Statistics 120
Multipanel Conditioning II

Trellis Graphics

- The Trellis graphics system in R is written by Deepayan Sarkar of the University of Wisconsin, using the “Grid” graphics system written by Paul Murrell of Auckland.
- The system is a reimplementation of the the original Bell Labs Trellis system created by Bill Cleveland and Rick Becker.
- These class notes should show you all you need to know about producing simple Trellis displays.
- More extensive documentation is available on the class web site.

Using Trellis Graphics in R

- The trellis graphics system exists in parallel with the normal R graphics system.
- You cannot mix commands from the two systems, but Trellis provides equivalents to most of the normal graphics system commands.
- In order to produce Trellis plots you must load the "Lattice" library.

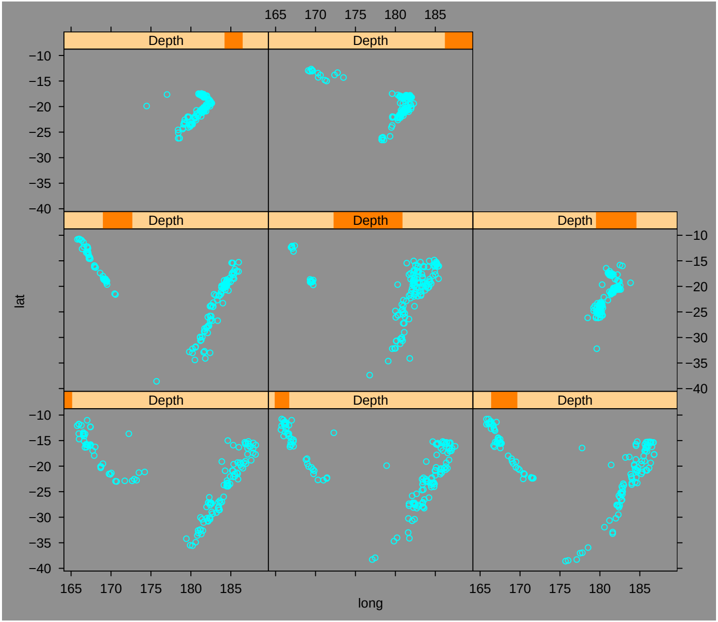
```
> library(lattice)
```

A Trellis Example

- This example shows a series of scatter plots of earthquake foci corresponding to various depth intervals.
- This is a minimal Trellis plot example, with no customisation.

```
> Depth = equal.count(quakes$depth,
                      number = 8,
                      overlap = .1)

> xyplot(lat ~ long | Depth, data = quakes)
```

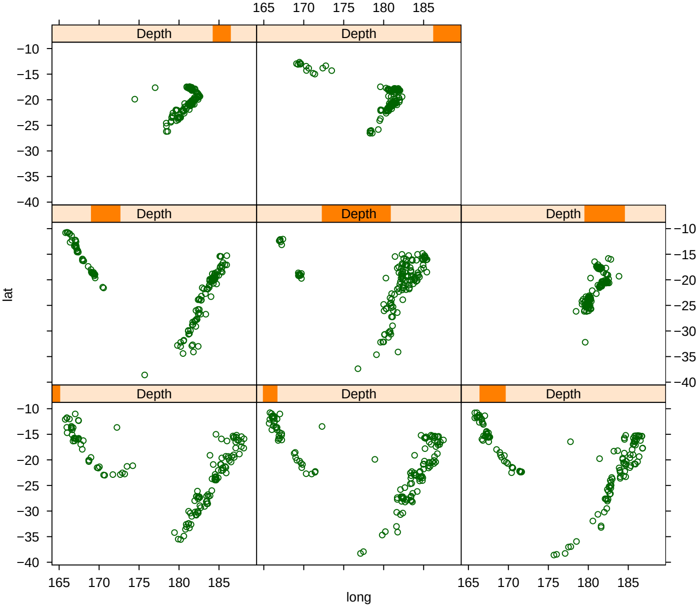

Depth
lat
long
165 170 175 180 185
−10 −15 −20 −25 −30 −35 −40

Choice of Colour Scheme

- The default colour scheme used by Trellis uses light colours on a medium-gray background.
- This is a bad choice of colour scheme because there is less contrast between foreground colours and the background than there might be.
- It is a good idea to use an alternative colour scheme which uses a dark colours on a white background.

```
> lset(col.whitebg())

> xyplot(lat ~ long | Depth, data = quakes)
```

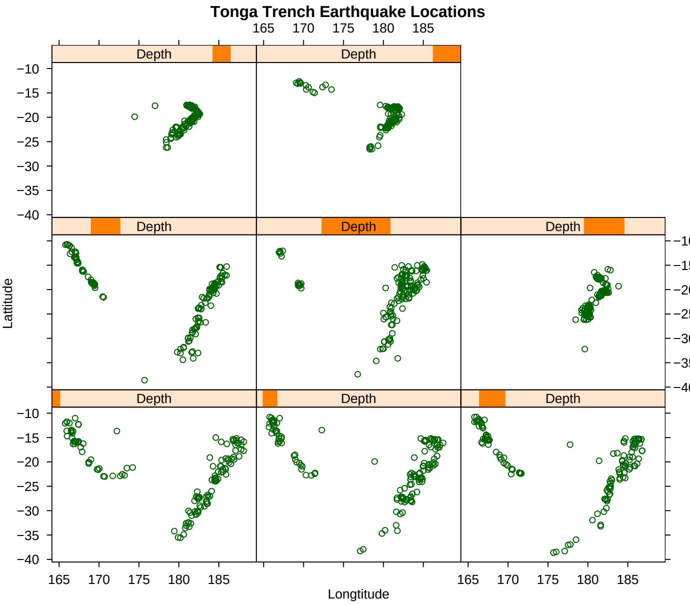


●First ●Prev ●Next ●Last ●Go Back ●Full Screen ●Close ●Quit

Titles and Axis Annotation

- As with all graphics it is possible to add a title and axis annotation using `main=`, `lab=` and `ylab=` arguments.

```
> xyplot(lat ~ long | Depth, data = quakes,
     main = "Tonga Trench Earthquake Locations",
     xlab = "Longtitude",
     ylab = "Lattitude")
```


Tonga Trench Earthquake Locations
Depth
Lattitude
Longtitude
165
170
175
180
185
−10
−15
−20
−25
−30
−35
−40

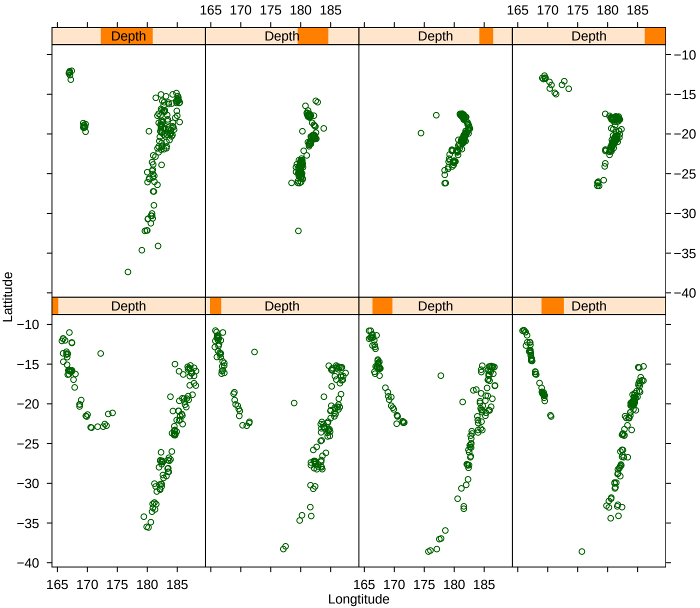
Layout Control

- By default, Trellis usually chooses a good plot layout, but sometimes it is useful to override the choice using the `layout` argument.

- The `layout` argument should be a vector of three values giving the number of rows, number of columns and number of pages desired for the display.

