Multivariate displays

Multivariate descriptive displays or plots are designed to reveal the relationship among several variables simultaneously.. As was the case when examining relationships among pairs of variables, there are several basic characteristics of the relationship among sets of variables that are of interest. These include:

- the form of the relationships
- the strength of the relationships, and
- the dependence of the relationships on external (usually to the pairs of variables being examined) circumstances.

The easiest way to get the data for the multivariate plotting examples is to download a copy of the workspace **`geog495.RData`** and "load" it, or read it in. Otherwise, all of the individual data sets are available to download from the GeogR data page.
To get the workspace, right-click on this link [geog495.RData] and save it to your working folder. (If you've forgotten where that is, type **`getwd()`** on the command line). Then read it in to R:

```
load("geog495.RData")
```

1. Enhanced 2-D Scatter plots

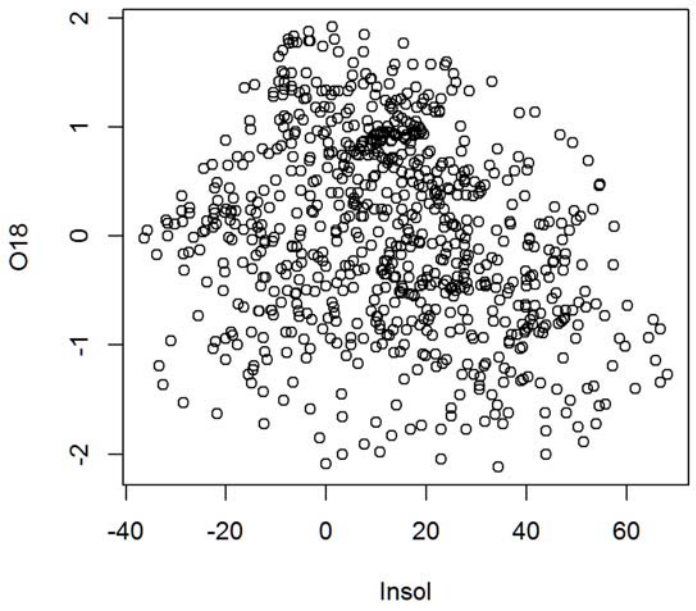
The scatter diagram or scatter plot is the workhorse bivariate plot, and can be enhanced to illustrate relationships among three (or four) variables.

The color-coded scatter plot (color plot)

A basic "color plot"" displays the values of three variables at a time using colored symbols, where the value of one variable determines the relative position of the symbol along the X-axis and the value of a second variable determines the relative position of the symbol along the Y-axis, and the value of the third variable is used to determine the color of the symbol.

The Specmap data set illustrated the variations over time of oxygen-isotope data (that records global ice volume, negative values mean little ice or globally warm conditions, positive values, large ice sheets, and globally cold conditions) which should theoretically depend on insolation (incoming solar radiation) at 65 N, which has been called the "pacemaker of the ice ages". However, a simple plot of Insolation and O18 (and correlation) suggests otherwise:

```
attach(specmap)
plot(O18 ~ Insol)
```



The cloud of points (at first glace) is quite amorphous, and the correlation coefficient is also quite low:

```
cor(O18, Insol)
```

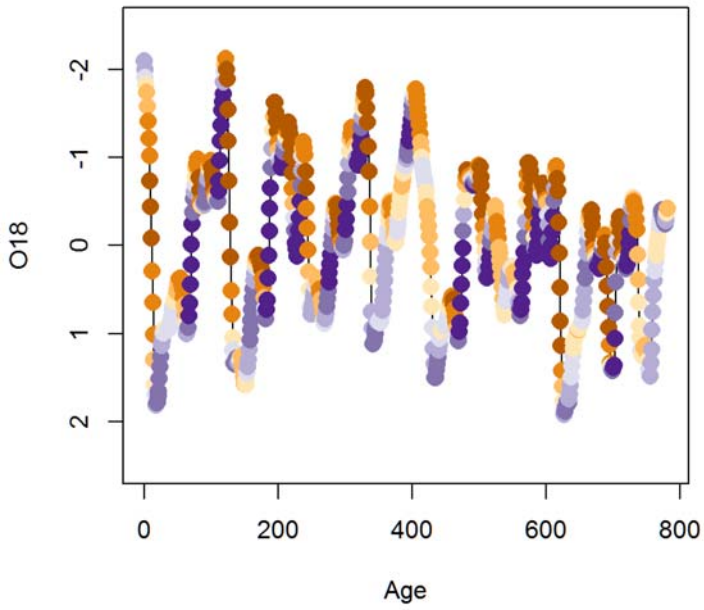
```
## [1] -0.2415094
```

Plotting **`O18`** as a function of **`Age`**, and color coding the symbols by **`Insol`** levels, reveals the nature of the control of ice volume by insolation:

```
library(RColorBrewer)
library(classInt)
plotvar <- Insol
nclr <- 8
plotclr <- brewer.pal(nclr,"PuOr")
plotclr <- plotclr[nclr:1] # reorder colors
class <- classIntervals(plotvar, nclr, style="quantile")
```

```
colcode <- findColours(class, plotclr)

plot(O18 ~ Age, ylim=c(2.5,-2.5), type="l")
points(O18 ~ Age, pch=16, col=colcode, cex=1.5)
```



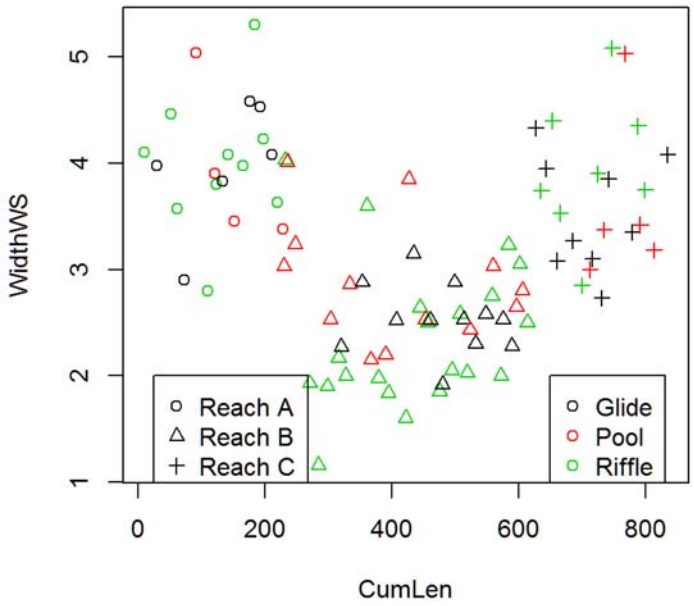

```
detach(specmap)
```

Now it's possible to see that warm (and warming) intervals (points near the top of the plot) tend to have high (orange) solar radiation values, while cooling and cold intervals follow periods of declining solar radiation (blue).

Color and symbols

Information from four variables at a time can also be displayed. In this example for the Summit Cr. data (a scatter plot of **WidthWS** as a function of **CumLen**), the plotting character is determined by Reach and its color by HU. Although these are factors, numerical variables could also be plotted.

```
attach(sumcr)
plot(WidthWS ~ CumLen, pch=as.integer(Reach), col=as.integer(HU))
legend(25, 2, c("Reach A", "Reach B", "Reach C"), pch=c(1,2,3), col=1)
legend(650, 2, c("Glide", "Pool", "Riffle"), pch=1, col=c(1,2,3))
```

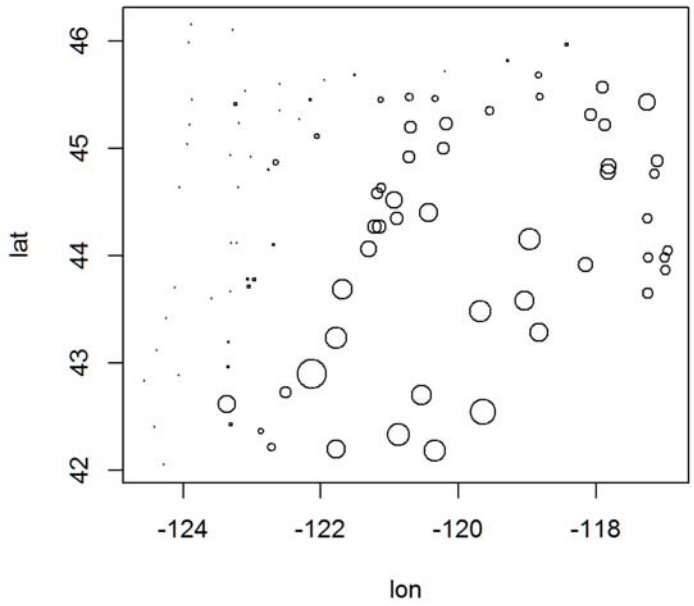


Note the use of two applications of the **`legend()`** function: the circles indicate the upstream grazed reach (reach A), the triangles indicate the cattle-exclosure reach (B), and the pluses indicate the downstream grazed reach (C), while black indicates glides, red indicates pools, and green indicates riffles.

The bubble plot

The bubble plot displays the values of three variables at a time using graduated symbols (usually circles), where the value of one variable determines the relative position of the symbol along the X-axis and the value of a second variable determines the relative position of the symbol along the Y-axis, and the value of the third variable is used to determine the size of the symbol. Here's a crude map of the elevations of the Oregon climate stations, which reflects the overall topography of the state.

```
attach(orstationc)
plot(lon, lat, type="n")
symbols(lon, lat, circles=elev, inches=0.1, add=T)
```



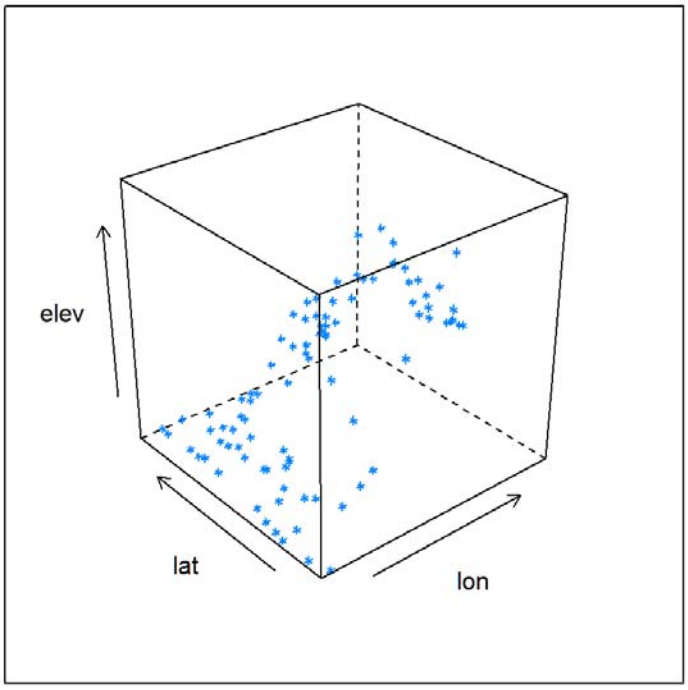
2. 3-D Scatter plots

3-D scatter plots (as distinct from scatter plot matrices involving three variables), illustrate the relationship among three variables by plotting them in a three-dimensional "workbox". There are a number of basic enhancements of the basic 3-D scatter plot, such as the addition of drop lines, lines connecting points, symbol modification and so on.

3-D point-cloud plot

This plot displays the values of three variables at a time by plotting them in a 3-D "workbox" where the value of one variable determines the relative position of the symbol along the X-axis and the value of a second variable determines the relative position of the symbol along the Y-axis, and the value of the third variable is used to determine the relative position along the Z-axis. This plot makes use of the **lattice** package.

```
library(lattice)
cloud(elev ~ lon*lat)
```



Notice that you can still see the outline of the state, because elevation is a fairly well behaved variable.

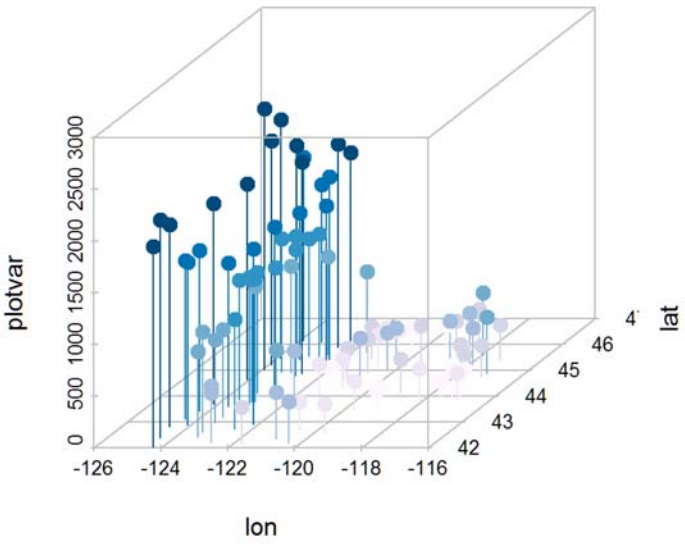
3-D Scatter plots (using the **scatterplot3d** package)

