WRF Forecasted Frost Likelihood:

The script entitled Frost\_Kenya.py was created to adapt a previous frost algorithm to ingest forecast output from the Weather and Research Forecast (WRF) model. The original frost detection and warning algorithm, created by Ceres Albers, was developed for a wireless sensor network and was designed to run in real-time as the sensors reported weather conditions. Developed in Excel, the algorithm checks the reporting sensors against thresholds of temperature, humidity and wind speed to determine the likelihood of frost. There are 15 warning conditions that are triggered by separate thresholds that range from ‘frost conditions may be likely soon’ to ‘severely damaging frost likely’. The original methodology and full documentation can be found in the document: FROST FORECASTING USING THE WIRELESS SENSOR NETWORK FOR HUNTSVILLE ALABAMA. Please refer to this document for the assumptions and caveats relative to applicability.

In general, the algorithm was kept intact in the python scripting transfer. However, in ingesting a 48 forecast, certain forecasted warnings based on ‘current’ data were omitted. For example the original algorithm predicted the likely hood of future frost with the rate of temperature decline after sunset. Now we simply run the forecast through the frost period to determine whether frost conditions occur.

Below is an overview of the script itself that goes into a little more detail that the comments. Note that line numbers are provided for the initial inspection, however, as soon as edits are made they will no longer be relevant.

The script is created in python in a linux environment. There are several dependent modules that are required to run this script. Dependencies:

* Numpy
* Pandas
* Datetime
* Gdal
* Urllib2
* Matplotlib (only if you want to plot in python)

These can easily be installed using the *pip install* function. (I recommend this method as it will install individual dependencies)

**Definitions (lines 33-86)**

The script begins with a few GDAL definitions that will be used to capture the geographical information so the output tiff can be geo-referenced. Also there is a urllib2 definition that essentially acts like a wget function. These should not be edited.

**Set Working Directories and Global Variables \*\*YOUR INPUT REQUIRED\*\* (lines 90-113)**

Here you set the local working directories and variables. You will need to change these directories (~lines 94-96). These will store your temporary files, the intermediate weather files and your output. If you choose to not save the raw weather files (these can add up pretty quickly, 48\*3), lines 160 -162 should be edited, you can remove the date tag so they can be rewritten each time.

Set your GDAL exe (line 106)

The date variable is set using a datetime function; this will be called to identify file names (line 110)

**Functions begin (line 112)**

The body of the script begins by downloading a sample grib2 file from the ftp and captures its geo information. Next, two different arrays are set; one to hold the numerical indices (0-5), the other to hold each frost condition (alpha through omicron, from the original methodology).

The script loops through each of the 48 forecast hours:

Downloads the forecast grib2 file (this is overwritten each time)(line 142)

Extracts and saves the weather data as tiffs (saved in a local directory specified above). (line 150)

If you do not want to save all the weather files (temperature, windspeed, and relative humidity for each hour) simply remove the date tag for the weather files so they can be rewritten. (lines 153-155)

Gets the data for each variable as an array(line 162)

Then loops through each array element (pixel) and determines whether or not a frost condition is met. (line 188)

For simplicity, the script groups the 15 conditions into 5 different indices ranging from 0: no frost to 5: severe frost. Each condition from the original methodology is also retained in the condList array if needed. First general conditions are tested (i.e. like if RH is below 50 or is wind speeds are high). If the input does not meet these criteria, the loop is aborted and moves onto the next pixel as frost will not be likely.

Lastly, once all pixels have been processed for each forecast hour the two arrays will hold the entirety of the information (3d: hr, x, y). The script currently splits the index array and gets the max index that occurs each day and saves it out as a tiff in the output directory specified above. However, this can easily be edited to dump a tiff for each forecast hour if desired. (line 317)

A plotting routine is at the bottom for convenience, it’s currently commented out but can be used to plot the two tiffs created above in a simple python basemap if desired.