GRIDMASK

May 5, 2021

1 Make GRIDMASK File

author: Barron H. Henderson
date: 2021-05-05

- Prereequisites:
  - a CSV file with country codes and time zones
  - Python v3 and libraries, pandas, PseudoNetCDF (optionally matplotlib and pycno)
- Output: CMAQ-compliant file with GEOCODE and TZONES variables

2 Import Libraries

- All libraries can be installed via 'pip install .
- If you are not an administrator, add `--user` after install.
- Libraries are imported here.

[1]:
```
install_libs = False
if install_libs:
    !pip install pandas matplotlib pseudonetcdf pycno
```

[2]:
```
import pandas as pd
import PseudoNetCDF as pnc
```

3 Read CSV data

- I am reading directly from the URL, but you can replace with a local path.

[3]:
```
# Must include Country_Code and Time_Zone
csvpath = 'https://forum.cmascenter.org/uploads/short-url/
 ↪krMNkylTmfdK2n5KO3YLfD3P9cg.csv'
data = pd.read_csv(csvpath)
```

4 Create a GRIDDESC file

- Based on the user details, I am defining a GRIDDESC with centers starting at 43.8E, 23.7N.
- The EDGAR domain, may be centered on the 0.05degrees. You should confirm.

[4]:
```
with open('GRIDDESC_MidEast', 'w') as gdf:
    gdf.write("""' '
'LATLON'
1  0.0 0.0 0.0 0.0 0.0
' '
'MIDEAST_Opt1'
'LATLON'  43.75 23.7 0.1 0.1  201  163 1
' '""")
```

5 Create some IOAPI-like files

- Open a Grid Definition NetCDF IOAPI-like file
- Create an output file to hold resutls

[5]:
```
gf = pnc.pncopen('GRIDDESC_MidEast', format='griddesc', GDNAM='MIDEAST_Opt1')
```

[6]:
```
outf = gf.subset([])
delattr(outf, 'VAR-LIST')
GEOCODE = outf.createVariable('GEOCODE', 'i', ('TSTEP', 'LAY', 'ROW', 'COL'))
GEOCODE.long_name = "GEOCODE                                                                ␣
 ↪                      "
GEOCODE.units = "none            "
GEOCODE.var_desc = "                                                                        ␣
 ↪                     "
TZONES = outf.createVariable('TZONES', 'i', ('TSTEP', 'LAY', 'ROW', 'COL'))
TZONES.long_name = "TZONES                                                                  ␣
 ↪                     "
TZONES.units = "none            "
TZONES.var_desc = "                                                                         ␣
 ↪                    "
```

6 Load the Data

- data has X and Y variables that are 1-based columns and rows.
- Python uses 0-based indices, so we subtract 1 to index variables.
- The data is then loaded into the indexed 1d vector

[7]:
```
GEOCODE[0, 0, data.Y - 1, data.X - 1] = data.Country_Code
TZONES[0, 0, data.Y - 1, data.X - 1] = data.Time_Zone
```

7 Save the Data to Disk

- We have to create a dummy TFLAG and then overwrite it.
- This creates a time-independent file
- The time-independent file is saved as NETCDF3_CLASSIC.
  - You can make it much smaller using NETCDF4_CLASSIC with the complevel=1

[8]:
```
outf.SDATE = 2000001
outf.updatemeta()
outf.variables.move_to_end('TFLAG', last=False)
outf.SDATE = -635
outf.variables['TFLAG'][:, :, 0] = 0
outf.save('GRIDMASK.nc', format='NETCDF3_CLASSIC').close()
# outf.save('GRIDMASK.nc', format='NETCDF4_CLASSIC', complevel).close()
```

```
Adding dimensions
Adding globals
Adding variables
Defining TFLAG
Defining GEOCODE
Defining TZONES
Populating TFLAG
Populating GEOCODE
Populating TZONES
```

8 Plot to Confirm

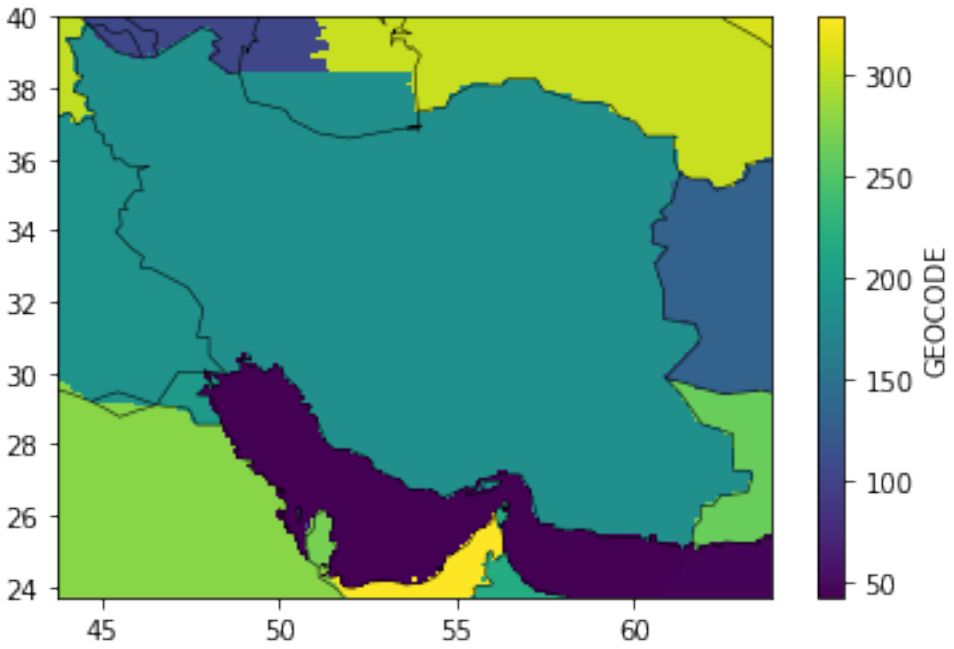
- Using matplotlib to plot and pycno for overlays.

[9]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import pycno
```

[10]:
```
cno = pycno.cno()
lon = gf.variables['longitude'][:]
lat = gf.variables['latitude'][:]
```

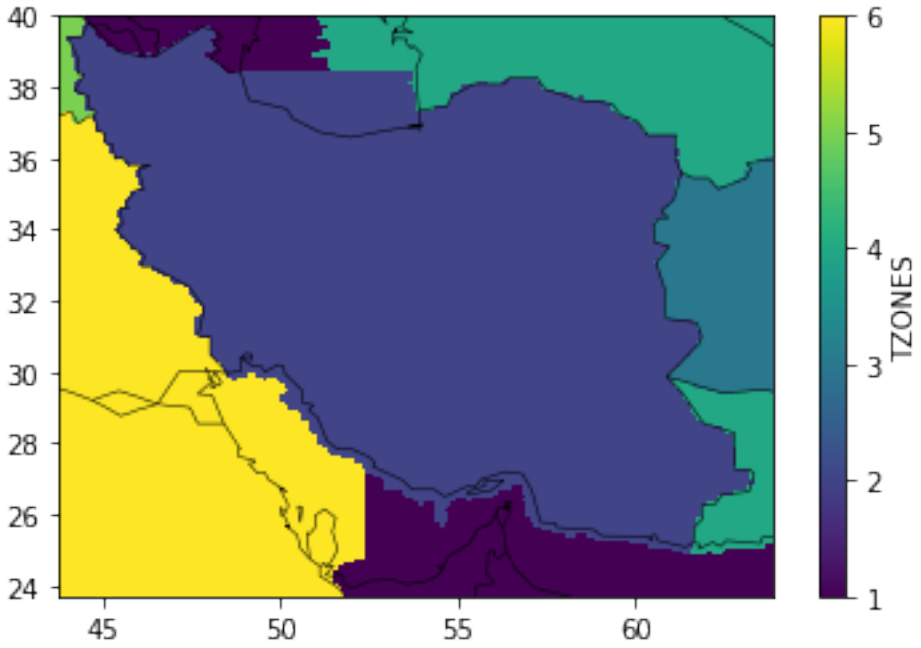
[11]:
```
plt.pcolormesh(lon, lat, outf.variables['GEOCODE'][0, 0], shading='nearest')
plt.colorbar(label='GEOCODE')
cno.draw()
```

[11]:
```
<matplotlib.collections.LineCollection at 0x2aabe71c1f28>
```



[12]:
```
plt.pcolormesh(lon, lat, outf.variables['TZONES'][0, 0], shading='nearest')
plt.colorbar(label='TZONES')
cno.draw()
```

[12]: <matplotlib.collections.LineCollection at 0x2aaad6db8c50>



[ ]: