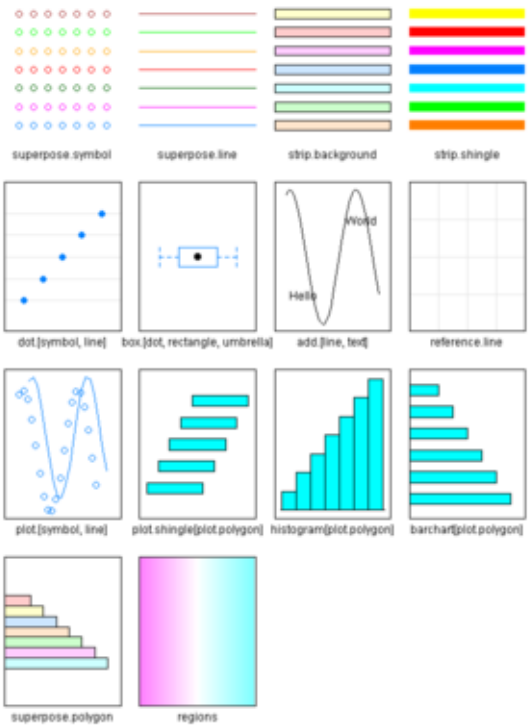
7.4 Figure 7.4

lattice

```
> show.settings()
```

ggplot2

```
The same not possible in ggplot2.
```



Chapter 8

Plot Coordinates and Axis Annotation

TOPICS COVERED:

- Packets
- The prepanel function, axis limits, and aspect ratio
- Axis annotation
- The scales argument

8.1 Figure 8.1

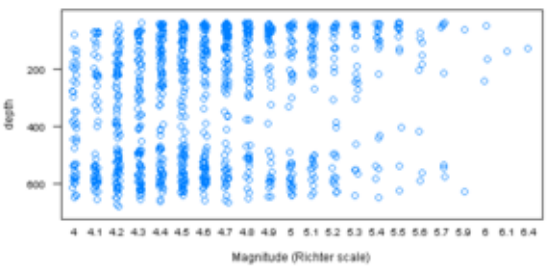
```
> library(lattice)
> library(ggplot2)
```

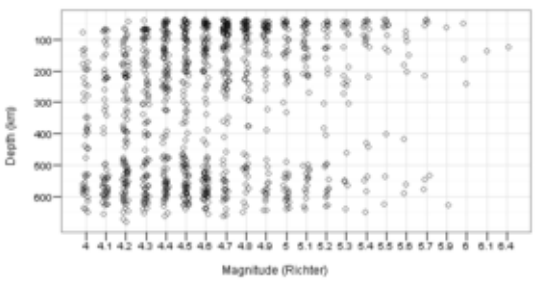
lattice

```
> pl <- stripplot(depth ~ factor(mag), data = quakes, jitter.data = TRUE,
+     scales = list(y = "free", rot = 0), prepanel = function(x,
+         y, ...) list(ylim = rev(range(y)))), xlab = "Magnitude (Richter scale)")
> print(pl)
```

ggplot2

```
> p <- ggplot(quakes, aes(factor(mag), depth))
> pg <- p + geom_point(position = position_jitter(width = 0.15),
+     alpha = 0.6, shape = 1) + xlab("Magnitude (Richter)") + ylab("Depth (km)") +
+     theme_bw() + scale_y_reverse()
> print(pg)
```





Note
Compare to Figure 3.16. Y-Axes reversed.

8.2 Figure 8.2

```
> data(biocAccess, package = "latticeExtra")
```

lattice

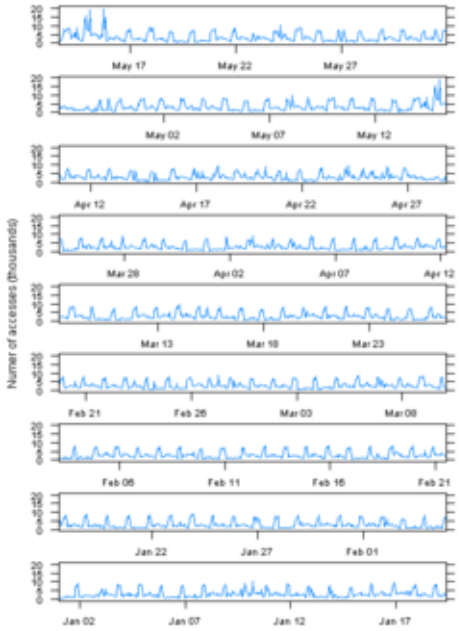
```
> pl <- xyplot(counts/1000 ~ time | equal.count(as.numeric(time),
+     9, overlap = 0.1), biocAccess, type = "l", aspect = "xy",
+     strip = FALSE, ylab = "Numer of accesses (thousands)", xlab = "",
+     scales = list(x = list(relation = "sliced", axs = "i"), y = list(alternating = FALSE) ↩
    ))
> print(pl)
```

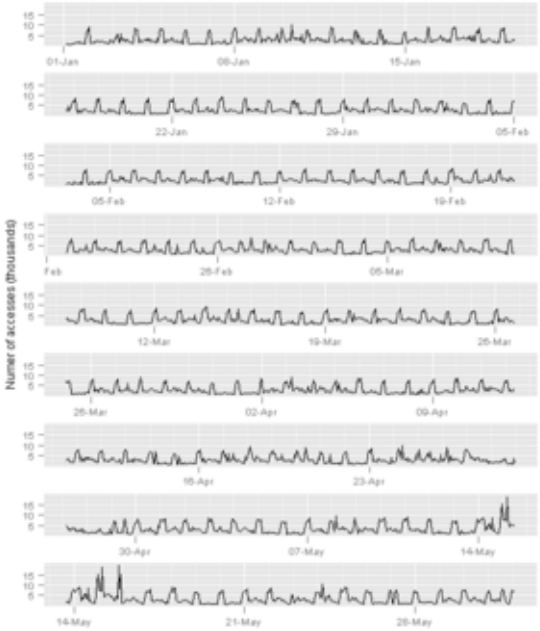
ggplot2

```
> fn <- function(data = quakes$mag, number = 4, ...) {
+     intrv <<- as.data.frame(co.intervals(data, number, ...))
+     mag <- sort(unique(data))
+     intervals <- ldply(mag, function(x) {
+         t(as.numeric(x < intrv$V2 & x > intrv$V1))
+     })
+     tmp <- melt(cbind(mag, intervals), id.var = 1)
+     tmp[tmp$value > 0, 1:2]
+ }
> biocAccess <- merge(biocAccess[, c("time", "counts")], fn(data = as.numeric( ↩
    biocAccess$time),
+     number = 9, overlap = 0.1), by.x = "time", by.y = "mag")
```

Note
Using custom function fn(), first defined in Figure 5.5.

```
> pg <- ggplot(biocAccess, aes(time, counts/1000)) + geom_line() +
+     facet_wrap(~variable, scales = "free_x", ncol = 1) + ylab("Numer of accesses ( ↩
    thousands)") +
+     xlab("") + opts(strip.background = theme_blank(), strip.text.x = theme_blank()) +
+     opts(panel.margin = unit(-0.25, "lines"))
> print(pg)
```





8.3 Figure 8.3

```
> data(Earthquake, package = "MEMSS")
```

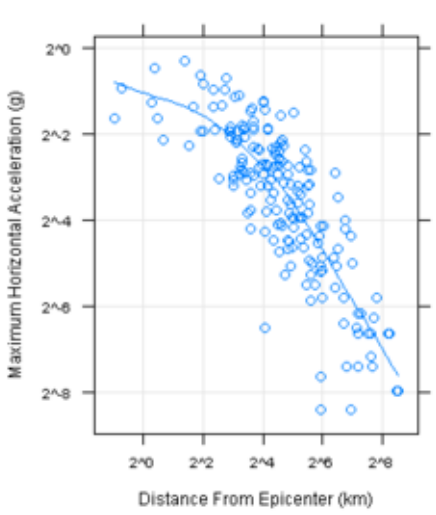
lattice

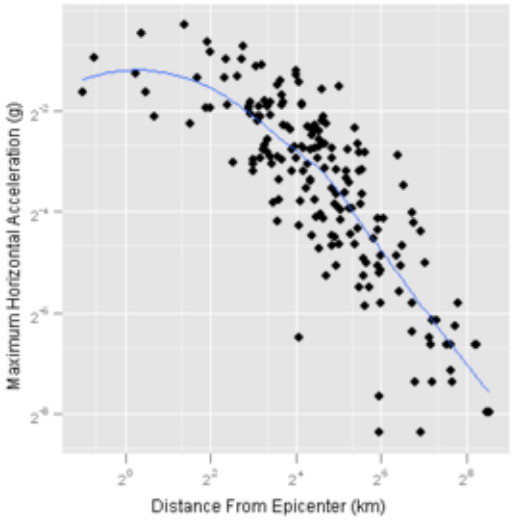
```
> pl <- xyplot(accel ~ distance, data = Earthquake, prepanel = prepanel.loess,
+     aspect = "xy", type = c("p", "g", "smooth"), scales = list(log = 2),
+     xlab = "Distance From Epicenter (km)", ylab = "Maximum Horizontal Acceleration (g)")
> print(pl)
```

ggplot2

```
> pg <- ggplot(Earthquake, aes(distance, accel)) + geom_point() +
+     geom_smooth(method = "loess", se = FALSE) + scale_x_log2() +
+     scale_y_log2() + xlab("Distance From Epicenter (km)") + ylab("Maximum Horizontal  ↩
    Acceleration (g)")
> print(pg)
```

Note
ggplot2 doesn't have the equivalent of `aspect="xy"` in `lattice`, which "*tries* to compute the aspect based on the 45 degree banking rule".





8.4 Figure 8.4

lattice

```
> yscale.components.log2 <- function(...) {
+     ans <- yscale.components.default(...)
+     ans$right <- ans$left
+     ans$left$labels$labels <- parse(text = ans$left$labels$labels)
+     ans$right$labels$labels <- MASS::fractions(2^(ans$right$labels$at))
+     ans
+ }
> logTicks <- function(lim, loc = c(1, 5)) {
+     ii <- floor(log10(range(lim))) + c(-1, 2)
+     main <- 10^(ii[1]:ii[2])
+     r <- as.numeric(outer(loc, main, "*"))
+     r[lim[1] <= r & r <= lim[2]]
+ }
> xscale.components.log2 <- function(lim, ...) {
+     ans <- xscale.components.default(lim = lim, ...)
+     tick.at <- logTicks(2^lim, loc = c(1, 3))
+     ans$bottom$ticks$at <- log(tick.at, 2)
+     ans$bottom$labels$at <- log(tick.at, 2)
+     ans$bottom$labels$labels <- as.character(tick.at)
+     ans
+ }
```

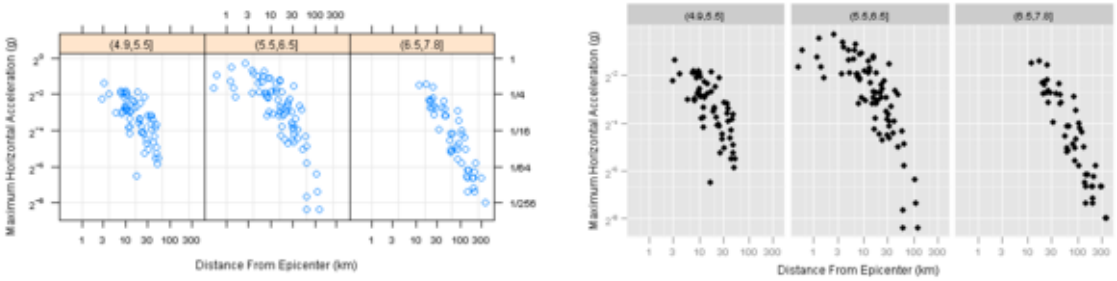
```
> pl <- xyplot(accel ~ distance | cut(Richter, c(4.9, 5.5, 6.5,
+     7.8)), data = Earthquake, type = c("p", "g"), scales = list(log = 2,
+     y = list(alternating = 3)), xlab = "Distance From Epicenter (km)",
+     ylab = "Maximum Horizontal Acceleration (g)", xscale.components = xscale.components. ↩
    log2,
+     yscale.components = yscale.components.log2)
> print(pl)
```

ggplot2

```
> Earthquake$magnitude <- cut(Earthquake$Richter, c(4.9, 5.5, 6.5,
+     7.8))
> ticks <- logTicks(range(Earthquake$distance), loc = c(1, 3))
```

```
> pg <- ggplot(Earthquake, aes(distance, accel)) + geom_point() +
+     facet_grid(~magnitude) + scale_y_log2() + scale_x_log2(breaks = ticks,
+     labels = ticks) + xlab("Distance From Epicenter (km)") +
+     ylab("Maximum Horizontal Acceleration (g)")
> print(pg)
```

Note
ggplot2 does not support the addition of a secondary axes.



8.5 Figure 8.5

```
> xscale.components.log10 <- function(lim, ...) {
+     ans <- xscale.components.default(lim = lim, ...)
+     tick.at <- logTicks(10^lim, loc = 1:9)
+     tick.at.major <- logTicks(10^lim, loc = 1)
+     major <- tick.at %in% tick.at.major
+     ans$bottom$ticks$at <- log(tick.at, 10)
+     ans$bottom$ticks$tck <- ifelse(major, 1.5, 0.75)
+     ans$bottom$labels$at <- log(tick.at, 10)
+     ans$bottom$labels$labels <- as.character(tick.at)
+     ans$bottom$labels$labels[!major] <- ""
+     ans$bottom$labels$check.overlap <- FALSE
+     ans
+ }
```

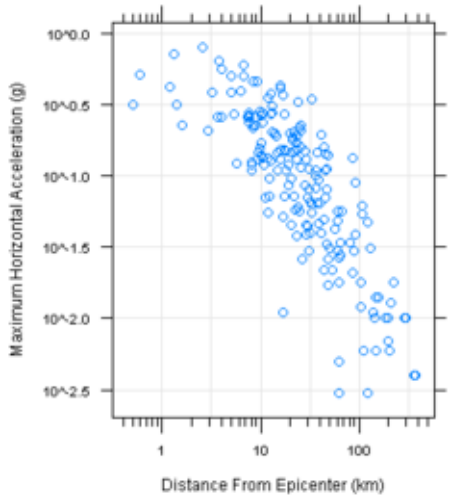
lattice

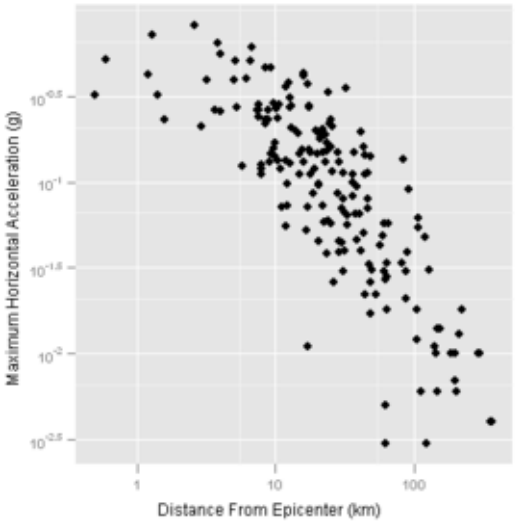
```
> pl <- xyplot(accel ~ distance, data = Earthquake, prepanel = prepanel.loess,
+     aspect = "xy", type = c("p", "g"), scales = list(log = 10),
+     xlab = "Distance From Epicenter (km)", ylab = "Maximum Horizontal Acceleration (g)",
+     xscale.components = xscale.components.log10)
> print(pl)
```

ggplot2

```
> ticks <- logTicks(range(Earthquake$distance), loc = c(1, 10))
> pg <- ggplot(Earthquake, aes(distance, accel)) + geom_point() +
+     scale_y_log10() + scale_x_log10(breaks = ticks, labels = ticks) +
+     xlab("Distance From Epicenter (km)") + ylab("Maximum Horizontal Acceleration (g)")
> print(pg)
```

Note
ggplot2 doesn't have the equivalent of `aspect="xy"` in `lattice`, which "*tries* to compute the aspect based on the 45 degree banking rule".





8.6 Figure 8.6

lattice

```
> axis.CF <- function(side, ...) {
+     if (side == "right") {
+         F2C <- function(f) 5 * (f - 32)/9
+         C2F <- function(c) 32 + 9 * c/5
+         ylim <- current.panel.limits()$ylim
+         prettyF <- pretty(ylim)
+         prettyC <- pretty(F2C(ylim))
+         panel.axis(side = side, outside = TRUE, at = prettyF,
+             tck = 5, line.col = "grey65", text.col = "grey35")
+         panel.axis(side = side, outside = TRUE, at = C2F(prettyC),
+             labels = as.character(prettyC), tck = 1, line.col = "black",
+             text.col = "black")
+     }
+     else axis.default(side = side, ...)
+ }
```

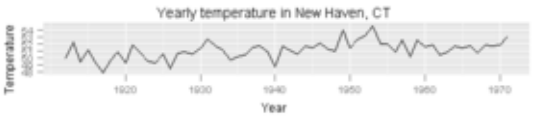
```
> pl <- xyplot(nhtemp ~ time(nhtemp), aspect = "xy", type = "o",
+     scales = list(y = list(alternating = 2, tck = c(1, 5))),
+     axis = axis.CF, xlab = "Year", ylab = "Temperature", main = "Yearly temperature in  ↩
    New Haven, CT",
+     key = list(text = list(c("(Celcius)", "(Fahrenheit)"), col = c("black",
+         "grey35")), columns = 2))
> print(pl)
```

ggplot2

```
> temp <- data.frame(nhtemp)
> temp$year <- seq(1912, 1971)
> pg <- ggplot(temp, aes(year, nhtemp)) + geom_line() + xlab("Year") +
+     ylab("Temperature") + opts(title = "Yearly temperature in New Haven, CT")
> print(pg)
```

Note
ggplot2 doesn't support the addition of secondary axes.


Yearly temperature in New Haven, CT
(Celsius)
(Fahrenheit)
Temperature
Year


Yearly temperature in New Haven, CT
Temperature
1920
1930
1940
1950
1960
1970
Year

Chapter 9

Labels and Legends

TOPICS COVERED:

- Labels (main, sub, xlab, ylab)
- Legends, color keys
- Legends in grouped displays; auto.key
- Dropping unused levels of grouping variable
- Page annotation

9.1 Figure 9.1

```
> library(lattice)
> library(ggplot2)
```

```
> data(Cars93, package = "MASS")
```

lattice

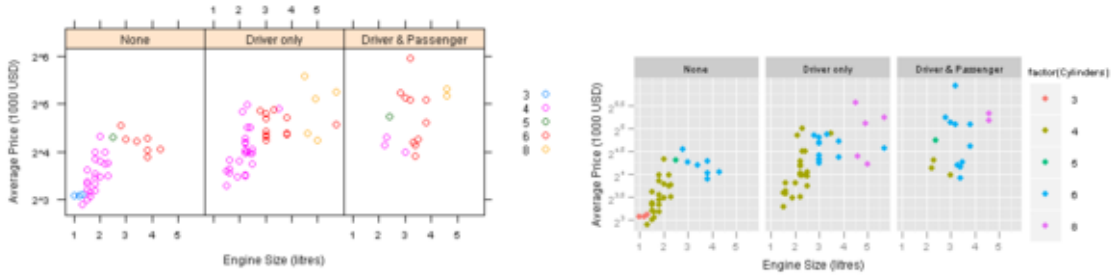
```
> sup.sym <- Rows(trellis.par.get("superpose.symbol"), 1:5)
```

```
> pl <- xyplot(Price ~ EngineSize | reorder(AirBags, Price), data = Cars93,
+     groups = Cylinders, subset = Cylinders != "rotary", scales = list(y = list(log = 2,
+         tick.number = 3)), xlab = "Engine Size (litres)", ylab = "Average Price (1000 USD ↩
    )",
+     key = list(text = list(levels(Cars93$Cylinders)[1:5]), points = sup.sym,
+         space = "right"))
> print(pl)
```

ggplot2

```
> Cars93$AirBags <- with(Cars93, reorder(AirBags, Price))
```

```
> pg <- ggplot(Cars93, aes(EngineSize, Price, colour = factor(Cylinders))) +
+     geom_point(subset = .(Cylinders != "rotary")) + facet_grid(~AirBags) +
+     xlab("Engine Size (litres)") + ylab("Average Price (1000 USD)") +
+     scale_y_log2()
> print(pg)
```



9.2 Figure 9.2

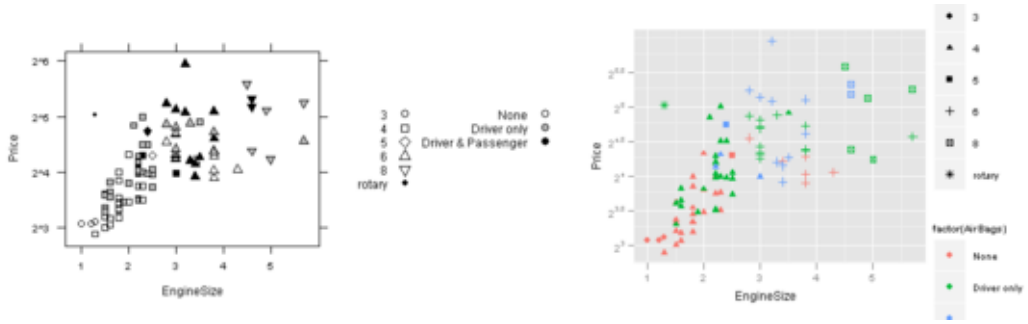
lattice

```
> my.pch <- c(21:25, 20)
> my.fill <- c("transparent", "grey", "black")
```

```
> pl <- with(Cars93, xyplot(Price ~ EngineSize, scales = list(y = list(log = 2,
+     tick.number = 3)), panel = function(x, y, ..., subscripts) {
+     pch <- my.pch[Cylinders[subscripts]]
+     fill <- my.fill[AirBags[subscripts]]
+     panel.xyplot(x, y, pch = pch, fill = fill, col = "black")
+ }, key = list(space = "right", adj = 1, text = list(levels(Cylinders)),
+     points = list(pch = my.pch), text = list(levels(AirBags)),
+     points = list(pch = 21, fill = my.fill), rep = FALSE)))
> print(pl)
```

ggplot2

```
> pg <- ggplot(Cars93, aes(EngineSize, Price, shape = factor(Cylinders),
+     colour = factor(AirBags))) + geom_point() + scale_y_log2()
> print(pg)
```



9.3 Figure 9.3

lattice

```
> library(latticeExtra)
> hc1 <- hclust(dist(USArrests, method = "canberra"))
> hc1 <- as.dendrogram(hc1)
> ord.hc1 <- order.dendrogram(hc1)
> hc2 <- reorder(hc1, state.region[ord.hc1])
> ord.hc2 <- order.dendrogram(hc2)
> region.colors <- trellis.par.get("superpose.polygon")$col
```

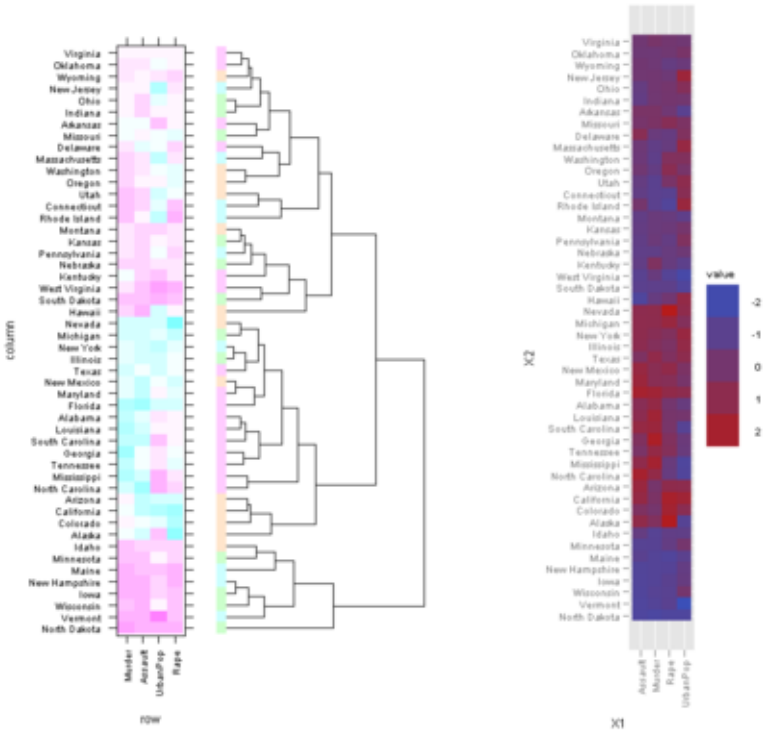
```
> pl <- levelplot(t(scale(USArrests))[, ord.hc2], scales = list(x = list(rot = 90)),
+     colorkey = FALSE, legend = list(right = list(fun = dendrogramGrob,
+         args = list(x = hc2, ord = ord.hc2, side = "right", size = 10,
+             size.add = 0.5, add = list(rect = list(col = "transparent",
+                 fill = region.colors[state.region])), type = "rectangle"))))
> print(pl)
```

ggplot2

```
> USArrests2 <- melt(t(scale(USArrests)))
> USArrests2$X2 <- factor(USArrests2$X2, levels = state.name[ord.hc2])
```

```
> pg <- ggplot(USArrests2, aes(X1, X2, fill = value)) + geom_tile() +
+     opts(axis.text.x = theme_text(angle = 90, hjust = 1, colour = "grey50")) +
+     opts(aspect.ratio = 50/5)
> print(pg)
```

Note
ggplot2 does not support the inclusion of dendrograms in the legend.



Chapter 10

Data Manipulation and Related Topics

TOPICS COVERED:

- Non-standard evaluation in the context of lattice
- The extended formula interface
- Combining data sources, subsetting
- Shingles
- Ordering levels of categorical variables
- Controlling the appearance of strips

10.1 Figure 10.1

```
> library(lattice)
> library(ggplot2)
```

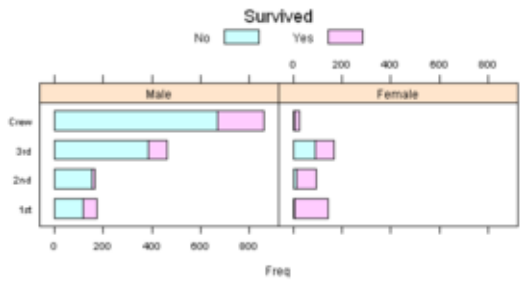
```
> Titanic1 <- as.data.frame(as.table(Titanic[, , "Adult", ]))
```

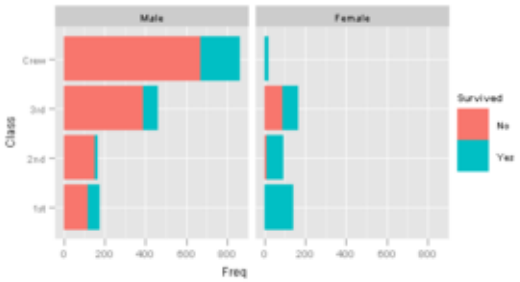
lattice

```
> pl <- barchart(Class ~ Freq | Sex, Titanic1, groups = Survived,
+     stack = TRUE, auto.key = list(title = "Survived", columns = 2))
> print(pl)
```

ggplot2

```
> pg <- ggplot(Titanic1, aes(Class, Freq, fill = Survived)) + geom_bar(stat = "identity") +
+     coord_flip() + facet_grid(~Sex)
> print(pg)
```





10.2 Figure 10.2

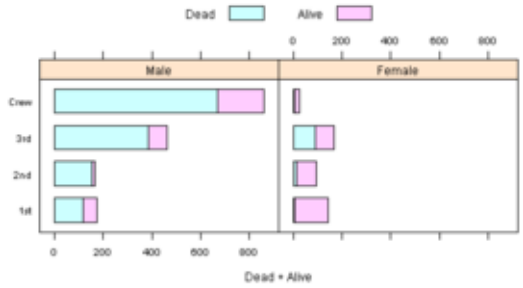
lattice

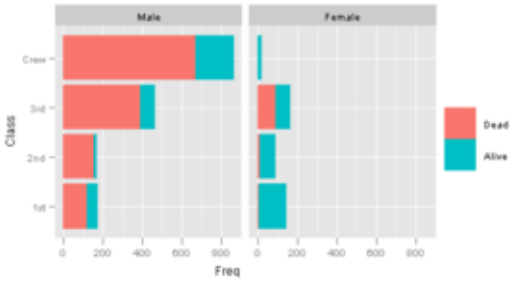
```
> Titanic2 <- reshape(Titanic1, direction = "wide", v.names = "Freq",
+     idvar = c("Class", "Sex"), timevar = "Survived")
> names(Titanic2) <- c("Class", "Sex", "Dead", "Alive")
```

```
> pl <- barchart(Class ~ Dead + Alive | Sex, Titanic2, stack = TRUE,
+     auto.key = list(columns = 2))
> print(pl)
```

ggplot2

```
> pg <- ggplot(Titanic1, aes(Class, Freq, fill = Survived)) + geom_bar(stat = "identity") +
+     coord_flip() + facet_grid(~Sex) + scale_fill_discrete("",
+     labels = c("Dead", "Alive"))
> print(pg)
```





10.3 Figure 10.3

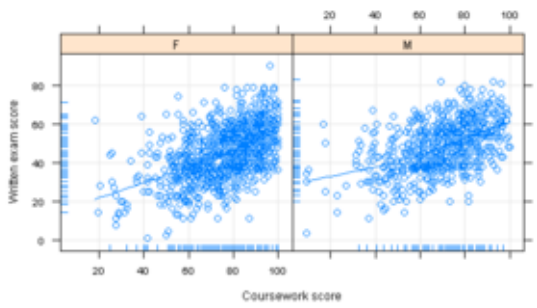
```
> data(Gcsemv, package = "mlmRev")
```

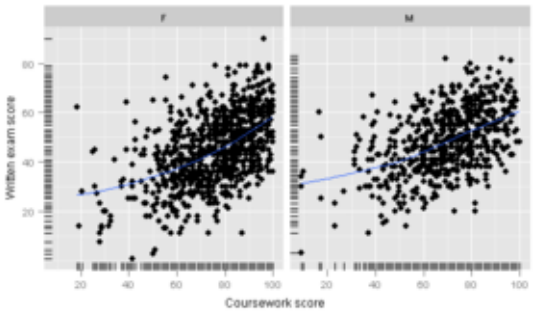
lattice

```
> pl <- xyplot(written ~ course | gender, data = Gcsemv, type = c("g",
+     "p", "smooth"), xlab = "Coursework score", ylab = "Written exam score",
+     panel = function(x, y, ...) {
+         panel.xyplot(x, y, ...)
+         panel.rug(x = x[is.na(y)], y = y[is.na(x)])
+     })
> print(pl)
```

ggplot2

```
> pg <- ggplot(Gcsemv, aes(course, written)) + geom_point() + geom_smooth(method = "loess",
+     se = F) + geom_rug() + facet_grid(~gender) + xlab("Coursework score") +
+     ylab("Written exam score")
> print(pg)
```





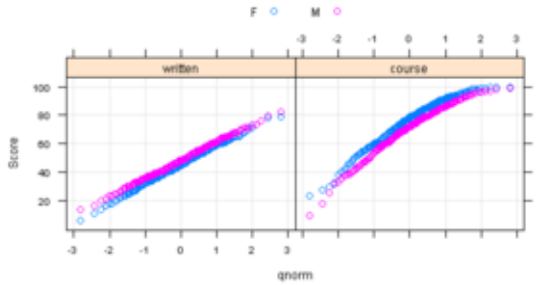
10.4 Figure 10.4

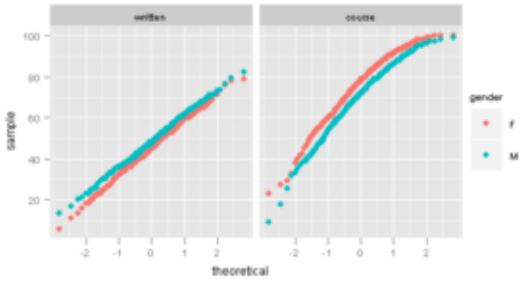
lattice

```
> pl <- qqmath(~written + course, Gcsemv, type = c("p", "g"), outer = TRUE,
+     groups = gender, auto.key = list(columns = 2), f.value = ppoints(200),
+     ylab = "Score")
> print(pl)
```

ggplot2

```
> pg <- ggplot(melt(Gcsemv, id.vars = 1:3, na.rm = TRUE)) + geom_point(aes(sample = value,
+     colour = gender), stat = "qq", quantiles = ppoints(200)) +
+     facet_grid(~variable)
> print(pg)
```





10.5 Figure 10.5

```
> set.seed(20051028)
> x1 <- rexp(2000)
> x1 <- x1[x1 > 1]
> x2 <- rexp(1000)
```

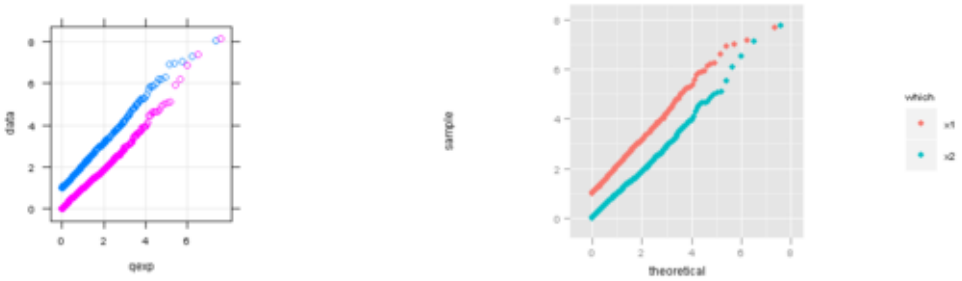
lattice

```
> pl <- qqmath(~data, make.groups(x1, x2), groups = which, distribution = qexp,
+     aspect = "iso", type = c("p", "g"))
> print(pl)
```

Note
`make.groups()` function in `lattice` package "combines two or more vectors, possibly of different lengths, producing a data frame with a second column indicating which of these vectors that row came from". Another alternative would be to use `combine()` function contained in `gtools` package.

ggplot2

```
> pg <- ggplot(make.groups(x1, x2)) + geom_point(aes(sample = data,
+     colour = which), stat = "qq", distribution = qexp) + coord_equal()
> print(pg)
```



10.6 Figure 10.6

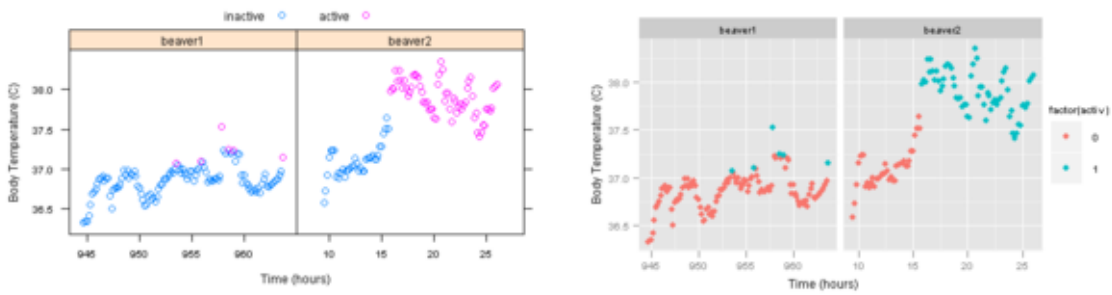
```
> beavers <- make.groups(beaver1, beaver2)
> beavers$hour <- with(beavers, time%/%100 + 24 * (day - 307) +
+     (time%%100)/60)
```

lattice

```
> pl <- xyplot(temp ~ hour | which, data = beavers, groups = activ,
+     auto.key = list(text = c("inactive", "active"), columns = 2),
+     xlab = "Time (hours)", ylab = "Body Temperature (C)", scales = list(x = list(relation ↩
    = "sliced")))
> print(pl)
```

ggplot2

```
> pg <- ggplot(beavers, aes(hour, temp, colour = factor(activ))) +
+     geom_point() + facet_grid(~which, scales = "free_x") + xlab("Time (hours)") +
+     ylab("Body Temperature (C)")
> print(pg)
```



10.7 Figure 10.7

```
> data(USAge.df, package = "latticeExtra")
```

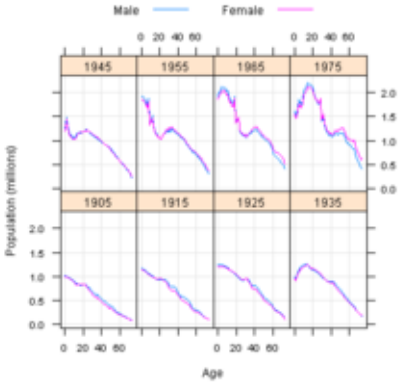
lattice

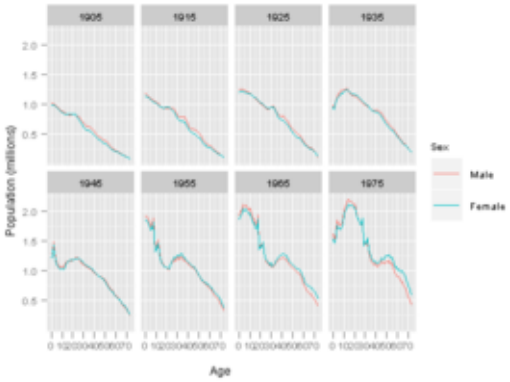
```
> pl <- xyplot(Population ~ Age | factor(Year), USAge.df, groups = Sex,
+     type = c("l", "g"), auto.key = list(points = FALSE, lines = TRUE,
+         columns = 2), aspect = "xy", ylab = "Population (millions)",
+     subset = Year %in% seq(1905, 1975, by = 10))
> print(pl)
```

ggplot2

```
> pg <- ggplot(USAge.df, aes(Age, Population, colour = Sex)) +
+     geom_line(subset = .(Year %in% seq(1905, 1975, by = 10))) +
+     facet_wrap(~Year, ncol = 4) + ylab("Population (millions)")
> print(pg)
```

Note
`ggplot2` doesn't have the equivalent of `aspect="xy"` in `lattice`, which "*tries* to compute the aspect based on the 45 degree banking rule".





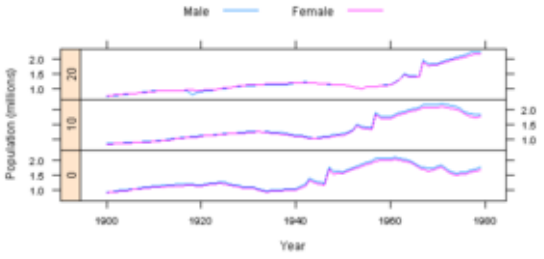
10.8 Figure 10.8

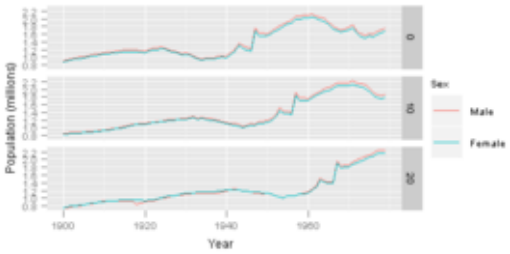
lattice

```
> pl <- xyplot(Population ~ Year | factor(Age), USAge.df, groups = Sex,
+     type = "l", strip = FALSE, strip.left = TRUE, layout = c(1,
+         3), ylab = "Population (millions)", auto.key = list(lines = TRUE,
+         points = FALSE, columns = 2), subset = Age %in% c(0,
+         10, 20))
> print(pl)
```

ggplot2

```
> pg <- ggplot(USAge.df, aes(Year, Population, colour = Sex)) +
+     geom_line(subset = .(Age %in% c(0, 10, 20))) + facet_grid(Age ~
+     .) + ylab("Population (millions)")
> print(pg)
```





10.9 Figure 10.9

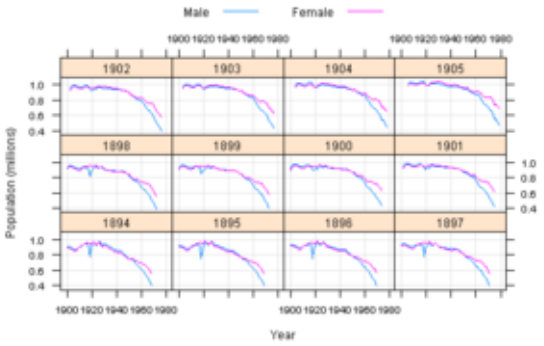
lattice

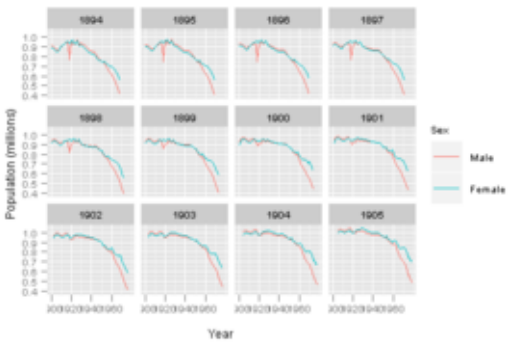
```
> pl <- xyplot(Population ~ Year | factor(Year - Age), USAge.df,
+     groups = Sex, subset = (Year - Age) %in% 1894:1905, type = c("g",
+         "l"), ylab = "Population (millions)", auto.key = list(lines = TRUE,
+         points = FALSE, columns = 2))
> print(pl)
```

ggplot2

```
> USAge.df$YearAge <- with(USAge.df, Year - Age)
```

```
> pg <- ggplot(USAge.df, aes(Year, Population, colour = Sex)) +
+     geom_line(subset = .((YearAge) %in% 1894:1905)) + facet_wrap(~YearAge) +
+     ylab("Population (millions)")
> print(pg)
```





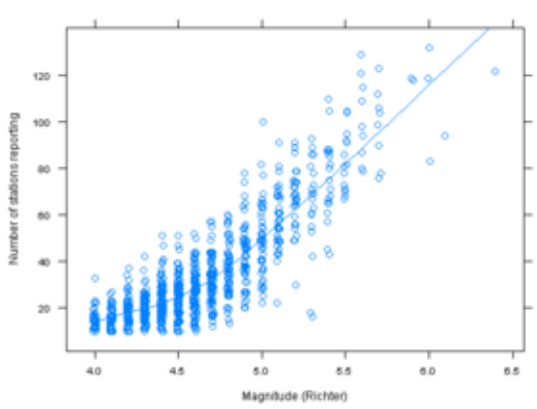
10.10 Figure 10.10

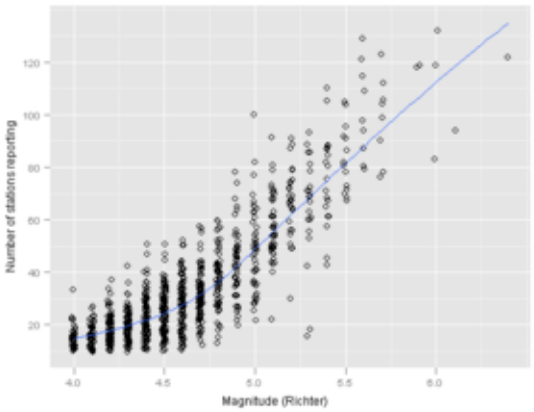
lattice

```
> pl <- xyplot(stations ~ mag, quakes, jitter.x = TRUE, type = c("p",
+     "smooth"), xlab = "Magnitude (Richter)", ylab = "Number of stations reporting")
> print(pl)
```

ggplot2

```
> pg <- ggplot(quakes, aes(mag, stations)) + geom_jitter(position = position_jitter(width = ↩
    0.01),
+     shape = 1) + geom_smooth(method = "loess", se = F) + xlab("Magnitude (Richter)") +
+     ylab("Number of stations reporting")
> print(pg)
```





10.11 Figure 10.11

```
> quakes$Mag <- equal.count(quakes$mag, number = 10, overlap = 0.2)
```

lattice

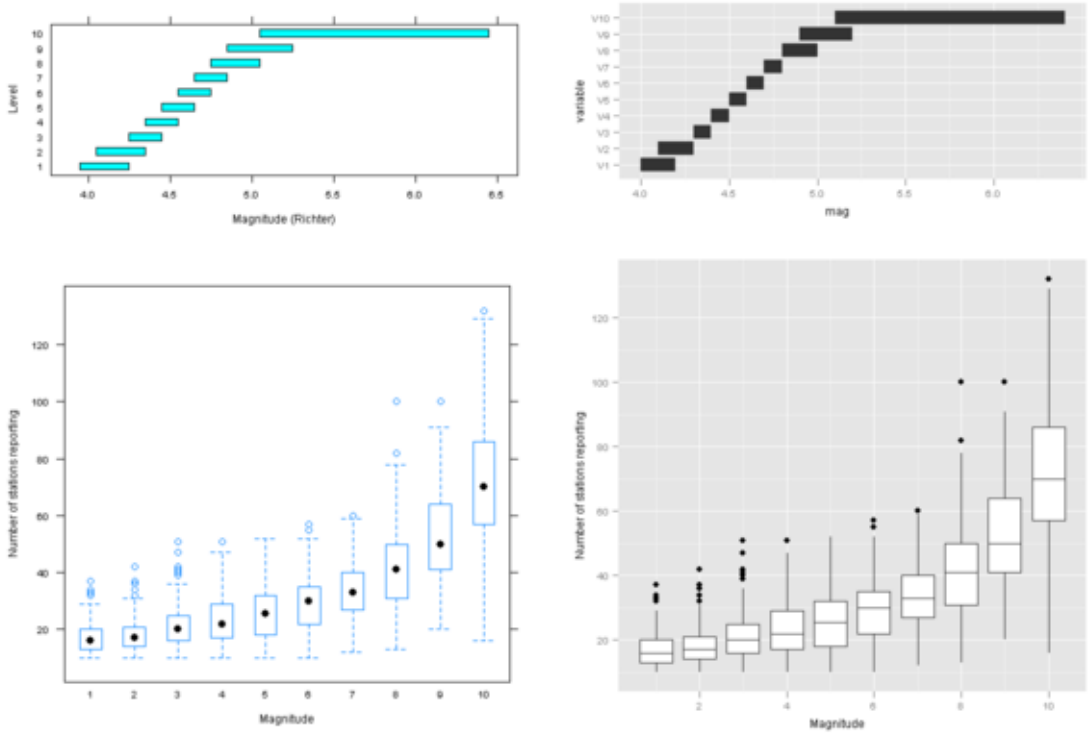
```
> ps.mag <- plot(quakes$Mag, ylab = "Level", xlab = "Magnitude (Richter)")
> bwp.quakes <- bwplot(stations ~ Mag, quakes, xlab = "Magnitude",
+     ylab = "Number of stations reporting")
> plot(bwp.quakes, position = c(0, 0, 1, 0.65))
> plot(ps.mag, position = c(0, 0.65, 1, 1), newpage = FALSE)
```

ggplot2

```
> Layout <- grid.layout(nrow = 2, ncol = 1, heights = unit(c(1,
+     2), c("null", "null")))
> grid.show.layout(Layout)
> vplayout <- function(...) {
+     grid.newpage()
+     pushViewport(viewport(layout = Layout))
+ }
> subplot <- function(x, y) viewport(layout.pos.row = x, layout.pos.col = y)
> pr <- function() {
+     vplayout()
+     print(p1, vp = subplot(1, 1))
+     print(p2, vp = subplot(2, 1))
+ }
```

```
> fn <- function(data = quakes$mag, number = 4, ...) {
+     intrv <<- as.data.frame(co.intervals(data, number, ...))
+     mag <- sort(unique(data))
+     intervals <- ldply(mag, function(x) {
+         t(as.numeric(x < intrv$V2 & x > intrv$V1))
+     })
+     tmp <- melt(cbind(mag, intervals), id.var = 1)
+     tmp[tmp$value > 0, 1:2]
+ }
> quakes.ordered <- merge(quakes, fn(number = 10, overlap = 0.2))
> intrv <- with(intrv, paste(V1, V2, sep = "-"))
> quakes.ordered <- rename(quakes.ordered, c(variable = "magnitude"))
> quakes.ordered$magnitude <- factor(quakes.ordered$magnitude,
+     labels = intrv)
```

```
> p1 <- ggplot(fn(number = 10, overlap = 0.2), aes(variable, mag)) +
+     stat_summary(aes(xmin = as.numeric(variable) - 0.4, xmax = as.numeric(variable) +
+         0.4), fun.ymin = min, fun.ymax = max, geom = "rect") +
+     coord_flip()
> p2 <- ggplot(quakes.ordered, aes(as.numeric(magnitude), stations,
+     group = magnitude)) + geom_boxplot() + xlab("Magnitude") +
+     ylab("Number of stations reporting")
> pr()
```



10.12 Figure 10.12

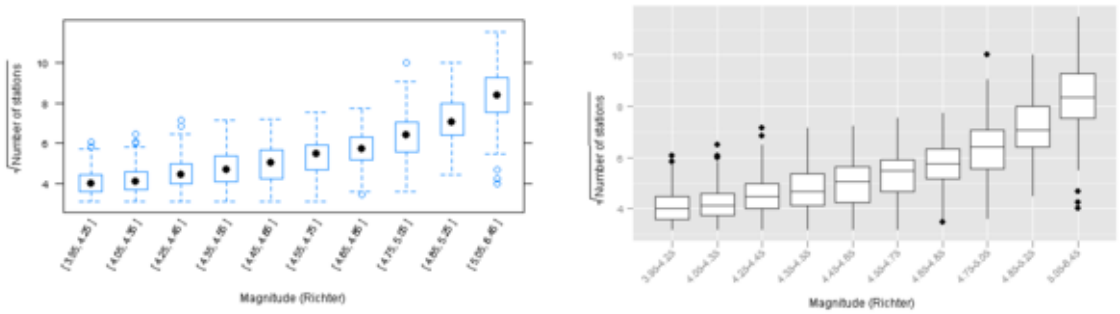
lattice

```
> pl <- bwplot(sqrt(stations) ~ Mag, quakes, scales = list(x = list(limits = as.character( ↩
    levels(quakes$Mag)),
+     rot = 60)), xlab = "Magnitude (Richter)", ylab = expression(sqrt("Number of stations ↩
    ")))
> print(pl)
```

ggplot2

```
> pg <- ggplot(quakes.ordered, aes(magnitude, sqrt(stations), group = magnitude)) +
+     geom_boxplot() + xlab("Magnitude (Richter)") + ylab(expression(sqrt("Number of  ↩
    stations"))) +
+     opts(axis.text.x = theme_text(angle = 45, hjust = 1, colour = "grey50"))
> print(pg)
```

Note
`ggplot2` doesn't have the equivalent of `aspect="xy"` in `lattice`, which "*tries* to compute the aspect based on the 45 degree banking rule".



10.13 Figure 10.13

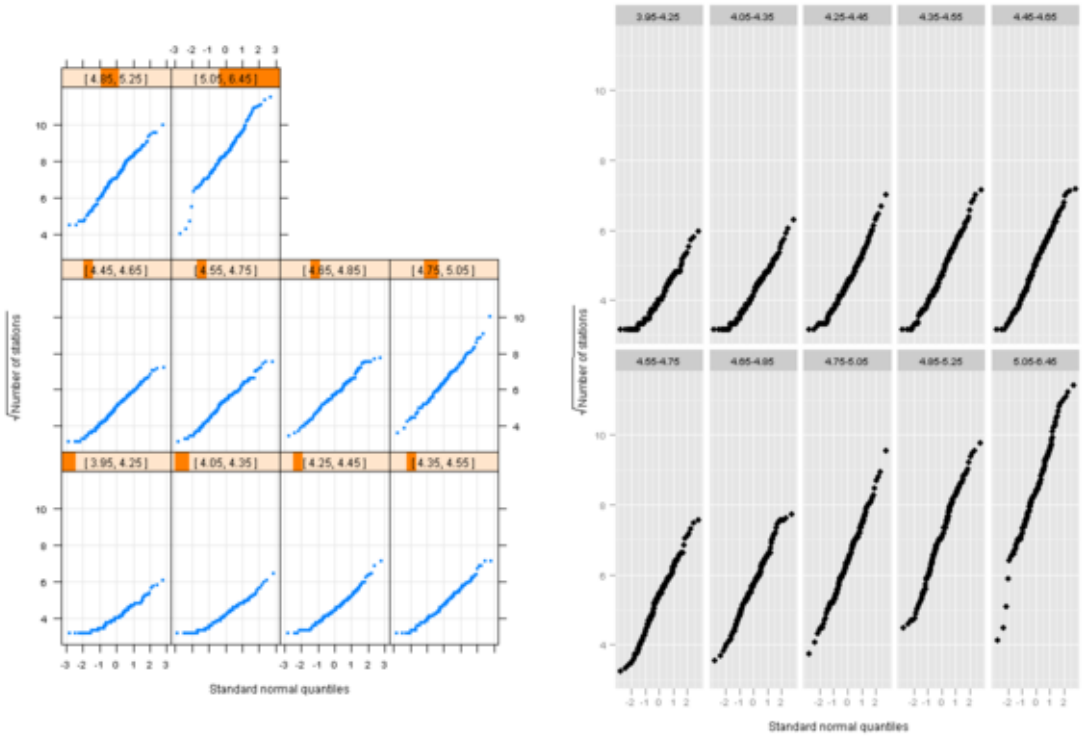
lattice

```
> pl <- qqmath(~sqrt(stations) | Mag, quakes, type = c("p", "g"),
+     pch = ".", cex = 3, prepanel = prepanel.qqmathline, aspect = "xy",
+     strip = strip.custom(strip.levels = TRUE, strip.names = FALSE),
+     xlab = "Standard normal quantiles", ylab = expression(sqrt("Number of stations")))
> print(pl)
```

ggplot2

```
> pg <- ggplot(quakes.ordered, aes(sample = sqrt(stations))) +
+     geom_point(stat = "qq") + facet_wrap(~magnitude, nrow = 2) +
+     scale_x_continuous("Standard normal quantiles") + scale_y_continuous(expression(sqrt ↩
    ("Number of stations")))
> print(pg)
```

Note
`ggplot2` doesn't have the equivalent of `aspect="xy"` in `lattice`, which "*tries* to compute the aspect based on the 45 degree banking rule".



