In [123…

```
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
from sklearn import datasets, linear_model, metrics
import seaborn as sns
plt.style.use('seaborn')
%matplotlib notebook
import statsmodels.api as sm
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
from datetime import datetime
from statsmodels.formula.api import ols
```

In [44]:

```
og_data = pd.read_csv("GlobalTemperatures.csv")
og_data.head()
```

Out[44]:

	date	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature	LandMax
0	1750-01-01	3.034	3.574	NaN	
1	1750-02-01	3.083	3.702	NaN	
2	1750-03-01	5.626	3.076	NaN	
3	1750-04-01	8.490	2.451	NaN	
4	1750-05-01	11.573	2.072	NaN	

In [45]:

```
temps = og_data
```

In [47]:

```
temps['date'] = pd.to_datetime(temps['date'])
```

In [265…

```
#This dataset in it’s original form consists of monthly temperature averages spanning fr
#The dataset has over 21K rows of data, but has several variables with empty values fro
#This study will begin with filtering the dataset to create a table from 1850 -  2015 a
#new data frame.
```

In [48]:

```
temps = temps[temps['date'] > '1849-12-31']
```

In [49]:

```
temps.head()
```

Out[49]:

	date	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature	LandI
1200	1850-01-01	0.749	1.105	8.242	
1201	1850-02-01	3.071	1.275	9.970	
1202	1850-03-01	4.954	0.955	10.347	
1203	1850-04-01	7.217	0.665	12.934	
1204	1850-05-01	10.004	0.617	15.655	

In [51]:

```
date = temps['date']
```

In [64]:

```
#new_dataframe = old_dataframe.filter(['Columns','you','want'], axis=1)
df = temps.filter(['date','LandAverageTemperature','LandMaxTemperature','LandAndOceanAv
```

In [71]:

```
df['date'] = pd.to_datetime(df['date'])
```

In [74]:

```
df['year'] = df['date'].dt.year
```

In [75]:

```
df.head()
```

Out[75]:

	date	LandAverageTemperature	LandMaxTemperature	LandAndOceanAverageTemperature	year
1200	1850-01-01	0.749	8.242	12.833	1850
1201	1850-02-01	3.071	9.970	13.588	1850
1202	1850-03-01	4.954	10.347	14.043	1850
1203	1850-04-01	7.217	12.934	14.667	1850
1204	1850-05-01	10.004	15.655	15.507	1850

In [150…

```
df1 = df.groupby('year',as_index=False)[['LandAverageTemperature','LandMaxTemperature',
```

In [266…

```
#At this point, the dataset still has to many monthly observations, so this block creat
#and averaged. This will reduce the dataset from 21K observations to ~166 observations.
```

