Representing Color

The problem of processing color information using a computer is very similar to the problem of working with textual data. Because computers can only process ones and zeros, before color data can be analyzed, we need to somehow transform color information into numerical information. As with character data, this is done by encoding the colors using a mapping that associates each color with one or more numbers. There are many different color models schemes, but one of the most widely used is the ***RGB color model***. This encoding is an ***additive model*** that represents each color as a sum of the relative contributions of red, green, and blue light. The RGB model represents each color as a list of three values (red, green, and blue), each of which can range from 0 to 255. Thus, RGB colors are 24 bit colors (why?). With this scheme, the color with the values [255, 255, 255] (i.e., red = 255, green = 255, and blue = 255) is white; [255, 0, 0] is pure red, [0, 255, 0] is pure blue, [0, 0, 0] is black, [255, 175, 175] is a pink, etc. Thus, with the RGB color scheme we can actually represent as many as 256 x 256 x 256 different colors (i.e., over 16 million colors!).

There are many references on the web that can assist you in finding the right combination of red, green, and blue values for generating particular colors, including:

- *RGB color model (Wikipedia) (http://en.wikipedia.org/wiki/RGB_color_model),*
- *RGB color code tables (http://www.tayloredmktg.com/rgb/)*
- *RGB color calculators (http://www.drpeterjones.com/colorcalc*

Using the `graphics.py` library, we can specify the use of specific colors using their RGB code, through the use of the command **`color_rgb`** as follows:

```
<color-name> = color_rgb(<red>, <green>, <blue>)
```

Color Mapping

To explore the use of color as a visualization tool, we are going to be doing some ***geospatial visualization***. The term "geospatial visualization" refers to a set of visualization tools and techniques used to illustrate a very specific type of multivariate data. Geospatial data are data that relate observations (e.g., temperature, elevation, population density, income, etc.) to the physical geographic location (geospatial) at which those observations were made. Like many visualizations, a geospatial visualization is a projection of data onto a coordinate space; however, the space is not defined by *Cartesian coordinates* (as in a scatter plot), but by *geographic coordinates* (e.g., longitude and latitude). Because geospatial visualizations are at a

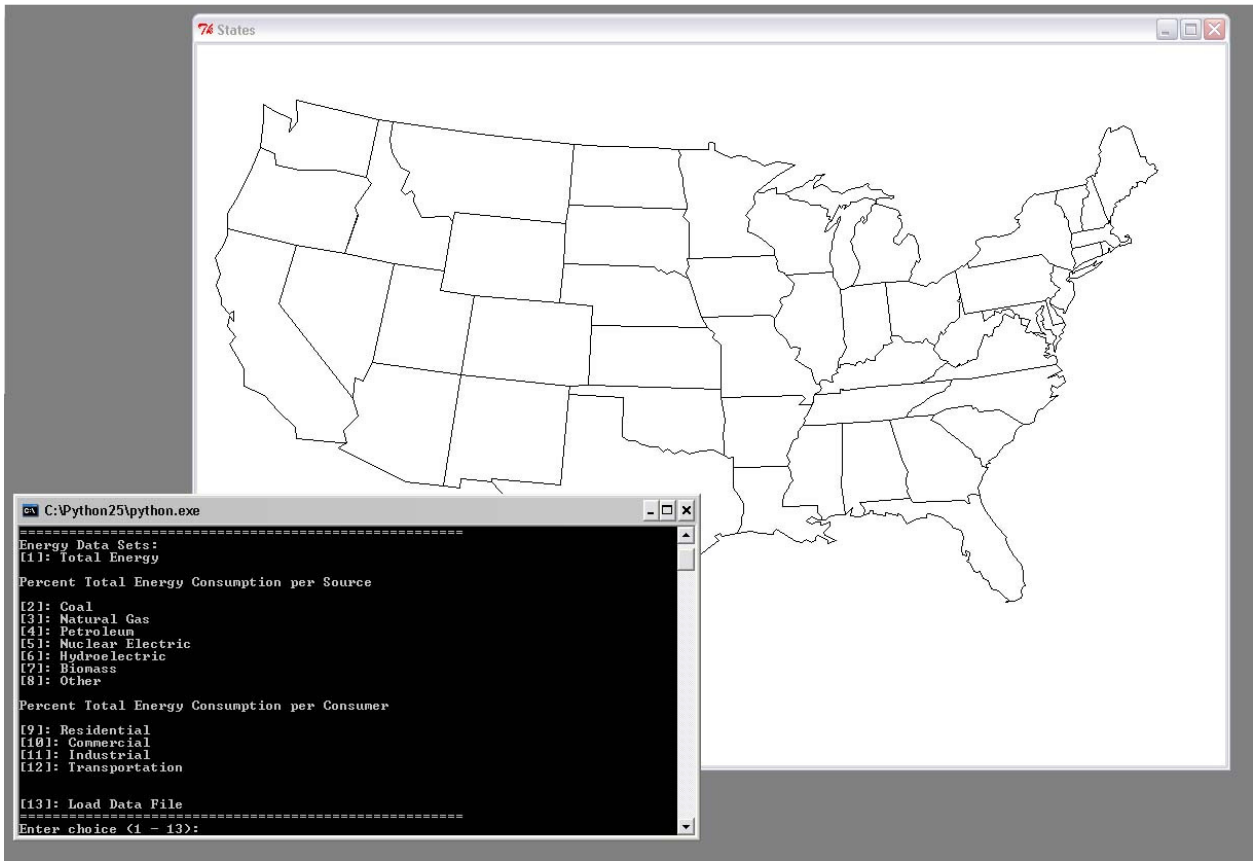
minimum a visualization of three-dimensional data (two dimensions defining the space, the third for the observation), it can be a challenge to convey the relationship between the spatial and observational data. One of the most effective ways to convey such information is to allow the spatial data itself to serve more than one function through the use of a color map.

A ***color map*** is an association of observed numerical data to a range of color values. You can think of it as an encoding scheme that maps numerical observations to a color according to some scaling function. To explore the use of color maps and the effectiveness of various types of color schemes, we are going to use a Python program (`colorMap.py`) to illustrate various data on a map of the contiguous 48 states of the US.

Do this: Download the files `colorMap.py`, `state.py`, `stateEnergy.txt`, and `statedata.txt` to your Python directory. The program also requires the `graphics.py` library. Two example data files are also provided: `election2004.txt` and `statePopulation.txt`, but are not necessary to run the program. The code for the mapping program itself is in the file `colorMap.py`. The file `state.py` defines a new object (State) in which a state's information (name, capital, and shape polygon from `statedata.txt`) can be stored. If you have been wondering how to create your own objects, check out the `state.py` file. The `stateEnergy.txt` file contains the information required for plotting the program's built-in data. **For this exercise, you actually do not need to modify any code so you can run the program simply by double clicking on the `colorMap.py` file to execute the script**. Certainly, I do encourage you to open up the `colorMap.py` file and read the code so that you understand what the program is doing. The actual function that generates the color map itself is called `colorMap`.

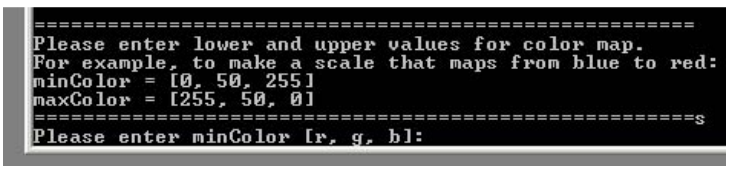
Using the `ColorMap` Program

When you execute the `colorMap` program, two windows will appear, a drawing window with a blank map of the US and a console window, waiting for some input:



When the `colorMap` software opens, it gives you two options. The first is to plot data belonging to a built-in dataset (summary statistics of the average energy consumption per state in the year 2005 source: http://www.eia.doe.gov/emeu/states/_seds.html). The second is to load other data from a file. To plot any of the built-in data, enter a number from 1-12 at the prompt. The menu (as illustrated above) lists which dataset is associated with each number. Enter 13 to load other data from a file. `colorMap` can plot both qualitative and quantitative data. Two example data files are provided (qualitative: the results of the 2004 presidential election `election2004.txt`; quantitative: state populations according to the 2000 census: `statePopulation.txt`).

After a dataset is successfully loaded, `colorMap` then prompts you to enter the RGB values that define the range of the color map:



The color mapping will range from the RGB color specified as the `minColor` to that specified by the `maxColor`. Enter RGB colors as a list of three numbers **`[<red>, <green>, <blue>]`**. The following table lists the RGB limits for generating some standard color mappings:

Mapping	colorMin	colorMax
Blue – Red	[0, 50, 255]	[255, 50, 0]
Cool – Warm	[0, 255, 0]	[255, 0, 0]
Grayscale	[0, 0, 0] (black)	[255, 255, 255] (white)
Tint	any color	[255, 255, 255]
Shade	any color	[0, 0, 0]

Exercise

Play with the `colorMap` program. Pick a dataset, either one of the built-in ones, the state population dataset, or one you assemble from other data found on the web. Do not use the election 2004 dataset; it is too simplistic.

If you choose to assemble your own data file, look at the sample datasets to assist you with the formatting. The file must be formatted correctly for the program to successfully read it. Alaska and Hawaii are not part of the contiguous 48 states, so do not include them in the file you create. Washington, DC is named the District of Columbia. Please provide me with the source for any data you choose to use (a url is fine).

1. Use the `colorMap` program to generate several color maps of your data. Try the basic color maps listed in the table above. How does the representation of the data change with the use of different mappings? Are there any mappings that distort the data? or create patterns/impressions that may not really exist? Are there some that are better than others? Pick a good a bad mapping and save by taking a screen snap shot (use Print Screen button) to submit. List the min and max colors used to establish each mapping.

2. Using either the same or a different dataset than used in #1, develop your own color maps. You can use the online RGB table or RGB calculator to assist in picking a range of colors. Try finding a mapping that uses a triadic color scheme. Then find a mapping that uses an analogous color scheme. How does the each mapping alter the presentation of the data? Again, save an example of each and list the min and max colors used to establish each mapping.