Trellis Graphics

The Trellis graphics system is provided in R by the **lattice** package.

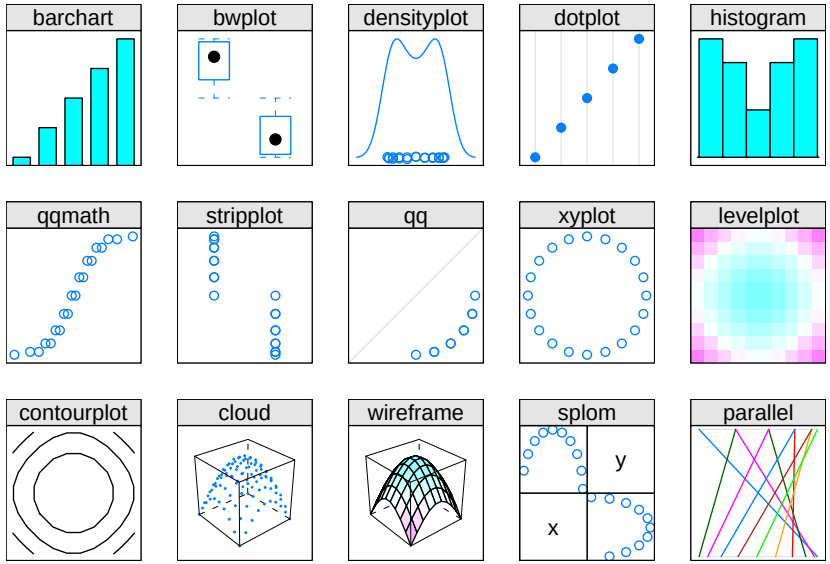
Fundamental ideas

1. Plots are based on principles of human perception.
2. Everything has a sensible default.
3. Anything is possible, given enough arguments.

Types of Functions

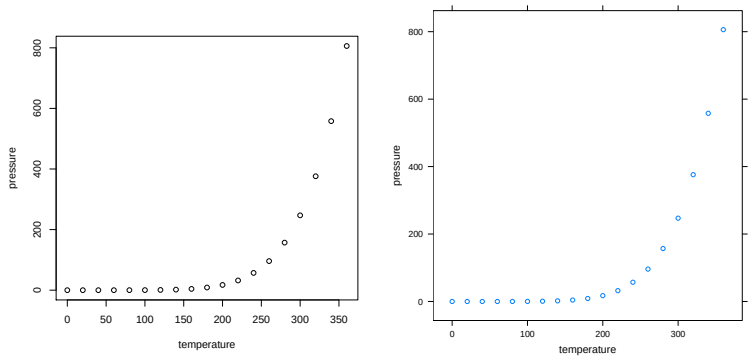
lattice mostly consists of high-level functions that draw (very sophisticated) plots.

```
barchart()      bwplot()        cloud()
contourplot()   densityplot()   dotplot()
histogram()     levelplot()     parallel()
qq()            qqmath()        splom()
stripplot()     wireframe()     xyplot()
```



Spot the Difference

```
plot(pressure)

xyplot(pressure ~ temperature, data=pressure)
```



Trellis Objects

Return values

The value of a high-level **lattice** function is a Trellis object, which contains a **complete description** of a plot.

Drawing objects

A Trellis object draws a plot when it gets printed; if you do not print the Trellis object, nothing is drawn.

Nothing gets drawn!

```
myplot <- xyplot(1:10 ~ 1:10)
```

Drawing All Plots in a Single Call

A Trellis plot can be very complicated, but all details of the plot, **including annotations**, can be included in the description.

Lattice Formulas

Basic formulas

The first argument to all high-level **lattice** functions is a formula that specifies which variables are to be plotted.

Examples

```
xyplot(y ~ x)
histogram( ~ x)
wireframe(z ~ x*y)
```

This is particularly convenient when used with the `data` argument so that the formula only needs to contain the names of variables within a data frame.

Graphical Parameters

Inline parameters

You can set a number of things just as you would with traditional graphics, by specifying an argument to a high-level function.

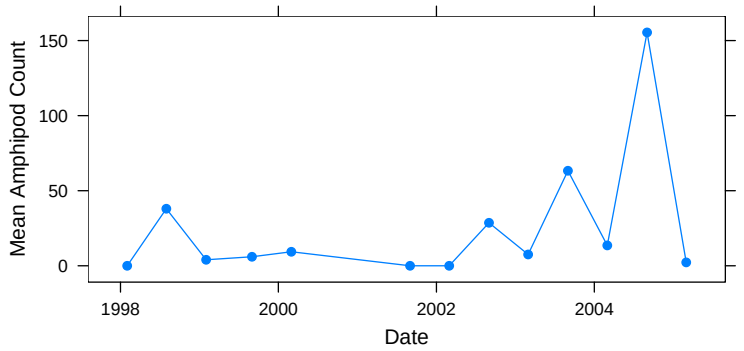
Themes

Lattice also maintains a global list of settings, accessible via `trellis.par.get()` and `trellis.par.set()`. Use `show.settings()` to get a visual summary of the current global settings.

