Trellis Graphics

- Trellis Graphics is a family of techniques for viewing complex, multi-variable data sets.
- The ideas have been around for a while, but were formalized by researchers at Bell Laboratories during the 1990s.
- The techniques were given the name *Trellis* because they usually result in a rectangular array of plots, resembling a garden trellis.
- A number of statistical software systems provide multi-panel conditioning plots under the name *Trellis* plots or *Crossplots*.

Trellis Graphics in R

- The Trellis graphics system in R was 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 and start a “trellis aware” device.

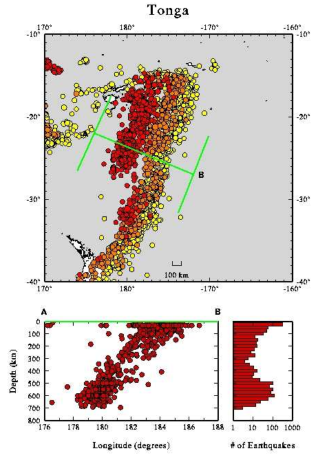
```
> library(lattice)
> trellis.device()
```

Conditioning

- Trellis plots are based on the idea of *conditioning* on the values taken on by one or more of the variables in a data set.
- In the case of a categorical variable, this means carrying out the same plot for the data subsets corresponding to each of the levels of that variable.
- In the case of a numeric variable, it means carrying out the same plots data subsets corresponding to intervals of that variable.

Example: Earthquake Locations

- R contains a data set called `quakes` which gives the location and magitude of earthquakes under the Tonga Trench, to the North of New Zealand.
- The spatial distribution of earthquakes in the area is of major interest, because this enables us to “see” the structure of the earthquake faults.
- Here is a plot from the Geology department at Berkeley, which tries to present the the spatial structure.



Tonga Trench Earthquakes

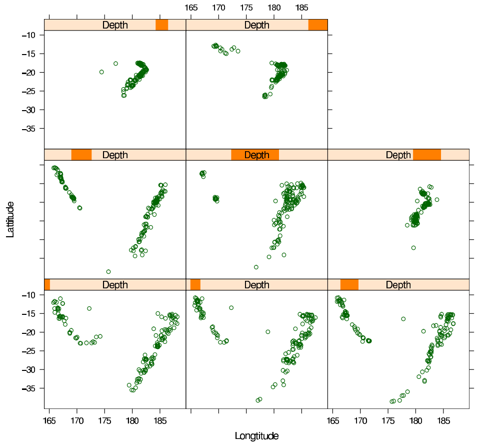
Yellow: 0 — 70 km
Orange: 71 — 300 km
Red: 300 — 800 km.

Problems with this Presentation

- There is a good deal of overplotting and this makes it hard to see all of the structure present in the data.
- The map makes it clear that we are looking down from above on the scene, but deeper quakes appear to be plotted on top of shallower ones.
- The division of depths into three intervals and presentation using colour is relatively crude.

A Trellis Plot

- We can overcome many of the problems of the previous plot by using a trellis display.
- We create the display by producing a sequence of graphs, each of which presents a different range of depths.
- In this case we will have a slight overlap of the intervals being plotted.



Explanation

- The plot is read left-to-right and bottom-to-top.
- Depth increases progressively through the plot.
- There are eight different depth intervals, each containing approximately the same number of earthquakes.
- Consecutive depth intervals overlap by a small amount.
- The range of depths covered by each interval is indicated in the bar above each plot.

Intepretation

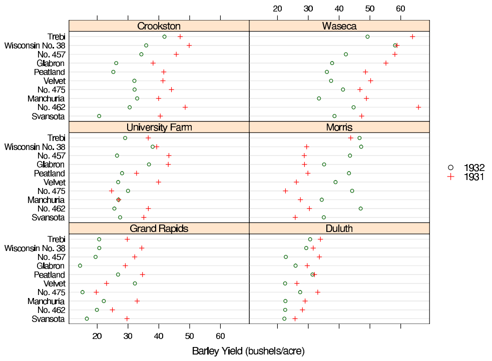
- The shallower earthquakes are concentrated on two inclined fault planes.
- The most easterly of these fault planes is the one which bisects New Zealand.
- The Westerly fault plane has mainly shallow earthquakes, while the Easterly fault plane has both shallow and deep earthquakes.
- The deep earthquakes show distinct small angular fishhook structure which is not visible in the earlier plot.