In [151…

```
df1.head()
```

Out[151…

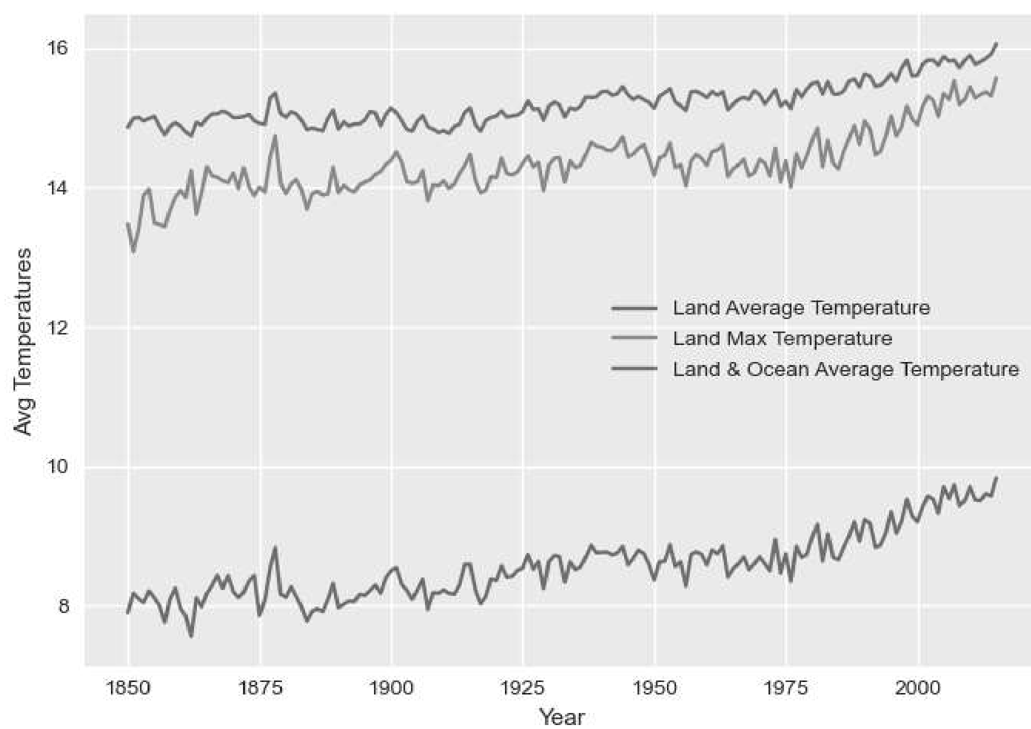
	year	LandAverageTemperature	LandMaxTemperature	LandAndOceanAverageTemperature
0	1850	7.900667	13.476667	14.867167
1	1851	8.178583	13.081000	14.991833
2	1852	8.100167	13.397333	15.006500
3	1853	8.041833	13.886583	14.955167
4	1854	8.210500	13.977417	14.991000

In [267…

```
#This block of code creates a plot of Annual Land Average Temperature (LAT), Annual Land
#(LOAT), and Land Max Temperature (LMT). The chart depicts the aggregated annual average
#The differences between the LAT and LAOT are quite significant and would be an interest
```

In [158…

```
sns.lineplot(x = df1['Year'],y = df1['LAT'], label='Land Average Temperature')
sns.lineplot(x = df1['Year'],y = df1['LMT'],label='Land Max Temperature')
sns.lineplot(x = df1['Year'],y = df1['LOAT'],label='Land & Ocean Average Temperature')
plt.xlabel('Year')
plt.ylabel('Avg Temperatures')
plt.legend()
```



Out[158…

```
<matplotlib.legend.Legend at 0x1438df66e50>
```

In [142…

```
df1_corr = df1.corr()
```

In [143…

```
df1_corr
```

Out[143…

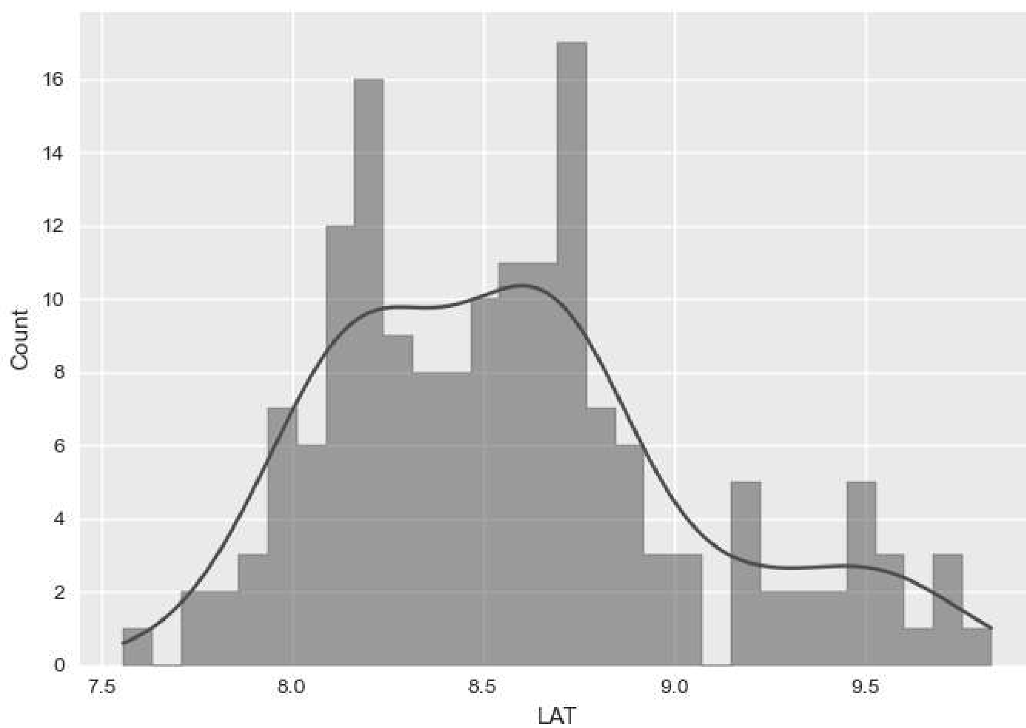
	year	LandAverageTemperature	LandMaxTemperature	LandAndO
year	1.000000	0.865682	0.813055	
LandAverageTemperature	0.865682	1.000000	0.937557	
LandMaxTemperature	0.813055	0.937557	1.000000	
LandAndOceanAverageTemperature	0.861091	0.969231	0.910445	

In [268…

```
#The next three histograms depict the frequency distribution of each variable. The idea
#often a given temperature range occurs. The LAT and LMT histograms appear to follow sor
#while the LOAT is skewed to the right.
```

In [179…

```
sns.histplot(data = df1['LAT'],color = 'red',label = 'LAT',kde=True,element='step',bins
```

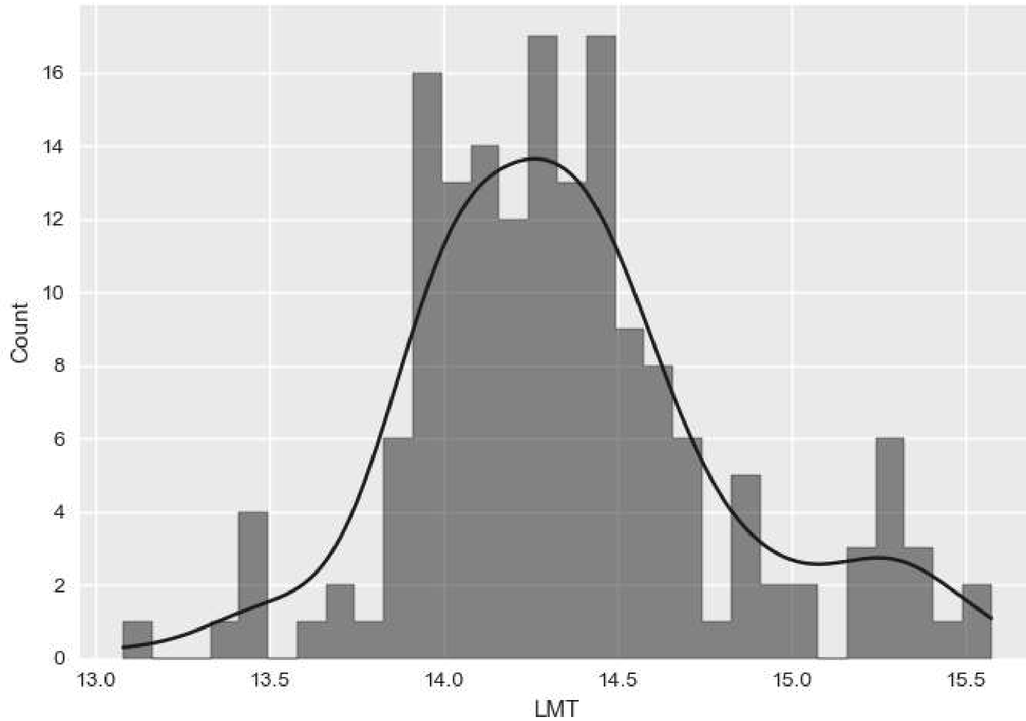


Out[179…

```
<AxesSubplot:xlabel='LAT', ylabel='Count'>
```

In [182…

```
sns.histplot(data = df1['LMT'],color='blue',label = 'LMT',element='step',bins=30,kde=Tr
```