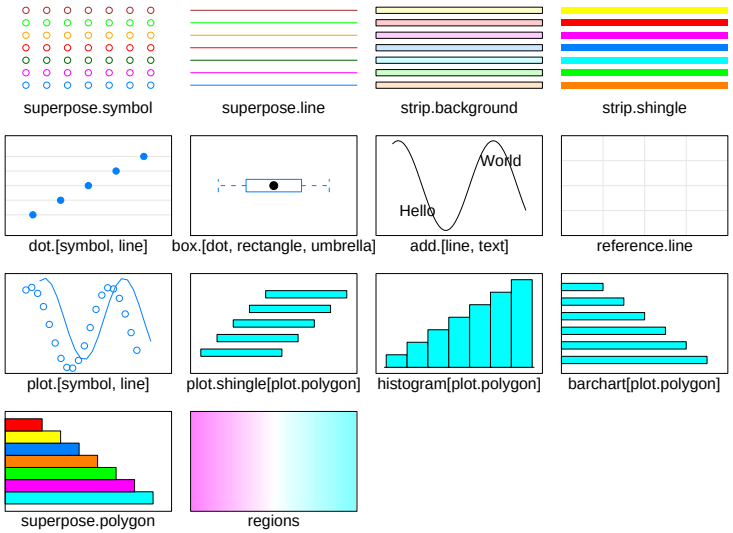
A Basic Trellis Plot

Standard arguments for graphical parameters

```
xyplot(Amphipods ~ Date, data=arcavg,
       subset=Station == "m" & Beach == "LN",
       type="o", pch=16, ylab="Mean Amphipod Count")
```



Trellis Settings



Trellis Settings

Getting settings

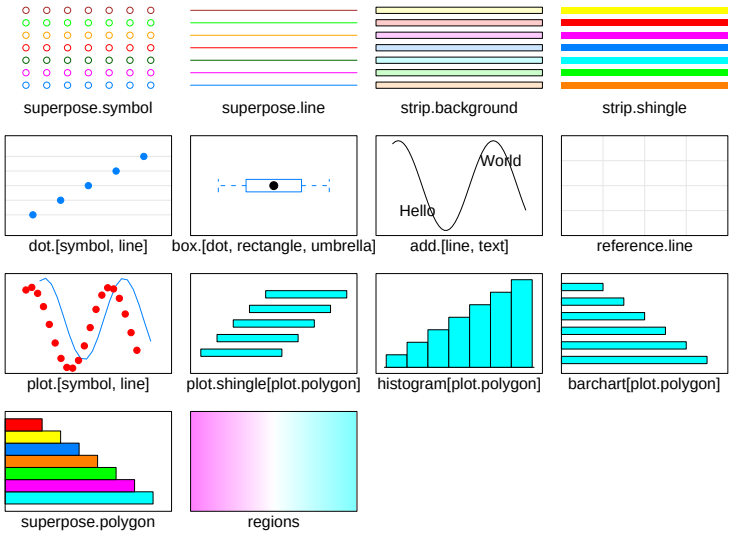
```
unlist(trellis.par.get("plot.symbol"))
```

```
      alpha         cex         col        font         pch
        "1"       "0.8"   "#000000"         "1"         "1"
       fill
"transparent"
```

Setting settings

```
trellis.par.set(plot.symbol=list(col="red",
                                 pch=16))
```

New Trellis Settings



Lattice Formulas

Conditioning variables

The formula can include one or more conditioning variables. This will produce multiple panels of the basic plot type.

Examples

```
xyplot(y ~ x | g)
xyplot(y ~ x | g1*g2)
xyplot(y ~ x | g1 + g2)
```

Conditioning Variables

Shingles

All conditioning variables must be categorical. Factors are obvious candidates, but a continuous variable can be "converted" using `equal.count()` to create a **shingle**.

Examples

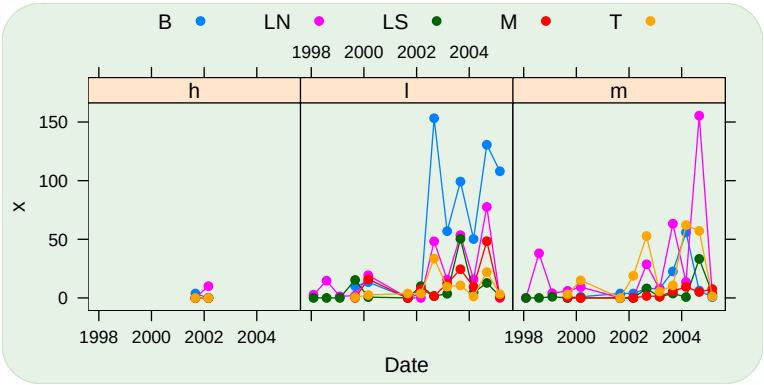
```
equal.count(sort(rnorm(10)), number=2)
```

```
Data:
 [1] -3.13518569 -0.86095696 -0.84712358 -0.58304740
 [5] -0.51451435 -0.49258249 -0.46881469 -0.19786042
 [9] -0.01911672  0.73303473

Intervals:
         min         max count
1 -3.1421024 -0.48566580     6
2 -0.5899641 -0.01220003     6

Overlap between adjacent intervals:
[1] 3
```

A Basic Trellis Plot

Multipanel conditioning

```
trellis.par.set(superpose.symbol=list(pch=16))

xyplot(Amphipods ~ Date | Station, group=Beach, data=arcavg,
       type="o", auto.key=list(columns=5))
```



Multiple Data Series

The `group` argument

Multiple data series can be plotted within the same panel by specifying a grouping variable.

The `auto.key` argument

A legend can be automatically generated by specifying `auto.key=TRUE`.

Panel Functions

Annotation *a priori*

Any customizations of a Trellis plot have to be included in the overall description of the plot via a **panel function**.

There are also `strip` and `prepanel` arguments for specifying other customizations.

Lattice versions of Traditional low-level functions

There are three sorts of functions you can use in a lattice panel function. Lattice panel functions, such as `panel.xyplot()`, `panel.abline()`, ...; grid functions (see later); lattice low-level functions, such as `llines()`, `ltext()`, ...

The lattice low-level functions are designed to protect traditional graphics users from the "terrors" of grid.

Panel Functions

Using lattice panel functions

In general, the lattice function `foo()` will use a panel function `panel.foo()`. However, there are exceptions, such as `xyplot()` using `panel.superpose()` (not `panel.xyplot()` when there is a `group` variable).

Using lattice panel function arguments

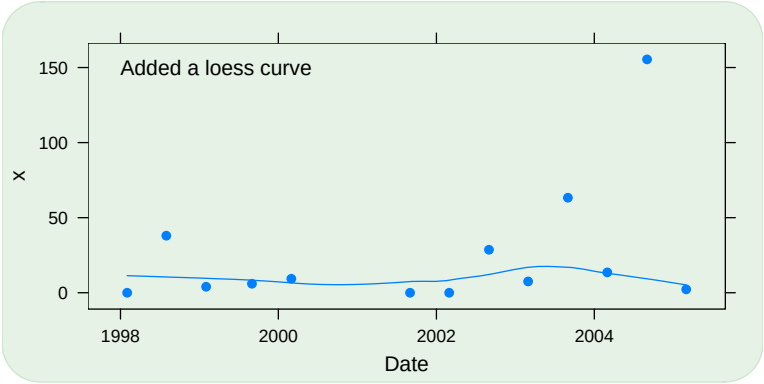
The documentation for each panel function describes the arguments for that function, but often you do not need to make use of all of them. You can "hide" all the arguments you do not need in an ellipsis argument (`...`) in your panel function.

Panel Functions

Lattice panel function

```
xyplot(Amphipods ~ Date, data=arcavg, pch=16,
       subset=Station == "m" & Beach == "LN",
       panel=function(x, y, ...) {
         ltext(as.Date("1998-1-1"), 150,
               "Added a loess curve", adj=0)
         panel.xyplot(x, y, ...)
         panel.loess(x, y, ...)
       })
```

A Basic Trellis Plot



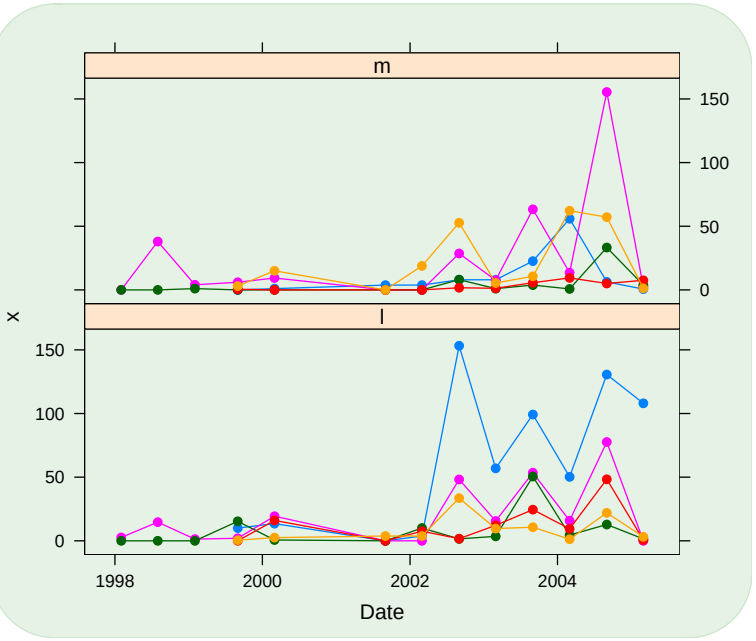
Arranging Lattice Plots

Arranging panels

For multipanel Trellis plots, positioning the panels within a plot is a task in itself. There are arguments such as `layout`, `skip`, `between`, and `aspect`.

Panel layout

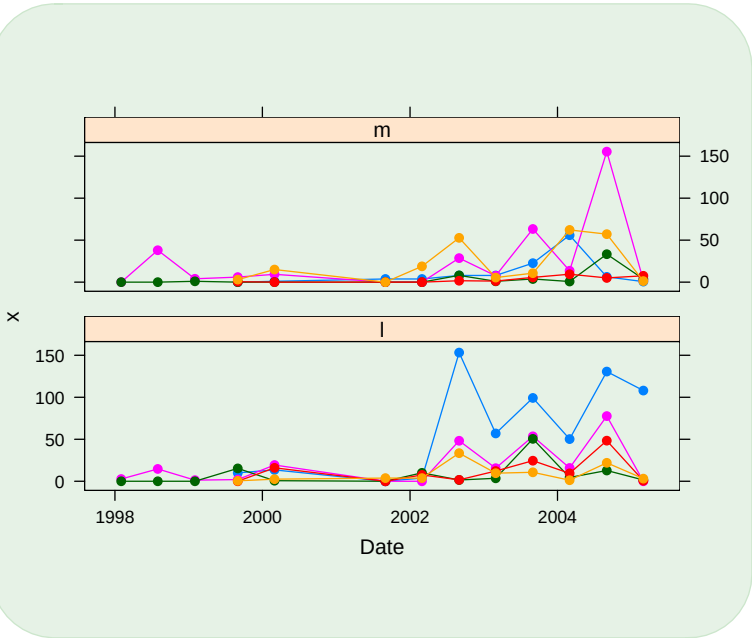
```
xyplot(Amphipods ~ Date | Station, group=Beach, data=arcavg,
       type="o", pch=16, subset=Station != "h",
       layout=c(1, 2))
```



A Basic Trellis Plot

Panel layout

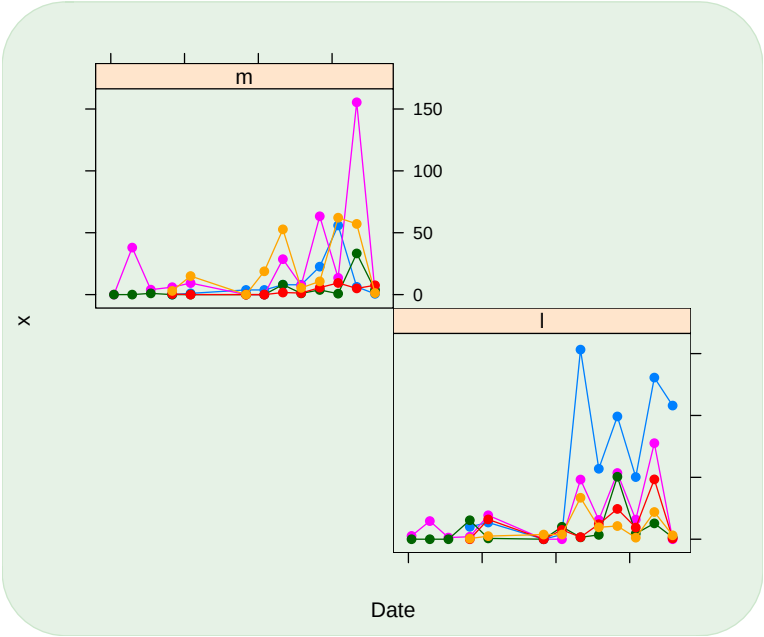
```
xyplot(Amphipods ~ Date | Station, group=Beach, data=arcavg,
       type="o", pch=16, subset=Station != "h",
       layout=c(1, 2),
       between=list(y=1), aspect=0.25)
```



A Basic Trellis Plot

Panel layout

```
xyplot(Amphipods ~ Date | Station, group=Beach, data=arcavg,
       type="o", pch=16, subset=Station != "h",
       layout=c(2, 2),
       skip=c(TRUE, FALSE, FALSE, TRUE))
```



Notes

Notes

Notes