**Upgrading StickNet Code**

**A Review of Aaron Hill’s vortexse repository**

**Python scripts**

**Probe\_info.py**

* SUPER IMPORTANT
* Contains manually filled dictionaries with each StickNet ID number and lat, lon, elevation, 4-digit location identifier, and location name.

**SNobject.py**

* Creates snfile class
* Functions:
  + Read\_realtime
  + Read\_daq (stesonet or finescale)
  + Read\_daq\_qced (stesonet or finescale)

**SNmods.py**

* Hard coded probe info probe\_locs
* Function: search\_dir
  + If dataset=”full”, tries to make meteograms for each sticknet
  + If dataset=”latest”, takes the most recent data from each sticknet file (has capability to make sure that the data is less than 5 minutes old, but that code is not active) and can write it to html to be viewed on the webpage.
  + Outside of full or latest designation, the data from latest is output in a dataframe. If full is called, I think this function may actually fail??

**Bolton.py**

* Contains many functions for thermodynamic calculations, especially those used in thetav and thetae. May replace with metpy?

**Functions.py**

* Contains many functions for wind/thermo calculations
* Contains the plot\_meteogram function that will need updating

**Functions\_radar.py**

* Contains the get\_radar\_from\_aws function. This function requires upgrading if you want to use it on a windows machine.

**Oban.py**

* Creates objective analysis plots of thetae, thetav, dewpoint, temperature, and MSLP
* Needs to be updated, uses basemap and outdated interpolation techniques

**Rt\_NEXTRAD\_obs.py**

* Creates radar and StickNet data plots. Makes obs plots with reflectivity, velocity, reflectivity and oban thetae, and reflectivity and oban thetav.
* Needs to be updated, uses basemap and outdated interpolation techniques
* Perhaps make plots both ref and vel? Side by side?

**Rt\_meteograms.py**

* Calls SNmods.search\_dir with dataset=full, which creates meteograms
* Needs to be updated to add thetav gradients

**Rt\_grid.py**

* Calls SNmods.search\_dir and gives it an html file (that is created in script). This creates the obs plot and writes the current obs into the html file which is read into the website (the obs grid).
* Needs to be updated, uses basemap

**Rt\_grid\_rapid.py**

* Same as rt\_grid.py, except for the rapid probes. The biggest difference seems to be the specified numbers given in “probes” for the search\_dir function. This may be unnecessary for future projects.
* Needs to be updated, uses basemap

**Bash scripts**

**5min\_update\_meteograms.sh**

* Uses a cronjob to create meteogram plots every 5 minutes and copy the plots to be published on the webpage
* Calls rt\_meteograms.py

**5min\_update\_oban.sh**

* Same as the 5min\_update\_meteogram script, but for oban plots
* Calls oban.py

**10min\_update.sh**

* Creates the radar plots every 10 minutes and copies it to website
* calls rt\_NEXRAD\_obs.py

**1min\_update.sh**

* runs every minute to “create plots, html scripts, etc”
* calls rt\_grid.py

cleanup.sh

* copies realtime files to a storage location

move\_files\_hd.sh

* the same as cleanup.sh?

move\_files\_period\_hd.sh

* same as move\_files\_hd.sh, except that you specify dates (can be a range, you give date and endate)

save\_files.sh

* move recent meteogram plots, surface plots, and oban fields to folders named after the date

Locations of hard coded things!

Functions.py -> plot\_meteogram(). Need to hardcode the save directory

SNmods.py -> probe\_locs., need to hardcode the year for proper sn location (THIS CAN EASILY BE MODIFIED)

Probe\_info.py -> dictionary that contains proper sticknet id, lat, lon, elevation, 4-letter identifier, and location.

Oban.py -> (incomplete), needs save directory

Rt\_nexrad\_sticket etc… has time zone location hardcoded

\*\* future plan, like probe\_info.py, create a dictionary that contains all the needed directories that the other scripts then read. That way, you won’t have to manually change anything besides the one script! This would make things really easy…

Features:

All geographical plots will automatically size themselves based on the number and location of sticknets that get\_sticknet\_data can find. The map scale updates based on the width of the plot.

The appropriate radar is automatically chosen based on the centroid of all sticknets.

The titles for the meteograms plots update based on how many hours of data they are given.

Oban ONLY plots if there are 4 or more sticknets available (interp will fail otherwise). Note: lower numbers of sticknets will likely produce not the best interpolations.

All other plots will plot for 1 or more available sticknet.

NOTE: info.py always needs to have every sticknet that you’re expecting to be available. Get\_sticknet\_data is only given the probes that are listed in this file.