Correlation between Frequency and Strength of Hurricanes and the Rising Sea Surface Temperatures

Casie Price

Arizona State University

GIS 322: Programming Principles of GIS II

Dr. Wenwen Li

April 26th, 2022

Introduction

Global warming is a hot topic of discussion currently across the globe. Learning what we as people are doing wrong and ways to correct what damage has already been done in order to curb and/or slow down climate change is at the forefront of some individual's minds. The oceans are warming at an alarming rate, so this has posed a question. Does the rising of the water temperature in the oceans contribute to an increase in frequency and strength of storms in forming in the Atlantic? Hurricanes use warm sea surface temperatures as a fuel for growth along with other specific circumstances such as wind. With hurricanes there is also a substantial threat of destruction if the storms make landfall. Understanding more about hurricanes and their impact may help better prepare. My project was to break down the 6 hurricanes seasons for 2016 thru 2021 and to see if there was a correlation between the frequency / number of storms and land-ocean temperature.

Data

The storm data was obtained from NOAA's National Hurricane Center ((National Oceanic and Atmospheric Administration, n.d.)). NOAA has the data broken down by year and individual storm with

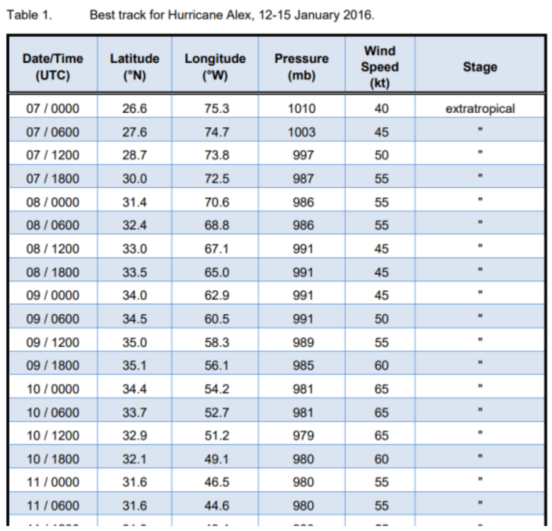


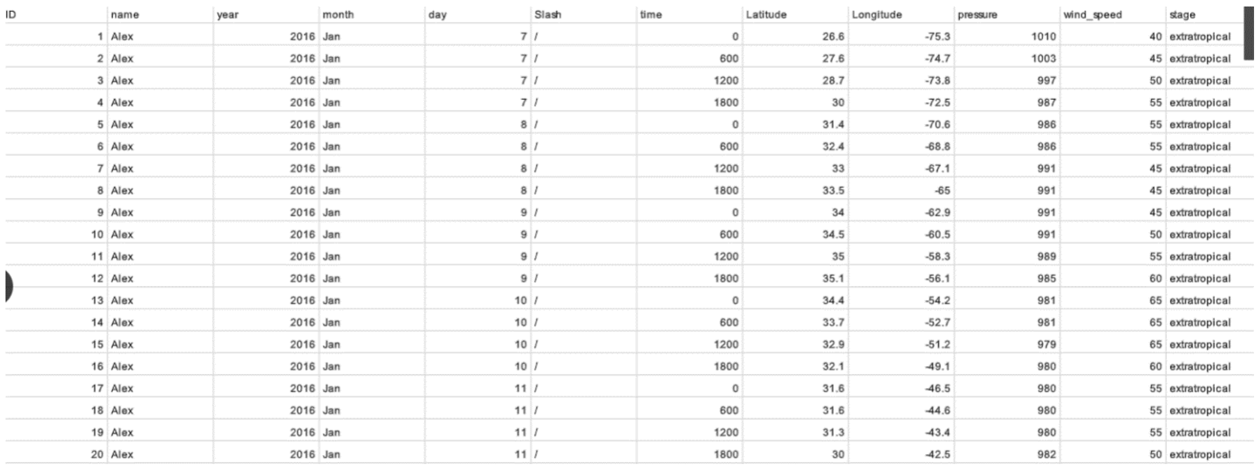
Table 1. Best track for Hurricane Alex, 12-15 January 2016.

Date/Time (UTC)	Latitude (°N)	Longitude (°W)	Pressure (mb)	Wind Speed (kt)	Stage
07 / 0000	26.6	75.3	1010	40	extratropical
07 / 0600	27.6	74.7	1003	45	"
07 / 1200	28.7	73.8	997	50	"
07 / 1800	30.0	72.5	987	55	"
08 / 0000	31.4	70.6	986	55	"
08 / 0600	32.4	68.8	986	55	"
08 / 1200	33.0	67.1	991	45	"
08 / 1800	33.5	65.0	991	45	"
09 / 0000	34.0	62.9	991	45	"
09 / 0600	34.5	60.5	991	50	"
09 / 1200	35.0	58.3	989	55	"
09 / 1800	35.1	56.1	985	60	"
10 / 0000	34.4	54.2	981	65	"
10 / 0600	33.7	52.7	981	65	"
10 / 1200	32.9	51.2	979	65	"
10 / 1800	32.1	49.1	980	60	"
11 / 0000	31.6	46.5	980	55	"
11 / 0600	31.6	44.6	980	55	"

the data in PDF files (Figure 1).

Figure 1- Example of NOAA Storm PDF

Due to this information being in a PDF, I compiled all the storm data for the years 2016 thru 2021 into a CSV file format (Figure 2).



ID	name	year	month	day	Slash	time	Latitude	Longitude	pressure	wind_speed	stage
1	Alex	2016	Jan	7	/	0	26.6	-75.3	1010	40	extratropical
2	Alex	2016	Jan	7	/	600	27.6	-74.7	1003	45	extratropical
3	Alex	2016	Jan	7	/	1200	28.7	-73.8	997	50	extratropical
4	Alex	2016	Jan	7	/	1800	30	-72.5	987	55	extratropical
5	Alex	2016	Jan	8	/	0	31.4	-70.6	986	55	extratropical
6	Alex	2016	Jan	8	/	600	32.4	-68.8	986	55	extratropical
7	Alex	2016	Jan	8	/	1200	33	-67.1	991	45	extratropical
8	Alex	2016	Jan	8	/	1800	33.5	-65	991	45	extratropical
9	Alex	2016	Jan	9	/	0	34	-62.9	991	45	extratropical
10	Alex	2016	Jan	9	/	600	34.5	-60.5	991	50	extratropical
11	Alex	2016	Jan	9	/	1200	35	-58.3	989	55	extratropical
12	Alex	2016	Jan	9	/	1800	35.1	-56.1	985	60	extratropical
13	Alex	2016	Jan	10	/	0	34.4	-54.2	981	65	extratropical
14	Alex	2016	Jan	10	/	600	33.7	-52.7	981	65	extratropical
15	Alex	2016	Jan	10	/	1200	32.9	-51.2	979	65	extratropical
16	Alex	2016	Jan	10	/	1800	32.1	-49.1	980	60	extratropical
17	Alex	2016	Jan	11	/	0	31.6	-46.5	980	55	extratropical
18	Alex	2016	Jan	11	/	600	31.6	-44.6	980	55	extratropical
19	Alex	2016	Jan	11	/	1200	31.3	-43.4	980	55	extratropical
20	Alex	2016	Jan	11	/	1800	30	-42.5	982	50	extratropical

Figure 2 - CSV Storm Data

I then created a map in ArcMap and exported the map as a SHAPE file which generated a geometric shape to use in my code (Figure 3).

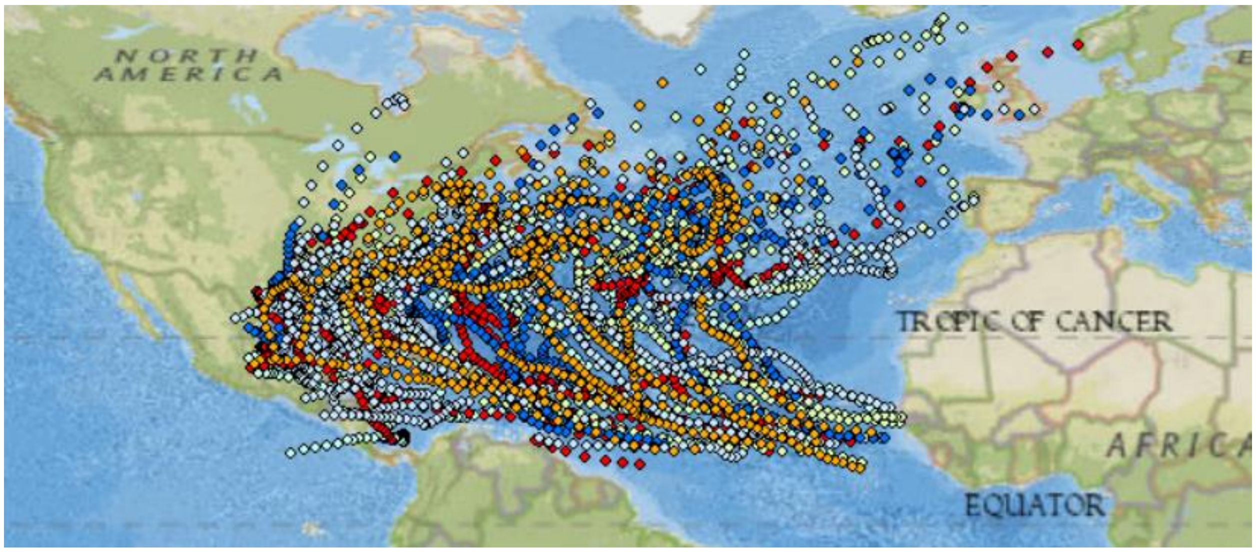
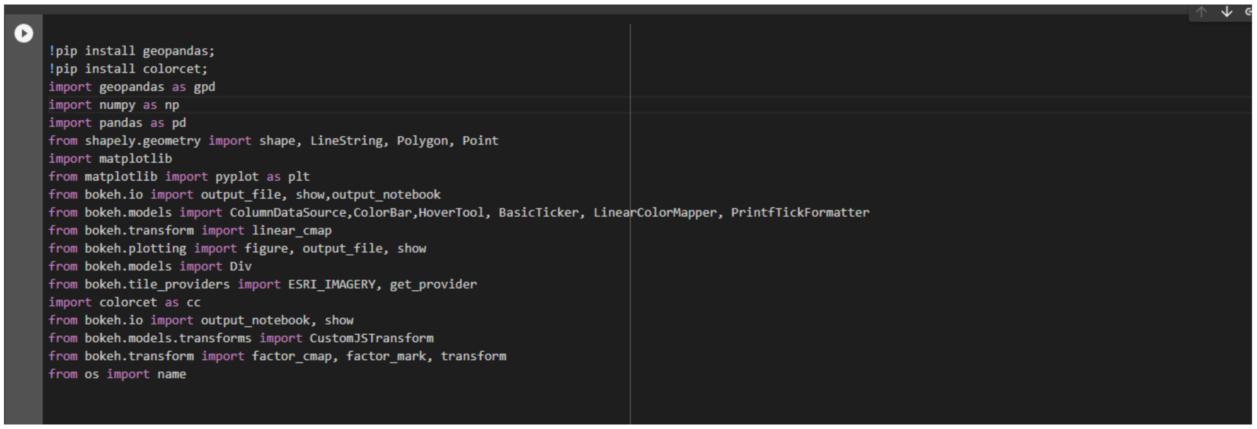


Figure 3 - ArcMap representation of Hurricane data 2016-2021

The global land-ocean temperature index was obtained off Kaggle ((Koustubhk, n.d.)) with their root source being data from NOAA. The NOAA link associated with the Kaggle file no longer directly connects you with the data page. A similar set of data can be found on NOAA here (NOAA merged land-ocean sea surface temperature dataset: NOAA Physical Sciences Laboratory). The data for the land-ocean temperature is done by using the annual mean or change in the temperature globally that year. This number is determined by the change in the long-term average. In my project I did multiply the annual mean change by 10 in order to obtain a number that is workable when compared to the rest of my data on a graph. This was done because the mean change is very small per year. Do not mistake this with our temperatures are staying the same as this number in recent years has continued to rise.

Body of Code

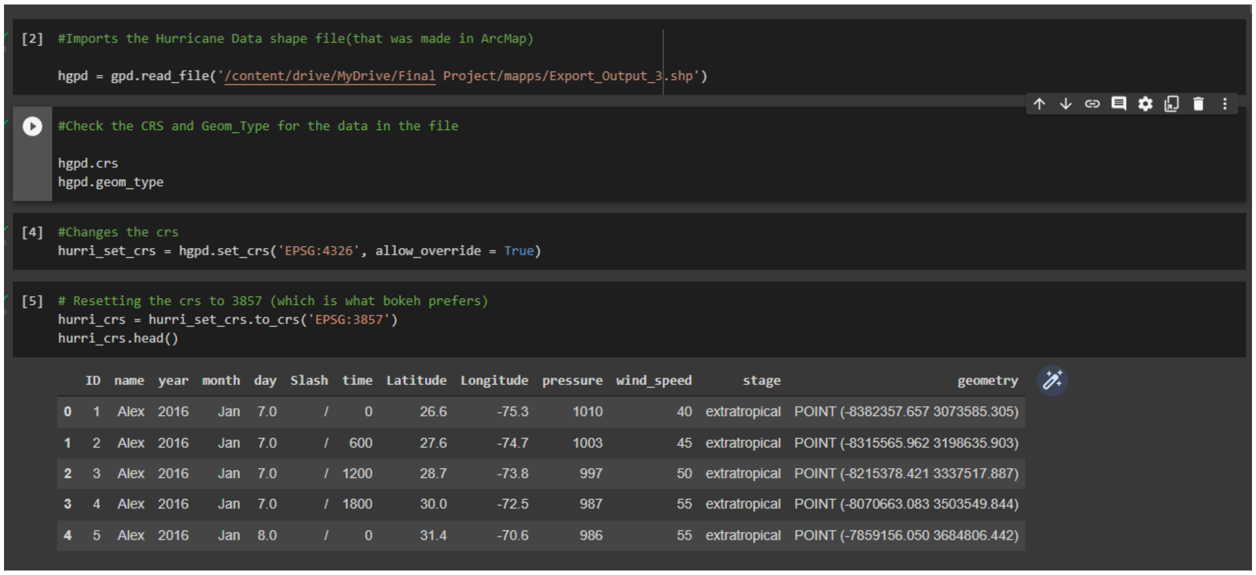
In this section I am going to explain the process and code of what I did in Google Colab to create the maps and graphs. In Figure 4, I installed and imported specific libraries needed for the project in order to execute the rest of the code throughout.



```
!pip install geopandas;
!pip install colorcet;
import geopandas as gpd
import numpy as np
import pandas as pd
from shapely.geometry import shape, LineString, Polygon, Point
import matplotlib
from matplotlib import pyplot as plt
from bokeh.io import output_file, show,output_notebook
from bokeh.models import ColumnDataSource,ColorBar,HoverTool, BasicTicker, LinearColorMapper, PrintfTickFormatter
from bokeh.transform import linear_cmap
from bokeh.plotting import figure, output_file, show
from bokeh.models import Div
from bokeh.tile_providers import ESRI_IMAGERY, get_provider
import colorcet as cc
from bokeh.io import output_notebook, show
from bokeh.models.transforms import CustomJSTransform
from bokeh.transform import factor_cmap, factor_mark, transform
from os import name
```

Figure 4 - Cell 1 of Code – Shows importation and installation of libraries

After running this cell, I imported my storm data by mounting google drive. I used GeoPandas to read the data I had stored as a shapefile. Setting the action to a variable hgpd(which stands for hurricane geopandas) and you can see this being done in Figure 5.



```
[2] #Imports the Hurricane Data shape file(that was made in ArcMap)

    hgpd = gpd.read_file('/content/drive/MyDrive/Final Project/mapps/Export_Output_3.shp')
```

```
    #Check the CRS and Geom_Type for the data in the file

    hgpd.crs
    hgpd.geom_type
```

```
[4] #Changes the crs
    hurri_set_crs = hgpd.set_crs('EPSG:4326', allow_override = True)
```

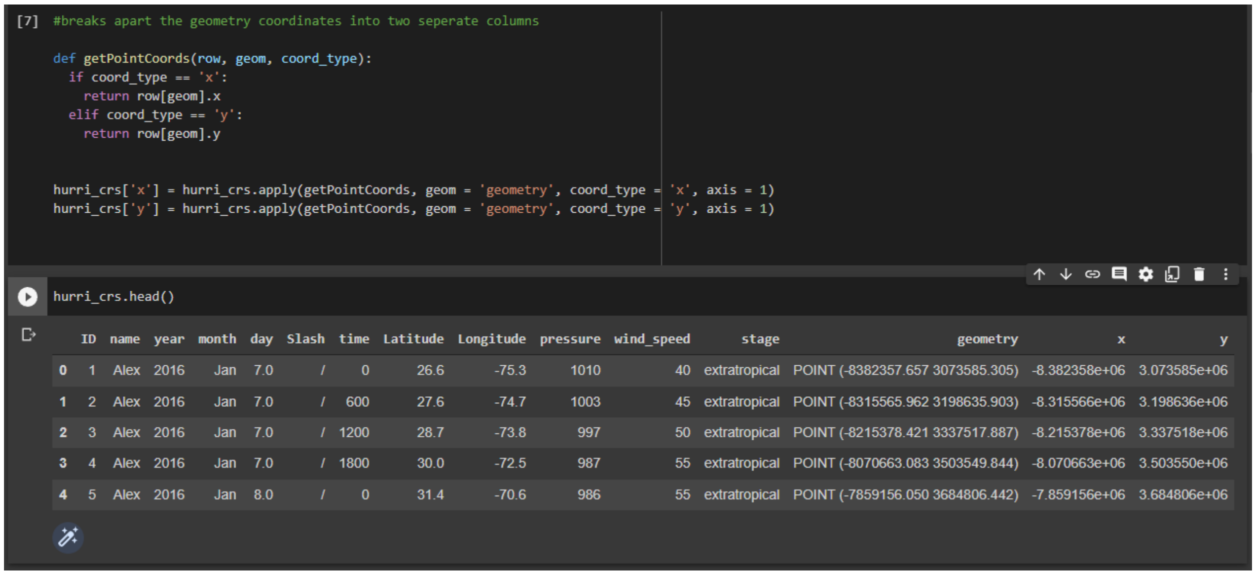
```
[5] # Resetting the crs to 3857 (which is what bokeh prefers)
    hurri_crs = hurri_set_crs.to_crs('EPSG:3857')
    hurri_crs.head()
```

	ID	name	year	month	day	Slash	time	Latitude	Longitude	pressure	wind_speed	stage	geometry
0	1	Alex	2016	Jan	7.0	/	0	26.6	-75.3	1010	40	extratropical	POINT (-8382357.657 3073585.305)
1	2	Alex	2016	Jan	7.0	/	600	27.6	-74.7	1003	45	extratropical	POINT (-8315565.962 3198635.903)
2	3	Alex	2016	Jan	7.0	/	1200	28.7	-73.8	997	50	extratropical	POINT (-8215378.421 3337517.887)
3	4	Alex	2016	Jan	7.0	/	1800	30.0	-72.5	987	55	extratropical	POINT (-8070663.083 3503549.844)
4	5	Alex	2016	Jan	8.0	/	0	31.4	-70.6	986	55	extratropical	POINT (-7859156.050 3684806.442)

Figure 5 - Reading Shape File and setting CRS

Also, in Figure 5 I am checking the geometry type of the data frame along with the CRS (coordinate referencing system). So, I set the CRS to EPSG 4326 which was the format it was originally set in when I used ArcMap and overrode any holds which allowed me to make sure it was correct. Due to bokeh not liking EPSG 4326 I then set the CRS to EPSG 3857 (Web Mercator format). This changes the geometry values to a set of values Bokeh understands. Following this I wrote a function to break apart the X and Y coordinates shown in Figure 6. This was done so later in the code I can call the X and Y for specific purposes.

This function is shown in Figure 6 which also shows the head or top 5 rows of what the data looks like.



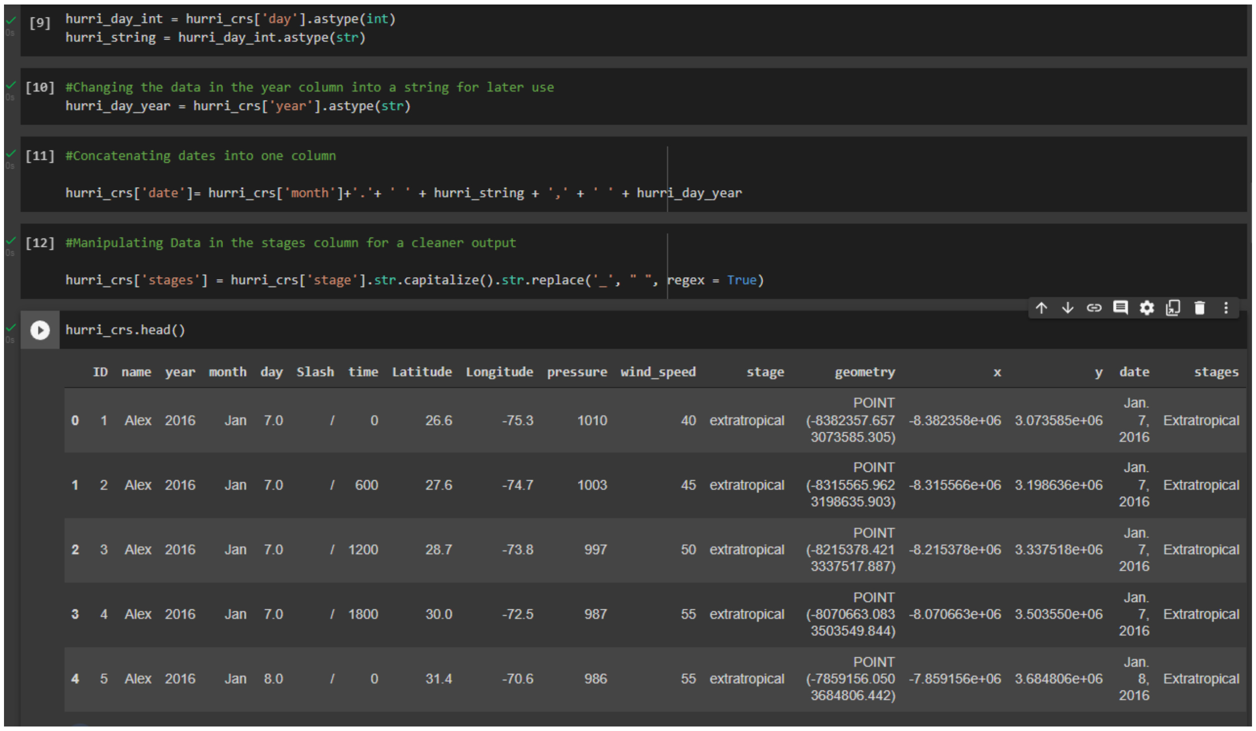
```
[7] #breaks apart the geometry coordinates into two seperate columns

    def getPointCoords(row, geom, coord_type):
      if coord_type == 'x':
        return row[geom].x
      elif coord_type == 'y':
        return row[geom].y


    hurri_crs['x'] = hurri_crs.apply(getPointCoords, geom = 'geometry', coord_type = 'x', axis = 1)
    hurri_crs['y'] = hurri_crs.apply(getPointCoords, geom = 'geometry', coord_type = 'y', axis = 1)
```

```
hurri_crs.head()
```

	ID	name	year	month	day	Slash	time	Latitude	Longitude	pressure	wind_speed	stage	geometry	x	y
0	1	Alex	2016	Jan	7.0	/	0	26.6	-75.3	1010	40	extratropical	POINT (-8382357.657 3073585.305)	-8.382358e+06	3.073585e+06
1	2	Alex	2016	Jan	7.0	/	600	27.6	-74.7	1003	45	extratropical	POINT (-8315565.962 3198635.903)	-8.315566e+06	3.198636e+06
2	3	Alex	2016	Jan	7.0	/	1200	28.7	-73.8	997	50	extratropical	POINT (-8215378.421 3337517.887)	-8.215378e+06	3.337518e+06
3	4	Alex	2016	Jan	7.0	/	1800	30.0	-72.5	987	55	extratropical	POINT (-8070663.083 3503549.844)	-8.070663e+06	3.503550e+06
4	5	Alex	2016	Jan	8.0	/	0	31.4	-70.6	986	55	extratropical	POINT (-7859156.050 3684806.442)	-7.859156e+06	3.684806e+06

Figure 6 - Function of X/Y Seperation

Figure 7 shows the manipulation of columns in the data frame: changing certain column information to strings for use later in the hover tool, the concatenation of the date information into a new column called “Date”, and the creation of a new column called “stages” in which the original data was manipulated to remove the “_” and to capitalize the first letter.

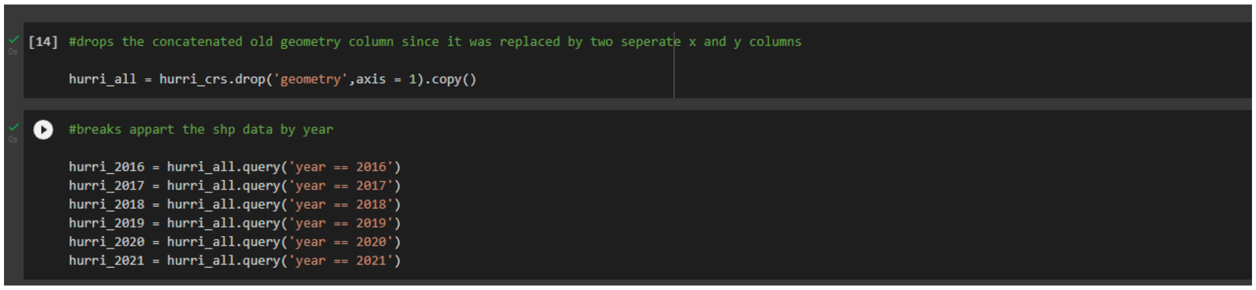


```
[9]  hurri_day_int = hurri_crs['day'].astype(int)
     hurri_string = hurri_day_int.astype(str)

[10] #Changing the data in the year column into a string for later use
     hurri_day_year = hurri_crs['year'].astype(str)

[11] #Concatenating dates into one column

     hurri_crs['date']= hurri_crs['month']+'.'+ ' ' + hurri_string + ',' + ' ' + hurri_day_year

[12] #Manipulating Data in the stages column for a cleaner output

     hurri_crs['stages'] = hurri_crs['stage'].str.capitalize().str.replace('_', " ", regex = True)

     hurri_crs.head()
```

```
   ID name  year month day Slash time Latitude Longitude pressure wind_speed stage         geometry                      x              y             date           stages
0  1  Alex  2016  Jan  7.0   /     0   26.6     -75.3     1010     40     extratropical POINT (-8382357.657 3073585.305) -8.382358e+06 3.073585e+06 Jan. 7, 2016 Extratropical
1  2  Alex  2016  Jan  7.0   /   600   27.6     -74.7     1003     45     extratropical POINT (-8315565.962 3198635.903) -8.315566e+06 3.198636e+06 Jan. 7, 2016 Extratropical
2  3  Alex  2016  Jan  7.0   /  1200   28.7     -73.8      997     50     extratropical POINT (-8215378.421 3337517.887) -8.215378e+06 3.337518e+06 Jan. 7, 2016 Extratropical
3  4  Alex  2016  Jan  7.0   /  1800   30.0     -72.5      987     55     extratropical POINT (-8070663.083 3503549.844) -8.070663e+06 3.503550e+06 Jan. 7, 2016 Extratropical
4  5  Alex  2016  Jan  8.0   /     0   31.4     -70.6      986     55     extratropical POINT (-7859156.050 3684806.442) -7.859156e+06 3.684806e+06 Jan. 8, 2016 Extratropical
```

Figure 7 - Manipulation and Creation of New Columns

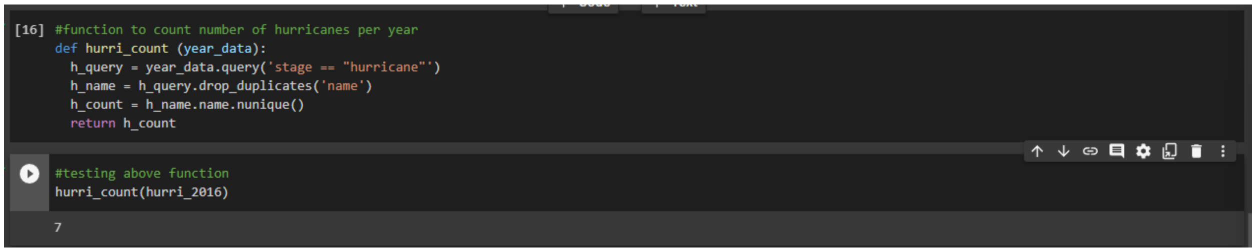
Figure 8 shows the removal of the original concatenated geometry column as it is no longer needed. Figure 8 also shows the breaking apart of the data by year. This information was assigned to variables that correlated with the respective year.



```
[14] #drops the concatenated old geometry column since it was replaced by two seperate x and y columns

     hurri_all = hurri_crs.drop('geometry',axis = 1).copy()

     #breaks appart the shp data by year

     hurri_2016 = hurri_all.query('year == 2016')
     hurri_2017 = hurri_all.query('year == 2017')
     hurri_2018 = hurri_all.query('year == 2018')
     hurri_2019 = hurri_all.query('year == 2019')
     hurri_2020 = hurri_all.query('year == 2020')
     hurri_2021 = hurri_all.query('year == 2021')
```

Figure 8 - Removal of column and manipulation of data by year.

Following this I also needed to pull out a count of individual hurricanes each year as shown in Figure 9. This function pulls out any rows that match "hurricane", but then continues to drop any that would have a duplicate name. This is done because with the data Hurricane Alex for example would be in the data frame numerous times with the stage hurricane due to the 4 times a day time-stamps. The function was created to only count individual hurricanes for each respective year. This information is later used in a plot and will be shown at the end of this report.



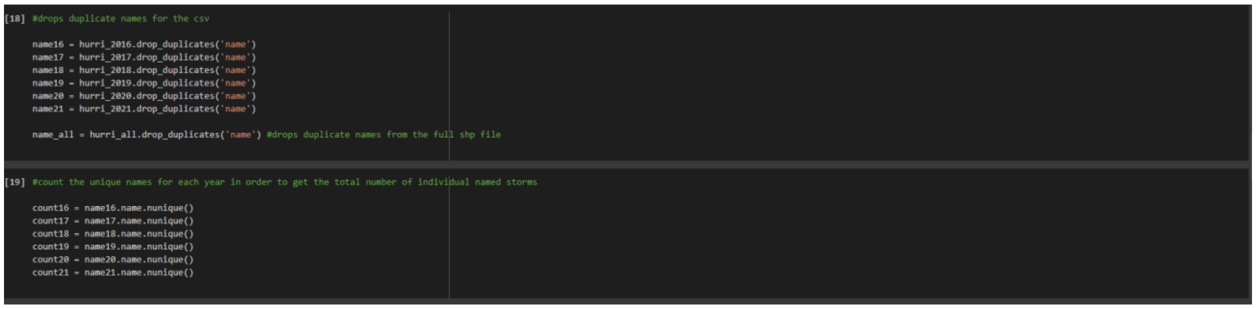
```
[16] #function to count number of hurricanes per year
     def hurri_count (year_data):
       h_query = year_data.query('stage == "hurricane"')
       h_name = h_query.drop_duplicates('name')
       h_count = h_name.name.nunique()
       return h_count
```

```
#testing above function
hurri_count(hurri_2016)
```

```
7
```

Figure 9 - Pulling out individual hurricane count based off year and name

Figure 10 shows each year and the whole data file with duplicate names being dropped without regards to any other data. These are then set to their own variable in order to be called later. Figure 10 also shows, the count or nunique() code in order to count the number of total storms per year.

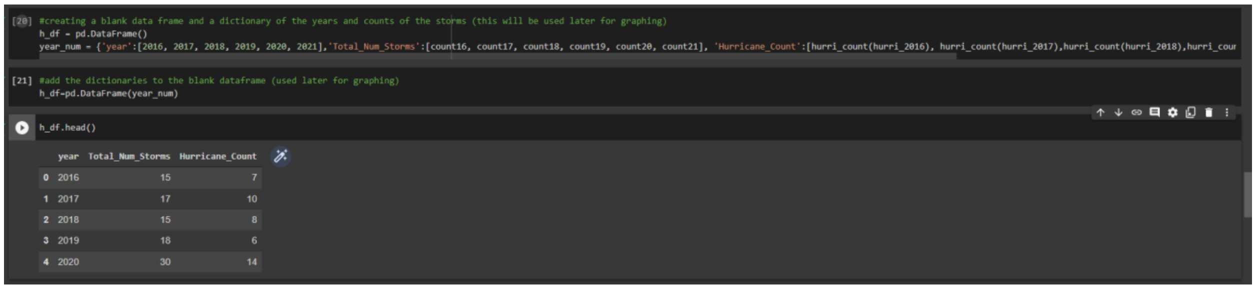


```
[18] #drops duplicate names for the csv

     name16 = hurri_2016.drop_duplicates('name')
     name17 = hurri_2017.drop_duplicates('name')
     name18 = hurri_2018.drop_duplicates('name')
     name19 = hurri_2019.drop_duplicates('name')
     name20 = hurri_2020.drop_duplicates('name')
     name21 = hurri_2021.drop_duplicates('name')

     name_all = hurri_all.drop_duplicates('name') #drops duplicate names from the full shp file
```

```
[19] #count the unique names for each year in order to get the total number of individual named storms

     count16 = name16.name.nunique()
     count17 = name17.name.nunique()
     count18 = name18.name.nunique()
     count19 = name19.name.nunique()
     count20 = name20.name.nunique()
     count21 = name21.name.nunique()
```

Figure 10 - Duplicate name drop and unique filtering

Figure 11 is making a new pandas data frame and creating a dictionary. This information will be used for the plot map. I only needed 3 points of arguments from the storm data.



```
[20] #creating a blank data frame and a dictionary of the years and counts of the storms (this will be used later for graphing)
     h_df = pd.DataFrame()
     year_num = {'year':[2016, 2017, 2018, 2019, 2020, 2021],'Total_Num_Storms':[count16, count17, count18, count19, count20, count21], 'Hurricane_Count':[hurri_count(hurri_2016), hurri_count(hurri_2017),hurri_count(hurri_2018),hurri_cour
```

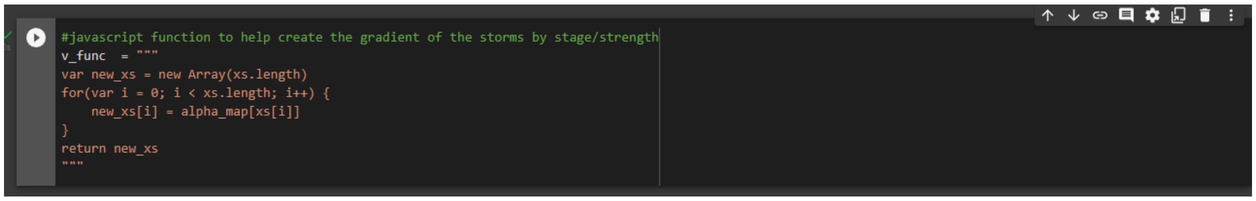
```
[21] #add the dictionaries to the blank dataframe (used later for graphing)
     h_df=pd.DataFrame(year_num)
```

```
h_df.head()
```

	year	Total_Num_Storms	Hurricane_Count
0	2016	15	7
1	2017	17	10
2	2018	15	8
3	2019	18	6
4	2020	30	14

Figure 11 - New Data Frame Creation for Plot

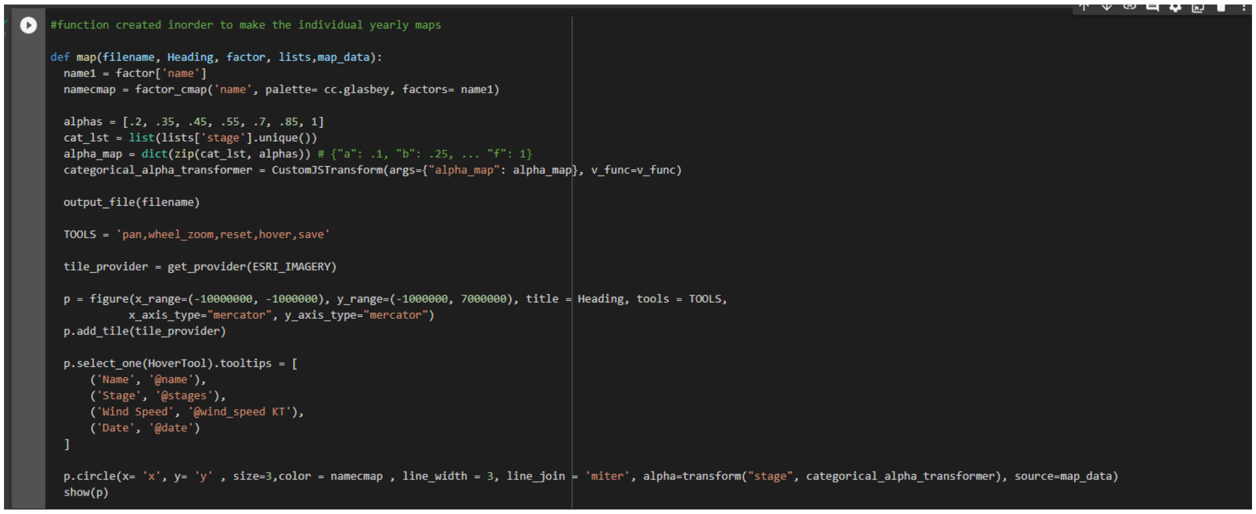
In the creation of my code, I realized I wanted to have a form of gradient associated with each storm so you could have a visual representation of when the storms changed stages throughout its storm path. Figure 12 shows a few lines of JavaScript which is later called in when creating the map which will be shown partially in the next figure.



```
#javascript function to help create the gradient of the storms by stage/strength
v_func  = """
var new_xs = new Array(xs.length)
for(var i = 0; i < xs.length; i++) {
    new_xs[i] = alpha_map[xs[i]]
}
return new_xs
"""
```

Figure 12 - Javascript

Figure ** is a function that is created to make each yearly map. I am going to break down this figure into more detailed steps.

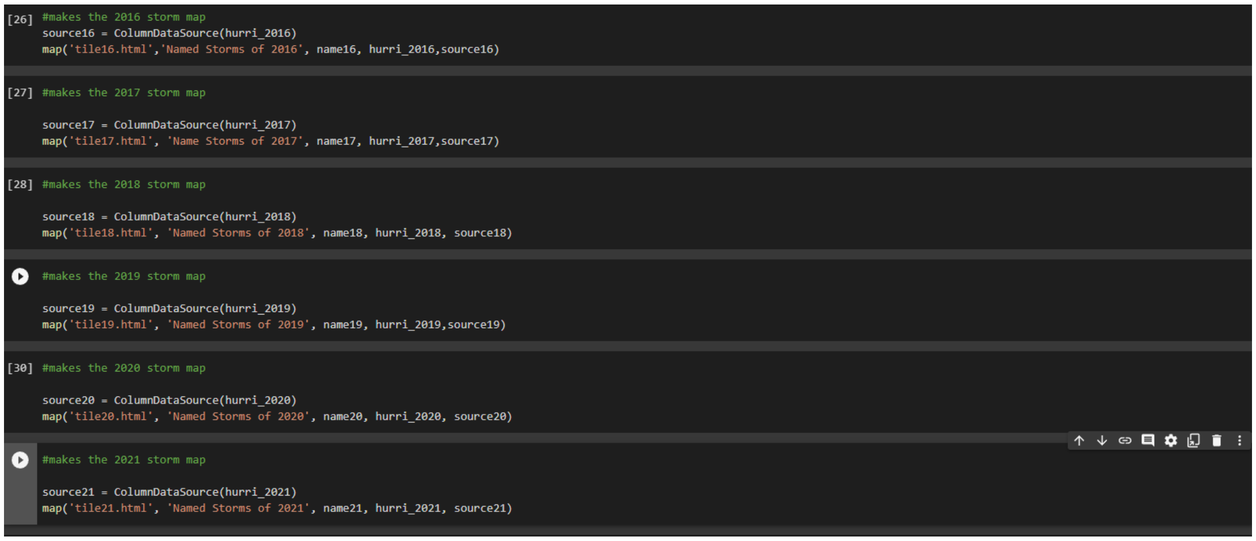


```
#function created inorder to make the individual yearly maps

def map(filename, Heading, factor, lists,map_data):
  name1 = factor['name']
  namecmap = factor_cmap('name', palette= cc.glasbey, factors= name1)

  alphas = [.2, .35, .45, .55, .7, .85, 1]
  cat_lst = list(lists['stage'].unique())
  alpha_map = dict(zip(cat_lst, alphas)) # {"a": .1, "b": .25, ... "f": 1}
  categorical_alpha_transformer = CustomJSTransform(args={"alpha_map": alpha_map}, v_func=v_func)

  output_file(filename)

  TOOLS = 'pan,wheel_zoom,reset,hover,save'

  tile_provider = get_provider(ESRI_IMAGERY)

  p = figure(x_range=(-10000000, -1000000), y_range=(-1000000, 7000000), title = Heading, tools = TOOLS,
            x_axis_type="mercator", y_axis_type="mercator")
  p.add_tile(tile_provider)

  p.select_one(HoverTool).tooltips = [
      ('Name', '@name'),
      ('Stage', '@stages'),
      ('Wind Speed', '@wind_speed KT'),
      ('Date', '@date')
  ]

  p.circle(x= 'x', y= 'y' , size=3,color = namecmap , line_width = 3, line_join = 'miter', alpha=transform("stage", categorical_alpha_transformer), source=map_data)
  show(p)
```

Figure 13 - Individual Map Function

The function itself will have 5 variables that need to be passed in for the function itself to work. The top two lines are used for creating a large color spectrum (due to the many variables) using the glasbey palette from Colorcet which was imported at the beginning of the code. Lines 5-8 are tied to the JavaScript form Figure 12, this creates the alpha gradient based off the "stage" of the storm. The next few lines show the output_file, bokeh tools that are going to be used, and the tile used which was pulled in from Bokeh and ESRI imagery. The rest of the code in figure 13 is the actual map figure. Parts of this code will refer to variables that will be typed in part of the function, so it is all customizable such as the map title and source.

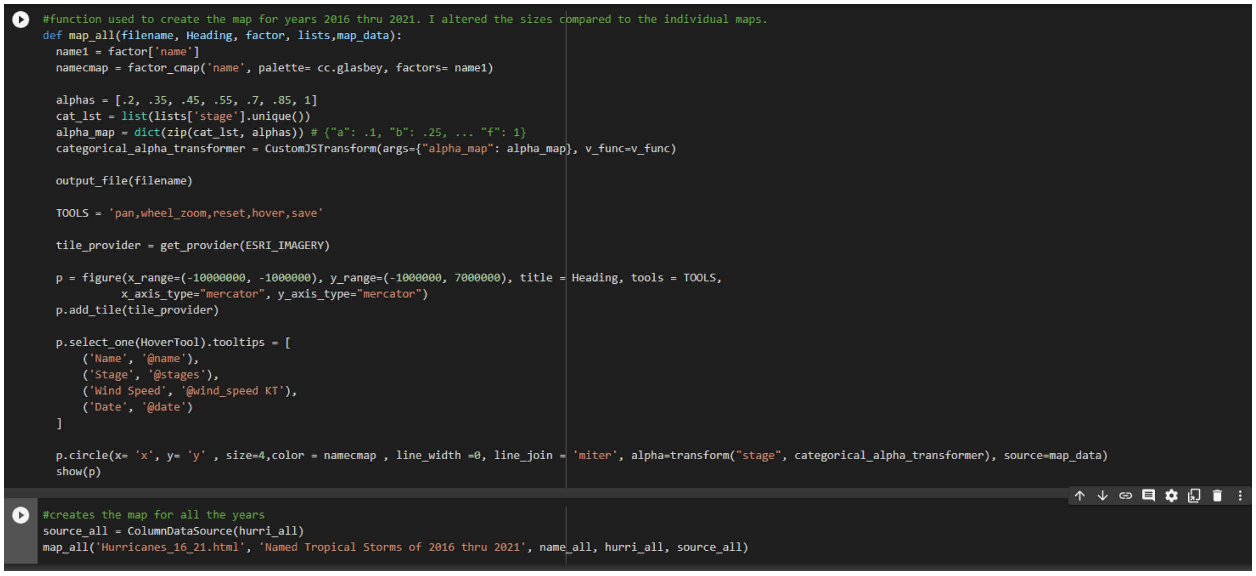
Figure 14 is showing the map function being run for each individual year which is also generating each map html file.



```
[26] #makes the 2016 storm map
     source16 = ColumnDataSource(hurri_2016)
     map('tile16.html','Named Storms of 2016', name16, hurri_2016,source16)

[27] #makes the 2017 storm map

     source17 = ColumnDataSource(hurri_2017)
     map('tile17.html', 'Name Storms of 2017', name17, hurri_2017,source17)

[28] #makes the 2018 storm map

     source18 = ColumnDataSource(hurri_2018)
     map('tile18.html', 'Named Storms of 2018', name18, hurri_2018, source18)

     #makes the 2019 storm map

     source19 = ColumnDataSource(hurri_2019)
     map('tile19.html', 'Named Storms of 2019', name19, hurri_2019,source19)

[30] #makes the 2020 storm map

     source20 = ColumnDataSource(hurri_2020)
     map('tile20.html', 'Named Storms of 2020', name20, hurri_2020, source20)

     #makes the 2021 storm map

     source21 = ColumnDataSource(hurri_2021)
     map('tile21.html', 'Named Storms of 2021', name21, hurri_2021, source21)
```

Figure 14 - Calling the Map Function

Figure 15 is very similar to figure 13 but with some minor modifications to the sizes of the symbology due to this function being used to generate a map of all the storm years combined. Also, in this figure you will the function being called in order to generate the overall map.

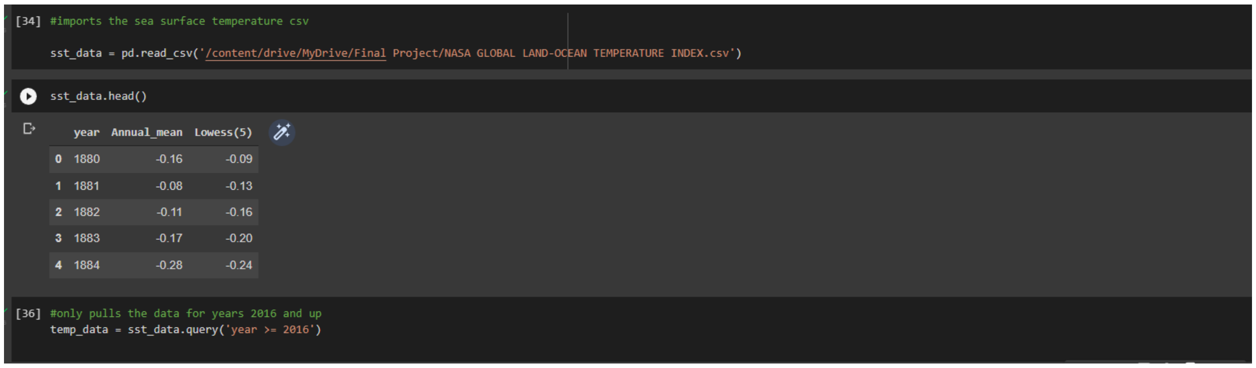


```
#function used to create the map for years 2016 thru 2021. I altered the sizes compared to the individual maps.
def map_all(filename, Heading, factor, lists,map_data):
  name1 = factor['name']
  namecmap = factor_cmap('name', palette= cc.glasbey, factors= name1)

  alphas = [.2, .35, .45, .55, .7, .85, 1]
  cat_lst = list(lists['stage'].unique())
  alpha_map = dict(zip(cat_lst, alphas)) # {"a": .1, "b": .25, ... "f": 1}
  categorical_alpha_transformer = CustomJSTransform(args={"alpha_map": alpha_map}, v_func=v_func)

  output_file(filename)

  TOOLS = 'pan,wheel_zoom,reset,hover,save'

  tile_provider = get_provider(ESRI_IMAGERY)

  p = figure(x_range=(-10000000, -1000000), y_range=(-1000000, 7000000), title = Heading, tools = TOOLS,
          x_axis_type="mercator", y_axis_type="mercator")
  p.add_tile(tile_provider)

  p.select_one(HoverTool).tooltips = [
      ('Name', '@name'),
      ('Stage', '@stages'),
      ('Wind Speed', '@wind_speed KT'),
      ('Date', '@date')
  ]

  p.circle(x= 'x', y= 'y' , size=4,color = namecmap , line_width =0, line_join = 'miter', alpha=transform("stage", categorical_alpha_transformer), source=map_data)
  show(p)
```

```
#creates the map for all the years
source_all = ColumnDataSource(hurri_all)
map_all('Hurricanes_16_21.html', 'Named Tropical Storms of 2016 thru 2021', name_all, hurri_all, source_all)
```

Figure 15 - 2016-2021 Map Function

I am now at the point in the code where I am importing the land-ocean temperature data. This data is imported using pandas as there is no need for the geometry of GeoPandas. Figure 16 shows the import, the head (top 5 rows), and the filtering of the data in the data frame to only years greater than or equal to 2016. This is so the land-ocean temperature data matches the data from the storms.



```
[34] #imports the sea surface temperature csv

     sst_data = pd.read_csv('/content/drive/MyDrive/Final Project/NASA GLOBAL LAND-OCEAN TEMPERATURE INDEX.csv')
```

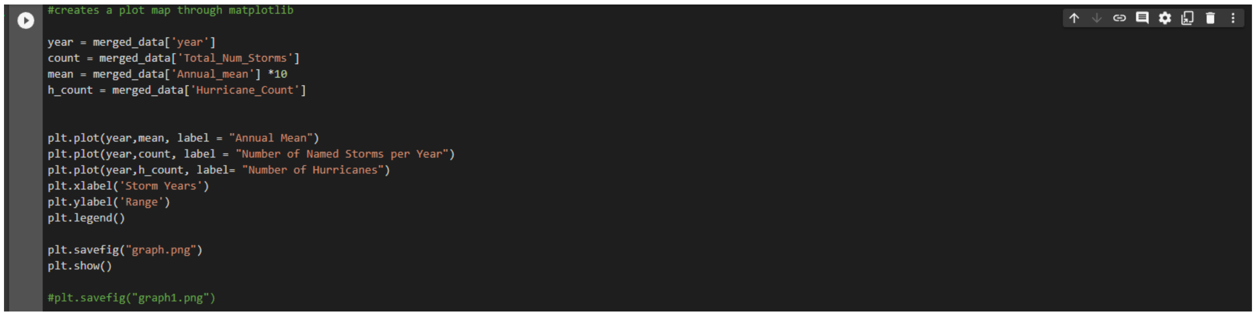
```
sst_data.head()
```

```
   year  Annual_mean  Lowess(5)
0  1880        -0.16      -0.09
1  1881        -0.08      -0.13
2  1882        -0.11      -0.16
3  1883        -0.17      -0.20
4  1884        -0.28      -0.24
```

```
[36] #only pulls the data for years 2016 and up
     temp_data = sst_data.query('year >= 2016')
```

Figure 16 - Import of Land-Ocean Temp Data

Figure ** is the code used to create the plot through MatPlotLib. In this code I am plotting the Annual Mean (which was multiplied by 10 as stated previously), the number of named storms per year, the number of individual hurricanes each year. The X axis is the storm years, and the Y axis is the range of data values.



```
#creates a plot map through matplotlib

year = merged_data['year']
count = merged_data['Total_Num_Storms']
mean = merged_data['Annual_mean'] *10
h_count = merged_data['Hurricane_Count']


plt.plot(year,mean, label = "Annual Mean")
plt.plot(year,count, label = "Number of Named Storms per Year")
plt.plot(year,h_count, label= "Number of Hurricanes")
plt.xlabel('Storm Years')
plt.ylabel('Range')
plt.legend()

plt.savefig("graph.png")
plt.show()

#plt.savefig("graph1.png")
```

Figure 17 - Plot Generation Code

Data Output and Imagery

This section of the project is reserved for the map imagery and graph. To get the full experience from this project I have included each HTML file with submission of this project.

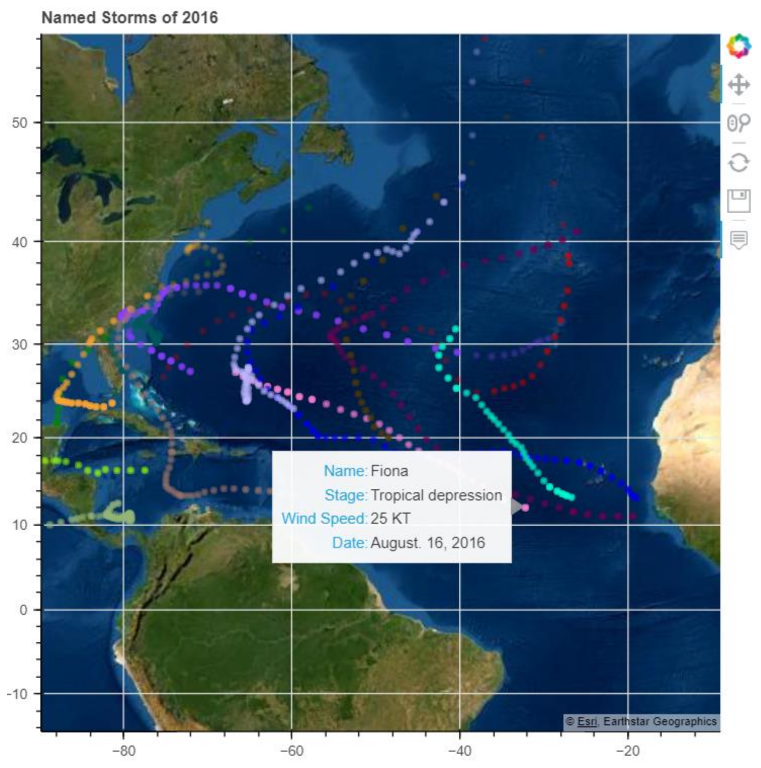


Figure 18 - 2016 Map

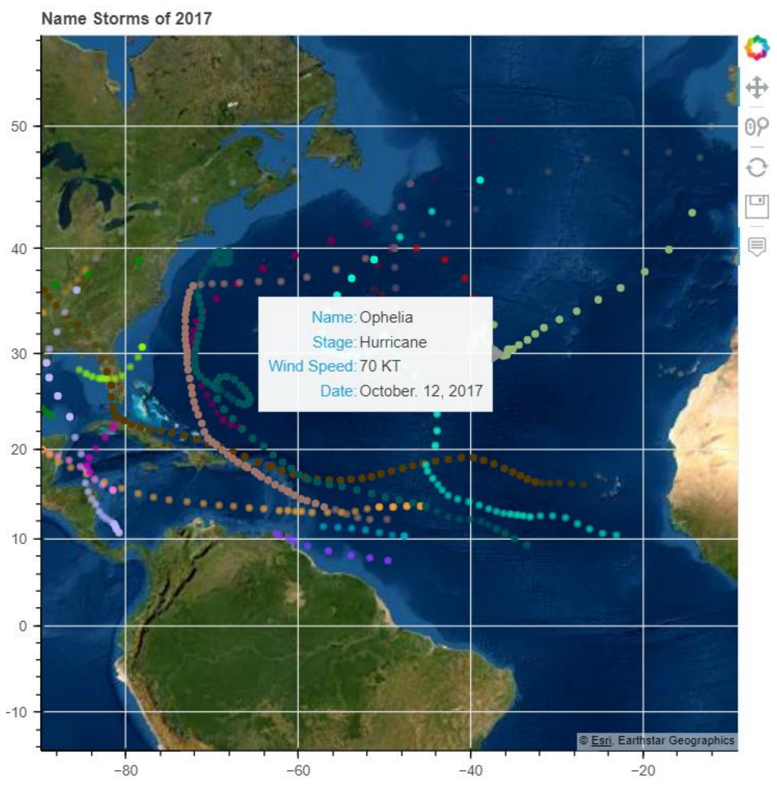


Figure 19 – 2017 Map

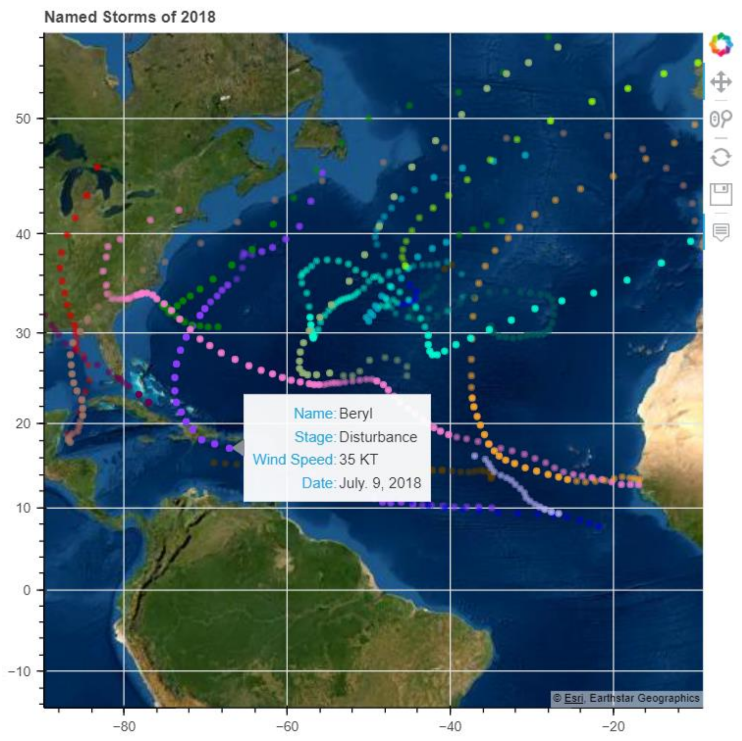


Figure 20 – 2018 Map

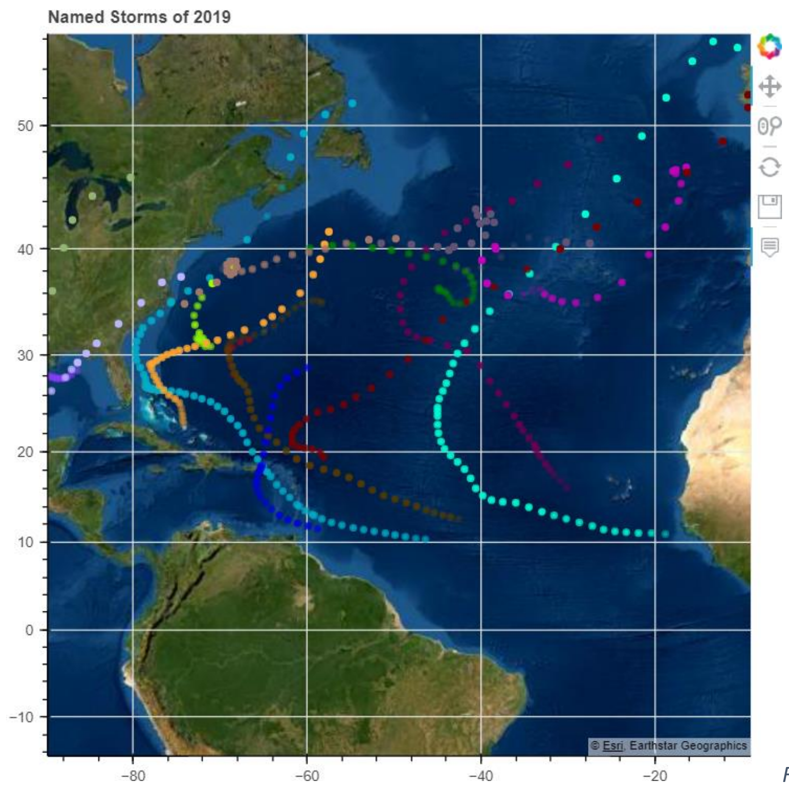


Figure 21 – 2019 Map

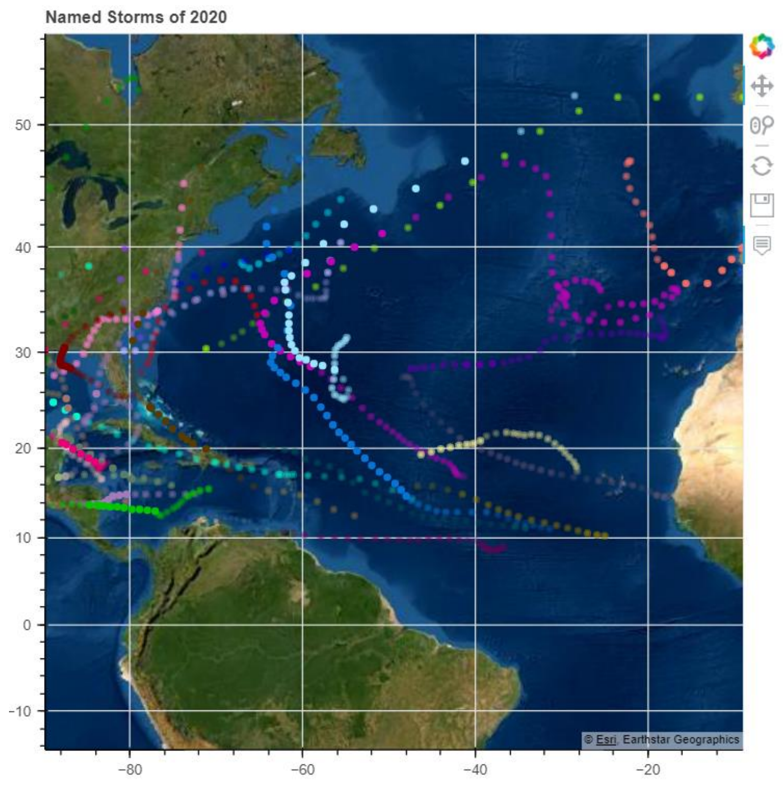


Figure 22 – 2020 Map

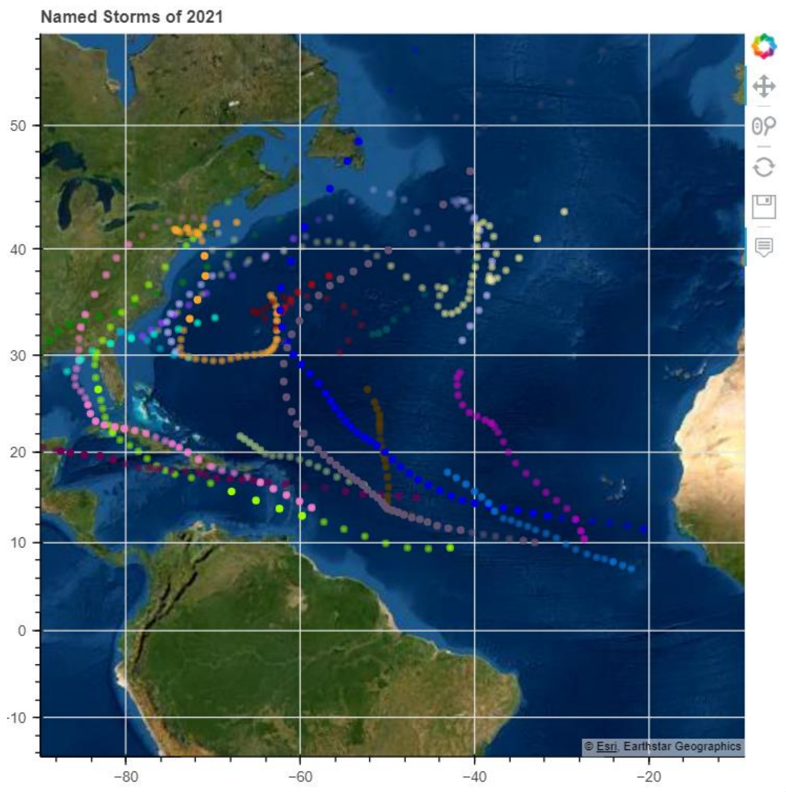


Figure 23 - 2021 Map

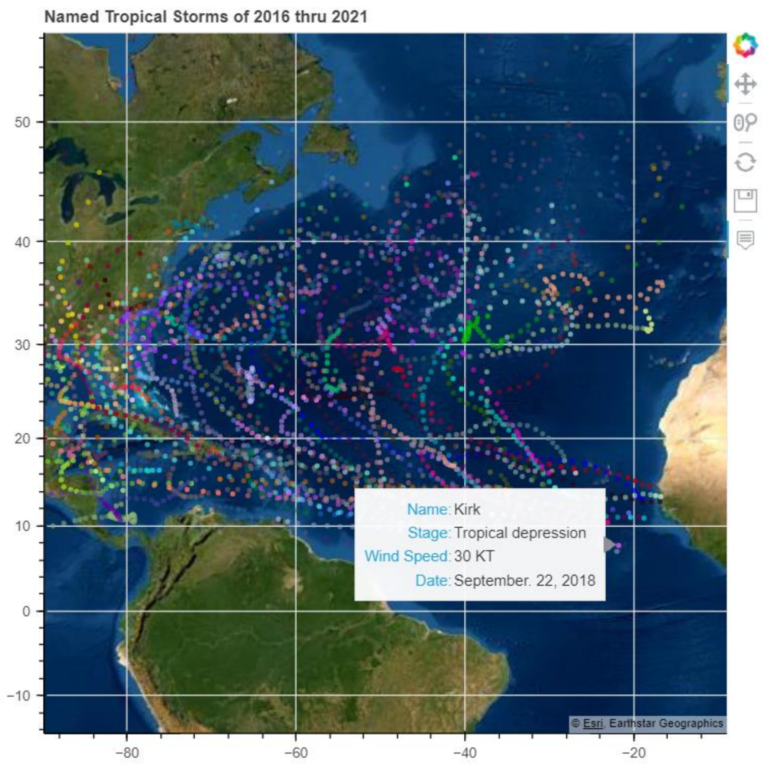

Named Tropical Storms of 2016 thru 2021
50
40
30
20
10
0
−10
−80
−60
−40
−20
Name: Kirk
Stage: Tropical depression
Wind Speed: 30 KT
Date: September. 22, 2018
© Esri, Earthstar Geographics

Figure 24 - All Storms 2016 -2021

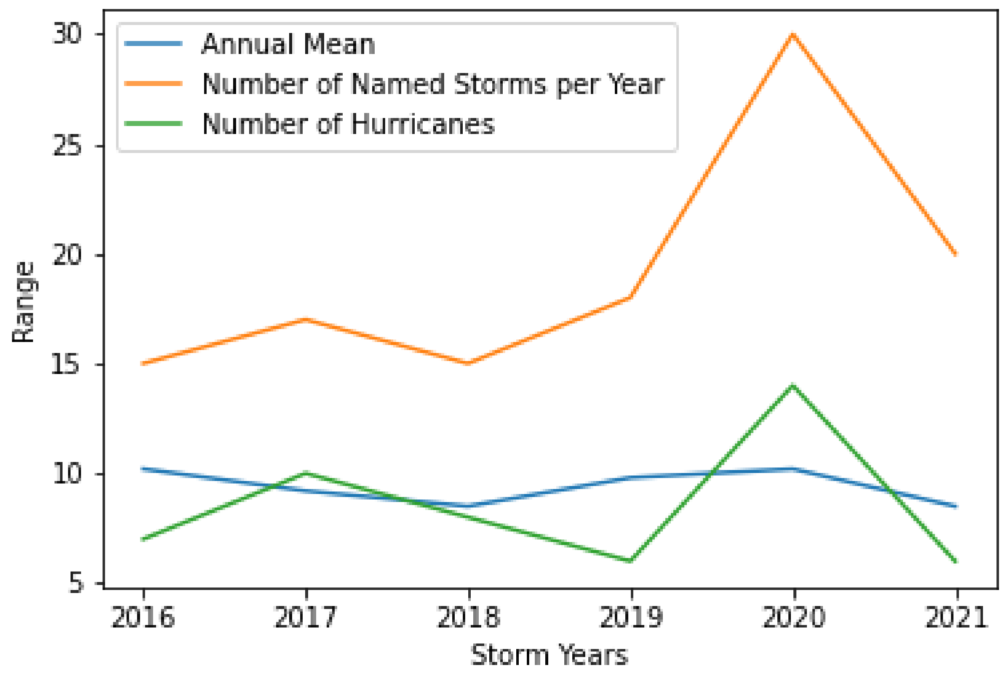


Figure 25 - Plot Map

Conclusion

As this project progressed my conclusion was known before, I even finished. The results in the end are inconclusive due to lack of data span. In order to see if there is a more relevant pattern or correlation between what is essentially sea surface temperature and tropical storms one must need a much broader range of data. This being said, I fully intend to continue compiling data from the National Hurricane Center to see if there is some kind of relevant correlation. You can see in the year of 2020 we had a large amount of named storms and a relevant peak in the annual mean of land-ocean temperature. This seems promising but if you look back to the year 2017 you would see the opposite affect where the Annual Mean is declining from the previous year. I hope in the future the year of 2020 and the amount and strength of storms associated with it will stay a strange anomaly of weather phenomena. I hope more information comes to light from the scientific community in regard to the impacts of climate change on weather and natural disasters in order to better prepare society for what may be yet to come.

References

Koustubhk. (n.d.). *Global Land-Ocean Temperature Index - 1880 to 2020*. Retrieved from Kaggle: https://www.kaggle.com/datasets/kkhandekar/global-landocean-temperature-index

National Oceanic and Atmospheric Administration. (n.d.). *National Hurricane Center and Central Pacific Hurricane Center*. Retrieved from NHC: https://www.nhc.noaa.gov/data/tcr/index.php?season=2008&basin=atl