36-315: Statistical Graphics and Visualization

Lab 10
Date: March 19, 2002 **Due: start of class March 25, 2002**

---

1 Introduction

In this lab you will make statistical maps using various color schemes. You will be using the `cplot` function in `three.s`, which you should download again since it has changed slightly.

From last time, recall that you can map your state via

```
library(maps)
map('state','kentucky')
```

Now we want to fill this map with data.

2 Statistical maps

In the `tracta.csv` file, there are two variables called `LAT` and `LON` which give the latitude and longitude of the center of each census tract. Thus you can plot the locations of the tracts on the map:

```
points(frame$LON,frame$LAT,cex=0.5)
```

Census tracts vary in size in order to contain around 4,000 people each. Therefore the density of tracts gives an idea of population density. To match up this plot with places in the state, check out the state maps at `http://www.lib.utexas.edu/maps/united_states.html` (just click on the name of your state). What places can you match up?

> **Question:** Pick three of the most heavily populated areas on your map and label them with their commonly-known names. Label them using S-PLUS. (You may want to refer back to labs 5 and 9.) As usual, hand in your plot and the specific code for making it.

Any of the remaining tract variables can be used to color in the map. For example, suppose we want a detailed map of population density (`POPPSQMI`). By approximating each tract as a single point, we can apply techniques from lab 6 to visualize how this `z` variable changes with `x` and `y`. In particular, the `loess` function can give a smooth function which is then contour plotted:

```
fit <- loess(log(POPPSQMI) ~ LON+LAT,frame,span=0.1)
cplot(fit)
cplot(fit,fill=T)
```

(The `span` above is only a suggestion, and may not be appropriate for your state.) By default, the `cplot` function will not show areas far away from the data. You can clip the output to the precise boundary of your state by giving `cplot` a polygon defining the state's border. Such a polygon is conveniently returned by `map` when `fill=T`:

```
m <- map('state','kentucky',fill=T)
cplot(fit,clip=m)
lines(m,col=2)
cplot(fit,fill=T,clip=m)
```

The `lines` command superposes the outline of the state on the plot. Unfortunately, the filled plot does not allow superposition. To make the filled plot look better, you can change the resolution, say `res=50`, as well as the symbol size, say `cex=0.5`.

3 Color

In class, we discussed the hue, saturation, and lightness terminology for describing colors. Hue is the informal notion of "color", like red or green. Saturation measures how pure the color is; lowering the saturation makes the color look progressively "washed out", until it becomes gray. Geometrically, the colors form a cone with black as the tip. The line from the tip to the base is the line of gray values, or zero saturation. The radial distance from this line is the saturation, and rotation about the line is hue. A nice picture of this, as well as other interesting facts about color, can be found at `http://www.handprint.com/HP/WCL/color6.html#hsv`. The dissimilarity of two colors roughly corresponds to the distance between their positions in the cone. Thus the cone captures the fact that reducing lightness reduces the possible range of saturation, until eventually everything is black. The circular cross-section captures the fact that hues are cyclical, and are harder to distinguish when the saturation is low.

Different color schemes are appropriate for different kinds of plots. For example, when drawing points and lines, color variations are less perceptible so color changes must be dramatic. Only a few color levels can be used, and saturation must be high, especially against a white background. This is why `cplot(fill=F)` uses highly saturated red, green, blue, and black as its colors.

When painting large areas with color, as in `cplot(fill=T)`, you get more flexibility. For this situation, there are three main types of color schemes: categorical, sequential, and diverging. A **categorical** scheme is for encoding categorical values with no natural order. Like the previous case, the emphasis is on making the colors as saturated and distinct as possible.

A **sequential** scheme is for encoding numerical values. The colors are chosen to follow a smooth and easily interpretable path through the cone. The parameter easiest to interpret as magnitude is lightness; low values are represented by light colors and high values by dark colors. A gray color scheme is a prime example of this. But it is not the only possibility. Any fixed hue and saturation can be used instead of gray. You can change saturation instead of lightness, or in addition to lightness. It is even possible to slightly change hue during the progression, as long as the light-to-dark trend is maintained. The advantage of doing this is that the colors become more distinguishable, making the plot easier to read and allowing you to use more levels.

A **diverging** scheme is for numerical values that have a critical midpoint. It takes two sequential schemes, based on two different hues, and concatenates them so that one goes up from the midpoint and one goes down. The midpoint is a shared light color and moving away from the midpoint in either direction gives darker colors. Some people use lightness for the progression; others saturation. Saturation is perhaps the more natural choice, since it naturally leads to white at the midpoint.

To change color schemes in S-PLUS, use the function `graphsheet`:

```
graphsheet(color.table=gray.colors(4))
```

This creates a new window that will use 4 gray levels. Besides `gray.colors`, four other color schemes are provided in `three.s`: two sequential schemes called `OrRd.colors` and `YlGnBu.colors` and two

diverging schemes called `RYB.colors` and `BrBg.colors`. They are designed to be used with either four or five levels. The colors will be numbered 2 to 5 or 2 to 6, because 1 is always black. Every time you make a `cplot` in such a window, you need to tell it to use these numbers, via the argument `col`:

```
cplot(fit,col=2:5)
cplot(fit,col=2:6,nlevels=5)
```

4 Making your own plots

Pick one of the ethnicity variables (`PCTWHITE`, `PCTBLACK`, `PCTASIAN`, `PCTAMIND`, `PCTHISP`) and show how it varies across your state. Use appropriate smoothness, levels, aspect ratio, and symbol size. The aspect ratio should reflect the state's proper shape on a map. If you have a big and complicated state, use `xlim` and `ylim` to zoom in on an interesting area.

Now plot `MEDHHINC`. You will probably find an interesting pattern near the cities. Hand in code and both plots. Note: when printing your plots on a black and white printer, you must switch to a gray color scheme in order for it to be properly readable. Color plots printed on a black and white printer are useless and should not be turned in.

> **Question:** Compare the sequential and diverging color schemes. In your opinion, which is easiest to interpret? Do different schemes encourage different interpretations of the plot? Explain.