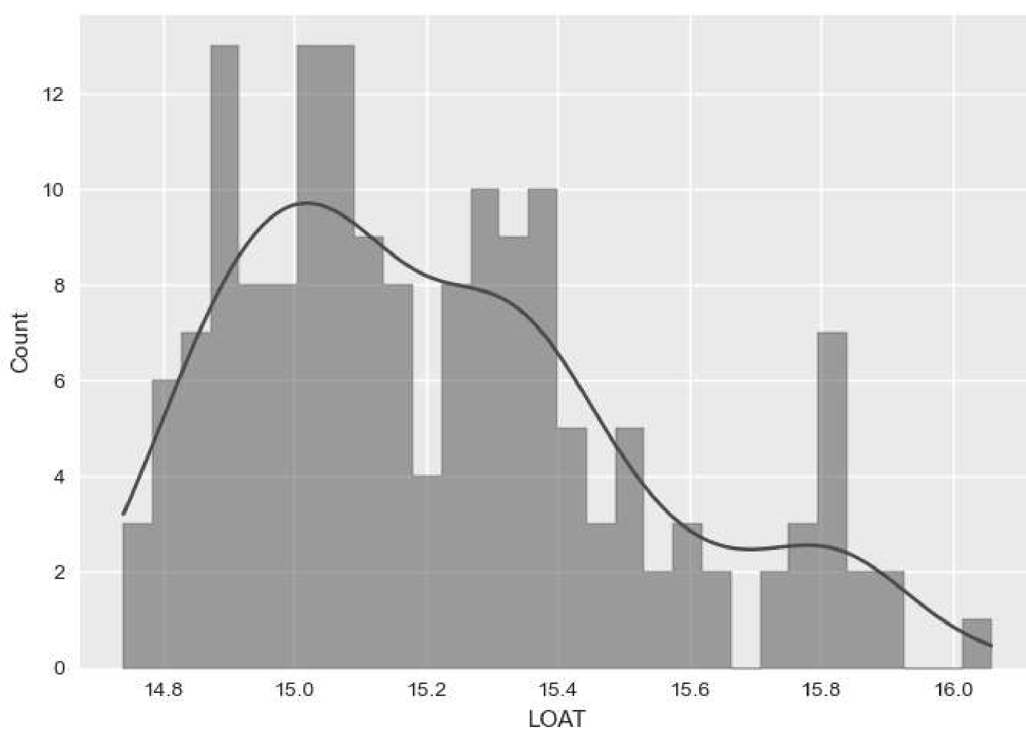


Out[182…

```
<AxesSubplot:xlabel='LMT', ylabel='Count'>
```

In [183…

```
sns.histplot(data = df1['LOAT'],color='green',label = 'LOAT',element='step',bins=30,kde
```

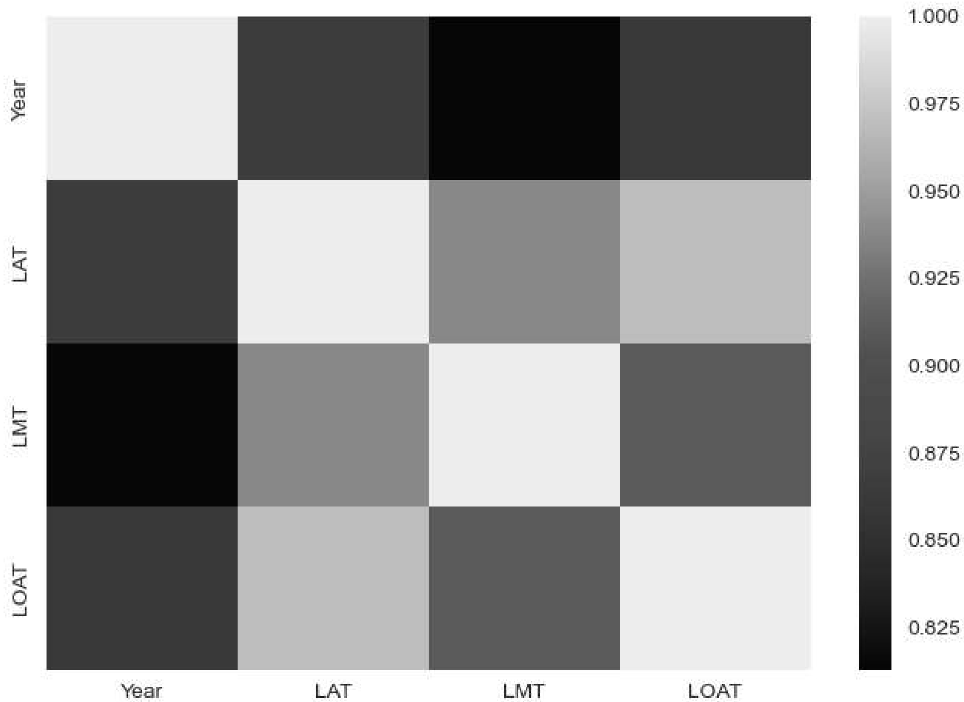


Out[183...

```
<AxesSubplot:xlabel='LOAT', ylabel='Count'>
```

In [121...

```
sns.heatmap(df1_corr)
```



Out[121...

```
<AxesSubplot:>
```

In [153...

```
#LandAverageTemperature LandMaxTemperature      LandAndOceanAverageTemperature
df1.columns = ['Year','LAT','LMT','LOAT']
```

In [269...

```
#The next bloc of code creates linear models for LAT, LMT, and LOAT against the year va
#to compare the each variable against the linear regression line. There appears to be s
#the actuals. This prompted a look at a linear regression of LAT against LMT and LOAT.
#significant and has a R2 of 95.72, which indicates that 95% of the variation in the LA
#attributable to the model itself. Approximately 4% of the variation in the model is by
```

In [146...

```
result = sm.OLS(df1['LAT'],df1[['LMT','LOAT']]).fit()
```

In [147...

```
print(result.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                    LAT   R-squared (uncentered):                   0.999
Model:                            OLS   Adj. R-squared (uncentered):              0.999
Method:                 Least Squares   F-statistic:                          1.177e+05
Date:                Tue, 15 Mar 2022   Prob (F-statistic):                  1.29e-259
Time:                        14:19:38   Log-Likelihood:                          10.942
```

```
No. Observations:                 166   AIC:                            -17.88
Df Residuals:                     164   BIC:                            -11.66
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
LMT            0.9632      0.079     12.157      0.000       0.807       1.120
LOAT          -0.3449      0.075     -4.614      0.000      -0.493      -0.197
==============================================================================
Omnibus:                       20.643   Durbin-Watson:                   0.654
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               41.914
Skew:                          -0.561   Prob(JB):                     7.91e-10
Kurtosis:                       5.191   Cond. No.                         129.
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a con
stant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

In [270…

```
#The following blocks of code will transition to the rolling average portion of the stu
#rolling average was created, and needed to have nine “NA” observations removed. This r
#to 157. This chart is a stand-alone depiction of the LOAT 10-year rolling average. The
#overall flow of the rolling averages against the periods.
```

In [215…

```
sma = df1
sma.head()
```

Out[215…

	Year	LAT	LMT	LOAT
0	1850	7.900667	13.476667	14.867167
1	1851	8.178583	13.081000	14.991833
2	1852	8.100167	13.397333	15.006500
3	1853	8.041833	13.886583	14.955167
4	1854	8.210500	13.977417	14.991000

In [221…

```
sma['10-Year'] = sma.LOAT.rolling(10).mean()
```

```
C:\Users\JOSHUA~1\AppData\Local\Temp/ipykernel_13704/3296305988.py:1: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  sma['10-Year'] = sma.LOAT.rolling(10).mean()
```

In [222…

```
sma = sma.dropna()
sma.head(15)
```

Out[222…

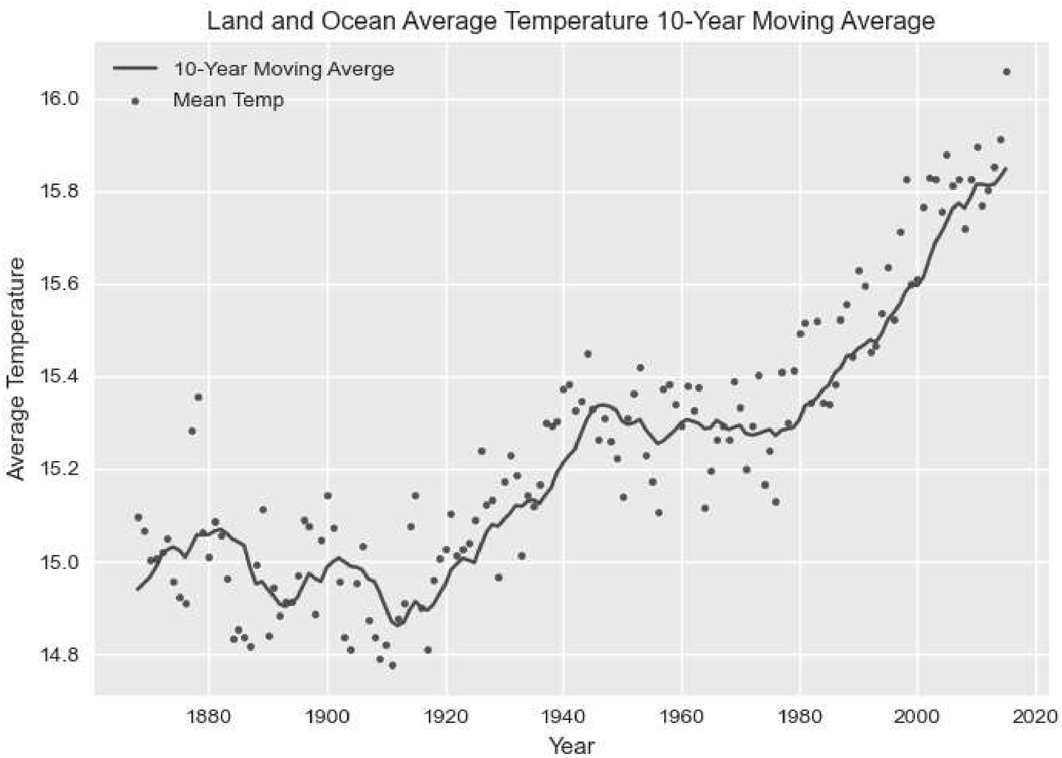
	Year	LAT	LMT	LOAT	10-Year
18	1868	8.247917	14.097917	15.096917	14.939508
19	1869	8.432083	14.069500	15.065500	14.953067
20	1870	8.201333	14.210083	15.004333	14.965208
21	1871	8.115083	13.983833	15.005917	14.985733
22	1872	8.193833	14.285083	15.019333	15.013658
23	1873	8.351083	14.010167	15.049250	15.025008
24	1874	8.433500	13.883500	14.957000	15.031058
25	1875	7.859583	14.008417	14.921917	15.023583
26	1876	8.080083	13.934833	14.909417	15.008825
27	1877	8.539583	14.430333	15.282667	15.031225
28	1878	8.829750	14.742167	15.357417	15.057275
29	1879	8.165833	14.065750	15.064417	15.057167
30	1880	8.118750	13.913417	15.008667	15.057600
31	1881	8.270917	14.050417	15.087167	15.065725
32	1882	8.128917	14.114250	15.056583	15.069450

In [271…

```
#This chart is a scatter plot of the mean observations, a red line of the moving averag
#the Land and Ocean Average Annual Temperature is been at or above the 10-year rolling
#has been well above the 10-year rolling average since the 2000’s.
```

In [260…

```
plt.scatter(sma['Year'],sma['LOAT'],color='black',label = "Mean Temp",alpha=0.65,s=9.5)
plt.plot(sma['Year'],sma['10-Year'],color='red',linestyle='-',label="10-Year Moving Ave
plt.title("Land and Ocean Average Temperature 10-Year Moving Average")
plt.xlabel("Year")
plt.ylabel("Average Temperature")
plt.legend()
```



Out[260…

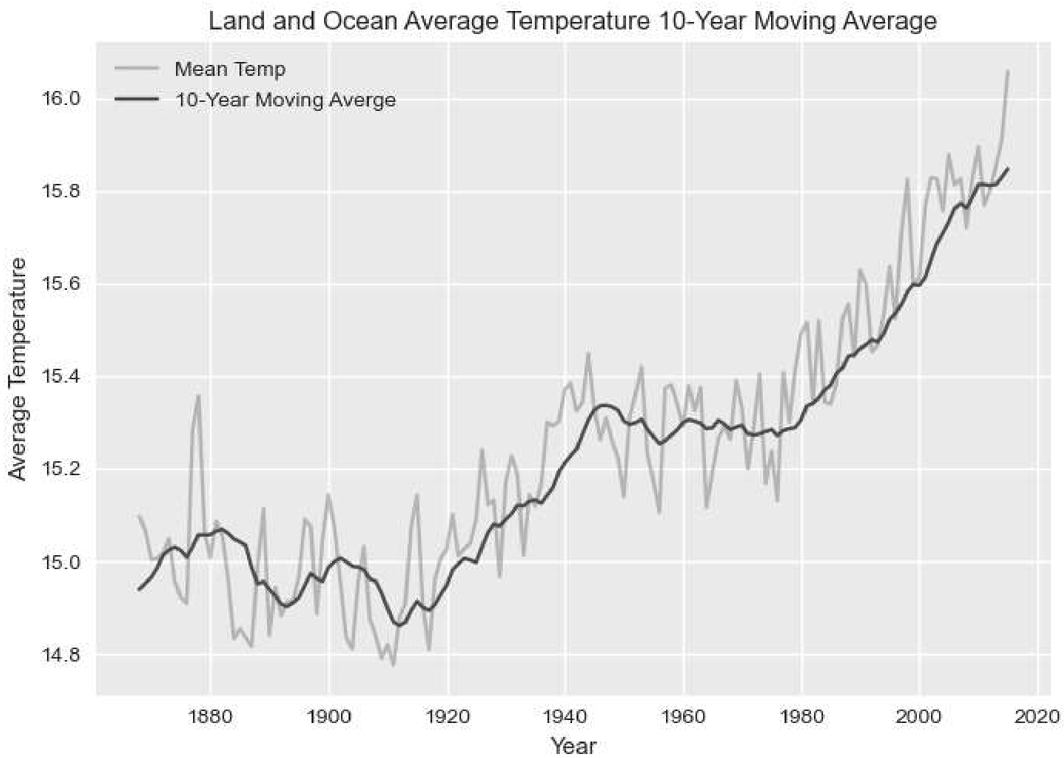
```
<matplotlib.legend.Legend at 0x1439ebb8fa0>
```

In [272…

```
#In this next chart, the actual observations are depicted in the grey line and the roll
#red line. This chart allows the reader to compare the rolling average to the actual ob:
#appears to be on the low end of many of the observations in the 2000’s. Research would
```

In [247…

```
plt.plot(sma['Year'],sma['LOAT'],color='grey',label = "Mean Temp",alpha=0.5)
plt.plot(sma['Year'],sma['10-Year'],color='red',linestyle='-',label="10-Year Moving Ave
plt.title("Land and Ocean Average Temperature 10-Year Moving Average")
plt.xlabel("Year")
plt.ylabel("Average Temperature")
plt.legend()
```



Out[247…

```
<matplotlib.legend.Legend at 0x1439d725280>
```

In [ ]:

In [ ]: