36-315: Statistical Graphics and Visualization

Lab 6
Date: February 19, 2002 **Due: start of class February 25, 2002**

---

1 Introduction

In this lab, you will design visualizations of the relationship between two Census variables and a third. These plots often show more than what is visible in pairwise scatterplots. However, they involve lots of choices; the same ones for a 2D scatterplot, plus several more. The first example will run through a checklist of the choices you'll be making.

For this lab you'll need the extra functions provided at

`www.stat.cmu.edu/~minka/courses/36-315/code/three.s`

(There is also an R version in that directory.) The data will be both census files from your state, `tracta.csv` and `tractb.csv`, assumed to be stored in a variable called `frame`. These lines should remove all problematic tracts:

```
i <- sapply(frame,function(a) all(!is.finite(a)))
frame <- frame[!i]
i <- apply(sapply(frame,is.finite),1,all)
frame <- frame[i,]
```

2 Example

In class it was shown that, in Kentucky at least, there is a relationship between population density and the percent of female residents. To see if this is true in your state, try making a 3D scatterplot:

```
plot3(log(POPPSQMI),PCTFEMHE,MEDHHINC,frame)
```

To better locate the points in 3D, draw lines to the floor:

```
plot3(log(POPPSQMI),PCTFEMHE,MEDHHINC,frame,type="h")
```

Surprised? Most of the data is up against the front wall.

The `plot3` function, like others in this lab, processes its arguments in a special way: if a fourth argument is given which is a data frame, the first three arguments are assumed to be columns of the frame. The axis labels are then chosen to be the names of the columns, so you don't have to specify them separately. Any extra parameters like `xlim` and `ylim` can be given after the fourth.

2.1 Asymmetric plots

A plot of three variables can be symmetric or asymmetric. A symmetric plot uses 3D perspective to represents all variables positionally. Unfortunately, it is difficult to grasp the relative position of points when they are in perspective. An asymmetric plot singles out one variable and gives it a special encoding (length, angle, area, or color). This usually works better. An exception is when the data traces out a clearly defined three-dimensional shape, e.g. a helix.

A bubble plot uses area for the third variable, and a vane plot uses angle:

```
bubbles(log(POPPSQMI),PCTFEMHE,MEDHHINC,frame)
vanes(log(POPPSQMI),PCTFEMHE,MEDHHINC,frame)
```

A bubble plot is like a perspective plot in that small bubbles can be interpreted as 'farther away'. The main problem with these plots is that they are susceptible to overplotting. A color encoding is resistant to overplotting, at the expense of resolution:

```
cplot.default(log(POPPSQMI),PCTFEMHE,MEDHHINC,frame)
```

Color plots can also show approximate contours, because the boundaries between colors correspond to similar values of z.

To print this out, you probably want to use grayscale. In S-PLUS, you can open a grayscale plot window via

```
trellis.device(color=F)
```

Al plots to this window will use grayscale. In R, you change the color scheme for each plot via the `color.palette` parameter:

```
cplot.default(log(POPPSQMI),PCTFEMHE,MEDHHINC,frame,color.palette=gray.colors)
```

The possible color palettes are: `gray.colors`, `topo.colors`, `terrain.colors`, `heat.colors`, and `cm.colors`.

2.2 Contours

A scatterplot between x and y can be enhanced by fitting and displaying a smooth curve. Similarly, the relationship between (x, y) and z can be emphasized by fitting a smooth surface. The surface can be displayed via contours or a perspective surface plot. The `lowess` function only works for two variables. For three or more variables, use the `loess` function:

```
fit <- loess(MEDHHINC~log(POPPSQMI)+PCTFEMHE,frame,span=0.3)
```

Here the z variable comes first, followed by , followed by the other variables separated by plus signs. The span parameter controls the neighborhood size, and should be made as small as possible without making the surface too noisy.

To make a basic contour plot of the fitted surface:

```
cplot(fit)
```

A contour plot can also be filled:

```
cplot(fit,fill=T)
```

Both of these plots include a scatterplot on top. To de-emphasize the points in the scatterplot, switch to a different plot symbol, like so:

```
cplot(fit,pch=20)
cplot(fit,pch=".")
```

Another option is the number of contour levels:

```
cplot(fit,nlevels=10)
```

How many contour levels should there be? The default value of five is generally acceptable since you don't want too many contour lines cluttering the display. For filled contours especially you want the boundaries between regions to be distinguishable.

What should the contour levels be? It is tempting to make them equally spaced, which is the default setting for most software packages. But usually it is better to use the quantiles of z for the levels. This ensures that each interval contains the same amount of data, and contours will not be wasted on isolated peaks and valleys. Quantiles also provide insensitivity to transformations of z. Quantiles are the default in `cplot`. You can force equal spacing by adding the parameter `equal=T`.

2.3 Perspective surface

The `persp` function will show the surface in perspective:

```
surface(fit)
```

This plot is good for getting a quick idea of surface shape, but not for seeing details. It is useful to think of this plot as a contour plot with an infinite number of equally-spaced levels. You don't get to see the boundaries between levels; you only get a gestalt. Furthermore, because of the perspective, you can't readily line up surface details with the axes or with the data.

Perspective plots have various options. You can control the viewing direction via `theta` and `phi`:

```
surface(fit,theta=120,phi=50)
```

and the grid resolution:

```
surface(fit,50)
```

In R, the surface is shaded, and you can change the lighting direction via `ltheta` and `lphi`:

```
surface(fit,ltheta=120,phi=50)
```

and the presence of borders:

```
surface(fit,100,border=NA)
```

Generally, the grid resolution should be low, unless the surface has fine detail in which case the borders should be omitted in favor of shading. The other parameters have to be chosen by trial and error.

2.4 Choosing z

Given three variables with none distinguished as a 'response', which one should be coded asymmetrically as the z variable? Differences on the z variable will be hardest to decode, because of the alternate encoding. Therefore the z variable should be the one with the simplest pattern and least amount of noise. Bubble sizes, vane angles, and point colors should vary smoothly across the plot and not be jumbled together. On a contour plot, the contours should be simple, such as parallel lines. If they aren't, try another variable as the z variable. It is okay if the scatter of x versus y gets more complicated if z gets simpler. Note that swapping the x and y variables is unimportant on these types of plots.

For the variables above, you will probably find that `PCTFEMHE` works better as z:

```
bubbles(MEDHHINC,log(POPPSQMI),PCTFEMHE,frame)
cplot.default(MEDHHINC,log(POPPSQMI),PCTFEMHE,frame)
```

With color plots and contour plots, transforming z generally isn't necessary because of the use of quantiles. But for bubble plots, vane plots, and surface plots, a transformation may help the display. This takes trial and error, because the best transformation usually depends on the encoding. A good transformation for a positional encoding, like a scatterplot, may not be the best for an area encoding.

2.5 Aspect ratio

Aspect ratio should be chosen to maximize change of orientation in curves. On a contour plot, the curves are the contours. You don't want an aspect ratio that makes the contours unusually straight and parallel. You want to show curvature and changes in slope. As for the points, they don't play a role in aspect ratio, because here the relationship between (x, y) and z is in the spotlight, not the relationship between x and y. To control the aspect ratio in S-PLUS, turn on "Editable Graphics" and drag the corners of the plot region.

On a surface plot, aspect ratio refers to the scaling of the z axis. In R, this is controlled by the parameter `expand`:

```
surface(fit,expand=0.5)
```

3 Making your own plots

Now it's time to make your own plots. For each triple of variables below, make one contour plot and one perspective plot which best shows their relationship.

1. MEDYRBLT,MEDHHINC,POPPSQMI
2. PCTCOLL4,PCI,PCTNFHHS

It is not specified which variables should be `x`, `y`, and `z`—it is up to you to make a good choice. One contour plot and one perspective plot for each triple will be accepted, for a total of four plots. But don't be surprised if you need to try out three times this many in order to get the right picture. Submit only the code used in your final plots.

Question: Interpret your plots. What are they saying about your state?