36-315: Statistical Graphics and Visualization

Lab 12

Date: April 8, 2003 **Due: end of lab**

In this lab, you are given the commands to type in, but with question marks (??) denoting things you must provide, based on the dataset you are using.

1. Download all the files for this lab into `My Documents`. Other folders will not necessarily work.
2. Unzip `maps.zip` by pressing right mouse and selecting `Extract All...`. In the wizard, click `Next`, `Next`, and `Finish`. If the computer complains about WinZip, move to another.
3. Open a Word document to record your work (plots and commands).

Start R

4. `Start -> Programs -> Class software -> R 1.5.1`
5. Set the working directory to `My Documents`:

   ```
   File -> Change dir...
   ```

6. Load the special functions for this lab:

   ```
   source("lab12.r")
   ```

Load your data

7. From the class web page, download the census data files for your state. Unzip them into `My Documents`. Then they can be loaded via

   ```
   a = read.csv("tracta.csv")
   b = read.csv("tractb.csv")
   frame = clean.census.data(a,b)
   dim(frame)
   frame[1,]
   ```

   The last two lines tell you how big `frame` is, and the values in a typical tract, respectively.

Basic maps

8. Make a scatterplot of the centers of the census tracts (`LAT` versus `LON`), with aspect ratio 1, and overlay a map of the States (without projection).

   ```
   plot(LAT~LON,frame,asp=1)
   map("state",add=T)
   ```

This provides a basic check of your data. For some states, the latitudes are shifted by 5 or 10 degrees. For example, for PA you need to shift up via

```
frame[,"LAT"] = frame[,"LAT"] + 5
```

(If you find other states that need to be shifted, please tell us.)

9. Create a map of the projected outline of your state, and store the result in a variable `m`.

   ```
   m = map("state",??,fill=T,proj="albers",par=c(37,39))
   ```

   Then project the tract locations, as in homework 10:

   ```
   frame[c("x","y")] = mapproject(frame[,"LON"],frame[,"LAT"])[c("x","y")]
   ```

   Tell R to use `m` for clipping the image maps you'll make later:

   ```
   formals(image.data.frame)$clip = m
   ```

10. Using the projected latitude and longitude values, map your response variable using centroid smoothing. If you did the last step correctly, the image should be clipped to the shape of your state.

    ```
    image(? ~ ? + ?,frame)
    ```

Load the tract map

11. From the class web page, download the census tract boundary files for your state. Unzip the files into `My Documents`. This map needs to be projected similarly to the state outline:

    ```
    mt = read.cob("tr??_d90_p.dat","tr??_d90_pa.dat")
    mt = map(mt,fill=T,proj='albers',par=c(37,39))
    ```

Tract maps

12. Make a choropleth map of your response variable, using the tract map. Use a large number of colors, and save the result, which is a vector of break values between colors:

    ```
    breaks = map.vector(mt,??,col=YR.colors(128))
    ```

13. Next is to make a smooth map using area smoothing. As a test, try this command:

    ```
    fit = map.smooth(mt,frame["TOTPOP"],span=1/20)
    image(fit,zlab="population density")
    ```

    This gives a smooth population density map. (Because of the complexity of the tract map, `map.smooth` can take 15 seconds or more to complete.)

14. Making smooth maps is complicated by the fact that many of the interesting census variables are ratios. For example, `PCTMCFAM` is the number of married couples in a tract divided by the number of families in the tract. The aggregation of the variables is done *before* the ratio. So to smooth this data, you have to smooth the numerator and denominator variables separately, then divide the result.

    In this step, you need to determine the variables being divided to give your response variable. Some examples of these ratios:

    ```
    POPPSQMI = TOTPOP/LANDSQMI
    PCI = TOTINC/TOTPOP
    PCTCOLL4 = COLLEGE4/AGE25.UP * 100
    PCTMCFAM = MCFAMS/FAMILIES * 100
    ```

    The variables `TOTINC` ("total income") and `AGE25.UP` ("persons over 25") do not exist in the dataset, but must be created:

    ```
    frame[,"TOTINC"] = frame[,"PCI"] * frame[,"TOTPOP"]
    frame[,"AGE25.UP"] = frame[,"COLLEGE4"] / frame[,"PCTCOLL4"]
    ```

    Some variables, like `MEDAGE`, are not ratios at all.

15. If your variable is not a ratio, type

    ```
    fit = map.smooth(mt,??,span=1/20)
    image(fit,zlab="")
    ```

    Otherwise, smooth the numerator and denominator variable in your ratio:

    ```
    fit = map.smooth(mt,??,span=1/20)
    fit.d = map.smooth(mt,??,span=1/20)
    ```

    Divide them, and plot a map:

    ```
    fit[,"z"] = fit[,"z"]/fit.d[,"z"]
    # if your variable is a percentage:
    fit[,"z"] = fit[,"z"] * 100
    image(fit,zlab="",breaks=breaks)
    ```

    This map should have the same color scheme as your earlier choropleth map (because of `breaks`).

16. Vary the span parameter in your smooths. If you make it small, e.g. `span=1/100`, you should get something resembling the choropleth. If you make it larger, e.g. `span=1/10`, you should get something increasingly smooth. Pick the span that gives you a (subjectively) nice picture of how your response variable behaves. You can also choose to omit `breaks`, allowing the colors to be automatically tuned for the smooth map.

17. Show us your graphs.

Centroid smoothing This is done in one step via the `image` function and a formula. For example:

```
image(PCI ~ x + y,frame)
```

The general usage:

```
image(<formula>,frame)
```

Choropleth map `map.vector` is similar to `map.averages` from lab 11. However, you give the data by extracting a column from `frame`, for example:

```
map.vector(mt,frame["TOTPOP"])
map.vector(<map object>,<column of frame>)
```

The column of frame is specified *without* a comma, that is `frame["TOTPOP"]` instead of `frame[,"TOTPOP"]`. This instructs R to include the names of the census tracts (try it). `map.vector` associates these names to map regions.

Area smoothing Area smoothing has two parts: computing the smooth and plotting the smooth:

```
fit = map.smooth(mt,frame["TOTPOP"],span=1/20)
image(fit)
```

In general, the usage is:

```
fit = map.smooth(<map object>,<column of frame>,span=<number>)
image(fit,zlab=<label>,breaks=<vector>)
```

The column of frame is specified as in `map.vector`. `breaks` is optionally used to make the color breaks agree with a previous plot.

Mass-preserving interpolation This is the same as above, except for `type="interp"`:

```
fit = map.smooth(mt,frame["TOTPOP"],span=1/20,type="interp")
```