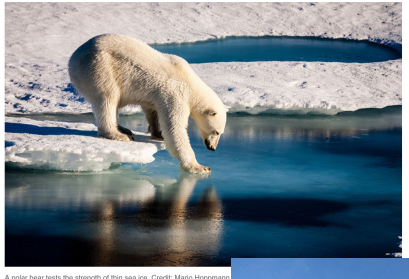
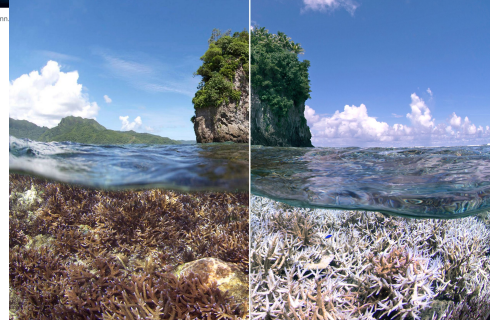
The Effects of Power Plants on Greenhouse Gas Emission within the US



A polar bear tests the strength of thin sea ice. Credit: Mario Hoppmann



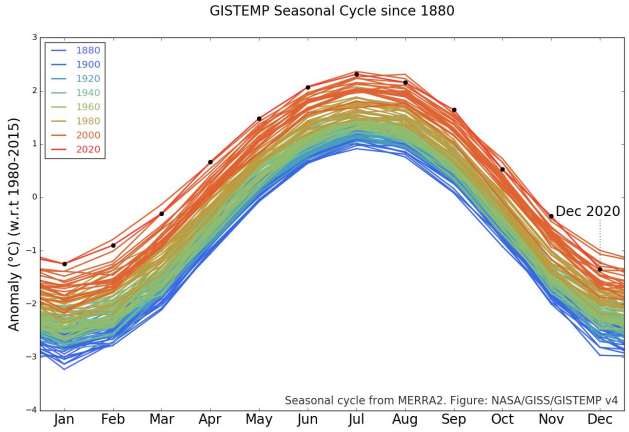
Natalia Karimova , Enoch Kwon, Jeremy Jackson, Parampreet Khanna, William Pryor

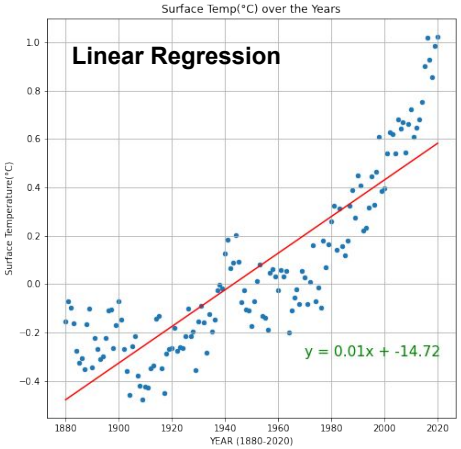
Is it actually getting hotter?

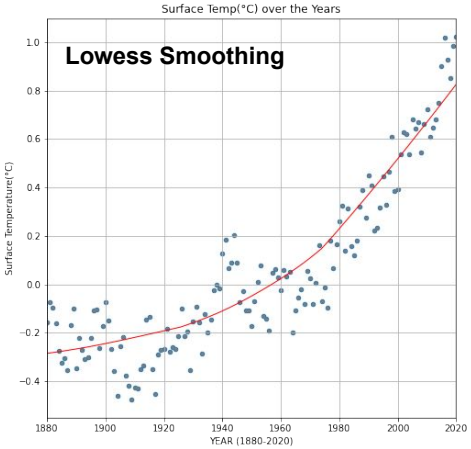
National Aeronautics and Space Administration
Goddard Institute for Space Studies

NASA GISS Surface Temp (1880 - 2020)*

seaborn







- Temperature analysis (*MERRA-2*) over 1980-2015 shown how much warmer each month data is than the annual global mean.
- We can see significant temperature rise in last 200 years (*GISTEMP v4*).
- Is it natural or anthropogenic?

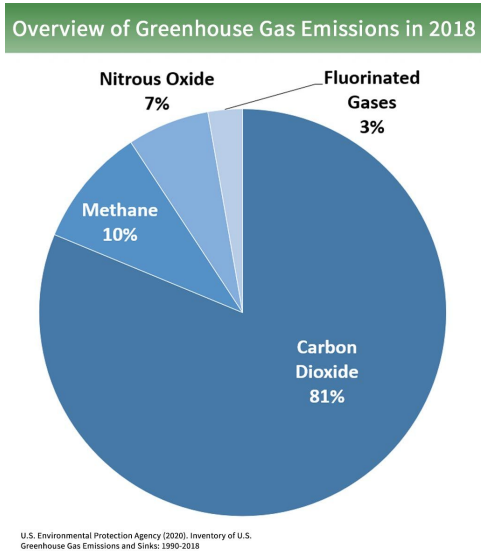
*https://data.giss.nasa.gov/gistemp/

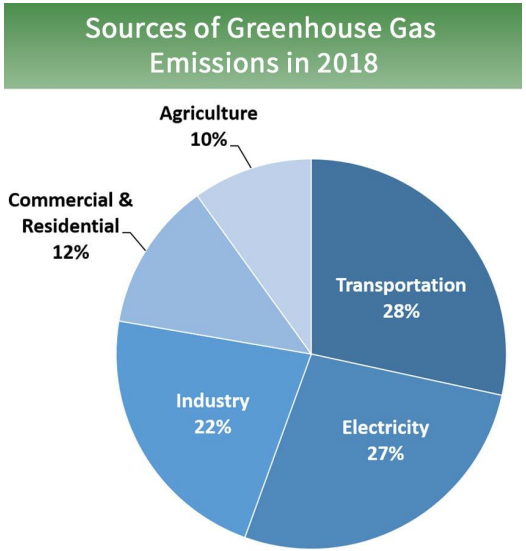
Motivation

According to the United States Environmental Protection Agency:

https://www.epa.gov/ghgemissions/inventory-us-greenhouse-gas-emissions-and-sinks

- Carbon dioxide(CO_2) is the main Greenhouse Gas (81%);
- Two main sources of these gases: Transportation and Electricity





Project Questions:

The effects of the power plants on CO2 emission within the US:

- Is it actually getting hotter?
- How many electric power plants (EPP) are in the US?
- What types of EPP are in US?
- What is the most popular type of EPP in the US?
- What types of Fossil EPP are in the US?
- What Fossil EPP type is the most efficient and clean: provides more energy and less CO2?

Data Collection

Sources:

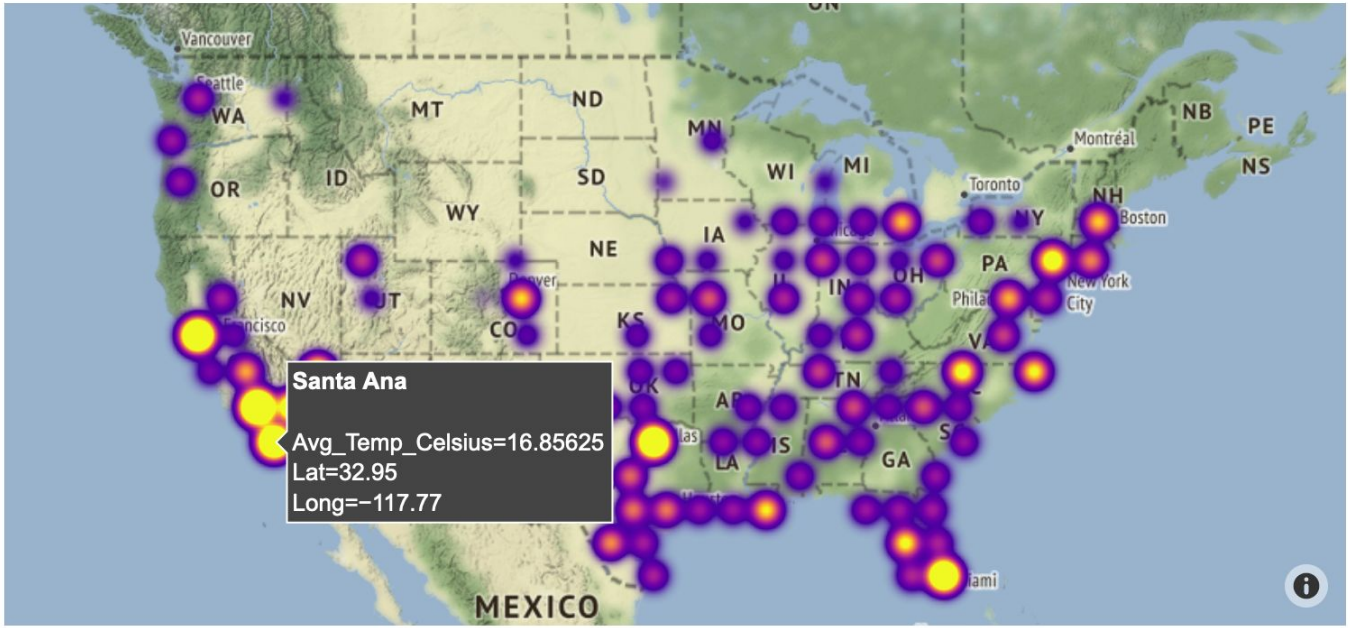
World Surface Temp -> US Surface Temperature Data 1750-2018:
https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data

USA Power Plants:
https://hifld-geoplatform.opendata.arcgis.com/datasets/power-plants/data

CO2 Emission Data:
https://www.eia.gov/electricity/data/emissions/

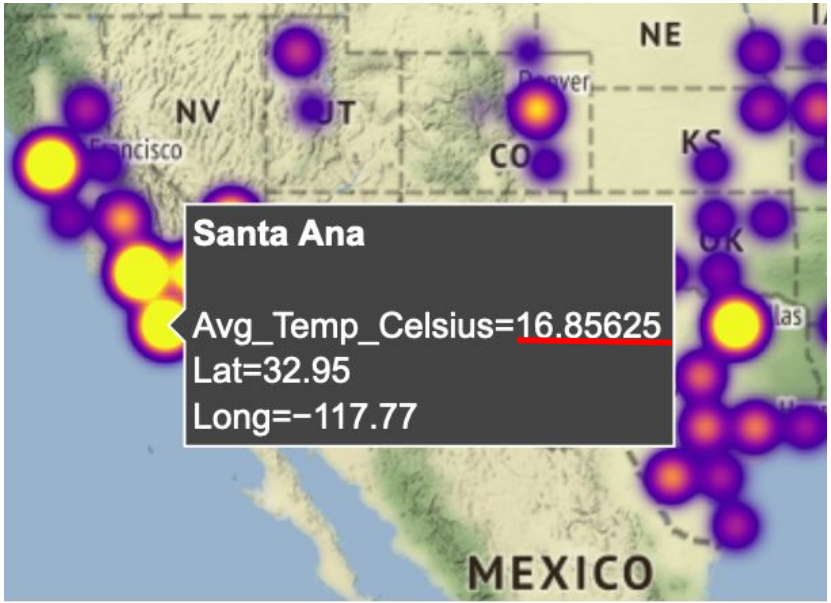
Average Yearly Temperature (°C) (1993-2013)

1993

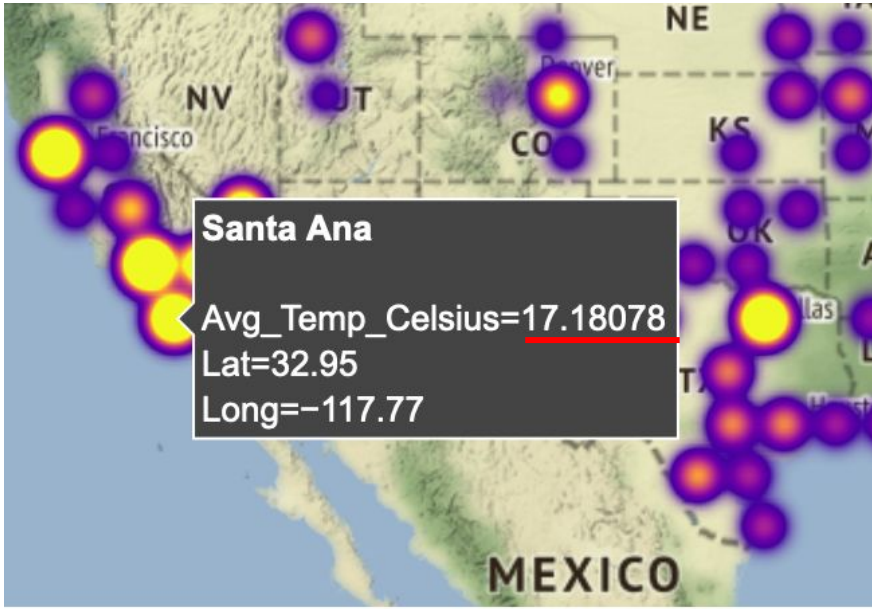


Results: Heat Map

1993 vs. 2013 Orange County (Santa Ana) Avg. Temp (°C)

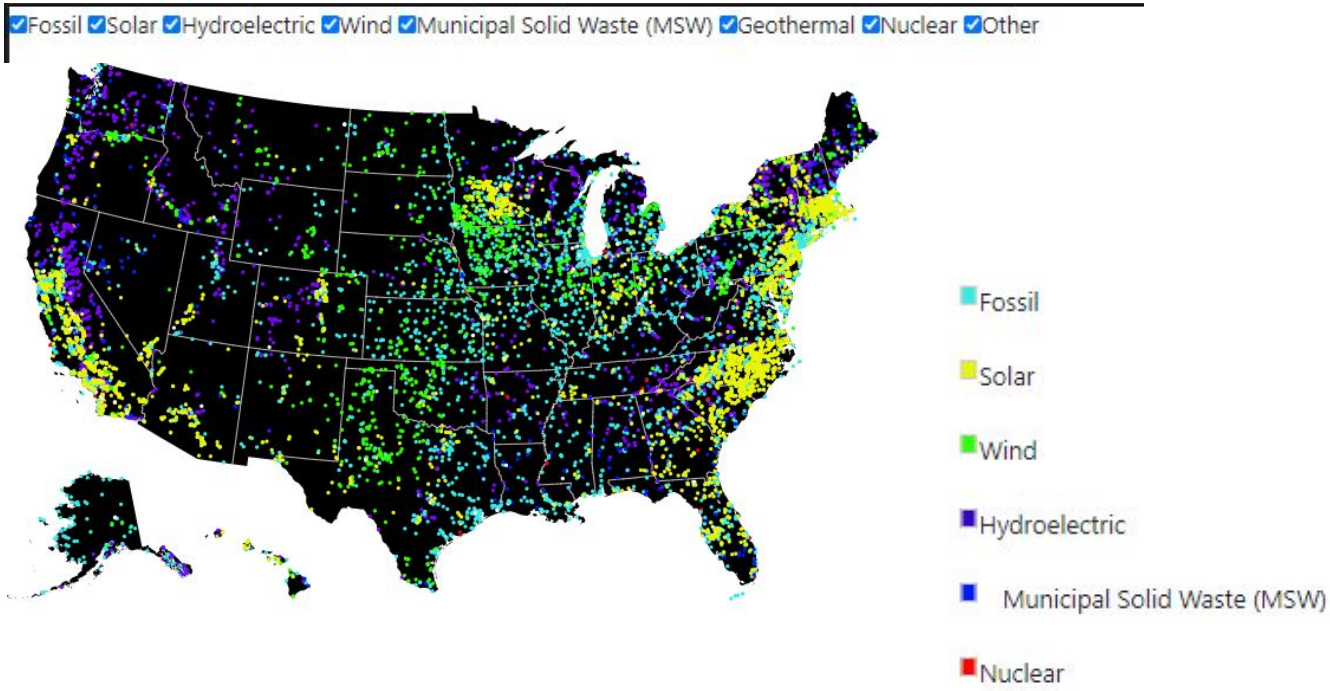


1993

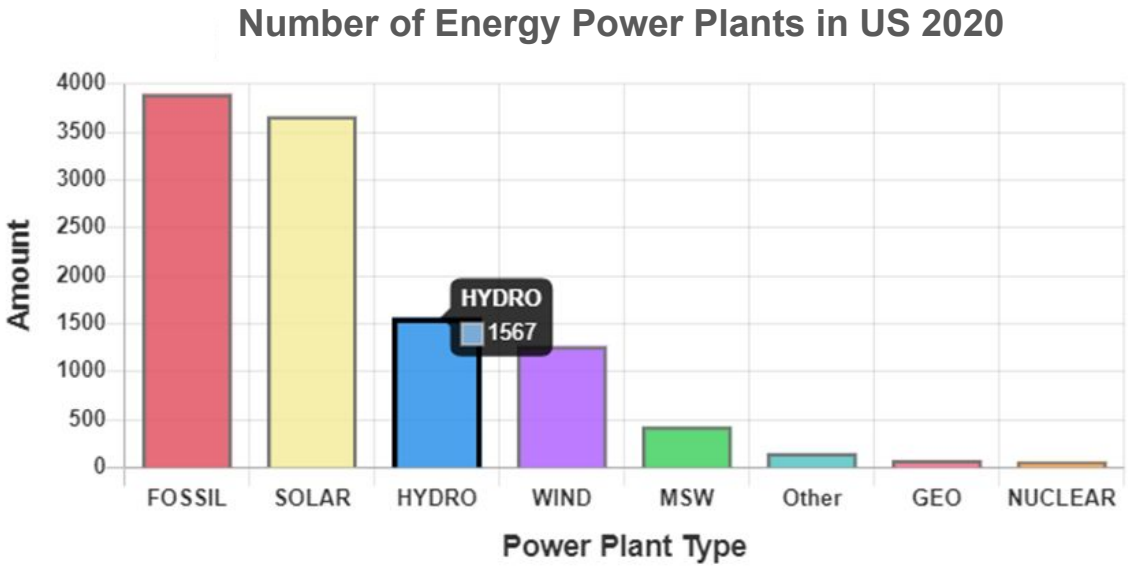


2013

USA Power Plant Map



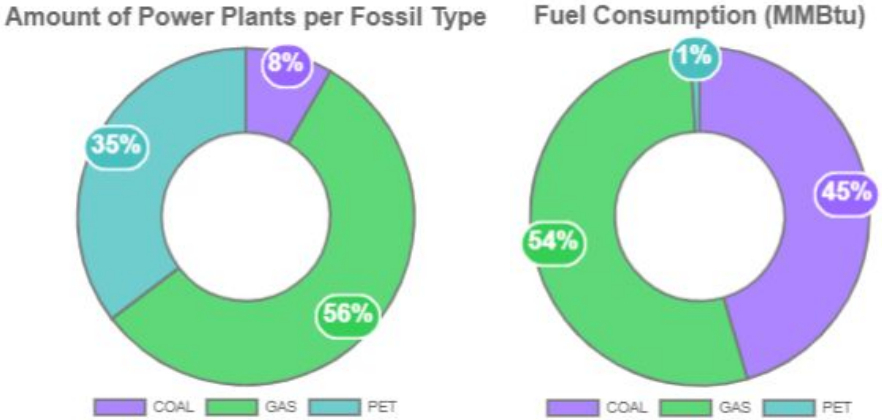
USA Power Plant: Chart.js library

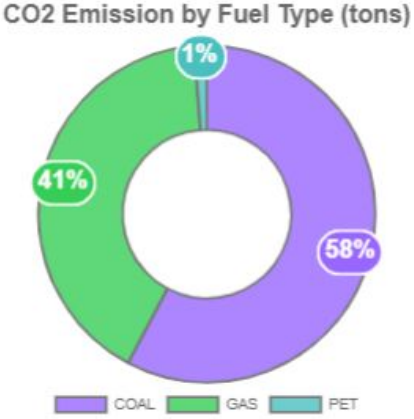


- As of July 09, 2020, there were 11,810 electric power plants in the United States.
- Fossil type of Power Plants is the most popular in US (~4000)

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.min.js"></script>
```

USA Power Plant - CO2 Emission: Chart.js library





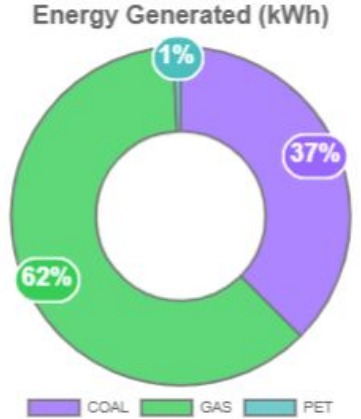


Table 1. Efficiency of Fossil Fuels

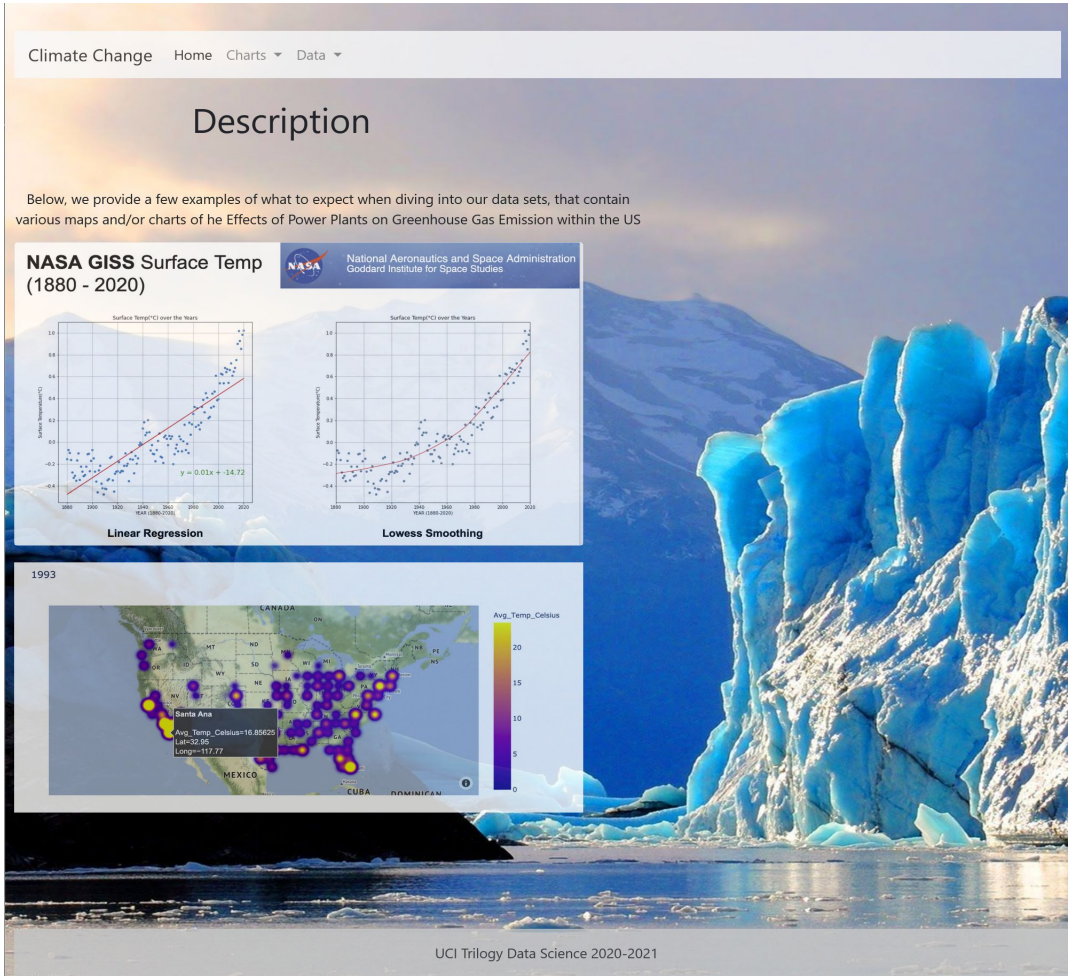
	Energy (kWh) / 1 MMBtu of fuel	CO2 (tones) / 1 MMBtu of fuel
COAL	95	0.11
GAS	133	0.06
PET	93	0.11

Natural gas: The cleanest burning fossil fuel: More efficient and produces less CO2.

<script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-datalabels"></script>

Web-site Scripting

- Utilized Flask to power our webpage.
- Imported python libraries such as:
  - Render_template
  - Url_for
  - Jinja2



General Conclusions:

- Fossil Fuel Power Plants are the most common in the U.S.
- Gas Power Plants are more efficient and clean type of the Fossil EPP. They provide more energy and less CO2, while coal power plants polluted the most
- More solar and wind power plants should slow the rate that the temperature is increasing

Future Plans:

1) If we had more time we would like to include **more years** to see the evaluation of CO2 emission from the **all types of EPP**. (Currently we have data only for Fossil EPP).

2) Additionally, we want to include other Greenhouse gases such as CH4 and NOx.

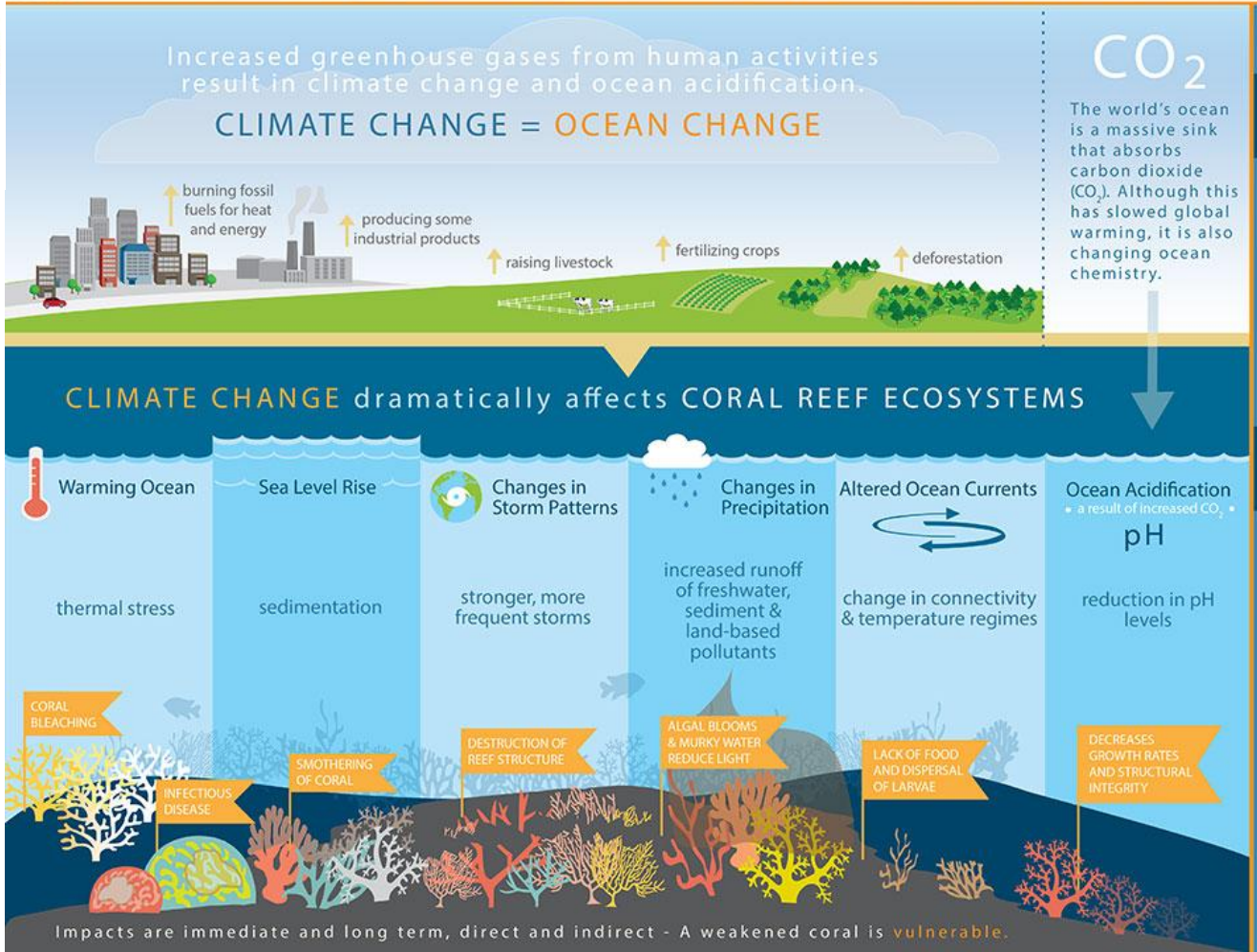
3) Electric Power Plants are not the only source of Greenhouse gases. Other sources should be considered: transportation, industry, agriculture etc.

Project Requirements

Python Flask	+
HTML/CSS web-page	+
JavaScript	+
MongoDB	+
Chart.js library	+
Large Data Sets	+
Web page have some user-drive interactions: ● Menu ● Dropdowns ● Chart Interaction ● Textboxes	+

Supporting Information:

THREATS TO CORAL REEFS CLIMATE CHANGE



HOW YOU CAN HELP

Shrink your carbon footprint to reduce greenhouse gases.

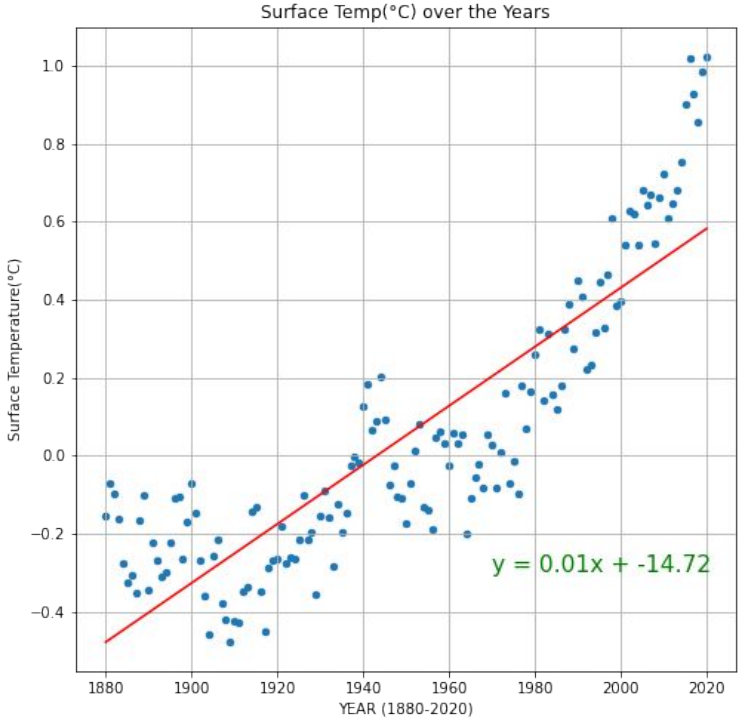
- Drive less.
- Reduce, reuse or recycle.
- Purchase energy-efficient appliances and lightbulbs.
- Print less. Download more. Use less water.

Do your part to help improve overall coral reef condition.

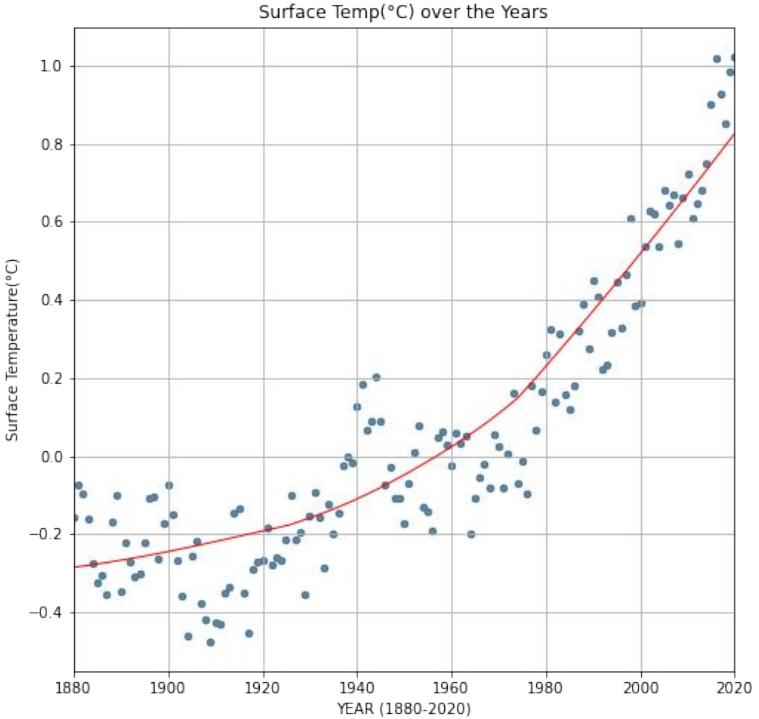
- Reduce the use of lawn and garden chemicals.
- DO NOT dump household chemicals in storm drains.
- Choose sustainable seafood. www.FishWatch.gov
- Learn about good reef etiquette and practice it when in the water.
- Volunteer for beach and waterway clean ups.

NASA GISS Surface Temp (1880 - 2020)

National Aeronautics and Space Administration
Goddard Institute for Space Studies



Linear Regression



Lowess Smoothing

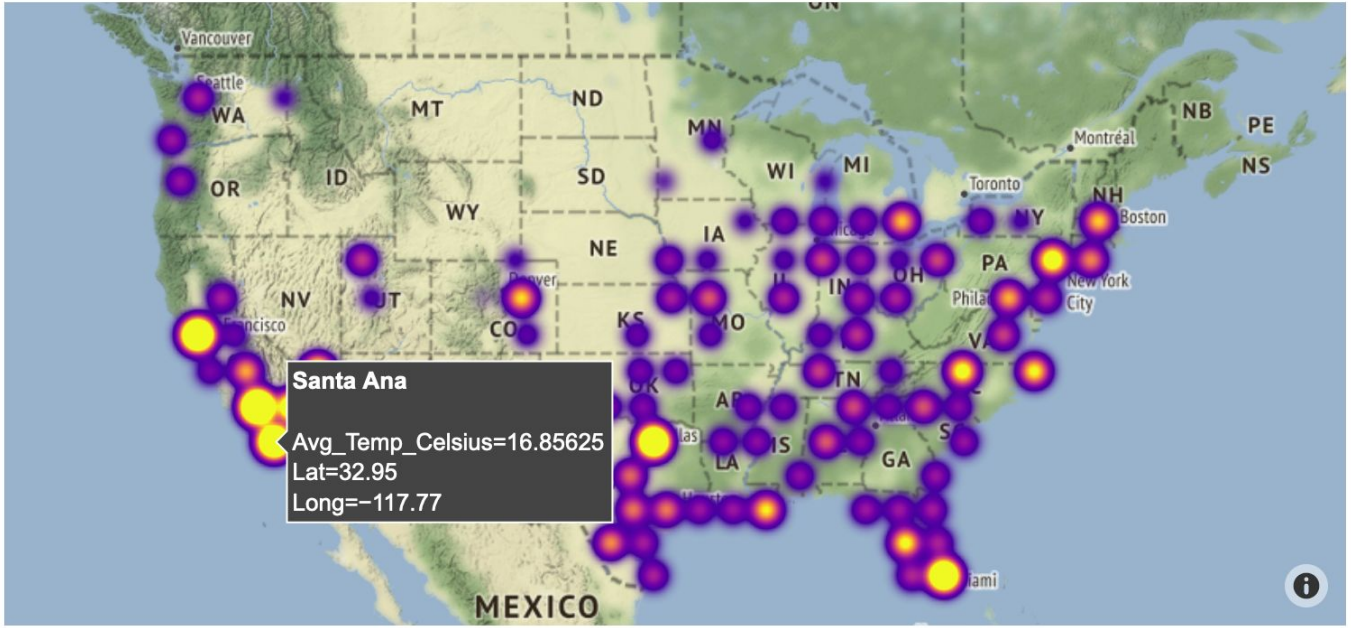
seaborn regplot (Lowess Smoothing) for non-linear data

```
df.plot(kind="scatter",
              x="year",
              y="gistemp",
              grid=True,
              figsize=(8,8),
              title="Surface Temp(°C) over the Years")


plt.ylabel('Surface Temperature(°C)')
plt.xlabel('YEAR (1880-2020)')

# define x and y values
x_values = df['year']
y_values = df['gistemp']


# Seaborne solution:
sns.regplot(x_values, y_values, data=df,  color="grey", scatter_kws={"s": 10},
             line_kws={"color":"r","alpha":1,"lw":1} ,lowess=True)
plt.xlabel("YEAR (1880-2020)"), plt.ylabel("Surface Temperature(°C)")


plt.savefig("../Images/Lowess_Smoothing.png")
plt.show()

```

https://seaborn.pydata.org/generated/seaborn.regplot.html

Average Yearly Temperature (°C) (1993-2013)

1993



MapBox Density HeatMap in Python

```
df = pd.read_csv('1993.csv')

import plotly.express as px
fig = px.density_mapbox(df, lat='Lat', lon='Long', z='Avg_Temp_Celsius', hover_name='City', radius=15,
                        title = "1993",
                        center=dict(lat=37.78, lon=-97.63), zoom=3,
                        mapbox_style="stamen-terrain")

fig.show()
```

https://plotly.com/python-api-reference/generated/plotly.express.density_mapbox.html

Power Plant Map

```
<!DOCTYPE html>
<meta charset="utf-8">

<!-- Load bootstrap -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrit

<!-- Our own CSS stylesheet -->
<link rel="stylesheet" href="assets/css/styles.css" media="screen">


<style>
    .land {
        /*fill: #ADFEFF;*/
        fill: #000;
    }

    .state-boundary {
        fill: none;
        stroke: rgb(240, 234, 234);
    }
    .bubble {
        fill-opacity: .5;
        /*stroke: #fff;*/
        stroke: none;
        stroke-width: .5px;
    }

</style>

<!-- Button -->
<div id="division" >
  <input type="checkbox" class="checkbox" value="FOSSIL" checked><label>Fossil</label>
  <input type="checkbox" class="checkbox" value="SOLAR" checked><label>Solar</label>
  <input type="checkbox" class="checkbox" value="HYDRO" checked><label>Hydroelectric</label>
  <input type="checkbox" class="checkbox" value="WIND" checked><label>Wind</label>
  <input type="checkbox" class="checkbox" value="MSW" checked><label>Municipal Solid Waste (MSW)</label>
  <input type="checkbox" class="checkbox" value="GEO" checked><label>Geothermal</label>
  <input type="checkbox" class="checkbox" value="NUCLEAR" checked><label>Nuclear</label>
  <input type="checkbox" class="checkbox" value="Other" checked><label>Other</label>
</div>

<!-- Create an element where the map will take place -->
<svg id="my_dataviz"></svg>

<script src="//d3js.org/d3.v4.min.js"></script>
<script src="//d3js.org/topojson.v2.min.js"></script>
<script>
```

```
// Set SVG width, height:
var width = 960*1.25,
    height = 500*1.25;

// Sclae projection to fit in SVG:
var projection = d3.geoAlbersUsa()
        .scale(1000*1.25)
        .translate([width / 2, height / 2]);

var path = d3.geoPath()
        .projection(projection);

// Create SCG with specified width and height:
var svg = d3.select("body").append("svg")
        .attr("width", width)
        .attr("height", height);

// Define scale for bubble radius:
//var radius = d3.scale.sqrt()
//        .domain([0, 16*1000])
//        .range([0, 15*1000]);

// Read US map data:
d3.json("us.json", function(JSONerror, us) {
    if (JSONerror) throw JSONerror;

    // Add filled US land:
    svg.insert("path", ".graticule")
            .datum(topojson.feature(us, us.objects.land))
            .attr("class", "land")
            .attr("d", path);

    // Add state boundaries:
    svg.insert("path", ".graticule")
            .datum(topojson.mesh(us, us.objects.states, function(a, b) { return a !== b; }))
            .attr("class", "state-boundary")
            .attr("d", path);
    // Create a color scale
    var color = d3.scaleOrdinal()
        .domain(["FOSSIL", "SOLAR", "HYDRO","WIND","MSW","GEO","NUCLEAR","Other"])
        .range([ "#36ebe8", "#e9f50a", "#7c05fa", '#2fff05','#0019f7','#0025f7', '#fc0303','#faf7f7'])
```

Power Plant Map

```
d3.csv("all_power_plants_nvk.csv", function(d) {
  return {
  lat : +d.lat,
  long : +d.long,
  group: d.group
  };
},

function(data) {
  console.log(data.group)
// Add circles:
svg
  .selectAll("myCircles")
  .data(data)
  .enter()
  .append("circle")
    .attr("class", function(d){ return d.group })
    .attr("cx", function(d){ return projection([d.long, d.lat])[0] })
    .attr("cy", function(d){ return projection([d.long, d.lat])[1] })
    //.attr("r", 5)
    .attr("r", function(d){ return size(d.group) })
    .style("fill", function(d){ return color(d.group) })
    .attr("stroke", function(d){ return color(d.group) })
    .attr("stroke-width", 2)

});

// This function is gonna change the opacity and size of selected and unselected circles
function update(){

  // For each check box:
  d3.selectAll(".checkbox").each(function(d){
  cb = d3.select(this);
  grp = cb.property("value")

  // If the box is check, I show the group
  if(cb.property("checked")){
      svg.selectAll("."+grp).transition().duration(1000).style("opacity", 1).attr("r", function(d){ return size(d.group) })
  }
  // Otherwise I hide it

  else {
      svg.selectAll("."+grp).transition().duration(1000).style("opacity", 0).attr("r", 0)
      }
  })
}
```

```
  // This function is gonna change the opacity and size of selected and unselected circles
function update(){

  // For each check box:
  d3.selectAll(".checkbox").each(function(d){
  cb = d3.select(this);
  grp = cb.property("value")

  // If the box is check, I show the group
  if(cb.property("checked")){
      svg.selectAll("."+grp).transition().duration(1000).style("opacity", 1).attr("r", function(d){ return size(d.group) })
  }
  // Otherwise I hide it

  else {
      svg.selectAll("."+grp).transition().duration(1000).style("opacity", 0).attr("r", 0)
      }
  })
}

// When a button change, I run the update function
d3.selectAll(".checkbox").on("change",update);

// And I initialize it at the beginning
update()
```

Bar chart - JS

```
Chart.defaults.global.defaultFontFamily = 'Roboto';
Chart.defaults.global.defaultFontColor = '#333';
//Chart.defaults.global.defaultFontSize = 16;
Chart.defaults.global.legend.display = false;

function makeChart(allPowerPlants) {

    var fuelType = allPowerPlants.map(function(d) {return d.TYPE_Clear3});
    var ammountAllPP = allPowerPlants.map(function(d) {return +d.Number_of_Power_Plants_of_Each_Type});

  //var playerColors = powerPlants.map(function(d) {return d.Gender === 'Female' ? '#F15F36' : '#19A0AA

    var ctx4 = document.getElementById("allPowerPlants").getContext("2d");

    var chart4 = new Chart(ctx4, {
        type: 'bar',
        data: {
            labels: fuelType,
            datasets: [{
                //label: false,
                data: ammountAllPP,
                backgroundColor: [
                    'rgba(224,73,87, 0.8)',
                    'rgba(244,234,150, 0.8)',
                    'rgba(90,151,206, 0.8)',
                    'rgba(173,90,255, 0.8)',
                    'rgba(54,206,86, 0.8)',
                    'rgba(75,192,192, 0.8)',
                    'rgba(255,99,132, 0.8)',
                    'rgba(255,159,64, 0.8)'],
                borderWidth: 2,
                borderColor: '#777',
                hoverBorderWidth: 3,
                hoverBorderColor: "#000"

            }],
        },
        options: {
            title: {
                display: true,
                text: "Number of Energy Power Plants in US 2019",
                fontSize: 18
            },
            scales: {
                yAxes: [{

                    ticks: {
                        fontSize: 12,
                        fontStyle: 'bold'
                    },
                    scaleLabel: {
                        display: true,
                        labelString: 'Amount',
                        fontSize: 16,
                        fontStyle: 'bold'
                    }
                }],
                xAxes: [{
                    //scaleFontSize: 20,
                    ticks: {
                        fontSize: 12,
                        fontStyle: 'bold'
                    },
                    scaleLabel: {
                        display: true,
                        labelString: 'Power Plant Type',
                        fontSize: 16,
                        fontStyle: 'bold'
                    }
                }]

            }
        },
    })

}

// Request data using D3
d3.csv('amount_pp_by_type.csv').then(makeChart);
```

Doughnut chart - html

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>CO2 Charts</title>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/d3/5.5.0/d3.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.min.js"></script>


</head>

<body>
  <div class="container" class="chart-container" style="position: relative; height:20vh; width:40vw">
      <canvas id="co2EmissionChart"></canvas>
      <canvas id="co2EmissionChart1"></canvas>
      <canvas id="co2EmissionChart2"></canvas>
      <canvas id="co2EmissionChart3"></canvas>
  </div>
  <script src="chart_from_csv.js"></script>
  <script> src = "https://d3js.org/d3.v5.min.js" </script>
  <script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-datalabels"></script>
</body>

</html>
```

Doughnut chart - html

```
function makeChart(powerPlants) {

    var fuelType = powerPlants.map(function(d) {return d.Fuel_Type});
    var ammountPP = powerPlants.map(function(d) {return +d.Number_of_Power_Plants});
    var co2Emission = powerPlants.map(function(d) {return +d.CO2_Emission_tons});
    var fuelComs = powerPlants.map(function(d) {return +d.Fuel_Consumption_MMBtu});
    var energy = powerPlants.map(function(d) {return +d.Energy_Generated_kWh});
  //var playerColors = powerPlants.map(function(d) {return d.Gender === 'Female' ? '#F

    //Percentage calculations

    var ctx = document.getElementById("co2EmissionChart").getContext("2d");
    var ctx1 = document.getElementById("co2EmissionChart1").getContext("2d");
    var ctx2 = document.getElementById("co2EmissionChart2").getContext("2d");
    var ctx3 = document.getElementById("co2EmissionChart3").getContext("2d");

    var chart = new Chart(ctx, {
        type: 'doughnut',
        data: {
            labels: fuelType,
            datasets: [{
                label: 'Fossil Fuel Types',
                data: ammountPP,
                backgroundColor: [
                    'rgba(153,102,255, 0.8)',
                    'rgba(54,206,86, 0.8)',
                    'rgba(75,192,192, 0.8)',
                   'rgba(255,99,132, 0.8)',
                    'rgba(255,159,64, 0.8)'],
                borderWidth: 2,
                borderColor: '#777',
                hoverBorderWidth: 3,
                hoverBorderColor: "#000"

            }],
        },
```

```
    options: {
        title: {
            display: true,
            text: "Fuel Consumption (MMBtu)",
            fontSize: 20
        },
        legend: {
            position: 'bottom'
        },
        plugins: {
            datalabels: {
                labels: {
                    render: 'label',
                    fontColor: '#000',
                    position: "outside"
                },
                color: '#fff',
                anchor: 'end',
                align: 'start',
                offset: -10,
                borderWidth: 2,
                borderColor: '#fff',
                borderRadius: 25,
                backgroundColor: (context) => {
                    return context.dataset.backgroundColor;
                },
                font: {
                    weight: 'bold',
                    size: '20'
                },
                formatter: (value, ctx2) => {
                    let sum = 0;
                    let dataArr = ctx2.chart.data.datasets[0].data;
                    dataArr.map(data => {
                        sum += data;
                    });
                    let percentage = (value*100 / sum).toFixed(0)+"%";
                    return  percentage;
                },
```