The **scatterplot3d** package (by Ligges and Mächler) provides a way of constructing a 3-point cloud display with some nice embellishments. The first part of the code, like in making maps, does some setup like determining the number of colors to plot and getting their definitions. The second block produces the plot.

```
library(scatterplot3d)

# get colors for labeling the points
plotvar <- pann # pick a variable to plot
nclr <- 8 # number of colors
plotclr <- brewer.pal(nclr,"PuBu") # get the colors
colornum <- cut(rank(plotvar), nclr, labels=FALSE)
colcode <- plotclr[colornum] # assign color

# scatter plot
plot.angle <- 45
scatterplot3d(lon, lat, plotvar, type="h", angle=plot.angle, color=colcode, pch=20, cex.symbols=2,
  col.axis="gray", col.grid="gray")
```



The z-variable, in this case, annual precipitation, is plotted as a dot, and for interpretability a drop line is plotted below the dot. This simple addition facilitates finding the location of each point (where it hits the x-y, or latitude-longitude plane), as well as the value of annual precipitation.

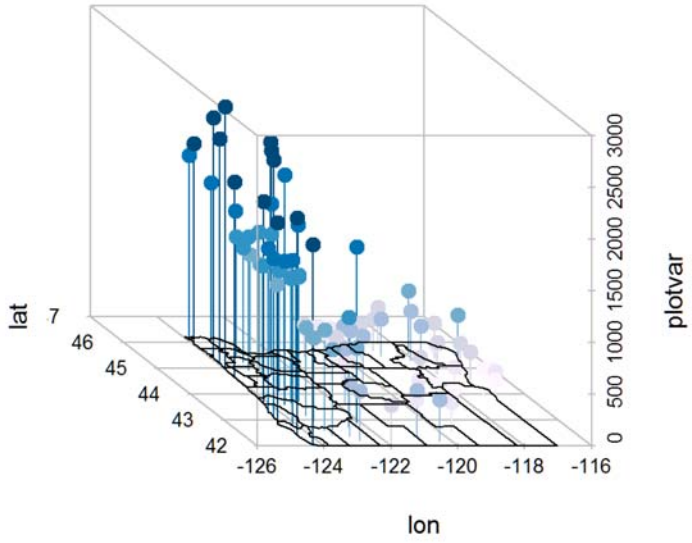
Maps can be added to the 3-D scatter plot to improve interpretability:

```
library(maps)

# get points that define Oregon county outlines
or.map <- map("county", "oregon", xlim=c(-125,-114), ylim=c(42,47), plot=F)

# get colors for labeling the points
plotvar <- pann # pick a variable to plot
nclr <- 8 # number of colors
plotclr <- brewer.pal(nclr,"PuBu") # get the colors
colornum <- cut(rank(plotvar), nclr, labels=FALSE)
colcode <- plotclr[colornum] # assign color

# scatterplot and map
plot.angle <- 135
s3d <- scatterplot3d(lon, lat, plotvar, type="h", angle=plot.angle, color=colcode, pch=20, cex.symbols=2, col.axis="gray", col.grid="gray")
s3d$points3d(or.map$x,or.map$y,rep(0,length(or.map$x)), type="l")
```



The **map()** function generates the outlines of a map of Oregon counties, and stores them in **or.map**, then the colors are figured out, and finally a 3-D scatter plot is made (using the **scatterplot3d()** function, and finally a 3-D scatter plot is made (using the scatterplot3d() function, and the points and droplines are added.

