In [ ]:
```
jupyter nbconvert --to html '/content/Extracting NTL data from GEE.ipynb'
```

In [ ]:
```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

# reminder that if you are installing libraries in a Google Colab instance you will be prompted to res

try:
    import geemap, ee
    import seaborn as sns
    import matplotlib.pyplot as plt
except ModuleNotFoundError:
    if 'google.colab' in str(get_ipython()):
        print("package not found, installing w/ pip in Google Colab...")
        !pip install geemap seaborn matplotlib
    else:
        print("package not found, installing w/ conda...")
        !conda install mamba -c conda-forge -y
        !mamba install geemap -c conda-forge -y
        !conda install seaborn matplotlib -y
    import geemap, ee
    import seaborn as sns
    import matplotlib.pyplot as plt
```

In [ ]:

```
try:
        ee.Initialize()
except Exception as e:
        ee.Authenticate()
        ee.Initialize()
```

In [ ]:

```
from tqdm import tqdm
import itertools
import statsmodels.api as sm
import warnings
warnings.filterwarnings('ignore')
from matplotlib import gridspec
import matplotlib.dates as mdates
from matplotlib.dates import DateFormatter
import geocoder
import datetime
import geemap.colormaps as cm
from scipy.stats import norm
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
data_path = "/content/drive/MyDrive/PhD Physics/4.1/Physics 305 - White Noise Analysis/Project/data/"
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/dri
ve", force_remount=True).
```

In [ ]:
```
select_regions_names = ['NCR',
                'CAR',
                'Region I',
                'Region II',
                'Region III',
                'Region IV-A',
                'Region IV-B',
                'Region V',
                'Region VI',
                'Region VII',
                'Region VIII',
                'Region IX',
                'Region X',
                'Region XI',
                'Region XII',
                'Region XIII',
                'ARMM']
select_regions = [2355, 2354, 2356, 2357,
           67165, 67166, 67167,
           2361, 2362, 2363, 2364,
           67159, 67160, 67161, 67162,
           2368, 67156]
```

In [ ]:
```
start_date, end_date = "2012-01-01", "2023-06-12"
black_marble = ee.ImageCollection('NOAA/VIIRS/001/VNP46A2').filterDate(start_date, end_date)
```

In [ ]:
```
# reg_geoms = ee.FeatureCollection("FAO/GAUL/2015/level1").filter(ee.Filter.inList('ADM1_CODE', select
# reg_geoms = ee.FeatureCollection("FAO/GAUL/2015/level1").filter(ee.Filter.inList('ADM1_CODE', select
```

In [ ]:
```
Region = 67167
region = ee.Feature(ee.FeatureCollection("FAO/GAUL/2015/level1").filter(ee.Filter.eq('ADM1_CODE', Regi
```

In [ ]:
```
def get_reg_mean(img):
    mean = img.reduceRegion(reducer=ee.Reducer.mean(), geometry=region, scale=500, maxPixels=1e9)#.get
    return img.set('date', img.date().format()).set('mean',mean)
```

In [ ]:
```
# function to reduce our collection of geometries'
reg_mean = black_marble.map(get_reg_mean)
nested_list = reg_mean.reduceColumns(ee.Reducer.toList(2), ['date', 'mean']).values().get(0)
data = pd.DataFrame(nested_list.getInfo(), columns=['date','mean'])
data['date'] = pd.to_datetime(data['date'])
data = data.set_index('date')
```

In [ ]:
```
data
```

Out[48]:

date	mean
2012-01-19	{'DNB_BRDF_Corrected_NTL': 0.41166322260291754...
2012-01-20	{'DNB_BRDF_Corrected_NTL': 0.5327007857679509,...
2012-01-21	{'DNB_BRDF_Corrected_NTL': 0.24979786297863849...
2012-01-22	{'DNB_BRDF_Corrected_NTL': 0.19265276211327628...
2012-01-23	{'DNB_BRDF_Corrected_NTL': 0.2368002957256932,...
...	...
2023-06-07	{'DNB_BRDF_Corrected_NTL': 0.4957945630814749,...
2023-06-08	{'DNB_BRDF_Corrected_NTL': 0.43047866083847375...
2023-06-09	{'DNB_BRDF_Corrected_NTL': 0.24208098661214958...
2023-06-10	{'DNB_BRDF_Corrected_NTL': 0.6273905798166924,...
2023-06-11	{'DNB_BRDF_Corrected_NTL': None, 'DNB_Lunar_Ir...

4142 rows × 1 columns

In [ ]:
```
xx = data['mean'].apply(pd.Series)
```

In [ ]:
```
xx
```

Out[53]:

date	DNB_BRDF_Corrected_NTL	DNB_Lunar_Irradiance	Gap_Filled_DNB_BRDF_Corrected_NTL	Latest_High_Quality_Retrieval	Mandatory_Qual
2012-01-19	0.411663	0.500000	0.365072	0.129124	0
2012-01-20	0.532701	0.500000	0.459410	0.090830	0
2012-01-21	0.249798	0.500000	0.222583	0.015499	0
2012-01-22	0.192653	0.500000	0.175084	0.033500	0
2012-01-23	0.236800	0.500000	0.236953	0.032556	0
...	...	...	...	...	
2023-	0.495795	43.200000	0.491732	0.344015	0

In [ ]:
```
data.to_csv(data_path+str(select_regions_names[6])+'_NTL.csv')
```

In [ ]:
```
for i in tqdm(range(0, len(select_regions))[6:8]):
    reg = select_regions[i]
    region = ee.Feature(ee.FeatureCollection("FAO/GAUL/2015/level1").filter(ee.Filter.eq('ADM1_CODE',
    reg_mean = black_marble.map(get_reg_mean)
    nested_list = reg_mean.reduceColumns(ee.Reducer.toList(2), ['date','mean']).values().get(0)
    data = pd.DataFrame(nested_list.getInfo(), columns=['date','mean'])
    data['date'] = pd.to_datetime(data['date'])
    data = data.set_index('date')
    data = data['mean'].apply(pd.Series)
    data.to_csv(data_path+str(select_regions_names[i])+'_NTL_VNP46A2.csv')
```

```
  0%|          | 0/2 [05:42<?, ?it/s]

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-59-b7abaa67a11f> in <cell line: 1>()
      4     reg_mean = black_marble.map(get_reg_mean)
      5     nested_list = reg_mean.reduceColumns(ee.Reducer.toList(2), ['date','mean']).values().get(0)
----> 6     data = pd.DataFrame(nested_list.getInfo(), columns=['date','mean'])
      7     data['date'] = pd.to_datetime(data['date'])
      8     data = data.set_index('date')

/usr/local/lib/python3.10/dist-packages/ee/computedobject.py in getInfo(self)
    103       The object can evaluate to anything.
    104     """
--> 105     return data.computeValue(self)
    106 
    107   def encode(self, encoder: Optional[Callable[..., Any]]) -> Dict[str, Any]:

/usr/local/lib/python3.10/dist-packages/ee/data.py in computeValue(obj)
```

In [ ]:
```
for i in range(6, len(select_regions))[:1]: print(i)
```

```
6
```

In [ ]:
```

```

In [ ]:
```
data_new
```

Out[67]:

	mean
date	
2012-01-19	0.187736
2012-01-20	0.372040
2012-01-21	0.090909
2012-01-22	0.096918
2012-01-23	0.125274
...	...
2023-06-05	3.328632
2023-06-06	0.568010
2023-06-07	0.161307
2023-06-08	0.446584
2023-06-09	0.477217

3645 rows × 1 columns

In [ ]:

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

In [ ]:

In [ ]:

```
data_new.to_csv(data_path+str(select_regions_names[i])+'_NTL.csv')
```

In [ ]:
```
data
```

Out[41]:

date	mean
2023-01-06	25.168575
2023-01-07	28.910789
2023-01-08	30.496423
2023-01-10	14.652978
2023-01-11	19.637830
2023-01-12	38.645784
2023-01-16	17.948306
2023-01-18	76.867673
2023-01-19	18.453818
2023-01-20	28.499577
2023-01-24	23.550230
2023-01-25	24.718977
2023-01-27	19.152515
2023-01-29	28.965195
2023-01-30	26.036020
2023-01-31	27.806483

In [ ]:
```
def time_series_(lat, lon, buffer):
    def poi_mean(img):
        mean = img.reduceRegion(reducer=ee.Reducer.mean(), geometry=poi, scale=30).get('DNB_BRDF_Corre
        return img.set('date', img.date().format()).set('mean',mean)

    poi = ee.Geometry.Point(lat, lon).buffer(buffer)
    black_marble = ee.ImageCollection('NOAA/VIIRS/001/VNP46A2').filterDate(start_date, end_date).selec
    poi_reduced_imgs = black_marble.map(poi_mean)
    nested_list = poi_reduced_imgs.reduceColumns(ee.Reducer.toList(2), ['date', 'mean']).values().get(
    data = pd.DataFrame(nested_list.getInfo(), columns=['date','mean'])
    data['date'] = pd.to_datetime(data['date'])
    data = data.set_index('date')
    return data
```

In [ ]:

```
# function to reduce our collection of geometries'
def get_reg_avg_rad(img):
    return img.reduceRegions(reducer=ee.Reducer.mean(), collection=reg_geoms, scale=500)

# function to get individual image dates
def get_date(img):
    return img.set('date', img.date().format())

# map these functions to our image collection
reduced_regions = black_marble.map(get_reg_avg_rad).flatten()
dates = black_marble.map(get_date)

# get lists
key_cols = ['ADM1_CODE','mean']

regions_list = reduced_regions.reduceColumns(ee.Reducer.toList(len(key_cols)), key_cols).values()
dates_list = dates.reduceColumns(ee.Reducer.toList(1), ['date']).values()

# some numpy maneuvers to structure our data
df = pd.DataFrame(np.asarray(regions_list.getInfo()).squeeze(), columns=key_cols)
dates = np.asarray(dates_list.getInfo()).squeeze()
```

In [ ]:

```python
for regions in select_regions:
    # df.loc[df['ADM1_CODE']==regions,'dates'] = dates[:len(df.loc[df['ADM1_CODE']==regions,'dates'])]
    df.loc[df['ADM1_CODE']==regions,'dates'] = dates

# as we've done before, convert date and set index
df['dates'] = pd.to_datetime(df['dates'])
df.set_index('dates', inplace=True)

# we'll also convert our mean datatype to float
df['mean'] = df['mean'].astype(float)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-30-996c3faee783> in <cell line: 1>()
      1 for regions in select_regions:
      2     # df.loc[df['ADM1_CODE']==regions,'dates'] = dates[:len(df.loc[df['ADM1_CODE']==regions,'da
tes'])] #for VNP46A2
----> 3     df.loc[df['ADM1_CODE']==regions,'dates'] = dates
      4 
      5 # as we've done before, convert date and set index

/usr/local/lib/python3.10/dist-packages/pandas/core/indexing.py in __setitem__(self, key, value)
    816 
    817         iloc = self if self.name == "iloc" else self.obj.iloc
--> 818         iloc._setitem_with_indexer(indexer, value, self.name)
    819 
    820     def _validate_key(self, key, axis: int):

/usr/local/lib/python3.10/dist-packages/pandas/core/indexing.py in _setitem_with_indexer(self, indexer,
value, name)
   1736                         # if not Series (in which case we need to align),
   1737                         #  we can short-circuit
-> 1738                         empty_value[indexer[0]] = arr
   1739                         self.obj[key] = empty_value
   1740                         return
```

```
ValueError: NumPy boolean array indexing assignment cannot assign 31 input values to the 16 output valu
es where the mask is true
```

In [ ]:

```

```

In [ ]:

```python
def get_coords(loc):
    g = geocoder.bing(loc, key='Ag-hhlLXX66GFF0QB3A84aPX16lgpoz-nGdT28B6IfGf_skCcDZ8XqG6qefUQ2-1')
    lat, lon = g.json['lat'], g.json['lng']

    return lat, lon

def moving_average(a, n) :
    ret = np.cumsum(a, dtype=float)
    ret[n:] = ret[n:] - ret[:-n]
    return ret[n - 1:] / n
```

In [ ]:
```
def map_viewer(lat, lon, loc, buffer, start_date, end_date, zoom):
    poi = ee.Geometry.Point(lon, lat).buffer(buffer)
    viirs = ee.ImageCollection("NOAA/VIIRS/DNB/MONTHLY_V1/VCMSLCFG").filterDate(start_date, end_date)
    viirs_image = ee.ImageCollection("NOAA/VIIRS/DNB/MONTHLY_V1/VCMSLCFG").filterDate(start_date, end_

    def poi_mean(img):
        mean = img.reduceRegion(reducer=ee.Reducer.mean(), geometry=poi, scale=30).get('avg_rad')
        return img.set('date', img.date().format()).set('mean',mean)

    poi_reduced_imgs = viirs.map(poi_mean)
    viirs_clipped = viirs_image.clip(poi)

    viz_params = {'min':0,
              'max':5,
              'palette': cm.palettes.magma}

    Map = geemap.Map()
    Map.centerObject(poi, zoom=zoom)
    Map.add_basemap("HYBRID")
    Map.addLayer(poi, {}, str(loc), opacity = 0.5)

    Map.addLayer(poi, {}, str(loc) + 'POI')
    Map.addLayer(viirs_clipped, viz_params, str(loc) + 'NTL-clipped',  opacity=0.3)
    Map.addLayer(viirs_image, viz_params, 'VIIRS DNB',  opacity=0.50)
    Map.add_colorbar(viz_params, label="NTL radiance, $nW ·cm^{-2}·sr^{-1}$", layer_name = "colorbar")
    Map.addLayerControl()
    return Map
```

In [ ]:
```
def time_series_(lat, lon, buffer):
    def poi_mean(img):
        mean = img.reduceRegion(reducer=ee.Reducer.mean(), geometry=poi, scale=30).get('DNB_BRDF_Corre
        return img.set('date', img.date().format()).set('mean',mean)

    poi = ee.Geometry.Point(lat, lon).buffer(buffer)
    black_marble = ee.ImageCollection('NOAA/VIIRS/001/VNP46A2').filterDate(start_date, end_date).selec
    poi_reduced_imgs = black_marble.map(poi_mean)
    nested_list = poi_reduced_imgs.reduceColumns(ee.Reducer.toList(2), ['date', 'mean']).values().get(
    data = pd.DataFrame(nested_list.getInfo(), columns=['date','mean'])
    data['date'] = pd.to_datetime(data['date'])
    data = data.set_index('date')
    return data
```

In [ ]:
```
lat, lon = get_coords('Marawi City')
start_date, end_date = "2017-01-01", "2017-12-31"
black_marble_data = time_series_(lon, lat, buffer = 500)
```

In [ ]:
```
def Black_Marble_STL(lat, lon, loc, buffer, start_date, end_date, period):

    def time_series_(lat, lon, buffer):
        def poi_mean(img):
            mean = img.reduceRegion(reducer=ee.Reducer.mean(), geometry=poi, scale=30).get('DNB_BRDF_C
            return img.set('date', img.date().format()).set('mean',mean)

        poi = ee.Geometry.Point(lat, lon).buffer(buffer)
        black_marble = ee.ImageCollection('NOAA/VIIRS/001/VNP46A2').filterDate(start_date, end_date).s
        poi_reduced_imgs = black_marble.map(poi_mean)
        nested_list = poi_reduced_imgs.reduceColumns(ee.Reducer.toList(2), ['date', 'mean']).values().
        data = pd.DataFrame(nested_list.getInfo(), columns=['date','mean'])
        data['date'] = pd.to_datetime(data['date'])
        data = data.set_index('date')
        return data
```

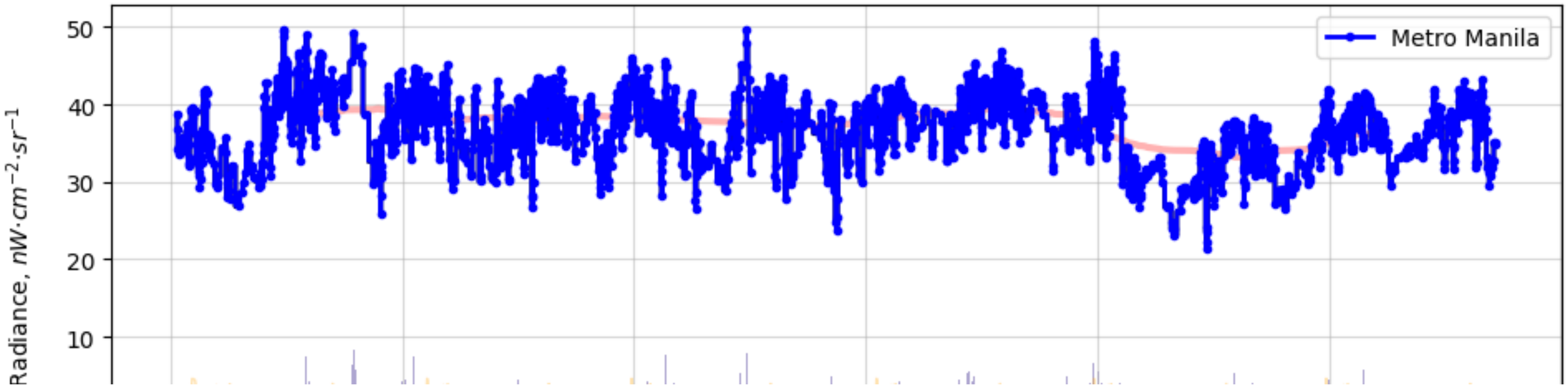
```
black_marble_data = time_series_(lon, lat, buffer)
y = moving_average(black_marble_data['mean'].tolist(), n = 4)

decomposition = sm.tsa.seasonal_decompose(y, model='additive', period = period)
decomposition_2 = sm.tsa.seasonal_decompose(y, model='additive', period = period*2)


fig, ax = plt.subplots(nrows=4,ncols=1, sharex='col',
                      gridspec_kw={'height_ratios':[2, 1,0.6,0.6]},
                      figsize=(10,8))

date = black_marble_data.index[3:]



ax[0].plot(date, decomposition.trend, 'r', alpha = 0.3, lw = 3)
ax[0].plot(date, decomposition.seasonal, color = 'orange', alpha = 0.3, lw = 1)
ax[0].plot(date, y-0.1, 'k', lw = 2, drawstyle = 'steps-mid')
ax[0].plot(date, y, 'b.-', lw = 2, drawstyle = 'steps-mid', label = str(loc))

ax[0].bar(date, decomposition.resid, width = 1,
        color = plt.cm.Spectral(decomposition.resid), alpha = 0.5)
ax[0].set_ylabel('NTL Radiance, $nW ·cm^{-2}·sr^{-1}$')

ax[1].plot(date, decomposition.trend-0.02, 'k', lw = 2)
ax[1].plot(date, decomposition.trend, 'r', label = 'trend', lw = 3)


ax[2].plot(date, decomposition.seasonal, color = 'orange',  label = 'seasonality')

residual = decomposition.resid
ax[3].bar(date, residual, width = 1.5, label = 'residual')
ax[3].bar(date, residual, width = 1.5,
        color = plt.cm.Spectral(residual))
```

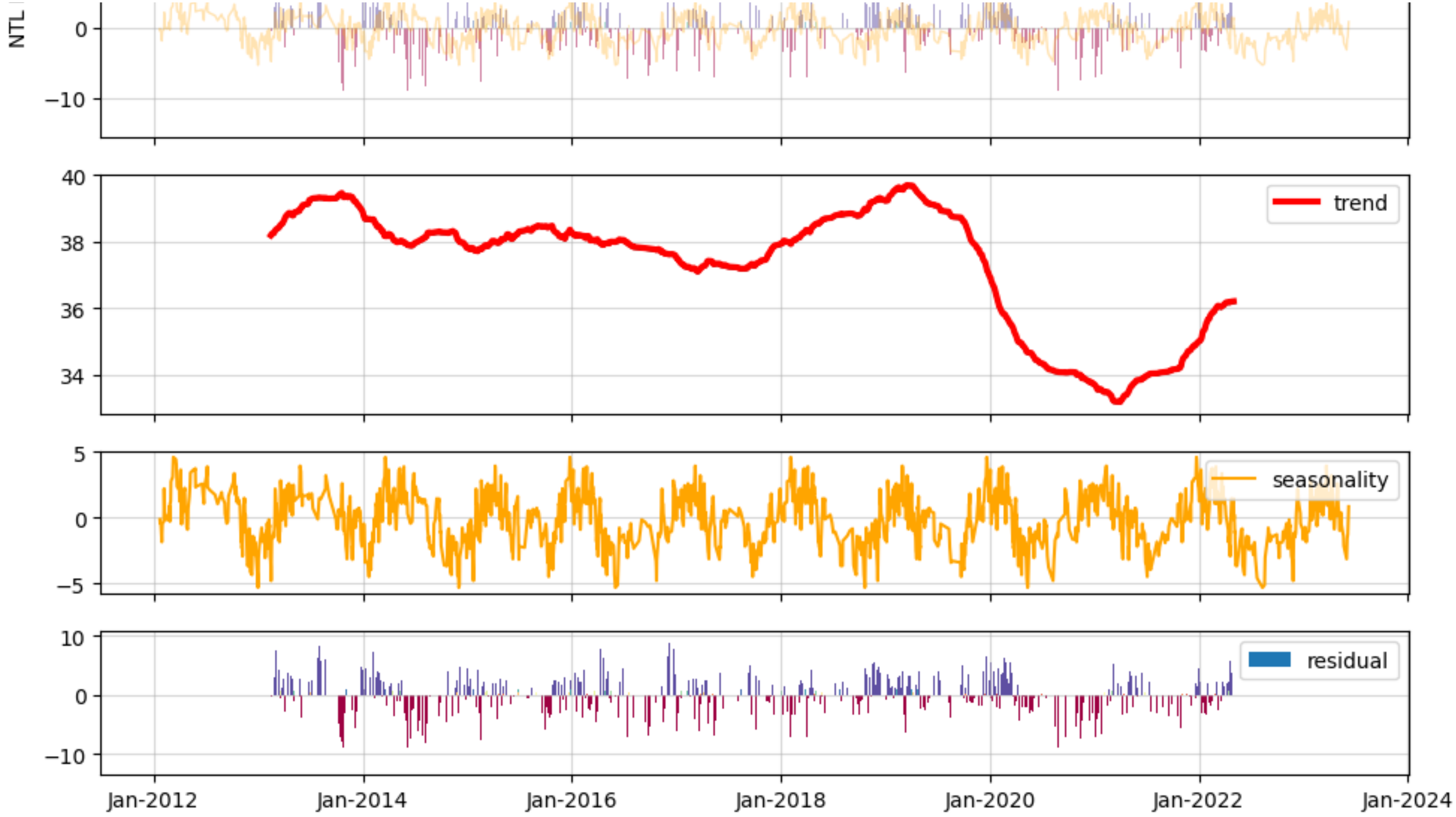
```
    for ax in ax:
      ax.legend(loc = 1)
      # ax.axvline(datetime.date(2017, 5, 18), color = 'g', lw = 10, alpha = 0.2) #For Marawi Seige
      # ax.axvline(datetime.date(2020, 3, 10), color = 'g', lw = 10, alpha = 0.2) #For COVID lockdown

      ax.grid(alpha = 0.5)

    dtFmt = mdates.DateFormatter('%b-%Y') # define the formatting
    plt.gca().xaxis.set_major_formatter(dtFmt) # apply the format to the desired axis
    plt.tight_layout()
    plt.show()
```

In [ ]:
```
loc = 'Metro Manila'
lat, lon = get_coords(loc)
map_viewer(lat, lon, loc, 10000, "2017-01-01", "2018-02-01", zoom = 10)
```

```
Map(center=[14.608647118873229, 121.03194456399703], controls=(WidgetControl(options=['position', 'tran
sparent…
```

In [ ]:
```
Black_Marble_STL(lat, lon, loc, 3000, "2012-01-19", "2023-06-12", period = 365)
```





Extracting Metro Manila NTL values

In [ ]:
```
lat, lon = get_coords('Metro Manila')
start_date, end_date = "2012-01-01", "2023-06-12"
black_marble_data = time_series_(lon, lat, buffer = 10000)
```

In [ ]:
```
import os
os.getcwd()
```

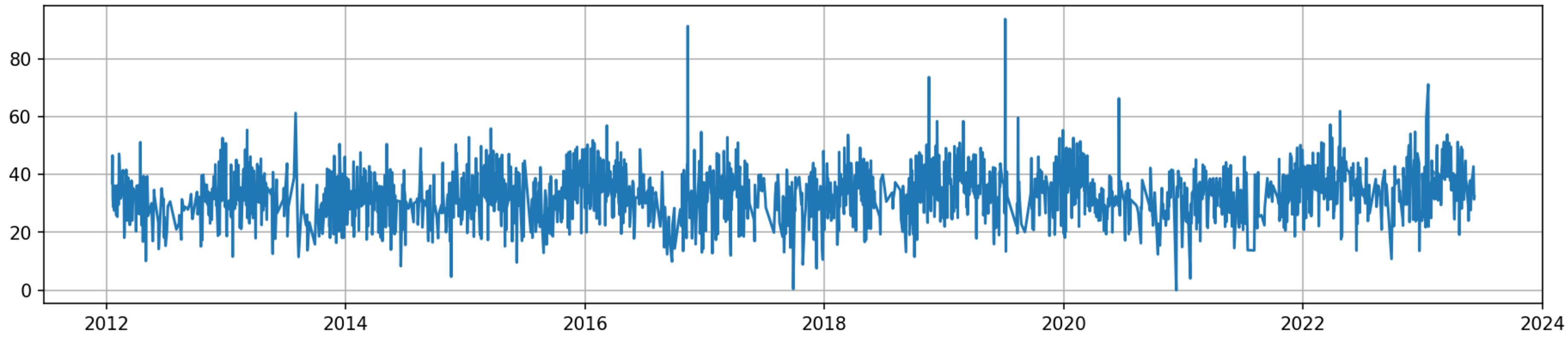
Out[28]: '/content'

In [ ]:
```
black_marble_data.to_csv('NTL_Metro_Manila.csv')
```

Timeseries Plot

In [ ]:
```
plt.figure(dpi = 150, figsize = (15,3))
plt.grid()
plt.plot(black_marble_data)
```

Out[12]: [<matplotlib.lines.Line2D at 0x790c050d1930>]



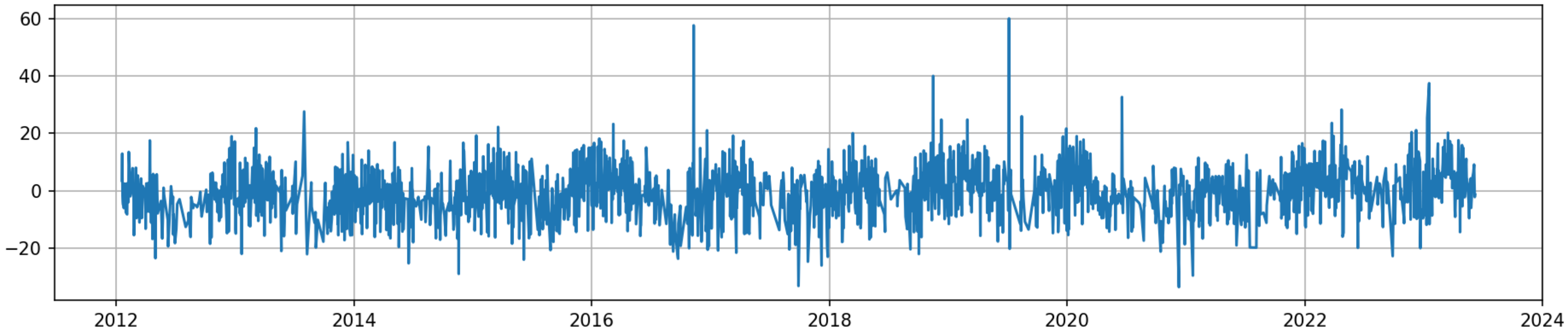
In [ ]:
```
len(black_marble_data)
```

Out[13]: 2472

Mean Centering

In [ ]:
```
data = black_marble_data.values
data = data - np.mean(data)

plt.figure(dpi = 150, figsize = (15,3))
plt.grid()
plt.plot(black_marble_data.index, data)
```

Out[14]: [<matplotlib.lines.Line2D at 0x790c034279d0>]



Cummulative Sum

In [ ]:
```
plt.figure(dpi = 150, figsize = (15,3))
plt.grid()
data_cumsum = np.cumsum(data)
plt.plot(black_marble_data.index, data_cumsum)
```

Out[15]:
```
[<matplotlib.lines.Line2D at 0x790c034d1a50>]
```

Generating Probability Density Functions (PDFs)

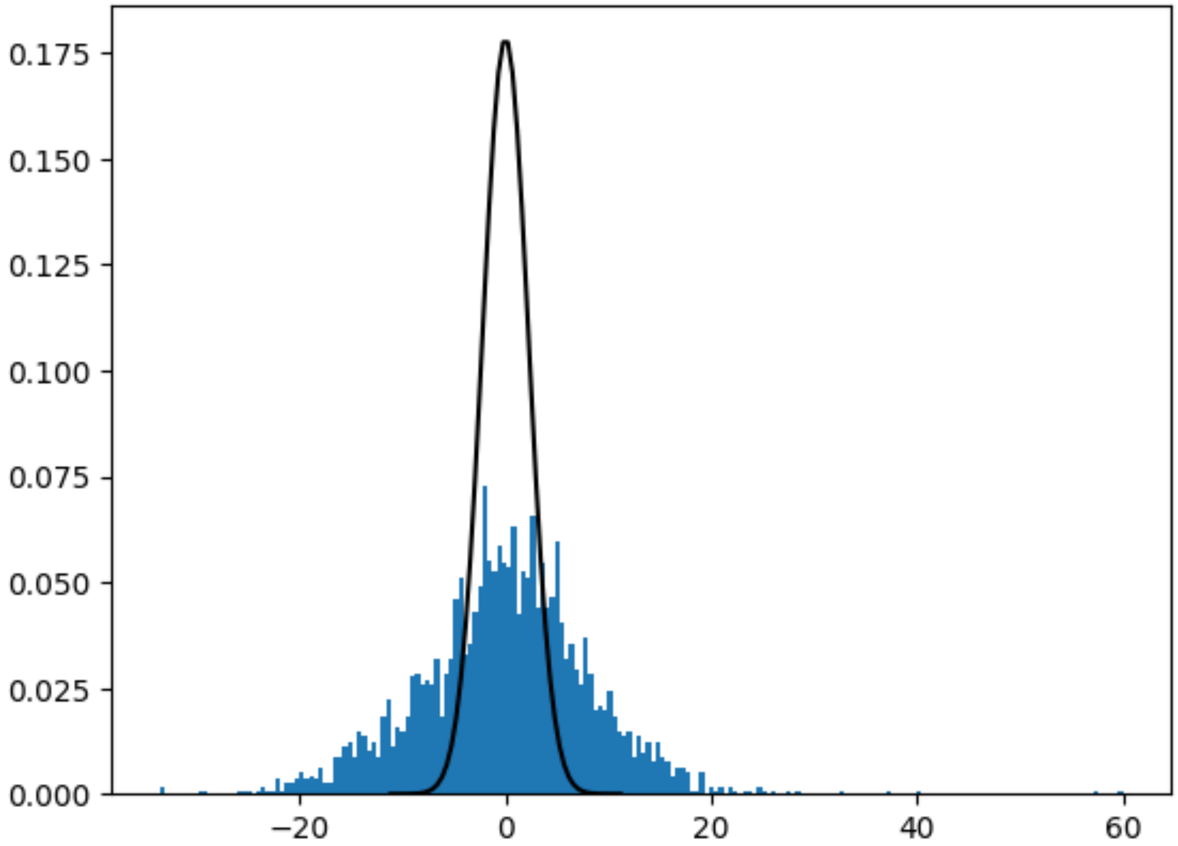
In [ ]:
```

```

In [ ]:
```
dt = 5.  # size of time step
sd = np.sqrt(dt)
```

In [ ]:
```
xx = np.linspace(-5, 5)*sd # gridded points from -5 to 5 in units of sd
xx_mean = 0.
xx_sd = sd
yy = norm.pdf(xx, xx_mean, xx_sd)
```

In [ ]:
```
n_bins = 200
plt.hist(data, density=True, bins=n_bins, label="nighttime lights")
plt.plot(xx, yy, 'k-', label="N(%.1f,%.2f)" % (xx_mean, xx_sd))
```

Out[21]: [<matplotlib.lines.Line2D at 0x790c05295450>]



In [ ]: