CHAPTER 14

Plotting on maps

The matplotlib basemap toolkit is an add-on for matplotlib that provides the capability to draw maps of the earth in various map projections, and plot data on those maps. This section shows how to use basemap to create simple maps, draw coastlines and political boundaries, draw lines of constant latitude and longitude, and plot geophysical data on the maps.

1. Setting up the map.

In order to represent the curved surface of the earth in a two-dimensional map, a map projection is needed. Since this cannot be done without distortion, there are many map projections, each with it's own advantages and disadvantages. Basemap provides 19 different map projections. Some are global, some can only represent a portion of the globe. When a Basemap class instance is created, the desired map projection must be specified, along with information about the portion of the earth's surface that the map projection will describe. There are two basic ways of doing this. One is to provide the latitude and longitude values of each of the four corners of the rectangular map projection region. The other is to provide the lat/lon value of the center of the map projection region along with the width and height of the region in map projection coordinates. The first script illustrates how to use both of these methods to create a simple map. It also shows how to draw the continents and political boundaries on the map.

Here is an example script that creates a map by specifying the latitudes and longitudes of the four corners

LISTING 14.1

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
# create map by specifying lat/lon values at corners.
resolution = 'l'; projection = 'lcc'
lat_0 = 60; lon_0 = -50
llcrnrlat, llcrnrlon = 8, -92
urcrnrlat, urcrnrlon  = 39, 63
m = Basemap(lat_0=lat_0,lon_0=lon_0,\
            llcrnrlat=llcrnrlat,llcrnrlon=llcrnrlon,\
            urcrnrlat=urcrnrlat,urcrnrlon=urcrnrlon,\
            resolution=resolution,projection=projection)
# draw coastlines. Make liness a little thinner than default.
m.drawcoastlines(linewidth=0.5)
# background fill color will show ocean areas.
m.drawmapboundary(fill_color='aqua')
# fill continents, lakes within continents.
m.fillcontinents(color='coral',lake_color='aqua')
# draw states and countries.
m.drawcountries()
m.drawstates()
plt.title('map region specified using corner lat/lon values')
plt.show()
```

After running this script, you should see a plot that looks similar to Figure 1.

Here is an example script that creates a map by specifying the center of the map, plus the width and height in meters.

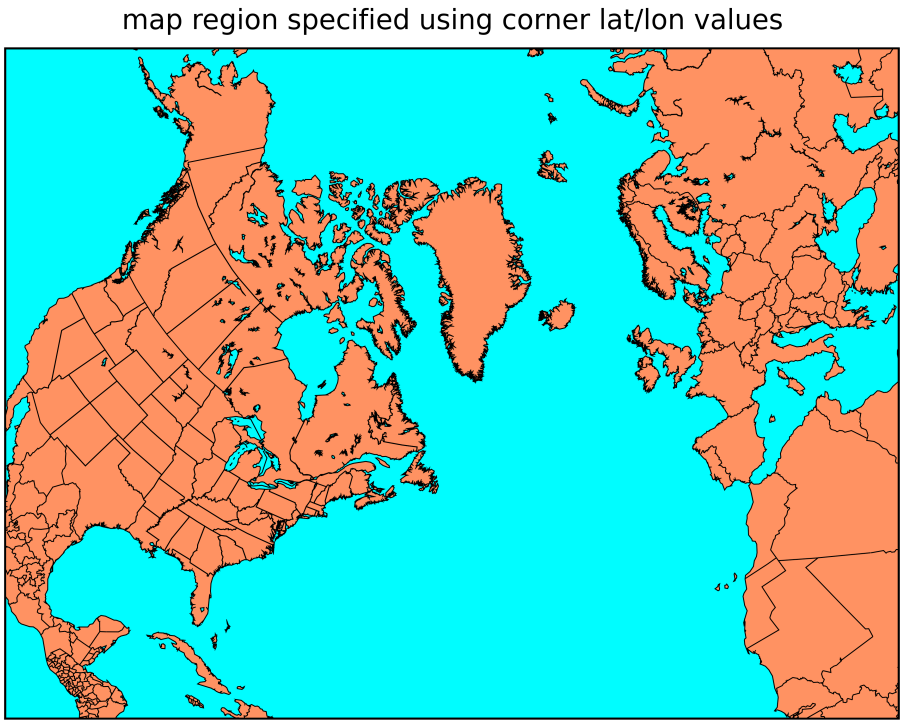

FIGURE 1. A map created by specifying the latitudes and longitudes of the four corners.

LISTING 14.2

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
# create map by specifying width and height in km.
resolution = 'l'; projection = 'lcc'
lon_0 = -50; lat_0 = 60
width = 12000000; height = 0.75*width
m = Basemap(lon_0=lon_0,lat_0=lat_0,\
            width=width,height=height,\
            resolution=resolution,projection=projection)
m.drawcoastlines(linewidth=0.5)
m.drawmapboundary(fill_color='aqua')
m.fillcontinents(color='coral',lake_color='aqua')
m.drawcountries()
m.drawstates()
plt.title('map region specified using width and height')
plt.show()
```

After running this script, you should see a plot that looks nearly identical to Figure 1.

The Basemap class instance can be used to convert latitudes and longitudes to coordinates on the map. To do this, simply call the instance as if it were a function, passing it the longitude and latitudes values to convert. The corresponding x and y values in map projection coordinates will be returned. The following example script shows how to use this to plot the locations of two cities (New York and London).

The Basemap method drawgreatcircle is then used to draw the great circle route between these cities on the map.

LISTING 14.3

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
# create map by specifying width and height in km.
resolution = 'l'; projection = 'lcc'
lon_0 = -50; lat_0 = 60
width = 12000000; height = 0.75*width
m = Basemap(lon_0=lon_0,lat_0=lat_0,width=width,height=height,
            resolution=resolution,projection=projection)
# nylat, nylon are lat/lon of New York
nylat = 40.78
nylon = -73.98
# lonlat, lonlon are lat/lon of London.
lonlat = 51.53
lonlon = 0.08
# convert these points to map projection coordinates
# (using __call__ method of Basemap instance)
ny_x, ny_y = m(nylon, nylat)
lon_x, lon_y = m(lonlon, lonlat)
# plot black dots at the two points.
# make sure dots are drawn on top of other plot elements (zorder=10)
m.scatter([ny_x,lon_x],[ny_y,lon_y],25,color='k',marker='o',zorder=10)
# connect the dots along a great circle.
m.drawgreatcircle(nylon,nylat,lonlon,lonlat,linewidth=2,color='k')
# put the names of the cities to the left of each dot, offset
# by a little. Use a bold font.
plt.text(ny_x-100000,ny_y+100000,'New York',fontsize=12,\
           color='k',horizontalalignment='right',fontweight='bold')
plt.text(lon_x-100000,lon_y+100000,'London',fontsize=12,\
           color='k',horizontalalignment='right',fontweight='bold')
m.drawcoastlines(linewidth=0.5)
m.drawmapboundary(fill_color='aqua')
m.fillcontinents(color='coral',lake_color='aqua')
m.drawcountries()
m.drawstates()
plt.title('NY to London Great Circle')
plt.show()
```

This should produce something similar to Figure 2.

Most maps include a graticule grid, a reference network of labelled latitude and longitude lines. Basemap does this with the drawparallels and drawmeridians instance methods. The longitude and latitude lines can be labelled where the intersect the map projection boundary. Following is an example script that draws a graticule on the map we've been working with.

LISTING 14.4

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np
# create map by specifying width and height in km.
resolution = 'l'
lon_0 = -50
lat_0 = 60
projection = 'lcc'
```

FIGURE 2. Drawing the locations of two cities, and connecting them along a great circle.

```
width = 12000000
height = 0.75*width
m = Basemap(lon_0=lon_0,lat_0=lat_0,width=width,height=height,
            resolution=resolution,projection=projection)
m.drawcoastlines(linewidth=0.5)
m.drawmapboundary(fill_color='aqua')
m.fillcontinents(color='coral',lake_color='aqua')
m.drawcountries()
m.drawstates()
# label meridians where they intersect the left, right and bottom
# of the plot frame.
m.drawmeridians(np.arange(-180,181,20),labels=[1,1,0,1])
# label parallels where they intersect the  left, right and top
# of the plot frame.
m.drawparallels(np.arange(-80,81,20),labels=[1,1,1,0])
plt.title('labelled meridians and parallels',y=1.075)
plt.show()
```

Running this script should produce a plot that looks like Figure 3.

2. Plotting geophysical data on the map.

One of the most common uses of Basemap is to visualize earth science data, such as output from climate models. These data often come on latitude/longitude grids. One common data format for storing such grids is NetCDF. Basemap includes a NetCDF file reader (written in pure python by

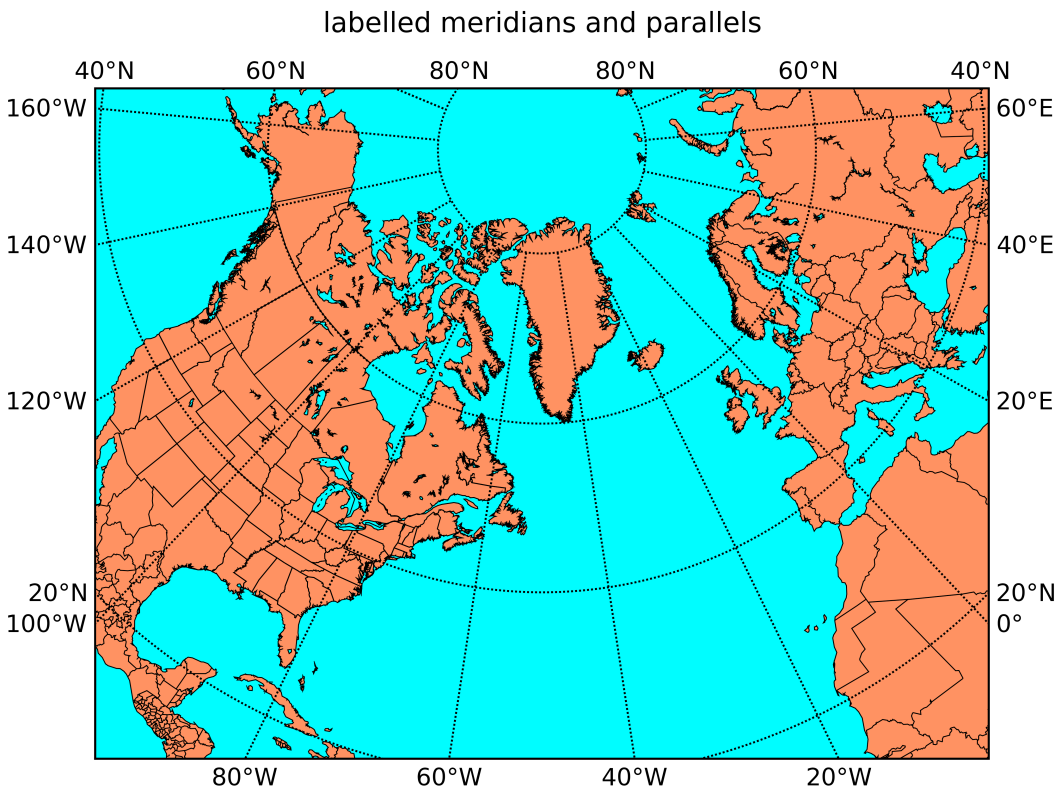

FIGURE 3. Drawing labelled meridians and parallels on the map (a graticule grid).

