CS4132 Data Analytics

An Analysis of The Global Light Pollution Standards

Prannaya Gupta (M21404)

*Done as part of **CS4232**: Data Analytics*

Table of Contents

Introduction

A side-effect of technological advancement has been the amount of light pollution in the world today. Ever since Thomas Edison's revolutionary invention of the light bulb, the world has been thrust into a landscape of light-afflicted skies. According to various studies, around 80% of people live under light pollution-afflicted skies every day, and whilst this may not affect the day-to-day life of an individual, astronomers are very much affected by the sudden illumination in the skies. Even the Singaporean sky is very much damaged by light pollution, with 99.5% of all stars being completely invisible without optical aid[1].

Credits: Bing Hui Yau, Unsplash

In this project, I aim to analyze the implications of the changes in Light Pollution levels over the past few years, utilising global and local data to find possibly relationships. As shown below, I aim to solve a list of Light Pollution-related questions that utilise alternative data to analyse and interpret patterns, including simple statistical and machine-learning related models.

Research Questions

A. What are locations of minimal light pollution intensity which are optimum for astronomical observation?

While one might think that the best locations are in the middle of wilderness or large water bodies (i.e. the ocean), as are where the observatories Arecibo and FAST are situated, some of these locations need to be filtered based on accessibility, especially since locations like the middle of the ocean are not feasible locations for people to assemble to watch and will thus not be a great location for astrophotography. Additionally, we must also be able to identify locations for observatories to be located.

In order to locate these, a plethora of factors are considered, like the amount of light pollution in the vicinity, the education level around that area (on the basis of the rank of the top university situated in the country, area available and population density in the area, and some computations indicate that the Auckland Islands are the best locations for astronomical observation, especially in the context of placing an observatory in the area.

B. How has the light pollution data around the world changed? Which countries are most susceptible to high light pollution in the future? Which countries are lessening in terms of light pollution?

There are many countries that have had rampant increases in Light Pollution over the past few years, while others have made an effort to reduce their already increased Light Pollution. We need to map the data in order to find out which countries are susceptible to the level of problematic light pollution that can be found in cities like Singapore.

We use the Time Series Data available and model possible time series progressions, worldwise and in individual countries. This can be used to make predictions and compare to other light pollution levels.

In the end, we find that there was a sudden increase in Light Pollution back in 2014, Palestine, Qatar and Gibraltar are the most susceptible to light pollution, and some areas like Bermuda have lessening amounts of Light Pollution.

C. What is the relation between the average energy consumption and general demographics in each region/country and the Light Pollution?

A reason for considering this is the lack of mathematical analysis as to how energy consumption in specific districts affect the light pollution there. General mathematics regarding data in Singapore itself hasn't been explored, hence using this is a good way of exploring something new. We can model the types of houses, general types of people living there, and the energy consumption based on that and from there, we can test how it affects nearby light pollution.

I intend to use the Economical Datasets that give data regarding the Singapore Energy Consumption statistics[20] and the Singapore Resident demographics[21]. The usage of the Absolute positional brightness datasets[4-7], filtering Singapore similarly to the previous research question. This can therefore be using in conjunction to find possible patterns.

Dataset

Numbered list of dataset (with downloadable links) and a brief description of each dataset used. Draw reference to the numbering when describing methodology (data cleaning and analysis).

In this project, we are using the following datasets:

1. The GeoNames Geographical Dataset
2. The Google Developers `countries.csv` Dataset
3. DataHub's Countries GeoJSON Dataset
4. DataHub's Natural Earth Polygons GeoJSON Dataset
5. UN's Population, Surface Area and Density Dataset
6. UN's Dataset of Population Growth and Indicators of Fertility and Mortality
7. UN's Dataset for Literacy Amongst Students
8. UN's Dataset for Labour Force and Unemployment
9. UN's Dataset for Employment by Industry
10. UN's Dataset for Energy Consumption
11. UN's Dataset for Educational Attainment
12. UN's Dataset for Population by Age, Sex, Educational Attainment
13. The Harmonized Global Nighttime Light (1992 - 2018) Dataset
14. The Globe at Night - Sky Brightness Monitoring Network (GaN-MN)

Set-Up and Imports

Firstly, we need to perform a simple set-up. This involves the following steps:

1. Import the `autoreload` extension which can allow us to avoid leaving Jupyter at a stagnant file state.
2. Install some libraries, like `np`, `imagecodecs`, `lxml` and `gdal`
   - **Note**: For gdal, which is the Python API Wrapper for the C library GDAL, the installation is a lot more complex than just pip install, hence there is a script. Additional info is provided in the corresponding section
3. Import libraries, which include:
   - Crucial Ones like `random`, `re` and `glob`
   - Mathematical Libraries like `numpy` and `scipy`
   - Data Wrangling Libraries like `pandas` and `lxml`
   - Plotting Libraries like `matplotlib`, `seaborn` and `plotly`
   - Web Scraping Libraries like `bs4` and `requests`
   - Image Processing Libraries like `cv2`, `skimage` and `PIL`
   - Machine Learning Libraries like `sklearn`

Installation Instructions

In this project, I am using the following external libraries:

- `scikit-image` : download using conda if possible.
- `pillow` : predownloaded in conda, install using pip if you are using a default python environment
- `opencv-python` : install using pip in any case
- `lxml` : install using pip if possible
- `geopy` : install using pip to avoid installation bugs
- `geopandas` : install using conda if possible
- `rasterio` and `rasterstats` : install using conda if possible
- `keplergl` : install using pip if possible
- `leafmap` : install using conda if possible
- `geojson-area` : install using pip if possible.

A sample installation script is shown below:

```
. '/c/ProgramData/Anaconda3/etc/profile.d'/conda.sh
conda activate
conda create --name data-analytics python==3.8
conda activate data-analytics
pip install opencv-python lxml geopy
conda install scikit-image geopandas rasterio rasterstats --yes
conda install leafmap -c conda-forge
pip install keplergl area
```

Directory Structure

Please make the `data` directory as follows:

```
data
├──country
├──gan
├──ibol
├──nightLight
└──stats
```

Activating the `autoreload` extension

The `autoreload` extension is a IPython too that exists to ensure that the kernel resets when files change around it. This specifically helps for modules that exhibit significant changes when being used, and has mostly just been found experimentally. Hence, this is effectively just a debugging step.

In [1]:
```
%load_ext autoreload
%autoreload 2
```

Imports

Some crucial modules used in this project include:

- `numpy` and `scipy` for numerical analysis.
- `pandas` and `geopandas` for storing geospatial and non-geospatial data.
- `matplotlib` and `seaborn` for plotting static plots.
- `bs4` for web scraping for data acquisition.
- `scikit-learn` for data modelling.
- `folium` and `plotly` for plotting interactive geospatial data.
- `rasterio` and `rasterstats` for analyses of raster files.

In [2]:
```
import random, time, math, datetime, os, re
from glob import glob
import sys, tarfile, gzip
from pprint import pprint
from functools import reduce

from IPython.display import clear_output, HTML, Markdown
from IPython import display

import warnings
warnings.filterwarnings("ignore")

import np, numpy
import pandas as pd
from lxml import objectify, etree, html
from io import StringIO, BytesIO

import scipy as sp
from scipy import stats, special

import matplotlib as mpl
import matplotlib.patches as mpl_patches
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams["figure.figsize"] = (16.0, 8.0)

from bs4 import BeautifulSoup
import requests, bs4

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, mean_squared_error
from sklearn.linear_model import LinearRegression

import folium
from folium import plugins
import plotly.express as px

import geopandas as gp
import rasterio as rs
import rasterstats as rstats

from area import area as geojson_area
```

Definitions

This section pertains to tools created with the express purpose of mathematical and image-based analyses.

Here, you can find the following:

1. Correlation Computational Methods
2. Custom Regression Lines
3. Pandas-based customized data reading functions

Correlation Computational Functions

Just to ensure that the correlation computation is substantially accurate, I am also using a self-developed system to identify correlation.

Gamma Function

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$$

Based on some simple manipulation, we get the following equations:

$$\Gamma(x) = \frac{1}{x} t^x e^{-t}|_0^\infty - \frac{1}{x} \int_0^\infty t^x (-e^{-t}) dt$$

$$= 0 + \frac{1}{x} \int_0^\infty t^x (e^{-t}) dt$$

$$= \frac{1}{x} \int_0^\infty t^x (e^{-t})\, dt = \frac{1}{x} \Gamma(x+1)$$

$$\implies \Gamma(x+1) = x\Gamma(x)$$

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$$

$$\implies \Gamma(1) = \int_0^\infty t^{1-1} e^{-t} dt = \int_0^\infty e^{-t}\, dt = (-e^{-t})|_0^\infty = 1$$

Therefore, we can see the following pattern:

$$\Gamma(1+1) = (1) \times \Gamma(1) = 1 \implies \Gamma(2) = 1!$$

$$\Gamma(1+2) = (2) \times \Gamma(2) = 2 \implies \Gamma(3) = 2!$$

$$\Gamma(1+3) = (3) \times \Gamma(3) = 6 \implies \Gamma(4) = 3!$$

$$\vdots$$

$$\Gamma(n+1) = n! \implies \Gamma(n) = (n-1)!$$

Since the actual factorial only takes in integers, we must introduce a gamma function with 1/2.

Hence, we can compute some more to get

$$\Gamma(\frac{1}{2}) = \sqrt{\pi}$$

Beta Function

$$\beta(x, y) = \int_0^1 t^{x-1} (1-t)^{y-1} dt = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$$

Incomplete Beta Function

The **incomplete beta function**, a generalization of the beta function, is defined as

$$B(x;\, a, b) = \int_0^x t^{a-1}\, (1-t)^{b-1}\, dt$$

Regularised Incomplete Beta Function

From this, we also introduce the concept of the **regularised incomplete beta function**, as shown below

$$I_x(a, b) = \frac{B(x;\, a, b)}{\beta(a, b)}$$

Some, actual values are as below:

$$I_0(a, b) = 0$$

$$I_1(a, b) = 1$$

$$I_x(a, 1) = x^a$$

$$I_x(1, b) = 1 - (1-x)^b$$

$$I_x(a, b) = 1 - I_{1-x}(b, a)$$

$$I_x(a, b) = I_x(a-1, b) - \frac{x^{a-1}(1-x)^b}{(a-1)\beta(a-1, b)}$$

$$I_x(a, b) = I_x(a, b-1) + \frac{x^a(1-x)^{b-1}}{(b-1)\beta(a, b-1)}$$

From here, you can define a simplified recursive function to compute the regularised incomplete beta function.

The Measure of Covariance

Covariance is used to determine how much the variables differ from their means.

First off, let take a dataset which has been normalised, i.e

$$\bar{x} = \bar{y} = 0$$

This can be achieved by subtracting each values by the mean, i.e

$$x_i := x_i - \bar{x}$$

$$y_i := y_i - \bar{y}$$

We can then compute this Covariance using:

$$cov(x, y) = \frac{1}{n} \sum_{i=1}^{n} x_i y_i$$

Pearson R Correlation Coefficient

First off, let take a dataset which has been normalised, i.e

$$\bar{x} = \bar{y} = 0$$

This can be achieved by subtracting each values by the mean, i.e

$$x_i := x_i - \bar{x}$$

From here, computing the Pearson R Correlation Coefficient is not very intensive.

$$r(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}}$$

The p Value

From here, we can also compute the P Value using the previous two functions. It is simply given by:

$$p(r) = 2I_{\frac{1-|r|}{2}}(\frac{n}{2} - 1, \frac{n}{2} - 1)$$

where n is the number of samples.

The Spearman Rank Correlation Coefficient

The Spearman Rank Correlation Cofficient is used to find correlation between specifc ranks in data. Below, we use the tied rank method, which simply finds the ranks of the actual data and then computes the Pearson Correlation Coefficient of these ranks.

Thus, the formula is as follows:

$$\rho(x, y) = r(rank(x), rank(y))$$

In [3]:

```
def gamma(n):
    """
    A Simplified Gamma Function for "powers" of 1/2
    Gamma Function for Integers are given by: Γ(n) = (n-1)!
    Gamma Function for Values ending with 1/2: Γ(n) = 0.5 * 1.5 * ... * (n-1) * √(π)

    Gamma Function is generally given by the following expression
    """
    return np.arange(1 - (n%1), n).prod() * np.pi ** (n%1)

def beta(x, y):
    """
    Actual Beta Function based ont he Simplified Gamma Function above.
    """
    return gamma(x)*gamma(y)/gamma(x+y)

def I(x, a, b):
    if gamma(a) == np.inf or gamma(b) == np.inf or gamma(a+b) == 0: return 0
    if x == 0 or x == 1: return x
    if b == 1: return x ** a
    if a == 1: return 1 - (1-x)**b
    if b > 1: return I(x, a, b-1) + (x**a * (1-x)**(b-1))/((b-1)*beta(a, b-1))
    if a > 1: return I(x, a-1, b) - (x**(a-1) * (1-x)**b)/((a-1)*beta(a-1, b))
    return ((-1)**a) * I(x/(x-1), a, 1-a-b) * beta(a, 1-a-b)


def corr(data):
    norm = data - data.mean(axis=0)
    return norm.prod(axis=1).sum() / np.sqrt((norm**2).sum(axis=0).prod())

def p(r, bound):
    return 2*I((1-abs(r))/2, bound, bound)

def pearson(data):
    r = corr(data)
    return r, p(r, data.shape[0]/2-1)

def pearsonr(x, y):
    x, y = x - x.mean(axis=0), y - y.mean(axis=0)
    r = (x.T @ y).sum(axis=0) / np.sqrt(((x**2).sum(axis=0) * (y**2).sum(axis=0)))
    p_val = 2 * np.vectorize(lambda r: p(r, x.shape[0]/2-1))(r)
    return np.stack((r, p_val), axis=len(r.shape))


def rank(x):
    return pd.Series(x).rank().to_numpy()

def spearman(x, y):
    return pearsonr(rank(x), rank(y))
    #return 1 - (6*((rank(x) - rank(y))**2).sum() / ((lambda x: x**3 - x)(len(x))))


def rankedCorrelation(df):
    return (lambda dt: dt.loc[np.abs(dt.Correlation).sort_values(ascending=False).index])((lambda corr: pd.DataFrame(corr[corr.index.str.contains("Light") & (~corr.index.str.conta
```

Regression Lines

In this section, we adopt some alternative practices to incorporate Maclaurin and Taylor Series fitting methods into this project.

An Introduction to Taylor Series

Whilst we do know that Maclaurin Series operates based on the following simplistic approximation:

$$f(x) \approx \sum_{r=0}^{N} \frac{f^{(r)}(0)}{r!} x^r$$

where N is sufficiently large number.

We can also simplify this expression by placing limit as below:

$$f(x) = \lim_{N \to \infty} \sum_{r=0}^{N} \frac{f^{(r)}(0)}{r!} x^r = \sum_{r=0}^{\infty} \frac{f^{(r)}(0)}{r!} x^r$$

Let's thus consider that we apply this operation for the function

$$g(x) = \frac{1}{1+x}$$

From here, we can get some very simple values as shown below:

$$
\begin{aligned}
g^{(0)}(x) &= (1+x)^{-1} \\
g^{(1)}(x) &= g'(x) \\
&= \frac{d}{dx}\left[(1+x)^{-1}\right] \\
&= (-1)\cdot(1+x)^{-2} \\
&= -(1+x)^{-2} \\
g^{(1)}(x) &= (-1)^1 \cdot 1! \cdot (1+x)^{-2}
\end{aligned}
$$

$$
\begin{aligned}
g^{(2)}(x) &= g''(x) \\
&= \frac{d^2}{dx^2}\left[(1+x)^{-1}\right] \\
&= \frac{d}{dx}\left[-(1+x)^{-2}\right] \\
&= (-1)\cdot(-2)\cdot(1+x)^{-3} \\
&= 2(1+x)^{-3} \\
g^{(2)}(x) &= (-1)^2 \cdot 2! \cdot (1+x)^{-3}
\end{aligned}
$$

$$
\begin{aligned}
g^{(3)}(x) &= g'''(x) \\
&= \frac{d^3}{dx^3}\left[(1+x)^{-1}\right] \\
&= \frac{d}{dx}\left[2(1+x)^{-3}\right] \\
&= (2)\cdot(-3)\cdot(1+x)^{-4} \\
&= -6(1+x)^{-4} \\
g^{(3)}(x) &= (-1)^3 \cdot 3! \cdot (1+x)^{-4}
\end{aligned}
$$

Hence, we derive the following expression:

$$g^{(k)}(x) = (-1)^k \cdot k! \cdot (1+x)^{-k-1}$$

Hence, for any k, we get the following:

$$g^{(k)}(0) = (-1)^k \cdot k!$$

Thus, we get the following:

$$
\begin{aligned}
g(x) &= \sum_{r=0}^{\infty} \frac{g^{(r)}(0)}{r!} x^r \\
&= \sum_{r=0}^{\infty} \frac{(-1)^r \cdot r!}{r!} x^r \\
\frac{1}{1+x} &= \sum_{r=0}^{\infty} (-x)^r \\
\frac{1}{1+x} &= \sum_{r=0}^{\infty} [1-(1+x)]^r
\end{aligned}
$$

From here, we define a value $y = 1 + x$. Thus, we can get the following:

$$\frac{1}{y} = \sum_{r=0}^{\infty} (1-y)^r$$

Thus, we get the Maclaurin Series of $f(x) = \frac{1}{x}$ to be as follows:

$$f(x) = \frac{1}{x} = \sum_{r=0}^{\infty} (1-x)^r = \sum_{r=0}^{\infty} (-1)^r (x-1)^r$$

However, the introduction of the alternate variable proffers us a solution to solve this value in the case where $f(0)$ and subsequent derivatives do not exist. Hence, we can introduce a term a such that the following is valid:

$$f(x) = \sum_{r=0}^{\infty} \frac{f^{(r)}(a)}{r!} (x-a)^r$$

This is the definition of the **Taylor Series**, wherein we generalise Maclaurin Series itself to be a form of Taylor Series such that $a = 0$. However, when $a \neq 0$, Taylor Series is used.

For example, applying $a = 1$ on the function f mentioned above, we can derive the Taylor Series of f. Firstly, we notice the following:

$$f^{(k)}(1) = (-1)^f \cdot f!$$

This piece of information has not changed from function g. Hence, we can now apply the Taylor Series:

$$
\begin{aligned}
f(x) &= \sum_{r=0}^{\infty} \frac{f^{(r)}(1)}{r!} (x-1)^r \\
&= \sum_{r=0}^{\infty} \frac{(-1)^r \cdot r!}{r!} (x-1)^r \\
\frac{1}{x} &= \sum_{r=0}^{\infty} (1-x)^r \\
f(x) &= \sum_{r=0}^{\infty} (-1)^r (x-1)^r
\end{aligned}
$$

Hence, for a function $f(x)$, to be expanded about a point $x = a$, let us define a new function $g(x)$ such that $g(x-a) = f(x)$. Then,

$$g(x) = g(0) + g'(0)x + \frac{1}{2}g''(0)x^2 + \dots$$

$$g(x) = f(a) + f'(a)x + \frac{1}{2}f''(a)x^2 + \dots$$

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \dots$$

Hence,

$$f(x) = \sum_{r=0}^{\infty} \frac{f^{(r)}(a)}{r!} (x-a)^r$$

for some a, such that f(a) exists.

Functions to compute the Maclaurin and Taylor derivations

Using `numpy`'s `polyfit` function, we can compute the coefficients of a function represented simply as a Taylor Series or Maclaurin Series Expansion, up to a degree of 100.

In [4]:
```
def maclaurin(x, y, N=100):
```

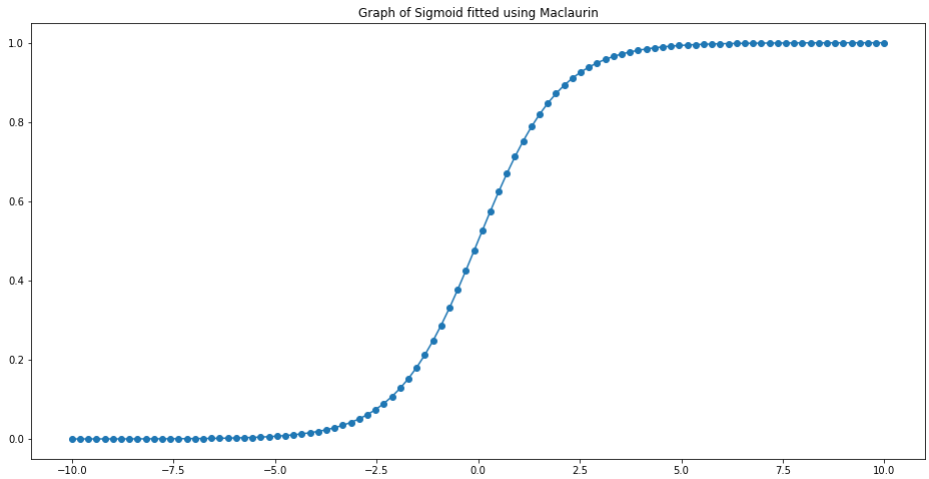
```
    return (np.polyfit(x, y, N)[:, np.newaxis] * (x ** np.arange(N, -1, -1)[:, np.newaxis])).sum(axis=0)

def taylor(x, y, a, N=100):
    return maclaurin((x-a), y, N)

def taylorplot(x, y, **kwargs):
    sns.lineplot(x, taylor(x, y, x.min()), ax=plt.gca())
```

Validation of these Methods

To prove that these methods work, we investigate based on two curves: the sigmoid curve used in Logistic Regresssion and the activation functions in Deep Learning, and the sine curve which can be used easily for Taylor Series.

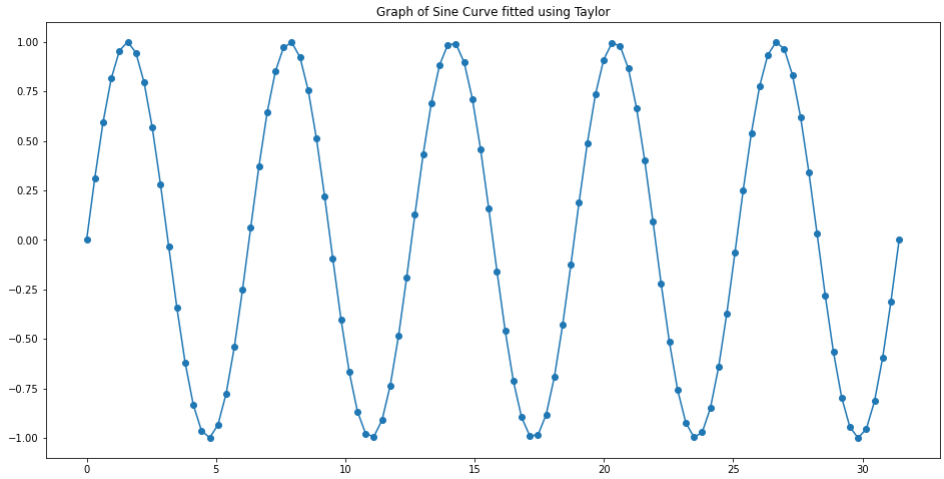
In [5]:
```
x = np.linspace(-10, 10, 100)
z = 1 / (1 + np.exp(-x))
plt.scatter(x, z)
plt.plot(x, maclaurin(x, z))
plt.title("Graph of Sigmoid fitted using Maclaurin")
```

Out[5]:
```
Text(0.5, 1.0, 'Graph of Sigmoid fitted using Maclaurin')
```



In [6]:
```
x = np.linspace(0,10*np.pi, 100)
y = np.sin(x)
plt.scatter(x, y)
plt.plot(x, taylor(x, y, np.pi))
plt.title("Graph of Sine Curve fitted using Taylor")
```

Out[6]:
```
Text(0.5, 1.0, 'Graph of Sine Curve fitted using Taylor')
```



`pandas` Extension Functions

This snippet contains functions to retrieve large online sources instead of simply relying on the pandas scraping function, which runs into some errors.

In [5]:
```
def read_csv(loc, *args, **kwargs):
    if re.search(r"[(http(s)?):\/\/(www\.)?a-zA-Z0-9@:%._\+~#=]{2,256}\.[a-z]{2,6}\b([-a-zA-Z0-9@:%_\+.~#?&//=]*)", loc):
        return read_csv(requests.get(loc, allow_redirects=True).content, *args, **kwargs)
    loc = loc.strip("\n")
    if re.search(r"[\n:<>\"/\|?*]", loc):
        return pd.read_csv(StringIO(loc), *args, **kwargs)
    else:
        return pd.read_csv(loc, *args, **kwargs)
```

Methodology

You should demonstrate the data science life cycle here (from data acquisition to cleaning to EDA and analysis etc). For data cleaning, be clear in which dataset (or variables) are used, what has been done for missing data, how was merging performed, explanation of data transformation (if any). If data is calculated or summarized from the raw dataset, explain the rationale and steps clearly.

Data Acquisition and Cleaning

I will be acquiring the following datasets in this project:

1. Country-Based Datasets
   - The GeoNames Geographical Dataset
   - The Google Developers `countries.csv` Dataset
   - DataHub's Countries GeoJSON Dataset
   - DataHub's Natural Earth Polygons GeoJSON Dataset
2. UN Datasets
   - UN's Population, Surface Area and Density Dataset
   - UN's Dataset of Population Growth and Indicators of Fertility and Mortality
   - UN's Dataset for Literacy Amongst Students

- UN's Dataset for Labour Force and Unemployment
- UN's Dataset for Employment by Industry
- UN's Dataset for Energy Consumption

3. Light Pollution Datasets
   - The Harmonized Global Nighttime Light (1992 - 2018) Dataset-Dataset)
   - The Globe at Night - Sky Brightness Monitoring Network (GaN-MN)-Dataset)
   - 4.

Country-Based Datasets

The GeoNames Geographical Dataset

The GeoNames geographical database covers all countries and contains over eleven million placenames that are available for download free of charge. The dataset contains some key information like the Continent, Area in km^2 and Population. I have renamed some columns for clarity and replaced some `pandas` reading errors, like NA, which depicts the continent North America, being read as a null value.

In [8]:
```
geocountries = pd.read_html(requests.get("https://www.geonames.org/countries/").content)[1].rename(columns={"ISO-3166alpha2":"CountryCode", "ISO-3166alpha3":"CountryCode3", "ISO-3
geocountries.to_csv("data/country/geocountries.csv", index=False)
```

In [6]:
```
geocountries = pd.read_csv("data/country/geocountries.csv")
geocountries.Continent.replace(np.nan, "NA", inplace=True)
geocountries
```

Out[6]:

	CountryCode	CountryCode3	NumericCountryCode	fips	Country	Capital	Area in km²	Population	Continent
0	AD	AND	20	AN	Andorra	Andorra la Vella	468.0	77006	EU
1	AE	ARE	784	AE	United Arab Emirates	Abu Dhabi	82880.0	9630959	AS
2	AF	AFG	4	AF	Afghanistan	Kabul	647500.0	37172386	AS
3	AG	ATG	28	AC	Antigua and Barbuda	St. John's	443.0	96286	NA
4	AI	AIA	660	AV	Anguilla	The Valley	102.0	13254	NA
...	...	...	...	...	...	...	...	...	...
245	YE	YEM	887	YM	Yemen	Sanaa	527970.0	28498687	AS
246	YT	MYT	175	MF	Mayotte	Mamoudzou	374.0	279471	AF
247	ZA	ZAF	710	SF	South Africa	Pretoria	1219912.0	57779622	AF
248	ZM	ZMB	894	ZA	Zambia	Lusaka	752614.0	17351822	AF
249	ZW	ZWE	716	ZI	Zimbabwe	Harare	390580.0	14439018	AF

250 rows × 9 columns

Google's Lat-Long Dataset For Countries Worldwide

Google Developers' `countries.csv` which contains Latitude-Longitude data for each country. This indentifies a plausible center for the country.

I have cleared some of the data by getting the latitude, longitude and country name and adapt it to help with the merge later.

In [10]:
```
latlong = pd.read_html(requests.get("https://developers.google.com/public-data/docs/canonical/countries_csv").content)[0][["name", "latitude", "longitude"]].rename(columns={"name"
latlong.to_csv("data/country/latlong.csv", index=False)
```

In [7]:
```
latlong = pd.read_csv("data/country/latlong.csv")
latlong
```

Out[7]:

	Country	latitude	longitude
0	Andorra	42.546245	1.601554
1	United Arab Emirates	23.424076	53.847818
2	Afghanistan	33.939110	67.709953
3	Antigua and Barbuda	17.060816	-61.796428
4	Anguilla	18.220554	-63.068615
...	...	...	...
240	Yemen	15.552727	48.516388
241	Mayotte	-12.827500	45.166244
242	South Africa	-30.559482	22.937506
243	Zambia	-13.133897	27.849332
244	Zimbabwe	-19.015438	29.154857

245 rows × 3 columns

Wikipedia's List of Countries and Dependencies by Population Density

Wikipedia is kind enough to have compiled a list of countries and dependencies by population density, which lists the number of people per square kilometer in some specific areas.

In [8]:
```
popden = pd.read_html(requests.get("https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population_density").content)[0]
popden.columns = popden.columns.map("".join)
popden = popden.drop(columns=["Areasqmi", "Density/sqmi", "Sources& DatesSources& Dates"]).rename(columns={"Country (or territory)Country (or territory)": "name", "PopulationPopul
popden.name = popden.name.apply(lambda name: name[:name.index("*")-1] if "*" in name else name[:name.index("(")-1] if "(" in name else name) #.str.contains("*")
popden
```

Out[8]:

	name	Population	Area	Density
0	Macau	631636	30.0	21055.0
1	Monaco	38682	2.0	19341.0
2	Singapore	5757499	716.0	8041.0
3	Hong Kong	7371730	1104.0	6677.0
4	Gibraltar	33718	6.0	5620.0
...	...	...	...	...
247	Mongolia	3170216	1564116.0	2.0
248	Pitcairn Islands	56	5.0	11.0
249	Falkland Islands	3234	12173.0	0.0
250	Svalbard and Jan Mayen	2655	62422.0	0.0
251	Greenland	56564	2166086.0	0.0

252 rows × 4 columns

DataHub's GeoJSON Datasets

DataHub's Countries GeoJSON Dataset

DataHub's Countries GeoJSON Dataset is a geodata data package providing geojson polygons for all the world's countries. The data comes from Natural Earth, a community effort to make visually pleasing, well-crafted maps with cartography or GIS software at small scale.

I have opened this dataset as a `geopandas.GeoDataFrame` object, as shown below. Following this, I have saved it in the formal definition, `world_countries.json` for later usage by `folium` and `plotly`.

In [12]:
```
countries_geojson = gp.read_file("https://datahub.io/core/geo-countries/r/countries.geojson").rename(columns={"ADMIN":"Country"}).replace("United States of America", "United States
countries_geojson.to_file("data/country/world_countries.json", driver='GeoJSON')
```

In [9]:
```
countries_geojson = gp.read_file("data/country/world_countries.json")
countries_geojson
```

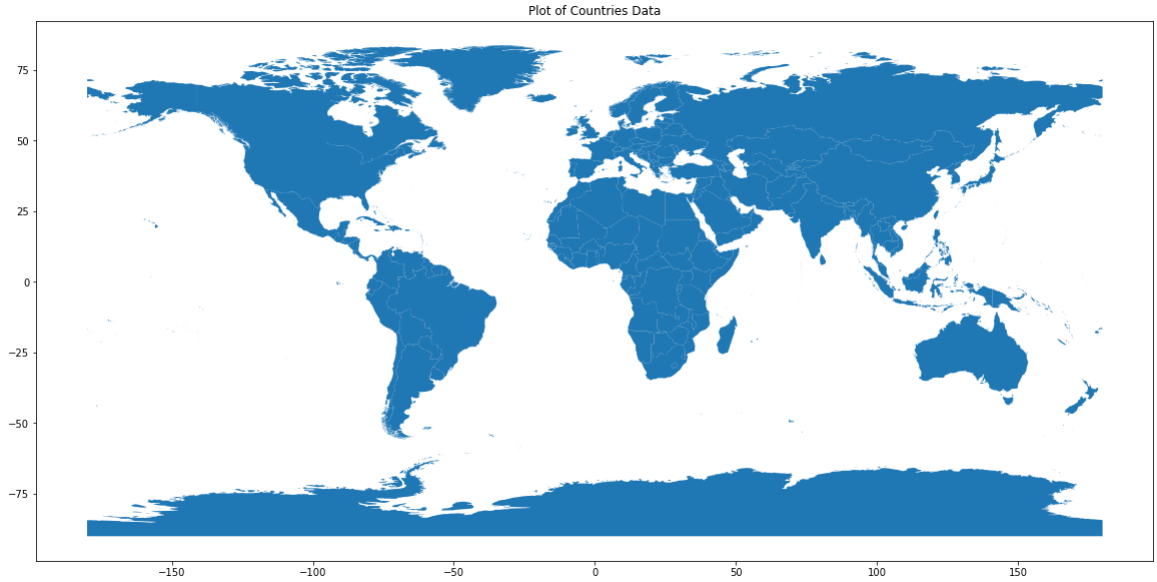
Out[9]:

	Country	ISO_A3	geometry
0	Aruba	ABW	POLYGON ((-69.99694 12.57758, -69.93639 12.531...
1	Afghanistan	AFG	POLYGON ((71.04980 38.40866, 71.05714 38.40903...
2	Angola	AGO	MULTIPOLYGON (((11.73752 -16.69258, 11.73851 -...
3	Anguilla	AIA	MULTIPOLYGON (((-63.03767 18.21296, -63.09952 ...
4	Albania	ALB	POLYGON ((19.74777 42.57890, 19.74601 42.57993...
...	...	...	...
250	Samoa	WSM	MULTIPOLYGON (((-171.57002 -13.93816, -171.564...
251	Yemen	YEM	MULTIPOLYGON (((53.30824 12.11839, 53.31027 12...
252	South Africa	ZAF	MULTIPOLYGON (((37.86378 -46.94085, 37.83644 -...
253	Zambia	ZMB	POLYGON ((31.11984 -8.61663, 31.14102 -8.60619...
254	Zimbabwe	ZWE	POLYGON ((30.01065 -15.64623, 30.05024 -15.640...

255 rows × 3 columns

In [14]:
```
countries_geojson.plot(figsize=(20, 10)).set_title("Plot of Countries Data")
```

Out[14]:
```
Text(0.5, 1.0, 'Plot of Countries Data')
```



Verifying the Validity of the Countries GeoJSON Dataset

To verify how accurate the polygons are, we zoom in on Singapore, which is currently shown below. We plot a `leafmap` plot using the `keplergl` library to analyse how accurately the polygons fit to the countries.

As we can see, the data does not fill the country border fully but it is still considerably accurate on a large scale to be considered valid. Hence, we continue with using this dataset.

In [15]:
```
latlong[latlong.Country == "Singapore"]
```

Out[15]:

	Country	latitude	longitude
194	Singapore	1.352083	103.819836

In [16]:
```
import leafmap.kepler as leafmap
m = leafmap.Map(center=[1.352083, 103.819836], zoom=10)
m.add_gdf(countries_geojson, layer_name="Countries")
m.to_html("maps/countries.html")
m
```

DataHub's Natural Earth Polygons GeoJSON Dataset

DataHub's Natural Earth Polygons GeoJSON Dataset is a geodata data package geojson polygons for the largest administrative subdivisions in every country. The data comes from Natural Earth, a community effort to make visually pleasing, well-crafted maps with cartography or GIS software at small scale.

I have opened this dataset as a `geopandas.GeoDataFrame` object, as shown below. Following this, I have saved it in a local file, `world_locations.json` for later usage by `folium` and `plotly`.

In [17]:
```
states_geojson = gp.read_file("https://datahub.io/core/geo-ne-admin1/r/admin1.geojson").replace("United States of America", "United States").replace('United Republic of Tanzania',
states_geojson.to_file("data/country/world_locations.json", driver='GeoJSON')
del states_geojson
```

In [10]:
```
states_geojson = gp.read_file("data/country/world_locations.json").replace("United States of America", "United States").replace('United Republic of Tanzania', "Tanzania").replace(
states_geojson
```

Out[10]:

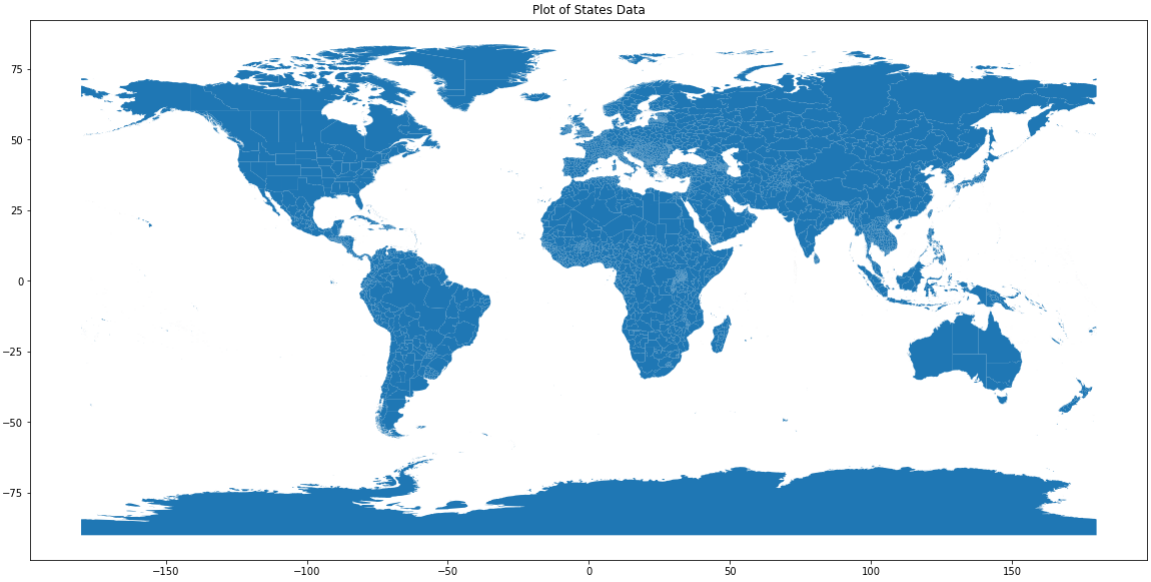
	ISO3166-1-Alpha-3	country	id	name	geometry
0	ABW	Aruba	5150	Aruba	POLYGON ((-69.99694 12.57758, -69.93639 12.531...
1	AFG	Afghanistan	1741	Badghis	POLYGON ((64.30624 35.39722, 64.32468 35.40177...
2	AFG	Afghanistan	1742	Hirat	POLYGON ((61.36393 35.59824, 61.36548 35.59850...

	ISO3166-1-Alpha-3	country	id	name	geometry
3	AFG	Afghanistan	1743	Bamyan	POLYGON ((67.74391 35.44342, 67.75476 35.44412...
4	AFG	Afghanistan	1744	Balkh	POLYGON ((67.25913 37.18515, 67.28145 37.18866...
...	...	...	...	...	...
4642	ZWE	Zimbabwe	529	Manicaland	POLYGON ((33.01165 -17.38399, 32.99745 -17.404...
4643	ZWE	Zimbabwe	530	Matabeleland South	POLYGON ((29.43994 -19.87930, 29.45699 -19.874...
4644	ZWE	Zimbabwe	531	Bulawayo	POLYGON ((28.49757 -20.06270, 28.50532 -20.062...
4645	ZWE	Zimbabwe	532	Masvingo	POLYGON ((31.04181 -19.25226, 31.19870 -19.248...
4646	ZWE	Zimbabwe	533	Mashonaland West	POLYGON ((30.01065 -15.64623, 30.05024 -15.640...

4647 rows × 5 columns

In [19]:
```
states_geojson.plot(figsize=(20, 10)).set_title("Plot of States Data")
```

Out[19]:
```
Text(0.5, 1.0, 'Plot of States Data')
```



UN Datasets

These are mostly datasets found on the UN Database. The data is quite dirty, so it needs to be cleaned. Some additional datasets do not have a CSV Link, hence they are attached with this project notebook. The datasets covered include:

- UN's Dataset of Population, Surface Area and Density
- UN's Dataset of Population Growth and Indicators of Fertility and Mortality
- UN's Dataset for Literacy Amongst Students
- UN's Dataset for Labour Force and Unemployment
- UN's Dataset for Employment by Industry
- UN's Dataset for Energy Consumption

UN's Dataset of Population, Surface Area and Density

Found on the UN Database, this dataset consists of multiple data samples per country per year, hence providing a large array of values for use. The database contains data regarding Population, Surface Area and Population Density, although the data needs to be cleaned.

In [20]:
```
popsaden = pd.read_csv("https://data.un.org/_Docs/SYB/CSV/SYB63_1_202105_Population,%20Surface%20Area%20and%20Density.csv")
popsaden.to_csv("data/country/popsaden.csv", index=False)
```

In [11]:
```
popsaden = pd.read_csv("data/country/popsaden.csv")
popsaden = popsaden.iloc[1:, 1:5].rename(columns={"Population, density and surface area":"Country", "Unnamed: 2":"Year", "Unnamed: 3":"Quantity", "Unnamed: 4":"Value"})
popsaden = popsaden[popsaden.Country.isin(popsaden["Country"].unique()[31:])]
popsaden.Year = popsaden.Year.astype(int)
popsaden = popsaden.groupby(["Country", "Year", "Quantity"]).Value.sum().unstack().rename(columns={"Population mid-year estimates (millions)": "Population", "Population mid-year es

popsaden.loc[:, ["ChildrenPercent", "ElderlyPercent"]]= popsaden[["ChildrenPopulation", "ElderlyPopulation"]].to_numpy()
popsaden[["Population", "MalePopulation", "FemalePopulation"]] = (popsaden[["Population", "MalePopulation", "FemalePopulation"]] * (10**6)).astype(int, errors="ignore")
popsaden.SexRatio = popsaden.SexRatio / 100
popsaden[["ChildrenPopulation", "ElderlyPopulation"]] = (popsaden[["ChildrenPopulation", "ElderlyPopulation"]] * popsaden[["Population"]].to_numpy() / 100).astype(int, errors="ignc
popsaden.SurfaceArea = popsaden.SurfaceArea * 1000

popsaden.reset_index(inplace=True)
isna = pd.DataFrame(popsaden.isna().any(), columns=["isna"])
popsaden[['MalePopulation', 'FemalePopulation', 'SexRatio', 'ChildrenPopulation', 'ElderlyPopulation', 'SurfaceArea']] = isna[isna["isna"]].apply(axis=1, func=lambda row: popsaden[
popsaden = popsaden.replace("United States of America", "United States").replace("Brunei Darussalam", "Brunei").replace("Republic of Korea", "South Korea").replace("Dem. People's F
popsaden
```

Out[11]:

Quantity	Country	Year	Population	MalePopulation	FemalePopulation	SexRatio	ChildrenPopulation	ElderlyPopulation	SurfaceArea	PopulationDensity	ChildrenPercent	ElderlyPercent
0	Afghanistan	2005	25654300.0	13239700.0	12414600.0	1.066462	1.227805e+07	9.384086e+05	652864.0	39.2952	47.8596	3.6579
1	Afghanistan	2010	29185500.0	14935800.0	14249800.0	1.048141	1.406102e+07	1.130646e+06	652864.0	44.7041	48.1781	3.8740
2	Afghanistan	2017	36296100.0	18652000.0	17644100.0	1.057122	1.585504e+07	1.475836e+06	652864.0	55.5956	43.6825	4.0661
3	Afghanistan	2019	38041800.0	19529700.0	18512000.0	1.054975	1.615723e+07	1.584631e+06	652864.0	58.2694	42.4723	4.1655
4	Albania	2005	3086800.0	1552000.0	1534800.0	1.011167	8.180298e+05	3.792597e+05	28748.0	112.6573	26.5009	12.2865
...	...	...	...	...	...	...	...	...	...	...	...	...
986	Zambia	2019	17861000.0	8843200.0	9017800.0	0.980638	7.941447e+06	6.044520e+05	752612.0	24.0265	44.4625	3.3842
987	Zimbabwe	2005	12076700.0	5783700.0	6293000.0	0.919065	4.993341e+06	5.694768e+05	390757.0	31.2180	41.3469	4.7155
988	Zimbabwe	2010	12697700.0	6049000.0	6648700.0	0.909805	5.283488e+06	5.587369e+05	390757.0	32.8234	41.6098	4.4003
989	Zimbabwe	2017	14236600.0	6777100.0	7459500.0	0.908508	6.064493e+06	6.367035e+05	390757.0	36.8013	42.5979	4.4723
990	Zimbabwe	2019	14645500.0	6983400.0	7662100.0	0.911413	6.174265e+06	6.705296e+05	390757.0	37.8583	42.1581	4.5784

991 rows × 12 columns

UN's Dataset of Population Growth and Indicators of Fertility and Mortality

Found on the UN Database, this dataset consists of multiple data samples per country per year, hence providing a large array of values for use. The database contains data regarding Population Increase, Life Expectancy, Infant and Maternal Mortality and Total Fertility Rate, although the data needs to be cleaned. The column names represent the following quantities:

- `InfantMortality` : Infant mortality for both sexes (for each live birth)
- `LifeExpectancy` : Life expectancy at birth for both sexes (years)
- `FemaleLifeExpectancy` : Life expectancy at birth for females (years)
- `MaleLifeExpectancy` : Life expectancy at birth for males (years)
- `MaternalMortality` : Maternal mortality ratio (deaths per person)
- `PopulationIncrease` : Population annual rate of increase (percent)
- `FertilityRate` : Total fertility rate (children per women)

In [22]:
```
humanity = pd.read_csv("https://data.un.org/_Docs/SYB/CSV/SYB62_246_201907_Population%20Growth,%20Fertility%20and%20Mortality%20Indicators.csv", error_bad_lines=False, engine='pyth
humanity.to_csv("data/country/humanity.csv", index=False)
del humanity
```

In [12]:
```
humanity = pd.read_csv("data/country/humanity.csv")
humanity = humanity.iloc[1:, 1:5].rename(columns={"Population growth and indicators of fertility and mortality":"Country", "Unnamed: 2":"Year", "Unnamed: 3":"Quantity", "Unnamed: 4
humanity = humanity[humanity.Country.isin(humanity.Country.unique()[31:])]
humanity.Year = humanity.Year.astype(int)
humanity = humanity.groupby(["Country", "Year", "Quantity"]).Value.sum().unstack().rename(columns={'Infant mortality for both sexes (per 1,000 live births)': "InfantMortality", 'Li

humanity.InfantMortality = humanity.InfantMortality / 1000
humanity.MaternalMortality = humanity.MaternalMortality / 100000

isna = pd.DataFrame(humanity.isna().any(), columns=["isna"])
humanity[["PopulationIncrease", "LifeExpectancy", "MaleLifeExpectancy", "FemaleLifeExpectancy", "FertilityRate", "InfantMortality", "MaternalMortality"]] = isna[isna["isna"]].apply
humanity = humanity.replace("United States of America", "United States").replace("Brunei Darussalam", "Brunei").replace("Republic of Korea", "South Korea").replace("Dem. People's R

humanity
```

Out[12]:

Quantity	Country	Year	PopulationIncrease	LifeExpectancy	MaleLifeExpectancy	FemaleLifeExpectancy	FertilityRate	InfantMortality	MaternalMortality
0	Albania	2005	-0.273	74.822	72.25	77.75	1.9465	0.021153	0.000300
1	Albania	2010	-0.920	75.639	73.16	78.54	1.6400	0.016781	0.000301
2	Albania	2015	-0.394	77.487	75.21	79.96	1.7135	0.009212	0.000286
3	Algeria	2005	1.314	71.827	70.47	73.24	2.3843	0.033620	0.001481
4	Algeria	2010	1.637	74.169	72.95	75.43	2.7240	0.029398	0.001473
...	...	...	...	...	...	...	...	...	...
746	Zambia	2010	2.753	51.750	49.57	53.95	5.6000	0.064721	0.002624
747	Zambia	2015	3.090	59.291	57.03	61.46	5.2000	0.053306	0.002242
748	Zimbabwe	2005	0.326	43.735	41.96	45.62	3.7200	0.065008	0.006290
749	Zimbabwe	2010	1.003	45.017	43.34	46.68	3.8854	0.062686	0.004458
750	Zimbabwe	2015	1.686	56.710	54.88	58.26	4.0897	0.051221	0.004433

751 rows × 9 columns

UN's Dataset for Literacy Amongst Students

Found on the UN Database, this dataset consists of multiple data samples per country per year, hence providing a large array of values for use. The database contains data regarding enrollment in primary, secondary and tertiary education levels, although the data needs to be cleaned. The column names represent the following quantities:

- `PrimaryStudents` : Students enrolled in primary education
- `PrimaryMaleGrossEnrolment` : Gross enrollment ratio - Primary (male)
- `PrimaryFemaleGrossEnrolment` : Gross enrollment ratio - Primary (female)
- `SecondaryStudents` : Students enrolled in secondary education
- `SecondaryMaleGrossEnrolment` : Gross enrollment ratio - Secondary (male)
- `SecondaryFemaleGrossEnrolment` : Gross enrollment ratio - Secondary (female)
- `TertiaryStudents` : Students enrolled in tertiary education
- `TertiaryMaleGrossEnrolment` : Gross enrollment ratio - Tertiary (male)
- `TertiaryFemaleGrossEnrolment` : Gross enrollment ratio - Tertiary (female)

In [24]:
```
education = pd.read_csv("https://data.un.org/_Docs/SYB/CSV/SYB63_309_202009_Education.csv")
education.to_csv("data/country/education.csv", index=False)
del education
```

In [13]:
```
education = pd.read_csv("data/country/education.csv").iloc[1:, 1:5].rename(columns={"Enrolment in primary, secondary and tertiary education levels":"Country", "Unnamed: 2":"Year",
education = education[education.Country.isin(education.Country.unique()[11:])]
education.Year = education.Year.astype(int)
education = education.groupby(["Country", "Year", "Quantity"]).Value.sum().unstack().rename(columns={
    'Students enrolled in primary education (thousands)': "PrimaryStudents",
    'Gross enrollment ratio - Primary (male)': "PrimaryMaleGrossEnrolment",
    'Gross enrollment ratio - Primary (female)': "PrimaryFemaleGrossEnrolment",
    'Students enrolled in secondary education (thousands)': "SecondaryStudents",
    'Gross enrollment ratio - Secondary (male)': "SecondaryMaleGrossEnrolment",
    'Gross enrollment ratio - Secondary (female)': "SecondaryFemaleGrossEnrolment",
    'Students enrolled in tertiary education (thousands)': "TertiaryStudents", 'Gross enrollment ratio - Tertiary (male)': "TertiaryMaleGrossEnrolment", 'Gross enrollment ratio - T

education[["PrimaryStudents", "SecondaryStudents", "TertiaryStudents"]] = 1000 * education[["PrimaryStudents", "SecondaryStudents", "TertiaryStudents"]]

isna = pd.DataFrame(education.isna().any(), columns=["isna"])
education[['PrimaryStudents', 'PrimaryMaleGrossEnrolment', 'PrimaryFemaleGrossEnrolment', 'SecondaryStudents', 'SecondaryMaleGrossEnrolment', 'SecondaryFemaleGrossEnrolment', 'Terti
education = education.replace("United States of America", "United States").replace("Brunei Darussalam", "Brunei").replace("Republic of Korea", "South Korea").replace("Dem. People's

education
```

Out[13]:

Quantity	Country	Year	PrimaryStudents	PrimaryMaleGrossEnrolment	PrimaryFemaleGrossEnrolment	SecondaryStudents	SecondaryMaleGrossEnrolment	SecondaryFemaleGrossEnrolment	TertiaryStudents	Terti
0	Afghanistan	2004	4430142.0	144.61190	62.5910	594306.0	28.35130	5.94270	27648.000000	
1	Afghanistan	2005	4318819.0	123.10600	71.8121	651453.0	27.59330	9.00690	189079.250000	
2	Afghanistan	2009	4945632.0	116.36470	76.5104	1716190.0	59.38230	28.58370	95185.000000	
3	Afghanistan	2010	5279326.0	118.61450	80.6355	2044157.0	66.89290	33.30270	189079.250000	
4	Afghanistan	2014	6217756.0	124.20940	86.7296	2602734.0	67.09690	37.33300	262874.000000	
...	...	...	...	...	...	...	...	...	...	
1172	Zambia	2017	3284841.0	97.54330	99.9042	NaN	NaN	NaN	56680.000000	
1173	Zimbabwe	2003	2361588.0	105.41620	103.2450	758229.0	42.60140	37.43540	108100.333333	
1174	Zimbabwe	2010	2512387.5	108.24595	105.9784	857845.0	48.04635	44.39055	94611.000000	
1175	Zimbabwe	2013	2663187.0	111.07570	108.7118	957461.0	53.49130	51.34570	94115.000000	
1176	Zimbabwe	2015	2512387.5	108.24595	105.9784	857845.0	48.04635	44.39055	135575.000000	

1177 rows × 11 columns

UN's Dataset for Labour Force and Unemployment

Found on the UN Database, this dataset consists of multiple data samples per country per year, hence providing a large array of values for use. The database contains data regarding Labour Force Participation and Unemployment Rate, although the data needs to be cleaned. The column names represent as follows:

- `LabourForce` : Labour force participation - Total
- `LabourForceMale` : Labour force participation - Male
- `LabourForceFemale` : Labour force participation - Female
- `UnemploymentRate` : Unemployment rate - Total
- `UnemploymentRateMale` : Unemployment rate - Male
- `UnemploymentRateFemale` : Unemployment rate - Female

In [26]:
```
labourForce = pd.read_csv("https://data.un.org/_Docs/SYB/CSV/SYB63_329_202009_Labour%20Force%20and%20Unemployment.csv")
labourForce.to_csv("data/country/labourForce.csv", index=False)
del labourForce
```

In [14]:
```
labourForce = pd.read_csv("data/country/labourForce.csv").iloc[1:, 1:5].rename(columns={"Labour force participation and unemployment":"Country", "Unnamed: 2":"Year", "Unnamed: 3":'
labourForce = labourForce[labourForce.Country.isin(labourForce.Country.unique()[25:])]
labourForce.Year = labourForce.Year.astype(int)
labourForce = labourForce.groupby(["Country", "Year", "Quantity"]).Value.sum().unstack().rename(columns={'Labour force participation - Total': "LabourForce", 'Labour force particip
isna = pd.DataFrame(labourForce.isna().any(), columns=["isna"])
labourForce[['LabourForce', 'LabourForceMale', 'LabourForceFemale', 'UnemploymentRate', 'UnemploymentRateMale', 'UnemploymentRateFemale']] = isna[isna["isna"]].apply(axis=1, func=
labourForce = labourForce.replace("United States of America", "United States").replace("Brunei Darussalam", "Brunei").replace("Republic of Korea", "South Korea").replace("Dem. Peop

labourForce
```

Out[14]:

Quantity	Country	Year	LabourForce	LabourForceMale	LabourForceFemale	UnemploymentRate	UnemploymentRateMale	UnemploymentRateFemale
0	Afghanistan	2005	48.49	78.82	15.80	11.43	10.78	14.91
1	Afghanistan	2010	47.43	78.40	14.94	11.48	10.88	14.82
2	Afghanistan	2015	48.37	76.20	18.76	11.39	10.68	14.43
3	Afghanistan	2020	48.94	74.61	21.77	11.16	10.37	14.06
4	Albania	2005	57.46	67.11	47.94	16.46	16.13	16.92
...	...	...	...	...	...	...	...	...
836	Zambia	2020	74.57	78.97	70.39	11.41	10.88	11.98
837	Zimbabwe	2005	82.20	88.15	77.06	4.47	4.31	4.64
838	Zimbabwe	2010	82.41	88.45	77.29	5.22	4.62	5.79
839	Zimbabwe	2015	83.01	89.17	77.84	5.30	4.69	5.90
840	Zimbabwe	2020	83.12	88.95	78.18	4.99	4.44	5.52

841 rows × 8 columns

UN's Dataset for Employment by Industry

Found on the UN Database, this dataset consists of multiple data samples per country per year, hence providing a large array of values for use. The database contains data regarding Employment by different Industry (eg Agriculture and Services), although the data needs to be cleaned.

Post-cleaning, the following describes each of the column names:

- `AgricultureEmployment` : Employment by industry: Agriculture (%) Male and Female
- `AgricultureEmploymentMale` : Employment by industry: Agriculture (%) Male
- `AgricultureEmploymentFemale` : Employment by industry: Agriculture (%) Female
- `IndustryEmployment` : Employment by industry: Industry (%) Male and Female
- `IndustryEmploymentMale` : Employment by industry: Industry (%) Male
- `IndustryEmploymentFemale` : Employment by industry: Industry (%) Female
- `ServicesEmployment` : Employment by industry: Services (%) Male and Female
- `ServicesEmploymentMale` : Employment by industry: Services (%) Male
- `ServicesEmploymentFemale` : Employment by industry: Services (%) Female

In [28]:
```
employment = pd.read_csv("https://data.un.org/_Docs/SYB/CSV/SYB63_200_202009_Employment.csv")
employment.to_csv("data/country/employment.csv", index=False)
del employment
```

In [15]:
```
employment = pd.read_csv("data/country/employment.csv").iloc[1:, 1:5].rename(columns={"Employment by economic activity":"Country", "Unnamed: 2":"Year", "Unnamed: 3":"Quantity", "Un
employment = employment[employment.Country.isin(employment.Country.unique()[23:-1])]
employment.Year = employment.Year.astype(int)
employment = employment.groupby(["Country", "Year", "Quantity"]).Value.sum().unstack().rename(columns={'Employment by industry: Agriculture (%) Male and Female': "AgricultureEmploy

isna = pd.DataFrame(employment.isna().any(), columns=["isna"])
employment[isna[isna["isna"]].index] = isna[isna["isna"]].apply(axis=1, func=lambda row: employment[row.name].fillna(employment.groupby("Country")[row.name].transform('mean'))).T
employment
```

Out[15]:

Quantity	Country	Year	AgricultureEmployment	AgricultureEmploymentMale	AgricultureEmploymentFemale	IndustryEmployment	IndustryEmploymentMale	IndustryEmploymentFemale	ServicesEmployment	Se
0	Afghanistan	2005	62.245	59.362	78.491	11.528	11.535	11.488	26.227	
1	Afghanistan	2010	54.685	51.512	72.966	14.391	14.064	16.280	30.924	
2	Afghanistan	2015	47.114	42.381	68.470	17.043	15.959	21.933	35.844	
3	Afghanistan	2020	42.352	36.408	64.832	18.252	16.509	24.845	39.396	
4	Albania	2005	47.214	38.604	59.199	15.770	21.402	7.931	37.016	
...	...	...	...	...	...	...	...	...	...	
797	Zambia	2020	48.499	43.744	53.642	10.783	16.658	4.427	40.718	
798	Zimbabwe	2005	64.549	57.144	71.890	9.555	15.708	3.454	25.897	
799	Zimbabwe	2010	65.529	59.341	71.615	9.243	15.466	3.121	25.229	
800	Zimbabwe	2015	67.169	62.857	71.371	7.141	12.269	2.145	25.690	
801	Zimbabwe	2020	66.270	62.885	69.578	6.545	11.306	1.891	27.186	

802 rows × 11 columns

UN's Dataset for Energy Consumption

Found on the UN Database, this dataset consists of multiple data samples per country per year, hence providing a large array of values for use. The database contains data regarding Energy Production, Trade and Consumption, although the data needs to be cleaned. The column names represent the following quantities:

- `EnergyProduction` : Primary energy production in petajoules
- `EnergySupply` : Total energy supply in petajoules
- `EnergySupplyPerCapita` : Energy supply per capita in gigajoules
- `EnergyTrade` : Net Energy imports, exports and bunkers in petajoules
- `EnergyStockChange` : Change in energy stocks in petajoules

In [30]:
```
energyConsumption = pd.read_csv("https://data.un.org/_Docs/SYB/CSV/SYB63_263_202009_Production,%20Trade%20and%20Supply%20of%20Energy.csv")
energyConsumption.to_csv("data/country/energyConsumption.csv", index=False)
```

```
del energyConsumption
```

In [16]:
```
energyConsumption = pd.read_csv("data/country/energyConsumption.csv").iloc[1:, 1:5].rename(columns={"Production, trade and supply of energy":"Country", "Unnamed: 2":"Year", "Unname
energyConsumption = energyConsumption[energyConsumption.Country.isin(energyConsumption.Country.unique()[7:])]
energyConsumption.Year = energyConsumption.Year.astype(int)
energyConsumption = energyConsumption.groupby(["Country", "Year", "Quantity"]).Value.sum().unstack().rename(columns={
    'Primary energy production (petajoules)': "EnergyProduction",
    'Total supply (petajoules)': "EnergySupply",
    'Supply per capita (gigajoules)': "EnergySupplyPerCapita",
    'Net imports [Imports - Exports - Bunkers] (petajoules)': "EnergyTrade",
    'Changes in stocks (petajoules)': "EnergyStockChange",
})[['EnergyProduction', 'EnergySupply', 'EnergySupplyPerCapita', 'EnergyTrade', 'EnergyStockChange']].astype(float).reset_index()
energyConsumption = energyConsumption.replace("United States of America", "United States").replace("Brunei Darussalam", "Brunei").replace("Republic of Korea", "South Korea").repla
isna = pd.DataFrame(energyConsumption.isna().any(), columns=["isna"])
energyConsumption[isna[isna["isna"]].index] = isna[isna["isna"]].apply(axis=1, func=lambda row: energyConsumption[row.name].fillna(energyConsumption.groupby("Country")[row.name].tr

energyConsumption
```

Out[16]:

Quantity	Country	Year	EnergyProduction	EnergySupply	EnergySupplyPerCapita	EnergyTrade	EnergyStockChange
0	Afghanistan	1990	19.0	46.0	4.0	28.0	0.000000
1	Afghanistan	1995	16.0	29.0	1.0	13.0	0.000000
2	Afghanistan	2000	18.0	25.0	1.0	8.0	0.000000
3	Afghanistan	2005	23.0	36.0	1.0	14.0	0.000000
4	Afghanistan	2010	41.0	136.0	5.0	95.0	0.000000
...	...	...	...	...	...	...	...
1731	Zimbabwe	2005	379.0	414.0	33.0	36.0	1.000000
1732	Zimbabwe	2010	369.0	395.0	31.0	26.0	-1.428571
1733	Zimbabwe	2015	450.0	473.0	34.0	47.0	24.000000
1734	Zimbabwe	2016	381.0	467.0	33.0	48.0	-39.000000
1735	Zimbabwe	2017	427.0	474.0	33.0	50.0	4.000000

1736 rows × 7 columns

Merging all the UN Data Together

In the end, we merge all of this data into a compound DataFrame object `undata`. I have decided to use the `OUTER JOIN` operation, then substituted values based on the known mean.

In [17]:
```
undata = reduce(lambda a,b: pd.merge(a, b, how="outer"), [popsaden, humanity, education, labourForce, employment, energyConsumption]).sort_values(["Country", "Year"], ignore_index=
isna = pd.DataFrame(undata.isna().any(), columns=["isna"])
undata[isna[isna["isna"]].index] = isna[isna["isna"]].apply(axis=1, func=lambda row: undata[row.name].fillna(undata.groupby("Country")[row.name].transform('mean'))).T
undata = undata.set_index(["Country", "Year"])
undata
```

Out[17]:

Country	Year	Quantity	Population	MalePopulation	FemalePopulation	SexRatio	ChildrenPopulation	ElderlyPopulation	SurfaceArea	PopulationDensity	ChildrenPercent	ElderlyPercent	...	IndustryEmploymentMale
Afghanistan	1990	32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	14.51675
	1995	32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	14.51675
	2000	32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	14.51675
	2004	32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	14.51675
	2005	25654300.0	13239700.0	12414600.0	1.066462	1.227805e+07	9.384086e+05	652864.0	39.295200	47.859600	3.657900	...	11.53500
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Zimbabwe	2015	13414125.0	6398300.0	7015825.0	0.912198	5.628896e+06	6.088617e+05	390757.0	34.675250	41.928175	4.541625	...	12.26900
	2016	13414125.0	6398300.0	7015825.0	0.912198	5.628896e+06	6.088617e+05	390757.0	34.675250	41.928175	4.541625	...	13.68725
	2017	14236600.0	6777100.0	7459500.0	0.908508	6.064493e+06	6.367035e+05	390757.0	36.801300	42.597900	4.472300	...	13.68725
	2019	14645500.0	6983400.0	7662100.0	0.911413	6.174265e+06	6.705296e+05	390757.0	37.858300	42.158100	4.578400	...	13.68725
	2020	13414125.0	6398300.0	7015825.0	0.912198	5.628896e+06	6.088617e+05	390757.0	34.675250	41.928175	4.541625	...	11.30600

2828 rows × 46 columns

Merging the Data

It is necessary to merge all the data, so that is pretty much all that is done here. I have decided to perform an INNER join operation on the data so that I have all the data necessary. This data is stored in a Pandas DataFrame `geocountries_latlong`. This has then been converted into a `geopandas.GeoDataFrame` point-based object named `countries`. Following this, I have plotted this data in the form of a scatterplot by Population.

In [18]:
```
geocountries_latlong = pd.merge(geocountries, latlong)
geocountries_latlong
```

Out[18]:

CountryCode	CountryCode3	NumericCountryCode	fips	Country	Capital	Area in km²	Population	Continent	latitude	longitude
0	AD	AND	20	AN	Andorra	Andorra la Vella	468.0	77006	EU	42.546245	1.601554
1	AE	ARE	784	AE	United Arab Emirates	Abu Dhabi	82880.0	9630959	AS	23.424076	53.847818
2	AF	AFG	4	AF	Afghanistan	Kabul	647500.0	37172386	AS	33.939110	67.709953
3	AG	ATG	28	AC	Antigua and Barbuda	St. John's	443.0	96286	NA	17.060816	-61.796428
4	AI	AIA	660	AV	Anguilla	The Valley	102.0	13254	NA	18.220554	-63.068615
...	...	...	...	...	...	...	...	...	...	...	...
236	YE	YEM	887	YM	Yemen	Sanaa	527970.0	28498687	AS	15.552727	48.516388
237	YT	MYT	175	MF	Mayotte	Mamoudzou	374.0	279471	AF	-12.827500	45.166244
238	ZA	ZAF	710	SF	South Africa	Pretoria	1219912.0	57779622	AF	-30.559482	22.937506
239	ZM	ZMB	894	ZA	Zambia	Lusaka	752614.0	17351822	AF	-13.133897	27.849332
240	ZW	ZWE	716	ZI	Zimbabwe	Harare	390580.0	14439018	AF	-19.015438	29.154857

241 rows × 11 columns

In [19]:
```
countries = gp.GeoDataFrame(geocountries_latlong, geometry=gp.points_from_xy(geocountries_latlong.longitude, geocountries_latlong.latitude))
countries
```

Out[19]:

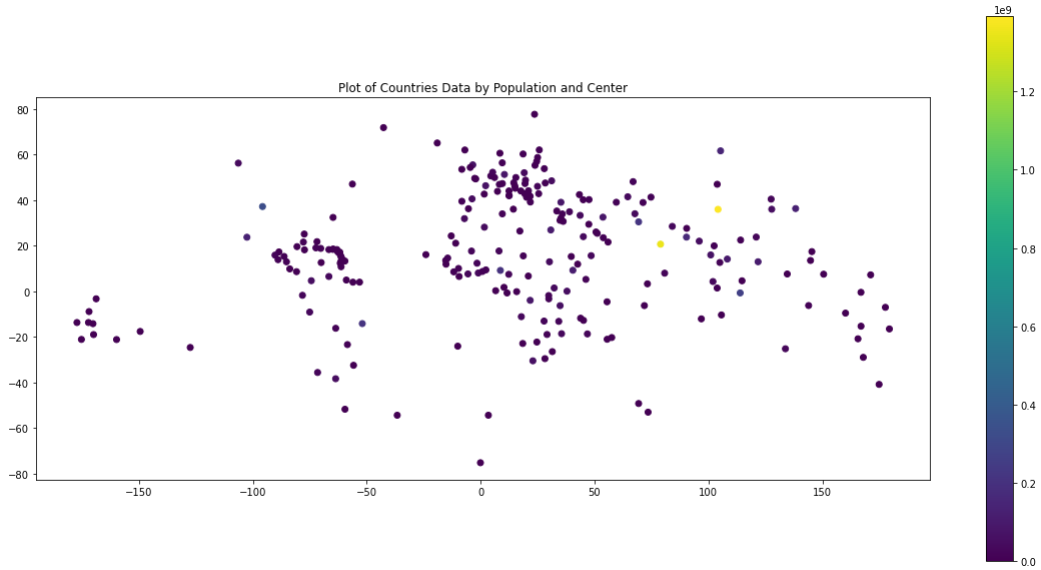
CountryCode	CountryCode3	NumericCountryCode	fips	Country	Capital	Area in km²	Population	Continent	latitude	longitude	geometry
0	AD	AND	20	AN	Andorra	Andorra la Vella	468.0	77006	EU	42.546245	1.601554	POINT (1.60155 42.54624)
1	AE	ARE	784	AE	United Arab Emirates	Abu Dhabi	82880.0	9630959	AS	23.424076	53.847818	POINT (53.84782 23.42408)
2	AF	AFG	4	AF	Afghanistan	Kabul	647500.0	37172386	AS	33.939110	67.709953	POINT (67.70995 33.93911)

	CountryCode	CountryCode3	NumericCountryCode	fips	Country	Capital	Area in km²	Population	Continent	latitude	longitude	geometry
3	AG	ATG	28	AC	Antigua and Barbuda	St. John's	443.0	96286	NA	17.060816	-61.796428	POINT (-61.79643 17.06082)
4	AI	AIA	660	AV	Anguilla	The Valley	102.0	13254	NA	18.220554	-63.068615	POINT (-63.06862 18.22055)
...	...	...	...	...	...	...	...	...	...	...	...	...
236	YE	YEM	887	YM	Yemen	Sanaa	527970.0	28498687	AS	15.552727	48.516388	POINT (48.51639 15.55273)
237	YT	MYT	175	MF	Mayotte	Mamoudzou	374.0	279471	AF	-12.827500	45.166244	POINT (45.16624 -12.82750)
238	ZA	ZAF	710	SF	South Africa	Pretoria	1219912.0	57779622	AF	-30.559482	22.937506	POINT (22.93751 -30.55948)
239	ZM	ZMB	894	ZA	Zambia	Lusaka	752614.0	17351822	AF	-13.133897	27.849332	POINT (27.84933 -13.13390)
240	ZW	ZWE	716	ZI	Zimbabwe	Harare	390580.0	14439018	AF	-19.015438	29.154857	POINT (29.15486 -19.01544)

241 rows × 12 columns

In [35]:
```
countries.plot(figsize=(20, 10), column="Population", legend=True).set_title("Plot of Countries Data by Population and Center")
```

Out[35]:
```
Text(0.5, 1.0, 'Plot of Countries Data by Population and Center')
```



In [20]:
```
world = gp.GeoDataFrame(pd.merge(geocountries, countries_geojson))
world
```

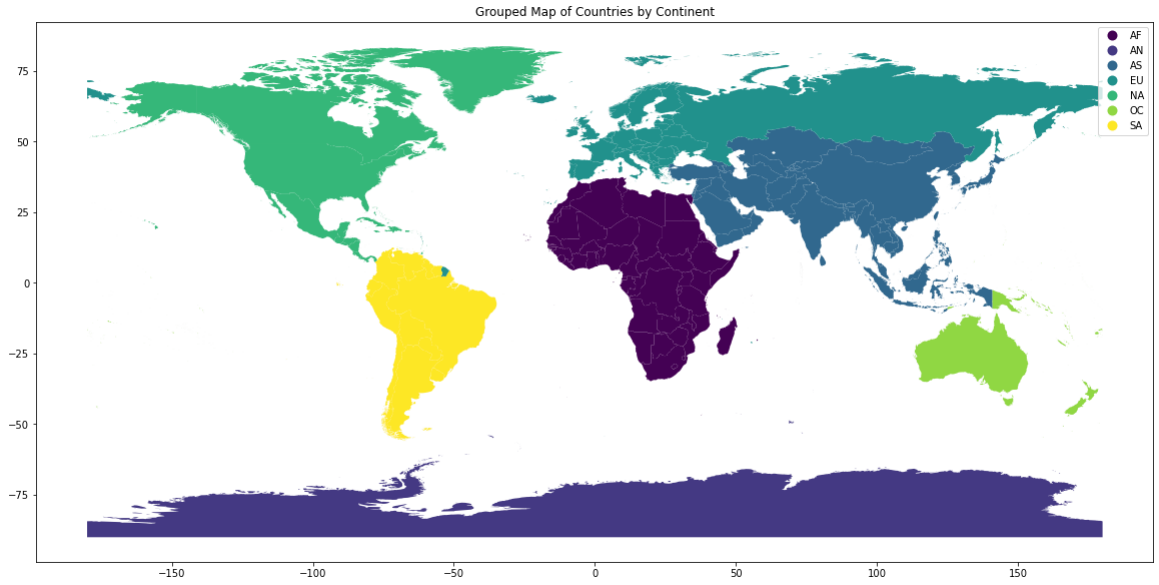
Out[20]:

	CountryCode	CountryCode3	NumericCountryCode	fips	Country	Capital	Area in km²	Population	Continent	ISO_A3	geometry
0	AD	AND	20	AN	Andorra	Andorra la Vella	468.0	77006	EU	AND	POLYGON ((1.70701 42.50278, 1.69750 42.49446, ...
1	AE	ARE	784	AE	United Arab Emirates	Abu Dhabi	82880.0	9630959	AS	ARE	MULTIPOLYGON (((53.86305 24.23469, 53.88860 24...
2	AF	AFG	4	AF	Afghanistan	Kabul	647500.0	37172386	AS	AFG	POLYGON ((71.04980 38.40866, 71.05714 38.40903...
3	AG	ATG	28	AC	Antigua and Barbuda	St. John's	443.0	96286	NA	ATG	MULTIPOLYGON (((-61.77302 17.12653, -61.75642 ...
4	AI	AIA	660	AV	Anguilla	The Valley	102.0	13254	NA	AIA	MULTIPOLYGON (((-63.03767 18.21296, -63.09952 ...
...	...	...	...	...	...	...	...	...	...	...	...
234	XK	XKX	0	KV	Kosovo	Pristina	10908.0	1845300	EU	-99	POLYGON ((20.86470 43.21734, 20.86160 43.21752...
235	YE	YEM	887	YM	Yemen	Sanaa	527970.0	28498687	AS	YEM	MULTIPOLYGON (((53.30824 12.11839, 53.31027 12...
236	ZA	ZAF	710	SF	South Africa	Pretoria	1219912.0	57779622	AF	ZAF	MULTIPOLYGON (((37.86378 -46.94085, 37.83644 -...
237	ZM	ZMB	894	ZA	Zambia	Lusaka	752614.0	17351822	AF	ZMB	POLYGON ((31.11984 -8.61663, 31.14102 -8.60619...
238	ZW	ZWE	716	ZI	Zimbabwe	Harare	390580.0	14439018	AF	ZWE	POLYGON ((30.01065 -15.64623, 30.05024 -15.640...

239 rows × 11 columns

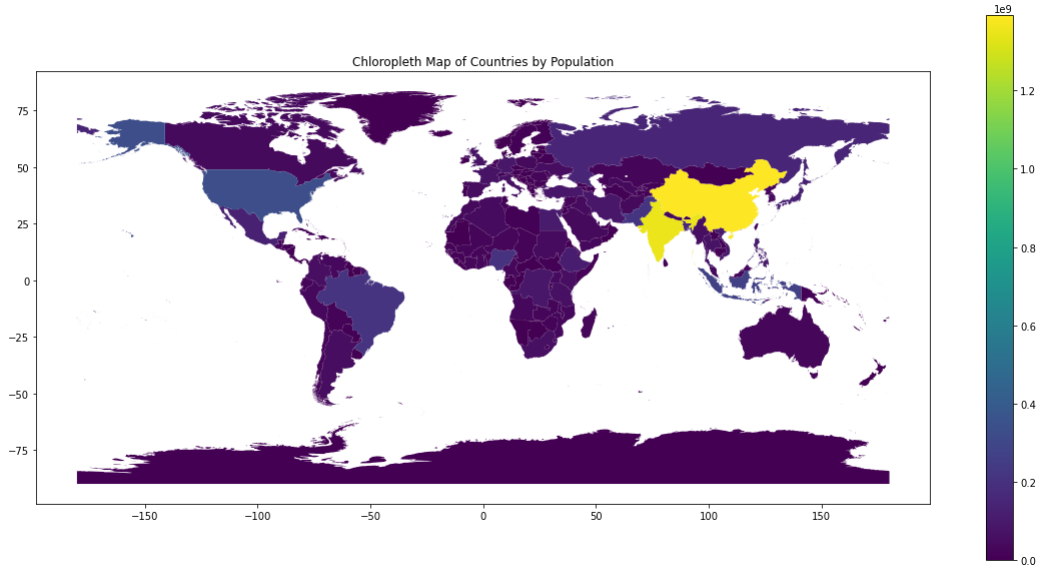
In [20]:
```
world.plot(column="Continent", figsize=(20, 10), legend=True, cmap="viridis").set_title("Grouped Map of Countries by Continent")
```

Out[20]:
```
Text(0.5, 1.0, 'Grouped Map of Countries by Continent')
```

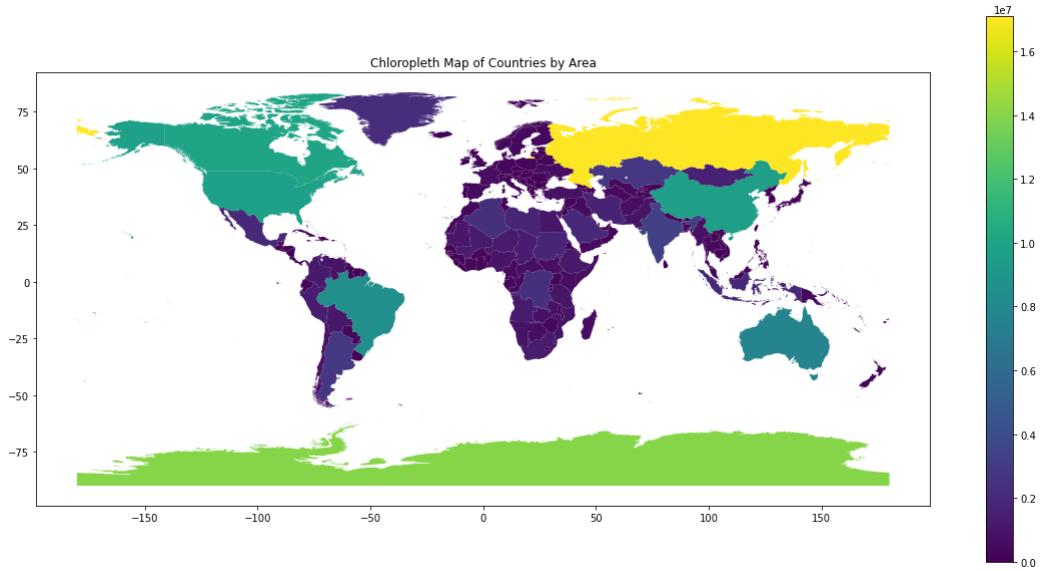


In [21]:
```
world.plot(column="Population", figsize=(20, 10), legend=True, cmap="viridis").set_title("Chloropleth Map of Countries by Population")
```

Out[21]:
```
Text(0.5, 1.0, 'Chloropleth Map of Countries by Population')
```



In [22]:
```
_ = world.plot(column="Area in km²", figsize=(20, 10), legend=True, cmap="viridis").set_title("Chloropleth Map of Countries by Area")
```



In [21]:
```
undata_world = gp.GeoDataFrame(pd.merge(undata.reset_index(), world[list(world.columns[:-5])+list(world.columns[-3:])], on="Country", how="outer").set_index(["Country", "Year"]))
undata_world
```

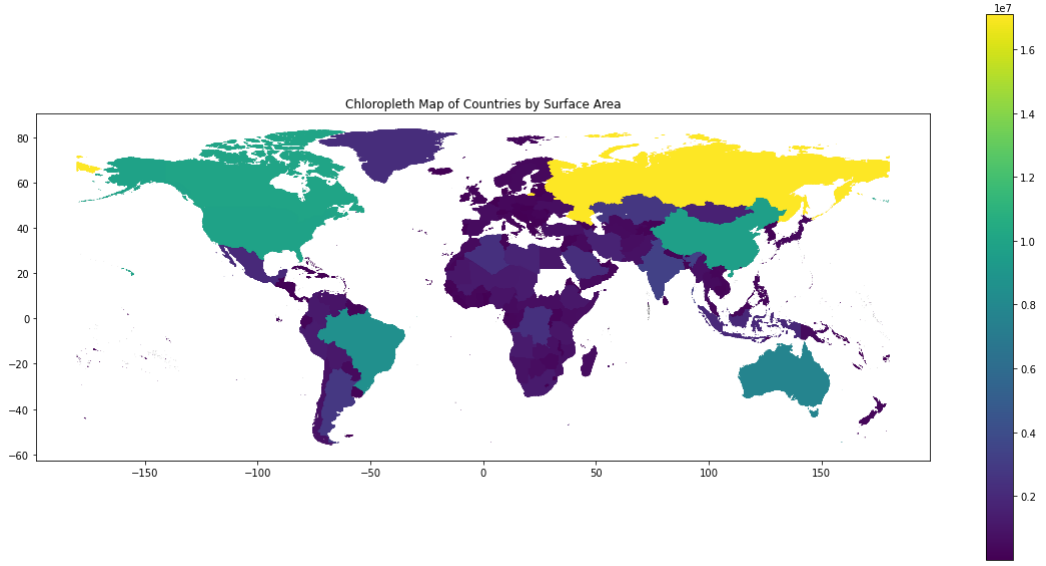
Out[21]:

	Population	MalePopulation	FemalePopulation	SexRatio	ChildrenPopulation	ElderlyPopulation	SurfaceArea	PopulationDensity	ChildrenPercent	ElderlyPercent	...	EnergyTrade	EnergyStock(
Country	**Year**												

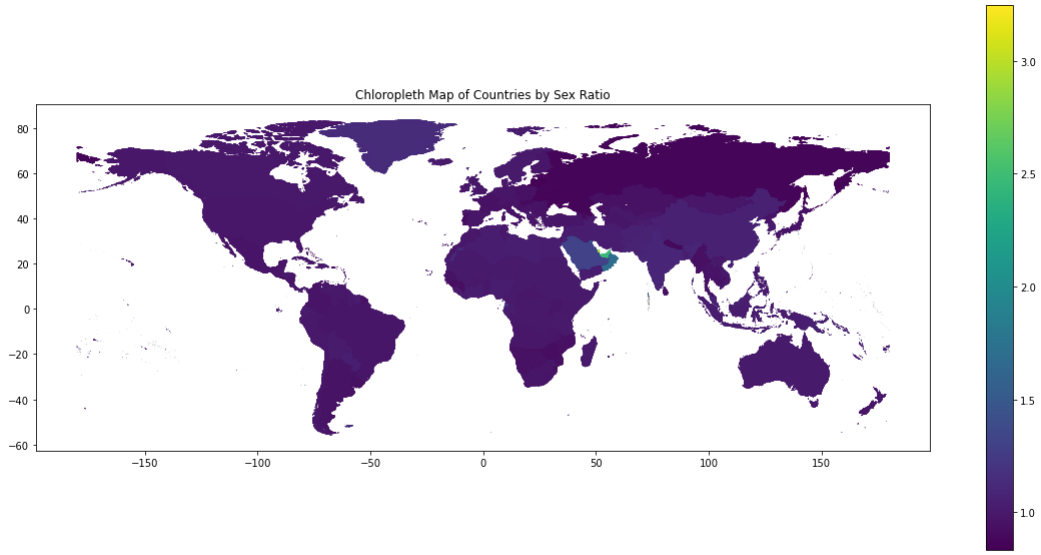
Country	Year	Population	MalePopulation	FemalePopulation	SexRatio	ChildrenPopulation	ElderlyPopulation	SurfaceArea	PopulationDensity	ChildrenPercent	ElderlyPercent	...	EnergyTrade	EnergyStock(
Afghanistan	**1990.0**	32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	28.00	
	1995.0	32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	13.00	
	2000.0	32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	8.00	
	2004.0	32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	40.75	
	2005.0	25654300.0	13239700.0	12414600.0	1.066462	1.227805e+07	9.384086e+05	652864.0	39.295200	47.859600	3.657900	...	14.00	
...	**...**	...	...	...	...	...	...	...	...	...	...	...	...	
Pitcairn Islands	**NaN**	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
French Southern Territories	**NaN**	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
Taiwan	**NaN**	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
U.S. Outlying Islands	**NaN**	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
Vatican City	**NaN**	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	

2839 rows × 54 columns

In [24]:
```
_ = undata_world.plot(column="SurfaceArea", figsize=(20, 10), legend=True, cmap="viridis").set_title("Chloropleth Map of Countries by Surface Area")
```



In [25]:
```
_ = undata_world.plot(column="SexRatio", figsize=(20, 10), legend=True, cmap="viridis").set_title("Chloropleth Map of Countries by Sex Ratio")
```



Light Pollution Datasets

The Harmonized Global Nighttime Light (1992 - 2018) Dataset

This dataset is specifically the largest, containing close to 20 billion data points (specifically `20322960028`), hence this data has to be acquired in chunks. All the data can be downloaded from a zip file as shown below. Here are the instructions:

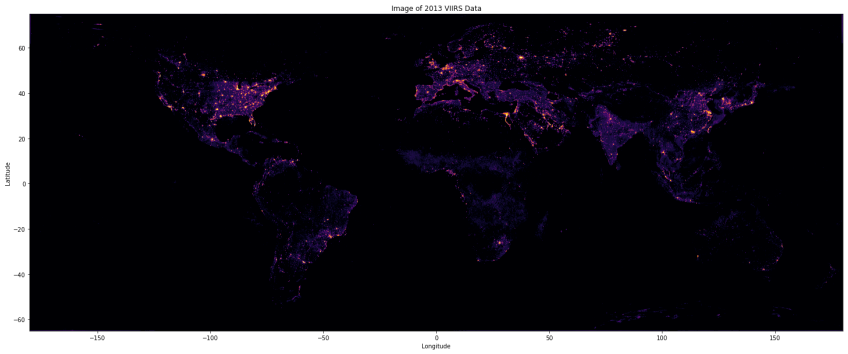
1. Download the zip file from https://figshare.com/ndownloader/articles/9828827/versions/2
2. Make a directory relative to this notebook called `data/nightLight`
3. Unzip all the contents in the zip file into the `nightLight` directory as created in step 2.
4. Remove the zip file, so as to conserve disk space.

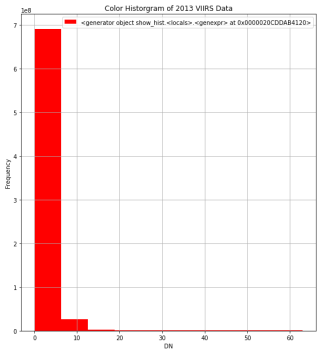
Installation can take place as follows:

```
curl https://figshare.com/ndownloader/articles/9828827/versions/2 > nightLight.zip
unzip nightLight.zip -d data/nightLight/
rm nightLight.zip
```

Due to the large amount of data, this data has been separated into country/state-wise and year-wise statistical summaries, containing the `mean`, `median`, `mode`, `min`, `max`, `count` and `std`. GeoTIFF files are simply converted into DataFrames as shown below.

In [9]:
```
fig, (axim, axhist) = plt.subplots(1, 2, figsize=(40, 10), gridspec_kw={'width_ratios': [3, 1]})
rf = rs.open("data/nightLight/DN_NTL_2013_simVIIRS.tif", "r")
show(rf, ax=axim, cmap="inferno")
show_hist(rf, ax=axhist)
axim.set(xlabel="Longitude", ylabel="Latitude", title="Image of 2013 VIIRS Data")
axhist.set_title("Color Historgram of 2013 VIIRS Data")
del rf
```





In [10]:
```
def retrieveGeoStats(raster_file, geojson=states_geojson, nodata=64):
    year = re.search(r"\d+", raster_file).group()
    print(f"Processing {year}...")
    rf = rs.open(raster_file,"r")
    values = pd.DataFrame(rstats.zonal_stats(geojson, rf.read(1), affine=rf.transform, stats=["count", "mean", "min", "max", 'median', 'std', "majority"], geojson_out=True, nodata=
    del rf
    return pd.concat([values, values.properties.apply(pd.Series)], axis=1).drop(columns=["id", "type", "properties", "bbox", "ISO3166-1-Alpha-3", "geometry"]).rename(columns={"maj

nightLight2013 = retrieveGeoStats(raster_file = "data/nightLight/DN_NTL_2013_simVIIRS.tif")
nightLight2013
```

```
Processing 2013...
```

Out[10]:

	country	name	min	max	mean	count	std	median	mode
0	Aruba	Aruba	7.0	57.0	28.666667	201	16.178831	28.0	7.0
1	Afghanistan	Badghis	0.0	32.0	0.236040	29334	1.336606	0.0	0.0
2	Afghanistan	Hirat	0.0	56.0	0.741159	78071	2.932197	0.0	0.0
3	Afghanistan	Bamyan	0.0	8.0	0.180535	25153	1.039803	0.0	0.0
4	Afghanistan	Balkh	0.0	61.0	2.011429	23275	5.675208	0.0	0.0
...	...	...	...	...	...	...	...	...	...
4642	Zimbabwe	Manicaland	0.0	46.0	1.915220	42982	3.263857	0.0	0.0
4643	Zimbabwe	Matabeleland South	0.0	52.0	0.677659	68170	2.175918	0.0	0.0
4644	Zimbabwe	Bulawayo	0.0	55.0	21.061511	569	15.322821	14.0	7.0
4645	Zimbabwe	Masvingo	0.0	26.0	0.701861	69850	2.066331	0.0	0.0
4646	Zimbabwe	Mashonaland West	0.0	49.0	4.252489	70118	3.467990	6.0	6.0

4647 rows × 9 columns

In [11]:
```
nightLight2013.to_csv("data/nightLight/nightLight2013.csv", index=False)
```

```
del nightLight2013
```

In [6]:

```
for raster_file in glob("data/nightLight/Harmonized*.tif"):
    year = re.search(r"\d+", raster_file).group()
    nightLightTemp = retrieveGeoStats(raster_file)
    nightLightTemp.to_csv(f"data/nightLight/harmonizedNightLight{year}.csv", index=False)
    del nightLightTemp
```

```
Processing 1992...
Processing 1993...
Processing 1994...
Processing 1995...
Processing 1996...
Processing 1997...
Processing 1998...
Processing 1999...
Processing 2000...
Processing 2001...
Processing 2002...
Processing 2003...
Processing 2004...
Processing 2005...
Processing 2006...
Processing 2007...
Processing 2008...
Processing 2009...
Processing 2010...
Processing 2011...
Processing 2012...
Processing 2013...
Processing 2014...
Processing 2015...
Processing 2016...
Processing 2017...
Processing 2018...
```

In [12]:

```
for raster_file in glob("data/nightLight/Harmonized*.tif"):
    year = re.search(r"\d+", raster_file).group()
    nightLightTemp = retrieveGeoStats(raster_file, nodata=0)
    nightLightTemp.to_csv(f"data/nightLight/harmonizedNightLightNoData{year}.csv", index=False)
    del nightLightTemp
print("Processing Completed!")
```

```
Processing 1992...
Processing 1993...
Processing 1994...
Processing 1995...
Processing 1996...
Processing 1997...
Processing 1998...
Processing 1999...
Processing 2000...
Processing 2001...
Processing 2002...
Processing 2003...
Processing 2004...
Processing 2005...
Processing 2006...
Processing 2007...
Processing 2008...
Processing 2009...
Processing 2010...
Processing 2011...
Processing 2012...
Processing 2013...
Processing 2014...
Processing 2015...
Processing 2016...
Processing 2017...
Processing 2018...
Processing Completed!
```

In [5]:

```
def retrieveGeoStats(raster_file, geojson=countries_geojson, nodata=64):
    year = re.search(r"\d+", raster_file).group()
    print(f"Processing {year}...")
    rf = rs.open(raster_file,"r")
    values = pd.DataFrame(rstats.zonal_stats(geojson, rf.read(1), affine=rf.transform, stats=["count", "mean", "min", "max", 'median', 'std', "majority"], geojson_out=True, nodata=
    del rf
    return pd.concat([values, values.properties.apply(pd.Series)], axis=1).drop(columns=["id", "type", "properties", "bbox", "ISO_A3", "geometry"]).rename(columns={"majority": "mo

nightLightIndiv = [retrieveGeoStats(raster_file, countries_geojson) for raster_file in glob("data/nightLight/Harmonized*.tif")]
nightLight = functools.reduce(pd.DataFrame.join, nightLightIndiv).stack().reset_index()
nightLight
```

```
Processing 1992...
Processing 1993...
Processing 1994...
Processing 1995...
Processing 1996...
Processing 1997...
Processing 1998...
Processing 1999...
Processing 2000...
Processing 2001...
Processing 2002...
Processing 2003...
Processing 2004...
Processing 2005...
Processing 2006...
Processing 2007...
Processing 2008...
Processing 2009...
Processing 2010...
Processing 2011...
Processing 2012...
Processing 2013...
Processing 2014...
Processing 2015...
Processing 2016...
Processing 2017...
Processing 2018...
```

Out[5]:

	Country	level_1	0
0	Aruba	min 1992	5.000000
1	Aruba	max 1992	63.000000
2	Aruba	mean 1992	30.731343
3	Aruba	count 1992	201.000000
4	Aruba	std 1992	18.759739
...	...	...	...
48190	Zimbabwe	mean 2018	2.009717
48191	Zimbabwe	count 2018	481200.000000

	Country	level_1	0
48192	Zimbabwe	std 2018	3.633314
48193	Zimbabwe	median 2018	0.000000
48194	Zimbabwe	mode 2018	0.000000

48195 rows × 3 columns

In [48]:
```
nightLight = functools.reduce(pd.DataFrame.join, nightLightIndiv).stack().reset_index()
nightLight.loc[:, ["Quantity", "Year"]] = nightLight.level_1.str.split().apply(pd.Series).rename(columns={0:"Quantity", 1:"Year"})
nightLightCSV = nightLight.rename(columns={0: "Value"}).drop(columns="level_1").set_index(["Year", "Quantity", "Country"]).unstack()["Value"]
nightLightCSV
```

Out[48]:

	Country	Afghanistan	Akrotiri Sovereign Base Area	Aland	Albania	Algeria	American Samoa	Andorra	Angola	Anguilla	Antarctica	...	Uzbekistan	Vanuatu	Vatican City	Venezuela	
Year	**Quantity**																
1992	**count**	901332.000000	137.000000	2190.000000	43766.000000	3.062687e+06	216.000000	711.000000	1.491515e+06	100.000000	73912.000000	...	698379.000000	14990.000000	0.0	1.076792e+06	40241
	max	59.000000	39.000000	53.000000	39.000000	6.300000e+01	32.000000	63.000000	6.300000e+01	21.000000	8.000000	...	63.000000	33.000000	0.0	6.300000e+01	6
	mean	0.027943	14.912409	2.225571	0.264269	5.061918e-01	9.685185	15.614627	3.853196e-02	9.650000	0.004127	...	1.726949	0.148032	0.0	1.132978e+00	
	median	0.000000	14.000000	0.000000	0.000000	0.000000e+00	8.000000	9.000000	0.000000e+00	9.000000	0.000000	...	0.000000	0.000000	0.0	0.000000e+00	
	min	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000e+00	0.000000	0.000000	...	0.000000	0.000000	0.0	0.000000e+00	
...	**...**	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2018	**mean**	2.047282	34.408759	4.557534	8.770347	2.811383e+00	18.037037	24.846695	1.631923e+00	15.900000	0.621469	...	3.258340	1.618412	0.0	4.400938e+00	
	median	0.000000	34.000000	0.000000	7.000000	0.000000e+00	13.500000	19.000000	0.000000e+00	14.000000	0.000000	...	0.000000	0.000000	0.0	0.000000e+00	
	min	0.000000	9.000000	0.000000	0.000000	0.000000e+00	0.000000	7.000000	0.000000e+00	0.000000	0.000000	...	0.000000	0.000000	0.0	0.000000e+00	
	mode	0.000000	51.000000	0.000000	7.000000	0.000000e+00	7.000000	8.000000	0.000000e+00	11.000000	0.000000	...	0.000000	0.000000	0.0	0.000000e+00	
	std	3.555359	14.801142	10.894895	7.885158	6.959879e+00	11.895320	17.029646	3.834231e+00	7.139328	1.978841	...	5.914810	6.371589	0.0	7.879986e+00	1

189 rows × 255 columns

In [49]:
```
nightLightCSV.to_csv("data/nightLight/nightLight.csv")
del nightLightCSV
del nightLight
del nightLightIndiv
```

In [34]:
```
nightLightNoDarkIndiv = [retrieveGeoStats(raster_file, countries_geojson, 0) for raster_file in glob("data/nightLight/Harmonized*.tif")]
nightLightNoDark = functools.reduce(pd.DataFrame.join, nightLightNoDarkIndiv).stack().reset_index()
nightLightNoDark
```

```
Processing 1992...
Processing 1993...
Processing 1994...
Processing 1995...
Processing 1996...
Processing 1997...
Processing 1998...
Processing 1999...
Processing 2000...
Processing 2001...
Processing 2002...
Processing 2003...
Processing 2004...
Processing 2005...
Processing 2006...
Processing 2007...
Processing 2008...
Processing 2009...
Processing 2010...
Processing 2011...
Processing 2012...
Processing 2013...
Processing 2014...
Processing 2015...
Processing 2016...
Processing 2017...
Processing 2018...
```

Out[34]:

	Country	level_1	0
0	Aruba	min 1992	5.000000
1	Aruba	max 1992	63.000000
2	Aruba	mean 1992	30.731343
3	Aruba	count 1992	201.000000
4	Aruba	std 1992	18.759739
...	...	...	...
48190	Zimbabwe	mean 2018	6.837552
48191	Zimbabwe	count 2018	141436.000000
48192	Zimbabwe	std 2018	3.449983
48193	Zimbabwe	median 2018	6.000000
48194	Zimbabwe	mode 2018	6.000000

48195 rows × 3 columns

In [51]:
```
nightLightNoDark = functools.reduce(pd.DataFrame.join, nightLightNoDarkIndiv).stack().reset_index()
nightLightNoDark.loc[:, ["Quantity", "Year"]] = nightLightNoDark.level_1.str.split().apply(pd.Series).rename(columns={0:"Quantity", 1:"Year"})
nightLightNoDarkCSV = nightLightNoDark.rename(columns={0: "Value"}).drop(columns="level_1").set_index(["Year", "Quantity", "Country"]).unstack()["Value"]
nightLightNoDarkCSV
```

Out[51]:

	Country	Afghanistan	Akrotiri Sovereign Base Area	Aland	Albania	Algeria	American Samoa	Andorra	Angola	Anguilla	Antarctica	...	Uzbekistan	Vanuatu	Vatican City	Venezuela	Vietnam	
Year	**Quantity**																	
1992	**count**	2088.000000	135.000000	266.000000	1481.000000	143436.000000	156.000000	626.000000	5237.000000	94.000000	60.000000	...	94961.000000	240.000000	0.0	86842.000000	26758.000000	24.
	max	59.000000	39.000000	53.000000	39.000000	63.000000	32.000000	63.000000	63.000000	21.000000	8.000000	...	63.000000	33.000000	0.0	63.000000	63.000000	5.
	mean	12.062261	15.133333	18.323308	7.809588	10.808354	13.410256	17.734824	10.974031	10.265957	5.083333	...	12.700635	9.245833	0.0	14.048295	6.439121	4.
	median	7.000000	14.000000	13.000000	6.000000	6.000000	13.000000	11.000000	7.000000	9.000000	5.000000	...	9.000000	6.000000	0.0	8.000000	5.000000	4.
	min	3.000000	4.000000	7.000000	4.000000	3.000000	3.000000	4.000000	3.000000	4.000000	4.000000	...	3.000000	4.000000	0.0	3.000000	2.000000	3.

Year	Quantity	Country	Afghanistan	Akrotiri Sovereign Base Area	Aland	Albania	Algeria	American Samoa	Andorra	Angola	Anguilla	Antarctica	...	Uzbekistan	Vanuatu	Vatican City	Venezuela	Vietnam	
...	...		...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2018	mean		6.765070	34.408759	12.460674	9.207076	10.239452	18.377358	24.846695	7.011606	16.060606	6.910486	...	8.777289	11.546882	0.0	9.268471	10.850339	6.
	median		6.000000	34.000000	7.000000	7.000000	7.000000	14.000000	19.000000	6.000000	14.000000	7.000000	...	7.000000	7.000000	0.0	7.000000	7.000000	6.
	min		6.000000	9.000000	6.000000	6.000000	6.000000	6.000000	7.000000	6.000000	7.000000	6.000000	...	6.000000	6.000000	0.0	6.000000	6.000000	6.
	mode		6.000000	51.000000	6.000000	7.000000	7.000000	7.000000	8.000000	6.000000	11.000000	7.000000	...	7.000000	6.000000	0.0	6.000000	7.000000	6.
	std		3.139037	14.801142	15.035082	7.826291	10.018264	11.743688	17.029646	5.044252	6.993241	0.286536	...	6.767632	13.228966	0.0	9.255113	10.719424	0.

189 rows × 255 columns

In [52]:
```
nightLightNoDarkCSV.to_csv("data/nightLight/nightLightNoDark.csv")
del nightLightNoDarkCSV
del nightLightNoDark
del nightLightNoDarkIndiv
```

In [22]:
```
nightLight = pd.read_csv("data/nightLight/nightLight.csv").set_index("Year")
nightLight
```

Out[22]:

Year	Quantity	Afghanistan	Akrotiri Sovereign Base Area	Aland	Albania	Algeria	American Samoa	Andorra	Angola	Anguilla	...	Uzbekistan	Vanuatu	Vatican City	Venezuela	Vietnam	Wal
1992	count	901332.000000	137.000000	2190.000000	43766.000000	3.062687e+06	216.000000	711.000000	1.491515e+06	100.000000	...	698379.000000	14990.000000	0.0	1.076792e+06	402417.000000	166.
1992	max	59.000000	39.000000	53.000000	39.000000	6.300000e+01	32.000000	63.000000	6.300000e+01	21.000000	...	63.000000	33.000000	0.0	6.300000e+01	63.000000	5.
1992	mean	0.027943	14.912409	2.225571	0.264269	5.061918e-01	9.685185	15.614627	3.853196e-02	9.650000	...	1.726949	0.148032	0.0	1.132978e+00	0.428158	0.
1992	median	0.000000	14.000000	0.000000	0.000000	0.000000e+00	8.000000	9.000000	0.000000e+00	9.000000	...	0.000000	0.000000	0.0	0.000000e+00	0.000000	0.
1992	min	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000e+00	0.000000	...	0.000000	0.000000	0.0	0.000000e+00	0.000000	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2018	mean	2.047282	34.408759	4.557534	8.770347	2.811383e+00	18.037037	24.846695	1.631923e+00	15.900000	...	3.258340	1.618412	0.0	4.400938e+00	8.588959	5.
2018	median	0.000000	34.000000	0.000000	7.000000	0.000000e+00	13.500000	19.000000	0.000000e+00	14.000000	...	0.000000	0.000000	0.0	0.000000e+00	7.000000	6.
2018	min	0.000000	9.000000	0.000000	0.000000	0.000000e+00	0.000000	7.000000	0.000000e+00	0.000000	...	0.000000	0.000000	0.0	0.000000e+00	0.000000	0.
2018	mode	0.000000	51.000000	0.000000	7.000000	0.000000e+00	7.000000	8.000000	0.000000e+00	11.000000	...	0.000000	0.000000	0.0	0.000000e+00	7.000000	6.
2018	std	3.555359	14.801142	10.894895	7.885158	6.959879e+00	11.895320	17.029646	3.834231e+00	7.139328	...	5.914810	6.371589	0.0	7.879986e+00	10.506223	1.

189 rows × 256 columns

In [23]:
```
nightLight2013 = states_geojson.merge(pd.read_csv("data/nightLight/nightLight2013.csv").replace("United States of America", "United States").replace('United Republic of Tanzania',
nightLight2013
```

Out[23]:

	ISO3166-1-Alpha-3	country	id	name	geometry	min	max	mean	count	std	median	mode
0	ABW	Aruba	5150	Aruba	POLYGON ((-69.99694 12.57758, -69.93639 12.531...	7.0	57.0	28.666667	201	16.178831	28.0	7.0
1	AFG	Afghanistan	1741	Badghis	POLYGON ((64.30624 35.39722, 64.32468 35.40177...	0.0	32.0	0.236040	29334	1.336606	0.0	0.0
2	AFG	Afghanistan	1742	Hirat	POLYGON ((61.36393 35.59824, 61.36548 35.59850...	0.0	56.0	0.741159	78071	2.932197	0.0	0.0
3	AFG	Afghanistan	1743	Bamyan	POLYGON ((67.74391 35.44342, 67.75476 35.44412...	0.0	8.0	0.180535	25153	1.039803	0.0	0.0
4	AFG	Afghanistan	1744	Balkh	POLYGON ((67.25913 37.18515, 67.28145 37.18866...	0.0	61.0	2.011429	23275	5.675208	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
4663	ZWE	Zimbabwe	529	Manicaland	POLYGON ((33.01165 -17.38399, 32.99745 -17.404...	0.0	46.0	1.915220	42982	3.263857	0.0	0.0
4664	ZWE	Zimbabwe	530	Matabeleland South	POLYGON ((29.43994 -19.87930, 29.45699 -19.874...	0.0	52.0	0.677659	68170	2.175918	0.0	0.0
4665	ZWE	Zimbabwe	531	Bulawayo	POLYGON ((28.49757 -20.06270, 28.50532 -20.062...	0.0	55.0	21.061511	569	15.322821	14.0	7.0
4666	ZWE	Zimbabwe	532	Masvingo	POLYGON ((31.04181 -19.25226, 31.19870 -19.248...	0.0	26.0	0.701861	69850	2.066331	0.0	0.0
4667	ZWE	Zimbabwe	533	Mashonaland West	POLYGON ((30.01065 -15.64623, 30.05024 -15.640...	0.0	49.0	4.252489	70118	3.467990	6.0	6.0

4668 rows × 12 columns

NASA's EaN Blue Marble 2016 Dataset

Link: https://visibleearth.nasa.gov/images/144898/earth-at-night-black-marble-2016-color-maps

Satellite images of Earth at night—often referred to as "night lights"—have been a curiosity for the public and a tool of fundamental research for at least 25 years. They have provided a broad, beautiful picture, showing how humans have shaped the planet and lit up the darkness. Produced every decade or so, such maps have spawned hundreds of pop-culture uses and dozens of economic, social science, and environmental research projects.

These images show Earth's night lights as observed in 2016. The data were reprocessed with new compositing techniques that select the best cloud-free nights in each month over each land mass.

The images are available as JPEG and GeoTIFF, in three different resolutions: 0.1 degrees (`3600x1800`), 3km (`13500x6750`), and 500m (`86400x43200`). The 500m global map is divided into tiles (21600x21600) according to a gridding scheme.

In [45]:
```
# Black Marble
blackMarble2016 = list(np.vectorize("https://eoimages.gsfc.nasa.gov/images/imagerecords/144000/144898/BlackMarble_2016_{}_geo.tif".format)(np.array(["A", "B", "C", "D"], dtype=obj
blackMarble2016
```

Out[45]:
```
['https://eoimages.gsfc.nasa.gov/images/imagerecords/144000/144898/BlackMarble_2016_A1_geo.tif',
 'https://eoimages.gsfc.nasa.gov/images/imagerecords/144000/144898/BlackMarble_2016_A2_geo.tif',
 'https://eoimages.gsfc.nasa.gov/images/imagerecords/144000/144898/BlackMarble_2016_B1_geo.tif',
 'https://eoimages.gsfc.nasa.gov/images/imagerecords/144000/144898/BlackMarble_2016_B2_geo.tif',
 'https://eoimages.gsfc.nasa.gov/images/imagerecords/144000/144898/BlackMarble_2016_C1_geo.tif',
 'https://eoimages.gsfc.nasa.gov/images/imagerecords/144000/144898/BlackMarble_2016_C2_geo.tif',
 'https://eoimages.gsfc.nasa.gov/images/imagerecords/144000/144898/BlackMarble_2016_D1_geo.tif',
 'https://eoimages.gsfc.nasa.gov/images/imagerecords/144000/144898/BlackMarble_2016_D2_geo.tif']
```

The Globe at Night (GaN) Dataset

Globe at Night collects data based on specific locations, and in this case, contains a column named `LimitingMag` which can be related to Light Pollution standards in the region. The following commands showcase a way to download the dataset programmatically, while also removing unnecessary datasets.

In [46]:
```
gan_url = "https://www.globeatnight.org/"
files = [gan_url + i["href"] for i in BeautifulSoup(requests.get(gan_url+"maps.php").content, "lxml").findAll(href=re.compile("\.csv$"))]
!mkdir "data/gan"
gan = []
for file in files:
    filename = "data/gan/"+file.split("/")[-1]
    print(file, "==>", filename)
```

```
    file = BytesIO(requests.get(file, allow_redirects=True).content)
    data = pd.read_csv(file, error_bad_lines=False)[["Latitude", "Longitude", "LocalDate", "LocalTime", "UTDate", "UTTime", "LimitingMag", "Country"]]
    data = data[data.LimitingMag > 0]
    data.LocalTime = pd.to_datetime(data.apply(lambda row: row["LocalDate"] + " " + row["LocalTime"], axis=1), format='%Y-%m-%d %H:%M')
    data.UTTime = pd.to_datetime(data.apply(lambda row: row["UTDate"] + " " + row["UTTime"], axis=1), format='%Y-%m-%d %H:%M')
    data.loc[:, "Year"] = int(filename[-8:-4])
    data = data[["Latitude", "Longitude", "LocalTime", "UTTime", "LimitingMag", "Country", "Year"]]
    data.to_csv(filename)
    gan.append(data)

gan = pd.concat(gan, ignore_index=True)
gan.to_csv("data/gan/GaN.csv", index=False)
```

```
A subdirectory or file data/gan already exists.
https://www.globeatnight.org/2020data/GaN2020.csv ==> data/gan/GaN2020.csv
b'Skipping line 2574: expected 18 fields, saw 19\n'
https://www.globeatnight.org/2019data/GaN2019.csv ==> data/gan/GaN2019.csv
b'Skipping line 5547: expected 18 fields, saw 19\n'
https://www.globeatnight.org/2018data/GaN2018.csv ==> data/gan/GaN2018.csv
https://www.globeatnight.org/2017data/GaN2017.csv ==> data/gan/GaN2017.csv
https://www.globeatnight.org/2016data/GaN2016.csv ==> data/gan/GaN2016.csv
https://www.globeatnight.org/2015data/GaN2015.csv ==> data/gan/GaN2015.csv
https://www.globeatnight.org/2014data/GaN2014.csv ==> data/gan/GaN2014.csv
https://www.globeatnight.org/2013data/GaN2013.csv ==> data/gan/GaN2013.csv
https://www.globeatnight.org/2012data/GaN2012.csv ==> data/gan/GaN2012.csv
b'Skipping line 10111: expected 18 fields, saw 19\n'
https://www.globeatnight.org/2011data/GaN2011.csv ==> data/gan/GaN2011.csv
https://www.globeatnight.org/2010data/GaN2010.csv ==> data/gan/GaN2010.csv
https://www.globeatnight.org/2009data/GaN2009.csv ==> data/gan/GaN2009.csv
https://www.globeatnight.org/2008data/GaN2008.csv ==> data/gan/GaN2008.csv
https://www.globeatnight.org/2007data/GaN2007.csv ==> data/gan/GaN2007.csv
https://www.globeatnight.org/2006data/GaN2006.csv ==> data/gan/GaN2006.csv
```

In [24]:

```
gan = pd.read_csv("data/gan/GaN.csv").sort_values(["Year", "Country"], ignore_index=True)
gan.Country = gan.Country.str.replace("United States.*", "United States").str.replace("Republic of the Union of Myanmar", "Myanmar").replace("Republic of the Congo", "Congo Republ
gan.UTTime = pd.to_datetime(gan.UTTime)
gan.LocalTime = pd.to_datetime(gan.LocalTime)
gan = pd.merge(gan[gan.Latitude < 10**6], geocountries_latlong.rename(columns={"latitude":"countryLatitude", "longitude":"countryLongitude"}))
gan = gp.GeoDataFrame(gan, geometry=gp.points_from_xy(gan.Longitude, gan.Latitude))
gan
```

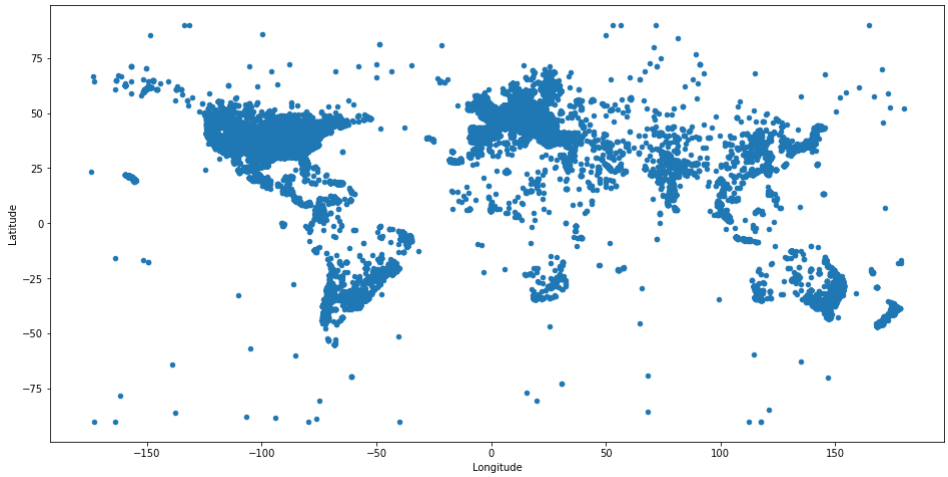
Out[24]:

	Latitude	Longitude	LocalTime	UTTime	LimitingMag	Country	Year	CountryCode	CountryCode3	NumericCountryCode	fips	Capital	Area in km²	Population	Continent	countryLatitude	country
0	34.56000	65.4300	2006-03-23 21:00:00	2006-03-23 16:30:00	7.0	Afghanistan	2006	AF	AFG	4	AF	Kabul	647500.0	37172386	AS	33.939110	
1	31.86000	64.2100	2010-03-06 12:30:00	2010-03-06 08:00:00	6.0	Afghanistan	2010	AF	AFG	4	AF	Kabul	647500.0	37172386	AS	33.939110	
2	31.51000	64.1300	2010-03-06 20:30:00	2010-03-06 16:00:00	6.0	Afghanistan	2010	AF	AFG	4	AF	Kabul	647500.0	37172386	AS	33.939110	
3	31.85650	64.2108	2010-03-11 20:16:00	2010-03-11 15:46:00	6.0	Afghanistan	2010	AF	AFG	4	AF	Kabul	647500.0	37172386	AS	33.939110	
4	31.85600	64.2100	2010-03-12 20:10:00	2010-03-12 15:40:00	6.0	Afghanistan	2010	AF	AFG	4	AF	Kabul	647500.0	37172386	AS	33.939110	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
189325	-29.04100	167.9610	2020-06-21 20:50:00	2020-06-21 09:50:00	6.0	Norfolk Island	2020	NF	NFK	574	NF	Kingston	34.6	1828	OC	-29.040835	
189326	-29.04090	167.9850	2020-06-21 21:13:00	2020-06-21 10:13:00	6.0	Norfolk Island	2020	NF	NFK	574	NF	Kingston	34.6	1828	OC	-29.040835	
189327	-29.03800	167.9630	2020-06-21 21:49:00	2020-06-21 10:49:00	5.0	Norfolk Island	2020	NF	NFK	574	NF	Kingston	34.6	1828	OC	-29.040835	
189328	7.34623	134.4530	2020-06-21 20:28:00	2020-06-21 11:28:00	3.0	Palau	2020	PW	PLW	585	PS	Melekeok	458.0	17907	OC	7.514980	
189329	52.22880	21.3435	2020-04-16 20:00:00	2020-04-16 18:00:00	3.0	Pitcairn Islands	2020	PN	PCN	612	PC	Adamstown	47.0	46	OC	-24.703615	

189330 rows × 18 columns

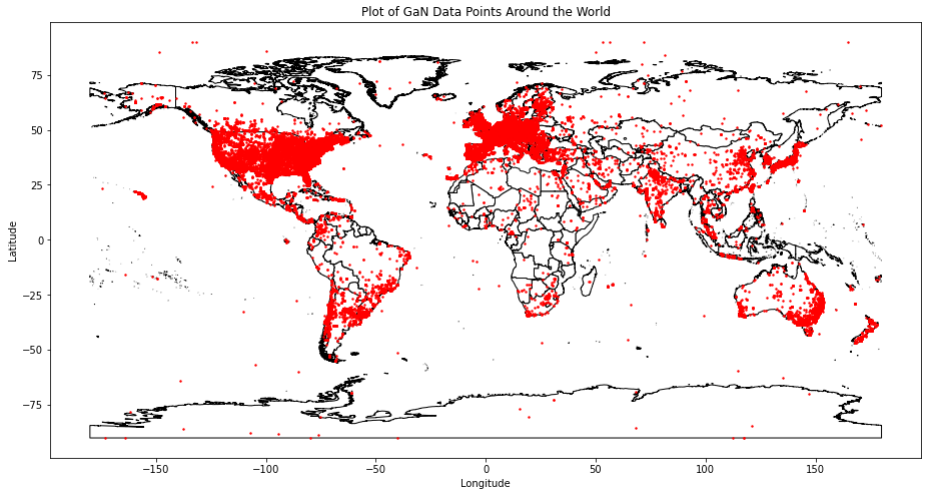
In [30]:

```
_ = gan.plot.scatter(x="Longitude", y="Latitude")
```



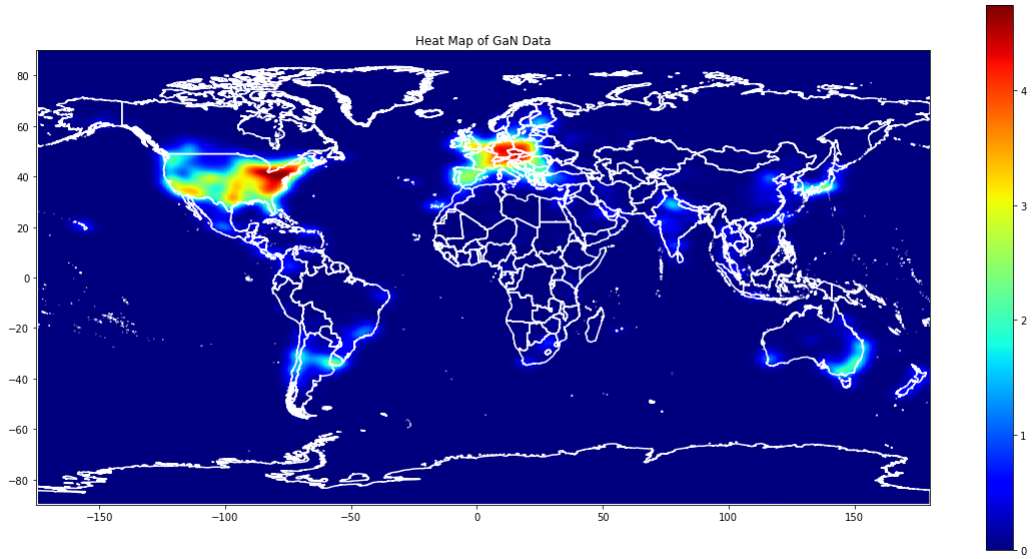
In [31]:

```
base = world.plot(color='white', edgecolor='black')
gan[["geometry"]].plot(ax=base, marker='o', color='red', markersize=2)
_ = (base.set_xlabel("Longitude"), base.set_ylabel("Latitude"), base.set_title("Plot of GaN Data Points Around the World"))
```



In [32]:

```
heatmap, xedges, yedges = np.histogram2d(gan.Latitude, gan.Longitude, bins=250)

logheatmap = np.log(heatmap)
logheatmap[np.isneginf(logheatmap)] = 0
logheatmap = sp.ndimage.filters.gaussian_filter(logheatmap, 2, mode='nearest')

plt.figure(figsize=(20, 10))

plt.imshow(logheatmap, cmap="jet", extent=[yedges[0], yedges[-1], xedges[-1], xedges[0]])
plt.colorbar()

ax = plt.gca()
ax.invert_yaxis()
ax.set_xlim(-175,180)

world.boundary.plot(edgecolor='white', ax=ax)
_ = ax.set_title("Heat Map of GaN Data")
```



In [25]:

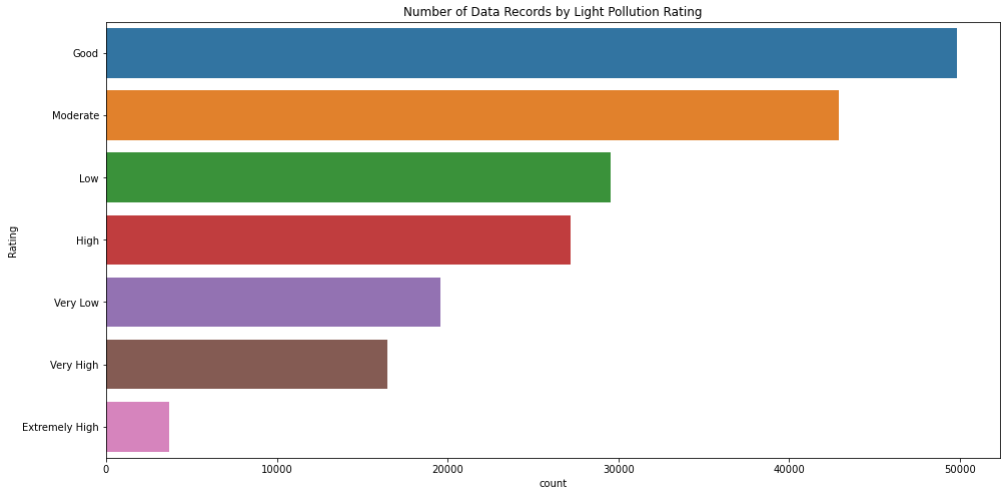
```
gan.loc[:, "Rating"] = gan.LimitingMag.apply(lambda p: "Extremely High" if p >= 7 else "Very High" if p >= 6 else "High" if p >= 5 else "Moderate" if p >= 4 else "Good" if p >= 3 e
gan.Rating.value_counts()
```

Out[25]:

```
Good              49844
Moderate          42924
Low               29529
High              27221
Very Low          19610
Very High         16488
Extremely High     3714
Name: Rating, dtype: int64
```

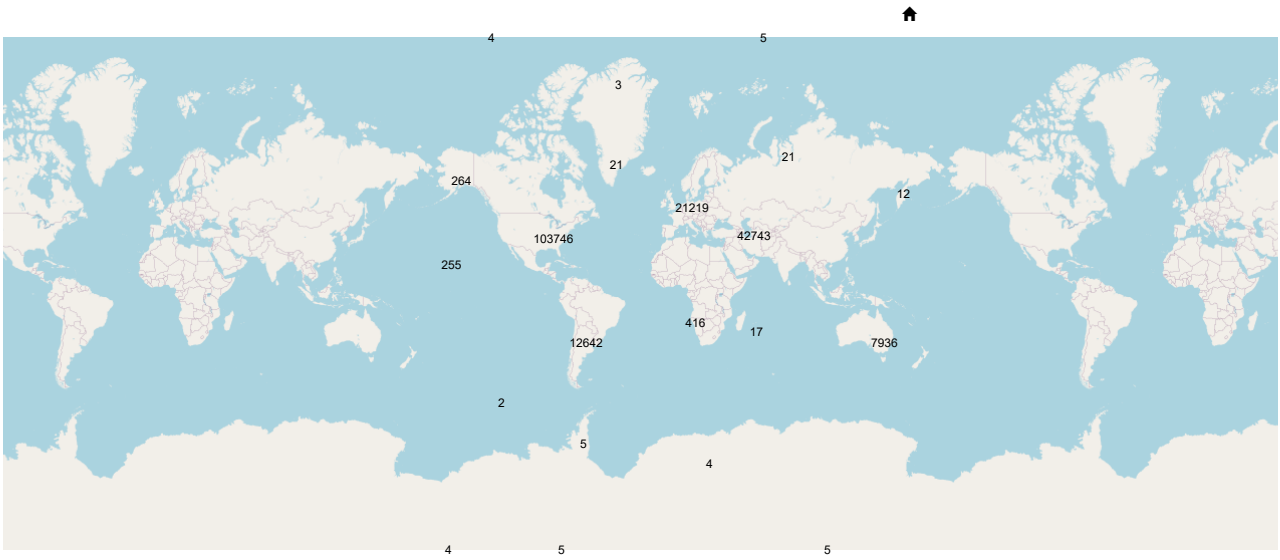
In [34]:

```
_ = sns.countplot(y=gan.Rating, order=gan.Rating.value_counts().index).set_title("Number of Data Records by Light Pollution Rating")
```



In [61]:
```
m = folium.Map(location=[0, 0], zoom_start=1)
plugins.FastMarkerCluster(gan[["Latitude", "Longitude", "LimitingMag"]].to_numpy()).add_to(m)
plugins.Fullscreen().add_to(m)
m.save("maps/ganMarkerClusters.html")
m
```

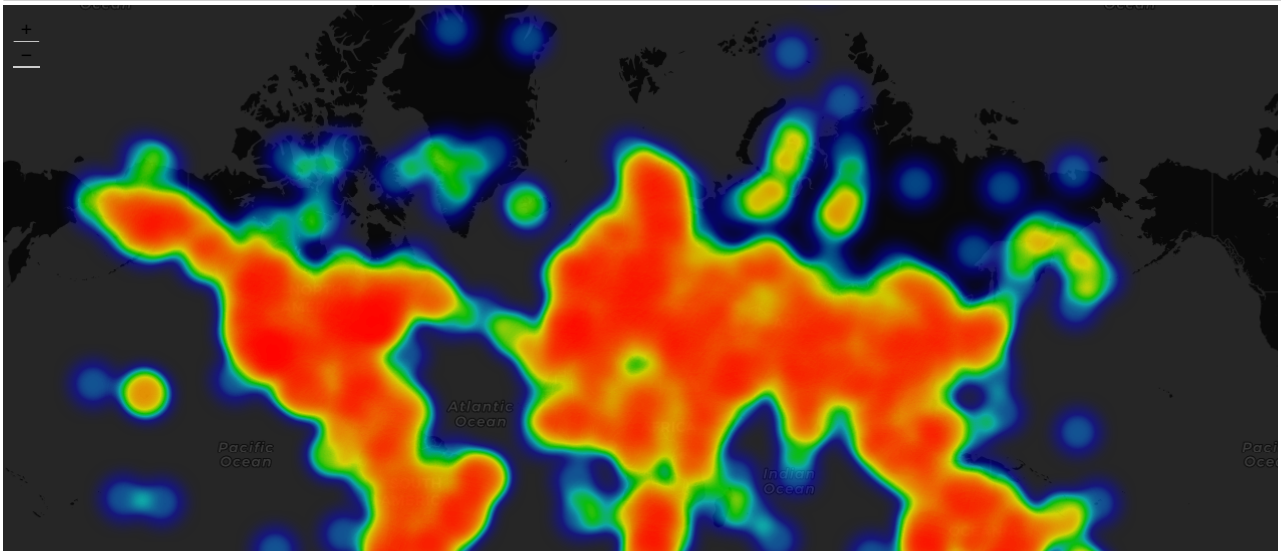
Out[61]: Make this Notebook Trusted to load map: File -> Trust Notebook

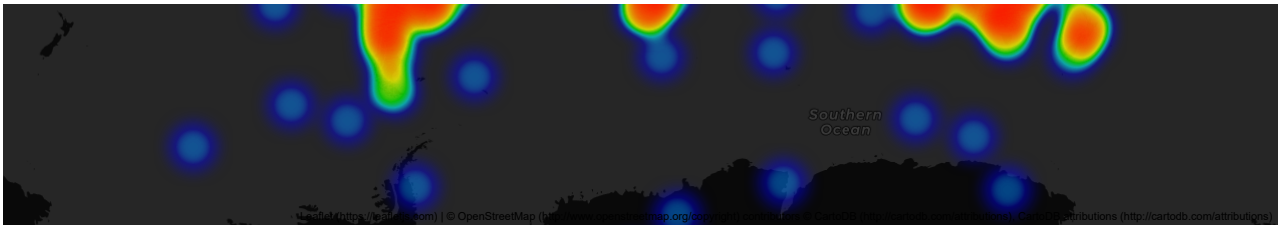


Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

In [62]:
```
latlonglim = gan.set_index("UTTime").sort_index()[["Latitude", "Longitude", "LimitingMag"]]
latlonglim.LimitingMag /= 7.0
m = folium.Map(location = [15,30], tiles='Cartodb dark_matter', zoom_start = 2)
plugins.HeatMap(latlonglim.to_numpy()).add_to(m)
plugins.Fullscreen().add_to(m)
m.save("maps/ganHeatMap.html")
m
```

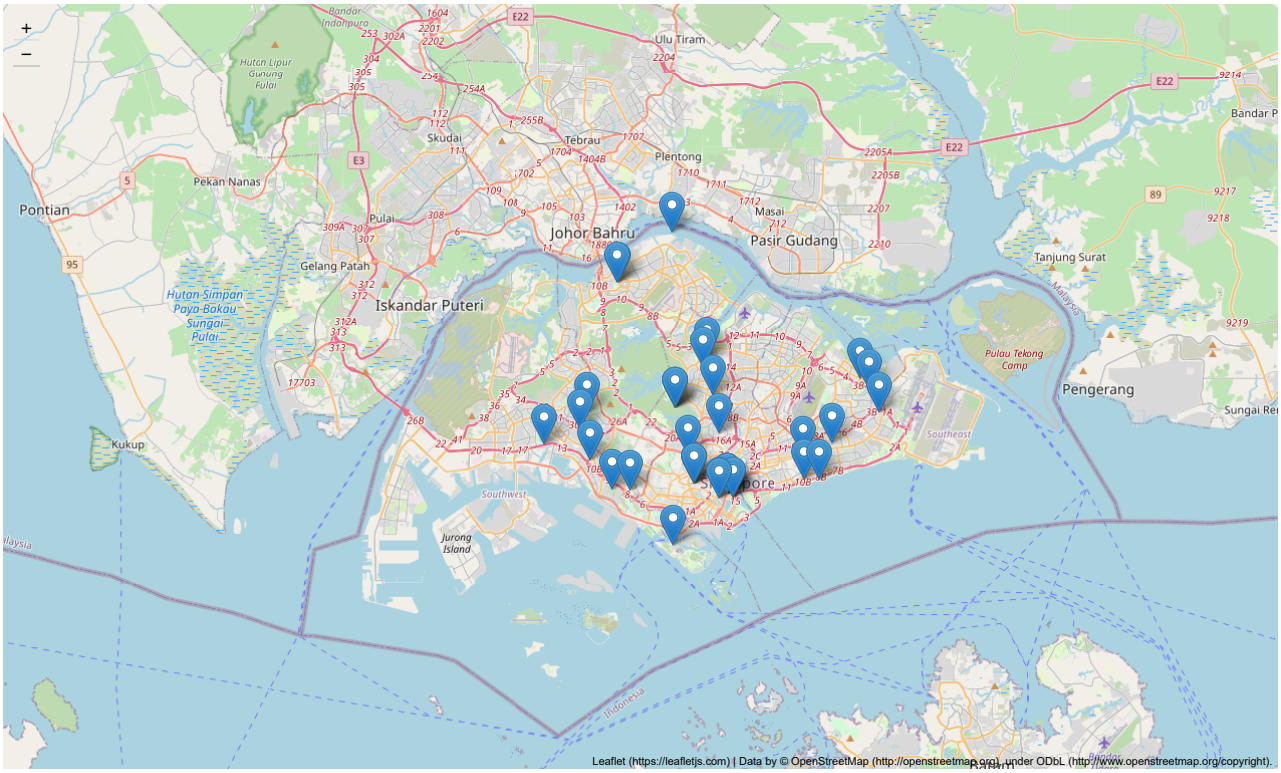
Out[62]:





In [159]:
```
ganSG = gan[gan.Country == "Singapore"].sort_values(["UTTime", "LimitingMag"])
m = folium.Map(location=[ganSG.countryLatitude.mean(), ganSG.countryLongitude.mean()], zoom_start=11)
ganSG.apply(lambda row: folium.Marker([row.Latitude, row.Longitude], popup="<span style=\"color:#0000FF\">{} back in {}</span>".format(row.LimitingMag, row.Year)).add_to(m), axis =
plugins.Fullscreen().add_to(m)
m.save("maps/ganSG.html")
m
```

Out[159]:



Exploratory Data Analysis

Research Question A: What are locations of minimal light pollution intensity which are optimum for astronomical observation?

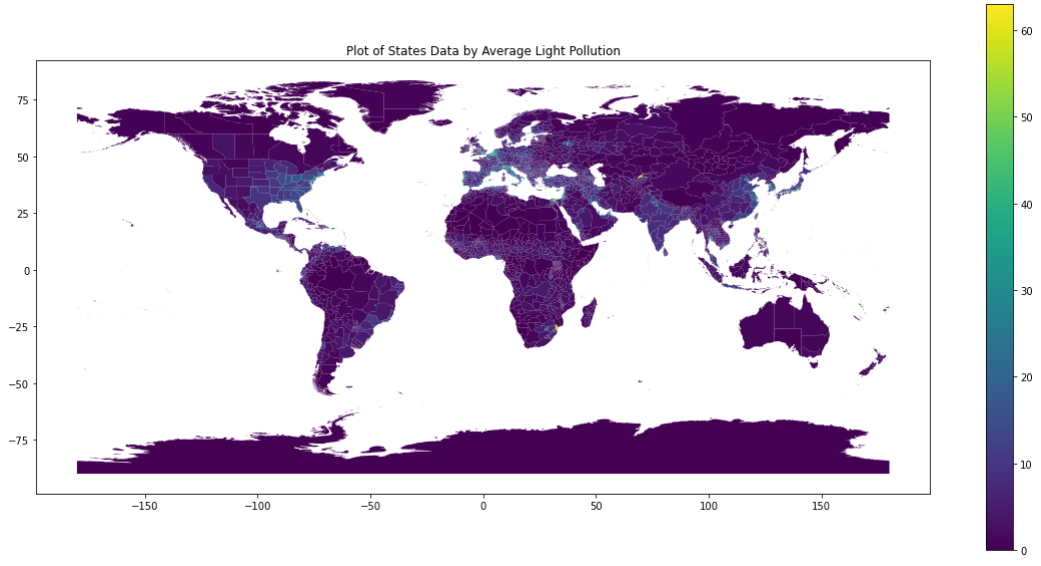
While one might think that the best locations are in the middle of wilderness or large water bodies (i.e. the ocean), as are where the observatories Arecibo and FAST are situated, some of these locations need to be filtered based on accessibility, especially since locations like the middle of the ocean are not feasible locations for people to assemble to watch and will thus not be a great location for astrophotography. Additionally, we must also be able to identify locations for observatories to be located.

For this question, I will be using the nightLight2013 Dataset as retrieved above. I have also placed the retrieval instructions here for new users.

```
nightLight2013 = states_geojson.merge(pd.read_csv("data/nightLight/nightLight2013.csv"))
```

In [31]:
```
nightLight2013.plot("mean", figsize=(20, 10), legend=True).set_title("Plot of States Data by Average Light Pollution")
```

Out[31]:
```
Text(0.5, 1.0, 'Plot of States Data by Average Light Pollution')
```



Based on States

Firstly, we analyse the data with respect to states to find the best specific states for Astronomical Observations. For this, consider countries that have an Average Light Pollution of `0`, which represents that it is in an optimal light pollution capacity.

In [26]:
```
nightLight2013.dropna(subset=["id"]).drop(columns=["ISO3166-1-Alpha-3", "id", "geometry", "count", "country", "name"])
```

Out[26]:

	min	max	mean	std	median	mode
0	7.0	57.0	28.666667	16.178831	28.0	7.0
1	0.0	32.0	0.236040	1.336606	0.0	0.0
2	0.0	56.0	0.741159	2.932197	0.0	0.0
3	0.0	8.0	0.180535	1.039803	0.0	0.0
4	0.0	61.0	2.011429	5.675208	0.0	0.0
...	...	...	...	...	...	...
4663	0.0	46.0	1.915220	3.263857	0.0	0.0
4664	0.0	52.0	0.677659	2.175918	0.0	0.0
4665	0.0	55.0	21.061511	15.322821	14.0	7.0
4666	0.0	26.0	0.701861	2.066331	0.0	0.0
4667	0.0	49.0	4.252489	3.467990	6.0	6.0

4634 rows × 6 columns

In [27]:
```
cleanedNightLight2013 = nightLight2013.dropna(subset=["id"])
zero = cleanedNightLight2013[(cleanedNightLight2013.drop(columns=["ISO3166-1-Alpha-3", "id", "geometry", "count", "country", "name"]) == 0).all(axis=1)].dropna().set_index(["countr
zero
```

Out[27]:

		ISO3166-1-Alpha-3	id	geometry	min	max	mean	count	std	median	mode
country	**name**										
Anguilla	**Island Harbour**	AIA	5117	POLYGON ((-63.00755 18.26734, -63.00739 18.273...	0.0	0.0	0.0	0	0.0	0.0	0.0
Aland	**Geta**	ALD	4813	MULTIPOLYGON (((19.72763 60.37295, 19.71738 60...	0.0	0.0	0.0	113	0.0	0.0	0.0
	Föglö	ALD	4814	MULTIPOLYGON (((20.44706 59.99518, 20.39858 59...	0.0	0.0	0.0	203	0.0	0.0	0.0
	Lemland	ALD	4818	MULTIPOLYGON (((19.92799 59.84052, 19.92213 59...	0.0	0.0	0.0	192	0.0	0.0	0.0
	Sottunga	ALD	4821	MULTIPOLYGON (((20.82333 60.06745, 20.82163 60...	0.0	0.0	0.0	31	0.0	0.0	0.0
...	**...**	...	...	...	...	...	...	...	...	...	...
Vanuatu	**Penama**	VUT	564	MULTIPOLYGON (((168.22682 -15.99090, 168.20427...	0.0	0.0	0.0	1475	0.0	0.0	0.0
Samoa	**Atua**	WSM	4987	POLYGON ((-171.58515 -13.88388, -171.59918 -13...	0.0	0.0	0.0	323	0.0	0.0	0.0
	Va'a-o-Fonoti	WSM	4988	MULTIPOLYGON (((-171.57002 -13.93816, -171.564...	0.0	0.0	0.0	61	0.0	0.0	0.0
	Gaga'ifomauga	WSM	4991	POLYGON ((-172.41418 -13.63687, -172.57875 -13...	0.0	0.0	0.0	495	0.0	0.0	0.0
	Vaisigano	WSM	4992	POLYGON ((-172.63899 -13.52207, -172.63181 -13...	0.0	0.0	0.0	144	0.0	0.0	0.0

94 rows × 10 columns

Astronomical observation often entails the creation of massive observatories and telescopes to witness that part of the sky. Hence, it is an important consideration to ensure that an observatory is maintained.

In this section, we use the Arecibo Observatory as the base of our investigation. A former observatory that recently collapsed in December 2020, the Arecibo Observatory is built as a radio reflector dish with a diameter of 305 metres, making it one of the largest observatories and easily one of the most recognisable telescopes on Earth. The observatory covers an area of about twenty acres, which roughly translates to about `0.0809371 square kilometers` . To account for additional space taken up by labs and rooms in the observatory, we round this up to about `0.1 square kilometers` .

Below is the Arecibo Observatory, as shown in the form of an ArcGIS Folium Map.

In [34]:
```
m = folium.Map(location=[18.3475711,-66.7540715], zoom_start=16)
tile = folium.TileLayer(tiles='https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', attr='Esri', name='Esri Satellite', overlay=False, c
plugins.Fullscreen().add_to(m)
m
```

Out[34]:

Another telescope to be considered is the Five-hundred-meter Aperture Spherical Radio Telescope (FAST) in Southern China, which has started to be used for astronomical observation tasks recently. As the name suggests, this telescope has a diameter roughly 500 metres, hence in order to derive the approximate area, we can apply a rough area-based scaling formula as shown below:

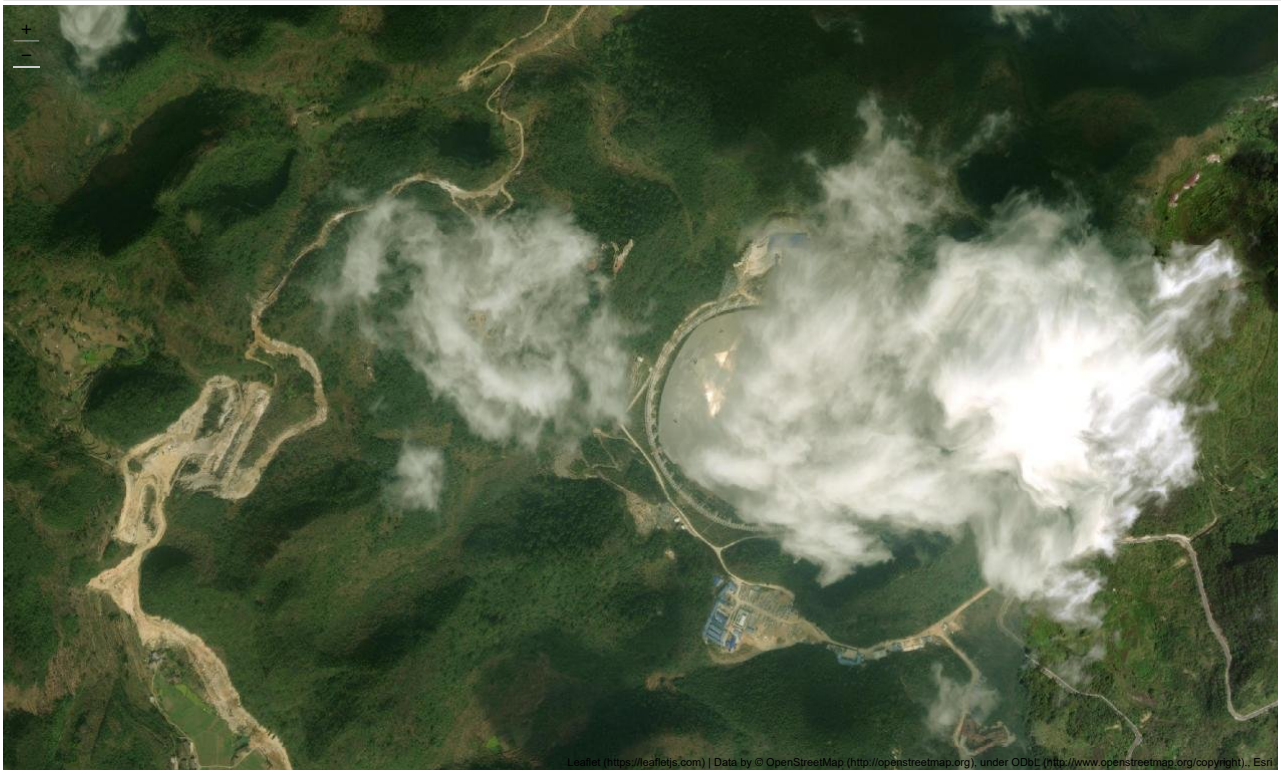
$$
\begin{aligned}
Area_{FAST} &= \frac{Area_{FASTRadarDisk}}{Area_{AcrediboRadarDisk}} Area_{Acredibo} \\
&= \frac{\pi r_{FAST}^2}{\pi r_{Acredibo}^2} Area_{Acredibo} \\
&= \frac{r_{FAST}^2}{r_{Acredibo}^2} Area_{Acredibo} \\
&= \frac{d_{FAST}^2}{d_{Acredibo}^2} Area_{Acredibo} \\
&= \left(\frac{500}{305}\right)^2 \cdot (0.0809371) \\
&= 0.21751437785541522 km^2
\end{aligned}
$$

Hence, we retrieve that FAST has a rough area coverage of `0.21751437785541522 square kilometers`, more that twice of that of Acredibo. Hence, this is also something of consideration.

Below is the FAST Observatory in a similar ArcGIS Folium Map, but unfortunately covered by clouds.

In [35]:
```
m = folium.Map(location=[25.6530326,106.854602], zoom_start=16)
tile = folium.TileLayer(tiles='https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', attr='Esri', name='Esri Satellite', overlay=False, c
plugins.Fullscreen().add_to(m)
m
```

Out[35]:



Keeping in mind possible human life living in the area, we state that we need an area of at least 1 square kilometer. This is to ensure that the telescope personnel also have the ability to live around it. Another measure of this is the Population Density, which is available with the `popsaden` DataFrame.

To check if this area is enough, we use the `geojson_area` package, which can be installed simply with the command:

```
pip install area
```

In [28]:
```
areas = (pd.DataFrame(eval(nightLight2013.geometry.to_json())["features"]).geometry.apply(geojson_area) / (10e6))
areaNightLight2013 = nightLight2013.reset_index()
areaNightLight2013.loc[:, "area"] = areas
areaNightLight2013.groupby("country").area.sum()
```

Out[28]:
```
country
Afghanistan                     68174.381441
Akrotiri Sovereign Base Area        9.892375
Aland                             107.135032
Albania                          3080.575554
Algeria                        231747.927378
                                     ...
West Bank                         596.331913
Western Sahara                   9089.943118
Yemen                           45566.711865
Zambia                          75641.981758
Zimbabwe                        39140.353879
Name: area, Length: 256, dtype: float64
```

As was established previously, this data is slightly off from the real result since the polygons are not fully in line with with the real shape of the country / city. However, this still gives a reasonable estimate for how large the area is and hence how capable it is to hold an observatory the size of Arecibo or even FAST.

In [29]:
```
idealAreaNightLight = zero.join(areaNightLight2013.set_index(["country", "name"])[["area"]]).reset_index().sort_values("area", ascending=False)
idealAreaNightLight
```

Out[29]:

	country	name	ISO3166-1-Alpha-3	id	geometry	min	max	mean	count	std	median	mode	area
11	Chad	Tibesti	TCD	5578	POLYGON ((18.91712 21.99685, 18.71913 21.84981...	0.0	0.0	0.0	150196	0.0	0.0	0.0	12048.318327
72	Somalia	Bakool	SOM	2312	POLYGON ((44.59681 4.91764, 44.67999 4.40996, ...	0.0	0.0	0.0	30456	0.0	0.0	0.0	2614.582286
73	Somalia	Jubbada Dhexe	SOM	2315	POLYGON ((42.97994 1.45996, 43.26995 1.02996, ...	0.0	0.0	0.0	22054	0.0	0.0	0.0	1898.018781
87	Uganda	Buvuma	UGA	5598	POLYGON ((33.38704 0.30262, 33.42621 0.21373, ...	0.0	0.0	0.0	9033	0.0	0.0	0.0	771.087525
21	French Southern Territories	Archipel des Kerguelen	ATF	5916	MULTIPOLYGON (((70.01612 -49.56422, 70.00709 -...	0.0	0.0	0.0	12442	0.0	0.0	0.0	697.983683
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8	Antigua and Barbuda	Redonda	ATG	5092	POLYGON ((-62.34559 16.93914, -62.34461 16.940...	0.0	0.0	0.0	0	0.0	0.0	0.0	0.037134
48	New Zealand	Bounty Islands	NZL	5473	MULTIPOLYGON (((179.07219 -47.71364, 179.07252...	0.0	0.0	0.0	1	0.0	0.0	0.0	0.036215
43	Mauritius	Cargados Carajos Shoals	MUS	5182	MULTIPOLYGON (((59.49830 -16.81935, 59.49204 -...	0.0	0.0	0.0	1	0.0	0.0	0.0	0.027513
39	Maldives	Vaavu	MDV	5235	MULTIPOLYGON (((73.51913 3.35659, 73.51922 3.3...	0.0	0.0	0.0	0	0.0	0.0	0.0	0.015986

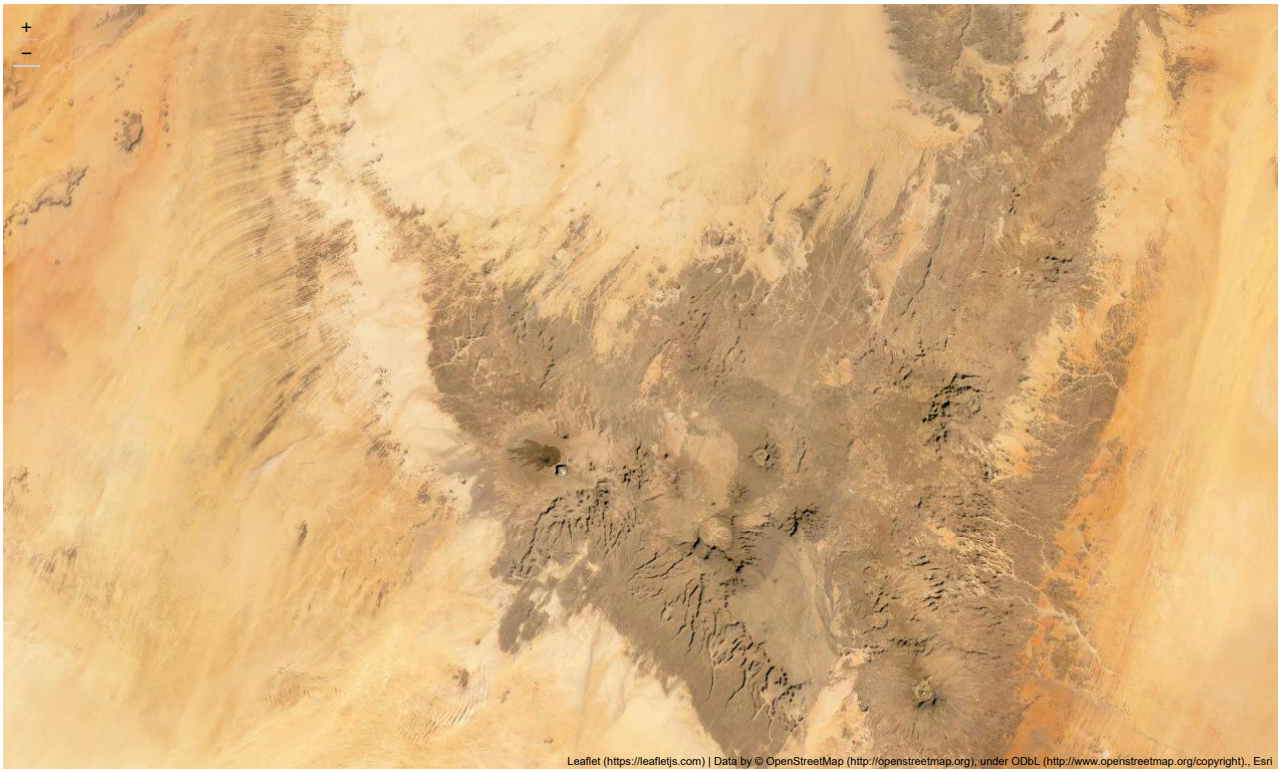
	country	name	ISO3166-1-Alpha-3	id	geometry	min	max	mean	count	std	median	mode	area
67	Serranilla Bank	Serranilla Bank	SER	5485	POLYGON ((-78.63707 15.86209, -78.64041 15.864...	0.0	0.0	0.0	0	0.0	0.0	0.0	0.010599

94 rows × 13 columns

Hence, another location in consideration is the Tibesti Mountain Range in Chad, which is a similar location to FAST in Guizhou and Arecibo in Puerto Rico, but in drylands, which may make it not as desirable as one would think. Below is Tibesti, as shown by the map.

In [38]:
```
m = folium.Map(location=[21.3010105,17.1899371], zoom_start=7.75)
tile = folium.TileLayer(tiles='https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', attr='Esri', name='Esri Satellite', overlay=False, c
plugins.Fullscreen().add_to(m)
m
```

Out[38]:



Next, we note the Population Density of each area. Since the population density of the individual territories is not available, we instead use the country's average population density, since it gives a reasonable estimate for the population density of that location. This population density data has been previously saved in the `popsaden` DataFrame, and we simply filter out the ones currently saved in the above datasets. Following this, we merge the data into `idealAreaNightLight` to get `denseNightLight2013`.

In [30]:
```
pops = popsaden.groupby("Country").PopulationDensity.mean().sort_values().reset_index()
pops = pops[pops.Country.isin(zero.reset_index().country.value_counts().index)].reset_index().groupby("Country").PopulationDensity.mean().sort_values()
pops
```

Out[30]:
```
Country
Greenland                      0.137980
Australia                      2.997850
Chad                          10.525000
New Zealand                   17.081100
Solomon Islands               20.575575
Somalia                       20.930750
Vanuatu                       21.142025
Latvia                        33.073525
Palau                         44.180880
Fiji                          47.192575
Samoa                         66.972200
Cook Islands                  76.535729
Slovenia                     101.705075
Northern Mariana Islands     124.152600
Tonga                        142.841325
Anguilla                     150.940750
Malawi                       168.287300
Uganda                       182.109200
Antigua and Barbuda          205.702825
United Kingdom               266.611625
American Samoa               282.356667
Nauru                        530.675000
Mauritius                    616.271550
Malta                       1325.946075
Maldives                    1427.247475
Name: PopulationDensity, dtype: float64
```

In [31]:
```
denseNightLight2013 = idealAreaNightLight.merge(pops.reset_index().rename(columns={"Country":"country", "PopulationDensity":"Density"}))
denseNightLight2013 = denseNightLight2013.sort_values("area", ignore_index=True, ascending=False).sort_values("Density", ignore_index=True)
denseNightLight2013
```

Out[31]:

	country	name	ISO3166-1-Alpha-3	id	geometry	min	max	mean	count	std	median	mode	area	Density
0	Greenland	Pituffik	GRL	2738	POLYGON ((-67.64287 76.51859, -67.91425 76.518...	0.0	0.0	0.0	0	0.0	0.0	0.0	65.763395	0.137980
1	Australia	Norfolk Island	AUS	2659	POLYGON ((159.04444 -31.51922, 159.06886 -31.5...	0.0	0.0	0.0	19	0.0	0.0	0.0	1.429299	2.997850
2	Chad	Tibesti	TCD	5578	POLYGON ((18.91712 21.99685, 18.71913 21.84981...	0.0	0.0	0.0	150196	0.0	0.0	0.0	12048.318327	10.525000
3	New Zealand	Campbell Islands	NZL	5478	POLYGON ((169.25042 -52.52026, 169.26044 -52.5...	0.0	0.0	0.0	252	0.0	0.0	0.0	13.443335	17.081100
4	New Zealand	The Snares	NZL	5480	POLYGON ((166.63722 -47.99358, 166.63209 -48.0...	0.0	0.0	0.0	28	0.0	0.0	0.0	1.570404	17.081100
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
70	Malta	Ta' Xbiex	MLT	5941	POLYGON ((14.50156 35.89517, 14.49801 35.89251...	0.0	0.0	0.0	0	0.0	0.0	0.0	0.055168	1325.946075
71	Maldives	Meemu	MDV	5236	MULTIPOLYGON (((73.35149 2.77074, 73.35141 2.7...	0.0	0.0	0.0	2	0.0	0.0	0.0	0.139145	1427.247475
72	Maldives	Dhaalu	MDV	5237	MULTIPOLYGON (((72.90642 2.67967, 72.90447 2.6...	0.0	0.0	0.0	1	0.0	0.0	0.0	0.168613	1427.247475
73	Maldives	Gaafu Alifu	MDV	5240	MULTIPOLYGON (((73.51336 0.38768, 73.50864 0.3...	0.0	0.0	0.0	1	0.0	0.0	0.0	0.119142	1427.247475
74	Maldives	Vaavu	MDV	5235	MULTIPOLYGON (((73.51913 3.35659, 73.51922 3.3...	0.0	0.0	0.0	0	0.0	0.0	0.0	0.015986	1427.247475

75 rows × 14 columns

From here, we can determine that Norfolk Island is one of the better areas to have an observatory located. Below is Norfolk Island in its full glory. Norfolk Island is situated between Northland, New Zealand and Brisbane, Australia, but is considerably remote such that light pollution is quite small. However, it may not be a location that working astronmers at universities would like to continually commute to, since it is only accessible by plane.

In [41]:
```
m = folium.Map(location=[-29.037,167.9557687], zoom_start=13)
tile = folium.TileLayer(tiles='https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', attr='Esri', name='Esri Satellite', overlay=False, c
plugins.Fullscreen().add_to(m)
m
```

Out[41]:

Another consideration would be proximity to a top-ranking university, since astronomers would prefer working nearer to a recognized universty with good equipment. To determine this information, we look at the Webometrics Ranking Web of Universities. We derive data from these sites and locate the best university in this area. This tells us the level of the university in the area.

In [32]:
```
geocountries = geocountries[~geocountries.Country.isin(["Anguilla", "Antigua and Barbuda", "Greenland"])]
geocountries.loc[:, "webometrics_continent"] = geocountries.Continent.replace("EU", "Europe").replace("AS", "Asia").replace("NA", "North_america").replace("AF", "Africa").replace("
webometrics = denseNightLight2013.merge(geocountries[~geocountries.webometrics_continent.isin(["antarctica"])][["Country", "webometrics_continent"]].rename(columns={"Country":"cou
webometrics.loc[:, "universityRank"] = webometrics.apply(axis=1, func=lambda row: pd.read_html(requests.get("https://www.webometrics.info/en/{}/{}%20".format(row.continent, row.co
webometrics.dropna(inplace=True)
webometrics
```

Out[32]:

	continent	country	universityRank
0	Africa	Chad	25331.0
1	Africa	Malawi	3858.0
2	Africa	Mauritius	2163.0
3	Africa	Somalia	11784.0
4	Africa	Uganda	1105.0
5	Asia	Maldives	13521.0
6	Europe	Latvia	1181.0
7	Europe	Malta	857.0
8	Europe	Slovenia	327.0
9	Europe	United Kingdom	6.0
10	Oceania	American Samoa	8567.0
11	Oceania	Australia	41.0
13	Oceania	Fiji	1958.0
15	Oceania	New Zealand	153.0
16	Oceania	Northern Mariana Islands	14570.0
17	Oceania	Palau	15146.0
18	Oceania	Samoa	10091.0
19	Oceania	Solomon Islands	18442.0
20	Oceania	Tonga	25131.0
21	Oceania	Vanuatu	29194.0

In [33]:
```
rankings = denseNightLight2013.merge(webometrics).sort_values("universityRank")
rankings.head()
```

Out[33]:

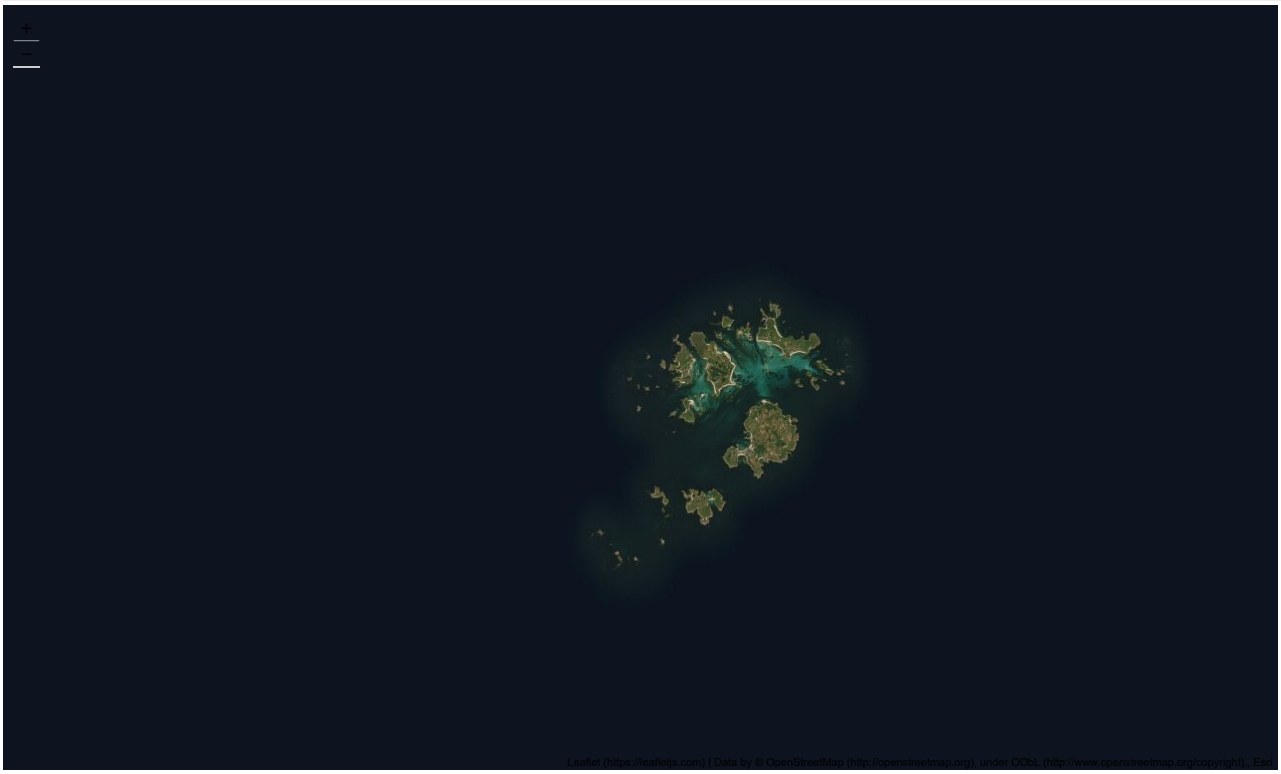
	country	name	ISO3166-1-Alpha-3	id	geometry	min	max	mean	count	std	median	mode	area	Density	continent	universityRank
52	United Kingdom	Isles of Scilly	GBR	5988	MULTIPOLYGON (((-6.34268 49.88394, -6.34724 49...	0.0	0.0	0.0	17	0.0	0.0	0.0	1.057997	266.611625	Europe	6.0
0	Australia	Norfolk Island	AUS	2659	POLYGON ((159.04444 -31.51922, 159.06886 -31.5...	0.0	0.0	0.0	19	0.0	0.0	0.0	1.429299	2.997850	Oceania	41.0
2	New Zealand	Campbell Islands	NZL	5478	POLYGON ((169.25042 -52.52026, 169.26044 -52.5...	0.0	0.0	0.0	252	0.0	0.0	0.0	13.443335	17.081100	Oceania	153.0
3	New Zealand	The Snares	NZL	5480	POLYGON ((166.63722 -47.99358, 166.63209 -48.0...	0.0	0.0	0.0	28	0.0	0.0	0.0	1.570404	17.081100	Oceania	153.0
4	New Zealand	Antipodes Islands	NZL	5477	POLYGON ((178.83904 -49.62916, 178.84392 -49.6...	0.0	0.0	0.0	143	0.0	0.0	0.0	7.894425	17.081100	Oceania	153.0

From here, we can determine that the Isles of Scilly are also one of the better areas to have an observatory located. Below is a map of the Isles of Scilly. While an interesting location, the Isles of Scilly are incredibly small and may hence not be the best for astronomical observatories.

In [44]:
```
m = folium.Map(location=[49.9339552,-6.3612226], zoom_start=11.495)
```

```
tile = folium.TileLayer(tiles='https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', attr='Esri', name='Esri Satellite', overlay=False, co
plugins.Fullscreen().add_to(m)
m
```

Out[44]:



To consider each value to an equal weightage, we apply the following index:

$$\lambda = rank\left(\lambda_{university}\right) + rank\left(\frac{1}{\lambda_{area}}\right) + rank\left(\lambda_{density}\right)$$

In [34]:

```
rankings.loc[:, "universityIndex"] = rankings.universityRank.rank()
rankings.loc[:, "invAreaRank"] = (1 / rankings.area).rank()
rankings.loc[:, "densityRank"] = rankings.Density.rank()
rankings.loc[:, "definitiveIndex"] = rankings[["universityIndex", "invAreaRank", "densityRank"]].sum(axis=1)
rankings = rankings.sort_values("definitiveIndex", ignore_index=True)
rankings.head()
```

Out[34]:

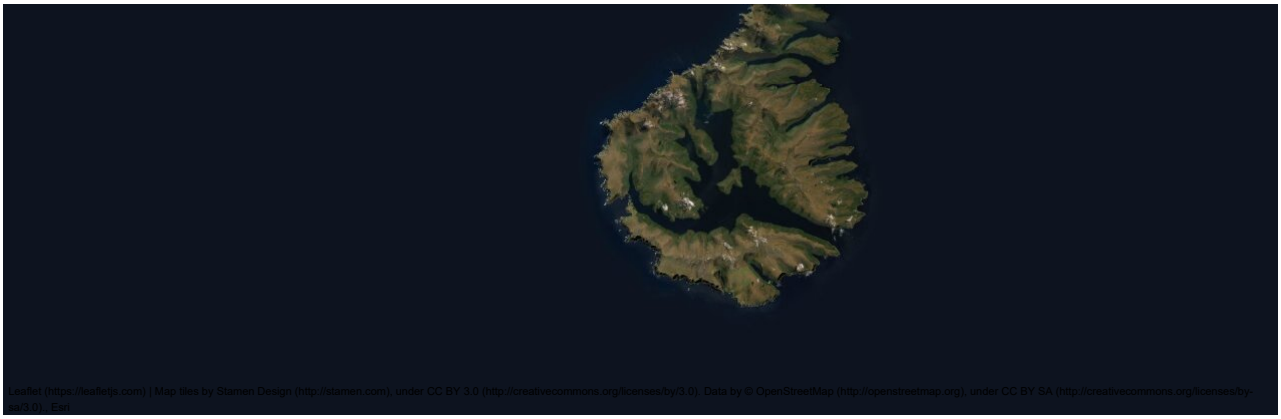
	country	name	ISO3166-1-Alpha-3	id	geometry	min	max	mean	count	std	median	mode	area	Density	continent	universityRank	universityIndex	invAreaRank	densityRank	definitiveIndex
0	New Zealand	Auckland Islands	NZL	5479	MULTIPOLYGON (((166.13697 -50.86435, 166.20525...	0.0	0.0	0.0	953	0.0	0.0	0.0	51.620572	17.08110	Oceania	153.0	6.5	17.0	6.5	30.0
1	New Zealand	Campbell Islands	NZL	5478	POLYGON ((169.25042 -52.52026, 169.26044 -52.5...	0.0	0.0	0.0	252	0.0	0.0	0.0	13.443335	17.08110	Oceania	153.0	6.5	29.0	6.5	42.0
2	New Zealand	Antipodes Islands	NZL	5477	POLYGON ((178.83904 -49.62916, 178.84392 -49.6...	0.0	0.0	0.0	143	0.0	0.0	0.0	7.894425	17.08110	Oceania	153.0	6.5	33.0	6.5	46.0
3	Australia	Norfolk Island	AUS	2659	POLYGON ((159.04444 -31.51922, 159.06886 -31.5...	0.0	0.0	0.0	19	0.0	0.0	0.0	1.429299	2.99785	Oceania	41.0	2.0	46.0	1.0	49.0
4	New Zealand	Kermadec Islands	NZL	5471	POLYGON ((-177.90954 -29.27858, -177.92488 -29...	0.0	0.0	0.0	55	0.0	0.0	0.0	3.932280	17.08110	Oceania	153.0	6.5	37.0	6.5	50.0

The next location is given to be the Auckland Islands, which are shown below. The Auckland Islands are shown to be quite close to Queenstown, New Zealand, and is surprisingly quite far away from Auckland itself. It is much more accessible compared to Norfolk Island, hence this may be the best location.

In [46]:

```
m = folium.Map(location=[-50.7092377,165.9796024], zoom_start=10, tiles="Stamen Terrain")
tile = folium.TileLayer(tiles='https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', attr='Esri', name='Esri Satellite', overlay=True, co
plugins.Fullscreen().add_to(m)
m
```

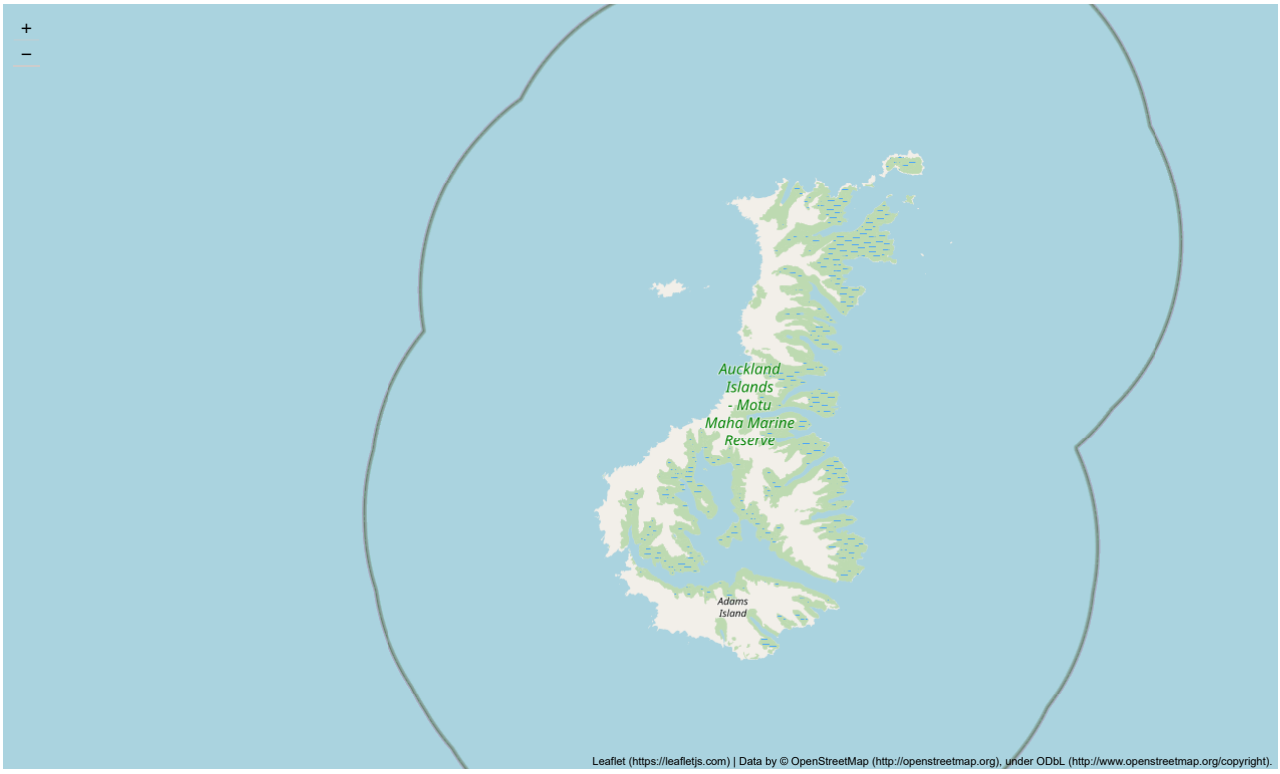
Out[46]:



In [47]:

```
m = folium.Map(location=[-50.7092377,165.9796024], zoom_start=10)
#tile = folium.TileLayer(tiles='https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', attr='Esri', name='Esri Satellite', overlay=True, co
plugins.Fullscreen().add_to(m)
m
```

Out[47]:



Hence, we conclude that the following are the best locations for astronomical observation (in order of suitability):

1. The Auckland Islands, New Zealand
2. Norfolk Island, Australia
3. Tibesti Mountain Region, Chad
4. THe Isles of Scilly, UK

Based on Countries

In some cases, astronomers would prefer to be situated in one specific country rather than move around a lot.

According to the 2010 US National Academies Decadal Survey of Astronomy, having to move so frequently, which is a career necessity for astronomers and astrophysicists, with many, as postdocs, relocating every two or three years until such time that they secure a permanent position, may be unattractive to people looking to start a family, especially impacting women.

In fact, according to a recent paper released in January 2021, which brought to a light a project known as ASTROMOVES that studies the diversity and mobility of astrophysicists today, most postdocs have a fixed salary despite the constant location changes, which may not be able to cover for the cost for relocating with a family, hence here we investigate the best countries for astronomical observation, specifically by astronomers and astrophysicists.

In [35]:

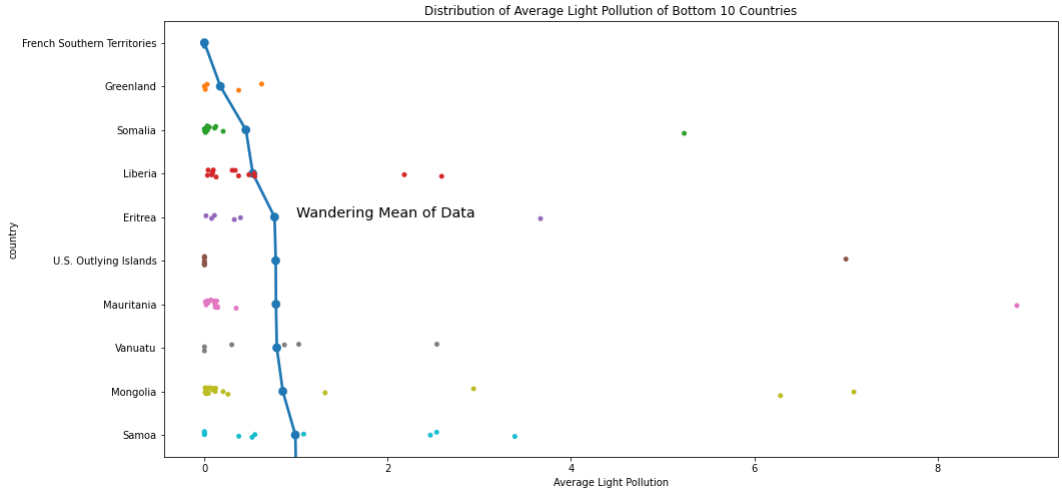
```
nightLight2013 = nightLight2013.dropna(subset=["id"])
rankedByMean = nightLight2013[nightLight2013.country.isin((lambda counts: counts[counts > 1].index)(nightLight2013.country.value_counts()))].groupby("country")[["mean"]].mean().so
rankedByMean
```

Out[35]:

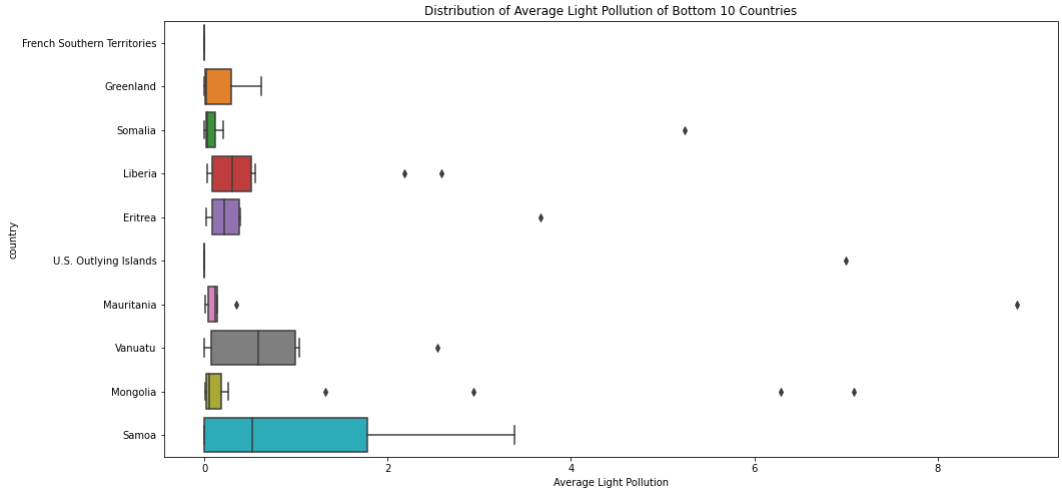
country	mean
French Southern Territories	0.000000
Greenland	0.173593
Somalia	0.452518
Liberia	0.527399
Eritrea	0.765346
...	...
San Marino	47.221897
Malta	53.549064
Hong Kong	55.235341
Bahrain	59.446822
Singapore	62.435109

211 rows × 1 columns

In [36]:
```
bottom10Countries = nightLight2013.set_index("country").loc[rankedByMean.iloc[:10].index].reset_index().rename(columns={"mean":"Average Light Pollution"})
meanBottom10Countries = rankedByMean.iloc[:11]
kwargs = dict(data=bottom10Countries, y="country", x="Average Light Pollution")
#sns.boxplot(**kwargs).set_title("Box Plot of Average Light Pollution of Top 10 Countries")
#sns.violinplot(**kwargs)
ax = sns.pointplot(data=meanBottom10Countries.reset_index(), y="country", x="mean", label="Wandering Mean")
sns.stripplot(data=bottom10Countries, y="country", x="Average Light Pollution", ax=ax, label="Average Light Pollution").set_title("Distribution of Average Light Pollution of Botto
_ = ax.text(1, 4, "Wandering Mean of Data", fontfamily="sans-serif", fontsize="x-large")
```



In [37]:
```
_ = sns.boxplot(data=bottom10Countries, y="country", x="Average Light Pollution").set_title("Distribution of Average Light Pollution of Bottom 10 Countries")
```



The countries Somalia and Mauritania are seemingly the least polluted outside of the French Southern Territories, although there are some outliers in both, especially in Mauritania. However, the left-skewed nature of the data is incredibly indicative of the number of problematic outliers.

Research Question B: How has the light pollution data around the world changed? Which countries are most susceptible to high light pollution in the future? Which countries are lessening in terms of light pollution?

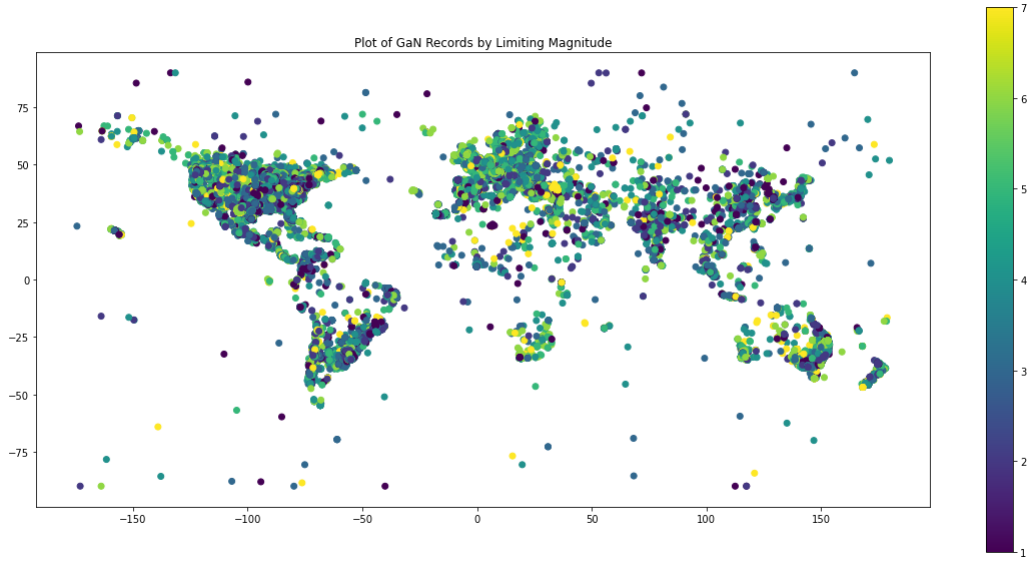
There are many countries that have had rampant increases in Light Pollution over the past few years, while others have made an effort to reduce their already increased Light Pollution. We need to map the data in order to find out which countries are susceptible to the level of problematic light pollution that can be found in cities like Singapore.

Due to the possession of the Time Series Data of night-time light pollution data, we can easily map locations based on relative changes in absolute light magnitude. I intend to use the geolocation datasets against the time-series night-time light datasets to find changes and from there and map it using a possible regression model so as to predict the curve of increase/decrease. This can be used to make predictions and compare to other light pollution levels.

Firstly, we find the average light pollution of each country in each year, and then plot a Time Series-based Choropleth Map using the Plotly Express Library. From here, we can get a quick grasp of how the Average Light Pollution of each country changes.

We use this on both the GaN and NightLight Datasets.

In [51]:
```
_ = gan.plot(figsize=(20, 10), column="LimitingMag", legend=True).set_title("Plot of GaN Records by Limiting Magnitude")
```



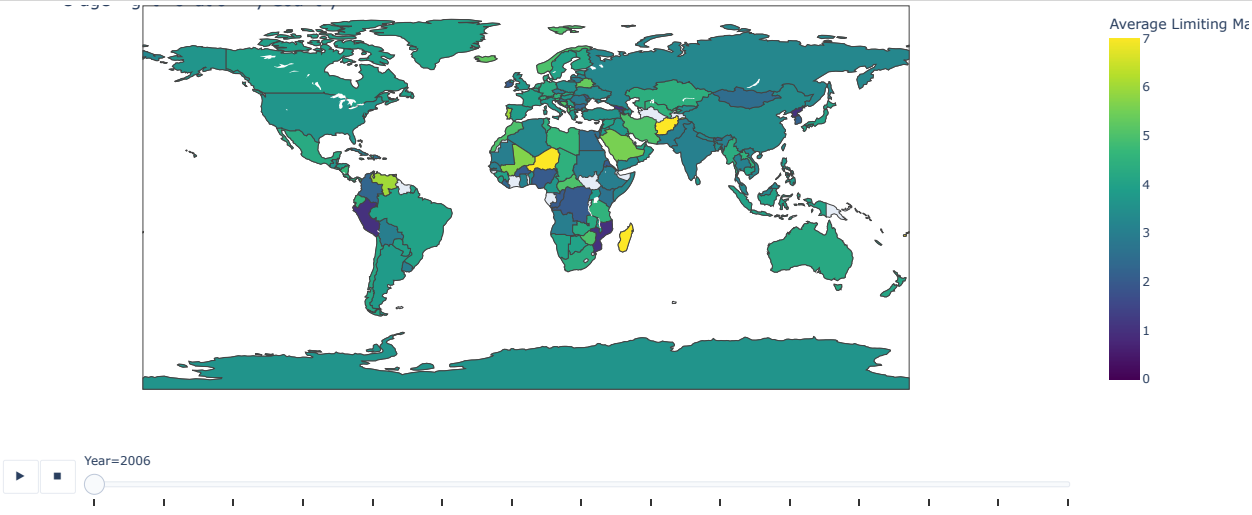
In [38]:
```
ganCountry = gan.groupby(["Country", "Year"]).LimitingMag.mean().reset_index().rename(columns={0: "LimitingMag"}).pivot("Year", "Country", "LimitingMag").apply(lambda col: col.fil
ganCountry
```

Out[38]:

	Year	Country	Average Limiting Magnitude
0	2006	Afghanistan	7.000000
1	2006	Albania	5.000000
2	2006	Algeria	3.272727
3	2006	Andorra	2.000000
4	2006	Angola	3.000000
...	...	...	...
2725	2020	Venezuela	3.333333
2726	2020	Vietnam	2.694872
2727	2020	Yemen	4.111111
2728	2020	Zambia	4.250000
2729	2020	Zimbabwe	3.666667

2730 rows × 3 columns

In [162]:
```
px.choropleth(ganCountry,
              locations = "Country",
              color = "Average Limiting Magnitude",
              animation_frame = "Year",
              color_continuous_scale="Viridis",
              locationmode="country names",
              range_color=(0, 7),
              title="Average Light Pollution By Country"
             ).update_layout(margin={"r":0,"t":0,"l":0,"b":0})
```



In [39]:
```
nightLightMean = nightLight.reset_index().set_index(["Quantity", "Year"]).loc["mean"].stack().reset_index().rename(columns={"level_1": "Country", 0: "Average Light Pollution"})
nightLightMean
```

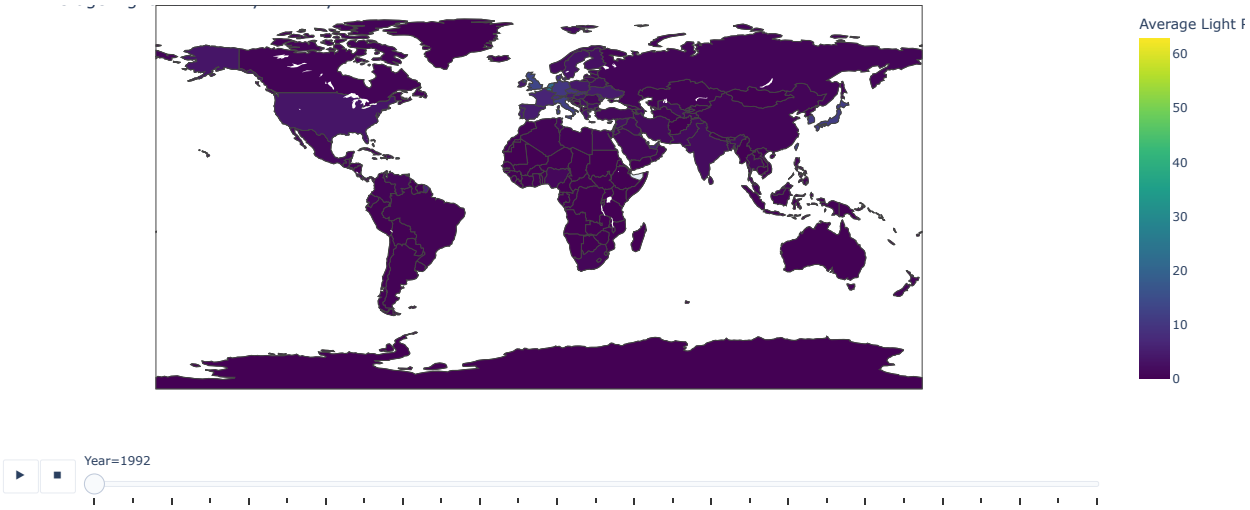
Out[39]:

	Year	Country	Average Light Pollution
0	1992	Afghanistan	0.027943
1	1992	Akrotiri Sovereign Base Area	14.912409
2	1992	Aland	2.225571
3	1992	Albania	0.264269
4	1992	Algeria	0.506192
...	...	...	...
6880	2018	Wallis and Futuna	5.855422
6881	2018	Western Sahara	0.053453

	Year	Country	Average Light Pollution
6882	2018	Yemen	1.301654
6883	2018	Zambia	3.077475
6884	2018	Zimbabwe	2.009717

6885 rows × 3 columns

In [40]:
```
px.choropleth(nightLightMean,
              locations = "Country", color = "Average Light Pollution", animation_frame = "Year", color_continuous_scale="Viridis",
              locationmode="country names", range_color=(0, nightLightMean["Average Light Pollution"].max()), title="Average Light Pollution By Country",
).update_layout(margin={"r":0,"t":0,"l":0,"b":0})
```



How has the light pollution data around the world changed?

Here, we go through light pollution in general, using two different algorithms as follows:

$$\mu_1 = \frac{1}{27}\sum_{y=1992}^{2018} \mu_y$$

$$\mu_2 = \frac{\sum_{y=1992}^{2018} n_y \mu_y}{\sum_{y=1992}^{2018} n_y}$$

We plot graphs based on both of these graphs.

In [41]:
```
pivotNightLight = nightLightMean.pivot("Country", "Year", "Average Light Pollution").sort_values(2018).rename(columns="nightLight{}".format)
pivotNightLight
```

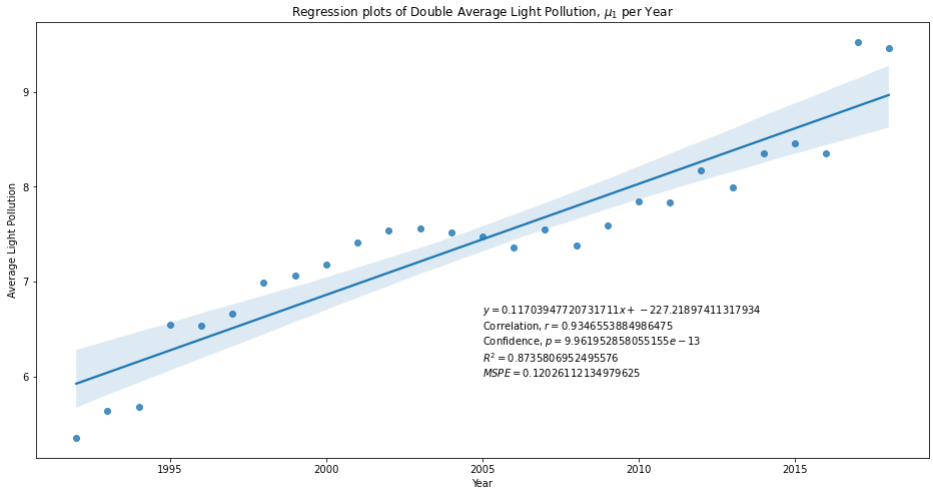
Out[41]:

Year	nightLight1992	nightLight1993	nightLight1994	nightLight1995	nightLight1996	nightLight1997	nightLight1998	nightLight1999	nightLight2000	nightLight2001	...	nightLight2009	nightLight2010	n
Country														
Clipperton Island	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	
Vatican City	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	
Coral Sea Islands	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	
Bajo Nuevo Bank (Petrel Is.)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	
Pitcairn Islands	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Gibraltar	38.200000	42.800000	41.000000	44.200000	46.600000	49.200000	45.600000	40.800000	39.400000	39.800000	...	54.250000	46.250000	
Bahrain	42.001305	42.835509	42.883812	45.193211	45.382507	45.016971	42.761097	43.122715	44.013055	44.310705	...	48.760582	52.612434	
Monaco	54.466667	53.833333	57.566667	59.433333	53.500000	59.100000	58.733333	58.000000	57.800000	58.466667	...	55.300000	51.866667	
Singapore	61.957983	61.228571	61.643697	62.297479	62.228571	61.470588	61.425210	61.668908	61.712605	61.845378	...	58.565217	58.886288	
Macao	49.500000	47.411765	49.117647	51.029412	53.470588	59.058824	54.088235	56.647059	55.117647	57.529412	...	56.897436	55.461538	

255 rows × 27 columns

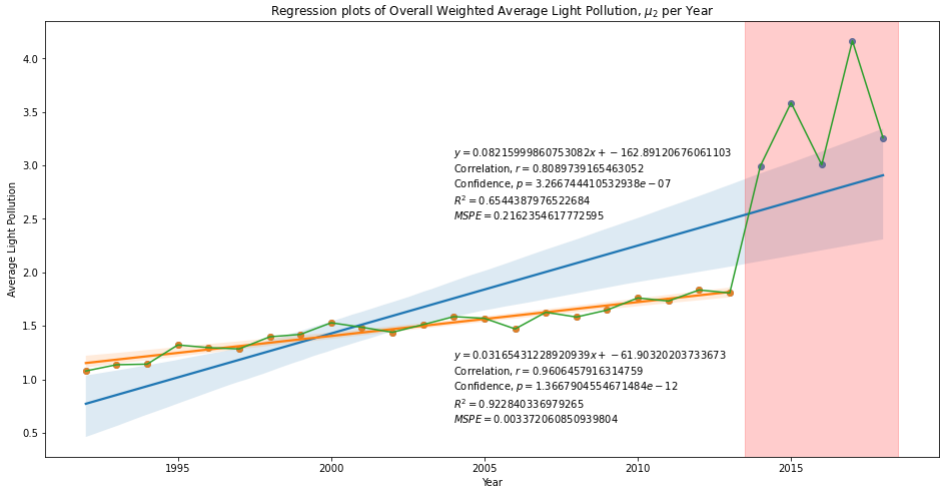
In [42]:
```
def summary(data, xloc, yloc):
    x, y = data.Year, data["Average Light Pollution"]
    m, c, r, p, stderr = stats.linregress(x=x, y=y)
    mspe = mean_squared_error(y, m*x + c)
    sns.regplot(x=x, y=y)
    plt.text(xloc, yloc, f"$y = {m} x + {c}$\nCorrelation, $r = {r}$\nConfidence, $p = {p}$\n$R^2 = {r**2}$\n$MSPE = {mspe}$")

yr_based = pivotNightLight.rename(columns=lambda yrstr: int(yrstr[-4:])).mean(axis=0).reset_index().rename(columns={0: "Average Light Pollution"})
summary(data=yr_based, xloc=2005, yloc=6)
plt.title("Regression plots of Double Average Light Pollution, $\mu_1$ per Year")
```

Out[42]:
```
Text(0.5, 1.0, 'Regression plots of Double Average Light Pollution, $\\mu_1$ per Year')
```



In [43]:
```
nightLightByQuan = nightLight[nightLight.Quantity.isin(["mean", "count"])].reset_index().set_index(["Quantity", "Year"])
fitted_mean_by_yr = ((nightLightByQuan.loc["mean"] * nightLightByQuan.loc["count"]).sum(axis=1) / nightLightByQuan.loc["count"].sum(axis=1)).reset_index().rename(columns={0:"Averag
summary(fitted_mean_by_yr, 2004, 2.5)
summary(fitted_mean_by_yr[fitted_mean_by_yr.Year.isin(range(1992, 2014))], 2004, 0.6)
sns.lineplot(data=fitted_mean_by_yr, x="Year", y="Average Light Pollution").axvspan(xmin=2013.5, xmax=2018.5, color="r", alpha=0.2)
plt.title("Regression plots of Overall Weighted Average Light Pollution, $\\mu_2$ per Year")
```

Out[43]:
```
Text(0.5, 1.0, 'Regression plots of Overall Weighted Average Light Pollution, $\\mu_2$ per Year')
```



It is clear that the original prediction was altered by the presence of outliers from 2014 onwards, and the second prediction, which has a much higher R^2 value and much lower $MSPE$, is a more accurate look at the still increasing light pollution over the years. However, after 2014, the value suddenly nearly doubles, and continutes on a pretty uneven trajectory.

Which countries have been most and least susceptible to Light Pollution?

To indentify susceptibility, we consider the following scenarios:

1. The largest range of values in average light pollution intensity
2. The greatest difference in values in 1992 and 2018 (positive and negative separately)
3. Countries with the highest and lowest average light pollution over the years

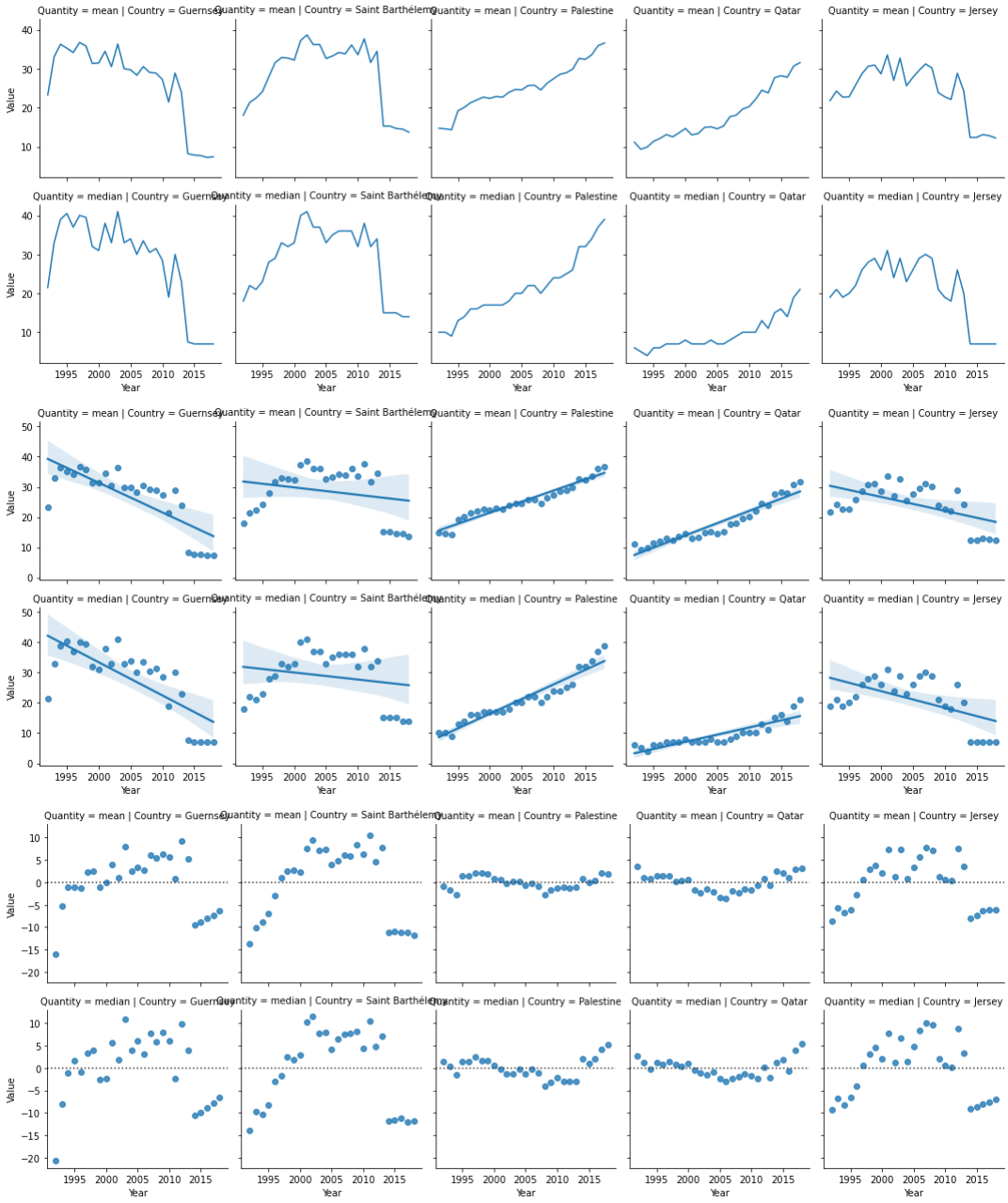
In [44]:
```
def predict(index):
    print(index)
    df = nightLight.reset_index().set_index(["Quantity", "Year"])[index.index].loc[["mean", "median"]].stack().reset_index().rename(columns={"level_2":"Country", 0: "Value"})
    for alg in [sns.lineplot, sns.regplot, sns.residplot]:
        sns.FacetGrid(df, row="Quantity", col="Country").map(alg, "Year", "Value")
```

Countries with the largest range of values in Average Light Pollution Intensity

To calculate this, we simply compute the max and min over the years and find the largest differences over the list of countries. The countries found are Guernsey, Saint Barthelemy, Palestine, Qatar and Jersey. Based on the results below, Palestine and Qatar have seen a clear increase in valus over the past few years, and their predictions are also the most accurate, according to the residual plot shown below. Guernsey, on the other hand, has seen a huge decrease in light pollution over the past few years.

In [45]:
```
nightLightHighLow = nightLight.reset_index().set_index(["Quantity", "Year"]).loc["mean"].T.stack().reset_index().rename(columns={"level_0": "Country", 0: "Value"}).groupby("Country
nightLightHighLow = (nightLightHighLow["max"] - nightLightHighLow["min"]).sort_values(ascending=False).iloc[:5]
predict(nightLightHighLow)
```

```
Country
Guernsey            29.507463
Saint Barthélemy    25.000000
Palestine           22.280746
Qatar               22.256849
Jersey              21.333333
dtype: float64
```



Countries with the greatest difference between values in 1992 and 2018.

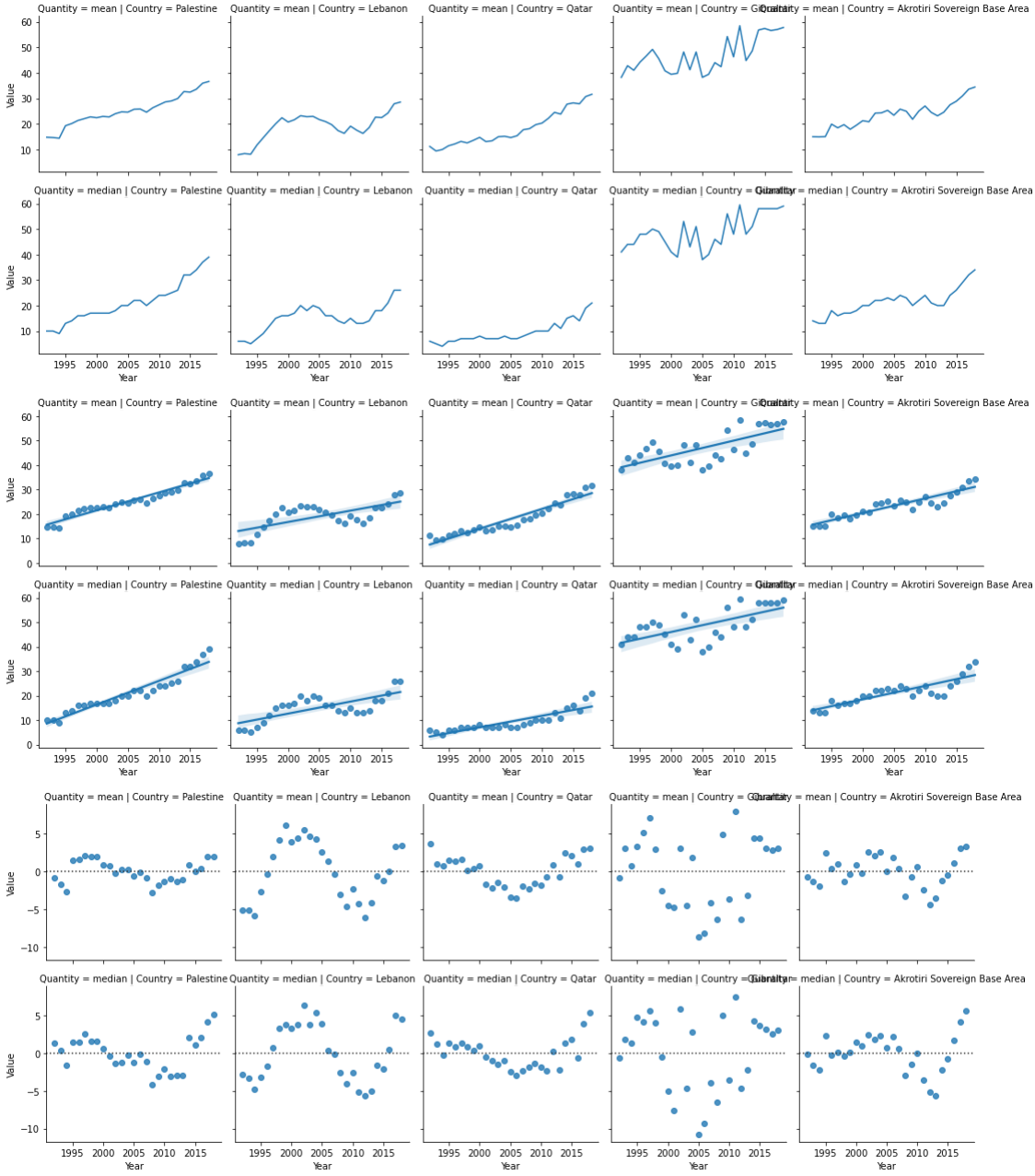
To calculate this, we simply took the difference, the took the top 5 positive and top 5 negative results. The top 5 most positive countries found are Palestine, Lebanon, Qatar, Gibraltar and Akrotiri, which are all increasing rapidly. The top 5 most negative countries are Saint Barthelemey, US Virgin Islands, Bermuda, Jersey and Guernsey. Based on the results, Bermuda, Jersey and Guernsey have largely decreasing results, while the US Virigin Islands are also in that situation but to a limited extent.

In [46]:
```
nightLightNet = nightLight.reset_index().set_index(["Quantity", "Year"]).loc["mean"].T[[1992, 2018]] #.T.stack().reset_index().rename(columns={"level_0": "Country", 0: "Value"})
nightLightNet = (nightLightNet[2018] - nightLightNet[1992]).sort_values(ascending=False)
nightLightNet
```

Out[46]:
```
Palestine                      21.896256
Lebanon                        20.643396
Qatar                          20.468919
Gibraltar                      19.600000
Akrotiri Sovereign Base Area   19.496350
                                  ...
Saint Barthélemy               -4.354839
U.S. Virgin Islands            -6.744186
Bermuda                        -7.445652
Jersey                         -9.600939
Guernsey                      -15.895522
Length: 255, dtype: float64
```

In [47]:
```
predict(nightLightNet.iloc[:5])
```

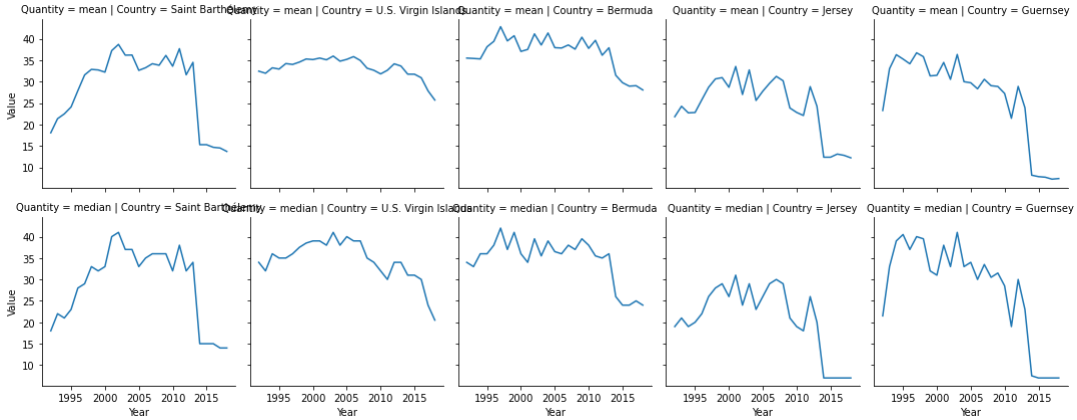
```
Palestine                      21.896256
Lebanon                        20.643396
Qatar                          20.468919
Gibraltar                      19.600000
Akrotiri Sovereign Base Area   19.496350
dtype: float64
```



In [48]:

```
predict(nightLightNet.iloc[-5:])
```

```
Saint Barthélemy       -4.354839
U.S. Virgin Islands    -6.744186
Bermuda                -7.445652
Jersey                 -9.600939
Guernsey              -15.895522
dtype: float64
```

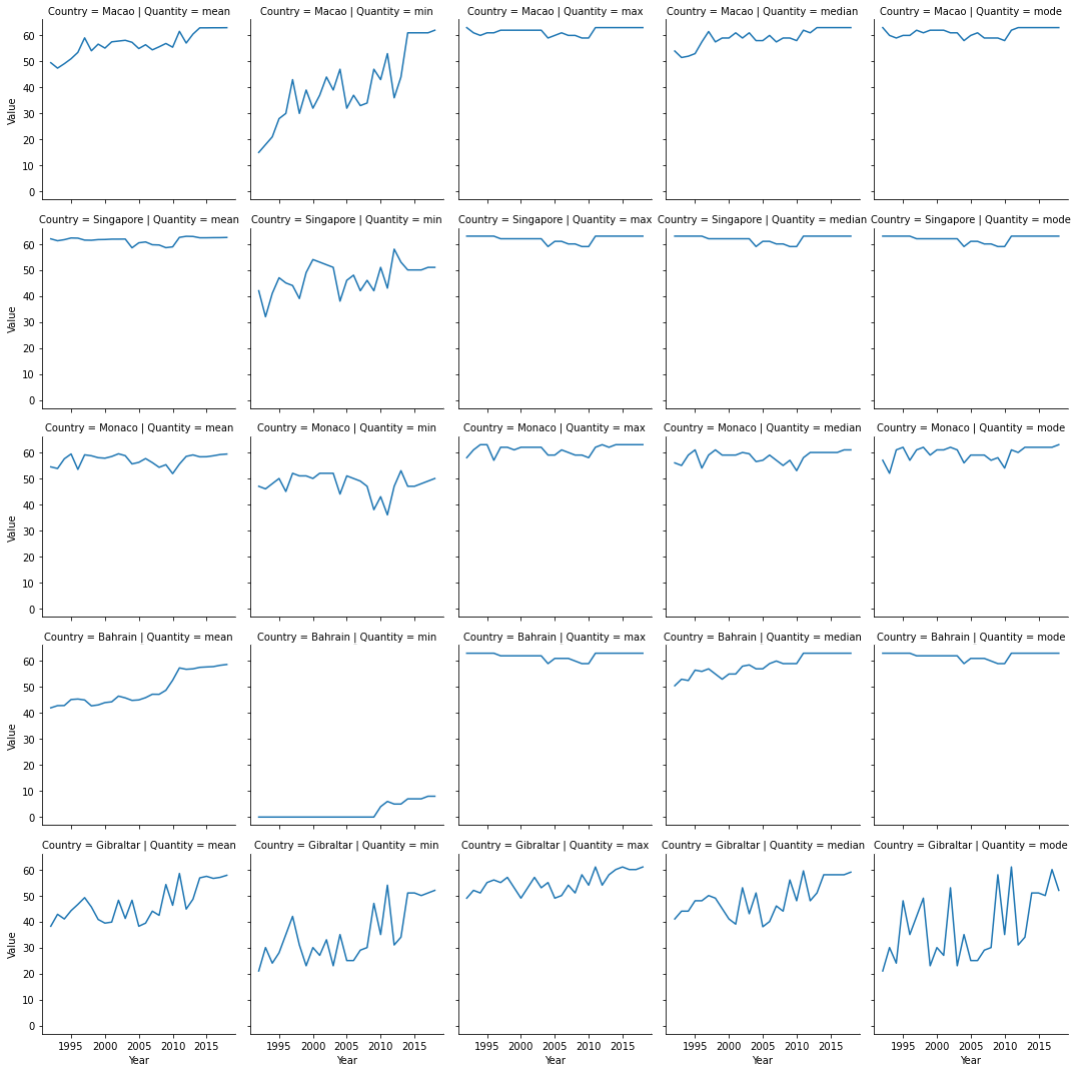


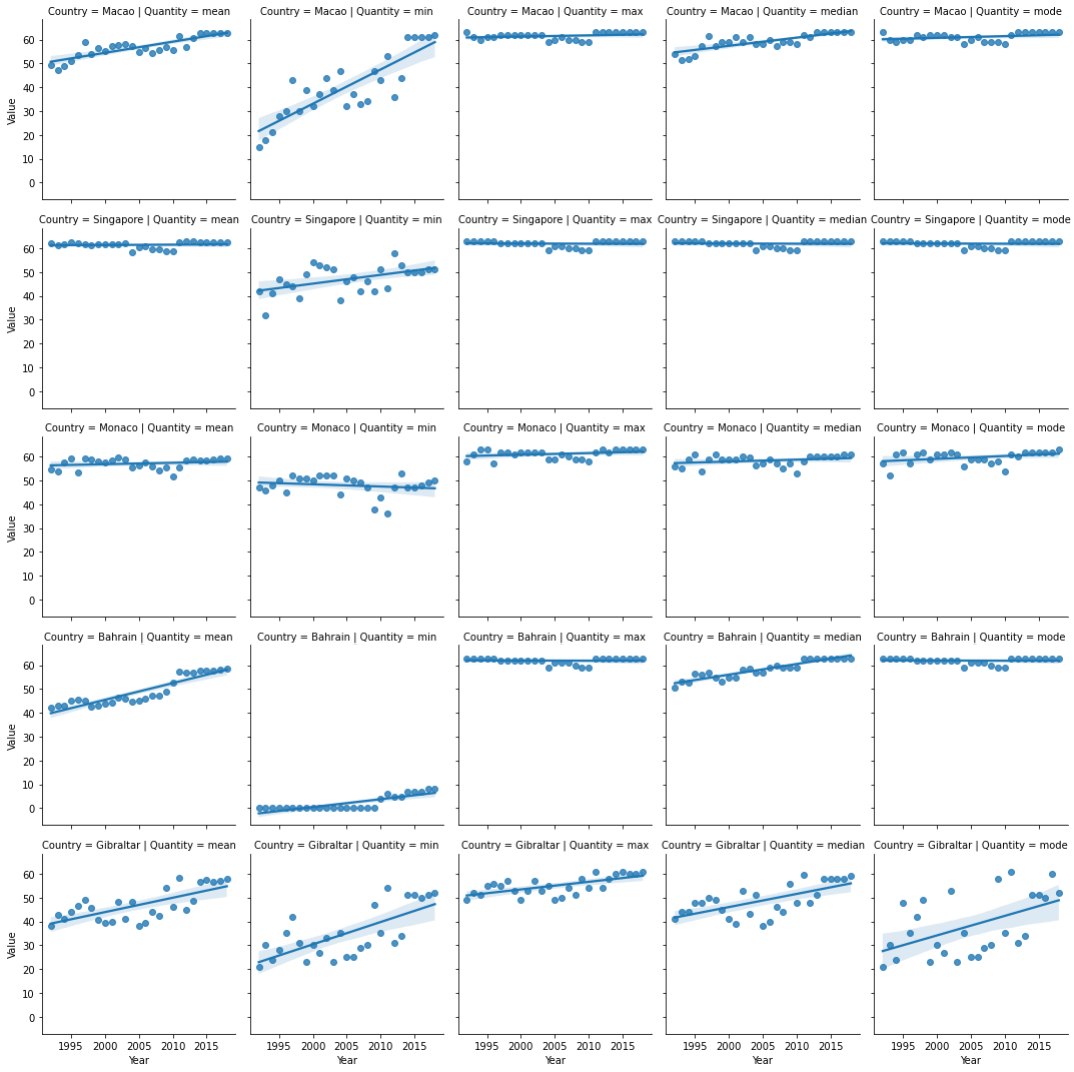
Countries with the largest amount of Light Pollution

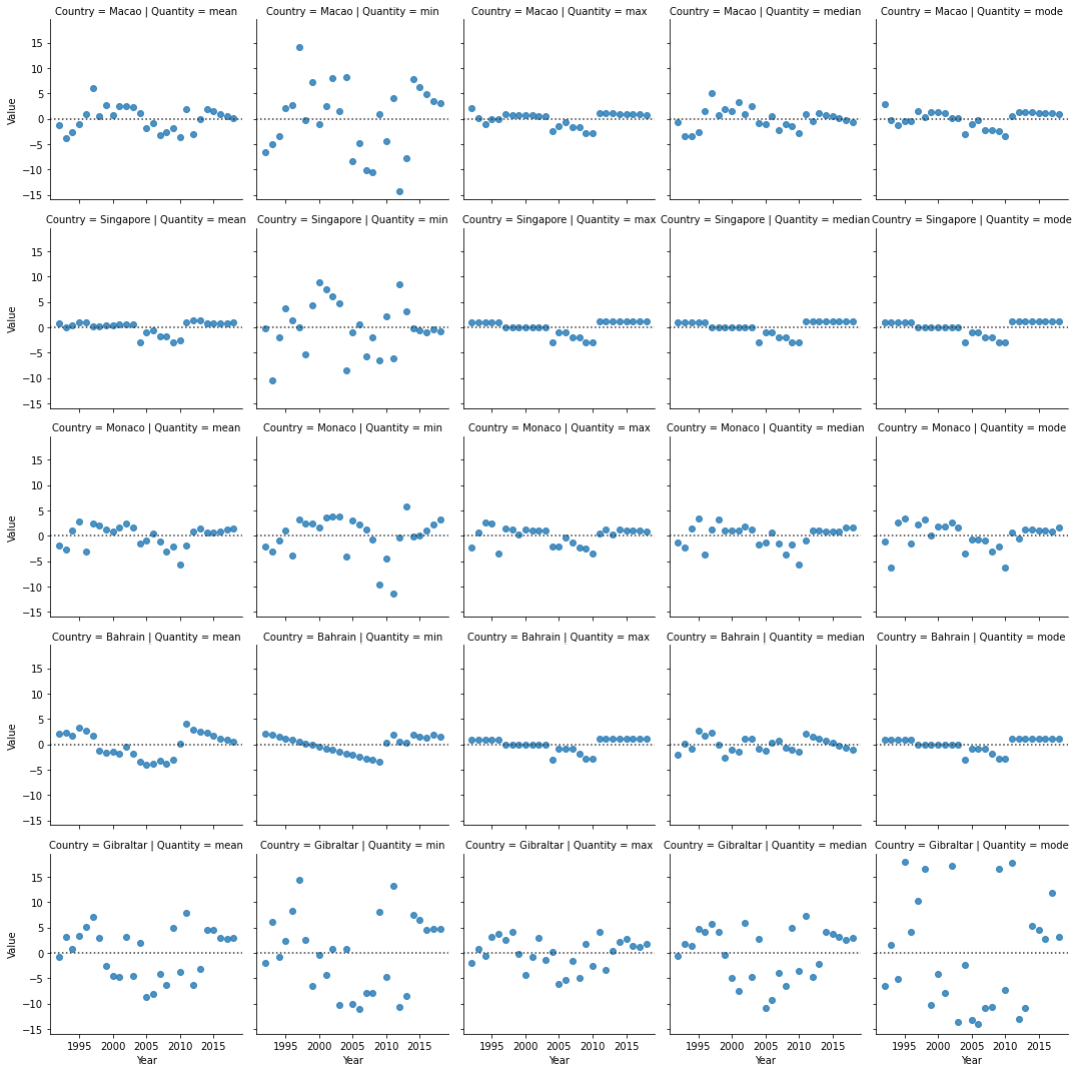
The countries in consideration are Macao, Singapore, Monaco, Bahrain and Gibraltar. Based on our results, Macao, Gibraltar and Singapore's minimum values are increasing, while Monaco's have decreased. As for mean, all but Singapore have had incresing light pollution values, whereas Singapore's have remained stagnated.

In [49]:

```
def nightLightFilter(slice):
    return nightLight.reset_index().set_index(["Year", "Quantity"]).T.sort_values((2018, "mean"), ascending=False).iloc[slice].T.stack().reset_index().set_index(["Quantity", "Year"

nightLightMax = nightLightFilter(slice(0, 5))
for alg in [sns.lineplot, sns.regplot, sns.residplot]:
    sns.FacetGrid(nightLightMax, col="Quantity", row="Country").map(alg, "Year", "Value")
```

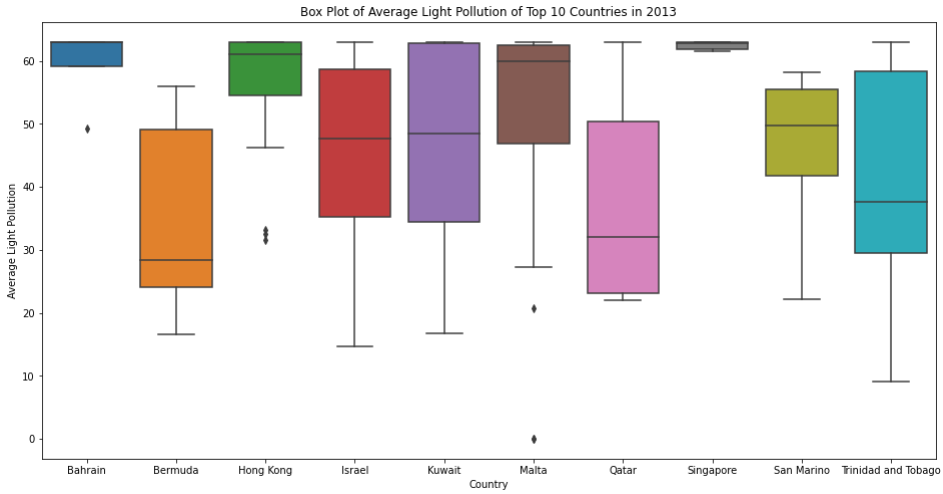

Country = Macao | Quantity = mean
Country = Macao | Quantity = min
Country = Macao | Quantity = max
Country = Macao | Quantity = median
Country = Macao | Quantity = mode
Country = Singapore | Quantity = mean
Country = Singapore | Quantity = min
Country = Singapore | Quantity = max
Country = Singapore | Quantity = median
Country = Singapore | Quantity = mode
Country = Monaco | Quantity = mean
Country = Monaco | Quantity = min
Country = Monaco | Quantity = max
Country = Monaco | Quantity = median
Country = Monaco | Quantity = mode
Country = Bahrain | Quantity = mean
Country = Bahrain | Quantity = min
Country = Bahrain | Quantity = max
Country = Bahrain | Quantity = median
Country = Bahrain | Quantity = mode
Country = Gibraltar | Quantity = mean
Country = Gibraltar | Quantity = min
Country = Gibraltar | Quantity = max
Country = Gibraltar | Quantity = median
Country = Gibraltar | Quantity = mode
Value
Year
0
10
20
30
40
50
60
1995
2000
2005
2010
2015


Country = Macao | Quantity = mean
Country = Macao | Quantity = min
Country = Macao | Quantity = max
Country = Macao | Quantity = median
Country = Macao | Quantity = mode
Country = Singapore | Quantity = mean
Country = Singapore | Quantity = min
Country = Singapore | Quantity = max
Country = Singapore | Quantity = median
Country = Singapore | Quantity = mode
Country = Monaco | Quantity = mean
Country = Monaco | Quantity = min
Country = Monaco | Quantity = max
Country = Monaco | Quantity = median
Country = Monaco | Quantity = mode
Country = Bahrain | Quantity = mean
Country = Bahrain | Quantity = min
Country = Bahrain | Quantity = max
Country = Bahrain | Quantity = median
Country = Bahrain | Quantity = mode
Country = Gibraltar | Quantity = mean
Country = Gibraltar | Quantity = min
Country = Gibraltar | Quantity = max
Country = Gibraltar | Quantity = median
Country = Gibraltar | Quantity = mode
Value
Year
0
10
20
30
40
50
60
1995
2000
2005
2010
2015

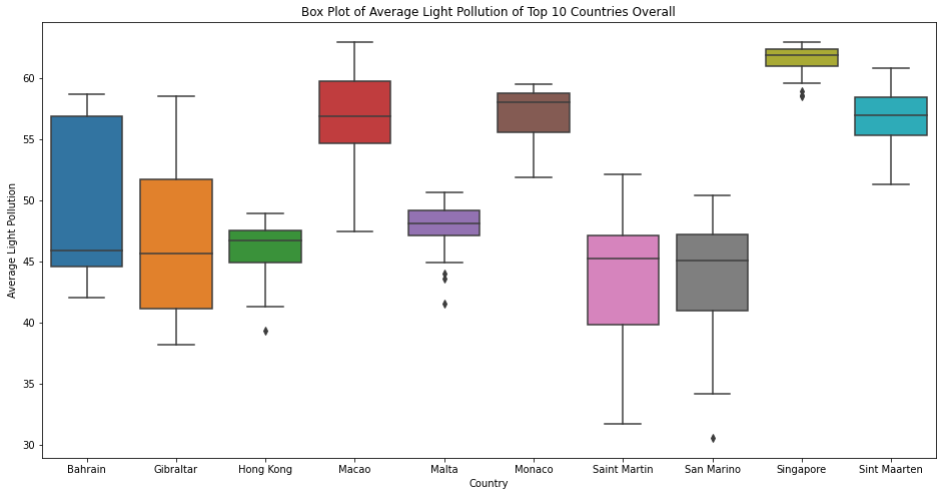


We now investigate general distribution in light pollutions for countries with the highest average light pollutions. This is done in the form of box plots and violin plots that give insight into the distribution of values in 2013 and overall. Singapore is clearly shown to be skewed towards the highest possible values, which Bahrain and Gibraltar are bimodal distributions. The values for Hong Kong, Malta, Sint Maarten, San Marino, Kuwait and Singapore are skewed towards the right, while other countries have data skewed to the left. Hong Kong, Malta and Bahrain possess a few outliers less that the median. Israel, Trinidad+Tobago and Kuwait have quite large ranges of values as well as large IQRs. The centeral tendency of many counries seems to be on the lower end, besides Singapore, Hong Kong and Malta.

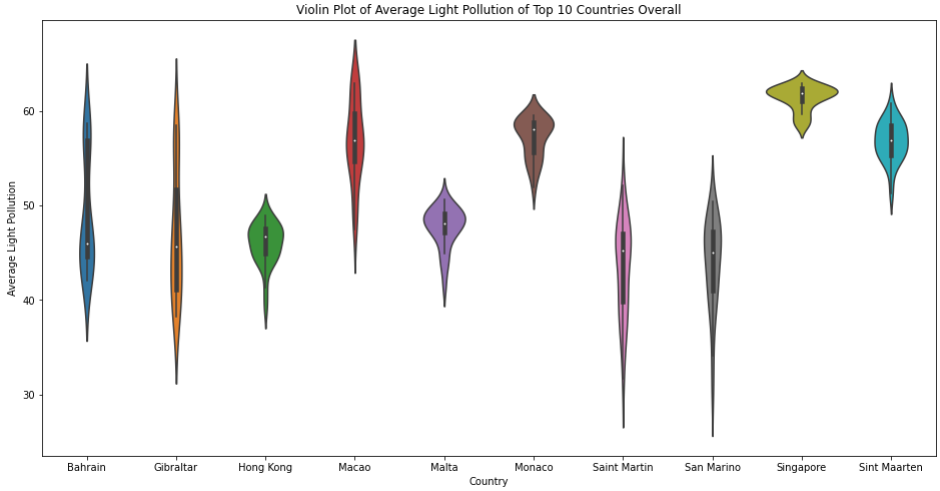
In [50]:
```
non_empty = nightLight2013[nightLight2013.country.isin((lambda counts: counts[counts > 1].index)(nightLight2013.country.value_counts()))].rename(columns={"country":"Country"})
_ = sns.boxplot(data=non_empty[non_empty.Country.isin(non_empty.groupby("Country")["mean"].mean().sort_values(ascending=False).iloc[:10].index)].rename(columns={"mean":"Average Lig
```



In [51]:
```
largestMeanLightByCountry = nightLightMean[nightLightMean.Country.isin(list(nightLightMean.groupby("Country")["Average Light Pollution"].mean().sort_values(ascending=False)[:10].ir
kwargs = dict(data=largestMeanLightByCountry, x="Country", y="Average Light Pollution")
_ = sns.boxplot(**kwargs).set_title("Box Plot of Average Light Pollution of Top 10 Countries Overall")
```

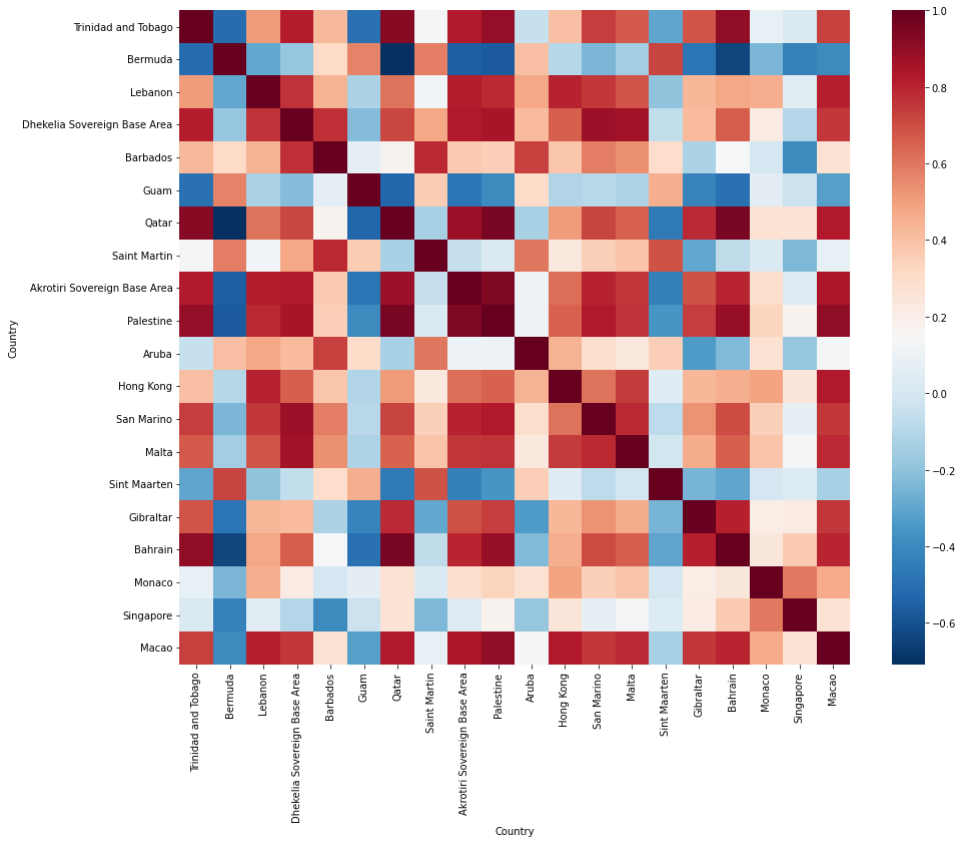


In [52]:
```
_ = sns.violinplot(**kwargs).set_title("Violin Plot of Average Light Pollution of Top 10 Countries Overall")
```



In [53]:
```
fig, ax = plt.subplots(1, figsize=(15, 12))
sns.heatmap(pivotNightLight.T.corr().dropna(how="all", axis=0).dropna(how="all", axis=1).iloc[-20:, -20:], cmap="RdBu_r", ax=ax)
```

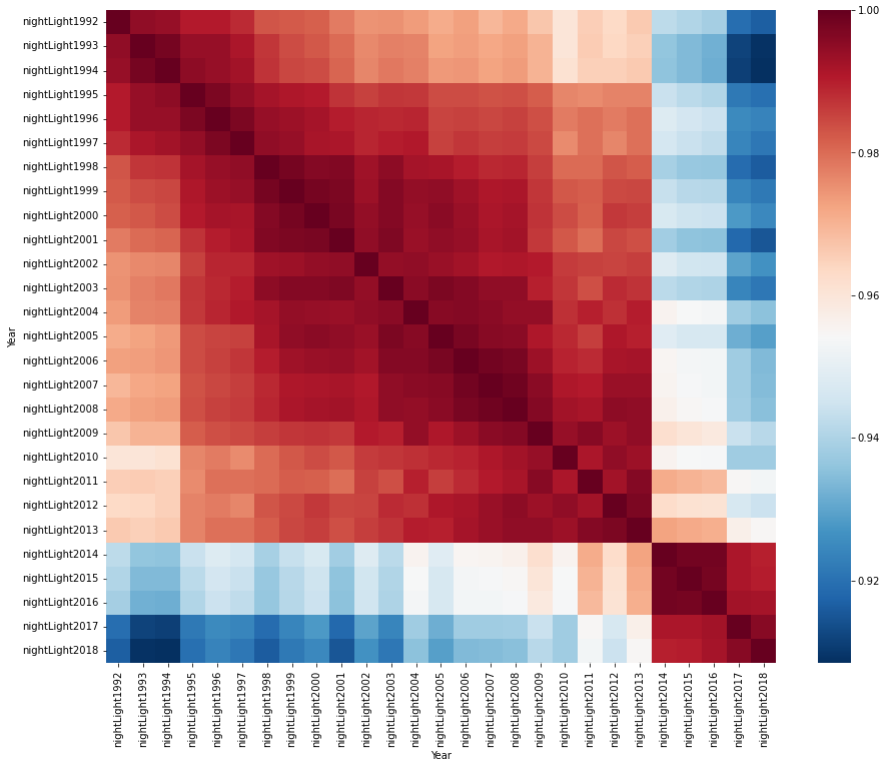
Out[53]:
```
<AxesSubplot:xlabel='Country', ylabel='Country'>
```



This shows that Bermuda is very negatively correlated to Qatar and Bahrain, which indicate that these are rapidly differing quantities.

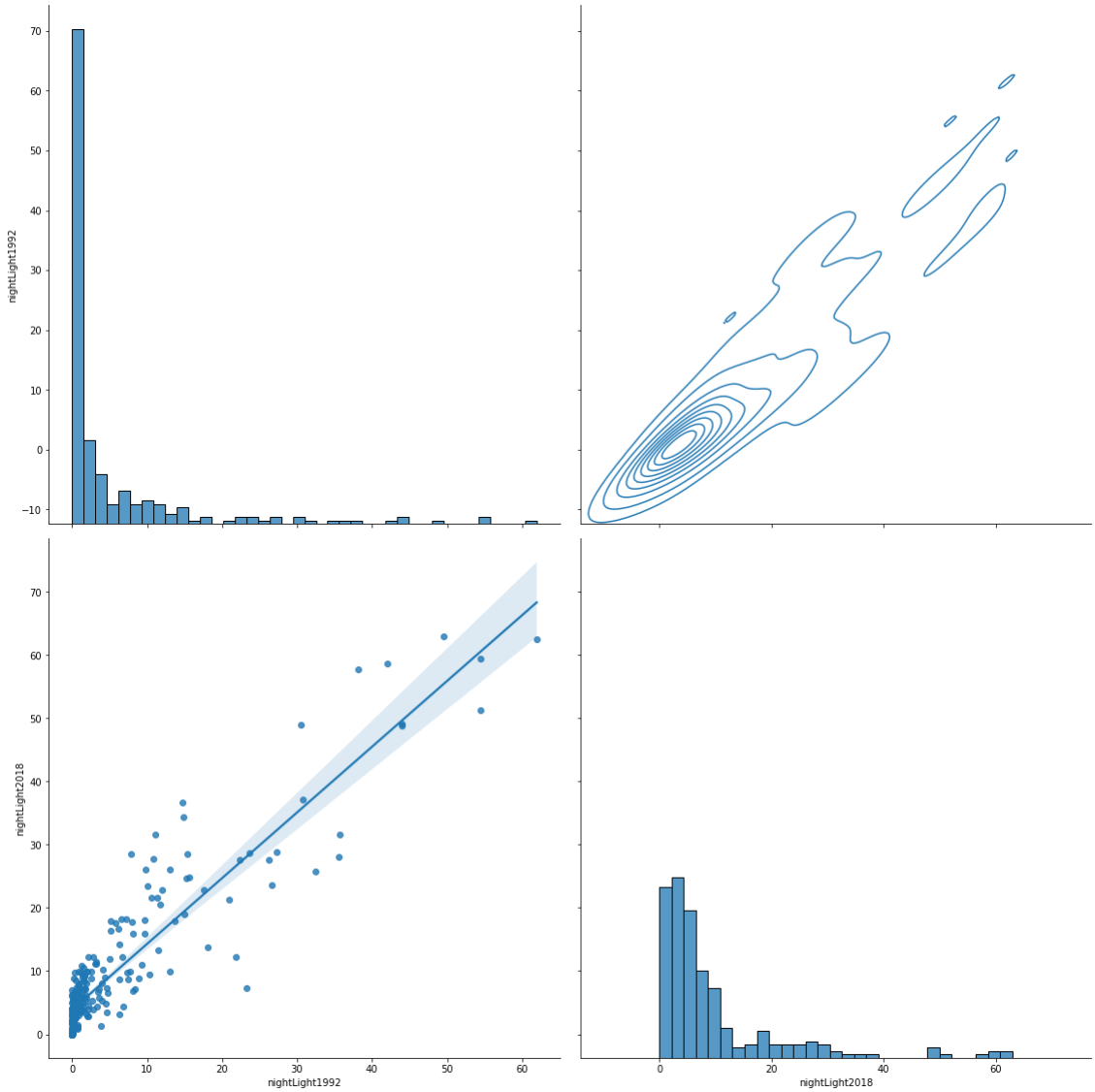
In [54]:
```
fig, ax = plt.subplots(1, figsize=(15, 12))
sns.heatmap(pivotNightLight.corr().dropna(how="all", axis=0).dropna(how="all", axis=1), cmap="RdBu_r", ax=ax)
```

Out[54]: <AxesSubplot:xlabel='Year', ylabel='Year'>



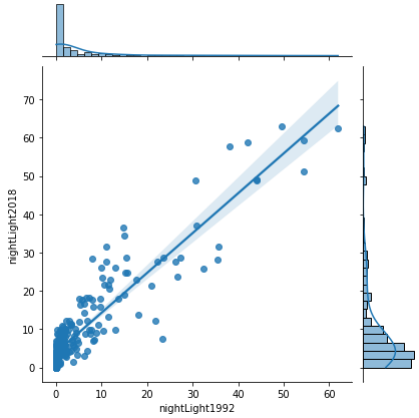
In [55]:
```
sns.PairGrid(pivotNightLight[["nightLight1992", "nightLight2018"]], height=8).map_diag(sns.histplot).map_lower(sns.regplot).map_upper(sns.kdeplot)
```

Out[55]: <seaborn.axisgrid.PairGrid at 0x1e4ae0522e0>



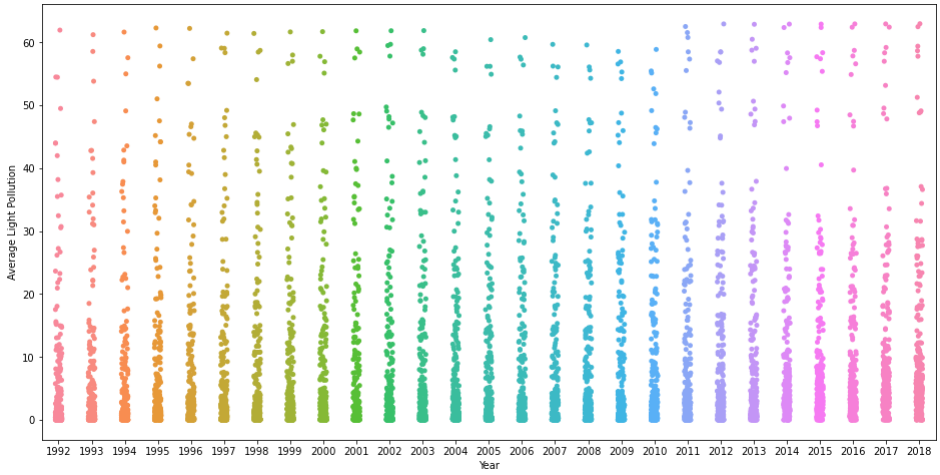
In [56]:
```
sns.jointplot(data=pivotNightLight[["nightLight1992", "nightLight2018"]], x="nightLight1992", y="nightLight2018", kind="reg")
```

Out[56]: <seaborn.axisgrid.JointGrid at 0x1e4a7d3bc40>



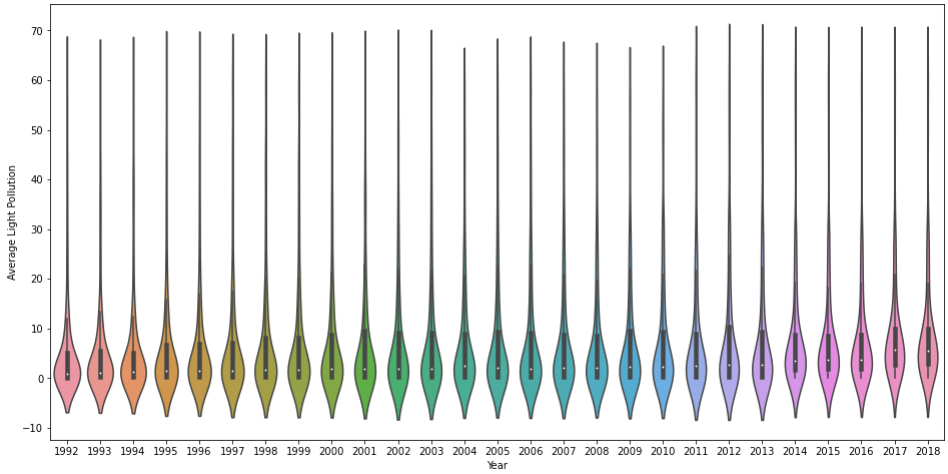
In [57]:
```
plt.figure(figsize=(16, 8))
sns.stripplot(x="Year", y="Average Light Pollution", data=nightLightMean)
```

Out[57]: <AxesSubplot:xlabel='Year', ylabel='Average Light Pollution'>



In [58]:
```
plt.figure(figsize=(16, 8))
sns.violinplot(x="Year", y="Average Light Pollution", data=nightLightMean)
```

Out[58]: <AxesSubplot:xlabel='Year', ylabel='Average Light Pollution'>



This shows that all the values are very well correlated to one another, indicating that values have not changed too much for many countries over the past 27 years, with a few exceptions.

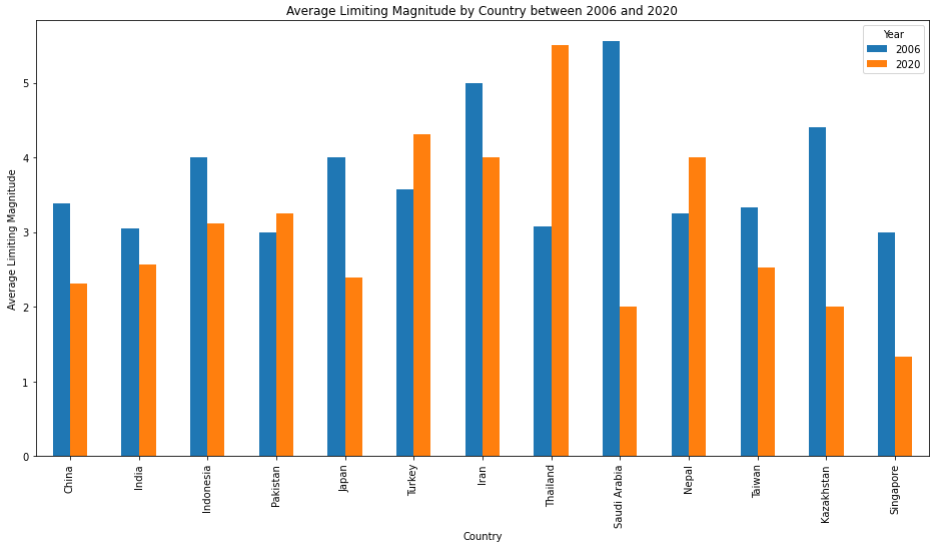
In [59]:
```
ganAsia = gan[gan.Continent == "AS"].pivot_table(index="Country", columns="Year", values="LimitingMag", aggfunc=np.mean)
asia_by_pop = gan[gan.Continent == "AS"].sort_values("Population", ascending=False)
ganAsia = ganAsia.loc[asia_by_pop.Country.unique()]
ganAsia.head()
```

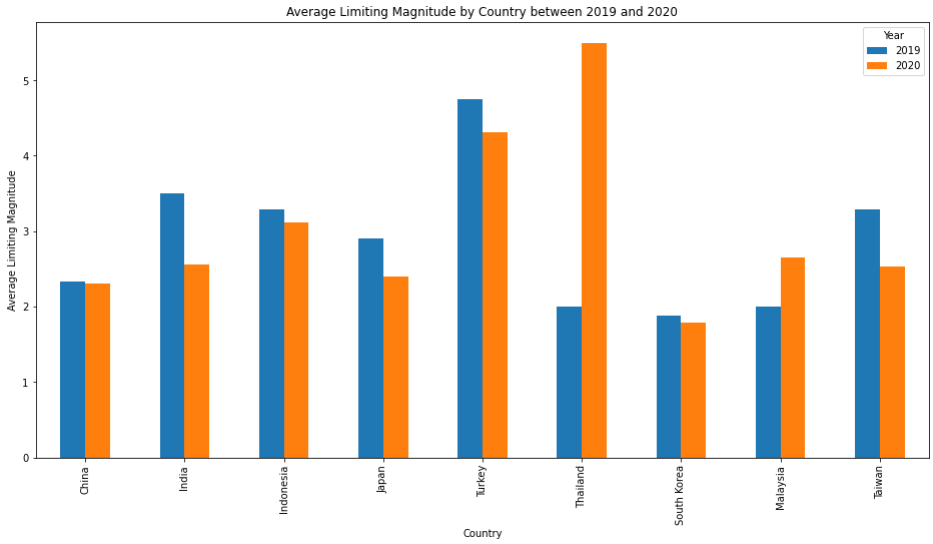
Out[59]:

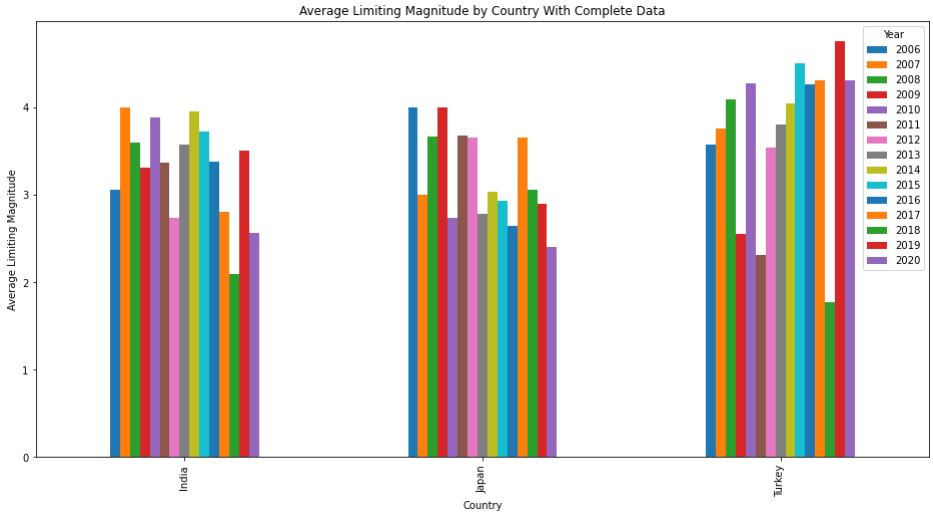
Year	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Country															
China	3.380952	NaN	NaN	3.400000	2.600000	3.285714	2.608014	3.080000	3.241379	5.049180	4.058824	4.157895	5.000000	2.333333	2.311111
India	3.051282	4.000000	3.6	3.311111	3.885714	3.364458	2.735825	3.576104	3.947368	3.717949	3.375000	2.808511	2.089552	3.500000	2.561644
Indonesia	4.000000	NaN	NaN	2.500000	NaN	3.384615	3.636364	3.636364	4.272727	3.250000	4.500000	3.333333	2.500000	3.294118	3.111111
Pakistan	3.000000	3.333333	3.4	4.000000	NaN	4.400000	1.000000	3.250000	3.652174	2.666667	NaN	4.000000	NaN	NaN	3.250000
Bangladesh	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.000000	NaN	6.000000	3.500000	NaN	NaN	NaN	2.000000

In [60]:
```
def findProportions(years=range(2006, 2021)):
    sorted_years = sorted(years)
    return ganAsia[sorted_years].dropna(how="any").plot.bar(xlabel="Country", ylabel="Average Limiting Magnitude", title=f"Average Limiting Magnitude by Country between {sorted_yea

findProportions([2006, 2020])
```

```
findProportions([2019, 2020])
_ = findProportions().set_title("Average Limiting Magnitude by Country With Complete Data")
```







This data shows that human observation has some limits, with many countries seeing a rapid decrease in data between 2006 and 2020, with the exception of Thailand. Covid-19 has also apparently had a similar decrease impact in apparent light pollution. Additonally, it seems that overall, India has had a decresing amount of Limiting Magnitude, as has Japan, although has been generally increasing.

Research Question C: What is the relation between the general demographics in each region/country and the Light Pollution?

To investigate this, we first merge the data amassed from the UN archives with that of the Night Light dataset. Then using an elaborate algorithm, we all categories that seems to show a strong correlation to the nightLight data columns.

In [61]:
```
countryData = undata_world.join(nightLight.reset_index().set_index(["Year", "Quantity"]).stack().reset_index().rename(columns={"level_2":"Country", 0: "Value"}).pivot(["Country", "
countryData = countryData[countryData.isna().sum(axis=1) <= 20]
countryData
```

Out[61]:

		Population	MalePopulation	FemalePopulation	SexRatio	ChildrenPopulation	ElderlyPopulation	SurfaceArea	PopulationDensity	ChildrenPercent	ElderlyPercent	...	Continent	ISO_A3	geome
Country	**Year**														
Afghanistan	**1990.0**														
		32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	AS	AFG	POLYG ((71.04 38.40 71.05 38.409

Country	Year	Population	MalePopulation	FemalePopulation	SexRatio	ChildrenPopulation	ElderlyPopulation	SurfaceArea	PopulationDensity	ChildrenPercent	ElderlyPercent	...	Continent	ISO_A3	geom
	1995.0	32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	AS	AFG	POLYG ((71.04 38.40 71.05 38.409
	2000.0	32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	AS	AFG	POLYG ((71.04 38.40 71.05 38.409
	2004.0	32294425.0	16589300.0	15705125.0	1.056675	1.458783e+07	1.282380e+06	652864.0	49.466075	45.548125	3.940875	...	AS	AFG	POLYG ((71.04 38.40 71.05 38.409
	2005.0	25654300.0	13239700.0	12414600.0	1.066462	1.227805e+07	9.384086e+05	652864.0	39.295200	47.859600	3.657900	...	AS	AFG	POLYG ((71.04 38.40 71.05 38.409
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Zimbabwe	**2015.0**	13414125.0	6398300.0	7015825.0	0.912198	5.628896e+06	6.088617e+05	390757.0	34.675250	41.928175	4.541625	...	AF	ZWE	POLYG ((30.01 -15.64 30.05 -15.6
	2016.0	13414125.0	6398300.0	7015825.0	0.912198	5.628896e+06	6.088617e+05	390757.0	34.675250	41.928175	4.541625	...	AF	ZWE	POLYG ((30.01 -15.64 30.05 -15.6
	2017.0	14236600.0	6777100.0	7459500.0	0.908508	6.064493e+06	6.367035e+05	390757.0	36.801300	42.597900	4.472300	...	AF	ZWE	POLYG ((30.01 -15.64 30.05 -15.6
	2019.0	14645500.0	6983400.0	7662100.0	0.911413	6.174265e+06	6.705296e+05	390757.0	37.858300	42.158100	4.578400	...	AF	ZWE	POLYG ((30.01 -15.64 30.05 -15.6
	2020.0	13414125.0	6398300.0	7015825.0	0.912198	5.628896e+06	6.088617e+05	390757.0	34.675250	41.928175	4.541625	...	AF	ZWE	POLYG ((30.01 -15.64 30.05 -15.6

2549 rows × 61 columns

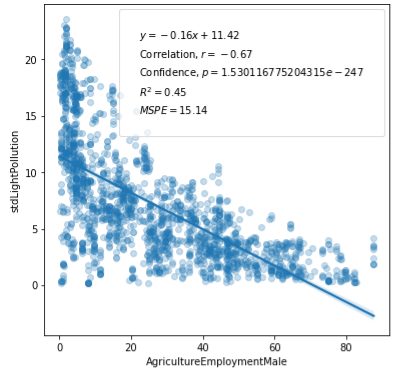
In [62]:
```
dt = (lambda dt: dt.loc[np.abs(dt.Correlation).sort_values(ascending=False).index])((lambda corr: pd.DataFrame(corr[(corr.index.str.contains("Light") | corr.index.str.contains("Mag
dt.loc[:, ["DemographicsCategory", "LightPollutionCategory"]] = dt.Categories.str.split(" and ").apply(pd.Series).rename(columns={0:"DemographicsCategory", 1:"LightPollutionCategor
dt = dt.drop(columns="Categories").sort_values(["DemographicsCategory", "LightPollutionCategory"]).set_index(["DemographicsCategory", "LightPollutionCategory"])
dt
```

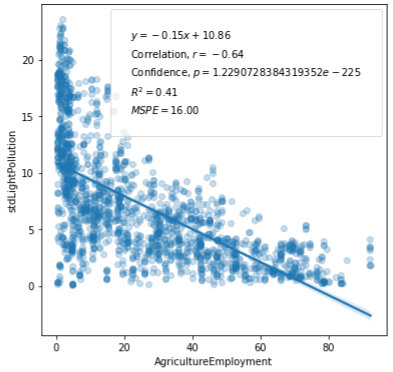
Out[62]:

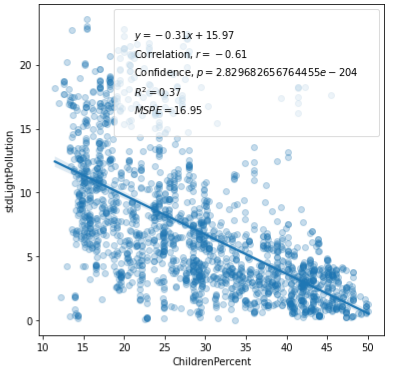
DemographicsCategory	LightPollutionCategory	Correlation
AgricultureEmployment	**stdLightPollution**	-0.643514
AgricultureEmploymentFemale	**stdLightPollution**	-0.584360
AgricultureEmploymentMale	**meanLightPollution**	-0.501465
	stdLightPollution	-0.669551
ChildrenPercent	**stdLightPollution**	-0.610607
EnergySupplyPerCapita	**stdLightPollution**	0.501009
FemaleLifeExpectancy	**stdLightPollution**	0.618253
FertilityRate	**stdLightPollution**	-0.551790
IndustryEmploymentMale	**stdLightPollution**	0.540194
InfantMortality	**stdLightPollution**	-0.577412
LifeExpectancy	**stdLightPollution**	0.629098
MaleLifeExpectancy	**stdLightPollution**	0.638168
MaternalMortality	**stdLightPollution**	-0.540934
PopulationDensity	**meanLightPollution**	0.566343
	medianLightPollution	0.593178
	minLightPollution	0.818255
	modeLightPollution	0.559177
SecondaryFemaleGrossEnrolment	**stdLightPollution**	0.534146
SecondaryMaleGrossEnrolment	**stdLightPollution**	0.540972
ServicesEmployment	**stdLightPollution**	0.582492
ServicesEmploymentFemale	**stdLightPollution**	0.588929
ServicesEmploymentMale	**stdLightPollution**	0.589073

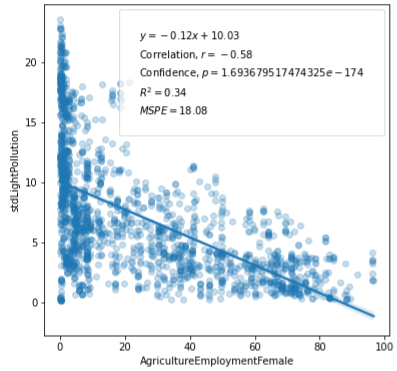
Now to investigate the correlations and their general R^2 values and $MSPE$ values, we run through a for loop as shown below:

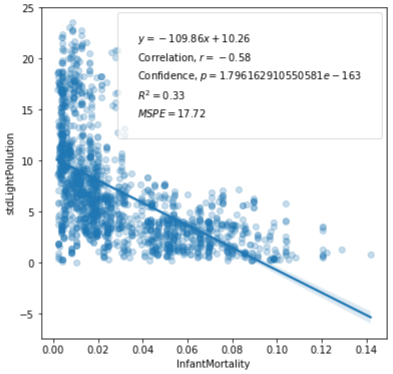
In [65]:
```
fig, axes = plt.subplots(7, 3,figsize=(21, 50))
for i, (x, y) in enumerate(dt.sort_values("Correlation").iloc[:-1].apply(axis=1, func=lambda row: row.name).reset_index()[0].tolist()):
    consideration = countryData[[x, y]].dropna()
    x, y = consideration[x], consideration[y]
    ax = axes[i//3, i%3]
    m, c, r, p, stderr = stats.linregress(x=x, y=y)
    mspe = mean_squared_error(y, m*x + c)
    sns.regplot(x=x, y=y, ax=ax, scatter_kws={'alpha': 0.25})

    handles = [mpl_patches.Rectangle((0, 0), 1, 1, fc="white", ec="white",
                                     lw=0, alpha=0)] * 5

    labels = ("$y = {:.2f} x + {:.2f}$\nCorrelation, $r = {:.2f}$\nConfidence, $p = {}$\n$R^2 = {:.2f}$\n$MSPE = {:.2f}$").format(m, c, r, p, r**2, mspe).split("\n")
    ax.legend(handles, labels, loc='best',
              fancybox=True, framealpha=0.7,
              handlelength=0, handletextpad=0, borderpad=2)
```

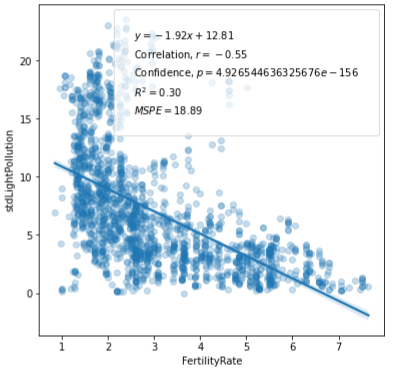

$y = -0.16x + 11.42$
Correlation, $r = -0.67$
Confidence, $p = 1.530116775204315e-247$
$R^2 = 0.45$
$MSPE = 15.14$
stdLightPollution
AgricultureEmploymentMale

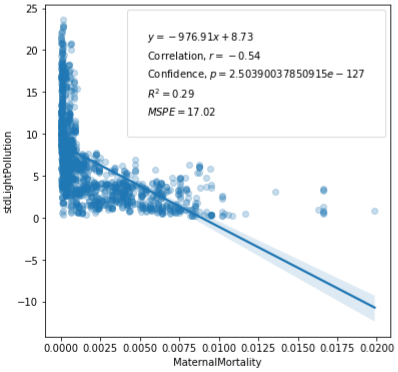

$y = -0.15x + 10.86$
Correlation, $r = -0.64$
Confidence, $p = 1.2290728384319352e-225$
$R^2 = 0.41$
$MSPE = 16.00$
stdLightPollution
AgricultureEmployment

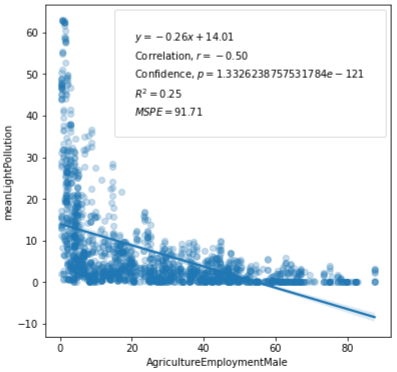

$y = -0.31x + 15.97$
Correlation, $r = -0.61$
Confidence, $p = 2.8296826567644455e-204$
$R^2 = 0.37$
$MSPE = 16.95$
stdLightPollution
ChildrenPercent

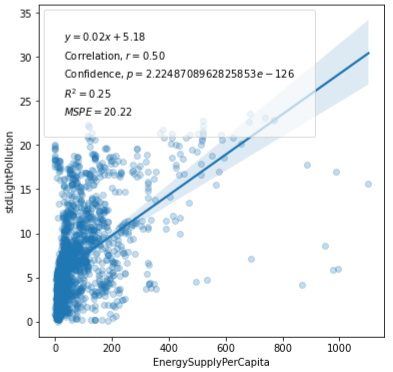

$y = -0.12x + 10.03$
Correlation, $r = -0.58$
Confidence, $p = 1.693679517474325e-174$
$R^2 = 0.34$
$MSPE = 18.08$
stdLightPollution
AgricultureEmploymentFemale

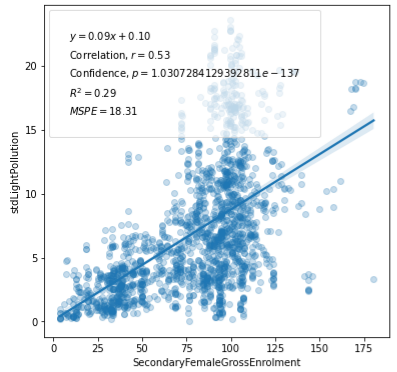

$y = -109.86x + 10.26$
Correlation, $r = -0.58$
Confidence, $p = 1.796162910550581e-163$
$R^2 = 0.33$
$MSPE = 17.72$
stdLightPollution
InfantMortality

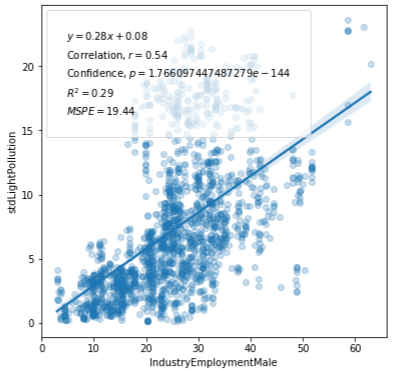

$y = -1.92x + 12.81$
Correlation, $r = -0.55$
Confidence, $p = 4.926544636325676e-156$
$R^2 = 0.30$
$MSPE = 18.89$
stdLightPollution
FertilityRate

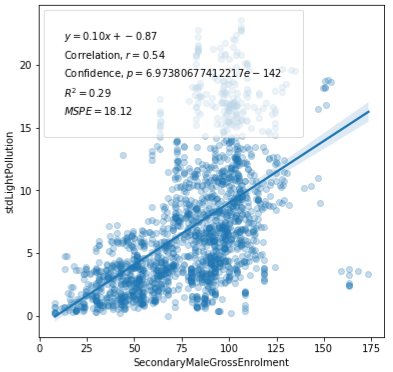

$y = -976.91x + 8.73$
Correlation, $r = -0.54$
Confidence, $p = 2.503900378509 15e-127$
$R^2 = 0.29$
$MSPE = 17.02$
stdLightPollution
MaternalMortality

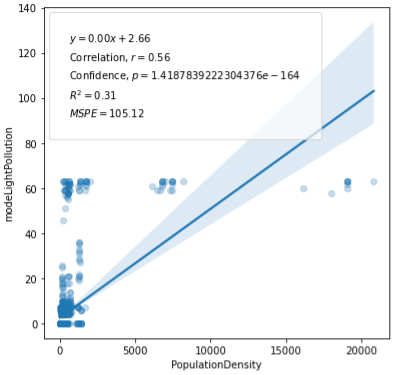

$y = -0.26x + 14.01$
Correlation, $r = -0.50$
Confidence, $p = 1.3326238757531784e-121$
$R^2 = 0.25$
$MSPE = 91.71$
meanLightPollution
AgricultureEmploymentMale

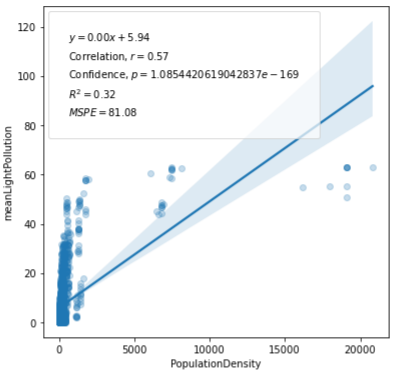

$y = 0.02x + 5.18$
Correlation, $r = 0.50$
Confidence, $p = 2.2248708962825853e-126$
$R^2 = 0.25$
$MSPE = 20.22$
stdLightPollution
EnergySupplyPerCapita

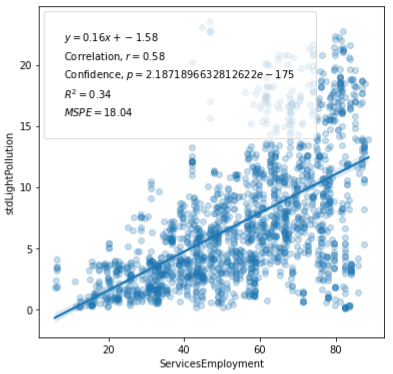

$y = 0.09x + 0.10$
Correlation, $r = 0.53$
Confidence, $p = 1.0307284129392811e-137$
$R^2 = 0.29$
$MSPE = 18.31$
stdLightPollution
SecondaryFemaleGrossEnrolment

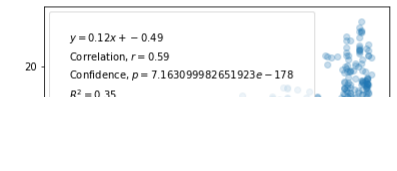

$y = 0.28x + 0.08$
Correlation, $r = 0.54$
Confidence, $p = 1.7660974474872 79e-144$
$R^2 = 0.29$
$MSPE = 19.44$
stdLightPollution
IndustryEmploymentMale

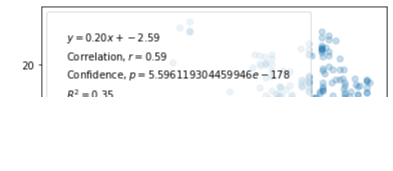

$y = 0.10x + -0.87$
Correlation, $r = 0.54$
Confidence, $p = 6.973806774122 17e-142$
$R^2 = 0.29$
$MSPE = 18.12$
stdLightPollution
SecondaryMaleGrossEnrolment

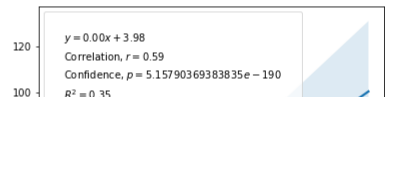

$y = 0.00x + 2.66$
Correlation, $r = 0.56$
Confidence, $p = 1.4187839222304376e-164$
$R^2 = 0.31$
$MSPE = 105.12$
modeLightPollution
PopulationDensity

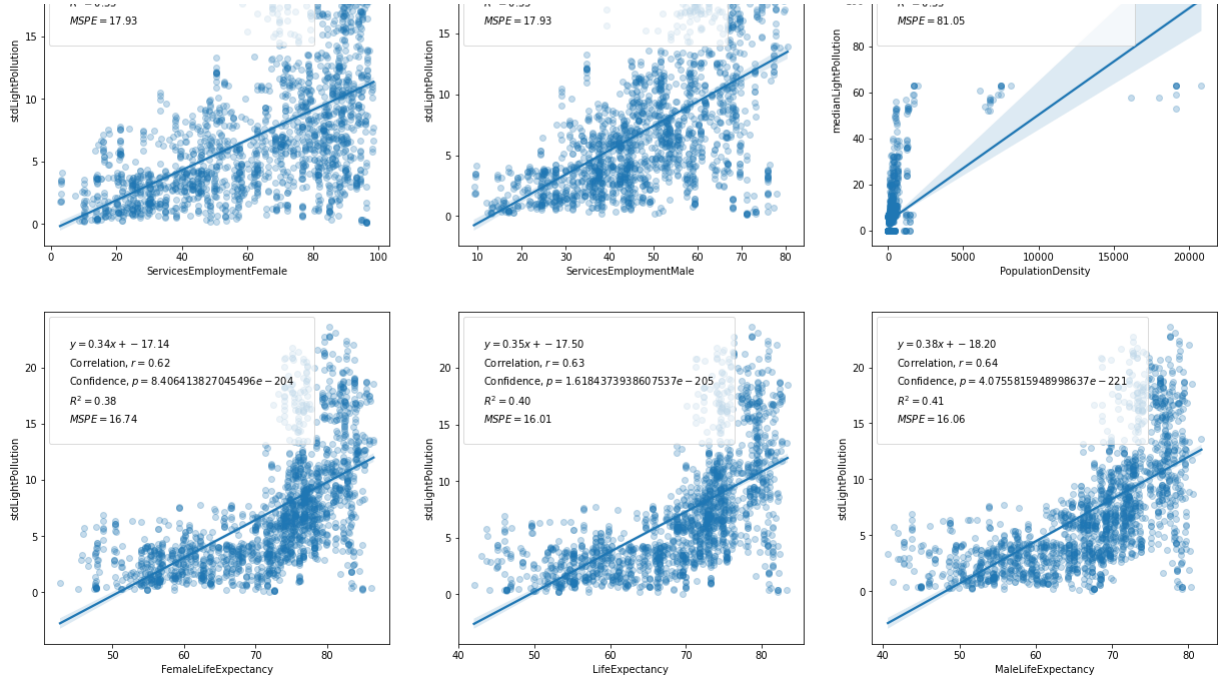

$y = 0.00x + 5.94$
Correlation, $r = 0.57$
Confidence, $p = 1.0854420619042837e-169$
$R^2 = 0.32$
$MSPE = 81.08$
meanLightPollution
PopulationDensity


$y = 0.16x + -1.58$
Correlation, $r = 0.58$
Confidence, $p = 2.187189663281262 2e-175$
$R^2 = 0.34$
$MSPE = 18.04$
stdLightPollution
ServicesEmployment


$y = 0.12x + -0.49$
Correlation, $r = 0.59$
Confidence, $p = 7.163099982651923e-178$


$y = 0.20x + -2.59$
Correlation, $r = 0.59$
Confidence, $p = 5.596119304459946e-178$


$y = 0.00x + 3.98$
Correlation, $r = 0.59$
Confidence, $p = 5.157903693838 35e-190$



We now investigate the best correlation found, which is the Population Density against the Minimum Light Pollution.

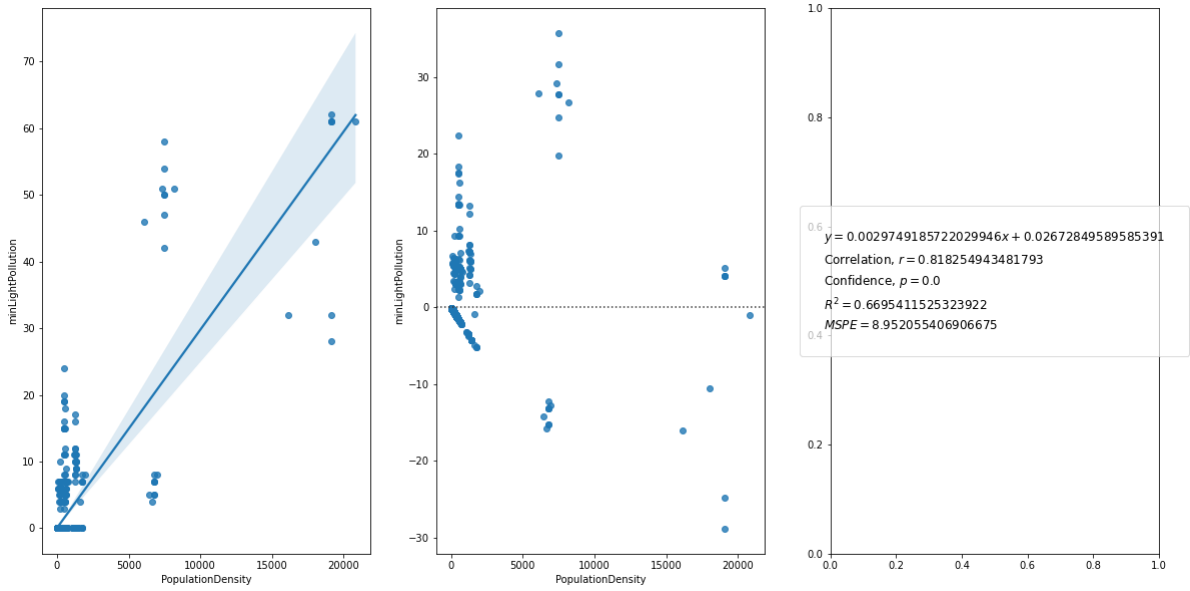
According to the following results,

$$R^2 = 0.67$$

$$MSPE = 677251$$

In [66]:
```
x, y = dt.sort_values("Correlation").iloc[-1].name
consideration = countryData[[x, y]].dropna()
x, y = consideration[x], consideration[y]
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(20, 10))
m, c, r, p, stderr = stats.linregress(x=x, y=y)
mspe = mean_squared_error(y, m*x + c)
sns.regplot(x=x, y=y, ax=ax1)

handles = [mpl_patches.Rectangle((0, 0), 1, 1, fc="white", ec="white", lw=0, alpha=0)] * 5
labels = f"$y = {m} x + {c}$\nCorrelation, $r = {r}$\nConfidence, $p = {p}$\n$R^2 = {r**2}$\n$MSPE = {mspe}$".split("\n")

ax3.legend(handles, labels, loc='center', fontsize='large',
           fancybox=True, framealpha=0.7,
           handlelength=0, handletextpad=0, borderpad=2)

sns.residplot(x=x,y=y,ax=ax2)
```

Out[66]:
```
<AxesSubplot:xlabel='PopulationDensity', ylabel='minLightPollution'>
```



We now train a MLRM to predict the average light pollution, using all the factors we have but normalized using the following algorithm:

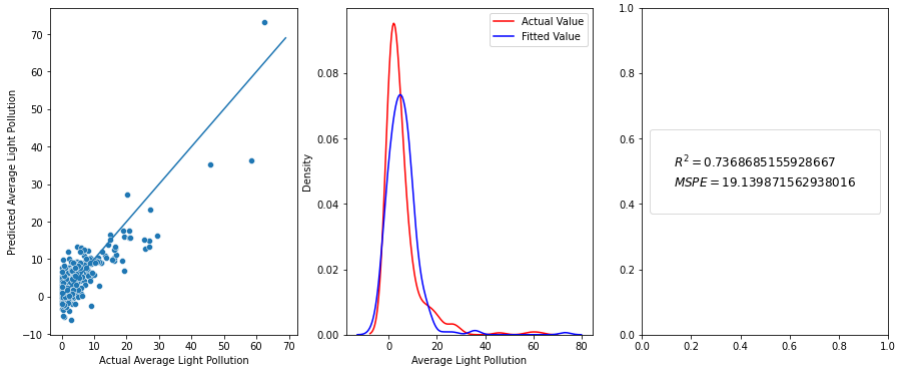
$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

In [67]:
```
dt.index = dt.index.swaplevel(0, 1)
data = pd.DataFrame(countryData.dropna())
X = data.loc[:, :'EnergyStockChange']
X = (X - X.min()) / (X.max() - X.min())
y = data.meanLightPollution

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

lm = LinearRegression().fit(X_train, y_train)

y_pred = lm.predict(X_test)

mspe = mean_squared_error(y_test, y_pred)
r2 = lm.score(X_train, y_train)
```

```
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 6))

sns.scatterplot(y_test, y_pred, ax=ax1).set(xlabel="Actual Average Light Pollution", ylabel="Predicted Average Light Pollution")
ax1.plot(np.arange(0, 70))

sns.kdeplot(y_test, color='r', label='Actual Value', ax=ax2).set_xlabel("Average Light Pollution")
sns.kdeplot(y_pred,  color='b', label='Fitted Value', ax=ax2)
ax2.legend()

handles = [mpl_patches.Rectangle((0, 0), 1, 1, fc="white", ec="white",
                                 lw=0, alpha=0)] * 2

labels = f"$R^2 = {r2}$\n$MSPE = {mspe}$".split("\n")

ax3.legend(handles, labels, loc="center", fontsize='large',
           fancybox=True, framealpha=0.7,
           handlelength=0, handletextpad=0, borderpad=2)

pd.Series(dict(zip(["Intercept"]+list(X.columns), [lm.intercept_]+list(lm.coef_)))).reset_index().rename(columns={"index":"Normalised Category", 0:"Weightage"}).sort_values("Weight
```

Out[67]:

	Normalised Category	Weightage
1	Population	5.293570e+06
34	AgricultureEmploymentMale	5.436634e+04
40	ServicesEmploymentMale	4.397763e+04
37	IndustryEmploymentMale	3.736779e+04
0	Intercept	1.823296e+04
35	AgricultureEmploymentFemale	3.265242e+03
41	ServicesEmploymentFemale	3.255133e+03
42	EnergyProduction	2.439759e+03
38	IndustryEmploymentFemale	1.850577e+03
45	EnergyTrade	1.368335e+03
12	LifeExpectancy	4.717234e+02
8	PopulationDensity	6.956451e+01
30	UnemploymentRate	6.468125e+01
5	ChildrenPopulation	2.099872e+01
23	SecondaryFemaleGrossEnrolment	1.316380e+01
15	FertilityRate	1.030637e+01
19	PrimaryMaleGrossEnrolment	6.524279e+00
26	TertiaryFemaleGrossEnrolment	5.954115e+00
11	PopulationIncrease	5.390424e+00
6	ElderlyPopulation	5.165643e+00
24	TertiaryStudents	4.262590e+00
10	ElderlyPercent	3.962378e+00
29	LabourForceFemale	3.884055e+00
21	SecondaryStudents	1.338709e+00
44	EnergySupplyPerCapita	1.143696e+00
27	LabourForce	-5.141083e-01
17	MaternalMortality	-6.024100e-01
28	LabourForceMale	-3.154849e+00
16	InfantMortality	-6.185609e+00
22	SecondaryMaleGrossEnrolment	-6.658631e+00
7	SurfaceArea	-7.586657e+00
20	PrimaryFemaleGrossEnrolment	-8.367341e+00
9	ChildrenPercent	-1.253430e+01
25	TertiaryMaleGrossEnrolment	-1.658423e+01
18	PrimaryStudents	-2.374053e+01
31	UnemploymentRateMale	-2.585655e+01
4	SexRatio	-3.888542e+01
32	UnemploymentRateFemale	-4.023891e+01
46	EnergyStockChange	-9.927031e+01
13	MaleLifeExpectancy	-1.953201e+02
14	FemaleLifeExpectancy	-2.853619e+02
43	EnergySupply	-2.991665e+03
36	IndustryEmployment	-4.539329e+04
39	ServicesEmployment	-6.704665e+04
33	AgricultureEmployment	-7.651231e+04
3	FemalePopulation	-2.576025e+06
2	MalePopulation	-2.717545e+06



From here, we can conclude that Population, Employment, Energy Production and Life Expectancies have a significant impact on the average light pollution.

Results Discussion

Present and discuss your research results. Treat each of your research questions separately. Focus in particular on the results that are most interesting, surprising, or important. Discuss the consequences or implications. Interpret the results: if the answers are unexpected, then see whether you can find an explanation for them, such as an external factor that your analysis did not account for. Include some visualization of your results (a graph, plot, bar chart, etc.). These plots should be created programmatically in the code you submit.

What are locations of minimal light pollution intensity which are optimum for astronomical observation?

We find that the Auckland Islands are the best for Astronomical Observation, and especially for placing an observatory around there. This is found based on the fact that the location ranks 7th in University Ranking and Population Density Ranking and 17th in Area Ranking, and has an overall score of 30, which is the lowest on the list. This is not surprising, since countries in Oceania do in fact have lower light pollution that most locations humanly habitable.

In [68]:
```
m = folium.Map(location=[-50.7092377,165.9796024], zoom_start=10, tiles="Stamen Terrain")
tile = folium.TileLayer(tiles='https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', attr='Esri', name='Esri Satellite', overlay=True, co
plugins.Fullscreen().add_to(m)
m
```

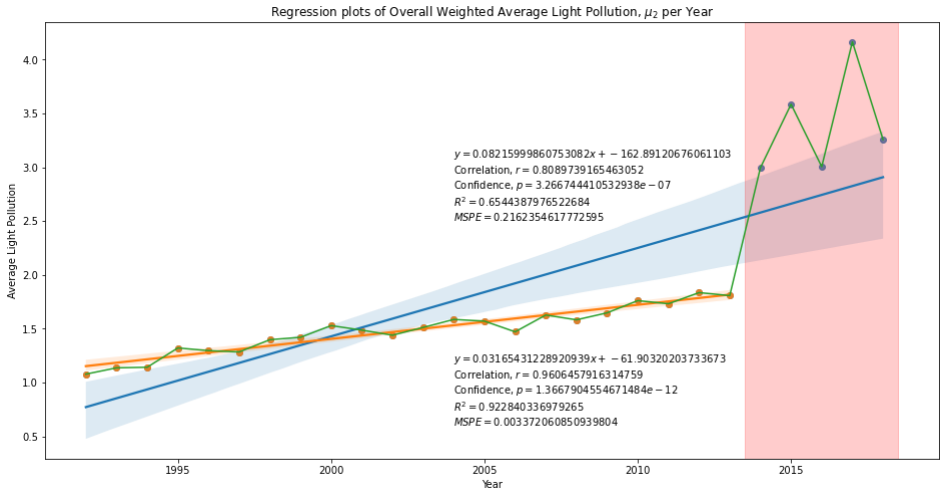
Out[68]:

How has Light Pollution Changed Over the Years

It seems that Light Pollution has generally incresed over the past few years, seeing a rapid increase between 2013 and 2014, likely due to the dataset used changing altogether.

In [69]:
```
nightLightByQuan = nightLight[nightLight.Quantity.isin(["mean", "count"])].reset_index().set_index(["Quantity", "Year"])
fitted_mean_by_yr = ((nightLightByQuan.loc["mean"] * nightLightByQuan.loc["count"]).sum(axis=1) / nightLightByQuan.loc["count"].sum(axis=1)).reset_index().rename(columns={0:"Averag
summary(fitted_mean_by_yr, 2004, 2.5)
summary(fitted_mean_by_yr[fitted_mean_by_yr.Year.isin(range(1992, 2014))], 2004, 0.6)
sns.lineplot(data=fitted_mean_by_yr, x="Year", y="Average Light Pollution").axvspan(xmin=2013.5, xmax=2018.5, color="r", alpha=0.2)
plt.title("Regression plots of Overall Weighted Average Light Pollution, $\\mu_2$ per Year")
```

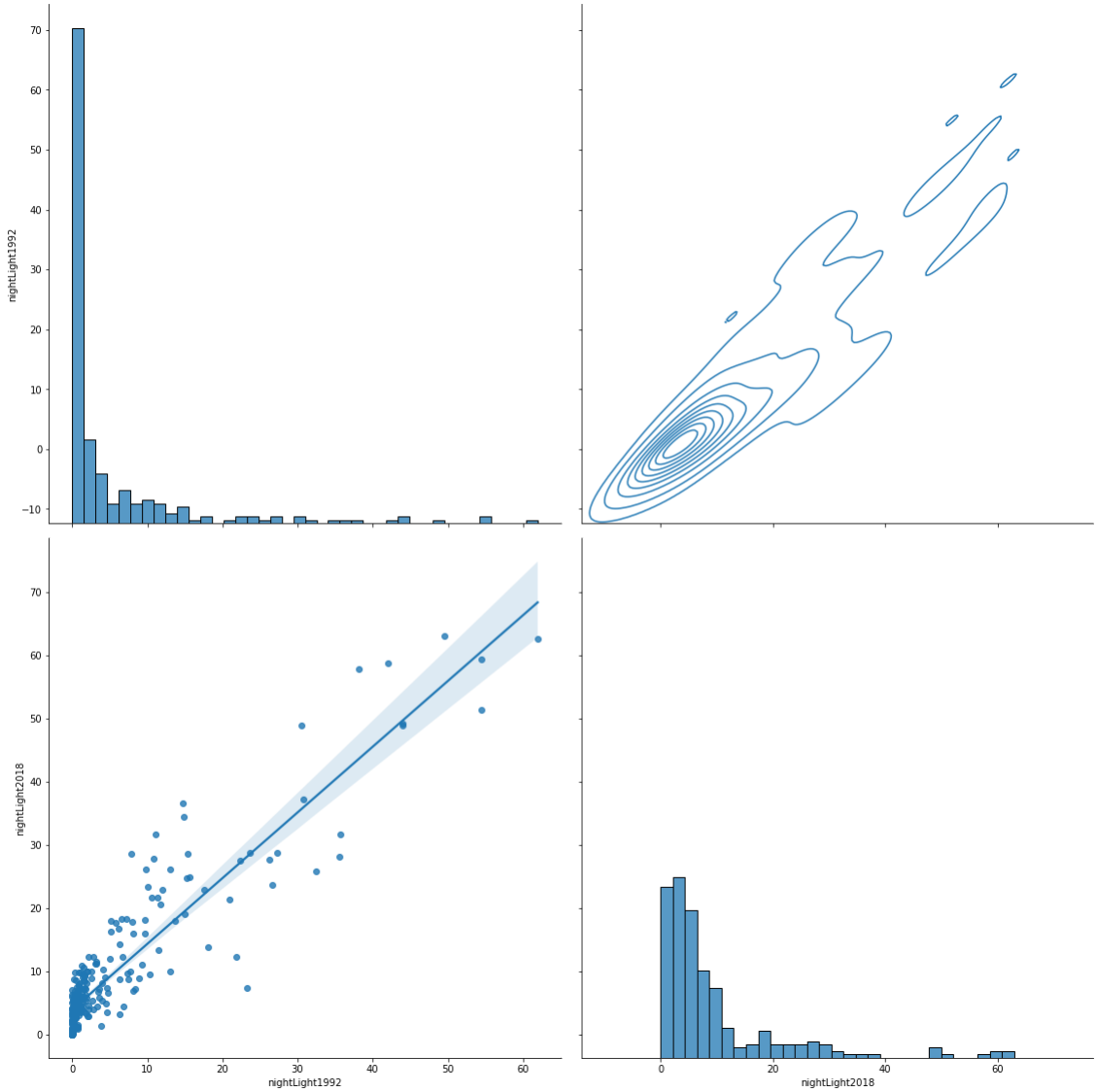
Out[69]:
```
Text(0.5, 1.0, 'Regression plots of Overall Weighted Average Light Pollution, $\\mu_2$ per Year')
```



It seems that the data has generally remained consistent between 1992 and 2018, despite rapid advancements in digital technology. This could be due to the fact that 1992 was already a year where digital technology was in its prime.

In [70]:
```
sns.PairGrid(pivotNightLight[["nightLight1992", "nightLight2018"]], height=8).map_diag(sns.histplot).map_lower(sns.regplot).map_upper(sns.kdeplot)
```

Out[70]:
```
<seaborn.axisgrid.PairGrid at 0x1e4a420cd60>
```



Which countries are most susceptible to light pollution and which have improved over the years?

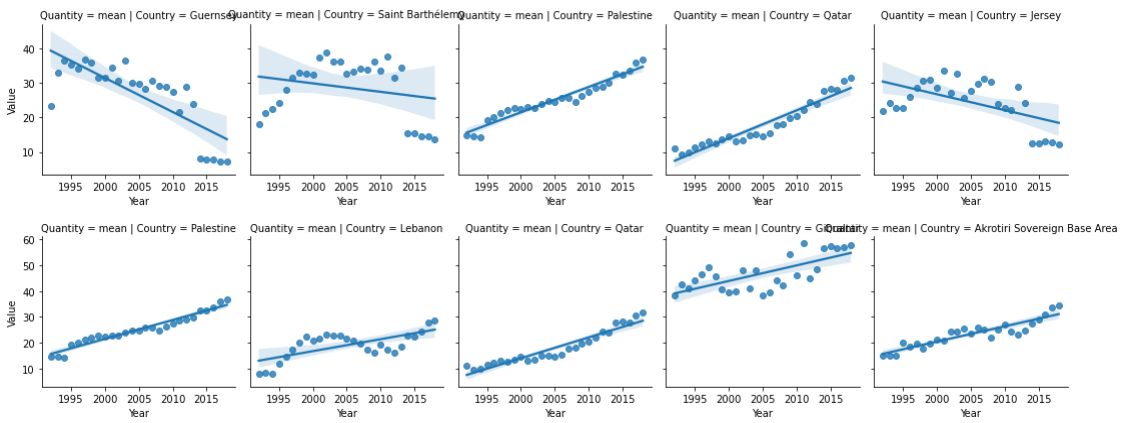
It appears that Palestine, Qatar, Macau and Gibraltar are most susceptible to light pollution, while Guernsey, Bermuda, US Virgin Islands and Jersey have improved very much.

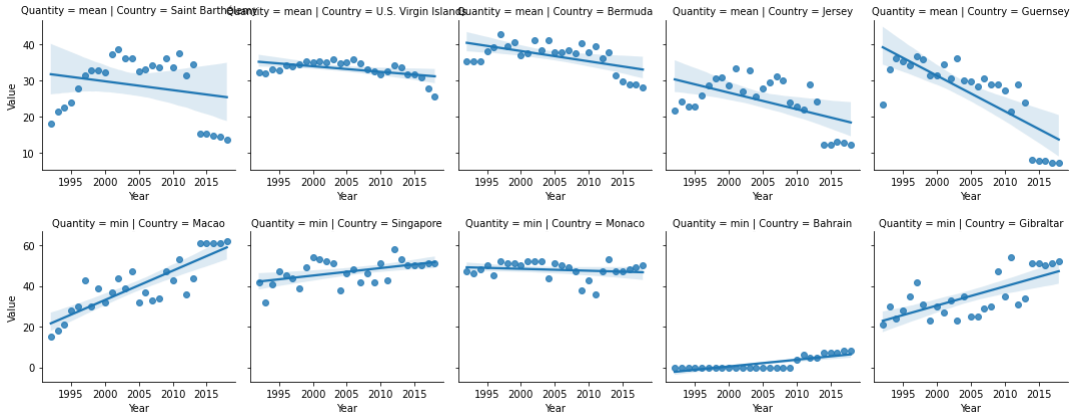
In [71]:

```
index = nightLightHighLow
df = nightLight.reset_index().set_index(["Quantity", "Year"])[index.index].loc[["mean"]].stack().reset_index().rename(columns={"level_2":"Country", 0: "Value"})
sns.FacetGrid(df, row="Quantity", col="Country").map(sns.regplot, "Year", "Value")

index = nightLightNet.iloc[:5]
df = nightLight.reset_index().set_index(["Quantity", "Year"])[index.index].loc[["mean"]].stack().reset_index().rename(columns={"level_2":"Country", 0: "Value"})
sns.FacetGrid(df, row="Quantity", col="Country").map(sns.regplot, "Year", "Value")

index = nightLightNet.iloc[-5:]
df = nightLight.reset_index().set_index(["Quantity", "Year"])[index.index].loc[["mean"]].stack().reset_index().rename(columns={"level_2":"Country", 0: "Value"})
sns.FacetGrid(df, row="Quantity", col="Country").map(sns.regplot, "Year", "Value")

index = nightLightFilter(slice(0, 5))
index = index[index.Quantity == "min"]
sns.FacetGrid(index, row="Quantity", col="Country").map(sns.regplot, "Year", "Value")
```

Out[71]:

```
<seaborn.axisgrid.FacetGrid at 0x1e4b5576b20>
```



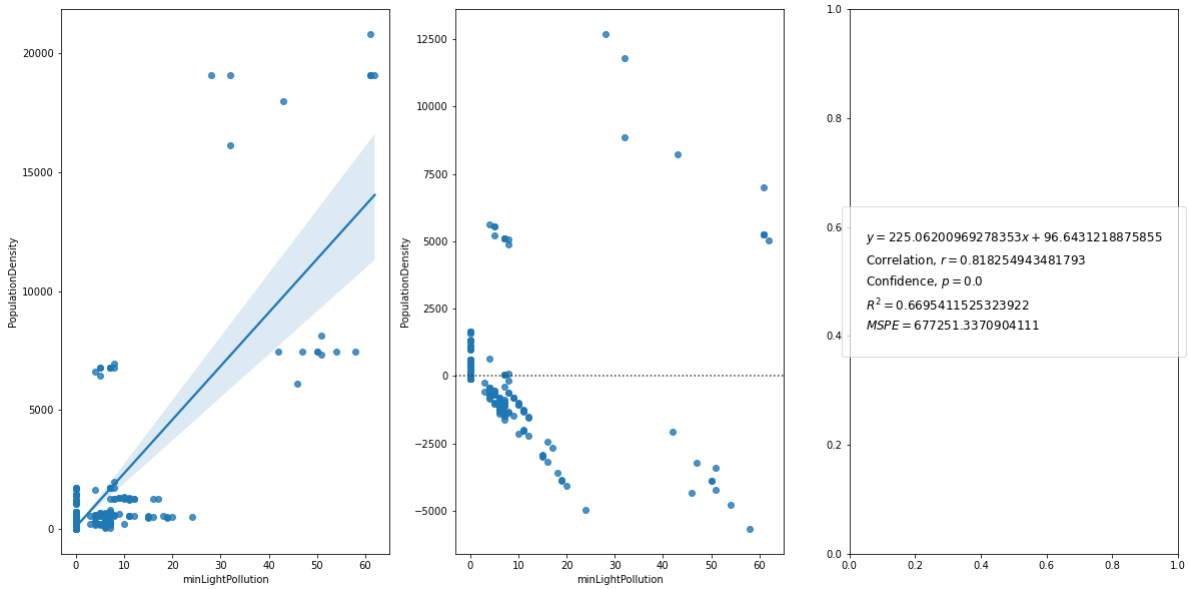


Which factors have a significant impact on the Light Pollution Standards?

We find that Population Density has a significant positive correlation with the Minimum Light Pollution, while Population, Employment, Energy Production and Life Expectancies have a significant impact on the average light pollution.

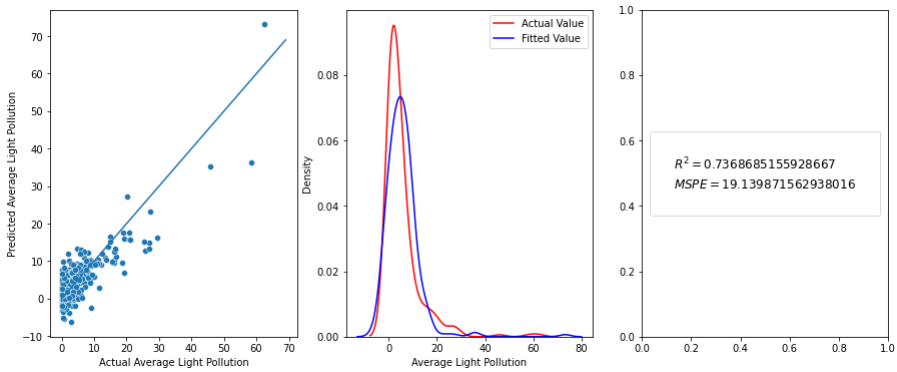
In [72]:
```
x, y = dt.sort_values("Correlation").iloc[-1].name
consideration = countryData[[x, y]].dropna()
x, y = consideration[x], consideration[y]
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(20, 10))
m, c, r, p, stderr = stats.linregress(x=x, y=y)
mspe = mean_squared_error(y, m*x + c)
sns.regplot(x=x, y=y, ax=ax1)

handles = [mpl_patches.Rectangle((0, 0), 1, 1, fc="white", ec="white", lw=0, alpha=0)] * 5
labels = f"$y = {m} x + {c}$\nCorrelation, $r = {r}$\nConfidence, $p = {p}$\n$R^2 = {r**2}$\n$MSPE = {mspe}$".split("\n")

ax3.legend(handles, labels, loc='center', fontsize='large',
           fancybox=True, framealpha=0.7,
           handlelength=0, handletextpad=0, borderpad=2)

sns.residplot(x=x,y=y,ax=ax2)
```

Out[72]:
```
<AxesSubplot:xlabel='minLightPollution', ylabel='PopulationDensity'>
```



In [73]:
```
dt.index = dt.index.swaplevel(0, 1)
data = pd.DataFrame(countryData.dropna())
X = data.loc[:, :'EnergyStockChange']
X = (X - X.min()) / (X.max() - X.min())
y = data.meanLightPollution

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

lm = LinearRegression().fit(X_train, y_train)

y_pred = lm.predict(X_test)

mspe = mean_squared_error(y_test, y_pred)
r2 = lm.score(X_train, y_train)

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 6))

sns.scatterplot(y_test, y_pred, ax=ax1).set(xlabel="Actual Average Light Pollution", ylabel="Predicted Average Light Pollution")
ax1.plot(np.arange(0, 70))

sns.kdeplot(y_test, color='r', label='Actual Value', ax=ax2).set_xlabel("Average Light Pollution")
sns.kdeplot(y_pred,  color='b', label='Fitted Value', ax=ax2)
ax2.legend()

handles = [mpl_patches.Rectangle((0, 0), 1, 1, fc="white", ec="white",
                                 lw=0, alpha=0)] * 2

labels = f"$R^2 = {r2}$\n$MSPE = {mspe}$".split("\n")

ax3.legend(handles, labels, loc="center", fontsize='large',
           fancybox=True, framealpha=0.7,
           handlelength=0, handletextpad=0, borderpad=2)

#pd.Series(dict(zip(["Intercept"]+list(X.columns), [lm.intercept_]+list(lm.coef_)))).reset_index().rename(columns={"index":"Normalised Category", 0:"Weightage"}).sort_values("Weigh
```

Out[73]:
```
<matplotlib.legend.Legend at 0x1e4b4debd60>
```



Conclusion and recommendations

Based on the results, countries like Singapore, Qatar and Palestine have to adopt a differing approach to combat the problem of light pollution, and countries with a high Population Density need to beware of possible increase in light pollution over the next few years.

References

Cite any references made, and links where you obtained the data.

You may wish to read about how to use markdown in Jupyter notebook to make your report easier to read. https://www.ibm.com/docs/en/db2-event-store/2.0.0?topic=notebooks-markdown-jupyter-cheatsheet

Readings

[1] Drake, N. (2019, April 3). Light pollution is getting worse, and Earth is paying the price. National Geographic. https://www.nationalgeographic.com/science/article/nights-are-getting-brighter-earth-paying-the-price-light-pollution-dark-skies

[2] Are the stars out tonight? (n.d.). Notre Dame Senior School ArcGIS Maps. Retrieved October 10, 2021, from https://notredamecobham.maps.arcgis.com/apps/Cascade/index.html?appid=0d5f0c8b80ca4fff9ff5aa6834b68a63

[3] Lazar, M. (2010). Shedding Light on the Global Distribution of Economic Activity. The Open Geography Journal, 3(1), 147–160. https://doi.org/10.2174/1874923201003010147

In [ ]: