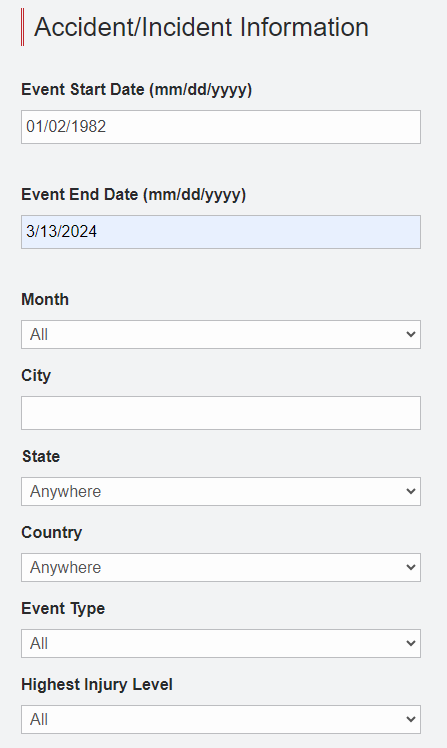
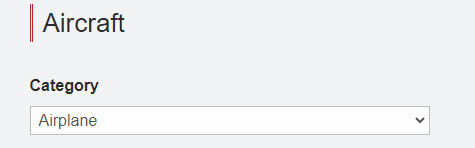
Project 1 Process

Timothy Masso

I have been interested in air disasters recently and with the recent aviation incidents at the beginning of the year such as the Alaskan Airlines flight where the door blew off, or the Japan Airlines flight which collided with the Japanese coast guard plane. Air Safety became a hot topic for the media and public.With the project I wanted to map and find out how risky air travel is. Are these incidents the outliers or the hidden truth that only now the media sheds light on.

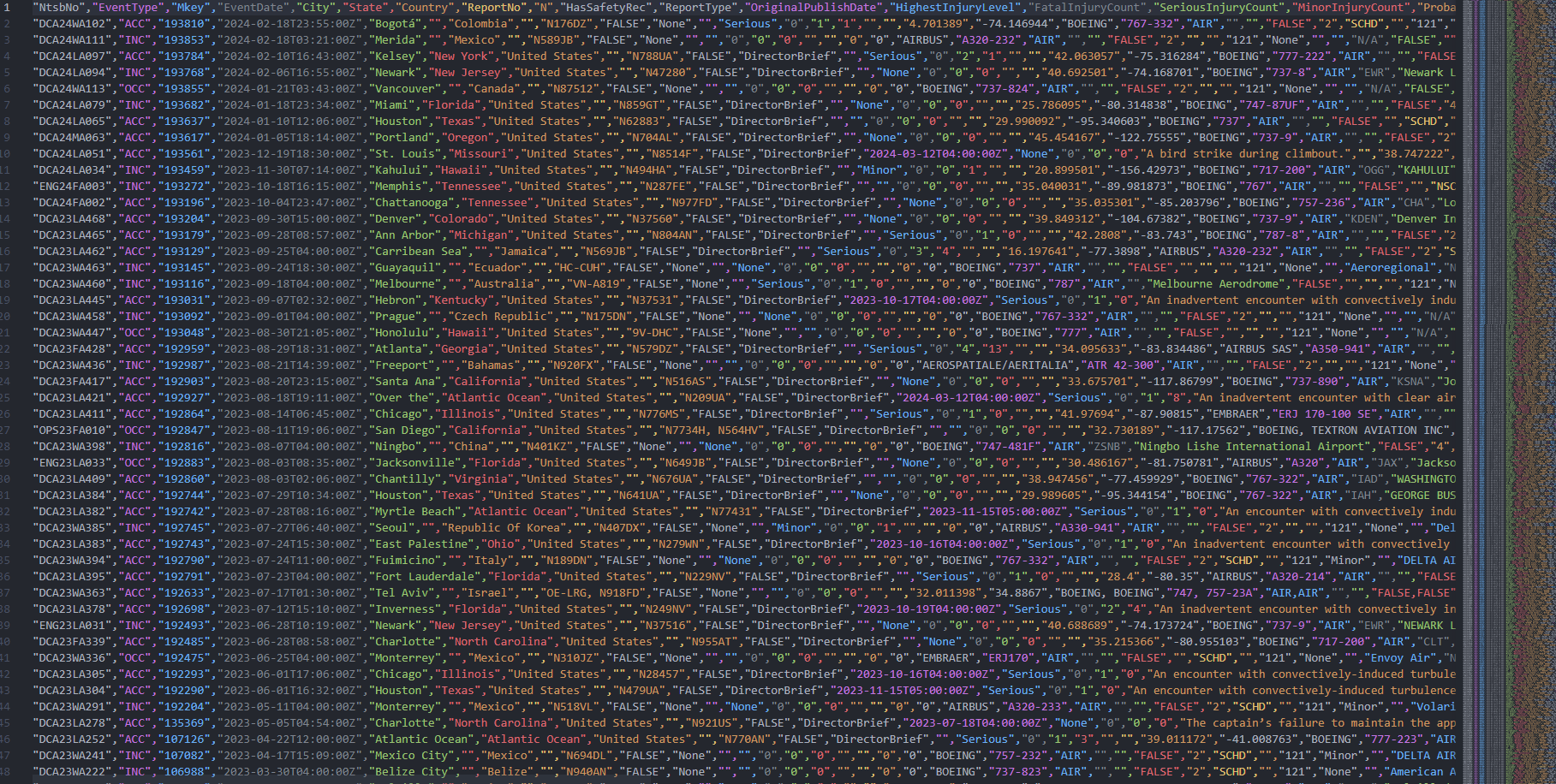
Gathering Data:

To gather data I knew the first place I could look was the NTSB's own investigation search query. Firstly I had to figure out how to work the search query which took some time and many exported files to see and get used to CSVs. But once I got used to it I then had to find out what Part number commercial aircraft flights fall under. Then after a quick google search I found it was Part: 121. For this project I took an arbitrary year 1982. Just to get a wide range of Data over 40 years. I put in Airplane for aircraft type, and only did commercial air flights.

I got back a lot of results which I then exported to a CSV file from the website. 

Organizing the Data:

Thankfully the NTSB is very thorough with the data they gather which means most of the important information i want to have is already there. The CSV contained "NtsbNo","EventType","Mkey","EventDate","City","State","Country","ReportNo","N","HasSafetyRec","ReportType","OriginalPublishDate","HighestInjuryLevel","FatalInjuryCount","SeriousInjuryCount","MinorInjuryCount","ProbableCause","EventID","Latitude","Longitude","Make","Model","AirCraftCategory","AirportID","AirportName","AmateurBuilt","NumberOfEngines","Scheduled","PurposeOfFlight","FAR","AirCraftDamage","WeatherCondition","Operator","ReportStatus","RepGenFlag","DocketUrl","DocketPublishDate".

Essentially everything I needed, but most importantly, NTSBno, Date it happened,coordinates of the incident, what airport it came from and is referenced with, injury categories, and aircraft damage. 

I did want to sort to the most used airframe producers over that time frame.to do that I knew the concept of the process is tell the code what string I'm looking for, to search the CSV, designate the column to look in, then tell it to only take the full row where that string appears in.

Getting specific rows from CSV:

I did some research on protocols that can parse CSV to be analyzed and found "papaparse” and downloaded it and read up on how to incorporate it into javascript. With the help of my brother I got started on the code. I started with making the code use node.js and papaparse with

const Papa = require("papaparse");

const fs = require("fs");

Then defining what I'm looking for

const csvFilePath = "Newnewdatafromntsb.csv";

onst outputPath = "newnewdata23.csv";

const makeColumn = "Make";

const targetMakes = [

"AIRBUS",

"BOEING",

"EMBRAER",

"Boeing",

"LOCKHEED",

"DOUGLAS",

"MCDONNELL",

"BOMBARDIER INC",

];

const countryColumn = "Country";

const targetCountry = [

"United States",

]

const damageColumn = "AirCraftDamage";

const targetDamage = [

"Destroyed",

"Substantial",

]

Then to tell the code to look at the rows and see if the info is there I made a function with the modifier filter

function findRows(csvData, columnName, targetMakes, targetDamage, columnName3, targetCountry, columnName4) {

return csvData.filter((row) => targetMakes.includes(row[columnName]) && targetDamage.includes(row[columnName3]) && targetCountry.includes(row[columnName4]) && row["Latitude"] != "" && row["Longitude"] != "" );

analyzeCSV(csvFilePath, makeColumn, targetMakes, outputPath, targetDamage, damageColumn, targetCountry, countryColumn);

function analyzeCSV(filePath, makeColumn, targetMakes, outputPath) {

const fileContent = fs.readFileSync(filePath, "utf8");

}

Then using papaparse use the previously made function to import the earlier defined consts

Papa.parse(fileContent, {

header: true,

skipEmptyLines: true,

complete: function (results) {

const rowsWithData = findRows(

results.data,

makeColumn,

targetMakes,

targetDamage,

damageColumn,

targetCountry,

countryColumn,

);

writeRowsToCSVFile(rowsWithData, outputPath);

},

});

}

Then to export the new data set write it to a new CSV file

function writeRowsToCSVFile(rows, outputPath) {

const csvContent = Papa.unparse(rows, { header: true });

fs.writeFileSync(outputPath, csvContent);

console.log(`Results exported to ${outputPath}`);

}

With all of that I get a new data set that I can now convert to geojson.

Converting to geojson:

Most of the process of reading the CSV is the same.

Knowing that a geojson file has to have a specific structure. I thought that I could manually set the header to each section. Which my brother had said is what we need to do. Setting the geojson format as an object. And making each section surrounded by the needed brackets and braces.

var geoJson = {};

geoJson.type = "FeatureCollection";

geoJson.name = "CommercialAirCrashes";

geoJson.features = [];

Then to name and make sure that those names in the properties will return the correct data in the correct expression. Numbers need to be numbers and strings need to be strings.

for (let i in r.data) {

var b = {"type": "Feature"};

b.properties = properties = {

NtsbNo: r.data[i].NtsbNo,

EventDate: r.data[i].EventDate,

Country: r.data[i].Country,

City: r.data[i].City,

State: r.data[i].State,

AccidentLat: parseFloat(r.data[i].Latitude),

AccidentLong: parseFloat(r.data[i].Longitude),

AirportName: r.data[i].AirportName,

AirportID: r.data[i].AirportID,

Registration: r.data[i].N,

FatalInjuryCount: r.data[i].FatalInjuryCount,

SeriousInjuryCount: r.data[i].SeriousInjuryCount,

MinorInjuryCount: r.data[i].MinorInjuryCount,

AirCarrier: r.data[i].Operator,

Make: r.data[i].Make,

Model: r.data[i].Model,

NumberOfEngines: r.data[i].NumberOfEngines,

AirCraftCategory: r.data[i].AirCraftCategory,

AirCraftDamage: r.data[i].AirCraftDamage,

ProbableCause: r.data[i].ProbableCause,

ObjectId: parseInt(i),

}

b.geometry = {

type: "Point",

coordinates: [parseFloat(r.data[i].Longitude), parseFloat(r.data[i].Latitude)],

};

Then to finally close the array and tell it to return the whole thing. Along with making it a json file rather than a CSV. concluding with exporting the new data.

geoJson.features.push(b);

}

return geoJson;

}

const data = JSON.stringify(a);

fs.writeFile("newnewData27.json", data, (err) => {

if (err) {

console.error('Error writing file:', err);

} else {

console.log("JSON data is saved.");

}

});

Now I have a nice and neat geojson data set.

{

"type": "FeatureCollection",

"name": "CommercialAirCrashes",

"features": [

{

"type": "Feature",

"properties": {

"NtsbNo": "DCA24LA065",

"EventDate": "2024-01-10T12:06:00Z",

"Country": "United States",

"City": "Houston",

"State": "Texas",

"AccidentLat": 29.990092,

"AccidentLong": -95.340603,

"AirportName": "",

"AirportID": "",

"Registration": "N62883",

"FatalInjuryCount": "0",

"SeriousInjuryCount": "0",

"MinorInjuryCount": "0",

"AirCarrier": "UNITED AIRLINES INC",

"Make": "BOEING",

"Model": "737",

"NumberOfEngines": "",

"AirCraftCategory": "AIR",

"AirCraftDamage": "Substantial",

"ProbableCause": "",

"ObjectId": 0,

},

"geometry": {

"type": "Point",

"coordinates": [

-95.340603,

29.990092

]

}

},

Next I wanted to Add the coordinates of the origin Airport to do this I had to again research and find a CSV with every airport code in the United States. Once I had that information I was able to start.

The beginning of the code is much the same as the rest.

File it's just one long if and else if statement with different modifiers. A distance value I can use later on but ended up being useless But I didn't realize till after I had done everything.

let csvData = fs.readFileSync('airport-codes-updated.csv', 'utf8');

let records = Papa.parse(csvData, { header: true }).data;

let csvMap = new Map(records.map(row => [row.ident, { Lat: parseFloat(row.Lat), Long: parseFloat(row.Long) }]));

let csvMapName = new Map(records.map(row => [row.name, { Lat: parseFloat(row.Lat), Long: parseFloat(row.Long) }]))

data.features.forEach(feature => {

let airportId = feature.properties.AirportID;

let AirportName = feature.properties.AirportName;

console.log('Processing feature with AirportID:', airportId);

if (csvMap.has(airportId)) {

console.log('Found matching record in CSV data');

feature.properties.AirportLongitude = csvMap.get(airportId).Long;

feature.properties.AirportLatitude = csvMap.get(airportId).Lat;

} else if (csvMap.has('K' + airportId)) {

console.log('Found matching record with K in CSV data');

feature.properties.AirportLongitude = csvMap.get('K' + airportId).Long;

feature.properties.AirportLatitude = csvMap.get('K' + airportId).Lat;

} else if (csvMap.has('P' + airportId)) {

console.log('Found matching record with P in CSV data');

feature.properties.AirportLongitude = csvMap.get('P' + airportId).Long;

feature.properties.AirportLatitude = csvMap.get('P' + airportId).Lat;

} else if (csvMapName.has(AirportName + ' Airport')) {

console.log('Found matching record with Airport in CSV data');

feature.properties.AirportLongitude = csvMapName.get(AirportName + ' Airport').Long;

feature.properties.AirportLatitude = csvMapName.get(AirportName + ' Airport').Lat;

}

else if (feature.properties.AirportName == "") {

console.log('Empty AirportName, setting AirportLatitude to AccidentLatitude');

feature.properties.AirportLatitude = feature.properties.AccidentLatitude;

feature.properties.AirportLongitude = feature.properties.AccidentLongitude;

}

else if (feature.properties.AirportID == "") {

console.log('Empty AirportID, setting AirportLatitude to AccidentLatitude');

feature.properties.AirportLatitude = feature.properties.AccidentLatitude;

feature.properties.AirportLongitude = feature.properties.AccidentLongitude;

}

else if (feature.properties.AccidentLat == "") {

console.log('Empty Accident, setting AirportLatitude to AccidentLatitude');

feature.properties.AccidentLat = feature.properties.AirportLatitude;

}

else if (feature.properties.AccidentLong == "") {

console.log('Empty Accidentlo, setting AirportLatitude to AccidentLatitude');

feature.properties.AccidentLong = feature.properties.AirportLongitude;

}

let distance = haversineDistance(

feature.properties.Lat,

feature.properties.Long,

feature.properties.AirportLatitude,

feature.properties.AirportLongitude

);

feature.properties.Distance = distance;

});

fs.writeFileSync('newnewDatacord42.json', 'let data = ' + JSON.stringify(data, null, 2) + ';');

console.log('Updated data written to New File');

Then finally to export it. Doing complete finish data set with all the information I needed.

Leaflet Code:

My leaflet code is very simple. It is just an icon code plus two sets of Marker functions one for airports one for incident locations, a line function and legend.

let map = L.map("map").setView([25, -90], 4);

L.tileLayer("https://tile.openstreetmap.org/{z}/{x}/{y}.png", {

maxZoom: 19,

attribution:

'&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>',

}).addTo(map);

let crashIcon = L.icon({

iconUrl: "Assets/airplane.png",

iconSize: [30, 30],

});

let towerIcon = L.icon({

iconUrl: "Assets/tower.png",

iconSize: [30, 30],

});

L.geoJson(dataFinal, {

pointToLayer: function (feature, latlng) {

if (feature.geometry.coordinates[0] !== 0) {

console.log(feature.geometry.coordinates[0]);

let marker = L.marker(latlng, { icon: crashIcon });

marker.bindPopup(

"Date: " +

feature.properties.EventDate +

"<br/>" +

"NtsbNo: " +

feature.properties.NtsbNo +

"<br/>" +

"Make: " +

feature.properties.Make +

"<br/>" +

"Model: " +

feature.properties.Model +

"<br/>" +

"Dead: " +

feature.properties.FatalInjuryCount +

"<br/>" +

"Serious: " +

feature.properties.SeriousInjuryCount +

"<br/>" +

"Minor: " +

feature.properties.MinorInjuryCount +

"<br/>" +

"ProbableCause: " +

feature.properties.ProbableCause

);

return marker;

} else {

return null;

}

},

}).addTo(map);

L.geoJSON(dataFinal, {

onEachFeature: function (feature, layer) {

if (

typeof feature.properties.AirportLatitude !== "undefined" &&

feature.geometry.coordinates[0] !== 0

) {

let marker = L.marker(

[

feature.properties.AirportLatitude,

feature.properties.AirportLongitude,

],

{ icon: towerIcon }

);

marker.bindPopup(

"Date: " +

feature.properties.EventDate +

"<br/>" +

"Airport:Name: " +

feature.properties.AirportName +

"<br/>" +

"Corresponding Accident: " +

feature.properties.NtsbNo,

);

marker.addTo(map);

var polyline = L.polyline(

[

[

feature.properties.AirportLatitude,

feature.properties.AirportLongitude,

],

feature.geometry.coordinates.reverse(),

],

{ color: "red" }

).addTo(map);

} else {

return null;

}

},

});

let legend = L.control({ position: "topright" });

legend.onAdd = function (map) {

let div = L.DomUtil.create("div", "legend");

div.style.backgroundColor = "white";

div.style.padding = "10px";

div.innerHTML = "Number of Serious Harmful Incidents Since 1982: " + dataFinal.features.length;

return div;

};

legend.addTo(map);

My HTML code is very basic just referencing where I get my code and data from.

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="map.css">

<meta charset="utf-8" />

<title>Leaflet Map</title>

<link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />

<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>

<!-- <script src="Data.js" type="text/javascript"></script> -->

<script src="DataFinal.js" type="text/javascript"></script>

</head>

<body>

<div id="map" style="height: 1000px;"></div>

<script src="MarkerCode.js"></script>

</body>

</html>

Kepler Process:

I imported my data and created the points and then arcs to each corresponding airport and incident. Then i tried to put a grid where it displayed how many had injuries.

.

Why this data is interesting:

With a total of 2529 incidents from 1982 until march of this year versus the 271 serious. Most serious incidents happen during landing or take off or at the airport and even with serious damage to the plane, passengers are not hurt. Compared to the millions of flights even in a year. Having learned about plane incidents on my own I do know that some pilots and companies don't say to write a report if it's not bad enough but the true number of incidents not reported is something I don't know. Over all my data displays that these incidents are not common. However, corporate negligence recently has made dangerous practices in the industry of airplane construction and maintenance come to the public and is a good thing as it can now make the negligence eradicated.