Multivariate displays – Trellis Graphics and Maps

6. More Lattice Plots

"Trellis" plots are the R version of Lattice plots that were originally implemented in the S language at Bell Labs. The aim of these plots is to extend the usual kind of univariate and bivariate plots, like histograms or scatter plots, to situations where some external variables, possibly categorical or "factor" variables, may influence the distribution of the data or form of a relationship. They do this by generating a trellis or lattice of plots that consist of an array of simple plots, arranged according to the values of some "conditioning" variables.

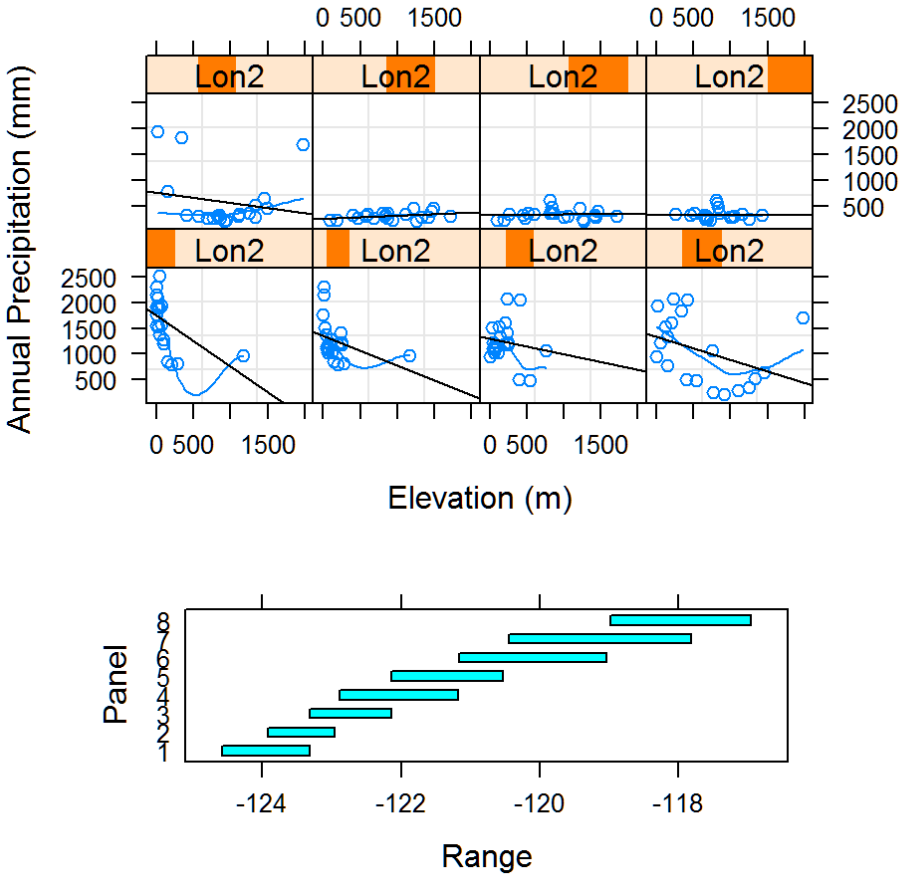
A Trellis-type plot

A multipanel plot, in which the individual panels are "conditioned" by the value of a third variable, here longitude, can be illustrated for the Oregon climate station data using the following script:

```
library(lattice)

attach(orstationc)
# make a factor variable indicating which longitude band a station falls in
Lon2 <- equal.count(lon,8,.5)
# plot the lattice plot
plot1 <- xyplot(pann ~ elev | Lon2,
    layout = c(4, 2),
    panel = function(x, y) {
        panel.grid(v=2)
        panel.xyplot(x, y)
        panel.loess(x, y, span = 1.0, degree = 1, family="symmetric")
        panel.abline(lm(y~x))
    },
    xlab = "Elevation (m)",
    ylab = "Annual Precipitation (mm)")
print(plot1, position=c(0,.375,1,1), more=T)

# add the shingles
print(plot(Lon2), position=c(.1,0.0,.9,.4))
```




```
detach(orstationc)
```

The idea here is to chop longitude into eight bands from west to east using the equal.count() function. (The third argument here, 0.5, indicates that the bands should overlap by 50 percent.) Then the lattice plot is made using the xyplot() function, which makes a separate scatter plot for each longitude band, showing the relationship between annual precipitation and elevation. A “shingles” plot is added at the bottom to indicate the range of longitudes that go into each plot.

Notice that in each panel, a straight regression line (more about regression later) and a smooth lowess curve have been added to help summarize the relationships. The panels are arranged in longitudinal order from low (west) to high (east, remember that in the western hemisphere, longitudes are negative). The plots are certainly interesting. The general idea is that precipitation should increase with increasing elevation, but at least for the western part of the state the reverse seems to be true! What is going on here is that proximity to the Pacific is a much more important control than elevation, and low elevation coastal and inland stations are quite wet. In the eastern part of the state (top row of panels), the expected relationship holds, but it’s kind of hard to see because the wet western part of the state stretches out the scale.,

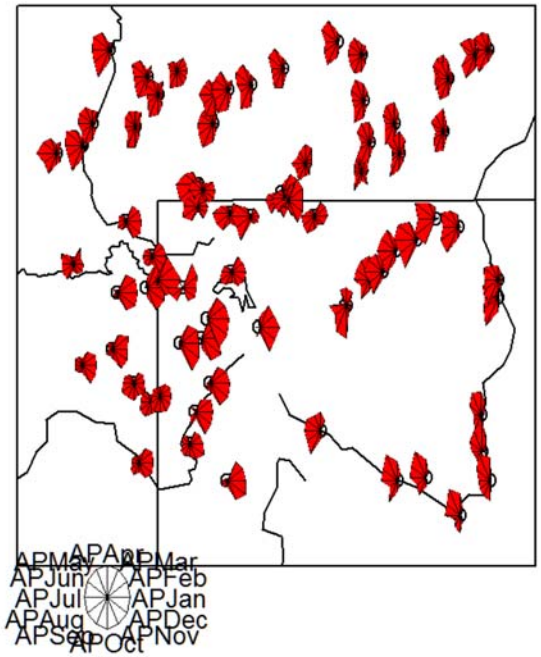
Some more plots

The following plots explore the seasonality of precipitation in the Yellowstone region. This first plot uses glyphs to show the values of twelve monthly precipitation variables as “spokes” of a wheel, where each variable is plotted relative to its overall range. The first block of code below sets things up, and the **`stars()`** function does the plotting.

```
library(sp)
attach(yellpratio)

# simple map
# plot Yellowstone shape files
plot(ynpstate.shp)
plot(ynplk.shp, add=T)
plot(ynprivers.shp, add=T)
points(Lon, Lat)

# stars plot for precipitation ratios
col.red <- rep("red",length(orstationc[,1]))
stars(yellpratio[,4:15], locations=as.matrix(cbind(Lon, Lat)),
col.stars=col.red, len=0.2, key.loc=c(-111.5,42.5), labels=NULL, add=T)
```

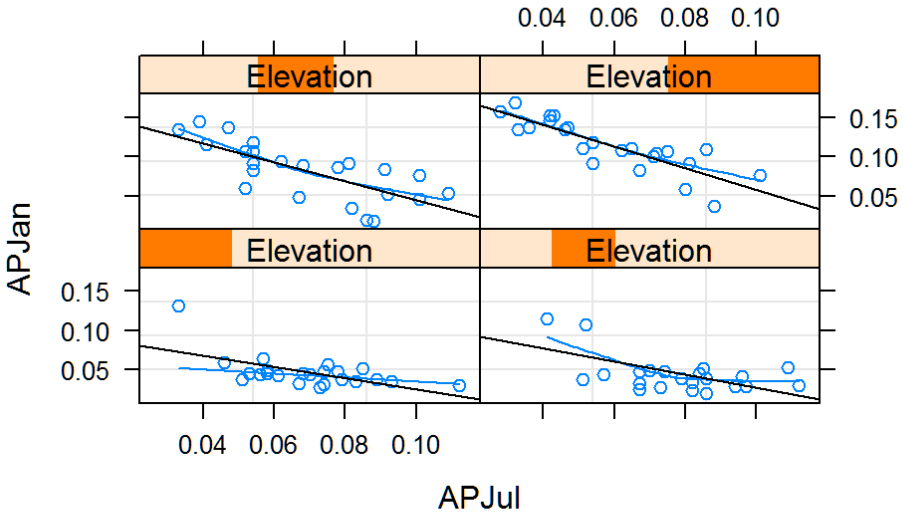


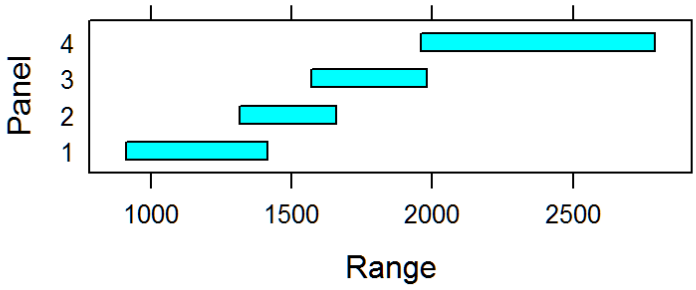
Here the stars wind up looking more like fans. The legend indicates that stations with fans that open out to the right are stations with winter precipitation maxima (like in the southwestern portion of the region) while those that open toward the left have summer precipitation maxima (like in the southeastern portion of the region).

The next examples show a couple of conditioning plots (coplots), that illustrate the relationship between January and July precipitation, as varies (is conditioned on) with elevation. The first block of code does some set up.

```
# create some conditioning variables
Elevation <- equal.count(Elev,4,.25)
Latitude <- equal.count(Lat,2,.25)
Longitude <- equal.count(Lon,2,.25)

# January vs July Precipitation Ratios by Elevation
plot2 <- xyplot(APJan ~ APJul | Elevation,
    layout = c(2, 2),
    panel = function(x, y) {
        panel.grid(v=2)
        panel.xyplot(x, y)
        panel.loess(x, y, span = 1.0, degree = 1, family="symmetric")
        panel.abline(lm(y~x))
    },
    xlab = "APJul",
    ylab = "APJan")
print(plot2, position=c(0,.375,1,1), more=T)
print(plot(Elevation), position=c(.1,0.0,.9,.4))
```

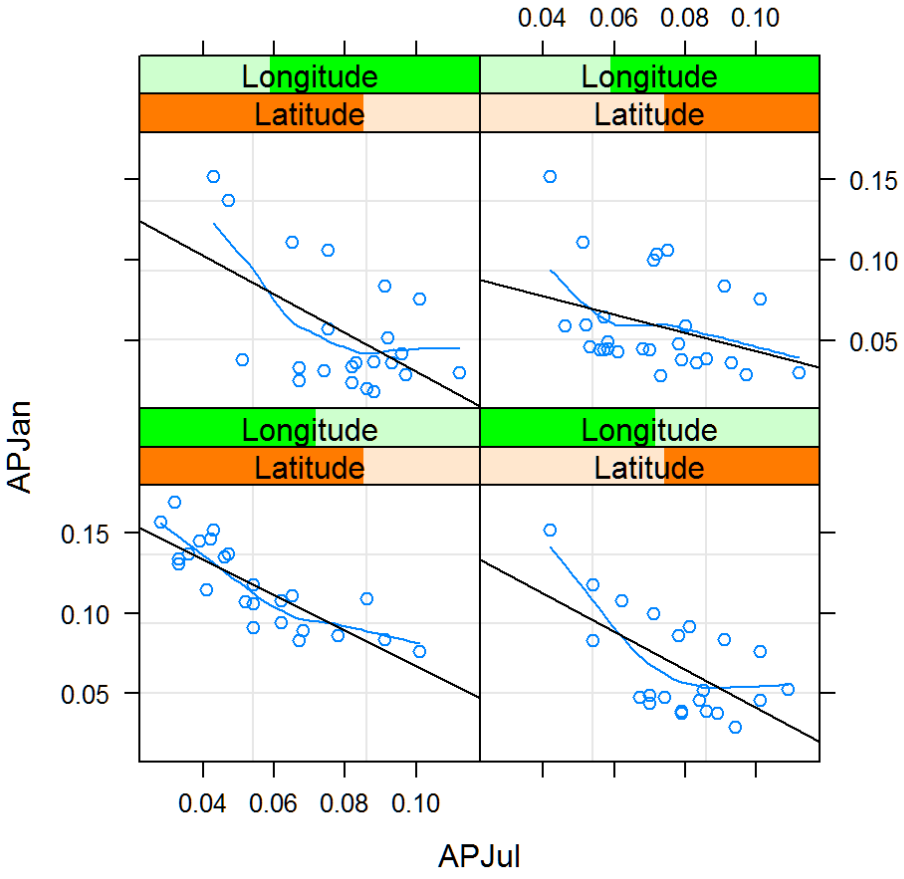




The plot shows that the relationship between January and July precipitation indeed varies with elevation. At low elevations, there is proportionally lower January precipitation for the same July values (lower two panels on the lattice plot), but at higher elevations, there is proportionally more (top two panels). This relationship points to some orographic (i.e. related to the elevation of the mountains) amplification of the winter precipitation.

The next plot shows the variation of the relationship between January and July precipitation as it varies spatially.

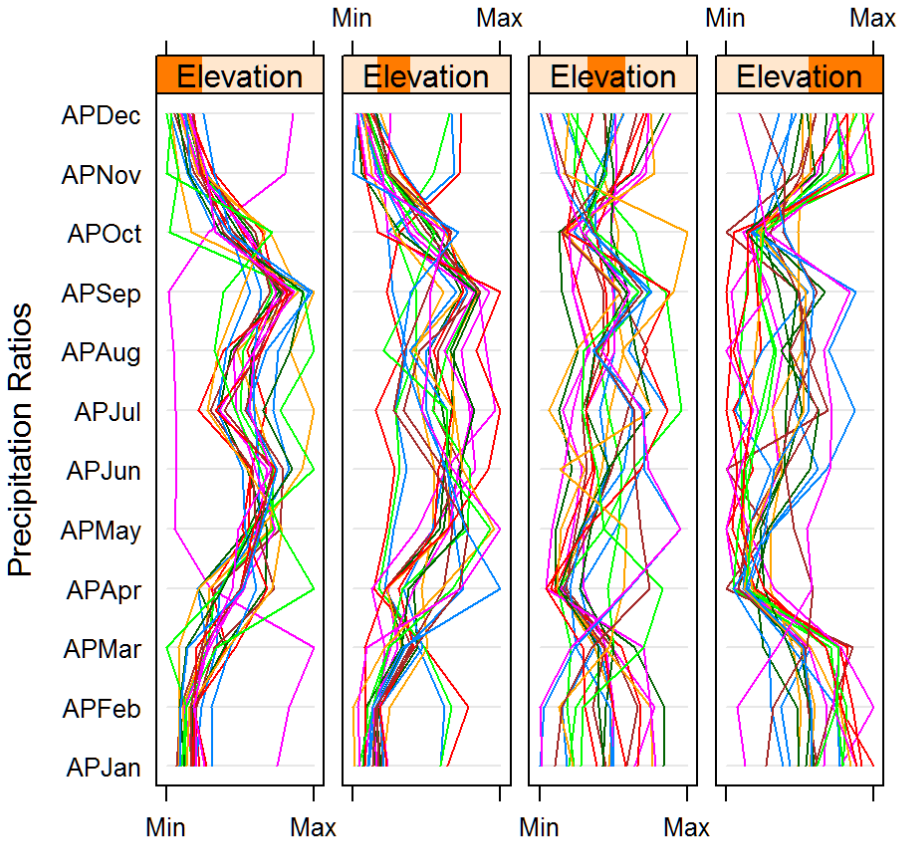
```
# January vs July Precipitation Ratios by Latitude and Longitude
plot3 <- xyplot(APJan ~ APJul | Latitude*Longitude,
    layout = c(2, 2),
    panel = function(x, y) {
        panel.grid(v=2)
        panel.xyplot(x, y)
        panel.loess(x, y, span = .8, degree = 1, family="gaussian")
        panel.abline(lm(y~x))
    },
    xlab = "APJul",
    ylab = "APJan")
print(plot3)
```



Notice that the steepest curve lies in the panel representing the southwestern part of the region (low latitude and low longitude, i.e. the bottom left panel), which suggests that winter (January) precipitation is relatively more import there, which is also apparent on the stars plot above.

Next, the general idea that seems to be emerging, that there variations within the region of the relative importance of summer and winter precipitation can be explored by a parallel-coordinate plot, that allow different precipitation "regimes" to be detected by the appearance of distinct "bundles" of curves.

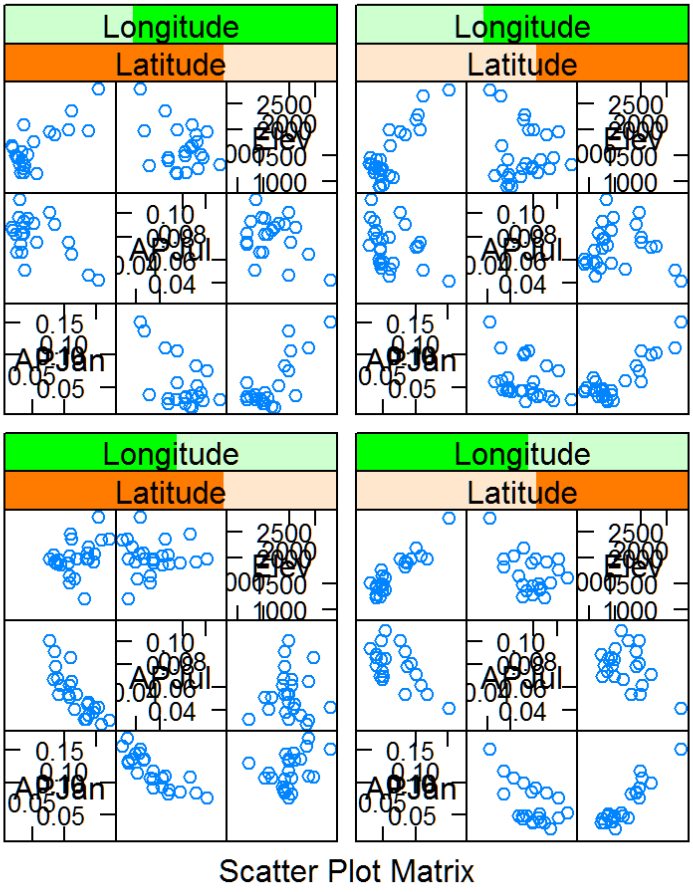
```
# Parallel plot of precipitation ratios
plot4 <- parallelplot(~yellpratio[,4:15] | Elevation,
    layout = c(4, 1),
    ylab = "Precipitation Ratios")
print(plot4)
```



Notice that at low elevations, most of the stations are behaving similarly, and showing a distinct summer precipitation maximum (and only one station seems to show a winter maximum). At high elevations, there is more variability but a general tendency for winter precipitation to dominate.

Lattice plots can extend many of the basic univariate and bivariate plots. For example, a set of scatter plot matrices can be generated, for the high/low latitude and longitude slices.

```
# Lattice plot of scatter plot matrices
plot5 <- splom(~cbind(APJan,APJul,Elev) | Latitude*Longitude)
print(plot5)
```



These plots provide a different prospective on the variations of precipitation across the region, but they're consistent with what the other plots show.

7. Lattice-like plots of maps

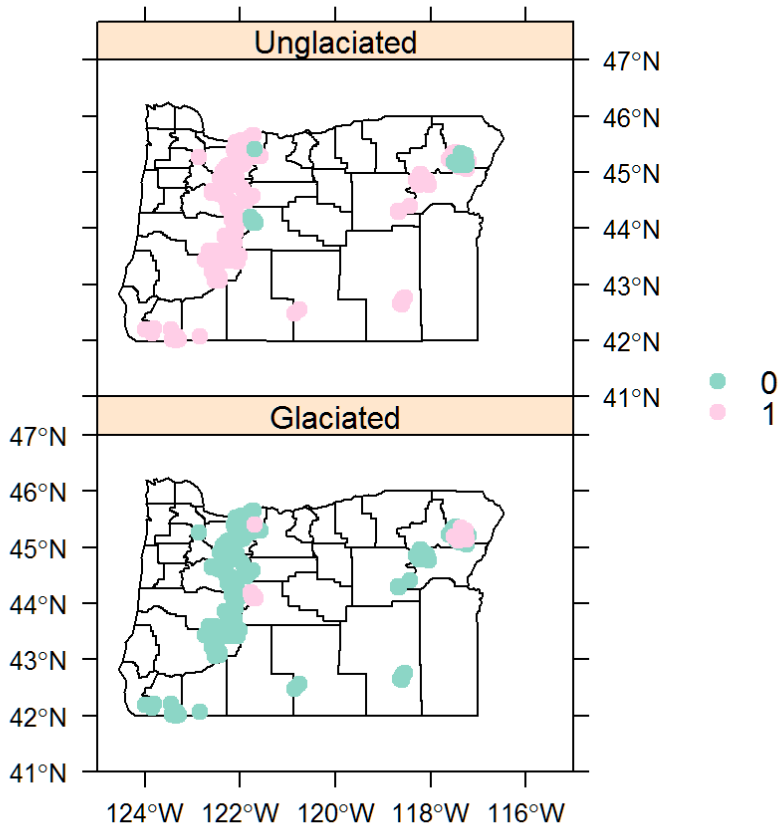
The **spplot()** function in the **sp** package is a Lattice-plot type method, and can be thought of as either extending the capabilities of Lattice plots to maps, or extending the ability of R to produce multi-panel maps.

The following example uses a data set of locations and elevations Oregon cirque basins (upland basins eroded by glaciers), and whether or not they are currently (early 21st century) glaciated. Whether a cirque is occupied by a glacier or not is basically determined by the trade-off between snow accumulation (and hence winter precipitation) and summer ablation (or melting, and hence summer temperature. Cirque basins not currently occupied by glaciers were, of course, occupied in the past, while those occupied today indicate where "glacier-safe" climate prevails (at least for now).

In the code below, the two **as.factor()** functions are used to turn the single variable **cirques.shp$Glacier**, which has the values "**G**" and "**U**", into two "binary" (0 or 1) variables. The variable **cirques.shp$Glaciated** will contain 1's for glaciated cirques, and 0 otherwise (i.e. unglaciated cirques), while the variable **cirques.shp$Unglaciated** will contain 1's for unglaciated cirques, and 0 otherwise. The two variables are obviously redundant (the elements would sum to 1 for each observation), but it makes the illustration of the method more transparent.

```
# multi-panel lattice plot
cirques.shp$Glaciated <- as.factor(ifelse(cirques.shp$Glacier=="G",1,0))
cirques.shp$Unglaciated <- as.factor(ifelse(cirques.shp$Glacier=="U",1,0))
basemap <- list("sp.lines", orotl.shp, fill=NA)

spplot(cirques.shp, c("Glaciated","Unglaciated"),
  col.regions=brewer.pal(8,"Set3"),
  sp.layout=list(basemap),
  xlim=c(-125,-115), ylim=c(41, 47),
  pch=19, aspect=cos((45/360)*(2*pi)),
  scales=list(draw=T), key.space="right")
```

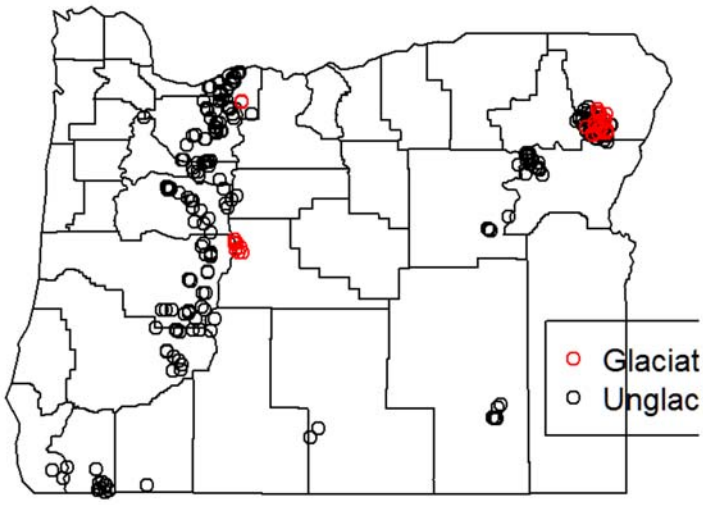


The top panel shows unglaciated cirques in pink and glaciated ones in turquoise, while the bottom panel shows the reverse, glaciated cirques in pink, unglaciated in turquoise. Note the **`aspect`** argument – this scales the horizontal and vertical axes of the plot in a way that makes the map look projected.

It's pretty easy to see where the glaciated cirques occur.

This way of mapping the cirques could also have been done by plotting a simple shape file, and then putting points on top, e.g.

```
plot(orotl.shp)
points(cirques.shp$Lon, cirques.shp$Lat, col=3-as.integer(cirques.shp$Glacier))
legend(-118, 43.5, c("Glaciated","Unglaciated"), pch=c(1,1), col=c(2,1))
```

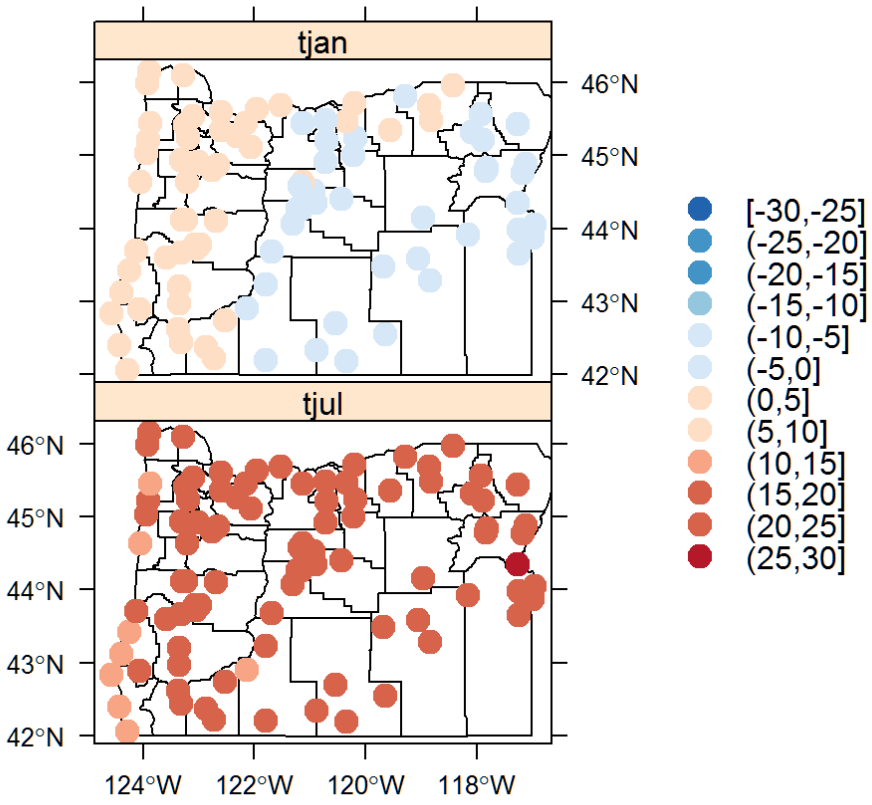


Finally, here are some multi- and single-panel plots of climate-station data, the interpretation of which is straightforward.

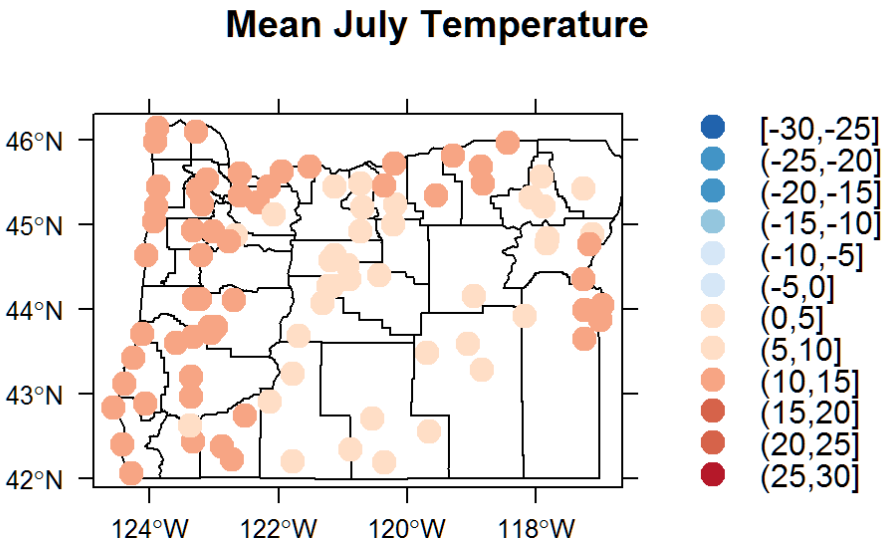
```
# lattice-type plots of climate-station data
nclr <- 8
plotclr <- brewer.pal(nclr,"RdBu")
plotclr <- plotclr[nclr:1] # reorder colors if appropriate

basemap <- list("sp.lines", orotl.shp, fill=NA)
cuts=c(-30,-25,-20,-15,-10.,-5,0,5,10,15,20,25,30)

spplot(orstations.shp, c("tjul", "tjan"), col.regions=plotclr,
     sp.layout=list(basemap), cuts=cuts,
     pch=19, cex=1.5, aspect=cos((45/360)*(2*pi)),
     scales=list(draw=T), key.space="right",
     main="Temperature"
     )
```




```
spplot(orstations.shp, c("tann"), col.regions=plotclr,
     sp.layout=list(basemap), cuts=cuts,
     pch=19, cex=1.5, aspect=cos((45/360)*(2*pi)),
     scales=list(draw=T), key.space="right",
     main="Mean July Temperature"
     )
```


Mean July Temperature
46°N
45°N
44°N
43°N
42°N
124°W
122°W
120°W
118°W
[-30,-25]
(-25,-20]
(-20,-15]
(-15,-10]
(-10,-5]
(-5,0]
(0,5]
(5,10]
(10,15]
(15,20]
(20,25]
(25,30]