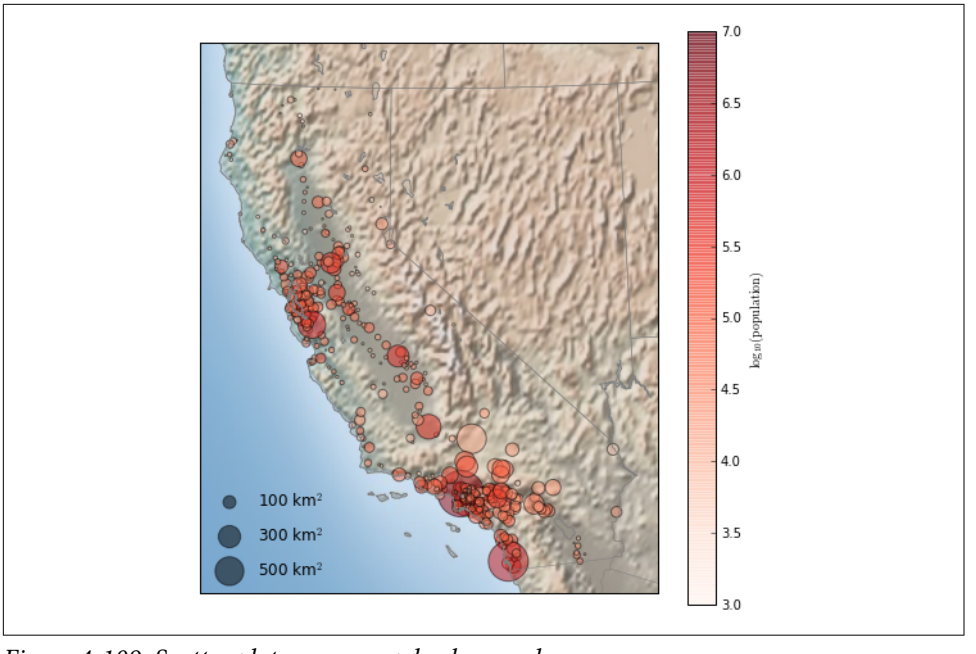

Figure 4-109. Scatter plot over a map background

This shows us roughly where larger populations of people have settled in California: they are clustered near the coast in the Los Angeles and San Francisco areas, stretched along the highways in the flat central valley, and avoiding almost completely the mountainous regions along the borders of the state.

Example: Surface Temperature Data

As an example of visualizing some more continuous geographic data, let's consider the "polar vortex" that hit the eastern half of the United States in January 2014. A great source for any sort of climatic data is NASA's Goddard Institute for Space Studies. Here we'll use the GIS 250 temperature data, which we can download using shell commands (these commands may have to be modified on Windows machines). The data used here was downloaded on 6/12/2016, and the file size is approximately 9 MB:

```
In[12]: # !curl -O http://data.giss.nasa.gov/pub/gistemp/gistemp250.nc.gz
        # !gunzip gistemp250.nc.gz
```

The data comes in NetCDF format, which can be read in Python by the `netCDF4` library. You can install this library as shown here:

```
$ conda install netcdf4
```

We read the data as follows:

```
In[13]: from netCDF4 import Dataset
        data = Dataset('gistemp250.nc')
```

The file contains many global temperature readings on a variety of dates; we need to select the index of the date we're interested in—in this case, January 15, 2014:

```
In[14]: from netCDF4 import date2index
        from datetime import datetime
        timeindex = date2index(datetime(2014, 1, 15),
                               data.variables['time'])
```

Now we can load the latitude and longitude data, as well as the temperature anomaly for this index:

```
In[15]: lat = data.variables['lat'][:]
        lon = data.variables['lon'][:]
        lon, lat = np.meshgrid(lon, lat)
        temp_anomaly = data.variables['tempanomaly'][timeindex]
```

Finally, we'll use the `pcolormesh()` method to draw a color mesh of the data. We'll look at North America, and use a shaded relief map in the background. Note that for this data we specifically chose a divergent colormap, which has a neutral color at zero and two contrasting colors at negative and positive values (Figure 4-110). We'll also lightly draw the coastlines over the colors for reference:

```
In[16]: fig = plt.figure(figsize=(10, 8))
        m = Basemap(projection='lcc', resolution='c',
                    width=8E6, height=8E6,
                    lat_0=45, lon_0=-100,)
        m.shadedrelief(scale=0.5)
        m.pcolormesh(lon, lat, temp_anomaly,
                     latlon=True, cmap='RdBu_r')
        plt.clim(-8, 8)
        m.drawcoastlines(color='lightgray')

        plt.title('January 2014 Temperature Anomaly')
        plt.colorbar(label='temperature anomaly (°C)');
```

The data paints a picture of the localized, extreme temperature anomalies that happened during that month. The eastern half of the United States was much colder than normal, while the western half and Alaska were much warmer. Regions with no recorded temperature show the map background.

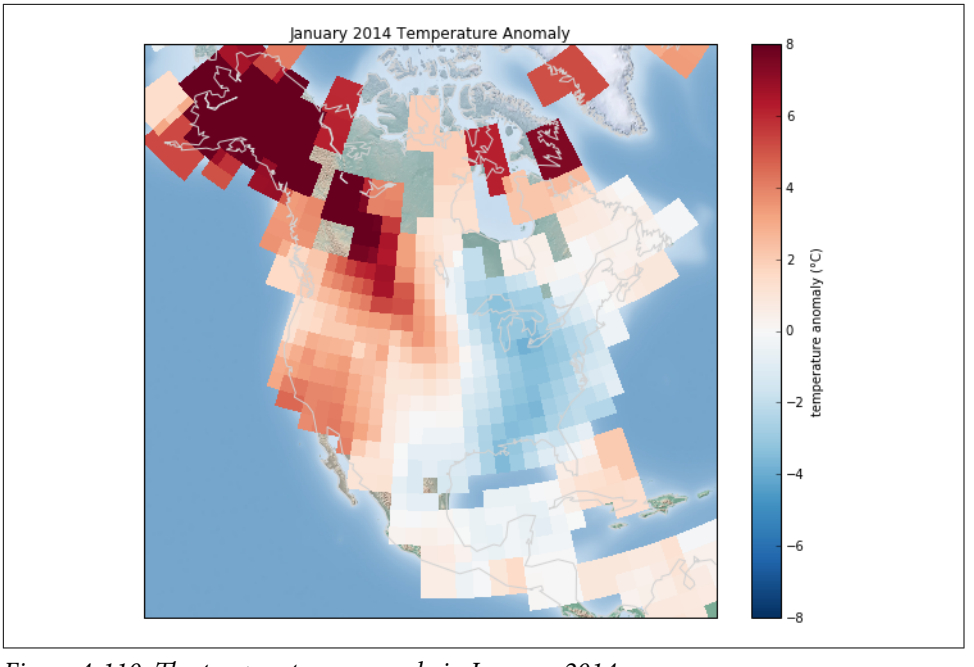

Figure 4-110. The temperature anomaly in January 2014

Visualization with Seaborn

Matplotlib has proven to be an incredibly useful and popular visualization tool, but even avid users will admit it often leaves much to be desired. There are several valid complaints about Matplotlib that often come up:

- Prior to version 2.0, Matplotlib's defaults are not exactly the best choices. It was based off of MATLAB circa 1999, and this often shows.
- Matplotlib's API is relatively low level. Doing sophisticated statistical visualization is possible, but often requires a *lot* of boilerplate code.
- Matplotlib predated Pandas by more than a decade, and thus is not designed for use with Pandas `DataFrames`. In order to visualize data from a Pandas `DataFrame`, you must extract each `Series` and often concatenate them together into the right format. It would be nicer to have a plotting library that can intelligently use the `DataFrame` labels in a plot.

An answer to these problems is Seaborn. Seaborn provides an API on top of Matplotlib that offers sane choices for plot style and color defaults, defines simple high-level functions for common statistical plot types, and integrates with the functionality provided by Pandas `DataFrames`.