Roberto De Almeida). You can also access remote datasets over the web using the OPeNDAP protocol - just give the NetCDFFile function a URL instead of a local file name and Roberto's pydap module (`http://pydap.org`) will be used. The pydap client is included in Basemap. If the PyNIO module (`http://www.pyngl.ucar.edu/Nio.shtml`) is installed, the NetCDFFile function can also be used to open the formats that PyNIO supports, like GRIB and HDF. Following is an example of how to read sea-surface temperature data from a NetCDF file and plot it on a global mollweide projection.

LISTING 14.5

```
from mpl_toolkits.basemap import Basemap, NetCDFFile
import matplotlib.pyplot as plt
import numpy as np
# read in netCDF sea-surface temperature data
# can be a local file, a URL for a remote opendap dataset,
# or (if PyNIO is installed) a GRIB or HDF file.
ncfile = NetCDFFile('data/sst.nc')
sst = ncfile.variables['sst'][:]
lats = ncfile.variables['lat'][:]
lons = ncfile.variables['lon'][:]
# create Basemap instance for mollweide projection.
# coastlines not used, so resolution set to None to skip
# continent processing (this speeds things up a bit)
m = Basemap(projection='moll',lon_0=0,lat_0=0,resolution=None)
# compute map projection coordinates of grid.
x, y = m(*np.meshgrid(lons, lats))
```

```
# plot with pcolor
im = m.pcolormesh(x,y,sst,shading='flat',cmap=plt.cm.gist_ncar)
# draw parallels and meridians, but don't bother labelling them.
m.drawparallels(np.arange(-90.,120.,30.))
m.drawmeridians(np.arange(0.,420.,60.))
# draw line around map projection limb.
# color map region background black (missing values will be this color)
m.drawmapboundary(fill_color='k')
# draw horizontal colorbar.
plt.colorbar(orientation='horizontal')
plt.show()
```

The resulting plot should look like Figure 4.

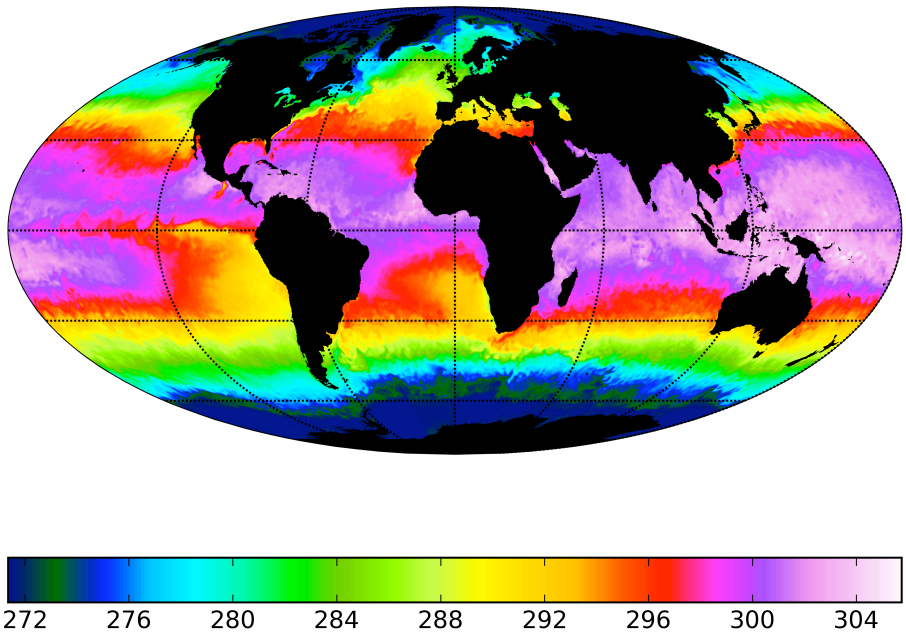

FIGURE 4. Sea surface temperature on a global mollweide projection.

Basemap also is capable of reading ESRI shapefiles, a very common GIS format. The script fillstates.py in the examples directory of the basemap source distribution shows how to read and plot polygons in a shapefile. There are many other useful examples in that directory that illustrate various ways of using basemap.