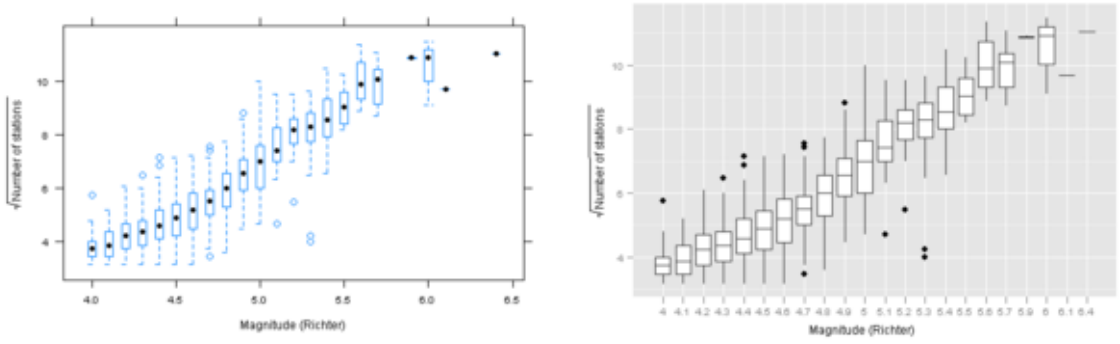
10.14 Figure 10.14

lattice

```
> pl <- xyplot(sqrt(stations) ~ mag, quakes, cex = 0.6, panel = panel.bwplot,
+     horizontal = FALSE, box.ratio = 0.05, xlab = "Magnitude (Richter)",
+     ylab = expression(sqrt("Number of stations")))
> print(pl)
```

ggplot2

```
> pg <- ggplot(quakes.ordered, aes(factor(mag), sqrt(stations))) +
+     geom_boxplot() + xlab("Magnitude (Richter)") + ylab(expression(sqrt("Number of  ↩
    stations")))
> print(pg)
```



10.15 Figure 10.15

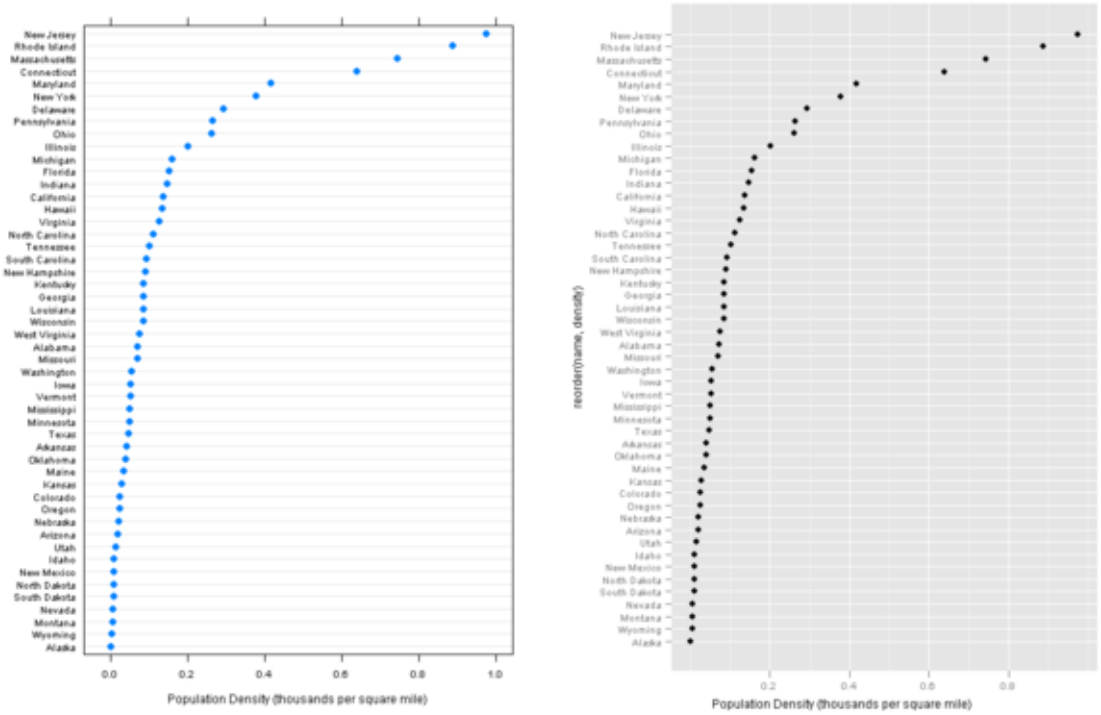
```
> state.density <- data.frame(name = state.name, area = state.x77[,
+     "Area"], population = state.x77[, "Population"], region = state.region)
> state.density$density <- with(state.density, population/area)
```

lattice

```
> pl <- dotplot(reorder(name, density) ~ density, state.density,
+     xlab = "Population Density (thousands per square mile)")
> print(pl)
```

ggplot2

```
> pg <- ggplot(state.density, aes(density, reorder(name, density))) +
+     geom_point() + xlab("Population Density (thousands per square mile)")
> print(pg)
```



10.16 Figure 10.16

lattice

```
> state.density$Density <- shingle(state.density$density, intervals = rbind(c(0,
+     0.2), c(0.2, 1)))
```

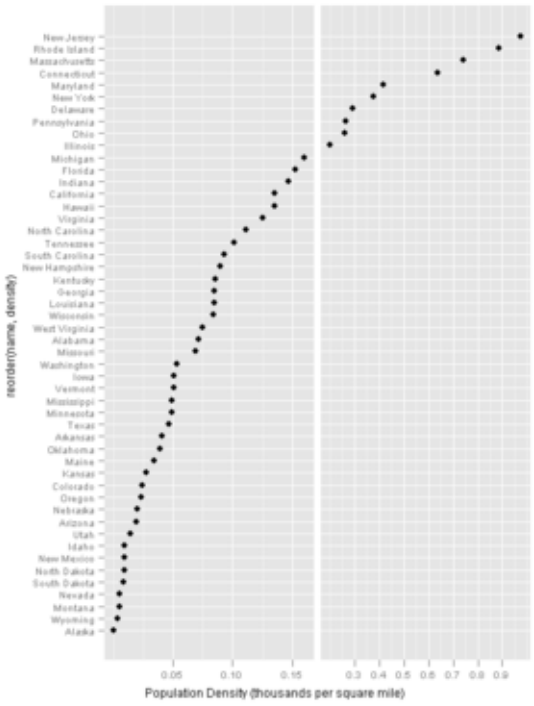
```
> pl <- dotplot(reorder(name, density) ~ density | Density, state.density,
+     strip = FALSE, layout = c(2, 1), levels.fos = 1:50, scales = list(x = "free"),
+     between = list(x = 0.5), xlab = "Population Density (thousands per square mile)",
+     par.settings = list(layout.widths = list(panel = c(2, 1))))
> print(pl)
```

ggplot2

```
> state.density$pos <- state.density$density > 0.2
```

```
> pg <- ggplot(state.density, aes(density, reorder(name, density))) +
+     geom_point() + facet_grid(~pos, scales = "free_x") + xlab("Population Density ( ↩
    thousands per square mile)") +
+     opts(strip.background = theme_blank(), strip.text.x = theme_blank())
> print(pg)
```

Note
It is not possible to change the facet width in `ggplot2`.



10.17 Figure 10.17

lattice

```
> cutAndStack <- function(x, number = 6, overlap = 0.1, type = "l",
+     xlab = "Time", ylab = deparse(substitute(x)), ...) {
+     time <- if (is.ts(x))
+         time(x)
+     else seq_along(x)
+     Time <- equal.count(as.numeric(time), number = number, overlap = overlap)
+     xyplot(as.numeric(x) ~ time | Time, type = type, xlab = xlab,
+         ylab = ylab, default.scales = list(x = list(relation = "free"),
+             y = list(relation = "free")), ...)
+ }
```

```
> pl <- cutAndStack(EuStockMarkets[, "DAX"], aspect = "xy", scales = list(x = list(draw =  ↩
    FALSE),
+     y = list(rot = 0)))
> print(pl)
```

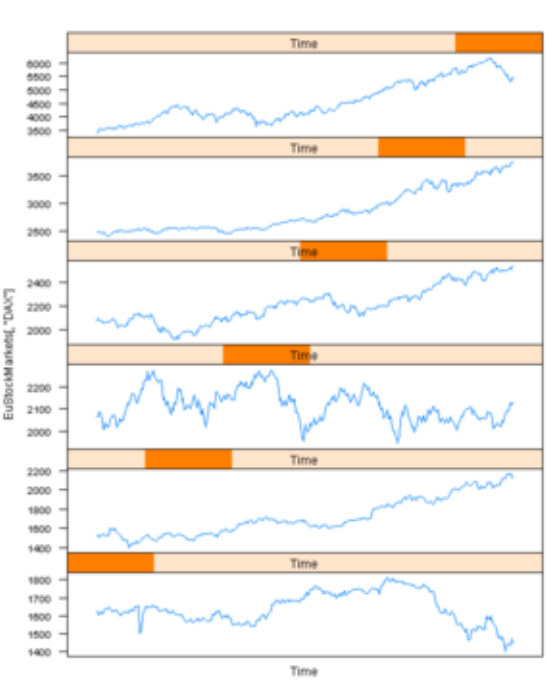
ggplot2

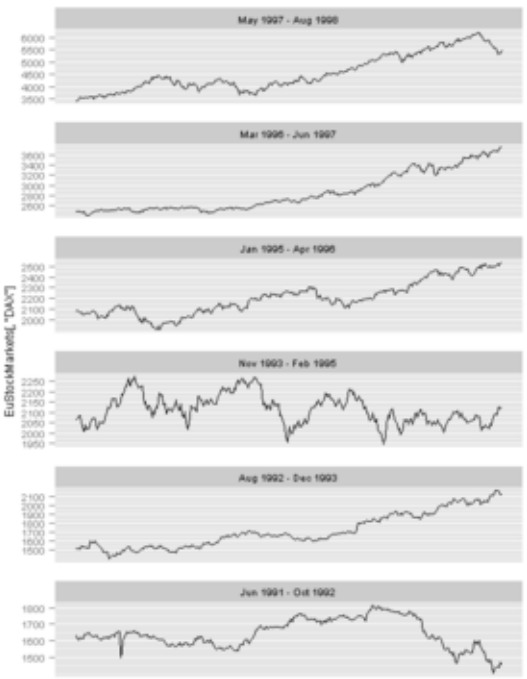
```
> library(zoo)
```

```
> cutAndStack_g <- function(data, number = 4, overlap = 0.1) {
+     ylab = deparse(substitute(data))
+     data <- as.data.frame(data)
+     data$id <- if (is.ts(data$x))
+         time(data$x)
+     else seq_along(data$x)
+     intrv <<- as.data.frame(co.intervals(data$id, number, overlap))
+     x <- sort(unique(data$id))
+     intervals <- ldply(x, function(x) {
+         t(as.numeric(x < intrv$V2 & x > intrv$V1))
+     })
+     tmp <- melt(cbind(x, intervals), id.var = 1)
+     tmp <- tmp[tmp$value > 0, 1:2]
+     tmp <- rename(tmp, c(x = "id"))
+     stock <- merge(data, tmp)
+     stock$variable <- factor(stock$variable, labels = with(intrv,
+         paste(as.yearmon(V1), as.yearmon(V2), sep = " - ")))
+     p <- ggplot(stock, aes(id, x)) + geom_line() + facet_wrap(~variable,
+         scales = "free", ncol = 1, as.table = FALSE) + scale_x_continuous("",
+         breaks = NA) + ylab(ylab)
+     print(p)
+ }
```

```
> cutAndStack_g(data = EuStockMarkets[, "DAX"], number = 6, overlap = 0.1)
```

Note
`ggplot2` doesn't have the equivalent of *aspect="xy"* in lattice, which "*tries* to compute the aspect based on the 45 degree banking rule".





10.18 Figure 10.18

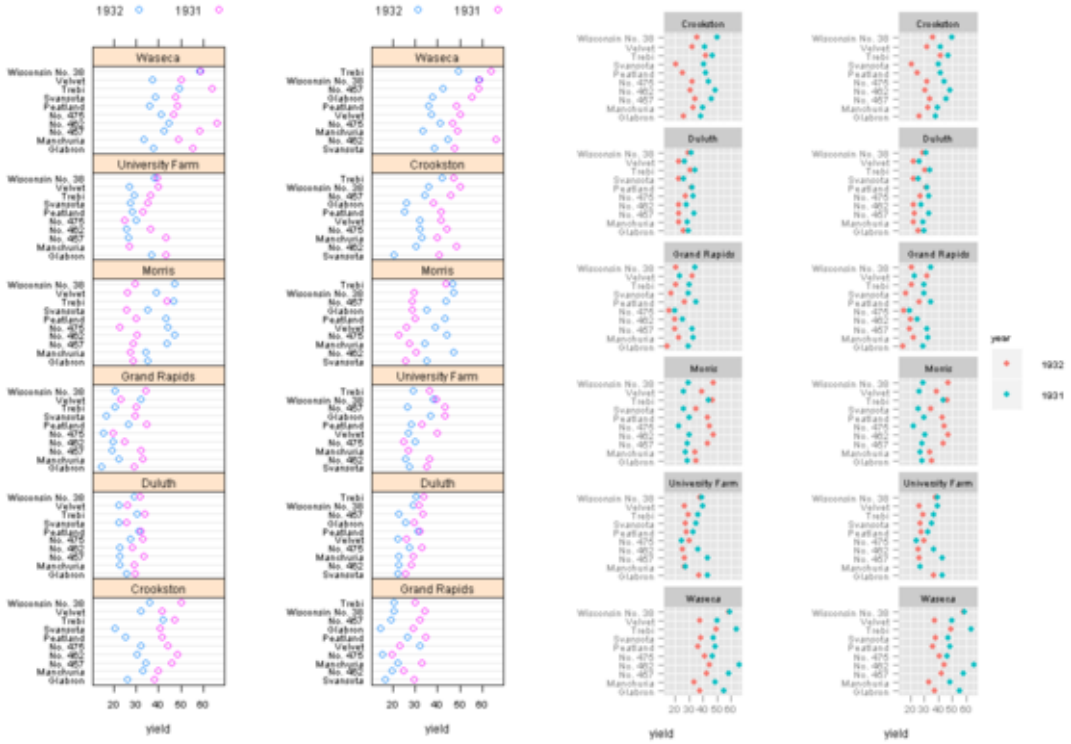
lattice

```
> bdp1 <- dotplot(as.character(variety) ~ yield | as.character(site),
+     barley, groups = year, layout = c(1, 6), auto.key = list(space = "top",
+         columns = 2), aspect = "fill")
> bdp2 <- dotplot(variety ~ yield | site, barley, groups = year,
+     layout = c(1, 6), auto.key = list(space = "top", columns = 2),
+     aspect = "fill")
> plot(bdp1, split = c(1, 1, 2, 1))
> plot(bdp2, split = c(2, 1, 2, 1), newpage = FALSE)
```

ggplot2

```
> Layout <- grid.layout(ncol = 3, widths = unit(c(2, 2, 0.75),
+     c("null", "null", "null")))
> vplayout <- function(...) {
+     grid.newpage()
+     pushViewport(viewport(layout = Layout))
+ }
> subplot <- function(x, y) viewport(layout.pos.row = x, layout.pos.col = y)
> pr <- function() {
+     vplayout()
+     print(p_left, vp = subplot(1, 1))
+     print(p_right, vp = subplot(1, 2))
+     print(legend, vp = subplot(1, 3))
+ }
```

```
> p <- ggplot(barley, aes(yield, variety, colour = year)) + geom_point() +
+     facet_wrap(~site, ncol = 1) + ylab("")
> legend <- p + opts(keep = "legend_box")
> p_right <- p + opts(legend.position = "none")
> barley$site <- factor(barley$site, levels = sort(levels(barley$site)))
> barley$variety <- factor(barley$variety, levels = sort(levels(barley$variety)))
> p_left <- p_right %+% barley
> pr()
```



