Instructions for using census tract rescaling code. See final report for description of running GIS code. The GIS code should not need to be rerun anytime soon since the census tract information it relies on and generates corresponds to the 2020 census tracts.

The final estimates will be made available once the method of transfer is established. This will also have the Combinations_Union.csv file generated by the GIS scripts.

Base Requirements: Python 3 or higher. Numpy and NETCDF4 modules. Any additional requirements by these modules (e.g. a particular iteration of Python 3) also need to be observed.

First let's assume a directory structure where the source code is in the current directory. Our mapping directory will be ../map_data, our output directory will be ../output, and our input files (netcdf gridded nclimgrid-d data) is in ../ncei_data. (We are using / assuming Unix. Use \ instead for windows).

1.) Put the ncei data in the ncei_data directory. For this, setup subdirectories (e.g. ncei_data/1981) for each year. Put each monthly .nc file in its proper subdirectory (all 1981 months in subdirectory 1981. Monthly subdirectories are not needed).

2.) Setup directories in the output directory (e.g. ../output/1981), for the output .txt files containing the estimates.

3.) Edit the mapproc* scripts to utilize these directories.

4.) Put the Combinations_Union.csv file in the map directory.

5.) Run mapproc.g2c.py and mapproc.c2g.py scripts. These will generate pickle files for the dictionaries that are used hereafter which will be put into the output directory.

6.) Edit forproc.py for directories and dates. Remember to ensure the correct directories exist as described at the beginning of this document. See notes on final step concerning what these dates represent.

Estimates have now been produced and are in the text files in ../output/yyyy.

7.) Create subdirectory ../output/err and ../output/err/stats.

8.) Edit devcomp.py and run it for the desired dates and field (one field at a time). This produces deviation arrays that will go into subdirectory err. There will be one for each month and each field. The numpy array has the shape (numdays, numlat, numlon), where numdays is the number of days in the month and numlat, numlon are the 2-d grid dimensions.

9.) Edit errstat.py and run it for the desired dates and field. Output will go to err/stats subdirectory and the filenames contain enough info to know what is in them. These are numpy arrays and can be loaded with np.load for further manipulation/use/plotting etc. Remember, the output files are slices of each year. In other words,

start_month= 2, end_month=4, start_year=1981, end_year=1990

will compute the stats for February-April of years 1981-1990. Hence it would be a multi-month, multi_year, single gridded average. The range includes the endpoints. So if one wanted a 30 year spring average, do something like,

start_month = 3, end_month=5, start_year = 1981, end_year = 2010. This will compute a single, but spatially variable mean deviation, mean square deviation, and rms deviation for the spring season (averaged over March – May) averaged over the 30-year period (1981-2010).

Most of the time loops in the source code behave this way. If one wants the estimated census tract temperatures for all times in March, 1980 – July 1985. One needs to run forproc.py 3 times:

1.) start_month=3, end_month=12, start_year=1980, end_year=1980
2.) start_month=1, end_month=12, start_year=1981, end_year=1984
3.) start_month=1, end_month=7, start_year=1985, end_year=1985.

Last modified 5/22/2021 by C. Travis Ashby