**Start time: 7:44 pm**

Notes:

UPshape.AC\_ID: assembly constituency identifier

UPshape.C\_CODE01: unique village code identifier (from 2001 Census)

UPshape.LONGITUDE

UPshape.LATITUDE

satellite: name of DMSP-OLS satellite

month

day

timestamp: time of orbit, UTC

cm: cloud mask indicator (1=significant cloud cover, reducing quality of visible band data)

li: lunar illumination (based on phase of moon on that day)

tir: thermal infrared band value

vis: visible band value (relative brightness values from 0-63, with 0 being dark and 63 being maximum brightness)

sam: samples (a technical value referring to overlap of suborbital paths. NOAA recommends excluding sample values 302-304, 1162-1164)

slm:     0=> good dark data, no impact from stray light

1=> ground solar elevation > cutoff (satellite dependent), some satellites (particularly the later ones, F16 & F18) manifest stray light only as "side glare", or brightening as you approach the terminator in a scanline.

2=> satellite solar elev > cutoff (satellite dependent).  This is a hard cutoff at a specific scan line.  The earlier satellites really need this earlier than the later satellites do.  You've probably seen this.

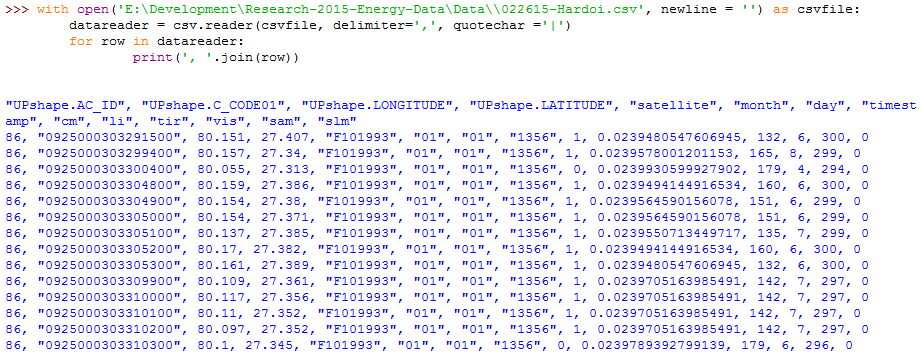
3=> both 1 & 2 above

255=> no-data value (outside of swath)

Researching D3 with large data sets; hardoi data is really big compared to the villages I’ve been working with.

I think I’ll clean up the data into a format that’s easier for me to work with. I’ll use a python script.

I’ll want to split the satellite ID off from the year. Since there are 80 villages and 20 years, graphing every day would be mean 365\*20 = 7300 data points for each village. That’s almost 4 times as many horizontal points as there are pixels in a standard 1920x1080 monitor, which means a lot of data that can’t even be rendered on a hardware level. Consolidating each month into a single point would give 12\*20 = 240 points for each village, with a total of 19200 points in total. That’s much more reasonable. Filters can be applied in the python script to pull out specific chunks of data.

**Timestamp: [8:15]** python script reads in the CSV file, now to do some formatting and data manipulation and save

Code snippet:

with open('E:\Development\Research-2015-Energy-Data\Data\\022615-Hardoi.csv', newline = '') as csvfile:

datareader = csv.reader(csvfile, delimiter=',', quotechar ='|')

for row in datareader:

print(', '.join(row))

Create a new file: Hardoi\_processed.csv

Input filters:

* Satellite: All, or speicifc one
* Li: range
* Tir: range
* Sam: range
* Slm: range

Output CSV:

* Village code
* long
* lat
* month
* year
* val

Break at 8:45 (cumulative time 1 hour)

Resumed at 10:00 pm

As the CSV data is not sorted chronologically, averaging months will require keeping months 1-12 for years 1993-2013 for each village in memory and then averaging the value as new rows are encountered/filtered out. Here is a date\_to\_index and index\_to\_date function to help with that.



Developing a script that reads in data and stores unique village codes

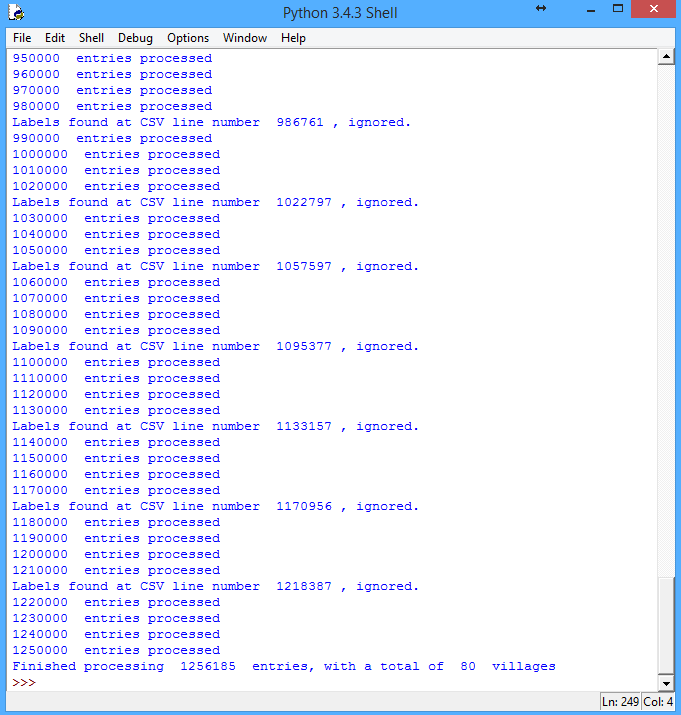
**Timestamp: [12:19]** Python shell crashed, rewriting code. Saving progress in logs.

**Timestamp: [12:41]**



**Timestamp [1:13] Restored code, final test run for today**





Found but with the updating average

Bug fixed by changing line

new\_data = [[0,0]]\*NUM\_DATA

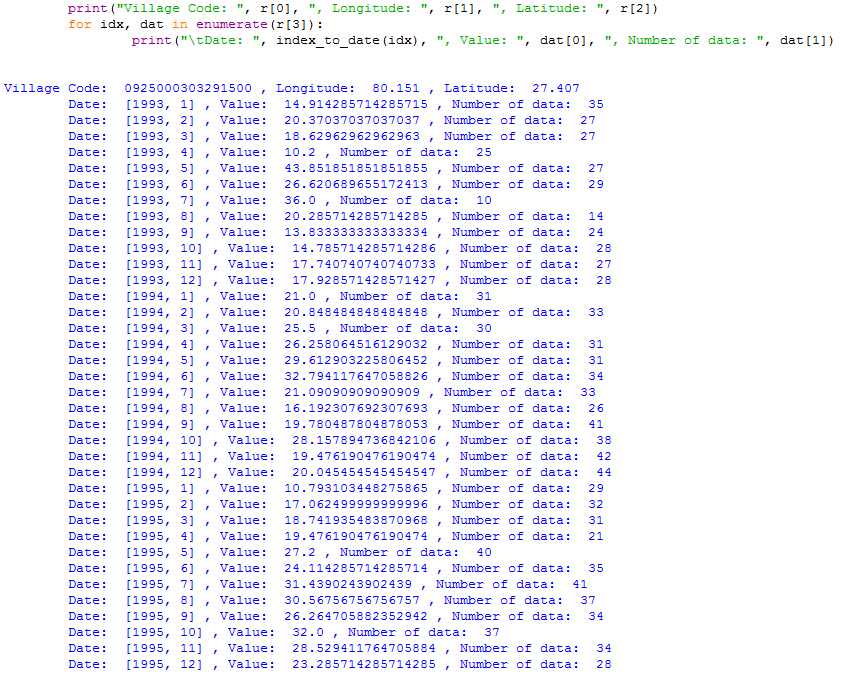
To

new\_data = [[0,0] for x in range(NUM\_DATA)]

Reason: old line was linking each data pair, resulting in every entry being an average of the entire set of data regardless of date

**Just a few more tweaks**

Results [snippet]:



Code to print formatted output, where r is a single village entry:



**Finish time 2:56 am (total time 6 hours)**

Todo: take the processed results and write it into the result csv

Todo: turn script into a module, which takes input/output filepaths and filtering parameters