10.19 Figure 10.19

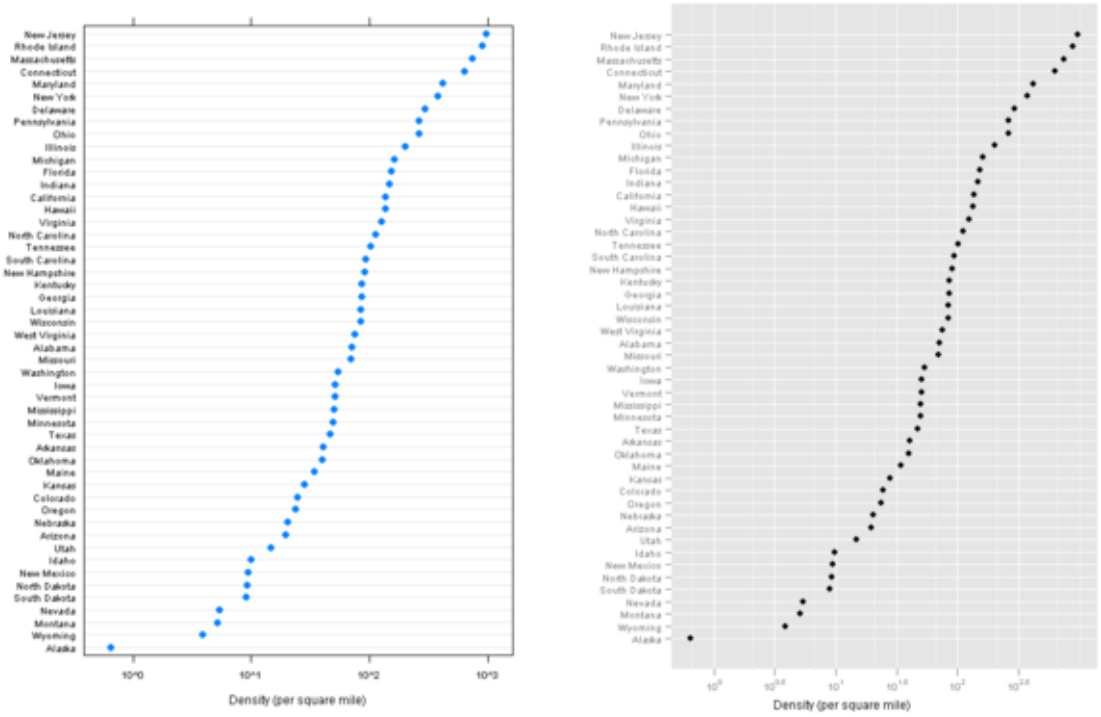
```
> state.density <- data.frame(name = state.name, area = state.x77[,
+     "Area"], population = state.x77[, "Population"], region = state.region)
> state.density$density <- with(state.density, population/area)
```

lattice

```
> pl <- dotplot(reorder(name, density) ~ 1000 * density, state.density,
+     scales = list(x = list(log = 10)), xlab = "Density (per square mile)")
> print(pl)
```

ggplot2

```
> pg <- ggplot(state.density, aes(density * 1000, reorder(name,
+     density))) + geom_point() + scale_x_log10() + xlab("Density (per square mile)") +
+     ylab("")
> print(pg)
```



10.20 Figure 10.20

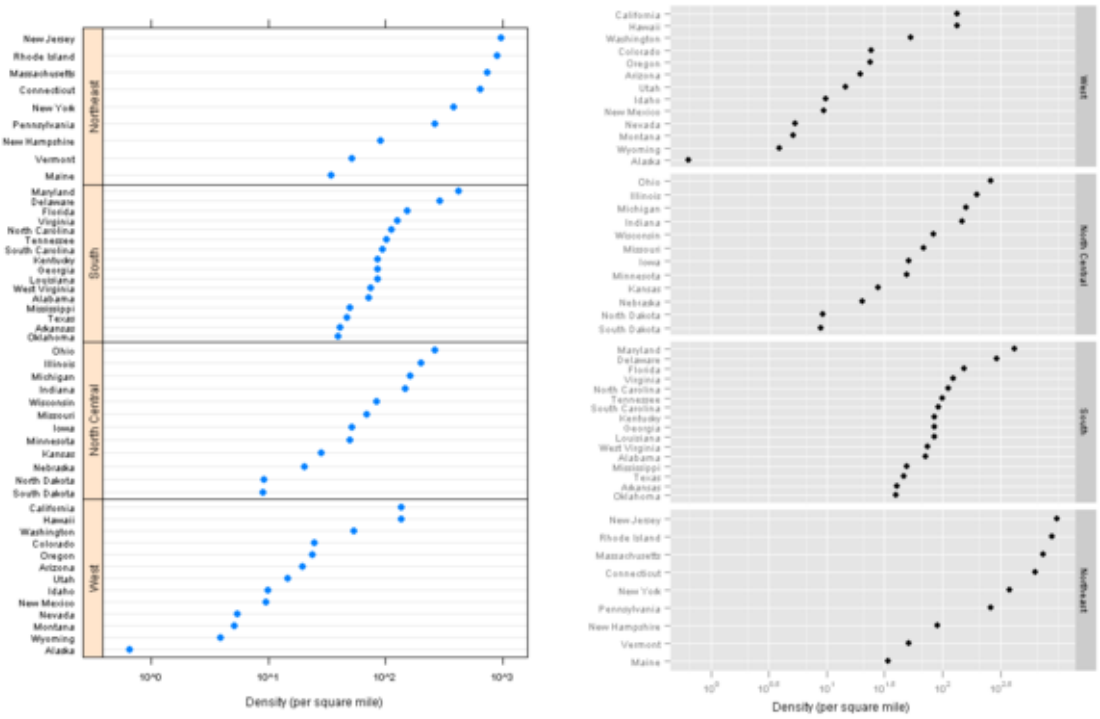
```
> state.density$region <- with(state.density, reorder(region, density,
+     median))
> state.density$name <- with(state.density, reorder(reorder(name,
+     density), as.numeric(region)))
```

lattice

```
> pl <- dotplot(name ~ 1000 * density | region, state.density,
+     strip = FALSE, strip.left = TRUE, layout = c(1, 4), scales = list(x = list(log = 10),
+         y = list(relation = "free")), xlab = "Density (per square mile)")
> print(pl)
```

ggplot2

```
> pg <- ggplot(state.density, aes(1000 * density, name)) + geom_point() +
+     facet_grid(region ~ ., scales = "free_y") + scale_x_log10() +
+     xlab("Density (per square mile)") + ylab("")
> print(pg)
```



10.21 Figure 10.21

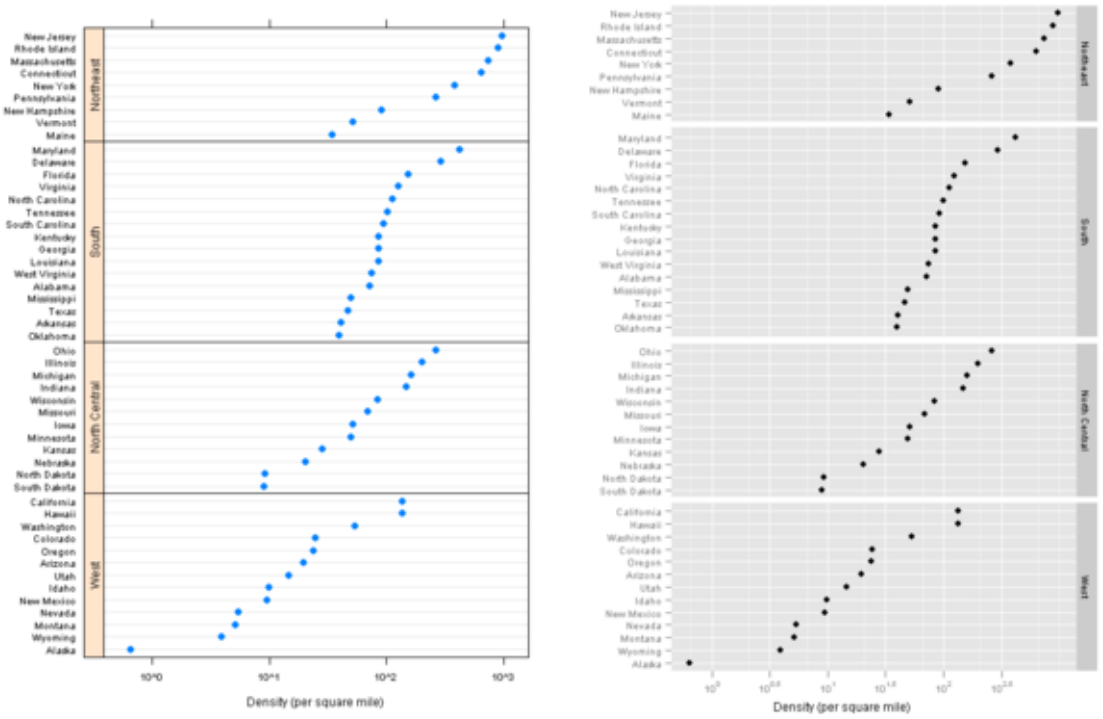
```
> library("latticeExtra")
```

lattice

```
> print(pl)
> print(resizePanels())
```

ggplot2

```
> pg <- pg + facet_grid(region ~ ., scales = "free_y", space = "free",
+     as.table = FALSE)
> print(pg)
```



10.22 Figure 10.22

```
> data(USCancerRates, package = "latticeExtra")
```

lattice

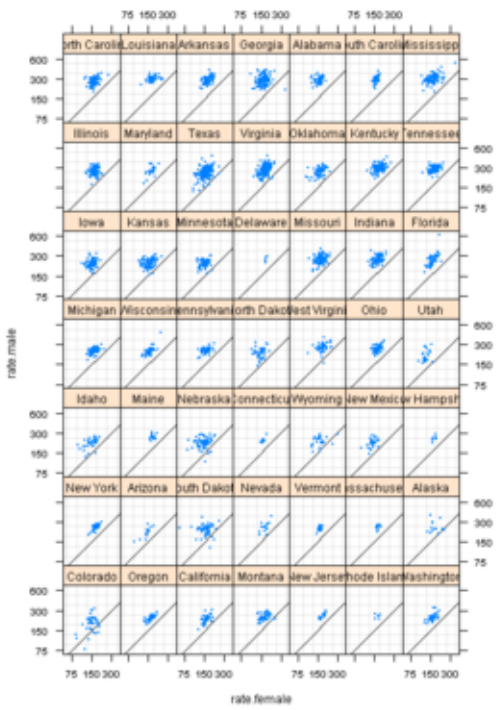
```
> pl <- xyplot(rate.male ~ rate.female | state, USCancerRates,
+     aspect = "iso", pch = ".", cex = 2, index.cond = function(x,
+         y) {
+         median(y - x, na.rm = TRUE)
+     }, scales = list(log = 2, at = c(75, 150, 300, 600)), panel = function(...) {
+         panel.grid(h = -1, v = -1)
+         panel.abline(0, 1)
+         panel.xyplot(...)
+     })
> print(pl)
```

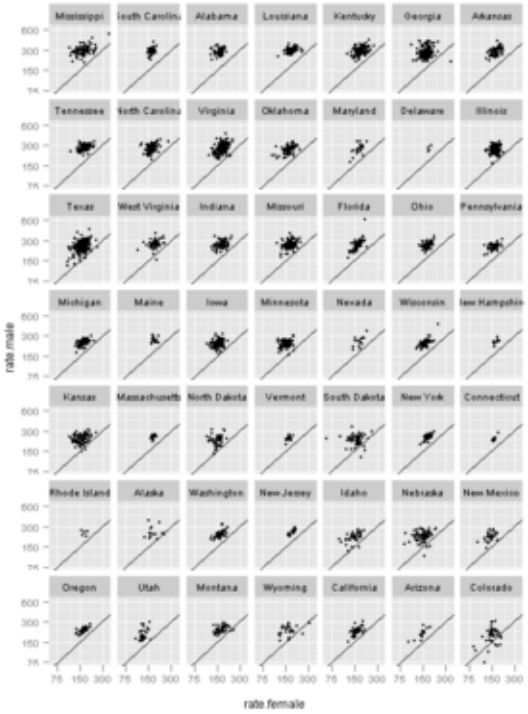
ggplot2

```
> med <- ddply(USCancerRates, .(state), summarise, med = median(rate.female -
+     rate.male, na.rm = TRUE))
> med$state <- with(med, reorder(state, med))
> USCancerRates$state <- factor(USCancerRates$state, levels = levels(med$state))
```

```
> pg <- ggplot(USCancerRates, aes(rate.female, rate.male)) + geom_point(size = 1) +
+     geom_abline(intercept = 0, slope = 1) + scale_x_log2(breaks = c(75,
+     150, 300, 600), labels = c(75, 150, 300, 600)) + scale_y_log2(breaks = c(75,
+     150, 300, 600), labels = c(75, 150, 300, 600)) + facet_wrap(~state,
+     ncol = 7)
> print(pg)
```

Note
For some strange reason the order of facets is different compared to `lattice` plot, although the formula to set the levels is the same!





10.23 Figure 10.23

```
> data(Chem97, package = "mlmRev")
```

lattice

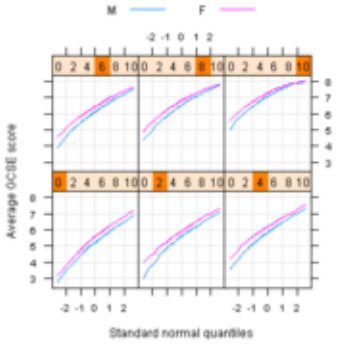
```
> strip.style4 <- function(..., style) {
+     strip.default(..., style = 4)
+ }
```

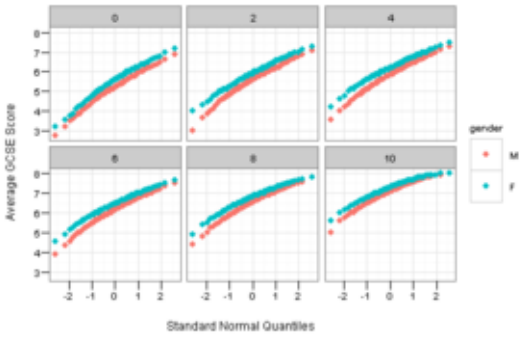
```
> pl <- qqmath(~gcsescore | factor(score), Chem97, groups = gender,
+     type = c("l", "g"), aspect = "xy", auto.key = list(points = FALSE,
+         lines = TRUE, columns = 2), f.value = ppoints(100), strip = strip.style4,
+     xlab = "Standard normal quantiles", ylab = "Average GCSE score")
> print(pl)
```

ggplot2

```
> pg <- ggplot(Chem97, aes(sample = gcsescore, colour = gender)) +
+     geom_point(stat = "qq", quantiles = ppoints(100)) + facet_wrap(~score) +
+     scale_x_continuous("Standard Normal Quantiles") + scale_y_continuous("Average GCSE  ↩
    Score") +
+     theme_bw()
> print(pg)
```

Note
`ggplot2` doesn't have the equivalent of `aspect="xy"` in `lattice`, which "*tries* to compute the aspect based on the 45 degree banking rule".





10.24 Figure 10.24

lattice

```
> strip.combined <- function(which.given, which.panel, factor.levels,
+     ...) {
+     if (which.given == 1) {
+         panel.rect(0, 0, 1, 1, col = "grey90", border = 1)
+         panel.text(x = 0, y = 0.5, pos = 4, lab = factor.levels[which.panel[which.given ↩
    ]])
+     }
+     if (which.given == 2) {
+         panel.text(x = 1, y = 0.5, pos = 2, lab = factor.levels[which.panel[which.given ↩
    ]])
+     }
+ }
```

```
> pl <- qqmath(~gcsescore | factor(score) + gender, Chem97, f.value = ppoints(100),
+     type = c("l", "g"), aspect = "xy", strip = strip.combined,
+     par.strip.text = list(lines = 0.5), xlab = "Standard normal quantiles",
+     ylab = "Average GCSE score")
> print(pl)
```

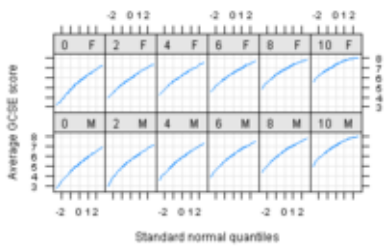
ggplot2

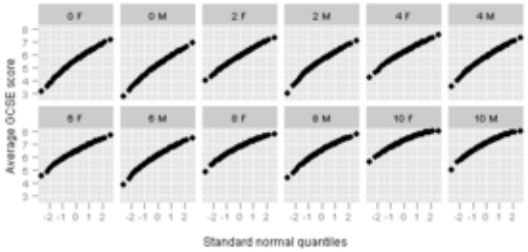
```
> Chem97 <- transform(Chem97, sc_gen = paste(score, gender))
```

```
> library(gtools)
> Chem97$sc_gen <- factor(Chem97$sc_gen, levels = mixedsort(unique(Chem97$sc_gen)))
```

```
> pg <- ggplot(Chem97, aes(sample = gcsescore)) + geom_point(stat = "qq",
+     quantiles = ppoints(100)) + facet_wrap(~sc_gen, nrow = 2) +
+     scale_x_continuous("Standard normal quantiles") + scale_y_continuous("Average GCSE  ↩
    score")
> print(pg)
```

Note
`ggplot2` doesn't have the equivalent of `aspect="xy"` in `lattice`, which "*tries* to compute the aspect based on the 45 degree banking rule".





10.25 Figure 10.25

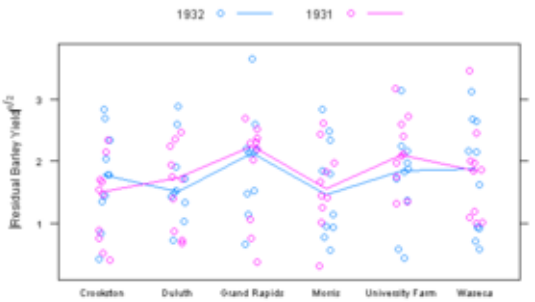
```
> morris <- barley$site == "Morris"
> barley$year[morris] <- ifelse(barley$year[morris] == "1931",
+     "1932", "1931")
```

lattice

```
> pl <- stripplot(sqrt(abs(residuals(lm(yield ~ variety + year +
+     site)))) ~ site, data = barley, groups = year, jitter.data = TRUE,
+     auto.key = list(points = TRUE, lines = TRUE, columns = 2),
+     type = c("p", "a"), fun = median, ylab = expression(abs("Residual Barley Yield")^{
+         1/2
+     }))
> print(pl)
```

ggplot2

```
> pg <- ggplot(barley, aes(site, sqrt(abs(residuals(lm(yield ~
+     variety + year + site)))), colour = year)) + geom_jitter(position = position_jitter(w ↩
    = 0.1)) +
+     geom_line(stat = "summary", fun.y = median, aes(group = year)) +
+     ylab(expression(abs("Residual Barley Yield")^{
+         1/2
+     }))
> print(pg)
```



Chapter 11

Manipulating the "trellis" object

TOPICS COVERED:

- Methods for "trellis" objects
- Tukey mean-difference plot
- Other specialized manipulations

11.1 Figure 11.1

```
> library(lattice)
> library(ggplot2)
```

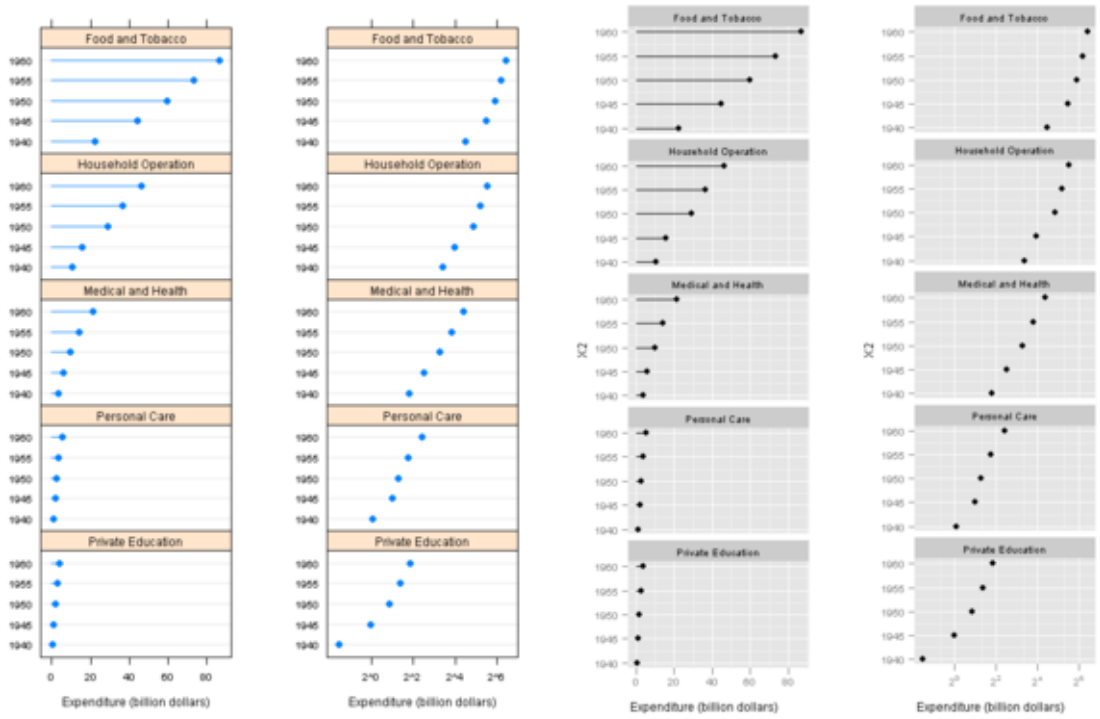
lattice

```
> dp.uspe <- dotplot(t(USPersonalExpenditure), groups = FALSE,
+     index.cond = function(x, y) median(x), layout = c(1, 5),
+     type = c("p", "h"), xlab = "Expenditure (billion dollars)")
> dp.uspe.log <- dotplot(t(USPersonalExpenditure), groups = FALSE,
+     index.cond = function(x, y) median(x), layout = c(1, 5),
+     scales = list(x = list(log = 2)), xlab = "Expenditure (billion dollars)")
> plot(dp.uspe, split = c(1, 1, 2, 1), more = TRUE)
> plot(dp.uspe.log, split = c(2, 1, 2, 1), more = FALSE)
```

ggplot2

```
> library(ggextra)
```

```
> p <- ggplot(melt(USPersonalExpenditure), aes(X2, value, ymin = 0,
+     ymax = value)) + geom_point() + coord_flip() + facet_wrap(~X1,
+     ncol = 1) + ylab("Expenditure (billion dollars)")
> p1 <- p + geom_linerange()
> p2 <- p + scale_y_log2()
> print(arrange(p1, p2))
```



11.2 Figure 11.2

lattice

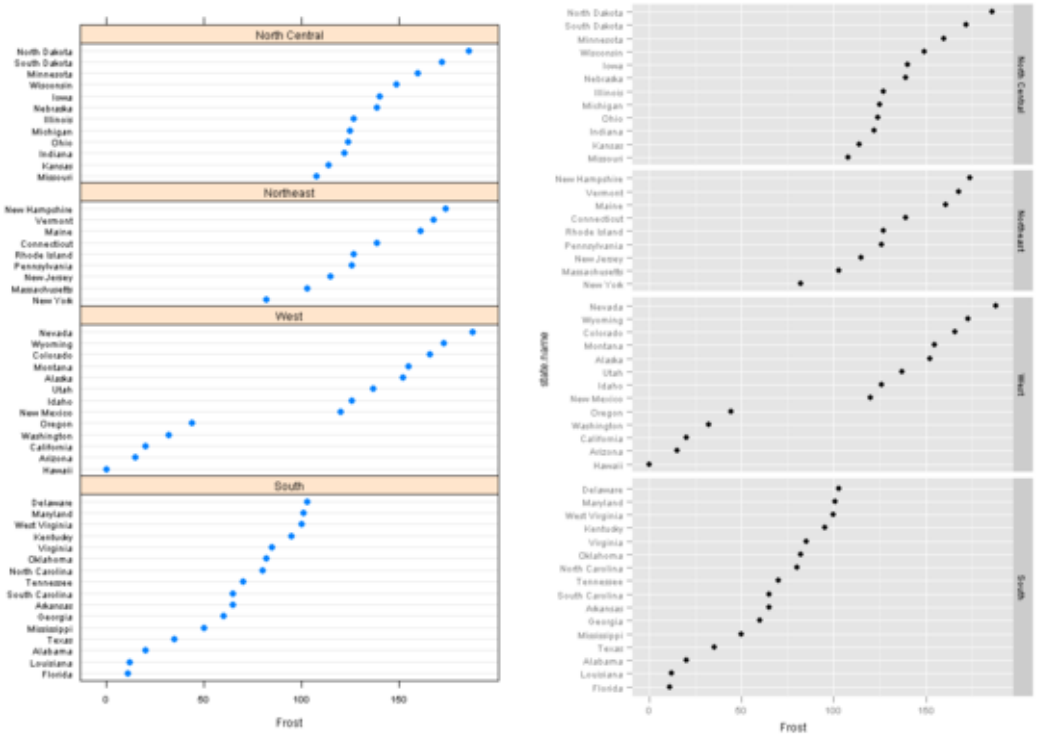
```
> state <- data.frame(state.x77, state.region, state.name)
> state$state.name <- with(state, reorder(reorder(state.name, Frost),
+     as.numeric(state.region)))
```

```
> dpfrost <- dotplot(state.name ~ Frost | reorder(state.region,
+     Frost), data = state, layout = c(1, 4), scales = list(y = list(relation = "free")))
> plot(dpfrost, panel.height = list(x = c(16, 13, 9, 12), unit = "null"))
```

ggplot2

```
> state$state.region <- with(state, reorder(state.region, Frost))
```

```
> pg <- ggplot(state, aes(Frost, state.name)) + geom_point() +
+     facet_grid(state.region ~ ., scales = "free_y", space = "free",
+         as.table = F)
> print(pg)
```



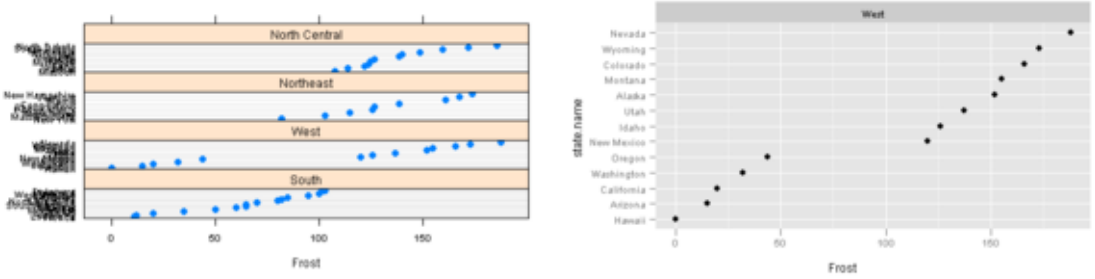
11.3 Figure 11.3

lattice

```
> plot(dpfrost, panel.height = list(x = c(16, 13, 9, 12), unit = "null"))
> print(update(trellis.last.object(), layout = c(1,
+     1))[2])
```

ggplot2

```
> pg <- pg %+% subset(state, state.region == "West") + facet_wrap(~state.region,
+     scales = "free_y")
> print(pg)
```



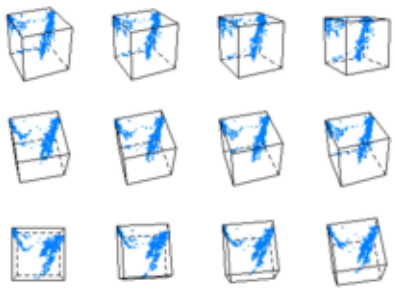
11.4 Figure 11.4

lattice

```
> npanel <- 12
> rot <- list(z = seq(0, 30, length = npanel), x = seq(0, -80,
+     length = npanel))
> quakeLocs <- cloud(depth ~ long + lat, quakes, pch = ".", cex = 1.5,
+     panel = function(..., screen) {
+         pn <- panel.number()
+         panel.cloud(..., screen = list(z = rot$z[pn], x = rot$x[pn]))
+     }, xlab = NULL, ylab = NULL, zlab = NULL, scales = list(draw = FALSE),
+     zlim = c(690, 30), par.settings = list(axis.line = list(col = "transparent")))
> pl <- quakeLocs[rep(1, npanel)]
> print(pl)
```

ggplot2

```
True 3d not supported in ggplot2.
```



11.5 Figure 11.5

```
> data(Chem97, package = "mlmRev")
```

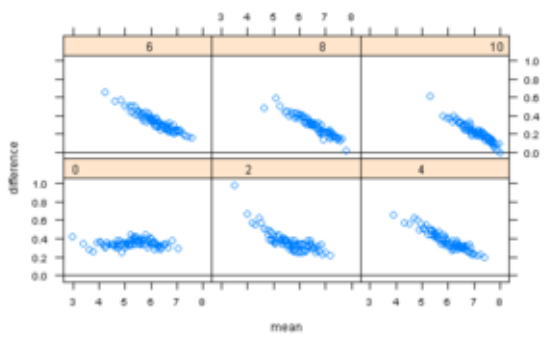
lattice

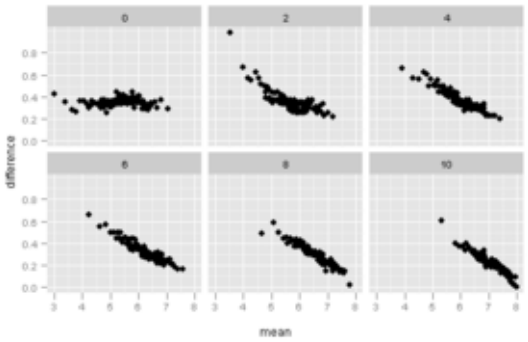
```
> ChemQQ <- qq(gender ~ gcsescore | factor(score), Chem97, f.value = ppoints(100),
+     strip = strip.custom(style = 5))
> pl <- tmd(ChemQQ)
> print(pl)
```

ggplot2

```
> q <- function(x, probs = ppoints(100)) {
+     data.frame(q = probs, value = quantile(x, probs))
+ }
> Chem97.q <- ddply(Chem97, c("gender", "score"), function(df) q(df$gcsescore))
> Chem97.df <- recast(Chem97.q, score + q ~ gender, id.var = 1:3)
```

```
> pg <- ggplot(Chem97.df, aes(M, F)) + geom_point(aes(x = (M +
+     F)/2, y = F - M)) + facet_wrap(~score) + xlab("mean") + ylab("difference")
> print(pg)
```





11.6 Figure 11.6

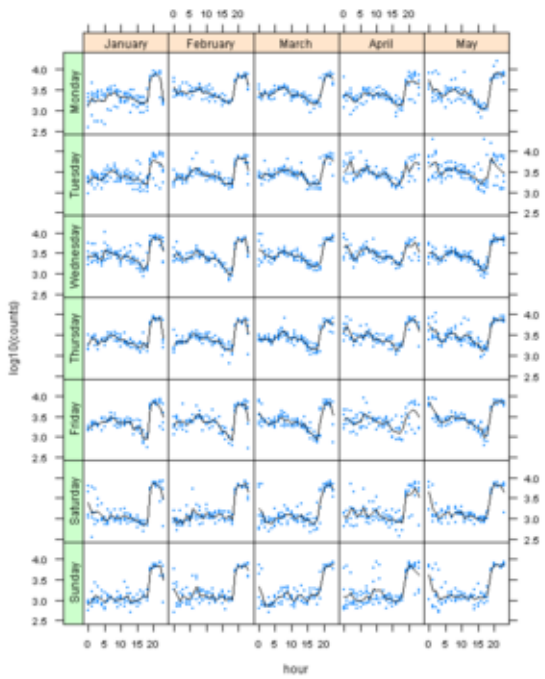
```
> library("latticeExtra")
> data(biocAccess)
```

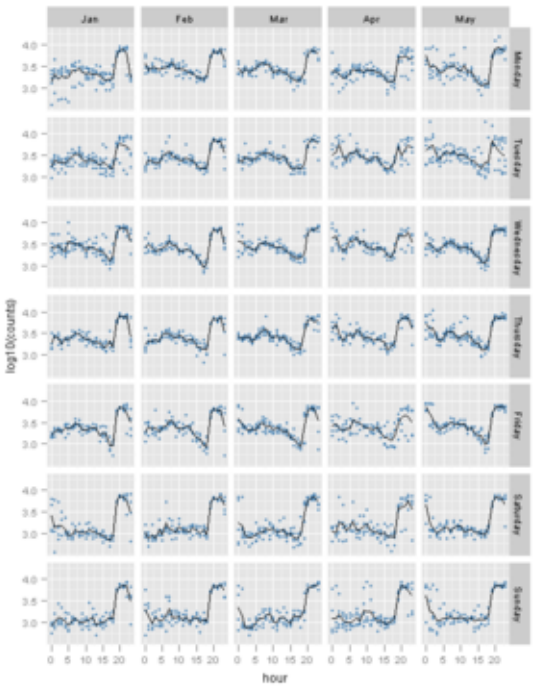
lattice

```
> baxy <- xyplot(log10(counts) ~ hour | month + weekday, biocAccess,
+     type = c("p", "a"), as.table = TRUE, pch = ".", cex = 2,
+     col.line = "black")
> dimnames(baxy)$month <- month.name[1:5]
> pl <- useOuterStrips(baxy)
> print(pl)
```

ggplot2

```
> pg <- ggplot(biocAccess, aes(hour, log10(counts))) + geom_point(colour = "steelblue",
+     size = 1) + geom_line(stat = "summary", fun.y = mean) + facet_grid(weekday ~
+     month)
> print(pg)
```





Chapter 12

Advanced Panel Functions

TOPICS COVERED:

- Built-in panel and accessors functions
- Examples

12.1 Figure 13.1

```
> library(lattice)
> library(ggplot2)
```

```
> grid <- data.frame(p = 11:30, q = 10)
> grid$k <- with(grid, factor(p/q))
```

```
> panel.hypotrochoid <- function(r, d, cycles = 10, density = 30) {
+     if (missing(r))
+         r <- runif(1, 0.25, 0.75)
+     if (missing(d))
+         d <- runif(1, 0.25 * r, r)
+     t <- 2 * pi * seq(0, cycles, by = 1/density)
+     x <- (1 - r) * cos(t) + d * cos((1 - r) * t/r)
+     y <- (1 - r) * sin(t) - d * sin((1 - r) * t/r)
+     panel.lines(x, y)
+ }
> panel.hypocycloid <- function(x, y, cycles = x, density = 30) {
+     panel.hypotrochoid(r = x/y, d = x/y, cycles = cycles, density = density)
+ }
> prepanel.hypocycloid <- function(x, y) {
+     list(xlim = c(-1, 1), ylim = c(-1, 1))
+ }
```

lattice

```
> pl <- xyplot(p ~ q | k, grid, aspect = 1, scales = list(draw = FALSE),
+     prepanel = prepanel.hypocycloid, panel = panel.hypocycloid)
> print(pl)
```

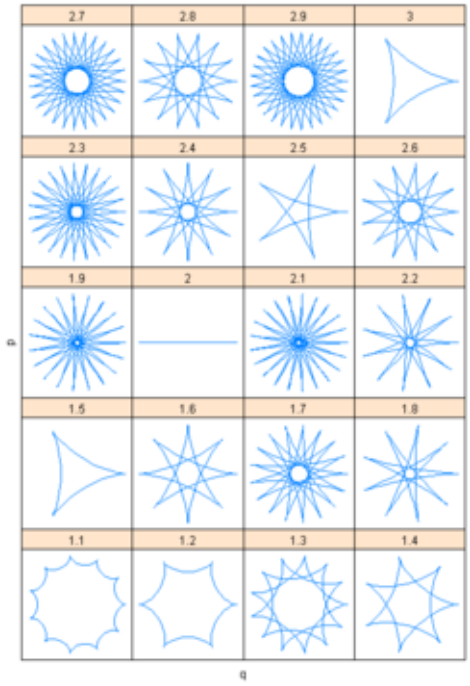
ggplot2

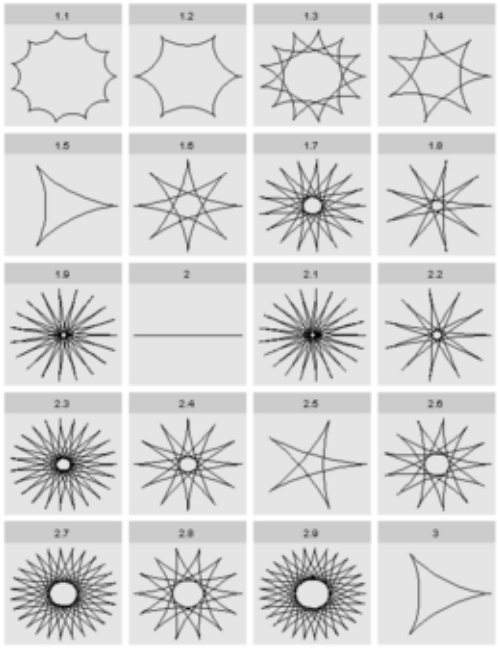
```
> panel.hypotrochoid.gg <- function(r, d, cycles = 10, density = 30) {
+     if (missing(r))
+         r <- runif(1, 0.25, 0.75)
+     if (missing(d))
+         d <- runif(1, 0.25 * r, r)
+     t <- 2 * pi * seq(0, cycles, by = 1/density)
+     x <- (1 - r) * cos(t) + d * cos((1 - r) * t/r)
+     y <- (1 - r) * sin(t) - d * sin((1 - r) * t/r)
+     data.frame(x, y)
+ }
> panel.hypocycloid.gg <- function(x, y, cycles = x, density = 30) {
+     panel.hypotrochoid.gg(r = x/y, d = x/y, cycles = cycles,
+         density = density)
+ }
```

Note
`panel.lines(x, y)` replaced with `data.frame(x, y)` in the `panel.hypotrochoid.gg` function.

```
> df <- ddply(grid, .(p, q, k), function(df) {
+     with(df, panel.hypocycloid.gg(q, p))
+ })
```

```
> pg <- ggplot(df, aes(x, y)) + geom_path() + facet_wrap(~k, ncol = 4) +
+     scale_x_continuous("", breaks = NA) + scale_y_continuous("",
+     breaks = NA)
> print(pg)
```





12.2 Figure 13.2

lattice

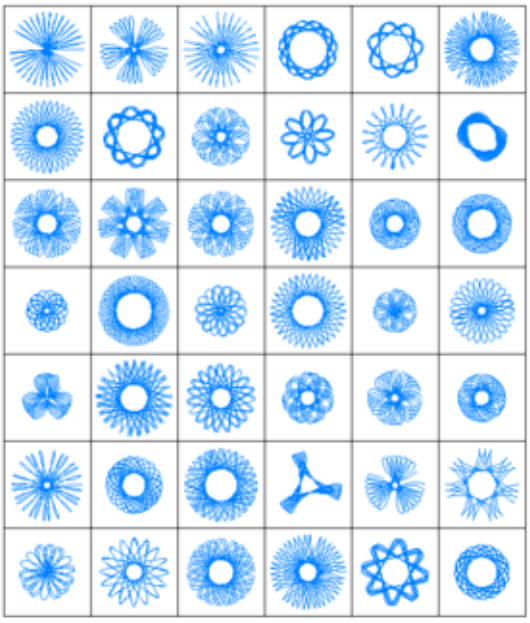
```
> set.seed(20070706)
```

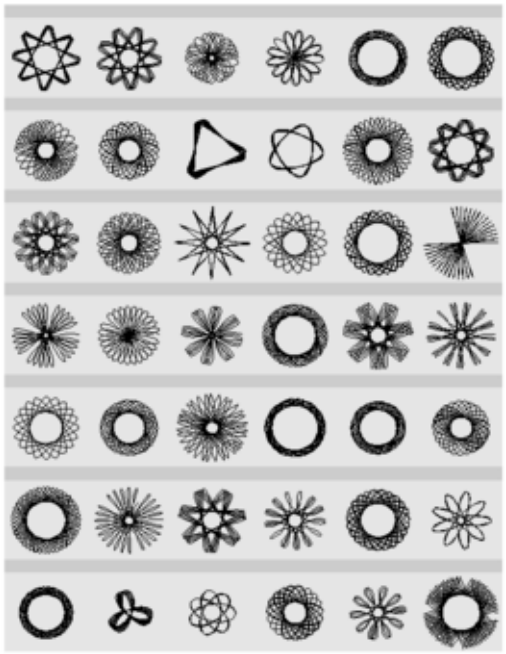
```
> pl <- xyplot(c(-1, 1) ~ c(-1, 1), aspect = 1, cycles = 15, scales = list(draw = FALSE),
+     xlab = "", ylab = "", panel = panel.hypotrochoid)
> print(pl[rep(1, 42)])
```

ggplot2

```
> df2 <- ldply(rep(1:42), function(k) {
+     data.frame(k, panel.hypotrochoid.gg(cycles = 15))
+ })
```

```
> pg <- ggplot(df2, aes(x, y)) + geom_path() + facet_wrap(~k, ncol = 6) +
+     scale_x_continuous("", breaks = NA) + scale_y_continuous("",
+     breaks = NA) + opts(panel.margin = 0, strip.text.x = theme_blank())
> print(pg)
```





12.3 Figure 13.3

```
> library("logspline")
```

lattice

```
> prepanel.ls <- function(x, n = 50, ...) {
+     fit <- logspline(x)
+     xx <- do.breaks(range(x), n)
+     yy <- dlogspline(xx, fit)
+     list(ylim = c(0, max(yy)))
+ }
> panel.ls <- function(x, n = 50, ...) {
+     fit <- logspline(x)
+     xx <- do.breaks(range(x), n)
```

```
+     yy <- dlogspline(xx, fit)
+     panel.lines(xx, yy, ...)
+ }
```

```
> faithful$Eruptions <- equal.count(faithful$eruptions, 4)
```

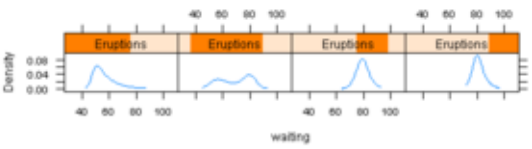
```
> pl <- densityplot(~waiting | Eruptions, data = faithful, prepanel = prepanel.ls,
+     panel = panel.ls)
> print(pl)
```

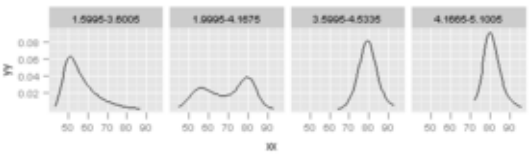
ggplot2

```
> fn <- function(data = faithful$eruptions, number = 4, ...) {
+     intrv <<- as.data.frame(co.intervals(data, number, ...))
+     eruptions <- sort(unique(data))
+     intervals <- ldply(eruptions, function(x) {
+         t(as.numeric(x < intrv$V2 & x > intrv$V1))
+     })
+     tmp <- melt(cbind(eruptions, intervals), id.var = 1)
+     tmp[tmp$value > 0, 1:2]
+ }
> faithful2 <- merge(faithful, fn())
> intrv <- with(intrv, paste(V1, V2, sep = "-"))
> faithful2 <- rename(faithful2, c(variable = "erupt"))
> faithful2$erupt <- factor(faithful2$erupt, labels = intrv)
```

```
> panel.ls.gg <- function(x, n = 50, ...) {
+     fit <- logspline(x)
+     xx <- do.breaks(range(x), n)
+     yy <- dlogspline(xx, fit)
+     data.frame(xx, yy, ...)
+ }
> a <- ddply(faithful2, .(erupt), function(df) {
+     panel.ls.gg(df$waiting)
+ })
```

```
> pg <- ggplot(a, aes(xx, yy)) + geom_line() + facet_grid(~erupt)
> print(pg)
```





12.4 Figure 13.4

```
> data(Chem97, package = "mlmRev")
```

lattice

```
> panel.bwtufte <- function(x, y, coef = 1.5, ...) {
+     x <- as.numeric(x)
+     y <- as.numeric(y)
+     ux <- sort(unique(x))
+     blist <<- tapply(y, factor(x, levels = ux), boxplot.stats,
```

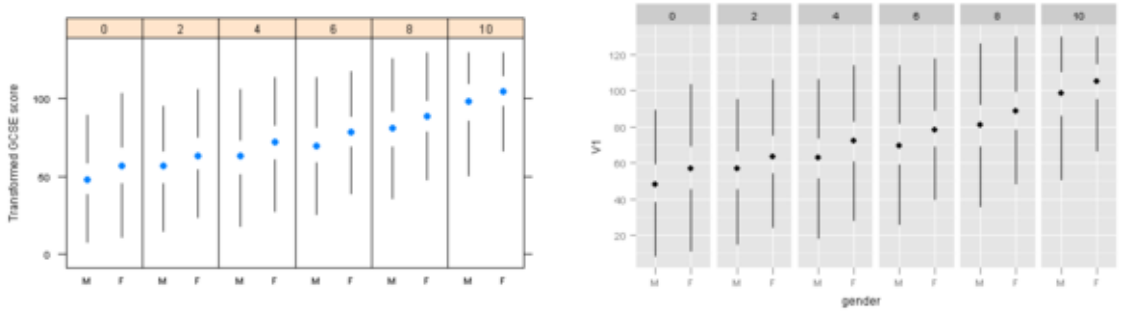
```
+         coef = coef, do.out = FALSE)
+     blist.stats <<- t(sapply(blist, "[[", "stats"))
+     blist.out <<- lapply(blist, "[[", "out")
+     panel.points(y = blist.stats[, 3], x = ux, pch = 16, ...)
+     panel.segments(x0 = rep(ux, 2), y0 = c(blist.stats[, 1],
+         blist.stats[, 5]), x1 = rep(ux, 2), y1 = c(blist.stats[,
+         2], blist.stats[, 4]), ...)
+ }
```

```
> pl <- bwplot(gcsescore^2.34 ~ gender | factor(score), Chem97,
+     panel = panel.bwtufte, layout = c(6, 1), ylab = "Transformed GCSE score")
> print(pl)
```

ggplot2

```
> dt <- ddply(Chem97, .(gender, score), function(df) {
+     boxplot.stats(df$gcsescore^2.34)$stats
+ })
```

```
> pg <- ggplot(dt, aes(x = gender)) + geom_linerange(aes(ymin = V1,
+     ymax = V2)) + geom_linerange(aes(ymin = V4, ymax = V5)) +
+     geom_point(aes(y = V3)) + facet_grid(~score)
> print(pg)
```



12.5 Figure 13.5

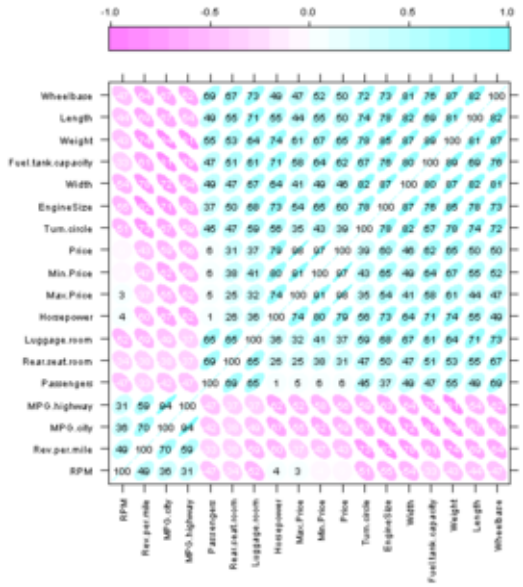
lattice

```
> data(Cars93, package = "MASS")
> cor.Cars93 <- cor(Cars93[, !sapply(Cars93, is.factor)], use = "pair")
> ord <- order.dendrogram(as.dendrogram(hclust(dist(cor.Cars93))))
> panel.corrgram <- function(x, y, z, subscripts, at, level = 0.9,
+     label = FALSE, ...) {
+     require("ellipse", quietly = TRUE)
+     x <- as.numeric(x)[subscripts]
+     y <- as.numeric(y)[subscripts]
+     z <- as.numeric(z)[subscripts]
+     zcol <- level.colors(z, at = at, ...)
+     for (i in seq(along = z)) {
+         ell <- ellipse(z[i], level = level, npoints = 50, scale = c(0.2,
+             0.2), centre = c(x[i], y[i]))
+         panel.polygon(ell, col = zcol[i], border = zcol[i], ...)
+     }
+     if (label)
+         panel.text(x = x, y = y, lab = 100 * round(z, 2), cex = 0.8,
+             col = ifelse(z < 0, "white", "black"))
+ }
```

```
> pl <- levelplot(cor.Cars93[ord, ord], at = do.breaks(c(-1.01,
+     1.01), 20), xlab = NULL, ylab = NULL, colorkey = list(space = "top"),
+     scales = list(x = list(rot = 90)), panel = panel.corrgram,
+     label = TRUE)
> print(pl)
```

ggplot2

```
Ellipses are not supported in ggplot2.
```



12.6 Figure 13.6

lattice

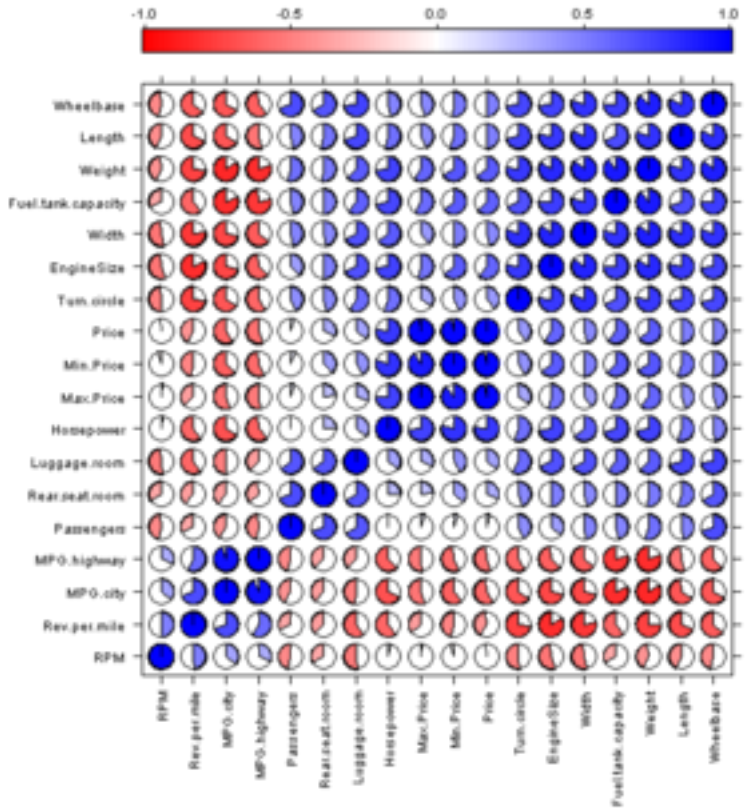
```
> panel.corrgram.2 <- function(x, y, z, subscripts, at = pretty(z),
+     scale = 0.8, ...) {
+     require("grid", quietly = TRUE)
+     x <- as.numeric(x)[subscripts]
+     y <- as.numeric(y)[subscripts]
+     z <- as.numeric(z)[subscripts]
+     zcol <- level.colors(z, at = at, ...)
+     for (i in seq(along = z)) {
+         lims <- range(0, z[i])
+         tval <- 2 * base::pi * seq(from = lims[1], to = lims[2],
+             by = 0.01)
+         grid.polygon(x = x[i] + 0.5 * scale * c(0, sin(tval)),
+             y = y[i] + 0.5 * scale * c(0, cos(tval)), default.units = "native",
+             gp = gpar(fill = zcol[i]))
+         grid.circle(x = x[i], y = y[i], r = 0.5 * scale, default.units = "native")
+     }
+ }
```

```
> pl <- levelplot(cor.Cars93[ord, ord], xlab = NULL, ylab = NULL,
+     at = do.breaks(c(-1.01, 1.01), 101), panel = panel.corrgram.2,
```

```
+     scales = list(x = list(rot = 90)), colorkey = list(space = "top"),
+     col.regions = colorRampPalette(c("red", "white", "blue")))
> print(pl)
```

ggplot2

```
Not supported in +ggplot2+.
```



12.7 Figure 13.7

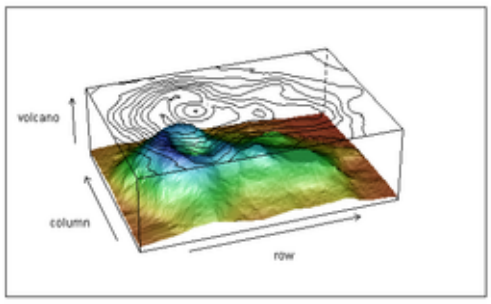
lattice

```
> panel.3d.contour <- function(x, y, z, rot.mat, distance, nlevels = 20,
+     zlim.scaled, ...) {
+     add.line <- trellis.par.get("add.line")
+     panel.3dwire(x, y, z, rot.mat, distance, zlim.scaled = zlim.scaled,
+         ...)
+     clines <- contourLines(x, y, matrix(z, nrow = length(x),
+         byrow = TRUE), nlevels = nlevels)
+     for (ll in clines) {
+         m <- ltransform3dto3d(rbind(ll$x, ll$y, zlim.scaled[2]),
+             rot.mat, distance)
+         panel.lines(m[1, ], m[2, ], col = add.line$col, lty = add.line$lty,
+             lwd = add.line$lwd)
+     }
+ }
```

```
> pl <- wireframe(volcano, zlim = c(90, 250), nlevels = 10, aspect = c(61/87,
+     0.3), panel.aspect = 0.6, panel.3d.wireframe = "panel.3d.contour",
+     shade = TRUE, screen = list(z = 20, x = -60))
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



12.8 Figure 13.8

lattice

```
> library("maps")
> county.map <- map("county", plot = FALSE, fill = TRUE)
```

```
> data(ancestry, package = "latticeExtra")
> ancestry <- subset(ancestry, !duplicated(county))
> rownames(ancestry) <- ancestry$county
> freq <- table(ancestry$top)
> keep <- names(freq)[freq > 10]
> ancestry$mode <- with(ancestry, factor(ifelse(top %in% keep,
+     top, "Other")))
> modal.ancestry <- ancestry[county.map$names, "mode"]
```

```
> library("RColorBrewer")
> colors <- brewer.pal(n = nlevels(ancestry$mode), name = "Pastel1")
```

```
> pl <- xyplot(y ~ x, county.map, aspect = "iso", scales = list(draw = FALSE),
+     xlab = "", ylab = "", par.settings = list(axis.line = list(col = "transparent")),
+     col = colors[modal.ancestry], border = NA, panel = panel.polygon,
+     key = list(text = list(levels(modal.ancestry), adj = 1),
+         rectangles = list(col = colors), x = 1, y = 0, corner = c(1,
+             0)))
> print(pl)
```

ggplot2

```
> counties <- map_data("county")
> counties$reg <- with(counties, paste(region, subregion, sep = ","))
> co_anc <- merge(counties, ancestry, by.x = "reg", by.y = "county")
> co_anc <- co_anc[order(co_anc$order), ]
```

```
> pg <- ggplot(co_anc, aes(long, lat, fill = mode, group = group)) +
+     geom_polygon() + scale_fill_brewer("", palette = "Pastel1")
> print(pg)
```

12.9 Figure 13.9

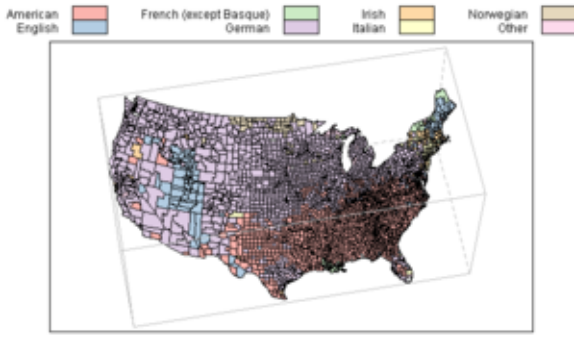
lattice

```
> rad <- function(x) {
+     pi * x/180
+ }
> county.map$xx <- with(county.map, cos(rad(x)) * cos(rad(y)))
> county.map$yy <- with(county.map, sin(rad(x)) * cos(rad(y)))
> county.map$zz <- with(county.map, sin(rad(y)))
> panel.3dpoly <- function(x, y, z, rot.mat = diag(4), distance,
+     ...) {
+     m <- ltransform3dto3d(rbind(x, y, z), rot.mat, distance)
+     panel.polygon(x = m[1, ], y = m[2, ], ...)
+ }
> aspect <- with(county.map, c(diff(range(yy, na.rm = TRUE)), diff(range(zz,
+     na.rm = TRUE)))/diff(range(xx, na.rm = TRUE)))
```

```
> pl <- cloud(zz ~ xx * yy, county.map, par.box = list(col = "grey"),
+     aspect = aspect, panel.aspect = 0.6, lwd = 0.01, panel.3d.cloud = panel.3dpoly,
+     col = colors[modal.ancestry], screen = list(z = 10, x = -30),
+     key = list(text = list(levels(modal.ancestry), adj = 1),
+         rectangles = list(col = colors), space = "top", columns = 4),
+     scales = list(draw = FALSE), zoom = 1.1, xlab = "", ylab = "",
+     zlab = "")
> print(pl)
```

ggplot2

```
ggplot2 currently does not support true 3d surfaces.
```



12.10 Figure 13.10

```
> library("latticeExtra")
> library("mapproj")
> data(USCancerRates)
> rng <- with(USCancerRates, range(rate.male, rate.female, finite = TRUE))
> nbreaks <- 50
> breaks <- exp(do.breaks(log(rng), nbreaks))
> breaks2 <- c(unique(breaks[1 + (0:(nbreaks - 1)%/%10) * 10]),
+     max(breaks) - 0.1)
```

lattice

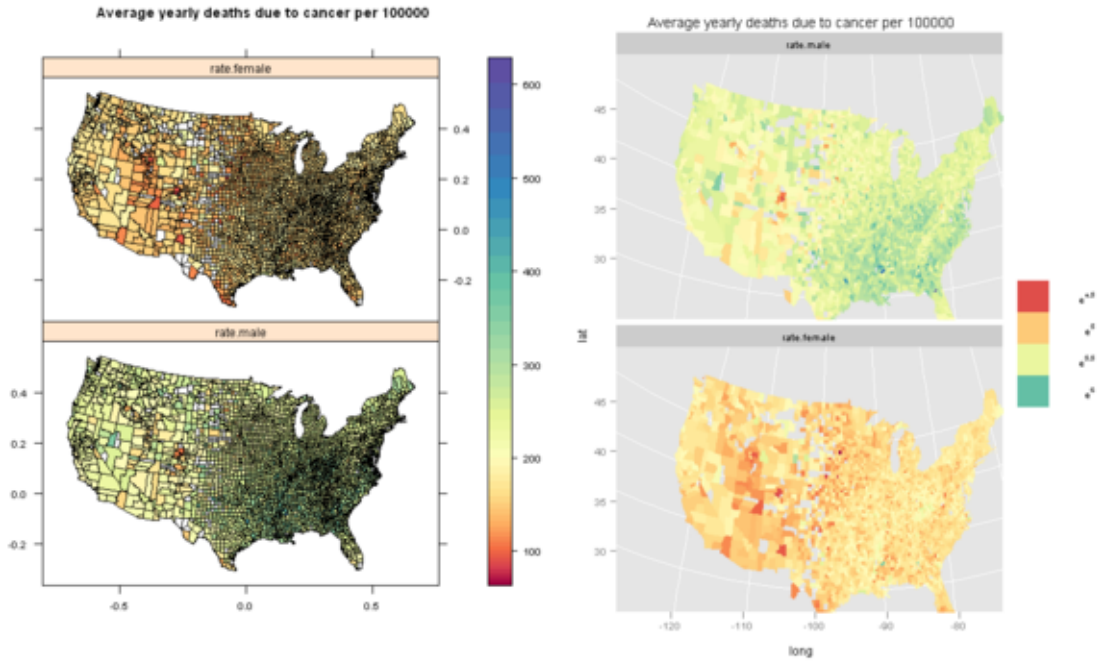
```
> pl <- mapplot(rownames(USCancerRates) ~ rate.male + rate.female,
+     data = USCancerRates, breaks = breaks, map = map("county",
+         plot = FALSE, fill = TRUE, projection = "tetra"), scales = list(draw = T),
+     xlab = "", main = "Average yearly deaths due to cancer per 100000")
> print(pl)
```

ggplot2

```
> USCancerRates.df <- namerows(USCancerRates, col.name = "reg")
> co_cancer <- merge(counties, USCancerRates.df, by = c("reg"))
> co_cancer <- co_cancer[order(co_cancer$order), ]
> co_cancer.m <- melt(co_cancer, measure.vars = c("rate.male",
+     "rate.female"), na.rm = TRUE)
> co_cancer.m$fill <- with(co_cancer.m, as.numeric(as.character(cut(value,
+     breaks, labels = comma(breaks[-1])))))
```

```
> brewer.div <- colorRampPalette(brewer.pal(11, "Spectral"))
```

```
> pg <- ggplot(co_cancer.m, aes(long, lat, group = reg, fill = fill)) +
+     geom_polygon() + coord_map(projection = "tetra") + facet_wrap(~variable,
+     ncol = 1) + scale_fill_gradientn("", colours = brewer.div(nbreaks),
+     trans = "log") + opts(title = "Average yearly deaths due to cancer per 100000")
> print(pg)
```



Chapter 13

New Trellis Displays

TOPICS COVERED:

- Examples of S3 and S4 methods
- Examples of new high level functions

13.1 Figure 14.1

```
> library(lattice)
> library(ggplot2)
> library(latticeExtra)
```

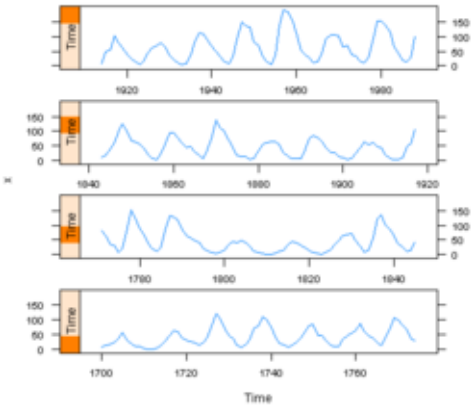
lattice

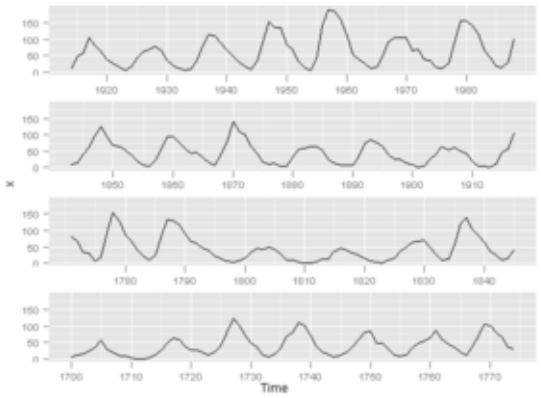
```
> pl <- xyplot(sunspot.year, aspect = "xy", strip = FALSE, strip.left = TRUE,
+     cut = list(number = 4, overlap = 0.05))
> print(pl)
```

ggplot2

```
> sunspot.g <- function(data, number = 4, overlap = 0.05) {
+     data <- as.data.frame(data)
+     data$id <- if (is.ts(data$x))
+         time(data$x)
+     else seq_along(data$x)
+     intrv <- as.data.frame(co.intervals(data$id, number, overlap))
+     x <- sort(unique(data$id))
+     intervals <- ldply(x, function(x) {
+         t(as.numeric(x < intrv$V2 & x > intrv$V1))
+     })
+     tmp <- melt(cbind(x, intervals), id.var = 1)
+     tmp <- tmp[tmp$value > 0, 1:2]
+     tmp <- rename(tmp, c(x = "id"))
+     merge(data, tmp)
+ }
```

```
> pg <- ggplot(sunspot.g(sunspot.year), aes(id, x)) + geom_line() +
+     facet_wrap(~variable, scales = "free_x", ncol = 1, as.table = FALSE) +
+     opts(strip.background = theme_blank(), strip.text.x = theme_blank()) +
+     opts(panel.margin = unit(-0.25, "lines")) + xlab("Time")
> print(pg)
```





13.2 Figure 14.2

```
> data(biocAccess, package = "latticeExtra")
```

lattice

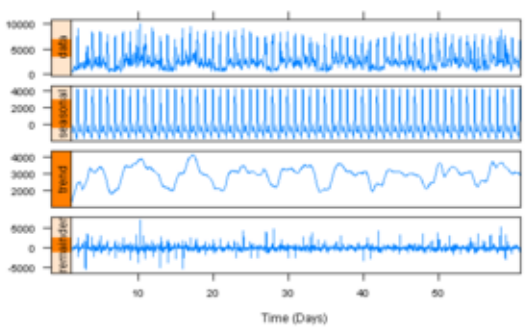
```
> ssd <- stl(ts(biocAccess$counts[1:(24 * 30 * 2)], frequency = 24),
+     "periodic")
```

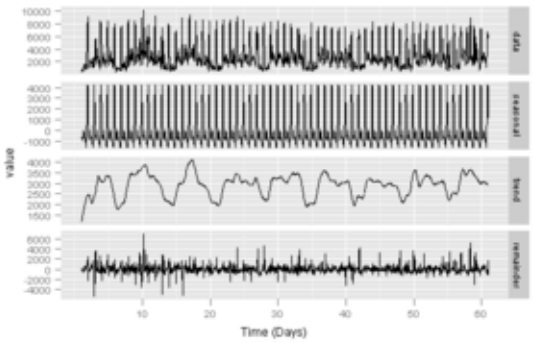
```
> pl <- xyplot(ssd, xlab = "Time (Days)")
> print(pl)
```

ggplot2

```
> time <- data.frame(data = ts(biocAccess$counts[1:(24 * 30 * 2)],
+     frequency = 24))
> time$id <- as.numeric(time(time$data))
> time$data <- as.numeric(time$data)
> time.series <- as.data.frame(ssd$time.series)
> time.series <- cbind(time, time.series)
> time.series <- melt(time.series, id.vars = "id")
```

```
> pg <- ggplot(time.series, aes(id, value)) + geom_line() + facet_grid(variable ~
+     ., scales = "free_y") + xlab("Time (Days)")
> print(pg)
```





13.3 Figure 14.3

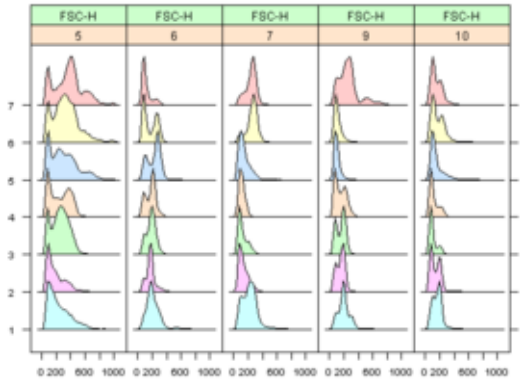
```
> library("flowViz")
> data(GvHD, package = "flowCore")
```

lattice

```
> pl <- densityplot(Visit ~ `FSC-H` | Patient, data = GvHD)
> print(pl)
```

ggplot2

```
It should be possible to produce a similar graph in ggplot2, however I was not able to ↩
    figure out how to extract the relevant data from an object of class "flowSet".
```



13.4 Figure 14.4

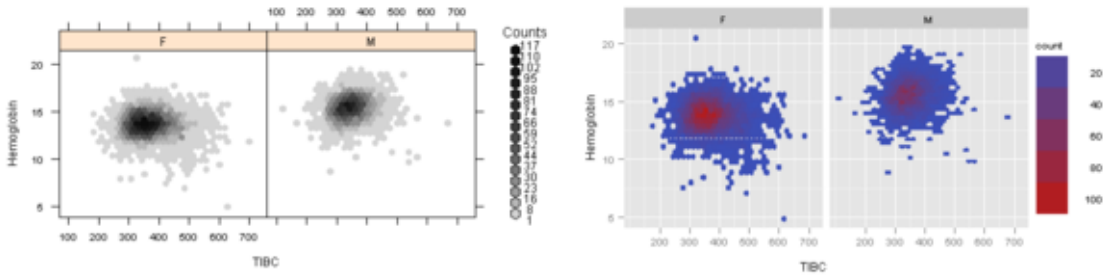
```
> library("hexbin")
> data(NHANES)
```

lattice

```
> pl <- hexbinplot(Hemoglobin ~ TIBC | Sex, data = NHANES, aspect = 0.8)
> print(pl)
```

ggplot2

```
> pg <- ggplot(NHANES, aes(TIBC, Hemoglobin)) + geom_hex() + facet_grid(~Sex) +
+     opts(aspect.ratio = 0.8)
> print(pg)
```



13.5 Figure 14.5

```
> data(Chem97, package = "mlmRev")
```

lattice

```
> panel.piechart <- function(x, y, labels = as.character(y), edges = 200,
+     radius = 0.8, clockwise = FALSE, init.angle = if (clockwise) 90 else 0,
+     density = NULL, angle = 45, col = superpose.polygon$col,
+     border = superpose.polygon$border, lty = superpose.polygon$lty,
+     ...) {
+     stopifnot(require("gridBase"))
+     superpose.polygon <- trellis.par.get("superpose.polygon")
+     opar <- par(no.readonly = TRUE)
+     on.exit(par(opar))
+     if (panel.number() > 1)
+         par(new = TRUE)
+     par(fig = gridFIG(), omi = c(0, 0, 0, 0), mai = c(0, 0, 0,
+         0))
+     pie(as.numeric(x), labels = labels, edges = edges, radius = radius,
+         clockwise = clockwise, init.angle = init.angle, angle = angle,
+         density = density, col = col, border = border, lty = lty)
+ }
> piechart <- function(x, data = NULL, panel = "panel.piechart",
+     ...) {
+     ocall <- sys.call(sys.parent())
+     ocall[[1]] <- quote(piechart)
+     ccall <- match.call()
+     ccall$data <- data
+     ccall$panel <- panel
+     ccall$default.scales <- list(draw = FALSE)
+     ccall[[1]] <- quote(lattice::barchart)
+     ans <- eval.parent(ccall)
+     ans$call <- ocall
+     ans
+ }
> pl <- piechart(VADeaths, groups = FALSE, xlab = "")
> print(pl)
```

ggplot2

```
> pg <- ggplot(as.data.frame.table(VADeaths), aes(x = factor(1),
+     y = Freq, fill = Var1)) + geom_bar(width = 1) + facet_wrap(~Var2,
+     scales = "free_y") + coord_polar(theta = "y")
> print(pg)
```