Example: Barley Yields

- This example is concerned with the yields obtained from field trials of barley seed.
- The data comes from the 1930s so there is no direct genetic modification going here.
- The trials were conducted in 1931 and 1932, using:
  - 10 different strains of barley
  - 6 different growing sites
- There are $2 \times 10 \times 6 = 120$ observations.
- It was suspected for a long time that there was something odd about this data set.

The Trellis Plot

- The plot we will look at shows that barley yields for each of the 10 strains at the 6 sites and for each year.
- The results for each site are plotted on a separate graph – i.e. we are working conditional on the site.
- The yields from the two years are superimposed on each of the plots.



The Trellis Technology

- There are a variety of displays which can be produced by Trellis, including:
  - Bar Charts
  - Dot Charts
  - Box and Whisker Plots
  - Histograms
  - Density Traces
  - QQ Plots
  - Scatter Plots
- A common framework is used to produce all these plots.

Some Terminology

- Every Trellis display consists of a series of rectangular *panels*, laid out in a regular row-by-column array.
- The indexing of the array is left-to-right, bottom-to-top.
- The x axes of all the panels are identical. This is also true for the y axes.
- Each panel of the a display corresponds to conditioning, either on the levels of a factor, or on sub-intervals of the range of a numeric variable.

Shingles

- The conditioning carried out in the earthquake plot is described by a *shingle*.
- A shingle consists of a number of overlapping intervals (like the shingles on a roof of a house).
- Assuming that the earthquake depths are contained in the variable `depth`, the shingle is created as follows.

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

- The shingle assigned to `Depth` has 8 intervals with adjacent intervals having 10% of their values in common.

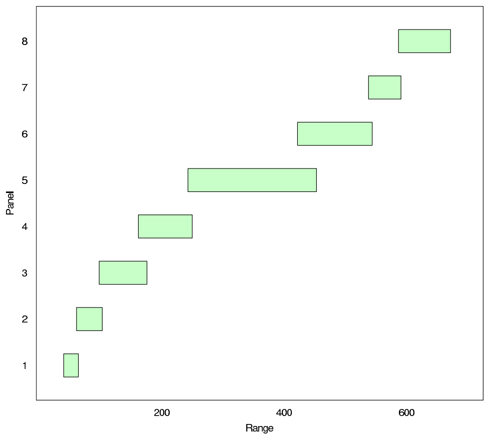
Shingles

- A shingle contains the numerical values it was created from and can be treated like a copy of that variable. For example:

```
> range(Depth)
[1]  40 680
> range(depth)
[1]  40 680
```

- A shingle also has the information attached to it. This can be displayed by printing or plotting the shingle.

```
> plot(Depth)
```

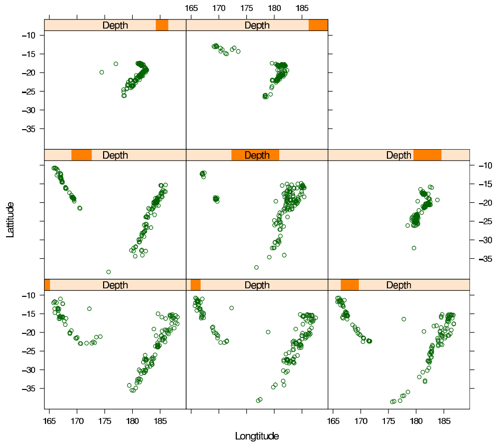


Producing the Plot

- The display of the earthquakes is produced by the function `xyplot`, which is the Trellis variant of a scatter plot function.
- The plot was produced as follows:

```
> Depth = equal.count(quakes$depth,
                      number = 8,
                      overlap = .1)
> xyplot(lat ~ long | Depth, data = quakes,
         xlab = "Longtitude",
         ylab = "Lattitude")
```

- There are two steps here (i) creating the shingle and (ii) producing the display.



The Plot Formula

- The first argument to `xyplot` is a symbolic formula describing the plot.
- In this case the formula is:

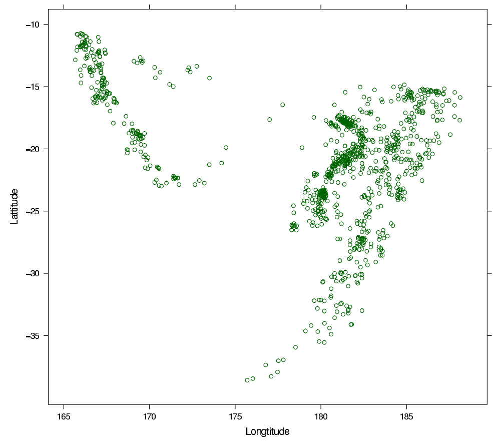
```
    lat ~ long | Depth
```

  which is an instruction to plot `lat` on the y axis against `long` on the x axis with conditioning intervals as described in `Depth`.
- The second argument to `xyplot` specifies which data frame the data for the plot should be obtained from.
- Additional arguments control other aspects of the plot.

Unconditional Plots

- The `xyplot` function can be used to produced an unconditional plot by omitting the conditioning specification from the plot formula.

```
> xyplot(lat ~ long, data = quakes,
         xlab = "Longtitude",
         ylab = "Lattitude")
```



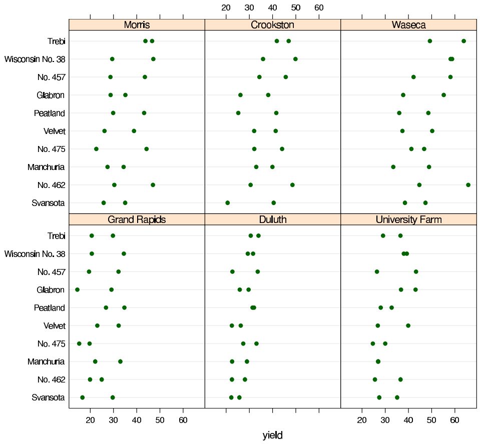
The Barley Yield Plot

- The barley yield plot is produced by the function `dotchart` which can be used to numeric values against a categorical variable.
- In this case, the numeric variable is the barley yield and the categorical variable is the seed strain.
- We also condition on the value of another variable, the growing site.

A First Attempt

- The following code is a simple attempt at creating a dot chart using similar code to that for the earthquakes.

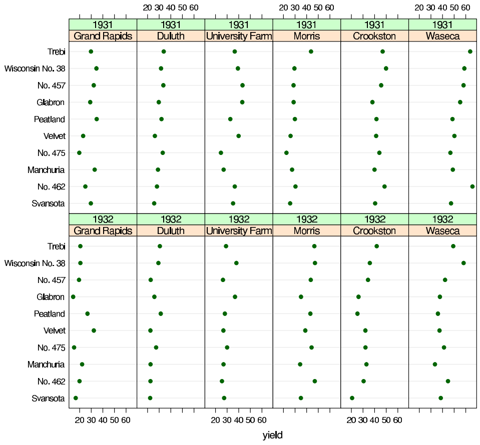
```
> dotplot(variety ~ yield | site,
          data = barley)
```



A Second Attempt

- We could also try conditioning on both site and year.

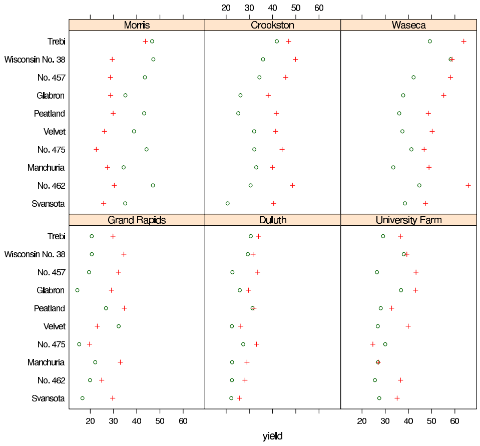
```
> dotplot(variety ~ yield | site * year,
          data = barley)
```



A Third Attempt

- What we need is to superimpose the two years for each site on a single panel.

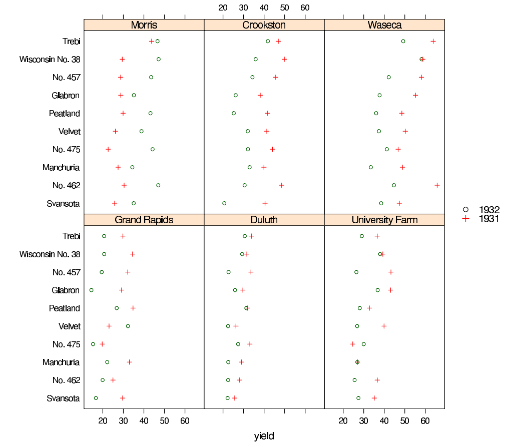
```
> dotplot(variety ~ yield | site,
          data = barley,
          panel = panel.superpose,
          group = year,
          pch = c(1, 3))
```



A Fourth Attempt

- The last plot is quite close.
- We need to add a legend which indicates which year is which.

```
> dotplot(variety ~ yield | site,
          data = barley,
          panel = panel.superpose,
          group = year, pch = c(1, 3),
          key = list(space = "right",
            transparent = TRUE,
            points = list(pch = c(1, 3),
              col = 1:2),
            text = list(c("1932", "1931"))))
```

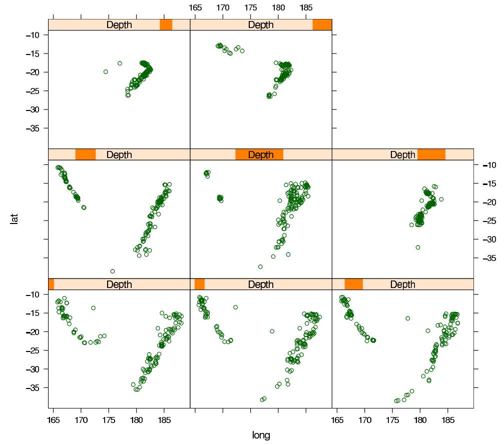


Choice of Colour Scheme

- 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.

```
> trellis.par.set(theme = col.whitebg())
```

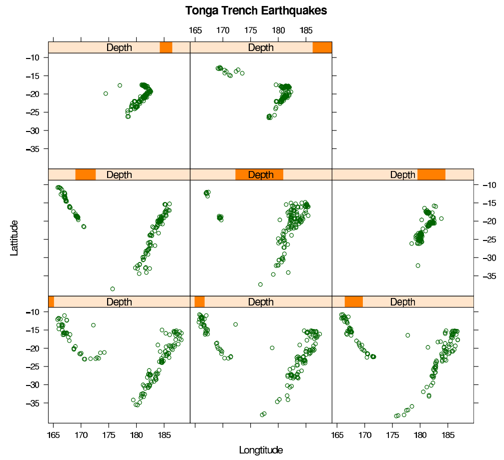
```
> xyplot(lat ~ long | Depth, data = quakes)
```



Titles and Axis Annotation

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

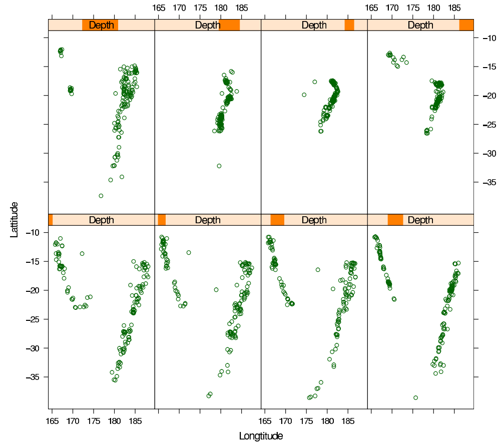
```
> xyplot(lat ~ long | Depth, data = quakes,
         main = "Tonga Trench Earthquakes",
         xlab = "Longtitude",
         ylab = "Lattitude")
```



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:

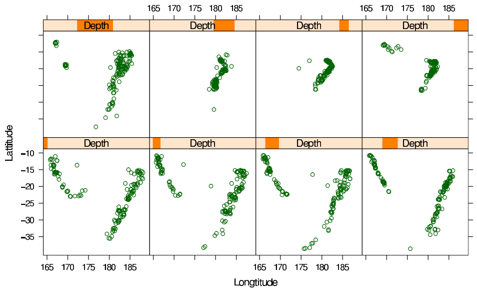
```
> xyplot(lat ~ long | Depth, data = quakes,
         layout = c(4, 2, 1),
         xlab = "Longtitude",
         ylab = "Lattitude")
```



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")
```



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

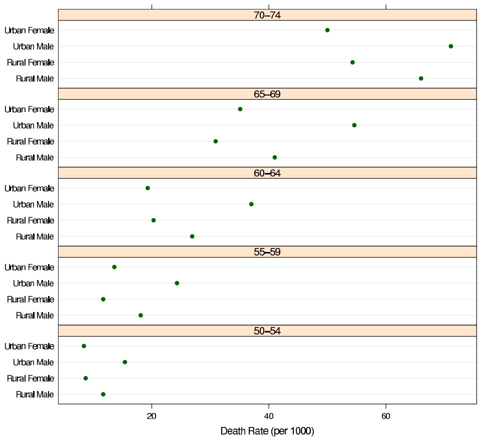
- The data values are stored by R as a matrix.
- We first have to turn the death rates into a vector and create the cross-classifying factors.

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

Dotchart 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.

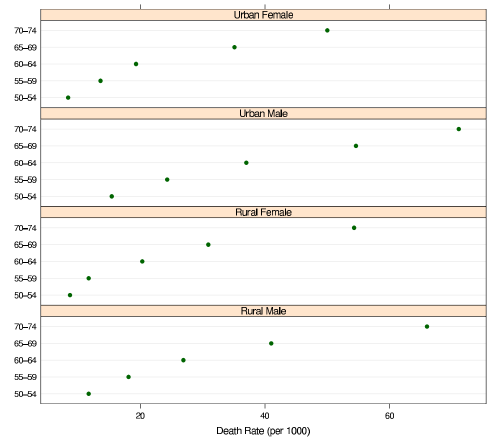
```
> dotplot(group ~ rate | age,
          xlab = "Death Rate (per 1000)",
          layout = c(1, 5, 1))
```



Dotchart 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.

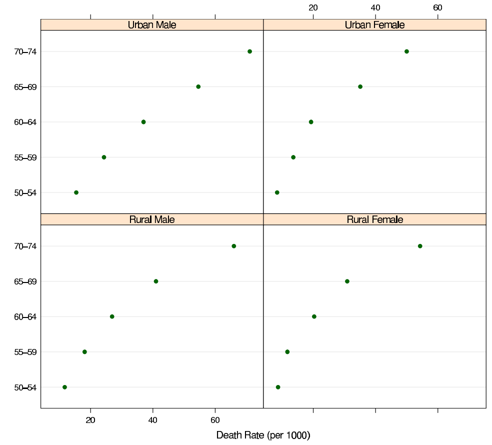
```
> dotplot(age ~ rate | group,
          xlab = "Death Rate (per 1000)",
          layout = c(1, 4, 1))
```



Dotchart 3

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

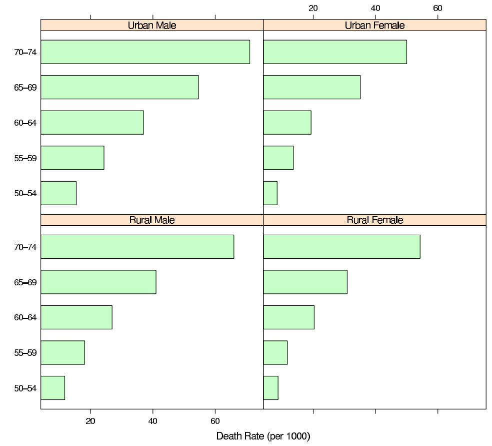
```
> dotplot(age ~ rate | group,
          xlab = "Death Rate (per 1000)",
          layout = c(2, 2, 1))
```



Alternative Displays

- 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))
```

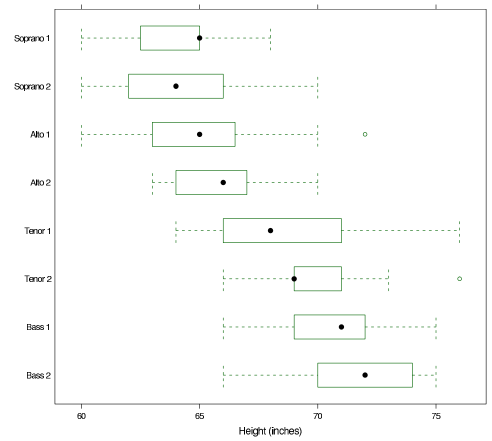


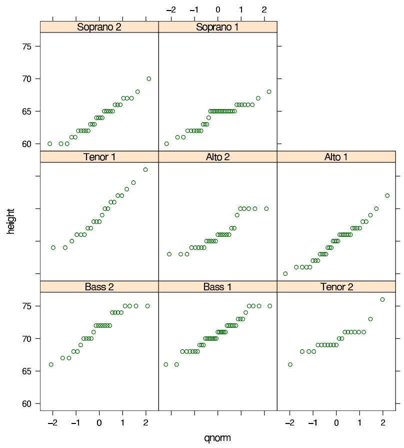
Heights of Singers

- 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)
```





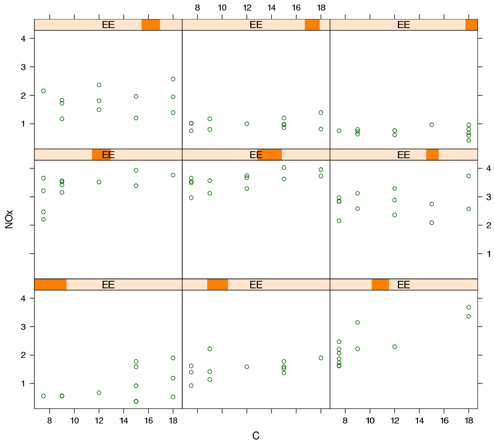
Measurement of Exhaust from Burning Ethanol

- 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.

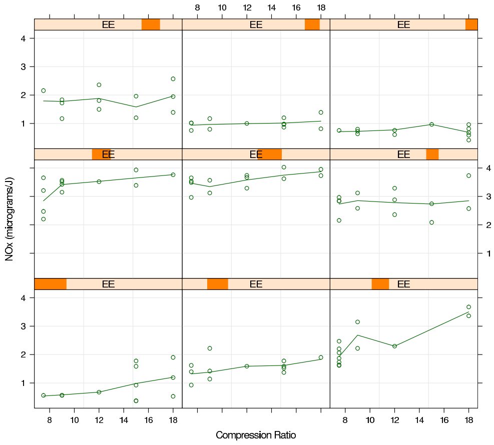
```
> EE = equal.count(ethanol$E, number = 9,
                   overlap = 1/4)
> xyplot(NOx ~ C | EE, data = ethanol)
```



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