Synoptic Meteorology I: Assignment #8

Due: 11 December 2018

Learning Objective: to learn how to use Python tools to obtain observed upper-air data and create a skew-T/ln-p diagram from those data after minimal processing.

Unidata's *metpy* and *siphon* packages provide routines by which observed upper-air data (from the University of Wyoming) can be downloaded in a Python-friendly format, then used to plot a skew-T diagram. This assignment steps you through the code used to both acquire and plot the data, then asks you to create a plot of your own, print it, and analyze it to determine selected parameters.

(**Note**: I recommend placing all of the relevant commands in a Python script rather than type them one-by-one directly into the Python command window. Save it to your Desktop or home directory before running the code so the image it produces will be saved to your Desktop or home directory. It will help you understand the code if you add comments between individual code blocks; each comment line in a Python script is prefaced with a #.)

As is usual, we need to first load the necessary modules and functions. There are five modules that we draw modules and functions from for this task: metpy, matplotlib, numpy, datetime, and siphon, with a total of seven import statements.

```
from metpy.plots import SkewT
from metpy.units import units
import metpy.calc as mpcalc
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime
from siphon.simplewebservice.wyoming import WyomingUpperAir
```

You'll note that in the first two and last two cases, we import the needed functions using statements of the form "from _____ import _____." This allows us to import only a single function – which we can then refer to without any prefixes – from the associated module.

Next, we define two variables that are used to specify the date, time, and location of our sounding. For example, the following code would obtain the 5 August 2018 0000 UTC Green Bay, WI (GRB) sounding:

```
date = datetime(2018, 8, 5, 0)
station = 'GRB'
```

The first line represents the only use of the datetime module in this script. Together, these lines provide all of the information that the siphon WyomingUpperAir function needs to obtain our data. When it does so, it returns it as a pandas dataframe, which is a powerful tool for working with any dataset (atmospheric datasets included!).

```
df = WyomingUpperAir.request_data(date, station)
```

A pandas dataframe can be thought of as akin to a spreadsheet, with different variables in different columns and different pressure levels in different rows. Feel free to print the dataframe out to the screen to get a glimpse of the data. However, we need to parse the individual variables for metpy to have the data it needs to create the skew-T diagram. Fortunately, pandas provides us with a way of doing so: we simply need to give it the column key and call the `values` attribute. While we do so, we can also use metpy's `units` function to append units to the data:

```
pressure = df['pressure'].values * units('hPa')
temperature = df['temperature'].values * units('degC')
dewpoint = df['dewpoint'].values * units('degC')
u = df['u_wind'].values * units('knot')
v = df['v_wind'].values * units('knot')
```

Note how the column keys correspond to those in the dataframe `df`. If we didn't have the ability to examine the dataframe to see these keys for ourselves, we could use `print(df.columns)` to print the column keys to the screen.

Next, we set up the parameters for our skew-T diagram. This involves determining the dimensions of the figure, creating the Python plotting object that serves as the building block for the skew-T, setting the *x*- and *y*-axis values, and telling Python to plot tick marks on both the *x*- and *y*-axes.

```
fig = plt.figure(figsize=(12, 12))
skew = SkewT(fig)
skew.ax.set_xlim(-60.,40.)
skew.ax.set_ylim(1000.,100.)
skew.ax.tick_params(axis='both',labelsize=14)
```

The above code creates a 12" x 12" figure. The isotherms range from -60°C to +40°C at the lowest pressure level, which is set to 1000 hPa. The skew-T diagram is truncated at the top at 100 hPa.

Although metpy is capable of drawing isotherms, adiabats, and mixing ratio lines without any user inputs, the spacing between adjacent lines tends to be smaller than desired. Fortunately, for all but the isotherms, it is easy to tell metpy how frequently to draw the dry adiabats, saturated adiabats, and mixing ratio lines. We also want to give metpy the range of vertical levels over which to draw the mixing ratio lines. For these tasks, we define four numpy arrays:

```
dads = np.arange(-60., 240., 2.)*units.degC
mads = np.arange(-60., 60., 2.)*units.degC
mrats = np.concatenate((np.arange(0.00001,0.0001,0.00001),
                        np.arange(0.0001, 0.001,0.0001),
                        np.arange(0.001,0.050,0.001))).reshape(-1, 1)
mrps = np.linspace(600.,1000.)*units.hPa
```

For `dads` (dry adiabats) and `mads` (moist adiabats), the arrays define surface temperatures from which to draw adiabats upward. In the examples given here, the numpy `arange` function is used to creates arrays from -60°C to 238°C (dry adiabats) and -60°C to 58°C (moist adiabats) with a 2°C interval for both arrays. Here, note that `arange` treats the last number (240°C or 60°C) as

exclusive, or not included, when evaluating the function. We use a large value on the high end of the range for the dry adiabats so that they fill the entire plot with dry adiabats (they arc up and to the left, but lines are drawn on the plot relative to the *x*-axis values at the *bottom* of the plot).

For `mrats` (mixing ratio lines, specified in kg kg^{-1}), the numpy `concatenate` function is used to merge three arrays together. The numpy `reshape` function is then used to change the resulting array from one with one row and many columns to one with one column and many rows. No units are appended to this variable because these values are in kg kg^{-1}, and thus are dimensionless. For `mrps` (range of pressure levels over which to draw mixing ratio lines), the numpy `linspace` function is used. The `linspace` default is to return a 50-element array, with values evenly spaced between the start (here, 600 hPa) and end (here, 1000 hPa) values provided to the function.

Now that we have established our plot parameters, we can reference the `skew` variable to plot the desired fields:

```
skew.plot(pressure, temperature, 'r', linewidth=4)
skew.plot(pressure, dewpoint, 'g', linewidth=4)
skew.plot_barbs(pressure, u, v)
skew.plot_dry_adiabats(dads, linestyles='solid', linewidth=1)
skew.plot_moist_adiabats(mads, linestyles='solid', linewidth=1)
skew.plot_mixing_lines(mrats, mrps, linewidth=1)
```

The metpy `skew` instance has functions named `plot` to plot vertical profiles of specific quantities, here for temperature and dew point temperature. The first argument is the vertical coordinate (here, pressure), the second argument is the variable to plot, and the remaining arguments are matplotlib *keyword arguments*. Here, only two such arguments are given: one to control the line color ('r' = red, 'g' = green) and another to control the line width in pixels. There are also separate functions to plot wind barbs, dry and moist adiabats, and mixing lines. The dry and moist adiabat plotting functions include another keyword argument, `linestyles`, that controls whether the plotted lines are solid, dashed, dotted, and so on.

We just need to add some labels, print the file to an image, and close things out and we're all set:

```
skew.ax.set_xlabel("Temperature (°C)", fontsize=16)
skew.ax.set_ylabel("Pressure (hPa)", fontsize=16)
title = station + " " + str(date)
fig.suptitle(title, fontsize=18)
plt.savefig("skew.png", bbox_inches='tight')
plt.close(fig)
```

The first two commands label the *x*- and *y*-axes using the `fontsize` keyword argument to control the font size used to plot these labels. The third line creates a new string variable called `title` to append the station ID and observation time and date into a single string. This variable is then used to title the plot. The figure is then saved to a file called skew.png in the same directory as the script. The `bbox_inches='tight'` keyword argument is used to reduce the white space around the figure. Finally, the figure instance that the skew-T diagram is drawn into is closed.

For this assignment, you are to create a skew-T for the time/date and location listed below. Use an 8.5" x 11" figure size, a dashed line style for the mixing ratio lines, and add your name to the figure title. Once this image has been created, print it. For an air parcel lifted from the surface, identify the potential temperature, wet bulb temperature, and equivalent potential temperature. Darken the process lines used to find each, preferably with a different color for each quantity. Turn your final skew-T diagram in by the start of class on the assignment due date.

Sounding Locations and Times

- **Sara**: 4 May 1999, 0000 UTC, Norman, OK (OUN)
- **Devon**: 19 May 2013, 1200 UTC. Norman, OK (OUN)
- **Austin**: 29 September 2012, 1200 UTC, Minneapolis/St. Paul, MN (MPX)
- **Hannah**: 14 January 2004, 1200 UTC, Wilmington, OH (ILN)
- **Alex**: 29 May 2008, 0000 UTC, Denver, CO (DNR)
- **Giorgio**: 27 October 2010, 0000 UTC, Green Bay, WI (GRB)
- **Ashley**: 11 May 2008, 0000 UTC, Birmingham, AL (BMX)