- For example, we can rearrange the earthquake plot as follows:

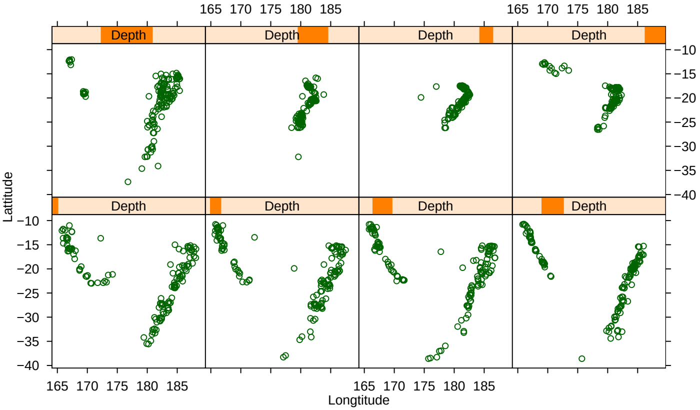
```
> xyplot(lat ~ long | Depth, data = quakes,
     layout = c(4, 2, 1),
     xlab = "Longtitude",
     ylab = "Lattitude")
```


Depth
Lattitude
Longtitude
165 170 175 180 185
−10 −15 −20 −25 −30 −35 −40

Aspect Ratio Control

- The panels in the previous plot are rather too tall relative to their widths.
- By default, plots are sized so that they they occupy the full surface of the output window.
- This can changed by specifying the aspect ratio for the plots.

```
> xyplot(lat ~ long | Depth, data = quakes,
      aspect = 1,
      layout = c(4, 2, 1),
      xlab = "Longtitude",
      ylab = "Lattitude")
```


Depth
Depth
Depth
Depth
Depth
Depth
Depth
Depth
Lattitude
Longtitude
165 170 175 180 185
−10 −15 −20 −25 −30 −35 −40

Trellis Examples

- For the rest of the lecture we will look at a variety of examples of Trellis plots.
- This is really just scratching the surface of what can be done with trellis.

Death Rates by Gender and Location

- In this example we'll look at the Virginia death rate data.
- The data values are death rates per 1000 of population cross-classified by age and population group.
- We are intested in how death rate changes with age and how the death rates in the different population groups compare.

Data Manipulation

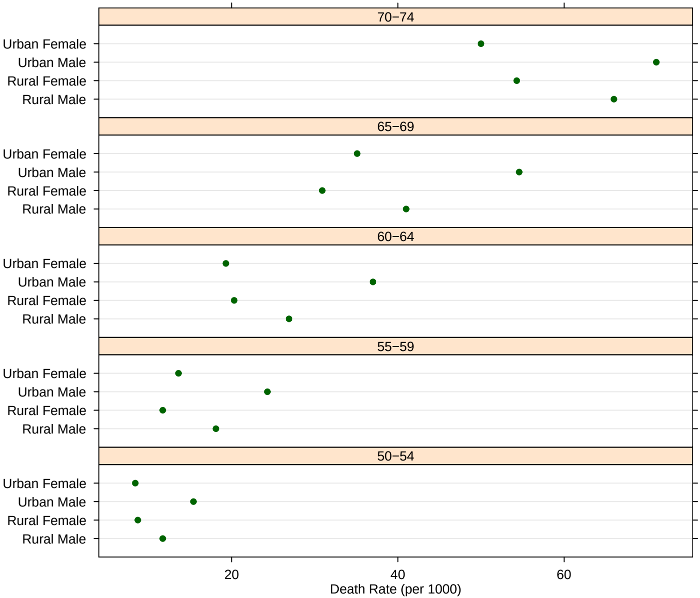
- The data values are stored by R as a matrix.
- We first have to turn the death rates into a vector and create the cross-classifying factors.

```
> data(VADeaths)
> rate = as.vector(VADeaths)
> age = row(VADeaths, as.factor=TRUE)
> group = col(VADeaths, as.factor=TRUE)
```

Display 1

- We start by displaying deaths against age, conditional on population group.
- The command below uses `layout` to force the panels to be stacked above each other to make comparisons easy.

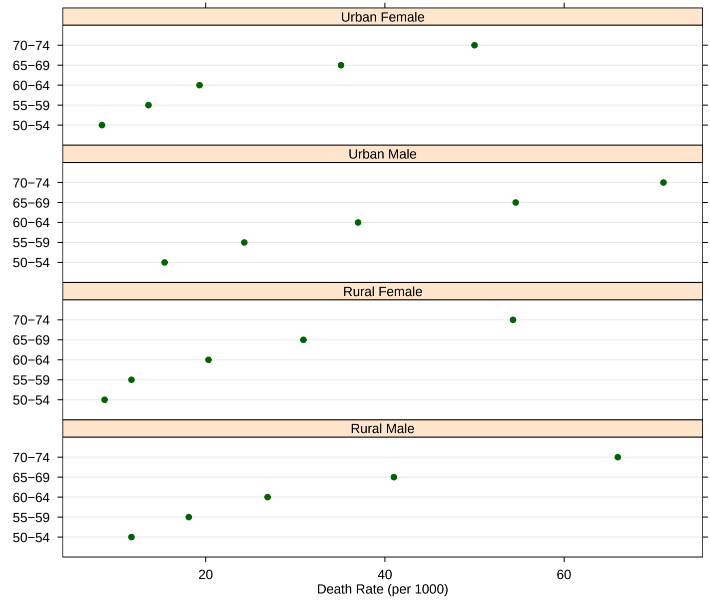
```
> dotplot(group ~ rate | age,
          xlab = "Death Rate (per 1000)",
          layout = c(1, 5, 1))
```


70−74
65−69
60−64
55−59
50−54
Urban Female
Urban Male
Rural Female
Rural Male
20
40
60
Death Rate (per 1000)

Display 2

- The first display is hard to read because the variantion within each age group is “noisy.”
- We could try to get around this by ordering the population categories differently.
- Alternatively we can interchange the roles of the cross-classifying variables.

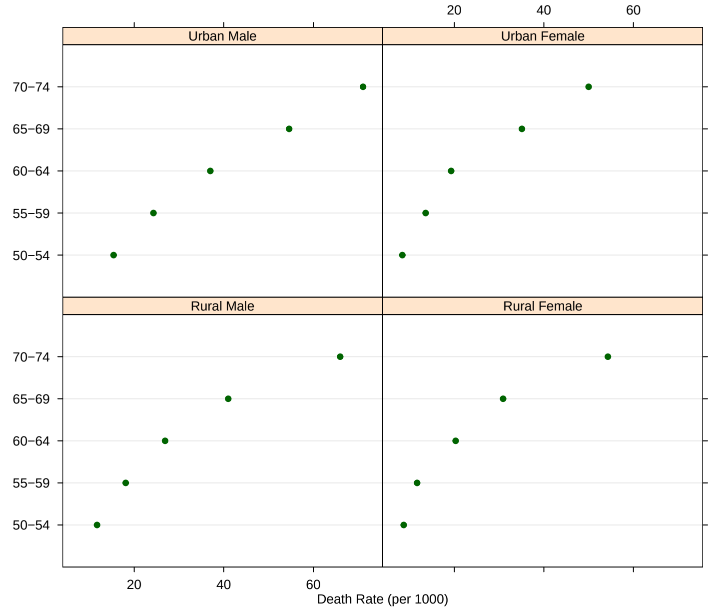
```
> dotplot(age ~ rate | group,
          xlab = "Death Rate (per 1000)",
          layout = c(1, 4, 1))
```


Urban Female
Urban Male
Rural Female
Rural Male
70−74
65−69
60−64
55−59
50−54
20
40
60
Death Rate (per 1000)

Display 3

- The second display is better than the first, but can improve it with a different ordering of the panels.
- We'll arange the panels in a 2×2 array.
- This will allow us to make direct male/female and urban/rural comparisons.

```
> dotplot(age ~ rate | group,
          xlab = "Death Rate (per 1000)",
          layout = c(2, 2, 1))
```

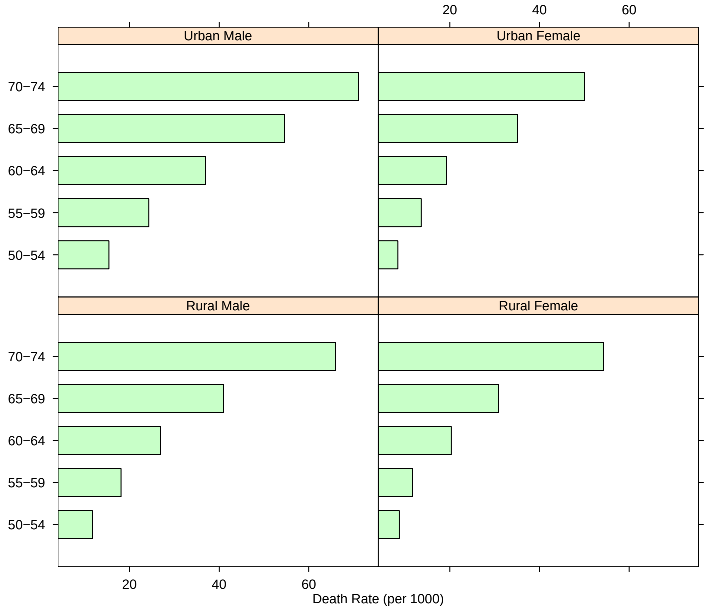


Urban Male
Urban Female
Rural Male
Rural Female
70−74
65−69
60−64
55−59
50−54
20
40
60
Death Rate (per 1000)

Display 4

- The previous displays presented the data in “dotchart” displays.
- There are other alternatives, barcharts for example.

```
> barchart(age ~ rate | group,
           xlab = "Death Rate (per 1000)",
           layout = c(2, 2, 1))
```



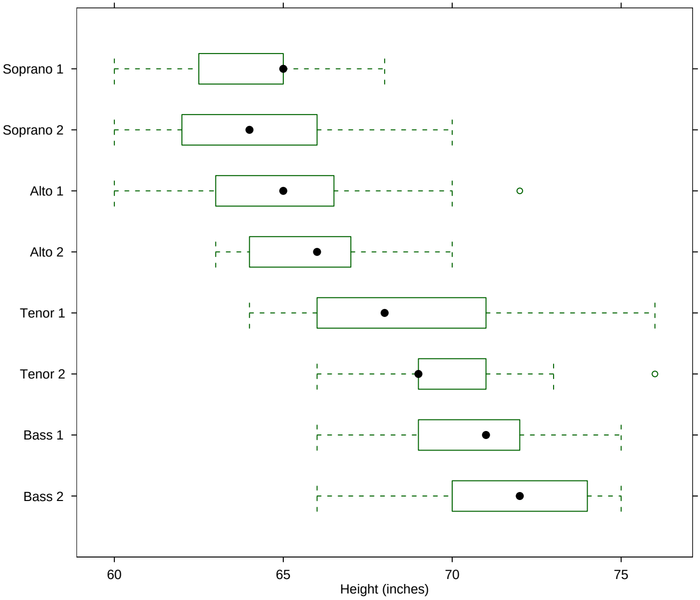
Urban Male
Urban Female
Rural Male
Rural Female
70−74
65−69
60−64
55−59
50−54
20
40
60
Death Rate (per 1000)

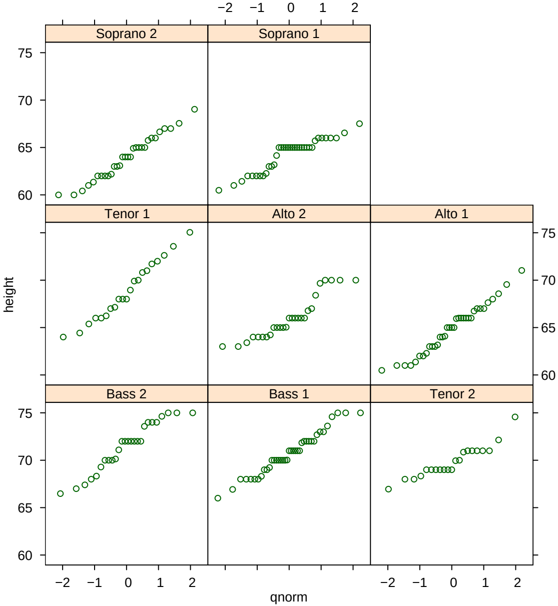
Heights of Singers

- In this example we'll examine the heights of the members of a large choral society.
- The values are in a data set called `singer` which is in the Lattice data library. They can be loaded with the `data` command once the Lattice library is loaded.
- The variables are named `height` (inches) and `voice.part`.

```
> bwplot(voice.part ~ height, data=singer,
         xlab="Height (inches)")

> qqmath(~ height | voice.part,
         aspect = 1, data = singer)
```


Soprano 1
Soprano 2
Alto 1
Alto 2
Tenor 1
Tenor 2
Bass 1
Bass 2
60
65
70
75
Height (inches)


Soprano 2
Soprano 1
Tenor 1
Alto 2
Alto 1
Bass 2
Bass 1
Tenor 2
height
qnorm
−2
−1
0
1
2
60
65
70
75

Measurement of Exhaust from Burning Ethanol

- The ethanol data frame records 88 measurements (rows) for three variables (columns) NOx , C , and E from an experiment in which ethanol was burned in a single cylinder automobile test engine.

- NOx gives the concentration of nitric oxide (NO) and nitrogen dioxide (NO2) in engine exhaust, normalised by the work done by the engine.

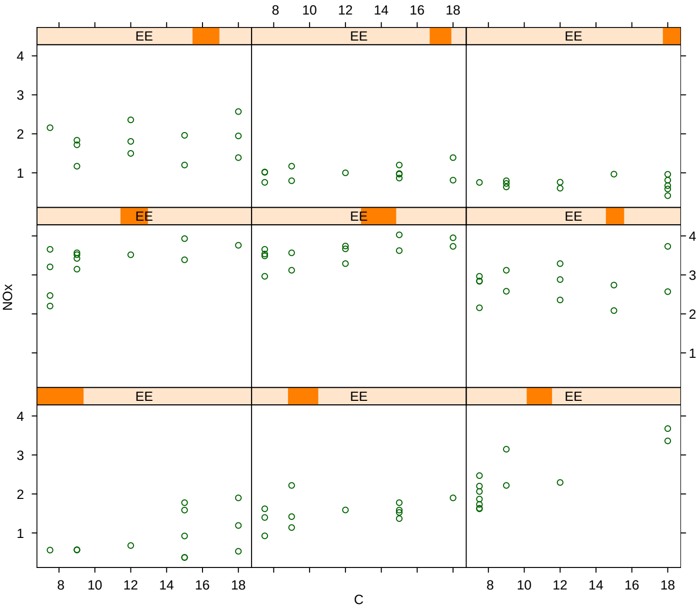
- C gives the compression ratio of the engine.

- E gives the equivalence ratio at which the engine was run – a measure of the richness of the air/ethanol mix.

Exploring the Relationship

- We can get a basic idea of the form of the relatioship between the variables using a simple conditioning plot.

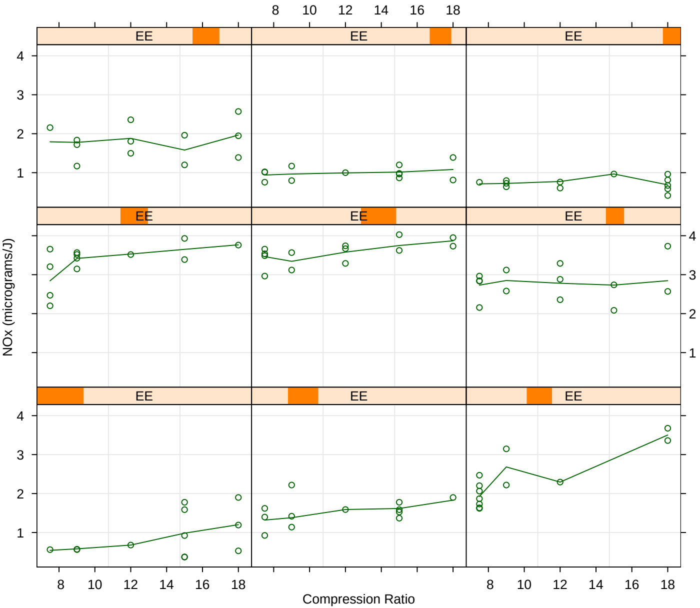
```
> data(ethanol)
> EE = equal.count(ethanol$E, number=9,
                   overlap=1/4)
> xyplot(NOx ~ C | EE, data = ethanol)
```


EE
EE
EE
EE
EE
EE
EE
EE
EE
8 10 12 14 16 18
1 2 3 4
NOx
C

A More Complex Plot

- We can enhance the previous plot by adding a smooth line through the points in each panel.
- This is done using the lowess smoother.

```
> xyplot(NOx ~ C | EE, data = ethanol,
         xlab = "Compression Ratio",
         ylab = "NOx (micrograms/J)",
         panel = function(x, y) {
             panel.grid(h=-1, v= 2)
             panel.xyplot(x, y)
             llines(lowess(x,y))
         })
```



EE
NOx (micrograms/J)
Compression Ratio
8
10
12
14
16
18
1
2
3
4