Sunrise logo

ENVIRONMENTAL ENFORCEMENT WATCH

EDGI Environmental Data & Governance Initiative

The notebook was collaboratively authored by EDGI following our authorship protocol: https://docs.google.com/document/d/1CtDN5ZZ4Zv70fHiBTmWkDJ9mswEipX6eCYrwicP66Xw/ (https://docs.google.com/document/d/1CtDN5ZZ4Zv70fHiBTmWkDJ9mswEipX6eCYrwicP66Xw/)

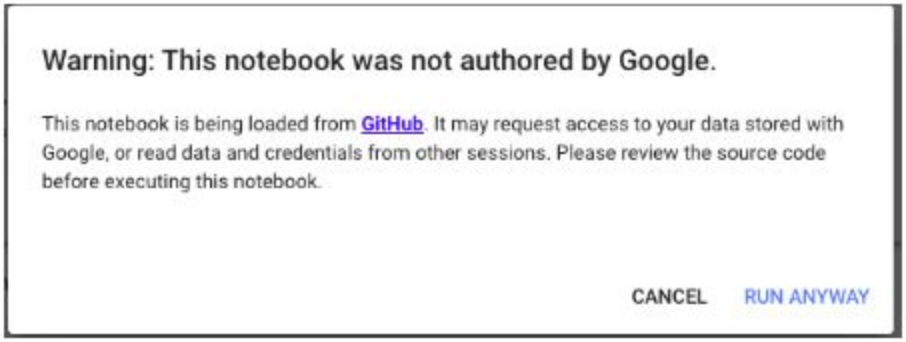
For more information about this project, visit https://www.environmentalenforcementwatch.org/ (https://www.environmentalenforcementwatch.org/)

How to Run

- A "cell" in a Jupyter notebook is a block of code performing a set of actions making available or using specific data. The notebook works by running one cell after another, as the notebook user selects offered options.
- If you click on a gray **code** cell, a little “play button” arrow appears on the left. If you click the play button, it will run the code in that cell (“**running** a cell”). The button will animate. When the animation stops, the cell has finished running.

```
# Import libraries
import urllib.parse
import pandas as pd
```

- You may get a warning that the notebook was not authored by Google. We know, we authored them! It’s okay. Click “Run Anyway” to continue.



- **It is important to run cells in order because they depend on each other.**
- Run all of the cells in a Notebook to make a complete report. Please feel free to look at and **learn about each result as you create it**!

Let's begin!

These first two cells give us access to some external Python code we will need. Hover over the "[ ]" on the top left corner of the cell below and you should see a "play" button appear. Click on it to run the cell then move to the next one.

1. Bring in some code that is stored in a Github project.

In [41]:
```
!git clone https://github.com/edgi-govdata-archiving/ECHO_modules.git
!git clone https://github.com/edgi-govdata-archiving/ECHO-Geo.git
!git clone -b first-draft --single-branch  https://github.com/edgi-govdata-archiving
```

```
Cloning into 'ECHO_modules'...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 27 (delta 7), reused 16 (delta 4), pack-reused 0
Unpacking objects: 100% (27/27), done.
Cloning into 'ECHO-Geo'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 11 (delta 2), reused 6 (delta 2), pack-reused 0
Unpacking objects: 100% (11/11), done.
Cloning into 'ECHO-Sunrise'...
remote: Enumerating objects: 91, done.
remote: Counting objects: 100% (91/91), done.
remote: Compressing objects: 100% (84/84), done.
remote: Total 91 (delta 53), reused 18 (delta 6), pack-reused 0
Unpacking objects: 100% (91/91), done.
```

2. Run some external Python modules.

In [42]:
```
# Import code libraries
%run ECHO_modules/DataSet.py
%run ECHO-Sunrise/utilities.py
import urllib.parse
import pandas as pd
!pip install geopandas
import geopandas
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import requests
import csv
import datetime
import ipywidgets as widgets
```

```
Requirement already satisfied: geopandas in /Users/enost/anaconda3/lib/python3.7/s
ite-packages (0.7.0)
Requirement already satisfied: fiona in /Users/enost/anaconda3/lib/python3.7/site-
packages (from geopandas) (1.8.13.post1)
Requirement already satisfied: pandas>=0.23.0 in /Users/enost/anaconda3/lib/python
3.7/site-packages (from geopandas) (0.23.4)
Requirement already satisfied: pyproj>=2.2.0 in /Users/enost/anaconda3/lib/python
3.7/site-packages (from geopandas) (2.6.1.post1)
Requirement already satisfied: shapely in /Users/enost/anaconda3/lib/python3.7/sit
e-packages (from geopandas) (1.7.0)
Requirement already satisfied: six>=1.7 in /Users/enost/anaconda3/lib/python3.7/si
te-packages (from fiona->geopandas) (1.11.0)
Requirement already satisfied: click<8,>=4.0 in /Users/enost/anaconda3/lib/python
3.7/site-packages (from fiona->geopandas) (6.7)
Requirement already satisfied: munch in /Users/enost/anaconda3/lib/python3.7/site-
packages (from fiona->geopandas) (2.5.0)
Requirement already satisfied: cligj>=0.5 in /Users/enost/anaconda3/lib/python3.7/
site-packages (from fiona->geopandas) (0.5.0)
Requirement already satisfied: attrs>=17 in /Users/enost/anaconda3/lib/python3.7/s
ite-packages (from fiona->geopandas) (18.2.0)
Requirement already satisfied: click-plugins>=1.0 in /Users/enost/anaconda3/lib/py
thon3.7/site-packages (from fiona->geopandas) (1.1.1)
Requirement already satisfied: python-dateutil>=2.5.0 in /Users/enost/anaconda3/li
b/python3.7/site-packages (from pandas>=0.23.0->geopandas) (2.7.5)
Requirement already satisfied: pytz>=2011k in /Users/enost/anaconda3/lib/python3.
7/site-packages (from pandas>=0.23.0->geopandas) (2018.5)
Requirement already satisfied: numpy>=1.9.0 in /Users/enost/anaconda3/lib/python3.
7/site-packages (from pandas>=0.23.0->geopandas) (1.15.1)
WARNING: You are using pip version 19.1, however version 20.1.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

3. What facilities does ECHO track in Mass?

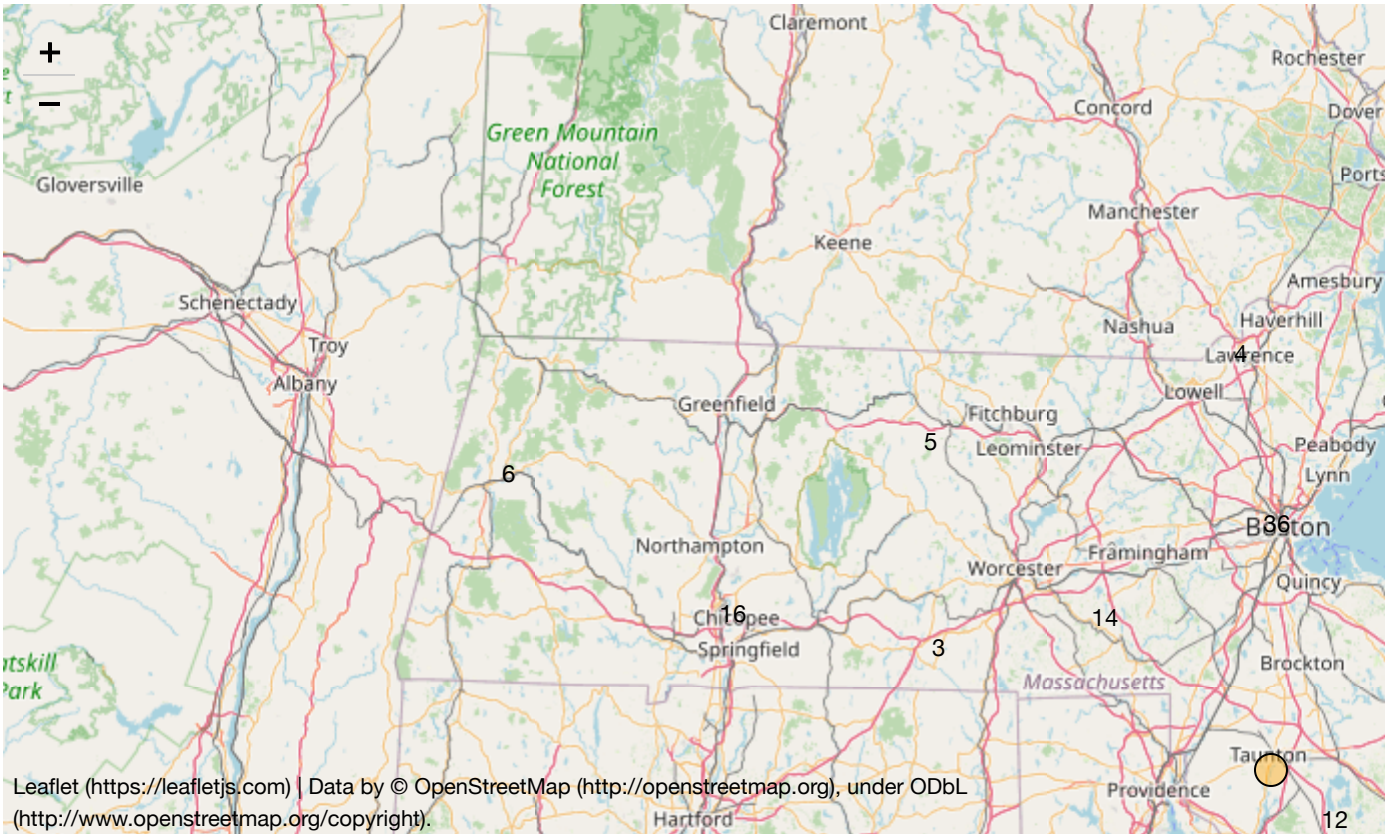
This may take some time to load - there are thousands of facilities!

In [43]:
```
echo_data_sql = "select * from ECHO_EXPORTER where FAC_STATE = 'MA' and FAC_ACTIVE_F

try:
    print(echo_data_sql)
    echo_data = get_data( echo_data_sql, 'REGISTRY_ID' )
    num_facilities = echo_data.shape[0]
    print("\nThere are %s EPA facilities in Massachussets tracked in the ECHO databa
    #mapper_marker(echo_data) # Not showing up...
except pd.errors.EmptyDataError:
    print("\nThere are no EPA facilities in this region.\n")
```

```
select * from ECHO_EXPORTER where FAC_STATE = 'MA' and FAC_ACTIVE_FLAG='Y' and GHG
_FLAG='Y'

There are 97 EPA facilities in Massachussets tracked in the ECHO database.
```

In [44]:
```
mapper_marker(echo_data)
```

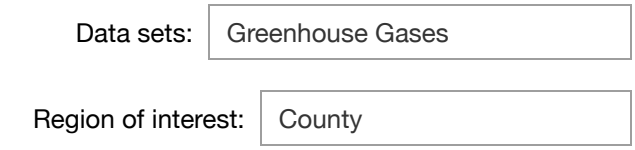
Out[44]:



4. Run this next cell to create to choose how you want to *zoom in*: what specific programs you want to look at and whether you want to view this information by county, congressional district or zip code.

Here's where you can learn more about the different programs...

In [45]:
```
%run ECHO_modules/make_data_sets.py

# Only list the data set if it has the correct flag set.
data_set_choices = []
for k, v in data_sets.items():
    if ( v.has_echo_flag( echo_data ) ):
        data_set_choices.append( k )

data_set_widget=widgets.Dropdown(
    options=list(data_set_choices),
    description='Data sets:',
    disabled=False,
    value='Greenhouse Gases'
)
display(data_set_widget)

region_field = {
    'Congressional District': { "field": 'FAC_DERIVED_CD113' },
    'County': { "field": 'FAC_COUNTY' },
    'Zip Code': { "field": 'FAC_DERIVED_ZIP' }
}

style = {'description_width': 'initial'}
select_region_widget = widgets.Dropdown(
    options=region_field.keys(),
    style=style,
    value='County',
    description='Region of interest:',
    disabled=False
)
display( select_region_widget )
```



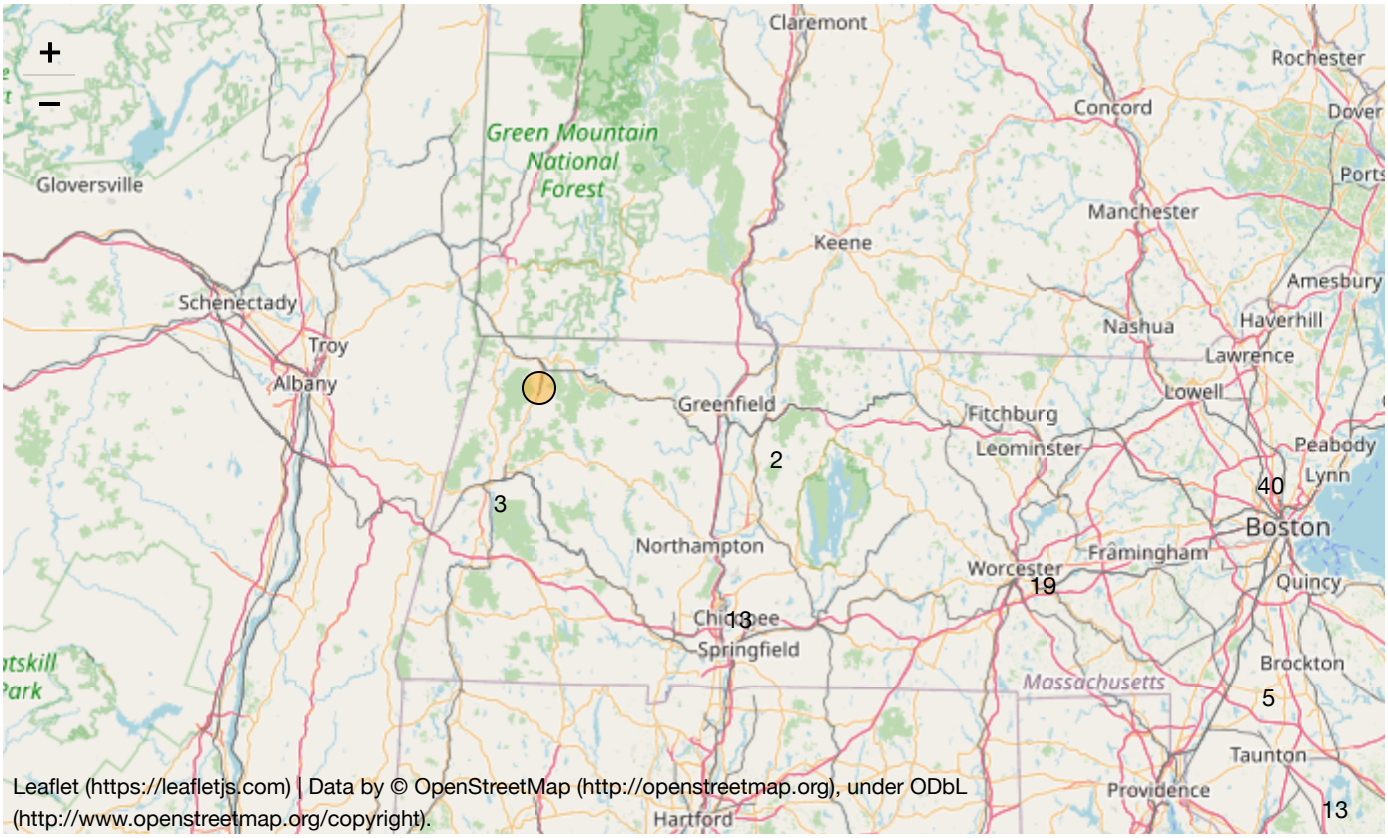
5. Here are all the facilities in this program

In [46]:

```
program = data_sets[ data_set_widget.value ]
program_data = None
key=dict() # Create a way to look up Registry IDs in ECHO_EXPORTER later

# We need to provide a custom list of program ids for some programs.
if ( program.name == "Air Inspections" or program.name == "Air Enforcements" ):
    # The REGISTRY_ID field is the index of the echo_data
    registry_ids = echo_data[echo_data['AIR_FLAG'] == 'Y'].index.values
    key = { i : i for i in registry_ids }
    program_data = program.get_data( ee_ids=registry_ids )
elif ( program.name == "Combined Air Emissions" ):
    ghg_registry_ids = echo_data[echo_data['GHG_FLAG'] == 'Y'].index.values
    tri_registry_ids = echo_data[echo_data['TRI_FLAG'] == 'Y'].index.values
    id_set = np.union1d( ghg_registry_ids, tri_registry_ids )
    registry_ids = list(id_set)
    program_data = program.get_data( ee_ids=registry_ids )
    key = { i : i for i in registry_ids }
elif ( program.name == "Greenhouse Gases" or program.name == "Toxic Releases" ):
    program_flag = program.echo_type + '_FLAG'
    registry_ids = echo_data[echo_data[ program_flag ] == 'Y'].index.values
    program_data = program.get_data( ee_ids=registry_ids )
    key = { i : i for i in registry_ids }
else:
    ids_string = program.echo_type + '_IDS'
    ids = list()
    registry_ids = list()
    for index, value in echo_data[ ids_string ].items():
        try:
            for this_id in value.split():
                ids.append( this_id )
                key[this_id]=index
        except ( KeyError, AttributeError ) as e:
            pass
    program_data = program.get_data( ee_ids=ids )

# Find the facility that matches the program data, by REGISTRY_ID.
# Add lat and lon, facility name and REGISTRY_ID as fac_registry_id.
# (Note: not adding REGISTRY_ID right now as it is sometimes interpreted as an int a
my_prog_data = pd.DataFrame()
no_data_ids = []