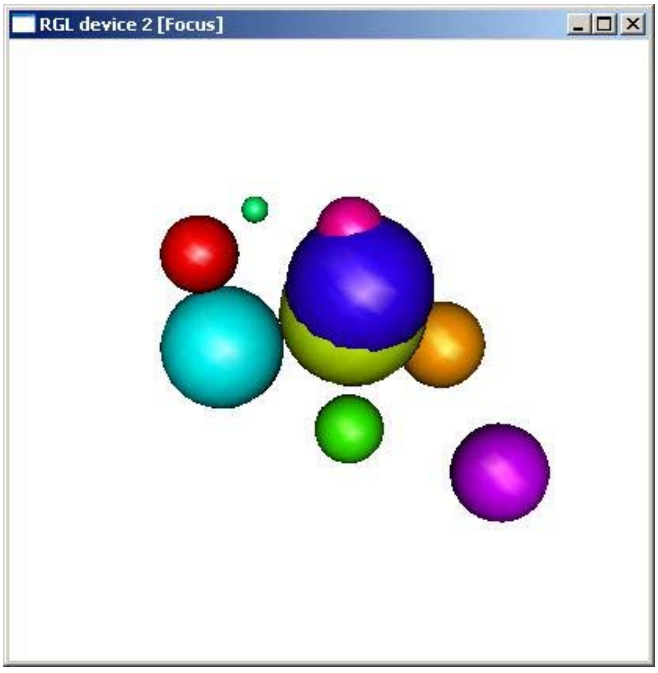
```
detach(orstationc)
```

OpenGL surface and point plots

The **rgl** package (by D. Alder) can be used to plot points (and surfaces and lines) in a 3-D space. The main feature that distinguishes this approach is the ability to rotate the cloud of points "on the fly." Here's what the code looks like, and when the image appears, it can be rotated and spun by dragging the mouse within the window. Holding down the left button while dragging rotates the balls, while holding down the right changes the perspective.

```
library(rgl)
example(rgl.surface)

rgl.clear()
example(rgl.spheres)
```



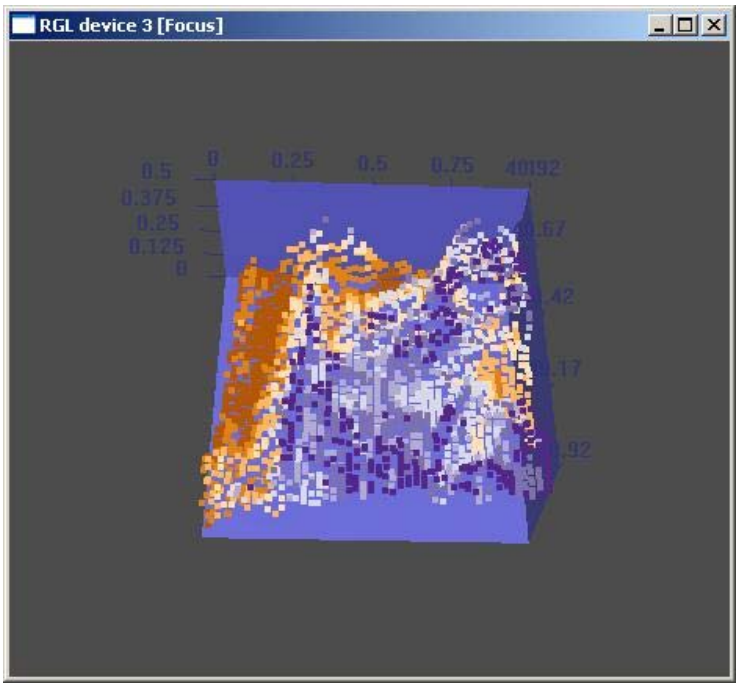
Here's a second example, for a gridded data set of Oregon climate data (elevation in this example) [orgrid.csv] (note that the "RGL device"" window has the habit of hiding behind other windows.

```
attach(orgrid)
```

```
plotvar <- elev # pick a variable to plot
nclr <- 8 # number of colors
plotclr <- brewer.pal(nclr,"PuOr") # get the colors
colornum <- cut(rank(plotvar), nclr, labels=FALSE)
colcode <- plotclr[colornum] # assign color

x <- (lon-min(lon))/(max(lon)-min(lon))
y <- (lat-min(lat))/(max(lat)-min(lat))
z <- (plotvar-min(plotvar))/(max(plotvar)-min(plotvar))
```

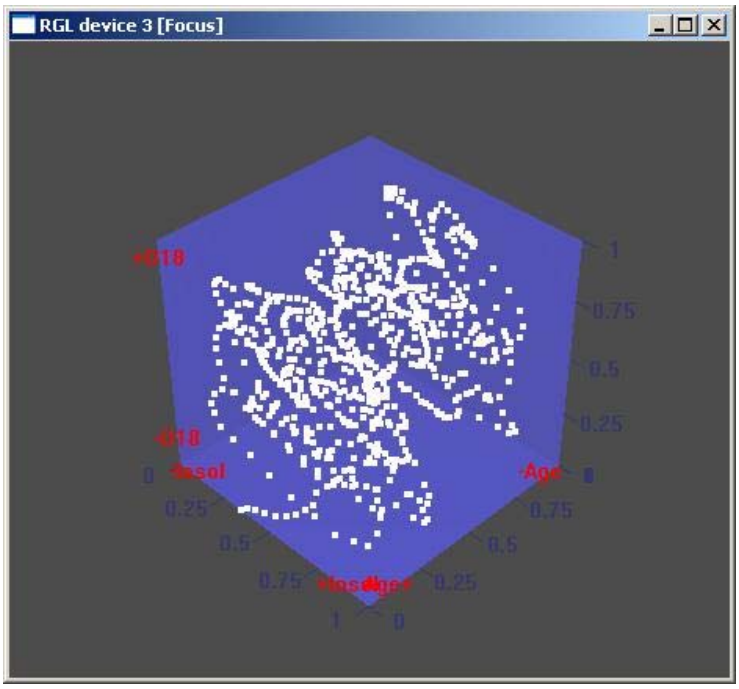
```
rgl.clear()
rgl.points(x, y, z/2, color=colcode, size=4)
rgl.bbox(color="#333377", emission="#333377", specular="#3333FF",
     shininess=5, alpha=0.8 )
detach(orgrid)
```



Here's another example. Plot the Specmap data in 3-D, adding some axes to the plot:

```
attach(specmap)
x <- (Age-min(Age))/(max(Age)-min(Age))
y <- (O18-min(O18))/(max(O18)-min(O18))
z <- (Insol-min(Insol))/(max(Insol)-min(Insol))
```

```
rgl.clear()
rgl.bbox(color="#333377", emission="#333377", specular="#3333FF",
     shininess=5, alpha=0.8 )
rgl.points(x, y, z, size=4)
rgl.texts(.9, 0, 1, "-Age", col="red")
rgl.texts(.1, 0, 1, "Age+", col="red")
rgl.texts(0, 0, .9, "+Insol", col="red")
rgl.texts(0, 0, .1, "-Insol", col="red")
rgl.texts(0, .9, 0, "+O18", col="red")
rgl.texts(0, .1, 0, "-O18", col="red")
```



3. Contour, level, and surface plots

Contour plots are the multivariate plot type that is likely the most familiar to geographers. In R, there are two kinds of contour plots, "2-D contour plots" in which contours are drawn on a standard set of scatter diagram axes, and levels plots that illustrate similar information by coloring or shading a grid of points on the 2-D space, and contouring is achieved visually. All contour plots are constructed by selecting three variables. There are additional ways of creating contour plots which will be discussed later.

2-D contour plot

The following script creates a contour plot for annual precipitation data at Oregon climate stations (the lines), overlays this on a levels or color plot, and also adds the station locations as dots. The **`interp()`** function from the **`akima`** package does the work of interpolating the scattered values of precipitation at the stations onto a regular grid (**`interp.out`**) that can be inferred from the squares in the levels plot. Note that areas within the plotting region that are "unsupported" by stations are treated as missing. This is usually preferable, but sometimes for aesthetic reasons one might want to smear the data into to empty regions. The **`image()`** function plots the color, while the **`contour()`** function plots the contour lines. The **`points()`** function adds the "control points""

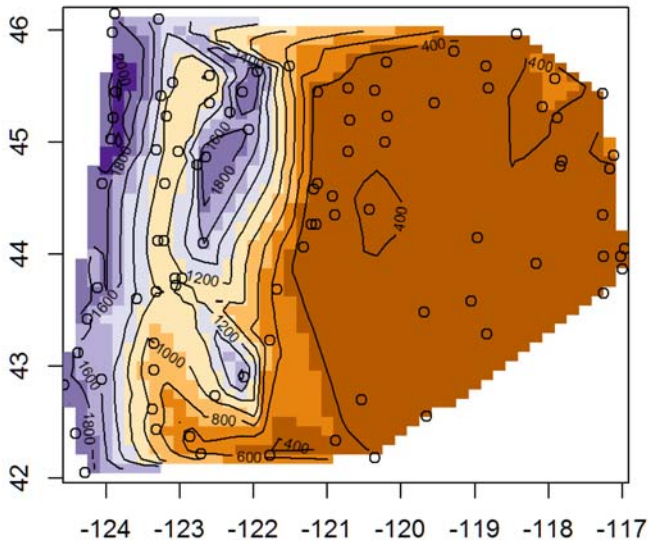
```
library(akima)
attach(orstationc)
```

```
## The following objects are masked from orgrid:
##
##     elev, lat, lon, pann, pjan, pjul, tann, tjan, tjul
```

```
x.ctrl <- lon
y.ctrl <- lat
z.ctrl <- pann
interp.out <- interp(x.ctrl, y.ctrl, z.ctrl,
    xo= seq(-124.5000, -116.8333, .1667), yo= seq(42.0000, 46.1667, .0833))
```

Plot the interpolated values a a color level(s) plot (using the **`image()`** function), add the contour lines on top, and finally add the control points.

```
image(interp.out, col=brewer.pal(8,"PuOr"))
contour(interp.out, xlab="", ylab="", add=T)
points(x.ctrl, y.ctrl)
```

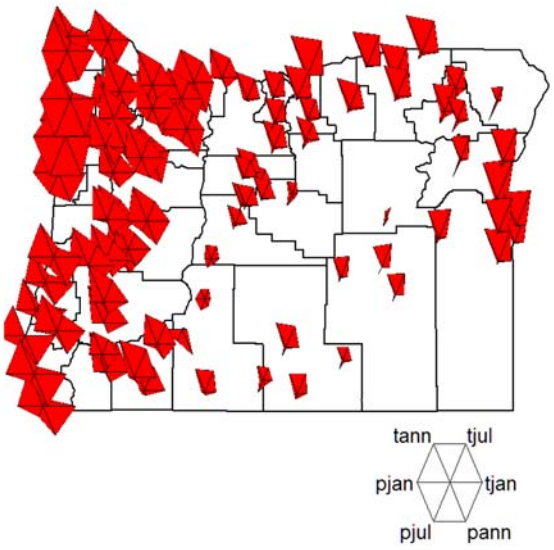


4. High-Dimensional Data Plots

There are a number of "high-dimensional" data plots implemented in R that use different graphical approaches for illustrating relationships among many variables at a time, and some recent developments will be illustrated in Lecture 19. One such plot is the scatter plot matrix which we have seen previously. The other plot types are most conveniently described by simply constructing them for a typical data set and seeing what they look like.

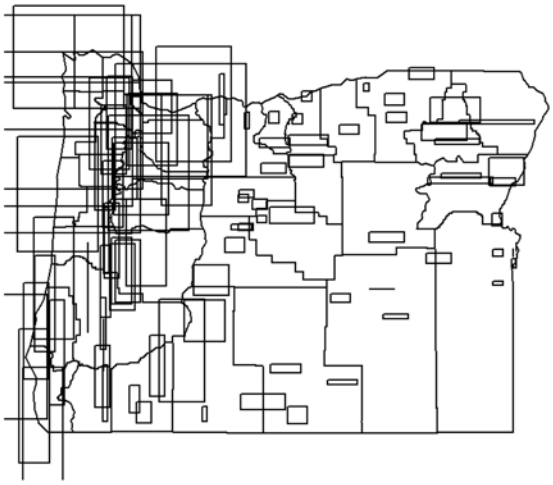
Stars plot

```
# stars
library(sp)
plot(orotl.shp)
col.red <- rep("red",length(orstationc[,1]))
stars(orstationc[,5:10], locations=as.matrix(cbind(lon, lat)),
col.stars=col.red, len=0.5, key.loc=c(-118,41.2), labels=NULL, add=T)
```



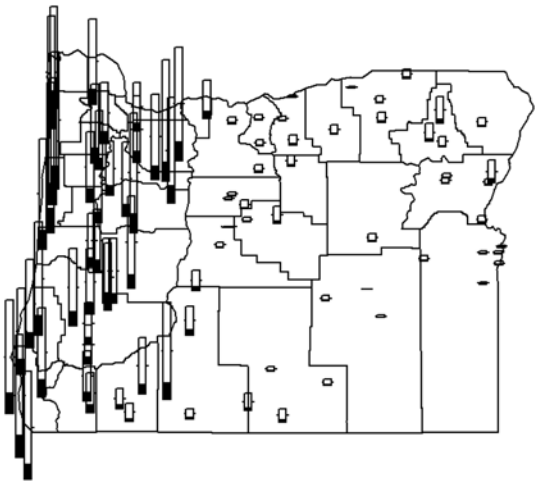
Rectangles plot

```
# rectangles
plot(orotl.shp)
pjul.scale <- (pjul-min(pjul))/(max(pjul)-min(pjul)) # width
pjan.scale <- (pjan-min(pjan))/(max(pjan)-min(pjan)) # height
rects <- cbind(pjul.scale, pjan.scale)
symbols(lon, lat, rectangles=rects, add=T)
```



Thermometers plot

```
# thermometers
plot(orotl.shp)
t1 <- rep(0.05, length(orstationc[,1])) # width
t2 <- (pann-min(pann))/(max(pann)-min(pann)) # height (minimax transformation)
t3 <- pjan/pann # shaded proportion
thermo <- cbind(t1, t2, t3)
symbols(lon, lat, thermometer=thermo, add=T)
```

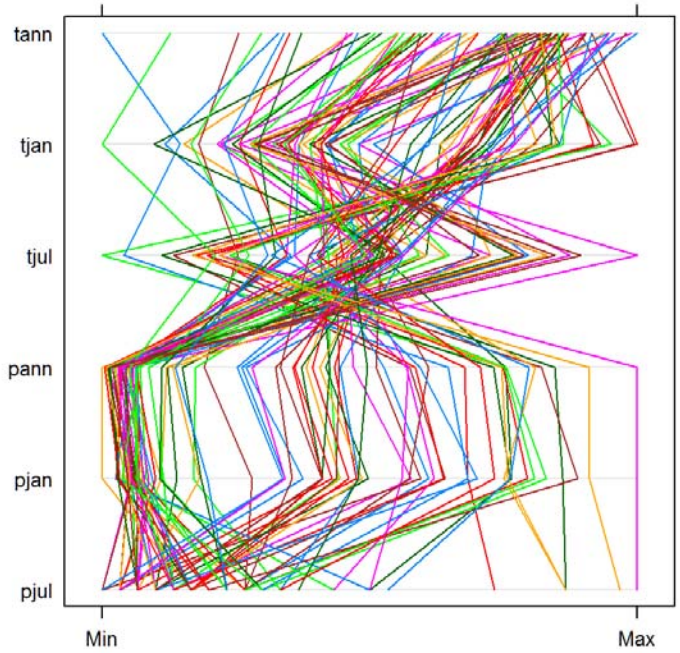


The stars plot (top) shows six variables at each station, January, July and annual temperature and precipitation. The values are scaled to the overall range of each variable. The

contrasts across the state in temperature and precipitation are easy to see, with the western half of the state being warmer and wetter than the eastern. The large range in temperature between January and July is also noticeable in the eastern half of the state. The rectangles plot (middle) shows January and July precipitation, with the values using the "minimax"" transformation (the difference between each value and the minimum value, divided by the range). Again the precipitation contrasts are easy to see. The "thermometers" plot (bottom) shows annual precipitation by the height of the rectangle and the ratio of January to annual precipitation is the shaded portion.

Parallel coordinates display

```
# parallel coordinates display
library(lattice)
parallelplot(~cbind(pjul,pjan,pann,tjul,tjan,tann))
```



The parallel coordinates display shows each observation by a line connecting the scaled value of each variable. Clusters of observations that have similar values of each variable can be seen as bundles of lines. For example, the bundle of lines with low precipitation values (at the lower left of the plot) can be seen to have the highest relative values of July temperature, and relatively low values of January temperature. Eastern Oregon, in other words. The parallel coordinates plot is the one "modern" data visualization method that can easily be constructed in Excel, but that's accidental.