ESCI 386 – Scientific Programming, Analysis and Visualization with Python

Lesson 15 - Basemap

About Basemap

- Basemap is a matplotlib toolkit for plotting data on geographically-referenced map projections.

- The documentation for basemap can be found at http://matplotlib.github.com/basemap

Using Basemap

- The first step to using basemap is to import it using the command

```
from mpl_toolkits.basemap import Basemap
```

- We then create a map object by typing

```
m = Basemap(projection = proj_type, kwds/args)
```

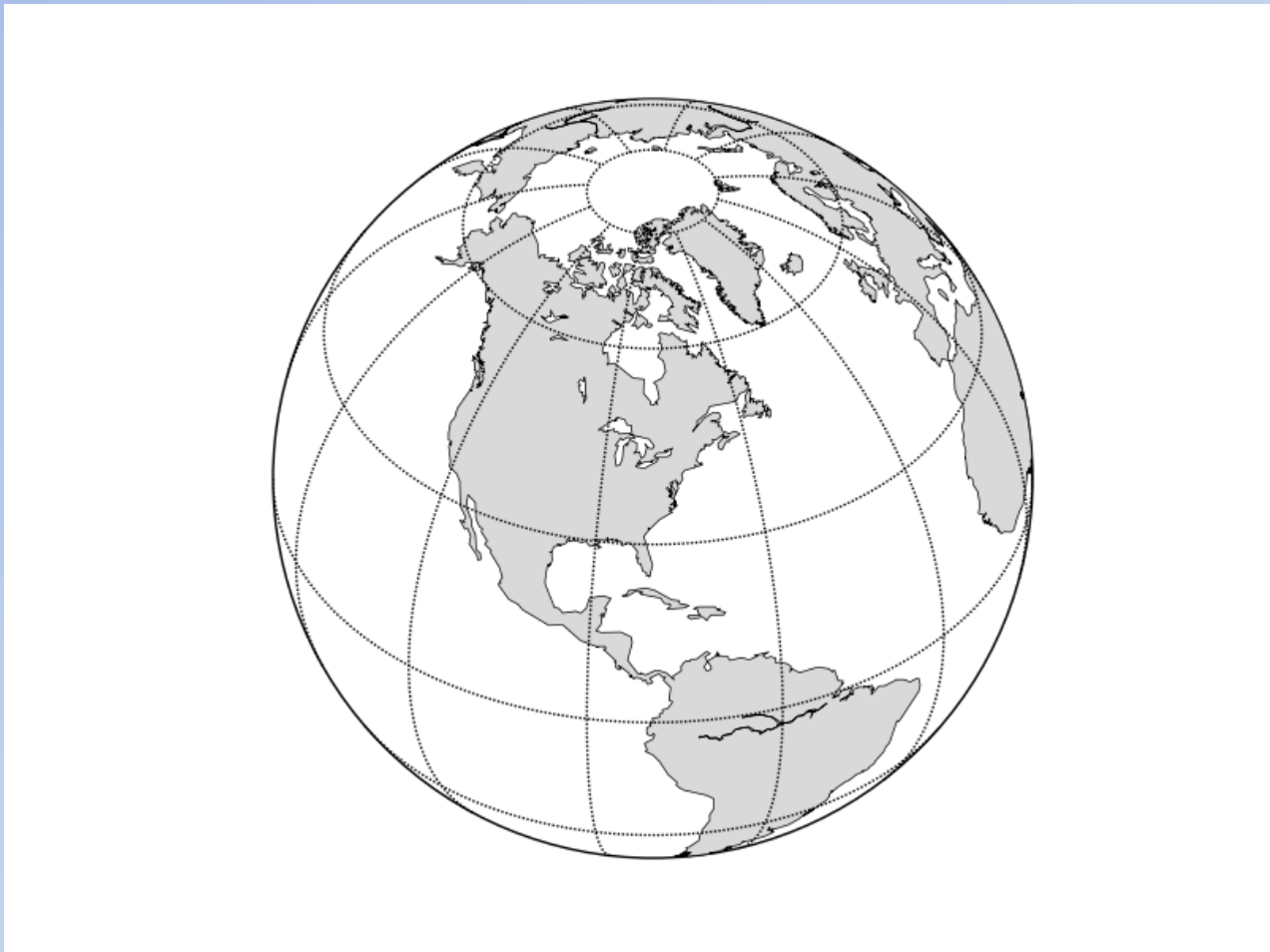
  - where *proj_type* is a valid projection type, and *kwds/args* are additional keywords and arguments which may depend on the projection chosen.

Example – Orthographic Projection

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np
m = Basemap(projection = 'ortho',lat_0 = 40, lon_0 = -80)
m.drawmapboundary(fill_color = 'white')
m.drawcoastlines(color = 'black', linewidth = 0.5)
m.fillcontinents(color = '0.85')
m.drawparallels(np.arange(-90, 91,30))
m.drawmeridians(np.arange(-180,180,30))
plt.show()
```

File: orthographic-example.py

Orthographic Projection Results



Available Projections

Azimuthal Equidistant	`'aeqd'`
Polyconic	`'poly'`
Gnomonic	`'gnom'`
Mollweide	`'moll'`
Transverse Mercator	`'tmerc'`
North-Polar Lambert Azimuthal	`'nplaea'`
Gall Stereographic Cylindrical	`'gall'`
Miller Cylindrical	`'mill'`
Mercator	`'merc'`
Sterographic	`'stere'`
North-Polar Stereographic	`'npstere'`
Cylindrial Equidistant	`'cyl'`
Oblique Mercator	`'omerc'`
Albers Equal Area	`'aea'`
South-Polar Azimuthal Equidistant	`'spaeqd'`

Hammer	`'hammer'`
Geostationary	`'geos'`
Near-Sided Perspective	`'nsper'`
van der Grinten	`'vandg'`
Lambert Azimuthal Equal Area	`'laea'`
McBryde-Thomas Flat-Polar Quartic	`'mbtfpq'`
Sinusoidal	`'sinu'`
South-Polar Stereographic	`'spstere'`
Lambert Conformal	`'lcc'`
North-Polar Azimuthal Equidistant	`'npaeqd'`
Equidistant Conic	`'eqdc'`
Orthographic	`'ortho'`
Cassini-Soldner	`'cass'`
South-Polar Lambert Azimuthal	`'splaea'`
Robinson	`'robin'`

Specifying Map Region

- Depending on the map projection the region is specified by either the following keywords:
  - `lon_0` – center longitude (degrees)
  - `lat_0` – center latitude (degrees)
  - `width` – width of domain (meters)
  - `height` - height of domain (meters)
- or
  - `llcrnrlon` – lower-left corner longitude (degrees)
  - `llcrnrlat` – lower-left corner latitude (degrees)
  - `urcrnrlon` – upper-right corner longitude (degrees)
  - `urcrnrlat` – upper-right corner latitude (degrees)

Some Additional Basemap Keywords

Keyword	Values	Purpose
`resolution`	`c`, `l`, `i`, `h`, or `f`	Resolution of the database for continents, lakes, etc. Initials stand for crude, low, intermediate, high, and full. The default is crude.
`area_thresh`	number representing square kilometers	Will not draw lakes or coastal features that have an area smaller than this threshold
`rsphere`	radius of the globe in meters	Defaults to 6370997. Can be changed, or even given as major and minor axes for plotting on an ellipsoid.

Methods for Drawing Features

- There are many methods for plotting coastlines, continents, rivers, etc.

- For many of the methods the `linewidth` and `color` can be specified, but not the line style. Solid is often the only option for line styles.

Methods for Drawing Features (cont.)

- `drawcoastlines()` – Draws coastlines.
- `drawcountries()` – Draws country boundaries.
- `drawgreatcircle(`*`lon1, lat1, lon2, lat2, del_s = f`*`)` – Draws a greatcircle between two lat/lon pairs. *`del_s`* is the spacing (in km) between points.
- `drawmapboundary()` – Draws boundary around map projection. The fill color is specified with the keyword `fill_color`.

Methods for Drawing Features (cont.)

- `drawmapscale(`*`lon, lat, lon0, lat0, length`*`)`

- Draws a scale at the position given by *`lon, lat`*. The distance is for the position of *`lon0, lat0`*.
  - Additional keywords are:
  - `units` – The units for the scale (`'km'` is default)
  - `barstyle` – `'simple'` or `'fancy'`
  - `fontsize` – default is 9
  - `color` – default is `'black'`
  - `labelstype` – `'simple'` or `'fancy'`
  - `format` – a string format statement of the type `'%3.1f'` and such.
  - `yoffset` – controls the scale height and annotation placement.
  - `fillcolor1, fillcolor2` – controls colors for alternating regions of scale for `'fancy'` style.

Methods for Drawing Features (cont.)

- `drawmeridians(`*`mlist`*`)`

- Draws meridians with values given by *`mlist`*. In addition to `color` and `linewidth`, additional keywords are:
  - `dashes` – Pattern for dashed lines, of the form `[on,off]` where on is the number of adjacent pixels turned on, while off is the number that are turned off. The default is `[1,1]`.
  - `labels` – Four values in the form [left, right, top, bottom] that control meridian labeling. These are Boolean in that 1 is on and 0 is off. So, `[1,0,1,0]` would label the meridians at the left and top of the plot.
  - `labelstyle` – Controls whether labels use `+/-` or `E/W`. The default is `E/W` unless `labelstyle='+/-'`.
  - `fmt` – This formats the labels using the format statements of the type `'%3.1f'` and such.
  - `xoffset, yoffset` – Label offsets from edge of map.
  - `latmax` – Controls maximum latitude for drawing meridians (default is 80).

Methods for Drawing Features (cont.)

- `drawparallels(plist)` – Draws latitude parallels with values given by mlist. Keywords are essentially the same as for `drawmeridians()`.
- `drawrivers()` – Draws rivers on map.
- `drawstates()` – Draws state boundaries.
- `fillcontinents(color = 'brown', lake_color = 'blue')` – Fills continents and lakes with specified colors.

Basemap Can Read GIS Shapefiles

- There is also a method for reading a GIS shapefile.
- This method is called `readshapefile()`.
- Details of its use can be found in the online documentation.

Plotting Data on Maps

- Data can be plotted on maps by using the `contour()`, `contourf()`, `plot()`, `quiver()`, `barbs()`, and `drawgreatcircle()` methods for map objects.
  - These methods work pretty much just like the corresponding axes methods, with some exceptions.

Plotting Data on Maps

- When plotting on the maps the latitudes and longitudes have to be converted into map coordinates.

- In the examples below, `m` is the map object created by `Basemap()`

- To convert lat/lon to map coordinates
  - `x,y = m(lon,lat)`

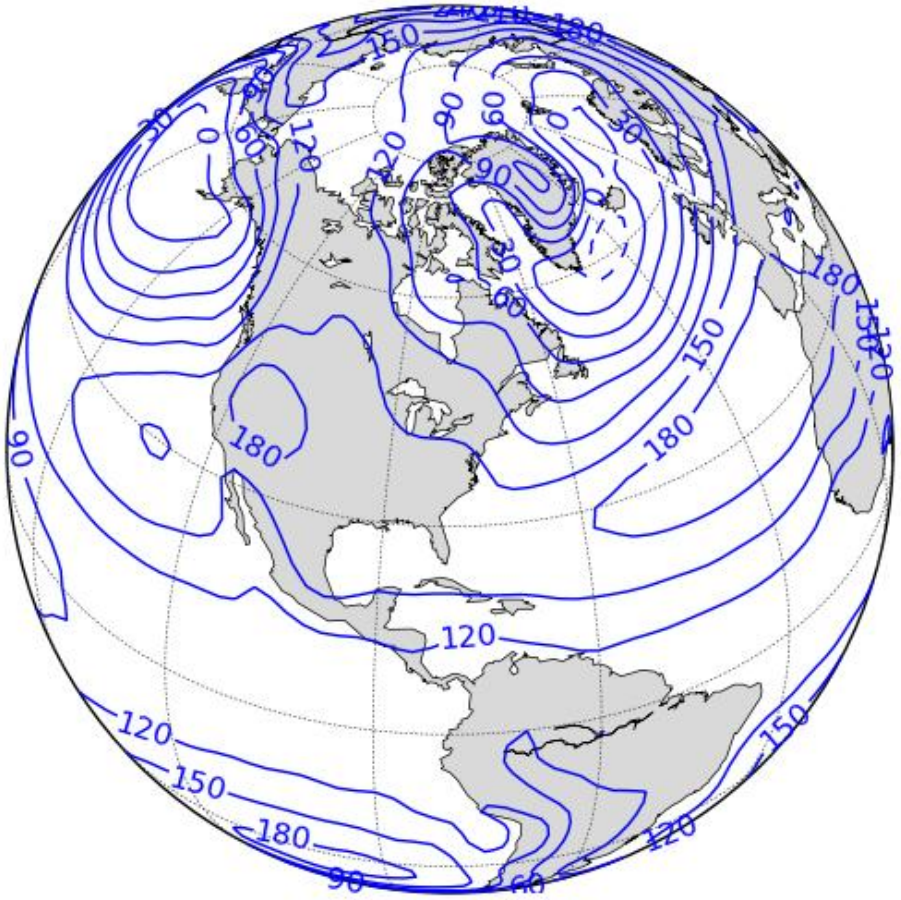
- To convert map coordinates to lat/long.
  - `lon, lat = m(x, y, inverse = True)`

Example

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np
m = Basemap(projection = 'ortho',lat_0 = 40, lon_0 = -80)
m.drawmapboundary(fill_color = 'white')
m.drawcoastlines(color = 'black',linewidth = 0.5)
m.fillcontinents(color = '0.85')
m.drawparallels(np.arange(-90, 91,30), color = '0.25', linewidth = 0.5)
m.drawmeridians(np.arange(-180,180,30), color = '0.25', linewidth = 0.5)
data = np.load('jan1000mb.npz')
lon = data['lon']
lat = data['lat']
z = data['z']
x,y = m(lon,lat)
cs = m.contour(x,y,z, levels = range(-180,360,30),colors = 'blue')
plt.clabel(cs, fmt = '%.0f', inline = True)
plt.show()
```

File: ortho-1000mb.py

Result



Play Time

- Play around with different projections and different types of plots (filled contour, etc.)

Creating 2-D arrays of x and y values from 1-D arrays.

- Often we need 2-D arrays of x and y coordinate values.
- We can create them from 1-D arrays of x and y using the numpy meshgrid() function.

```
>>> x
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> y
array([-3, -2, -1,  0,  1,  2,  3])
>>> x2,y2 = np.meshgrid(x,y)
>>> x2
array([[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
       [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
       [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
       [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
       [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
       [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

```
>>> y2
array([[-3, -3, -3, -3, -3, -3, -3, -3, -3, -3],
       [-2, -2, -2, -2, -2, -2, -2, -2, -2, -2],
       [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 1,  1,  1,  1,  1,  1,  1,  1,  1,  1],
       [ 2,  2,  2,  2,  2,  2,  2,  2,  2,  2],
       [ 3,  3,  3,  3,  3,  3,  3,  3,  3,  3]])
```

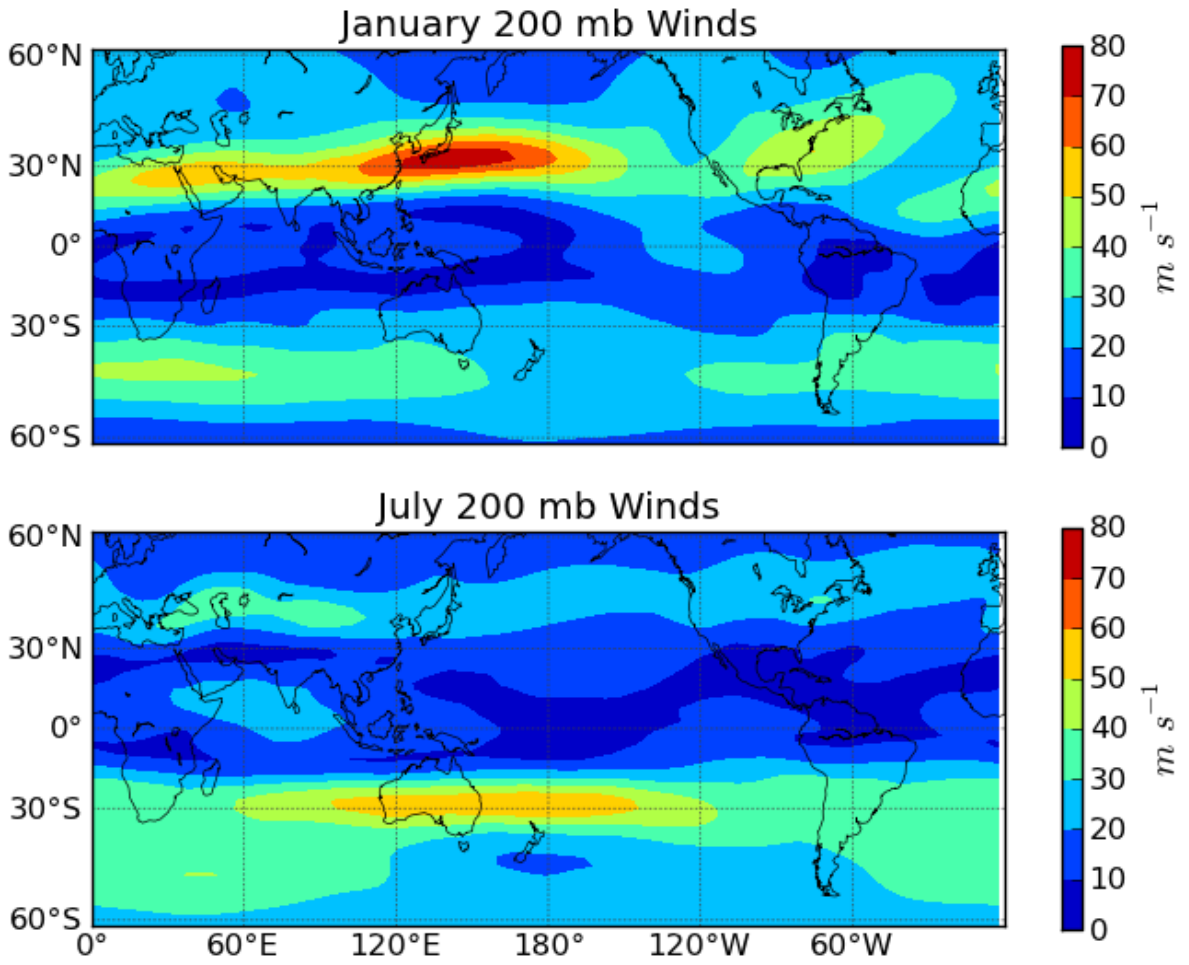
Exercise 6

- Using the NetCDF file

  `'wspd.mon.1968-1996.ltm.nc'`

  see if you can recreate the plot on the next slide.

Exercise 6 (cont.)



Using Cyclic Boundaries to Get Rid of White Stripe at Edge

- Notice the white strip at the right edge of the previous map.
- The data are defined on longitudes from 0 to 357.5 E, so it leaves a white stripe.
- We can use the addcyclic() function from basemap toolkits to create an additional column in the array that is the same as the very first column.

```
from mpl_toolkits.basemap import Basemap, addcyclic
.
.
wspd_July2, lons2 = addcyclic(wspd_July, lons)
.
.
```

Add extra column at end of the arrays in the argument list.