# Look through all the facilities in my area and program and get supplemental echo_d
if (program_data is None): # Handle no data
    print("Sorry, we don't have data for this program! That could be an error on our
else:
    for fac in program_data.itertuples():
        fac_id = fac.Index
        reg_id = key[fac_id] # Look up this facility's Registry ID through its Progr
        try:
            echo_row = pd.DataFrame(echo_data.loc[reg_id].copy()).T.reset_index() #
            echo_row = echo_row[['FAC_NAME', 'FAC_LAT', 'FAC_LONG']] # Keep only the
            program_row =  pd.DataFrame([list(fac)[1:]], columns=program_data.column
            full_row = pd.concat([program_row, echo_row], axis=1) # Join the EE row
            frames = [my_prog_data, full_row]
            my_prog_data = pd.concat( frames, ignore_index=False)
        except KeyError:
            # The facility wasn't found in the program data.
            no_data_ids.append( fac.Index )
```

In [47]:
```
# in ordert to map, roll up my_prog_data to facility level
fac = my_prog_data.drop_duplicates(subset=['PGM_SYS_ID']) # or whatever the key is
map_of_facilities = mapper_marker(fac)
map_of_facilities
```

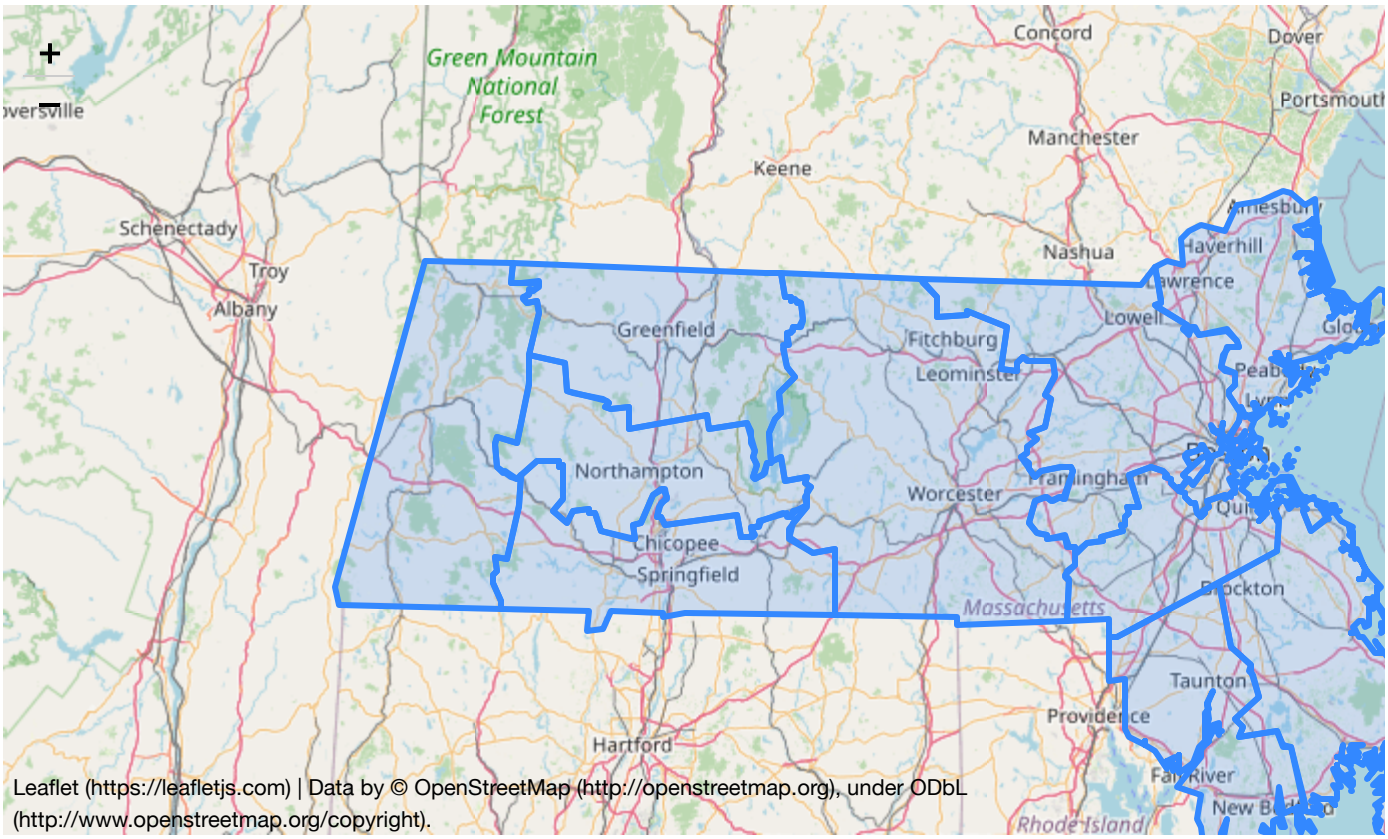
Out[47]:



6. Here are the geographies we're going to summarize this information at

In [48]:
```
# read in and map geojson for the selected geography
geo = "county" #select_region_widget.value.lower()
geo_json_data = geopandas.read_file("ECHO-Geo/ma_"+geo+".geojson")

m = folium.Map(
    #tiles='Mapbox Bright',
)
folium.GeoJson(
    geo_json_data,
).add_to(m)

bounds = m.get_bounds()
m.fit_bounds(bounds)

m
```

Out[48]:



7. Now we bring the geographic data and the facility data together. First, let's rank each geography.

In [49]:
```
# first, spatialize my_prog_data
gdf = geopandas.GeoDataFrame(
    my_prog_data, geometry=geopandas.points_from_xy(my_prog_data["FAC_LONG"], my_pro

join = geopandas.sjoin(gdf, geo_json_data, how="inner", op='intersects')

# get geo and attribute data column names
geo_column = {"county": "COUNTY"} # EXPAND
att_column = {"Greenhouse Gases": "ANNUAL_EMISSION"} # EXPAND
g = geo_column[geo]
a = att_column[program.name]

test = join.groupby(join[g])[[a]].agg("sum")
test.sort_values(by=a, ascending = False)
```

```
/Users/enost/anaconda3/lib/python3.7/site-packages/geopandas/tools/sjoin.py:61: Us
erWarning: CRS of frames being joined does not match!(None != epsg:4326)
  "(%s != %s)" % (left_df.crs, right_df.crs)
```

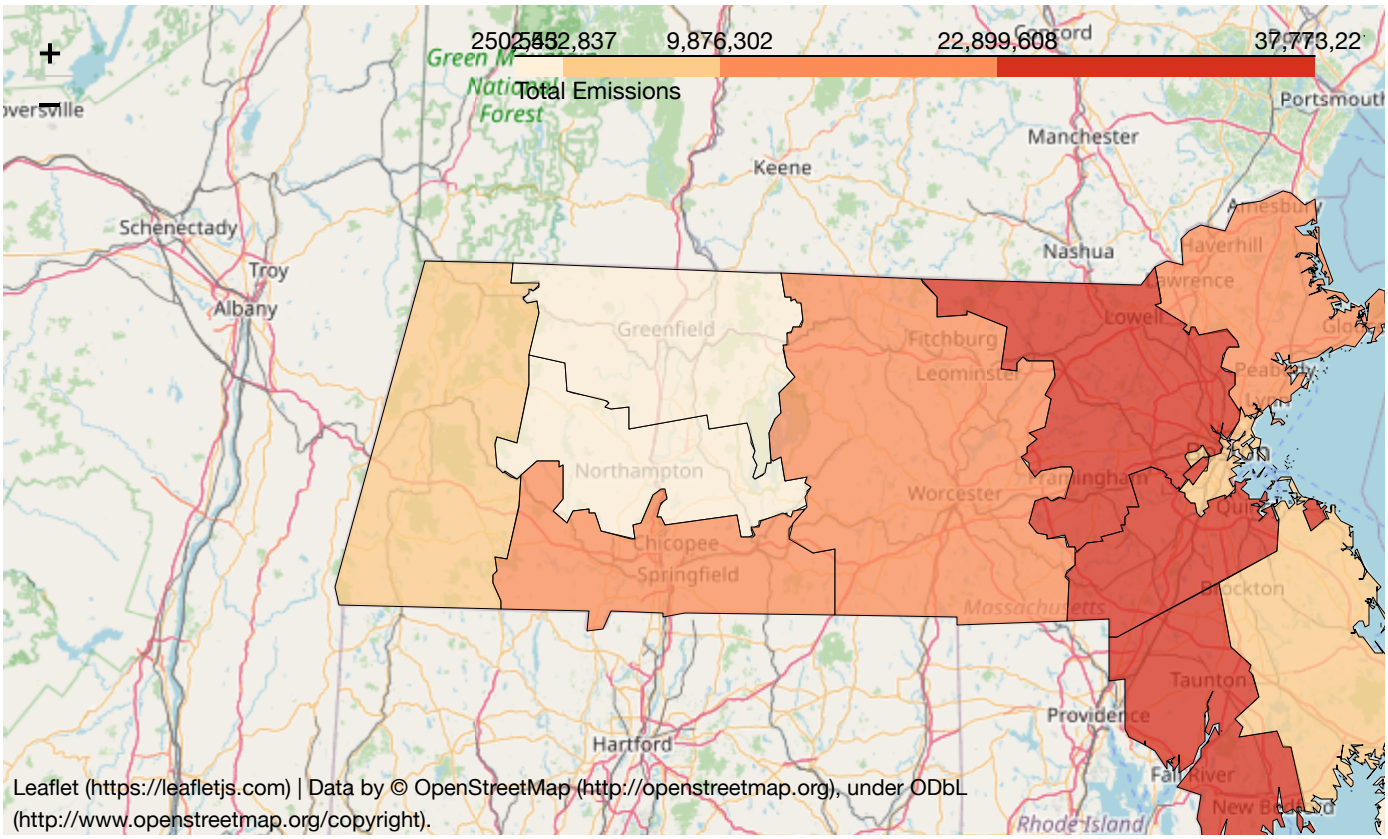
Out[49]:

COUNTY	ANNUAL_EMISSION
MIDDLESEX	3.777322e+07
BRISTOL	2.638697e+07
NORFOLK	2.533697e+07
WORCESTER	2.208715e+07
ESSEX	1.787330e+07
HAMPDEN	1.073491e+07
PLYMOUTH	9.017692e+06
SUFFOLK	5.148998e+06
BERKSHIRE	2.899780e+06
HAMPSHIRE	1.512007e+06
BARNSTABLE	1.159464e+06
FRANKLIN	2.505429e+05

8. Now, let's map it!

In [50]:
```
test.reset_index(inplace=True)
att_data = test.rename(columns={g: "geo", a: "value"})
mp = mapper_area(geo_json_data, att_data, g)
mp
```

Out[50]:



9. Rank individual facilities

In [51]:
```
ranked = my_prog_data.groupby(["PGM_SYS_ID", "FAC_NAME", "FAC_LAT", "FAC_LONG"])[[a]
ranked.reset_index(inplace=True)
ranked = ranked.set_index("PGM_SYS_ID")
ranked.sort_values(by=a, ascending=False)
```

Out[51]:

PGM_SYS_ID	FAC_NAME	FAC_LAT	FAC_LONG	ANNUAL_EMISSION
1000653	CONSTELLATION MYSTIC GENERATING STATION	42.390500	-71.067300	2.358097e+07
1007239	DOMINION ENERGY BRAYTON POINT POWER PLANT	41.709989	-71.192441	2.198041e+07
1001410	FORE RIVER GENERATING STATION	42.241669	-70.965851	1.404869e+07
1005710	SEMASS RESOURCE RECOVERY FACILITY	41.802300	-70.787500	8.307959e+06
1006864	ANP BELLINGHAM POWER PLANT	42.109971	-71.452954	7.081364e+06
1006657	ANP BLACKSTONE ENERGY GENERATING PLANT	42.059776	-71.515203	6.933836e+06
1001307	MILLENNIUM POWER PLANT	42.112351	-72.015097	6.183316e+06
1000657	GENON KENDALL, LLC	42.363464	-71.079669	5.658642e+06
1005179	COVANTA RESOURCE RECOVERY FACILITY	42.765400	-71.124025	4.971544e+06
1006267	MILLBURY RESOURCE RECOVERY FACILITY	42.220700	-71.767300	4.212732e+06
1004101	NESWC RESOURCE RECOVERY FACILITY	42.726075	-71.122203	4.001312e+06
1004287	WHEELABRATOR WASTE TO ENERGY PLANT	42.447211	-70.980472	3.305704e+06
1001298	BERKSHIRE POWER PLANT	42.048067	-72.647927	3.248573e+06
1002481	NATIONAL GRID CORPORATE HEADQUARTERS	42.396463	-71.271237	3.235105e+06
1007435	MASSPOWER COGENERATION FACILITY	42.156979	-72.522369	2.899814e+06
1000661	SALEM HARBOR STATION	42.525500	-70.877000	2.894193e+06
1001294	DIGHTON POWER PLANT	41.831268	-71.124005	2.708572e+06
1001207	BELLINGHAM COGENERATION FACILITY	42.093255	-71.481610	2.447390e+06
1000580	MEDICAL AREA TOTAL ENERGY PLANT	42.336667	-71.108333	2.121691e+06
1001289	MIT COGENERATION PLANT	42.360920	-71.093260	1.227893e+06
1006775	MILFORD POWER PLANT	42.128148	-71.514298	1.224417e+06
1000056	SPECIALTY MINERALS, INC.	42.643200	-73.113500	1.216708e+06
1002299	SOLUTIA CHEMICAL MANUFACTURING PLANT	42.154981	-72.526419	1.191962e+06
1000092	VEOLIA - STEAM GENERATING PLANT	42.349750	-71.057967	1.123345e+06
1000658	GENON CANAL GENERATING STATION	41.769800	-70.509100	1.042298e+06
1001277	NSTAR DBA EVERSOURCE ENERGY	42.204460	-71.159370	9.796276e+05
1005136	UMASS HEATING PLANT	42.389978	-72.537008	9.677601e+05
1004865	COVANTA RESOURCE RECOVERY PLANT	42.090524	-72.590858	9.140522e+05
1005731	PITTSFIELD GENERATING POWER PLANT	42.454880	-73.217340	9.089874e+05
1000659	HOLYOKE WATER POWER MOUNT TOM STATION	42.280600	-72.605400	8.777434e+05
...	...	...	...	...
1005037	ERVING PAPER MILLS	42.600080	-72.378380	2.505429e+05
1006452	LOGAN INTERNATIONAL AIRPORT	42.368120	-71.009410	2.472771e+05

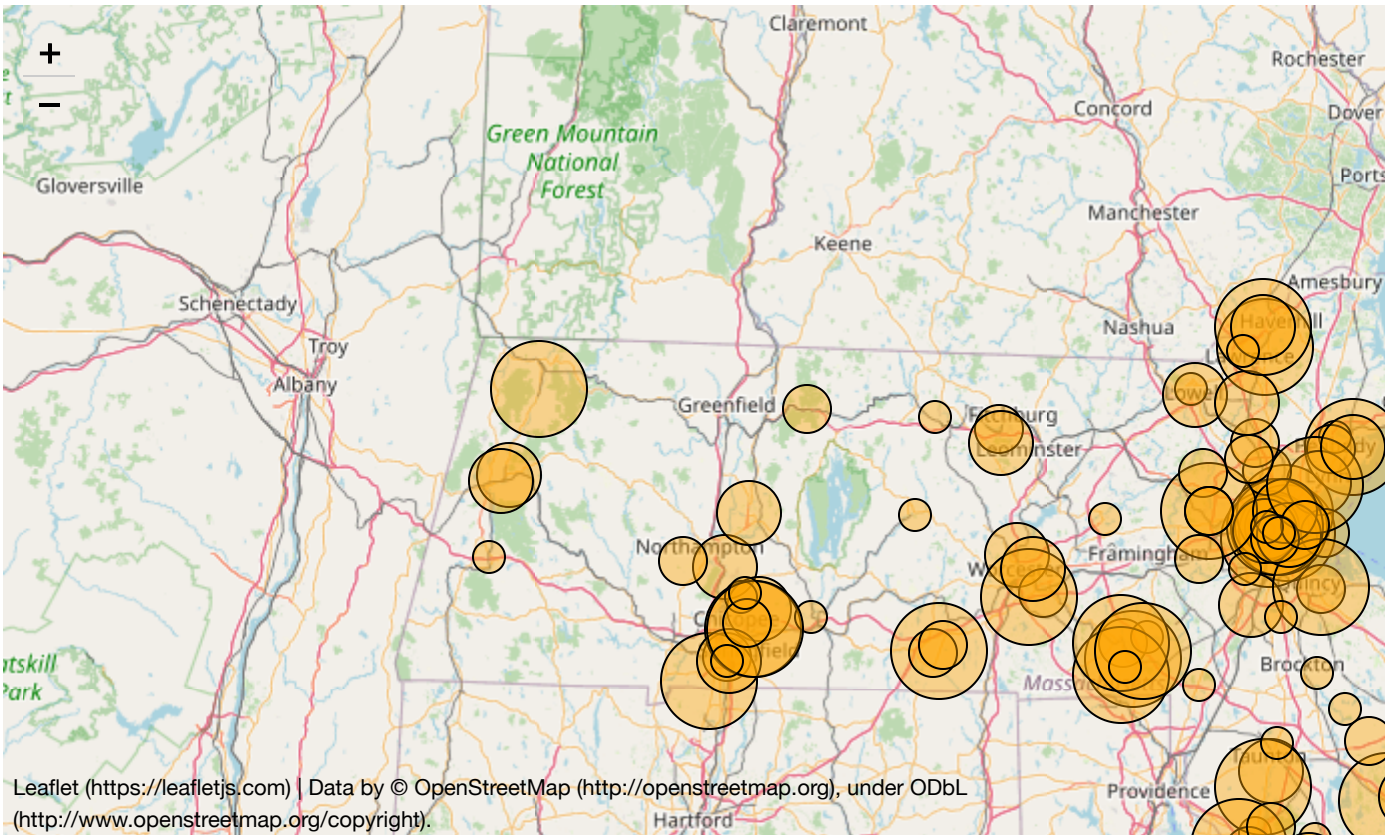
PGM_SYS_ID	FAC_NAME	FAC_LAT	FAC_LONG	ANNUAL_EMISSION
1009741	NATIONAL GRID CORPORATE HEADQUARTERS	42.396463	-71.271237	2.470448e+05
1010391	MIDDLEBOROUGH LANDFILL & TRAN	41.928430	-70.834540	2.363897e+05
1007547	FALL RIVER LANDFILL	41.752400	-71.105200	1.899989e+05
1002142	KRAFT ATLANTIC GELATIN PLANT	42.477336	-71.115135	1.895543e+05
1005473	OLDCASTLE STONE PRODUCTS	42.300790	-73.250840	1.855416e+05
1009856	INTEL FAB 17 SEMICONDUCTOR MANUFACTURING	42.379380	-71.556960	1.806738e+05
1007405	POLARTEC	42.717020	-71.179900	1.619111e+05
1003532	TAUNTON LANDFILL & GAS ENERGY RECOVERY	41.922990	-71.086477	1.547546e+05
1007948	BONDIS ISLAND LANDFILL	42.091040	-72.597480	1.447948e+05
1004958	BOURNE LANDFILL & TRANSFER STATION	41.731171	-70.583757	1.171668e+05
1004157	SOUTH HADLEY LANDFILL & RECYCLING CTR	42.219930	-72.556870	9.532115e+04
1007947	BFI RANDOLPH LANDFILL	42.180149	-71.076812	8.392150e+04
1007685	HOLYOKE SANITARY LANDFILL	42.227281	-72.547189	7.991847e+04
1003587	LANDFILL & GAS GENERATING FACILITY	42.387400	-72.079490	7.974464e+04
1005039	CRAPO HILL LANDFILL	41.724511	-70.984750	6.974125e+04
1010499	TENNESSEE GAS PIPELINE	42.078040	-71.506570	6.872380e+04
1009897	PRUDENTIAL CENTER	42.348630	-71.082730	6.275700e+04
1000655	EXELON WEST MEDWAY GENERATING STATION	42.139997	-71.446348	6.071122e+04
1005466	GARDNER STREET LANDFILL	42.277914	-71.173505	5.348625e+04
1000666	PEABODY MUNICIPAL LIGHT PLANT	42.543216	-70.928979	5.246158e+04
1007553	HARVARD BLACKSTONE STEAM PLANT	42.364290	-71.114800	4.950618e+04
1004557	PLAINVILLE LANDFILL	42.039162	-71.299930	4.378925e+04
1004556	GAS RECOVERY SYSTEMS	42.064180	-70.979535	3.160775e+04
1007550	HALIFAX LANDFILL	41.992300	-70.897890	1.328300e+04
1001230	LOWELL COGENERATION PLANT	42.640140	-71.322470	6.654268e+03
1003393	GARDNER LANDFILL	42.587200	-72.024700	4.729500e+03
1000660	SOMERSET POWER GENERATING STATION	41.737800	-71.145300	1.591991e+03
1004019	MUSTANG MOTORCYCLE PRODUCTS	42.179590	-72.366746	5.676420e+02

96 rows × 4 columns

10. Map individual facilities

In [52]:
```
ranked['quantile'] = pd.qcut(ranked[a], 4, labels=False)
mp = mapper_circle(ranked, a)
mp
```

Out[52]:



In [ ]: