California Wildfire Feature Engineering

Owners: Sam Neuschwander, Abigail Witt, Spencer Zwiener

Additional information about all datasets, features, and joins can be found in the "Project Writeup" document.

In [1]:
```
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn import preprocessing
plt.rcParams['figure.figsize'] = [40, 40]
# display all columns
pd.set_option('display.max_columns', None)
```

Transmission Lines Data

In [2]:
```
# read in shapefile of Transmission Lines
shp_df = gpd.read_file('Transmission_Line.shp')
```

Curation Step 1: To ensure our transmission lines only consists of lines that were created prior to the wildfires, in our wildfire dataset, we are removing any transmission lines data with a Creator_Date after 2014-04-01

In [3]:
```
# First keeping an object of our outliers, so we can see what was removed.
shp_df_outliers = shp_df.loc[shp_df['Creator_Da'] > '2014-04-01']
```

In [4]:
```
# Filtering to transmission lines data with a Creator_Date after 2014-04-01
shp_df = shp_df.loc[shp_df['Creator_Da'] < '2014-04-01']
```

In [5]:
```
# View Transmission data (post '2014-04-01')
shp_df.head()
```

Out[5]:

	OBJECTID	Name	kV	kV_Sort	Owner	Status	Circuit	Type	Legend	Length_Mil	
0	1	AMP 115kV	115	115.0	AMP	Operational	Single	OH	Other_110_161kV	2.0	
1	2	AMP 115kV	115	115.0	AMP	Operational	Single	OH	Other_110_161kV	3.0	
2	3	AMP 115kV	115	115.0	AMP	Operational	Single	OH	Other_110_161kV	1.0	
3	4	AMP 115kV	115	115.0	AMP	Operational	Single	OH	Other_110_161kV	1.0	
4	5	ANZA 34kV	34	34.0	ANZA	Operational	Single	OH	Other_33_92kV	24.0	1

In [6]:

```python
# Looking at dataset size
shp_df.shape
```

Out[6]:

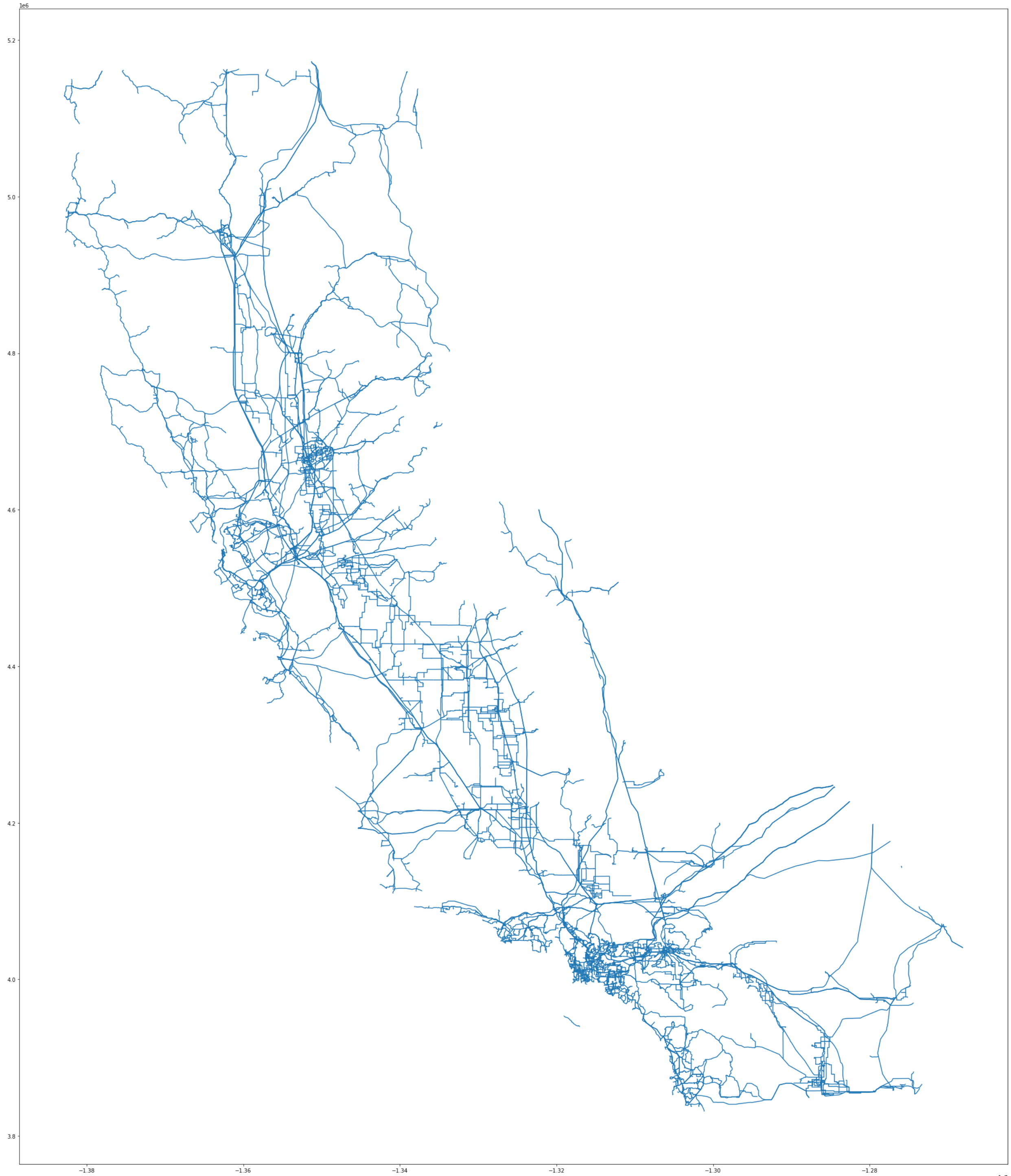
```
(6789, 21)
```

In [7]:

```python
shp_df.plot()
```

Out[7]:

```
<AxesSubplot:>
```



Curation Step 2: filter data to only Overhead powerlines (OH) as the others do not pose a threat of wildfire (underground or underwater)

In [8]:

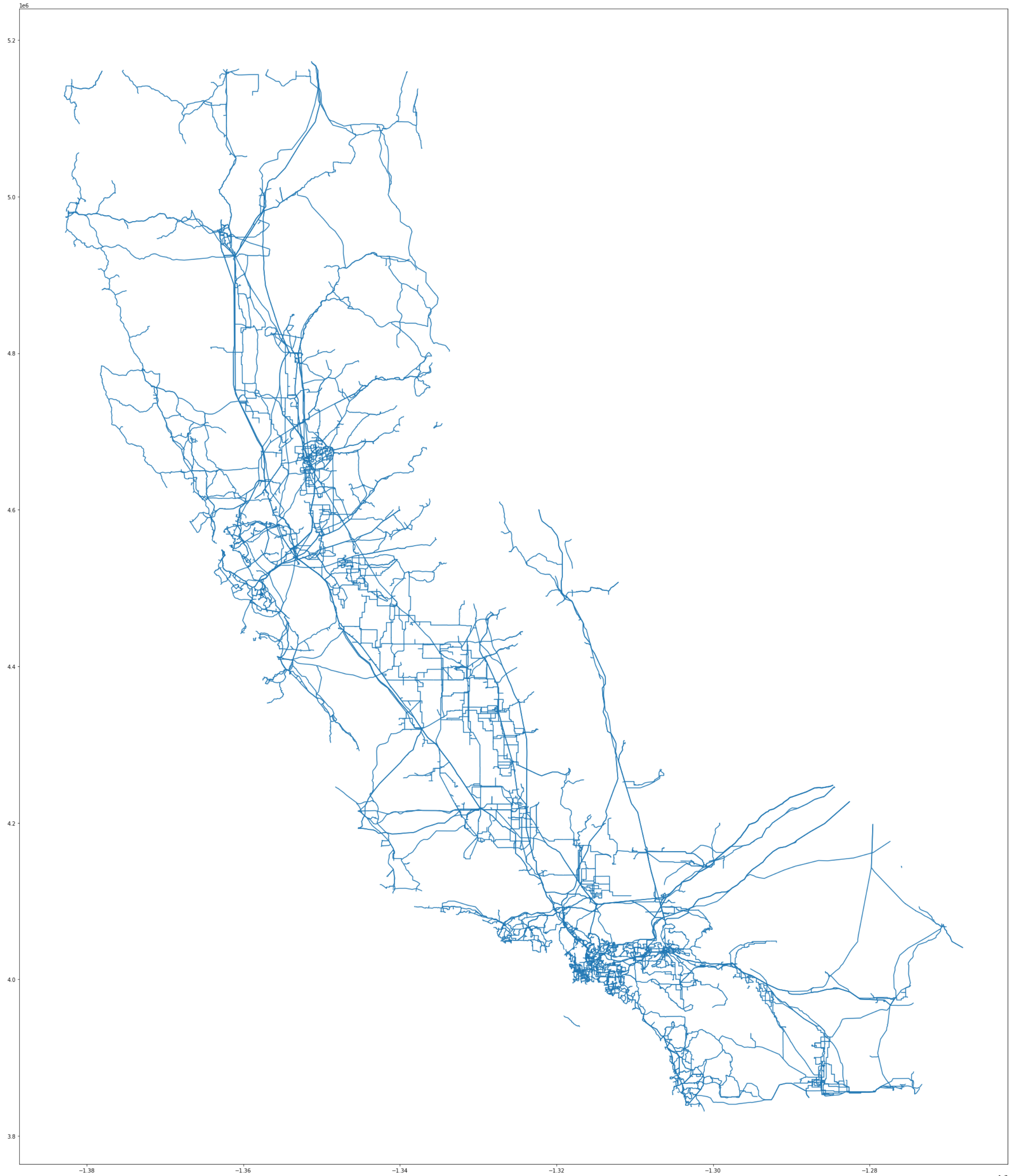
```
# filter data to only Overhead powerlines (OH) as the others do not pose a threat of w
oh_df = shp_df.loc[shp_df['Type'] == 'OH']
```

In [9]:

```
oh_df.plot()
```

Out[9]:

```
<AxesSubplot:>
```



In [10]:
```
# looking at dataset size
oh_df.shape
```

Out[10]:
```
(6662, 21)
```

Creating Buffer of 200 Meters that will be used to analyze Wildfires that are within transmission lines

In [11]:
```
# creating a buffer aroud the powerlines. This will help when merging with wildfires r
buffered_oh = oh_df.buffer(200)
# converting the buffered polygon from a geoseries, to a geodataframe.
buffered_oh = gpd.GeoDataFrame(geometry=buffered_oh)
```

In [12]:
```
# viewing the data
buffered_oh
```

Out[12]:

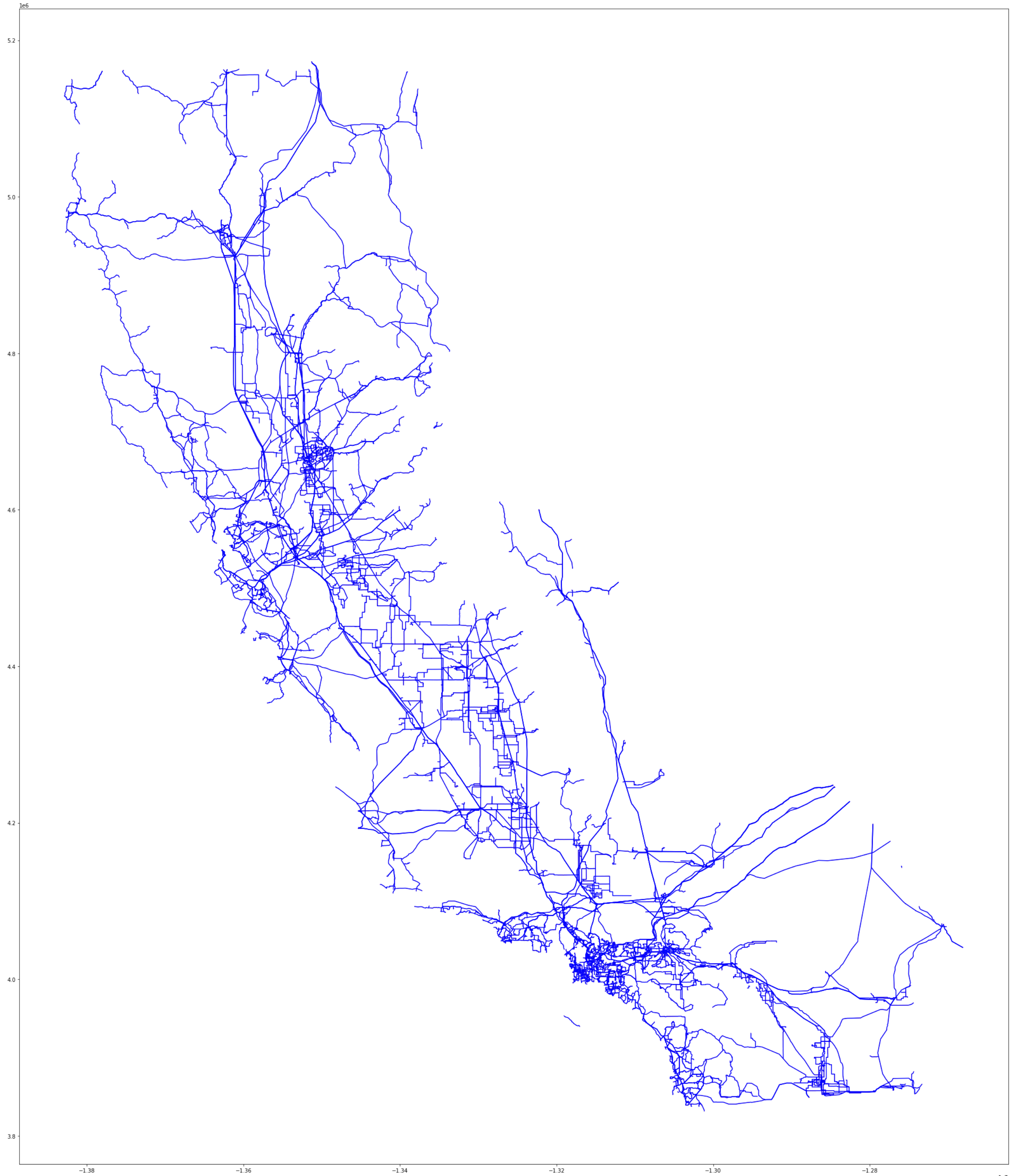
	geometry
0	POLYGON ((-13607826.280 4547379.955, -13607380...
1	POLYGON ((-13613614.310 4548105.534, -13613630...
2	POLYGON ((-13613623.671 4548511.067, -13613624...
3	POLYGON ((-13613312.050 4549705.819, -13613313...
4	POLYGON ((-12994264.451 3989303.082, -12994194...
...	...
6808	POLYGON ((-13650667.027 4959966.625, -13650678...
6809	POLYGON ((-13631097.569 4955268.974, -13631100...
6810	POLYGON ((-13630523.582 4954736.549, -13630596...
6811	POLYGON ((-12705691.078 4068341.189, -12705689...
6812	POLYGON ((-13362510.066 4215965.879, -13362223...

6662 rows × 1 columns

In [13]:
```
# confirming type
type(buffered_oh)
```

Out[13]:
```
geopandas.geodataframe.GeoDataFrame
```

In [14]:
```
#plot the lines and the buffers (the buffer is too small to show, can increase the siz
ax = buffered_oh.plot(color = 'red')

oh_df.plot(ax=ax, color = 'blue')
plt.show()
```



Wildfire Data

In [15]:
```
# read in shapefile of Wildfires, filtered to only the data within our boundbox -- pro
incidents_df = gpd.read_file('Incidents.shp')
```

In [16]:
```
fire_shp_df = incidents_df
fire_shp_df.shape
```

Out[16]:
```
(286411, 95)
```

Curation Step 1: Filter to wildfires in California

In [18]:
```
# Filtering dataset to only California Wildfires.
fire_shp_df = fire_shp_df.loc[fire_shp_df['POOState'] == 'US-CA']
```

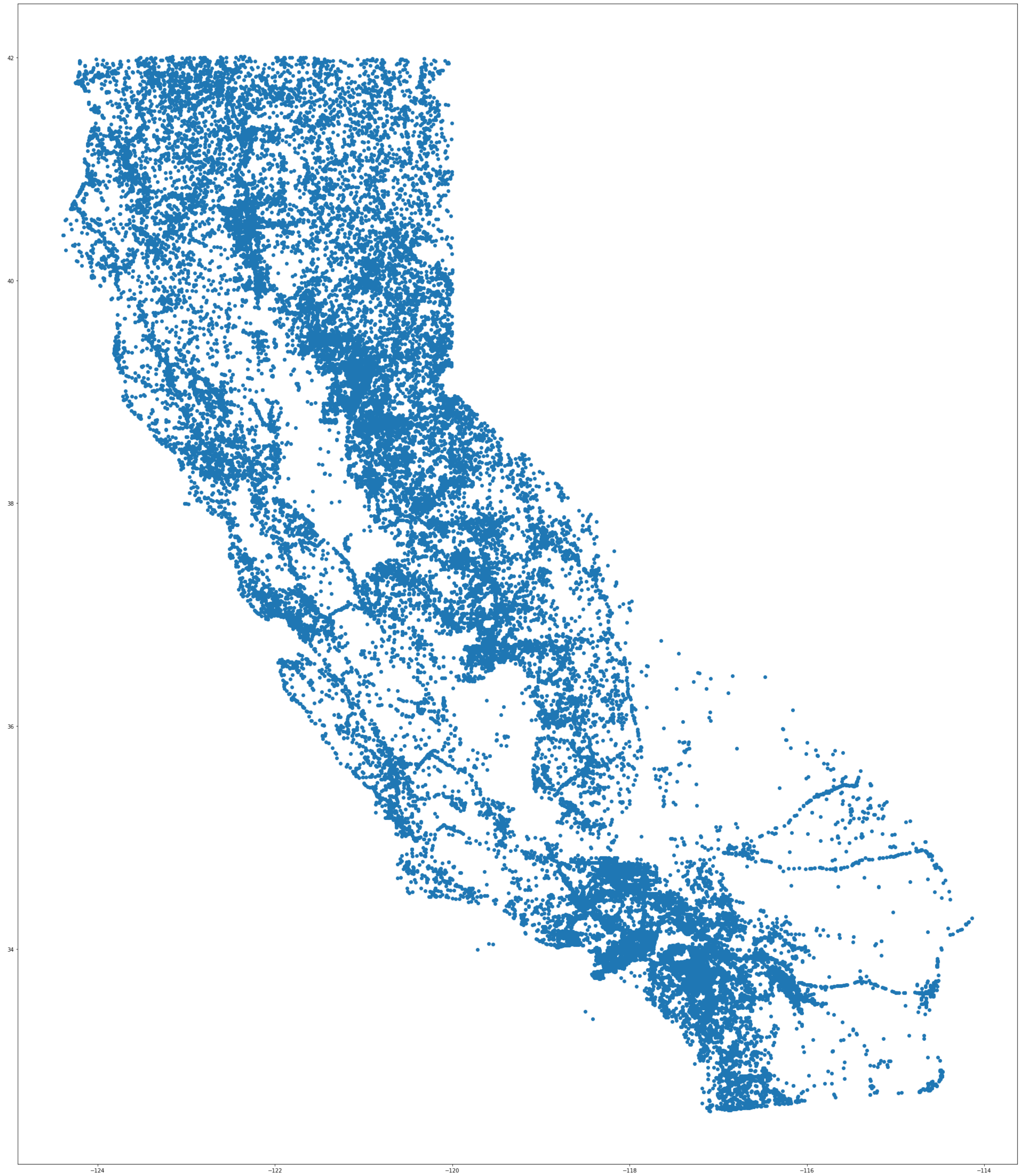
Curation Step 2: Filter out the outliers

In [19]:
```
# First keeping an object of our outliers, so we can see what was removed.
fire_shp_df_outliers = fire_shp_df.loc[fire_shp_df['FireDiscov'] < '2014-01-01']
```

In [20]:
```
# Cleaning up data. Two incidents had dates that looked to be outliers. Likely key err
fire_shp_df = fire_shp_df.loc[fire_shp_df['FireDiscov'] > '2014-01-01']
```

In [21]:
```
fire_shp_df.plot()
```

Out[21]:
```
<AxesSubplot:>
```



In [22]:
```
fire_shp_df.shape
```

Out[22]:
```
(72552, 95)
```

In [23]:
```
shp_df.crs
```

Out[23]:
```
<Projected CRS: EPSG:3857>
Name: WGS 84 / Pseudo-Mercator
Axis Info [cartesian]:
- X[east]: Easting (metre)
- Y[north]: Northing (metre)
Area of Use:
- name: World between 85.06°S and 85.06°N.
- bounds: (-180.0, -85.06, 180.0, 85.06)
Coordinate Operation:
- name: Popular Visualisation Pseudo-Mercator
- method: Popular Visualisation Pseudo Mercator
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

In [24]:
```
fire_shp_df.crs
```

Out[24]:
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

Curation Step 3: Update the coordinate reference system (CRS) to 3857 so that we can use meters as a distance measurement and have a common CRS with Transmission Lines data

In [25]:
```
# Updating CRS of Wildfire data
fire_shp_df = fire_shp_df.to_crs('epsg:3857')
```

In [26]:
```
fire_shp_df.crs
```

Out[26]:
```
<Projected CRS: EPSG:3857>
Name: WGS 84 / Pseudo-Mercator
Axis Info [cartesian]:
- X[east]: Easting (metre)
- Y[north]: Northing (metre)
Area of Use:
- name: World between 85.06°S and 85.06°N.
- bounds: (-180.0, -85.06, 180.0, 85.06)
Coordinate Operation:
- name: Popular Visualisation Pseudo-Mercator
- method: Popular Visualisation Pseudo Mercator
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

Join 1: Joing curated wildfire data to the curated, buffered Transmission lines using spatial join, within predicate

- Within means that the resulting geodataframe will contain rows where the geometries from wildfires are completely within the geometries of the buffered tranmsission lines http://shapely.readthedocs.io/en/latest/manual.html#binary-predicates

- This is required because the geometry component of the wildfire dataset is a Point

In [27]:
```
# Joining the widlfires to the transmission lines
join_df = gpd.sjoin(fire_shp_df, buffered_oh, how='left', predicate='within')
```

In [28]:
```
# shows that our data has now changed, sjoin worked
join_df.shape
```

Out[28]:
```
(77768, 96)
```

In [29]:
```
# getting a sense for how many sjoins occurred by counting the number of unique occurr
join_df['index_right'].nunique()
```

Out[29]:
```
2025
```

Feature 1: Close to Lines

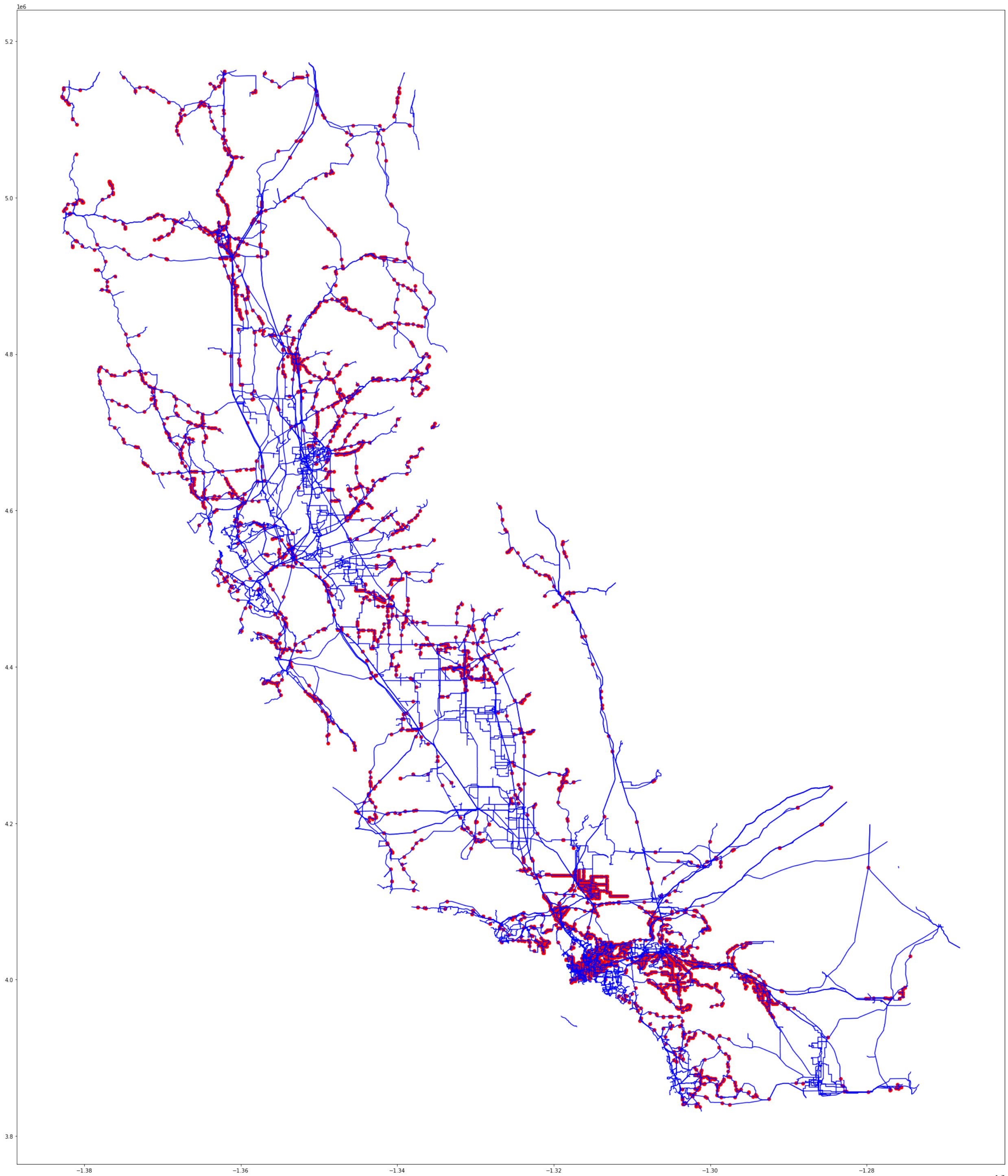
In [30]:
```
# creating a feature to mark a column as Y/N if they fall within proximity of Transmis

# Initialing the New Feature to 'N'
join_df['CloseToLines'] = 'N'
# Changing new feature to 'Y' if it was within a powerline from our spatial join (inde
join_df.loc[join_df['index_right'].notnull(), 'CloseToLines'] = 'Y'
```

In [31]:
```
# view the new feature
join_df.head()
```

Out[31]:

	OBJECTID	SourceOID	ABCDMisc	ADSPermiss	Containmen	ControlDat	CreatedByS	IncidentSi	D
0	1	7747595	NaN	DEFAULT	NaN	NaN	lacocad	NaN	
1	2	6384391	NaN	DEFAULT	NaN	NaN	firecode	NaN	
2	3	1383752	NaN	DEFAULT	NaN	NaN	firecode	NaN	
3	4	22499589	NaN	DEFAULT	NaN	NaN	cfcad	NaN	
4	5	23869477	NaN	DEFAULT	NaN	NaN	lacocad	NaN	

In [32]:
```
# ploting the transmission lines, and wildfires that fell within close proximity to th
ax = join_df[join_df['CloseToLines'] == 'Y'].plot(color='red')

oh_df.plot(ax=ax,color='blue')
plt.show()
```



Outlier Data

- This is the data that was removed from our Tranmission Line dataset and our Wildfire Dataset

In [33]:
```
# Transmission Line Data Outliers
shp_df_outliers
```

Out[33]:

	OBJECTID	Name	kV	kV_Sort	Owner	Status	Circuit	Type	Legend	Le
104	105	ESJ 250kV	250	250.0	ESJ	Proposed	Single	OH	Other_200_287kV	
105	106	ESJ 500kV	500	500.0	ESJ	Proposed	Single	OH	Other_345_500kV	
290	291	IID 230kV	230	230.0	IID	Operational	Double	OH	IID_230kV	
291	292	IID 230kV	230	230.0	IID	Operational	Double	OH	IID_230kV	
292	293	IID 230kV	230	230.0	IID	Operational	Double	OH	IID_230kV	
429	430	LADWP 230kV	230	230.0	LADWP	Proposed	Single	OH	LADWP_220_287kV	
3480	3481	PG&E 230kV	230	230.0	PG&E	Proposed	Single	OH	PG&E_230kV	
3481	3482	PG&E 230kV	230	230.0	PG&E	Operational	Double	OH	PG&E_230kV	
5729	5730	SCE 115kV	115	115.0	SCE	Operational	Single	OH	SCE_115_161kV	
5730	5731	SCE 115kV	115	115.0	SCE	Operational	Double	OH	SCE_115_161kV	
5731	5732	SCE 220kV	220	220.0	SCE	Operational	Single	OH	SCE_220_230kV	
5732	5733	SCE 500kV	500	500.0	SCE	Operational	Single	OH	SCE_500kV	

	OBJECTID	Name	kV	kV_Sort	Owner	Status	Circuit	Type	Legend	Le
5733	5734	SCE 500kV	500	500.0	SCE	Operational	Single	OH	SCE_500kV	
5734	5735	SCE 500kV	500	500.0	SCE	Operational	Single	OH	SCE_500kV	
5735	5736	SCE 500kV	500	500.0	SCE	Operational	Single	OH	SCE_500kV	
5736	5737	SCE 500kV	500	500.0	SCE	Operational	Single	OH	SCE_500kV	
5737	5738	SCE 500kV	500	500.0	SCE	Operational	Single	OH	SCE_500kV	
5738	5739	SCE 500kV	500	500.0	SCE	Operational	Single	OH	SCE_500kV	
5739	5740	SCE 500kV	500	500.0	SCE	Operational	Single	OH	SCE_500kV	
5740	5741	SCE 500kV	500	500.0	SCE	Operational	Single	OH	SCE_500kV	
5741	5742	SCE 500kV	500	500.0	SCE	Operational	Single	OH	SCE_500kV	
5742	5743	SCE 500kV	500	500.0	SCE	Operational	Double	OH	SCE_500kV	
5743	5744	SCE 220kV	220	220.0	SCE	Operational	Double	OH	SCE_220_230kV	
5744	5745	HARRY ALLEN-ELDORADO	NaN	NaN	NaN	Unknown	NaN	NaN	NaN	

	OBJECTID	Name	kV	kV_Sort	Owner	Status	Circuit	Type	Legend	Le
5745	5747	SCE 220kV	220	220.0	SCE	Operational	Single	OH	SCE_220_230kV	
5746	5748	SCE 115kV	115	115.0	SCE	Operational	Double	OH	SCE_115_161kV	
5747	5749	SCE 115kV	115	115.0	SCE	Operational	Double	OH	SCE_115_161kV	
5748	5750	SCE 115kV	115	115.0	SCE	Operational	Double	OH	SCE_115_161kV	
5749	5751	SCE 66kV	66	66.0	SCE	Operational	Double	OH	SCE_33_69kV	
5750	5752	SCE 66kV	66	66.0	SCE	Operational	Single	UG	SCE_33_69kV	
5751	5753	SCE 66kV	66	66.0	SCE	Operational	Single	UG	SCE_33_69kV	
5752	5754	SCE 66kV	66	66.0	SCE	Operational	Single	UG	SCE_33_69kV	
5753	5755	SCE 66kV	66	66.0	SCE	Operational	Single	UG	SCE_33_69kV	
6112	6117	SDG&E 230kV	230	230.0	SDG&E	Operational	Double	OH	SDG&E_230kV	
6113	6118	SDG&E 230kV	230	230.0	SDG&E	Operational	Double	UG	SDG&E_230kV	

In [34]:
```
# Wildfire Dataset
fire_shp_df_outliers
```

Out[34]:

	OBJECTID	SourceOID	ABCDMisc	ADSPermiss	Containmen	ControlDat	CreatedByS	Incident
38346	38357	7621579	NaN	DEFAULT	NaN	NaN	wildcad	Na
71283	71312	15936788	NaN	DEFAULT	NaN	NaN	firecode	Na

Weather Data

- California Average Temperatures
- California Precipitation
- Monthly grain for both

Join 2: Temporal join betwen the California Average Temperatures and California Precipitation data to make Weather Data

In [35]:

```
avg_temp_cali = pd.read_csv('avg_temp_california.csv', delimiter=',', header=4)
precip_cali = pd.read_csv('precipitation_california.csv', delimiter=',', header=4)
weather_df = avg_temp_cali.merge(precip_cali,on='Date',how='inner',suffixes=('.avgTemp
# merge on the date column because they are the same date format
# performing an inner join because we want data only for the months where we have both
weather_df['Date']=weather_df['Date'].astype(str)
weather_df.columns=['Date','Average Temperature','Temperature Anomaly','Precipitation'
weather_df.head()
```

Out[35]:

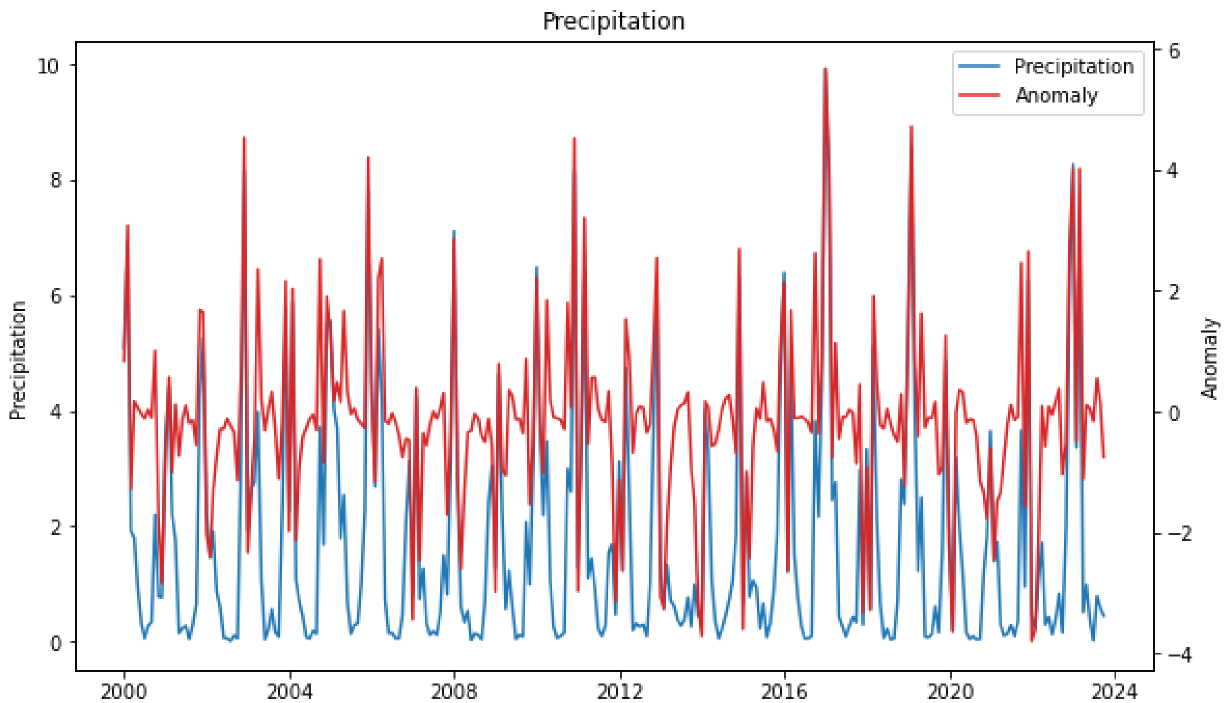
	Date	Average Temperature	Temperature Anomaly	Precipitation	Precipitation Anomaly
0	200001	45.8	3.5	5.09	0.84
1	200002	47.2	1.9	6.93	3.08
2	200003	50.2	1.4	1.92	-1.29
3	200004	57.8	3.8	1.80	0.17
4	200005	63.9	3.2	0.94	0.06

In [55]:

```
# copy dataframe for plotting
precip_cali_plot = precip_cali.copy()

# Convert 'Date' column to datetime format
precip_cali_plot['Date'] = pd.to_datetime(precip_cali_plot['Date'], format='%Y%m')

# Set 'Date' column as the index
precip_cali_plot.set_index('Date', inplace=True)

#setting the figure and axes
fig,ax = plt.subplots(figsize=(10, 6))

#plotting the first plot
```

```
line1 = ax.plot(precip_cali_plot.index, precip_cali['Value'], color='tab:blue', label
ax.set_ylabel('Precipitation')

#using twinx to generate another y axes and plotting
ax2 = ax.twinx()
line2 = ax2.plot(precip_cali_plot.index, precip_cali['Anomaly'], color='tab:red', labe
ax2.set_ylabel('Anomaly')

plt.title('Precipitation')

# Legend
lines = line1 + line2
labels = [line.get_label() for line in lines]
plt.legend(lines, labels, loc='upper right')
```

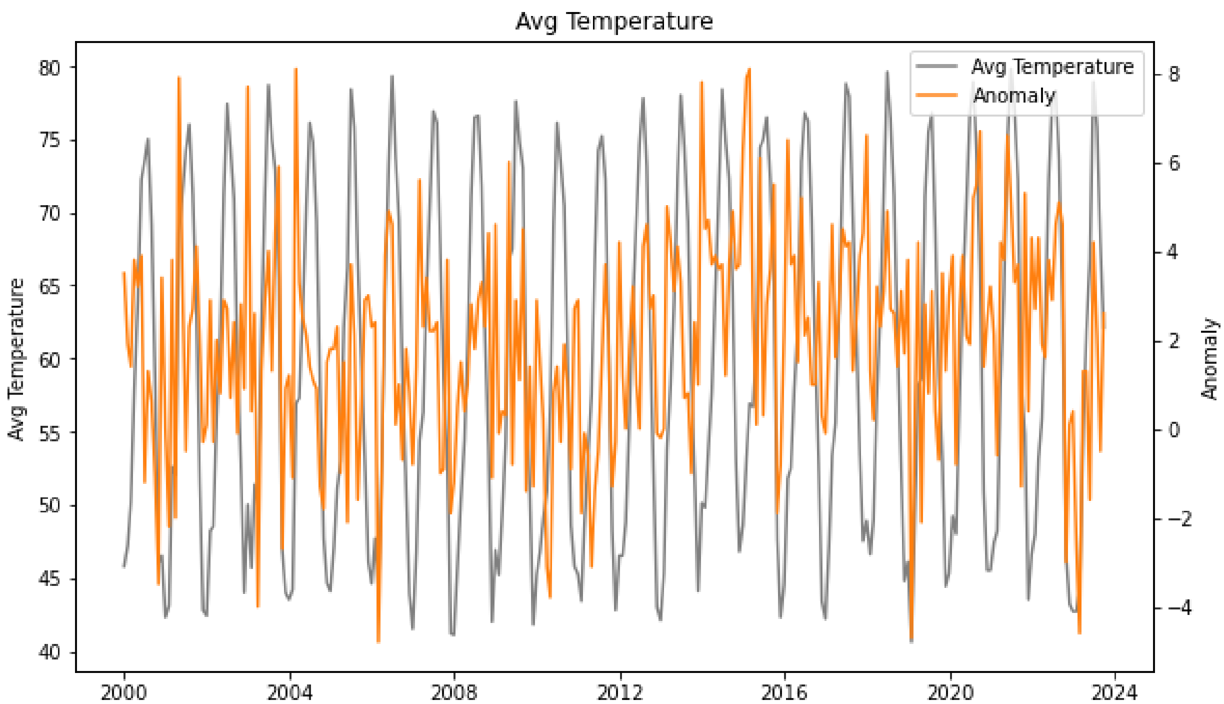
Out[55]: <matplotlib.legend.Legend at 0x2d06e6da340>



In [56]:
```
# copy dataframe for plotting
avg_temp_cali_plot = avg_temp_cali.copy()

# Convert 'Date' column to datetime format
avg_temp_cali_plot['Date'] = pd.to_datetime(avg_temp_cali_plot['Date'], format='%Y%m')

# Set 'Date' column as the index
avg_temp_cali_plot.set_index('Date', inplace=True)

#setting the figure and axes
fig,ax = plt.subplots(figsize=(10, 6))

#plotting the first plot
line1 = ax.plot(avg_temp_cali_plot.index, avg_temp_cali_plot['Value'], color='tab:gray
ax.set_ylabel('Avg Temperature')

#using twinx to generate another y axes and plotting
ax2 = ax.twinx()
line2 = ax2.plot(avg_temp_cali_plot.index, avg_temp_cali_plot['Anomaly'], color='tab:c
ax2.set_ylabel('Anomaly')
```

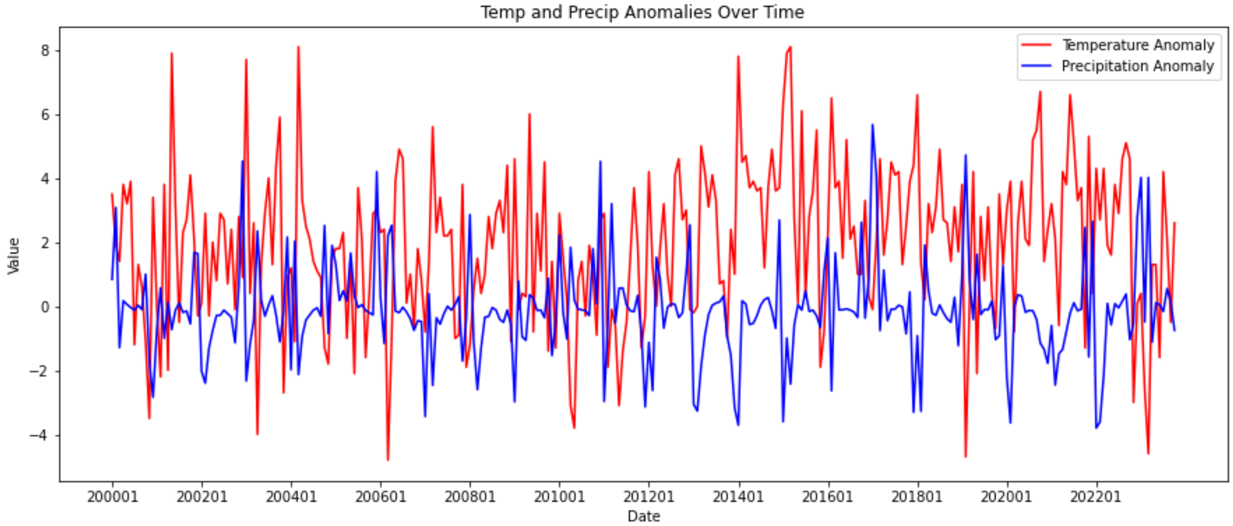
```
plt.title('Avg Temperature')

# Legend
lines = line1 + line2
labels = [line.get_label() for line in lines]
plt.legend(lines, labels, loc='upper right')
```

Out[56]: <matplotlib.legend.Legend at 0x2d0119bf550>



In [36]:

```
# Plotting the Time Series Weather Anamoly Data

plt.figure(figsize=(15, 6))
plt.plot(weather_df['Date'], weather_df['Temperature Anomaly'], label='Temperature Ano
plt.plot(weather_df['Date'], weather_df['Precipitation Anomaly'], label='Precipitation

plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Temp and Precip Anomalies Over Time')
plt.xticks(weather_df['Date'][::24])
plt.legend()
plt.show()
```



In [37]:
```
weather_join = join_df.copy() # making a copy of the join_df to use to merge to the we
weather_join.head()
```

Out[37]:

	OBJECTID	SourceOID	ABCDMisc	ADSPermiss	Containmen	ControlDat	CreatedByS	IncidentSi	D
0	1	7747595	NaN	DEFAULT	NaN	NaN	lacocad	NaN	
1	2	6384391	NaN	DEFAULT	NaN	NaN	firecode	NaN	
2	3	1383752	NaN	DEFAULT	NaN	NaN	firecode	NaN	
3	4	22499589	NaN	DEFAULT	NaN	NaN	cfcad	NaN	
4	5	23869477	NaN	DEFAULT	NaN	NaN	lacocad	NaN	

Feature for a merge: Fire Discovery Date

In [38]:
```
# Extracting Data
weather_join['Date']= weather_join['FireDiscov'].str.slice(stop=4) +weather_join['Fire
weather_join.head()
```

Out[38]:

	OBJECTID	SourceOID	ABCDMisc	ADSPermiss	Containmen	ControlDat	CreatedByS	IncidentSi	D
0	1	7747595	NaN	DEFAULT	NaN	NaN	lacocad	NaN	
1	2	6384391	NaN	DEFAULT	NaN	NaN	firecode	NaN	
2	3	1383752	NaN	DEFAULT	NaN	NaN	firecode	NaN	
3	4	22499589	NaN	DEFAULT	NaN	NaN	cfcad	NaN	
4	5	23869477	NaN	DEFAULT	NaN	NaN	lacocad	NaN	

Join 3: Temporal join between the weather data and wildfires using year and month (shown in the cell after the 'Feature 2' cell)

In [39]:
```
weather_merge = weather_join.merge(weather_df, on='Date',how='left',suffixes=('_left',
weather_merge=weather_merge.dropna(subset=['OBJECTID'])
weather_merge.head()
```

Out[39]:

	OBJECTID	SourceOID	ABCDMisc	ADSPermiss	Containmen	ControlDat	CreatedByS	IncidentSi	D
0	1	7747595	NaN	DEFAULT	NaN	NaN	lacocad	NaN	
1	2	6384391	NaN	DEFAULT	NaN	NaN	firecode	NaN	
2	3	1383752	NaN	DEFAULT	NaN	NaN	firecode	NaN	
3	4	22499589	NaN	DEFAULT	NaN	NaN	cfcad	NaN	
4	5	23869477	NaN	DEFAULT	NaN	NaN	lacocad	NaN	

Feature 2: Wildfire Count Per Temperature Feature

In [40]:

```
weather_merge= weather_merge.groupby(['Date'])['OBJECTID'].count()
weather_merge=weather_merge.reset_index()
weather_merge= weather_merge.merge(weather_df,on='Date',how='inner')
weather_merge.columns = ['Date','Wildfire Count Per Month','Average Temperature','Temp
weather_merge['Fire Count Per Temp']=weather_merge['Wildfire Count Per Month']/weather
display(weather_merge)
```

	Date	Wildfire Count Per Month	Average Temperature	Temperature Anomaly	Precipitation	Precipitation Anomaly	Fire Count Per Temp
0	201404	5	57.7	3.7	1.06	-0.57	0.086655
1	201405	87	64.6	3.9	0.35	-0.53	1.346749
2	201406	367	71.9	3.6	0.06	-0.29	5.104312
3	201407	793	78.4	3.7	0.22	0.04	10.114796
4	201408	445	74.9	1.2	0.45	0.21	5.941255
...	...	...	...	...	...	...	...
110	202306	2071	66.7	-1.6	0.40	0.05	31.049475
111	202307	3022	78.9	4.2	0.02	-0.16	38.301648
112	202308	1996	75.8	2.1	0.79	0.55	26.332454
113	202309	1329	68.0	-0.5	0.59	0.14	19.544118
114	202310	1253	62.1	2.6	0.45	-0.75	20.177134

115 rows × 7 columns

Feature 3: Wildfire Count Per Precipitation

In [41]:
```
# Now to add a similar feature, showing the wildfirecount per precipitation anomaly
weather_merge['Fire Count Per Precip']=weather_merge['Wildfire Count Per Month']/weath
display(weather_merge)
```

	Date	Wildfire Count Per Month	Average Temperature	Temperature Anomaly	Precipitation	Precipitation Anomaly	Fire Count Per Temp	Fire Count P Prec
0	201404	5	57.7	3.7	1.06	-0.57	0.086655	4.71698
1	201405	87	64.6	3.9	0.35	-0.53	1.346749	248.57142
2	201406	367	71.9	3.6	0.06	-0.29	5.104312	6116.66666
3	201407	793	78.4	3.7	0.22	0.04	10.114796	3604.54545
4	201408	445	74.9	1.2	0.45	0.21	5.941255	988.88888
...	...	...	...	...	...	...	...	...
110	202306	2071	66.7	-1.6	0.40	0.05	31.049475	5177.50000
111	202307	3022	78.9	4.2	0.02	-0.16	38.301648	151100.00000
112	202308	1996	75.8	2.1	0.79	0.55	26.332454	2526.58227
113	202309	1329	68.0	-0.5	0.59	0.14	19.544118	2252.54237
114	202310	1253	62.1	2.6	0.45	-0.75	20.177134	2784.44444

115 rows × 8 columns

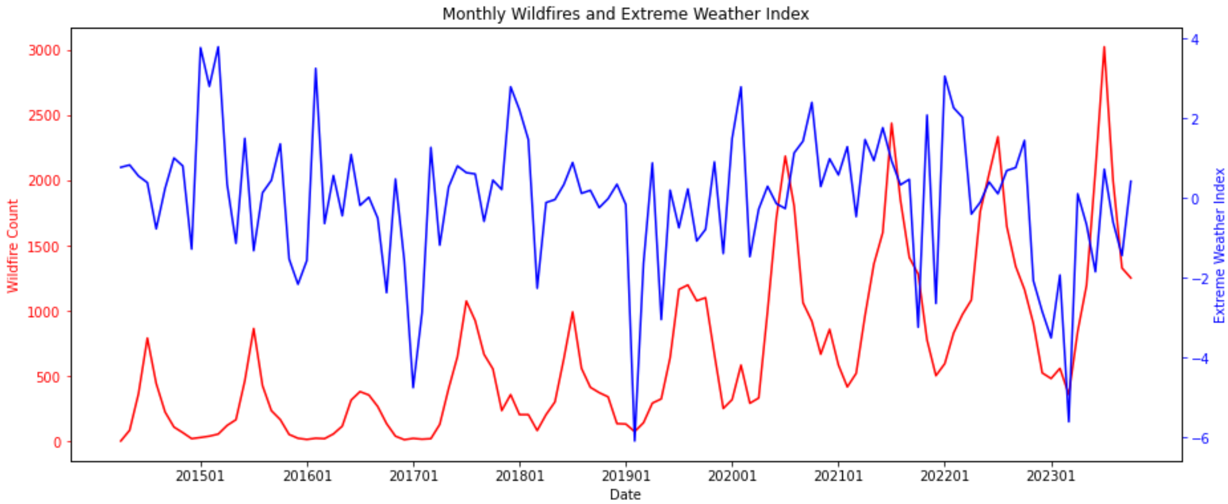
Feaure 4: Extreme Weather Index

In [42]:
```
weather_merge['Extreme Weather Index']=preprocessing.scale(weather_merge['Temperature
display(weather_merge)
```

	Date	Wildfire Count Per Month	Average Temperature	Temperature Anomaly	Precipitation	Precipitation Anomaly	Fire Count Per Temp	Fire Count P Prec
0	201404	5	57.7	3.7	1.06	-0.57	0.086655	4.71698
1	201405	87	64.6	3.9	0.35	-0.53	1.346749	248.57142
2	201406	367	71.9	3.6	0.06	-0.29	5.104312	6116.66666
3	201407	793	78.4	3.7	0.22	0.04	10.114796	3604.54545
4	201408	445	74.9	1.2	0.45	0.21	5.941255	988.88888
...	...	...	...	...	...	...	...	...
110	202306	2071	66.7	-1.6	0.40	0.05	31.049475	5177.50000
111	202307	3022	78.9	4.2	0.02	-0.16	38.301648	151100.00000
112	202308	1996	75.8	2.1	0.79	0.55	26.332454	2526.58227
113	202309	1329	68.0	-0.5	0.59	0.14	19.544118	2252.54237
114	202310	1253	62.1	2.6	0.45	-0.75	20.177134	2784.44444

115 rows × 9 columns

In [43]:
```
# Plotting Wildfire Count and Extreme WEather Index
fig, ax1 = plt.subplots(figsize=(15, 6))

ax1.plot(weather_merge['Date'], weather_merge['Wildfire Count Per Month'], label='Wild
ax1.set_xlabel('Date')
ax1.set_ylabel('Wildfire Count', color='red')
ax1.tick_params('y', colors='red')
ax1.ticklabel_format()

ax2 = ax1.twinx()
ax2.plot(weather_merge['Date'], weather_merge['Extreme Weather Index'], label='Extreme
ax2.set_ylabel('Extreme Weather Index', color='blue')
ax2.tick_params('y', colors='blue')

plt.title('Monthly Wildfires and Extreme Weather Index')
plt.xticks(weather_merge['Date'][9::12])
plt.show()
```



Feauture 5: Bucket Wildfire Size

In [44]:
```
bucketed_df = join_df.copy()
bucketed_df=bucketed_df.dropna(subset=['FinalAcres'])
acres_ranges = [0, 119, 1000000000] # using 119 as average acres burned by wildfires i
categories = ['Below Average', 'Above Average']
bucketed_df['Y/N Above Average']= pd.cut(bucketed_df['FinalAcres'], bins=acres_ranges,
display(bucketed_df)
```

	OBJECTID	SourceOID	ABCDMisc	ADSPermiss	Containmen	ControlDat	CreatedByS	Inci
44716	44731	21031590	NaN	FIREREPORTING	2021-10-11	2021-10-11	cfcad	221
49724	49741	6938378	NaN	DEFAULT	NaN	NaN	cfcad	
72909	72938	23870201	NaN	DEFAULT	2022-11-29	NaN	cfcad	
82318	82350	23844619	NaN	DEFAULT	2022-07-08	2022-07-11	cfcad	
82337	82373	23862544	NaN	DEFAULT	2022-09-26	2022-09-26	cfcad	
...	...	...	...	...	...	...	...	
286299	344801	28066130	NaN	DEFAULT	2023-11-16	2023-11-16	cfcad	
286305	344808	28066137	NaN	DEFAULT	2023-11-16	2023-11-16	cfcad	
286334	344841	28066170	NaN	DEFAULT	2023-11-17	2023-11-17	cfcad	
286348	344865	28066194	NaN	DEFAULT	2023-11-17	2023-11-17	cfcad	
286383	344901	28066230	NaN	DEFAULT	2023-11-17	NaN	cfcad	

7488 rows × 98 columns

Join 4: Joining the Weather data to the fire data tha contains the 'Close to Lines' feature

In [45]:
```
wildfire_df = join_df.copy() # making a copy to not disrupt other features if cells lc

wildfire_df['Date']= wildfire_df['FireDiscov'].str.slice(stop=4) +wildfire_df['FireDis
wildfire_df.head()
```

Out[45]:

	OBJECTID	SourceOID	ABCDMisc	ADSPermiss	Containmen	ControlDat	CreatedByS	IncidentSi	D
0	1	7747595	NaN	DEFAULT	NaN	NaN	lacocad	NaN	
1	2	6384391	NaN	DEFAULT	NaN	NaN	firecode	NaN	
2	3	1383752	NaN	DEFAULT	NaN	NaN	firecode	NaN	
3	4	22499589	NaN	DEFAULT	NaN	NaN	cfcad	NaN	
4	5	23869477	NaN	DEFAULT	NaN	NaN	lacocad	NaN	

In [46]:
```
wildfire_df = wildfire_df.merge(weather_merge, on='Date',how='inner',suffixes=('_left'
# doing inner merge because only want rows that have data for both the weather that mc
wildfire_df.head()
```

Out[46]:

	OBJECTID	SourceOID	ABCDMisc	ADSPermiss	Containmen	ControlDat	CreatedByS	IncidentSi	D
0	1	7747595	NaN	DEFAULT	NaN	NaN	lacocad	NaN	
1	618	7681575	NaN	DEFAULT	NaN	NaN	lacocad	NaN	
2	1313	7636376	NaN	DEFAULT	NaN	NaN	wildcad	1.0	
3	1406	7731174	NaN	DEFAULT	NaN	NaN	cfcad	NaN	
4	1524	7657574	NaN	DEFAULT	NaN	NaN	lacocad	NaN	

Feature 6: Extreme Weather and Close to Line Features

In [47]:

```
'''
Adding two features, one that shows if the fire was close to a transmission line and t
listed as extreme for that month, and one that indicates if the fire was not close to
weather was classified as still extreme.
'''

wildfire_df['Extreme Weather and Close to Transmission']= np.where((wildfire_df['Extre
wildfire_df['Extreme Weather and not Close to Transmission']= np.where((wildfire_df['E
wildfire_df.head()
```

Out[47]:

	OBJECTID	SourceOID	ABCDMisc	ADSPermiss	Containmen	ControlDat	CreatedByS	IncidentSi	D
0	1	7747595	NaN	DEFAULT	NaN	NaN	lacocad	NaN	
1	618	7681575	NaN	DEFAULT	NaN	NaN	lacocad	NaN	
2	1313	7636376	NaN	DEFAULT	NaN	NaN	wildcad	1.0	
3	1406	7731174	NaN	DEFAULT	NaN	NaN	cfcad	NaN	
4	1524	7657574	NaN	DEFAULT	NaN	NaN	lacocad	NaN	

Possible Next Step

We want to see how extreme weather interacts with transmission lines and wildfires.

-- to do this we can see if the proportion of wildfires started under extreme weather circumstances is higher near transmission lines than further away from transmission lines.

In [48]:
```
prop_near_lines = (wildfire_df['Extreme Weather and Close to Transmission']== True).su
prop_near_lines
```

Out[48]:
```
0.6375347029428096
```

In [49]:
```
prop_not_near_lines = (wildfire_df['Extreme Weather and not Close to Transmission']==
prop_not_near_lines
```

Out[49]:
```
0.6343324065911711
```

As we can see above, the proportion of wildfire in extreme weather conditions near powerlines (.637) is very close to the proportion of wildfires in extreme weather conditions, not near powerlines (.634), meaning extreme weather may not effect transmission line fire any more than any other type of wildfire. However, there is further analysis that could take place. We could reduce our buffer for wildfires near tranmission lines to see if we get a more

accurate representation of wildfires started by transmission lines, then rerun our numbers.