Kansas Water Data and Visualizations

March 14, 2024

1 E483 and E583 Kansas Water Quality Project

Caro Champion

Melanie Erkman

Erik Kreider

Greg Shoda

Muhammad Siddiq

[1]:
```
import pandas as pd
import numpy as np
import re
```

[2]:
```
water = pd.read_excel(r"C:\Users\gregs\OneDrive\Documents\Grad School\Spring␣
 ↪2024\Visual Analytics\Water␣
 ↪Project\kansas_water_quality_physchem_data_2018-2023 v2.xlsx")
```

[4]:
```
fips = pd.read_csv(r"C:\Users\gregs\OneDrive\Documents\Grad School\Spring␣
 ↪2024\Visual Analytics\Water Project\Kansas Fips Codes.txt", delimiter = '|')
```

[5]:
```
water = pd.merge(water, fips, left_on = 'LocationCounty', right_on =␣
 ↪'COUNTYNAME')
```

[6]:
```
water['ResultMeasureValue'] = pd.to_numeric(water['ResultMeasureValue'], errors␣
 ↪= 'coerce')
```

1.0.1 Seeing what states samples were taken in

[7]:
```
water['Location State'].unique()
```

[7]:
```
array(['KS', 'MO'], dtype=object)
```

1.0.2 Filtering the samples to just the state of Kansas

[8]:
```
water = water.loc[water['Location State'] == 'KS']
```

1.0.3 Finding the number of unique sites tested each year

[9]:

```
water['Year'] = water['ActivityStartDate'].dt.year
```

[10]:

```
for y in water['Year'].unique():
    specific_year = water.loc[water['Year'] == y]
    count = len(specific_year['MonitoringLocationIdentifier'].unique())

    print(f"There were {count} sites tested in {y}.")
```

```
There were 156 sites tested in 2018.
There were 109 sites tested in 2019.
There were 102 sites tested in 2021.
There were 66 sites tested in 2023.
There were 78 sites tested in 2020.
There were 117 sites tested in 2022.
```

1.0.4 Plotting our site locations

[12]:

```
import geopandas as gpd
import matplotlib.pyplot as plt
```

Kansas Shape file downloaded from: https://catalog.data.gov/dataset/tiger-line-shapefile-2019-state-kansas-current-county-subdivision-state-based

[42]:

```
usa = gpd.read_file(r"C:
 ↪\Users\gregs\Downloads\tl_2023_us_county\tl_2023_us_county.shp")

kansas = usa.loc[usa['STATEFP'] == '20']
```

[ ]:

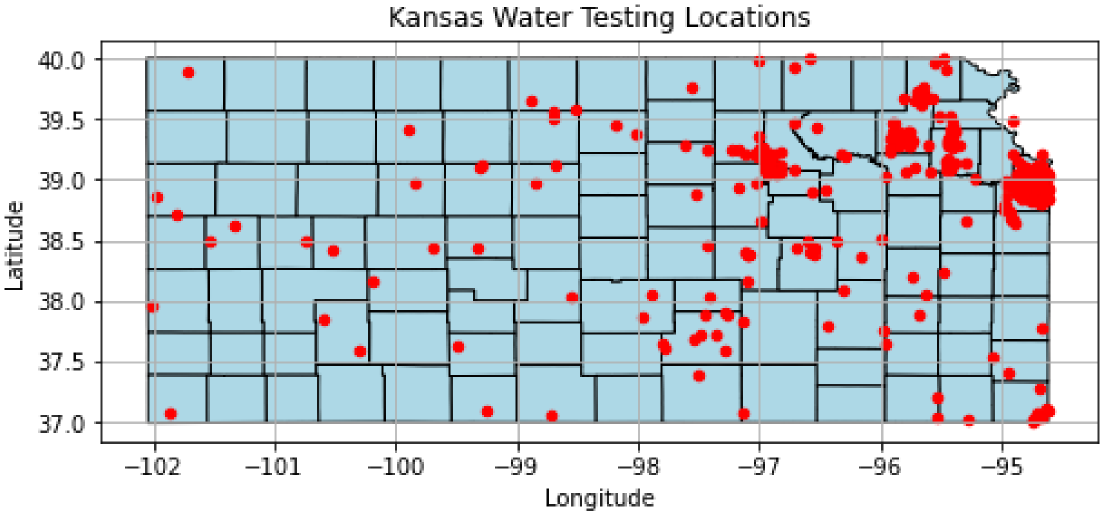
```

```

[35]:

```
latitude = list(water['LocationLatitudeMeasure'].unique())
longitude = list(water['LocationLongitudeMeasure'].unique())

plot_data = {'latitude': latitude,
             'longitude': longitude}

plot_df = pd.DataFrame(plot_data)

gdf = gpd.GeoDataFrame(plot_df, geometry = gpd.points_from_xy(plot_df.
 ↪longitude, plot_df.latitude))
```

[36]:

```
fig, ax = plt.subplots(figsize = (8,6))
kansas.plot(ax=ax, color = 'lightblue', edgecolor = 'black')
gdf.plot(ax = ax, color = 'red', markersize = 20)
plt.title('Kansas Water Testing Locations')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.grid(True)
plt.show()
```



1.0.5 Seeing the number of unique characteristics tested for in the water samples

[15]:
```
count = len(water['CharacteristicName'].unique())

print(f'There are {count} different characteristics that were tested for across␣
 ↪all water samples')
```

```
There are 1057 different characteristics that were tested for across all water
samples
```

1.0.6 While there are 1,057 different charactereistics that were tested over the course of all sites at all, we have identified four different characteristics we would like to key in on: Escherichia Coli, Dissolved Oxygen, Nitrogen, Phosphorous

[16]:
```
def character(char, df):
    #Filtering down to only the characteristic
    filter_df = df.loc[df['CharacteristicName'] == char]

    year_avg = filter_df.groupby('Year')['ResultMeasureValue'].mean().
 ↪reset_index()
    site_avg = filter_df.
 ↪groupby('MonitoringLocationIdentifier')['ResultMeasureValue'].mean().
 ↪reset_index()
    site_avg_per_year = filter_df.groupby(['Year',␣
 ↪'MonitoringLocationIdentifier'])['ResultMeasureValue'].mean().reset_index()
```

```
    county_avg = filter_df.groupby('COUNTYFP')['ResultMeasureValue'].mean()
    county_avg_per_year = filter_df.groupby(['Year',␣
↪'COUNTYFP'])['ResultMeasureValue'].mean()
    return year_avg, site_avg, site_avg_per_year, county_avg,␣
↪county_avg_per_year
```

[17]:

```
def characteristic_df(char):
    names = [char + '_year_avg', char+'_site_avg', char+'_site_avg_per_year',␣
↪char + '_county_avg', char + '_county_avg_per_year']
    return names
```

1.0.7 Working with Nitrogen

[63]:

```
nitrogen_df = water.loc[water['CharacteristicName'] == 'Nitrogen']
```

[66]:

```
nitrogen_year_avg = nitrogen_df.groupby('Year')['ResultMeasureValue'].mean().
↪reset_index()
```

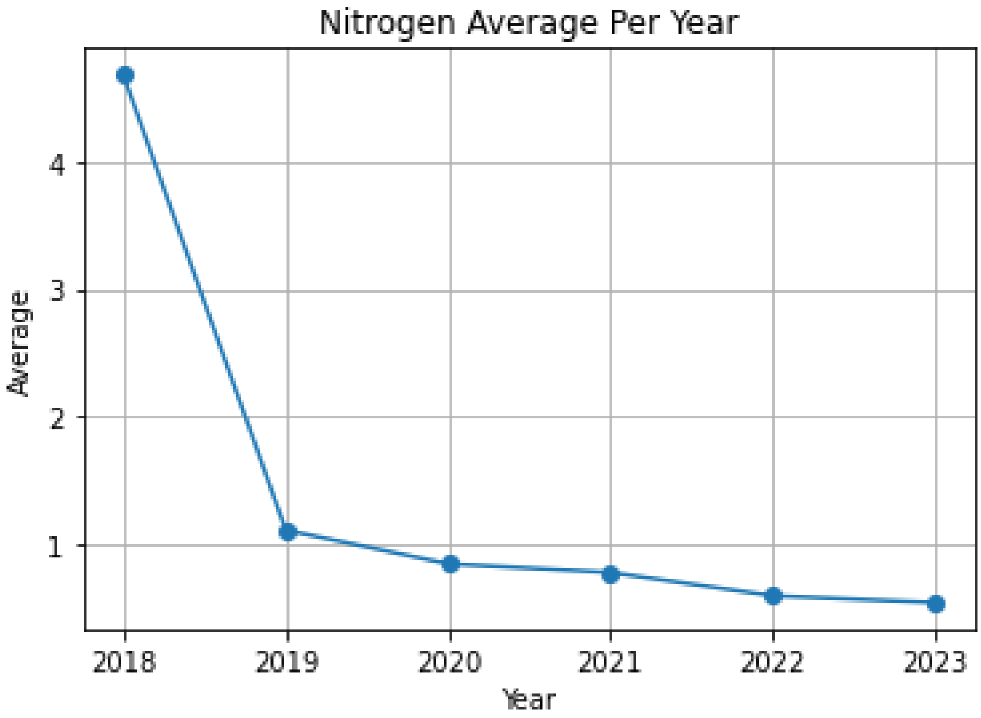
[67]:

```
nitrogen_year_avg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 2 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Year                6 non-null      int64
 1   ResultMeasureValue  6 non-null      float64
dtypes: float64(1), int64(1)
memory usage: 224.0 bytes
```

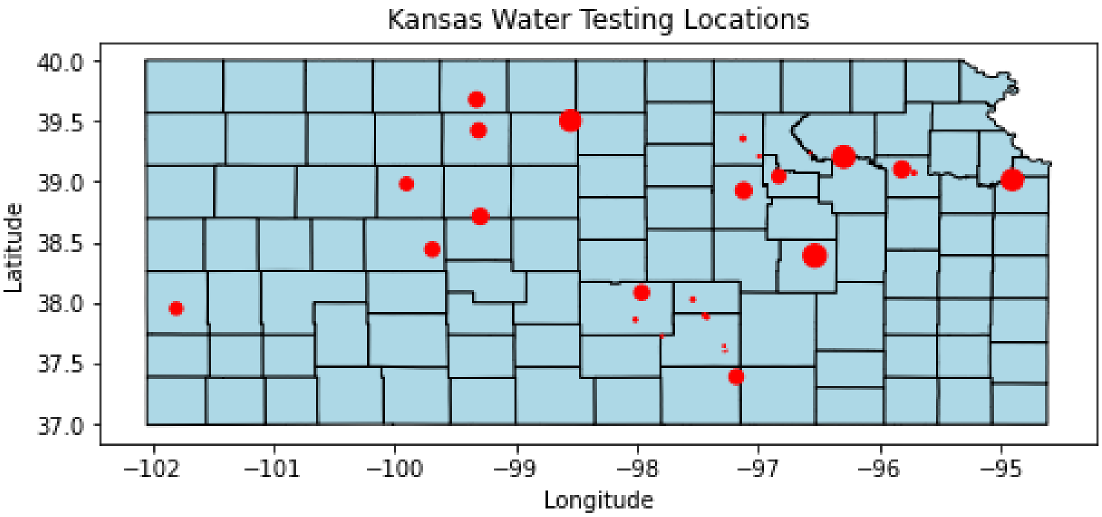
[70]:

```
x_values = nitrogen_year_avg['Year']
y_values = nitrogen_year_avg['ResultMeasureValue']

plt.plot(x_values, y_values, marker = 'o')
plt.title('Nitrogen Average Per Year')
plt.xlabel('Year')
plt.ylabel('Average')
plt.grid(True)
plt.show()
```



[73]:
```
nitrogen_site_avg = nitrogen_df.groupby(['MonitoringLocationIdentifier','LocationLatitudeMeasure', 'LocationLongitudeMeasure'])['ResultMeasureValue'].mean().reset_index()
```

[101]:
```
latitude = list(nitrogen_site_avg['LocationLatitudeMeasure'])
longitude = list(nitrogen_site_avg['LocationLongitudeMeasure'])
measure = list(nitrogen_site_avg['ResultMeasureValue'])

plot_data = {'latitude': latitude,
             'longitude': longitude,
             'measure': measure}

plot_df = pd.DataFrame(plot_data)

gdf = gpd.GeoDataFrame(plot_df, geometry = gpd.points_from_xy(plot_df.longitude, plot_df.latitude))
fig, ax = plt.subplots(figsize = (8,6))
kansas.plot(ax=ax, color = 'lightblue', edgecolor = 'black')
gdf.plot(ax = ax, color = 'red', markersize = plot_df['measure']*4)
plt.title('Kansas Water Testing Locations')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```

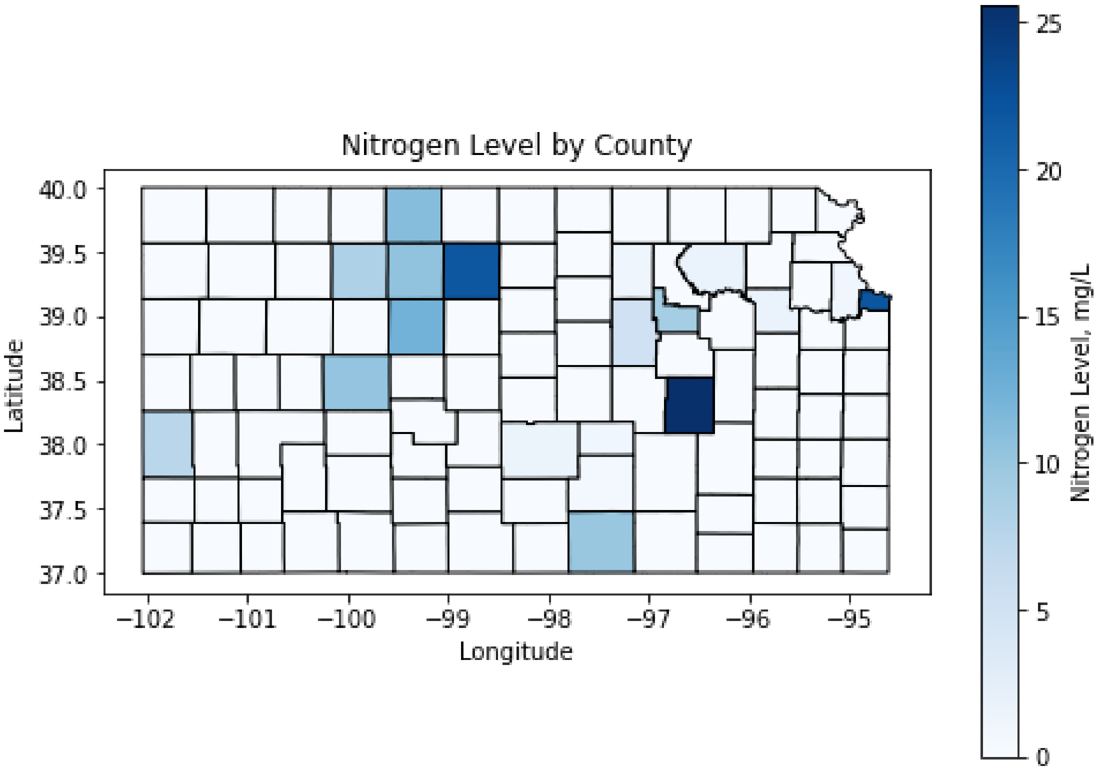


[93]:
```
nitrogen_county_avg = nitrogen_df.groupby('COUNTYFP')['ResultMeasureValue'].
 ↪mean().reset_index()
```

[95]:
```
kansas_nitrogen = pd.merge(kansas2, nitrogen_county_avg, on = "COUNTYFP", how =␣
 ↪'left')
```

[97]:
```
kansas_nitrogen['ResultMeasureValue'] = kansas_nitrogen['ResultMeasureValue'].
 ↪fillna(0)
```

[100]:
```
fig, ax = plt.subplots(figsize = (8,6))
kansas_nitrogen.plot(column = 'ResultMeasureValue', ax= ax, cmap = 'Blues',␣
 ↪edgecolor = 'black', legend = True, legend_kwds={'label':'Nitrogen Level, mg/
 ↪L'})
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Nitrogen Level by County')
plt.show()
```

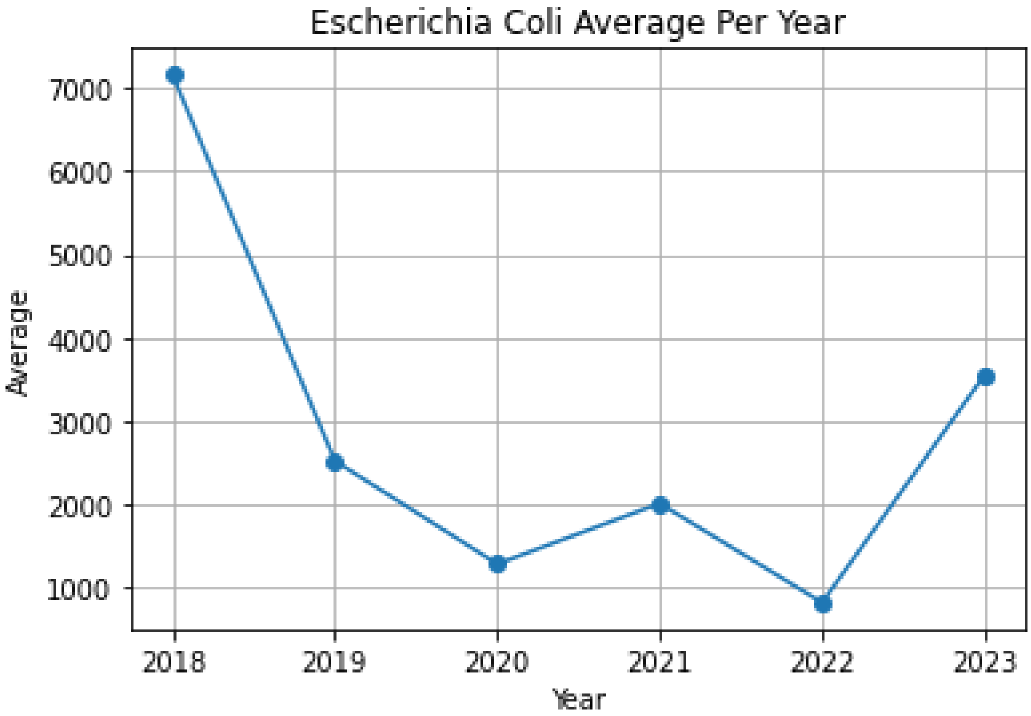


1.0.8 Working with Escherichia Coli

[102]:
```
escher_df = water.loc[water['CharacteristicName'] == 'Escherichia coli']
```

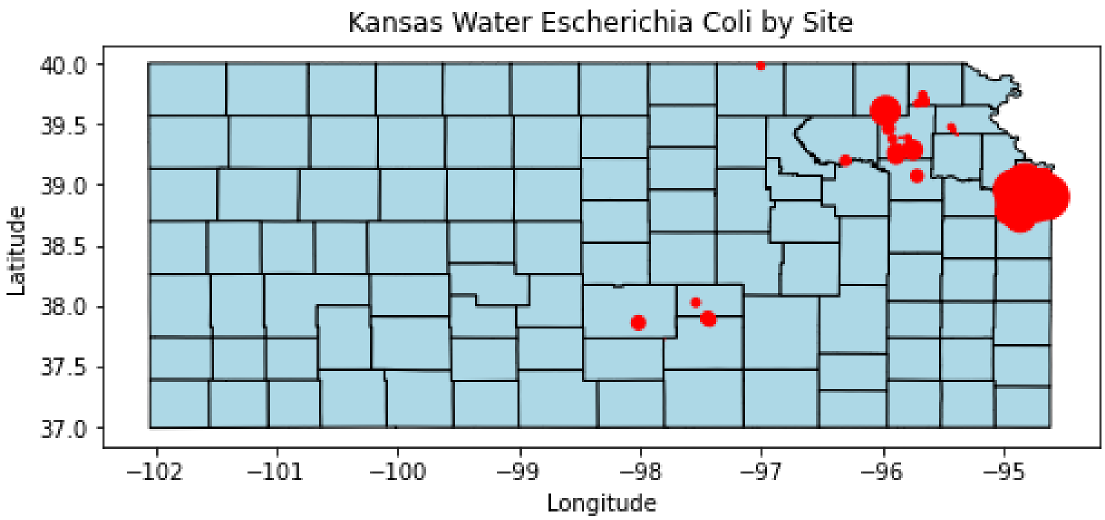
[103]:
```
escher_year_avg = escher_df.groupby('Year')['ResultMeasureValue'].mean().
 ↪reset_index()
```

[104]:
```
x_values = escher_year_avg['Year']
y_values = escher_year_avg['ResultMeasureValue']

plt.plot(x_values, y_values, marker = 'o')
plt.title('Escherichia Coli Average Per Year')
plt.xlabel('Year')
plt.ylabel('Average')
plt.grid(True)
plt.show()
```



[105]:
```
escher_site_avg = escher_df.
 ↪groupby(['MonitoringLocationIdentifier','LocationLatitudeMeasure',␣
 ↪'LocationLongitudeMeasure'])['ResultMeasureValue'].mean().reset_index()
```

[109]:
```
latitude = list(escher_site_avg['LocationLatitudeMeasure'])
longitude = list(escher_site_avg['LocationLongitudeMeasure'])
measure = list(escher_site_avg['ResultMeasureValue'])

plot_data = {'latitude': latitude,
             'longitude': longitude,
             'measure': measure}

plot_df = pd.DataFrame(plot_data)

gdf = gpd.GeoDataFrame(plot_df, geometry = gpd.points_from_xy(plot_df.
 ↪longitude, plot_df.latitude))
fig, ax = plt.subplots(figsize = (8,6))
kansas.plot(ax=ax, color = 'lightblue', edgecolor = 'black')
gdf.plot(ax = ax, color = 'red', markersize = plot_df['measure']*0.01)
plt.title('Kansas Water Escherichia Coli by Site')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```

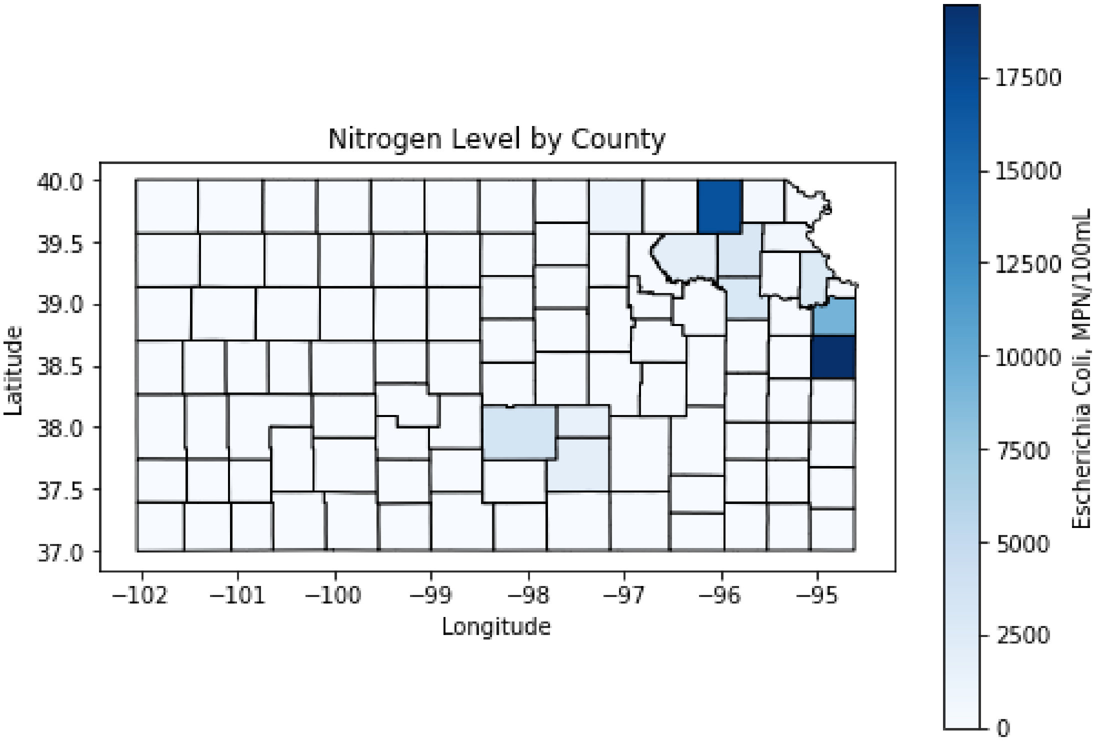


```
[120]: escher_county_avg = escher_df.groupby('COUNTYFP')['ResultMeasureValue'].mean().
        ↪reset_index()
```

```
[121]: kansas_escher = pd.merge(kansas2, escher_county_avg, on = "COUNTYFP", how =␣
        ↪'left')
```

```
[122]: kansas_escher['ResultMeasureValue'] = kansas_escher['ResultMeasureValue'].
        ↪fillna(0)
```

```
[123]: fig, ax = plt.subplots(figsize = (8,6))
       kansas_escher.plot(column = 'ResultMeasureValue', ax= ax, cmap = 'Blues',␣
        ↪edgecolor = 'black', legend = True, legend_kwds={'label':'Escherichia Coli,␣
        ↪MPN/100mL'})
       plt.xlabel('Longitude')
       plt.ylabel('Latitude')
       plt.title('Nitrogen Level by County')
       plt.show()
```

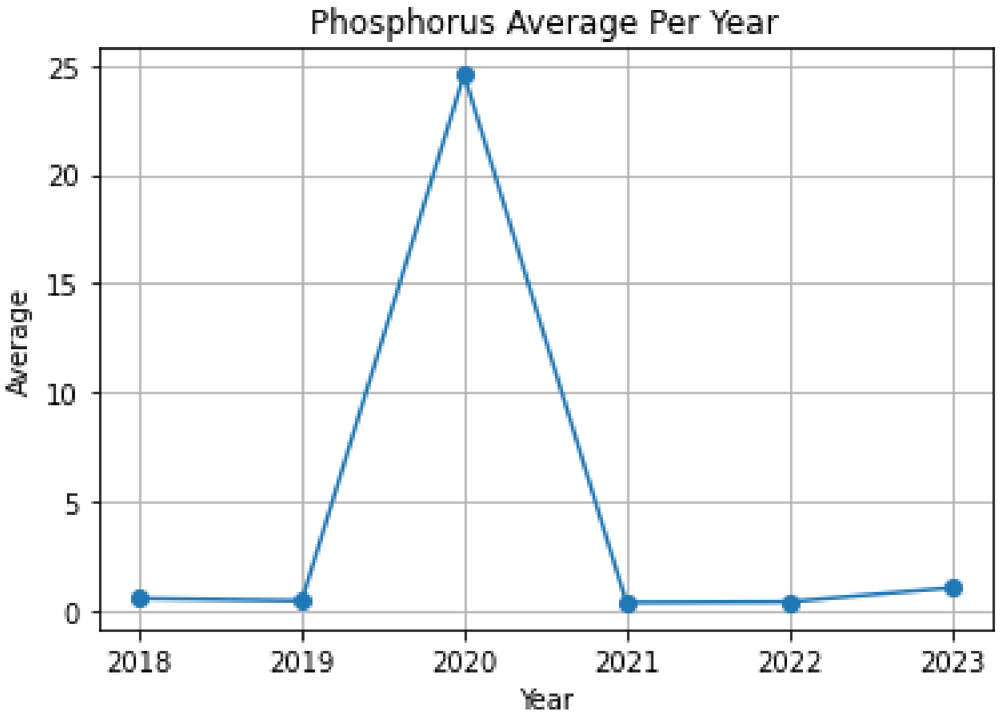


1.0.9 Working with Phosphorus

[127]:
```
phospho_df = water.loc[water['CharacteristicName'] == 'Phosphorus']
```

[128]:
```
phospho_year_avg = phospho_df.groupby('Year')['ResultMeasureValue'].mean().
 ↪reset_index()
```

[129]:
```
x_values = phospho_year_avg['Year']
y_values = phospho_year_avg['ResultMeasureValue']

plt.plot(x_values, y_values, marker = 'o')
plt.title('Phosphorus Average Per Year')
plt.xlabel('Year')
plt.ylabel('Average')
plt.grid(True)
plt.show()
```

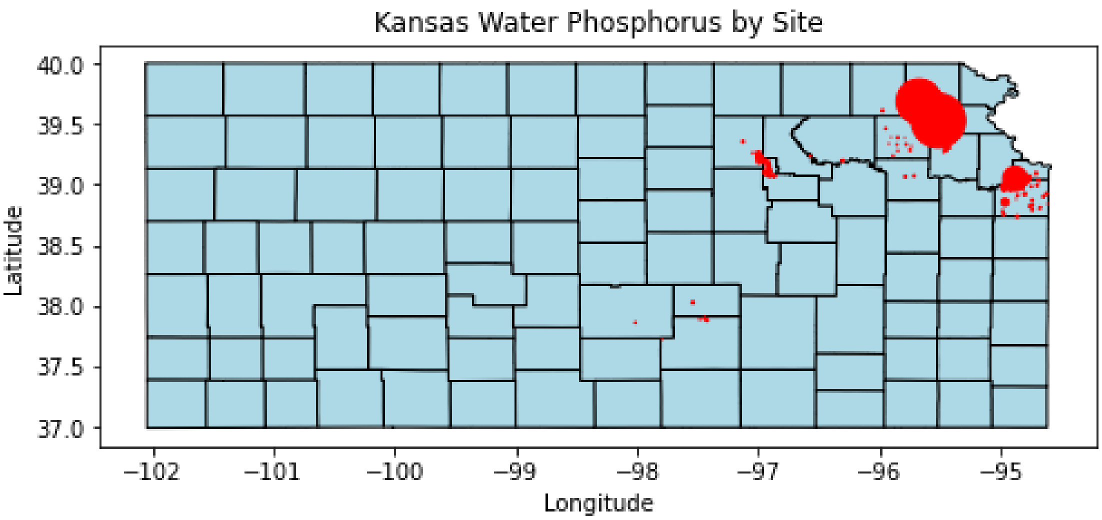


[130]:

```
phospho_site_avg = phospho_df.
 ↪groupby(['MonitoringLocationIdentifier','LocationLatitudeMeasure',␣
 ↪'LocationLongitudeMeasure'])['ResultMeasureValue'].mean().reset_index()
```

[132]:

```
latitude = list(phospho_site_avg['LocationLatitudeMeasure'])
longitude = list(phospho_site_avg['LocationLongitudeMeasure'])
measure = list(phospho_site_avg['ResultMeasureValue'])

plot_data = {'latitude': latitude,
             'longitude': longitude,
             'measure': measure}

plot_df = pd.DataFrame(plot_data)

gdf = gpd.GeoDataFrame(plot_df, geometry = gpd.points_from_xy(plot_df.
 ↪longitude, plot_df.latitude))
fig, ax = plt.subplots(figsize = (8,6))
kansas.plot(ax=ax, color = 'lightblue', edgecolor = 'black')
gdf.plot(ax = ax, color = 'red', markersize = plot_df['measure']*2)
plt.title('Kansas Water Phosphorus by Site')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```

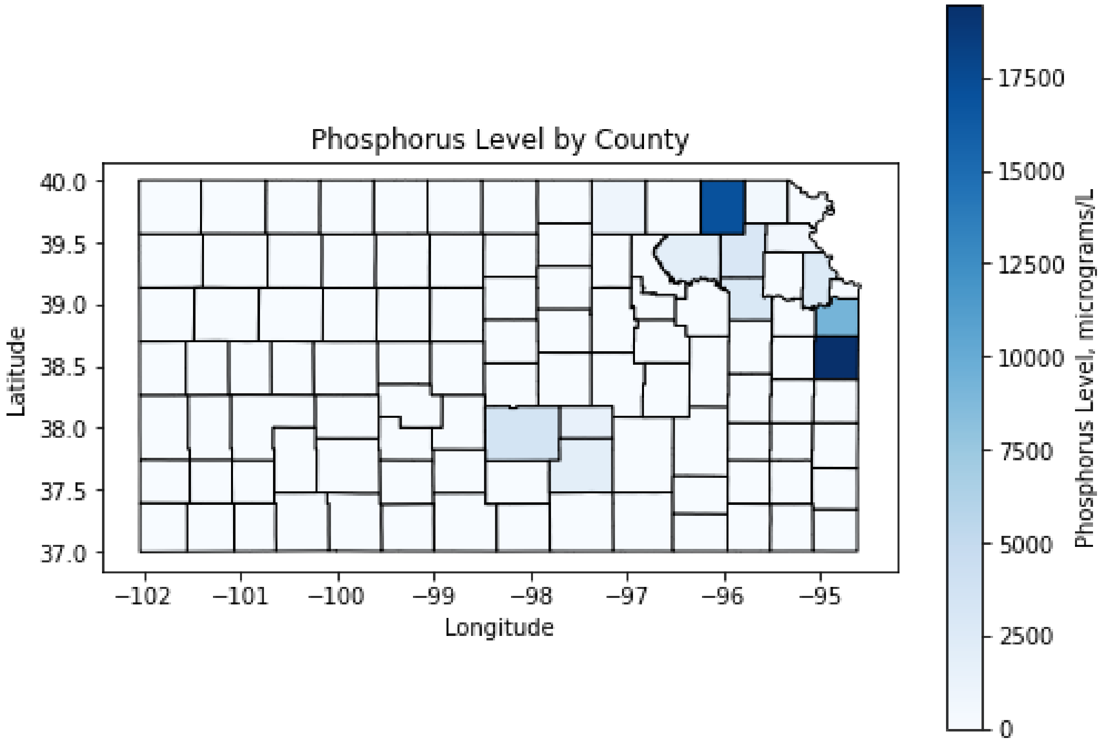


```
[133]: phospho_county_avg = phospho_df.groupby('COUNTYFP')['ResultMeasureValue'].
        ↪mean().reset_index()
```

```
[134]: kansas_phospho = pd.merge(kansas2, phospho_county_avg, on = "COUNTYFP", how =␣
        ↪'left')
```

```
[135]: kansas_phospho['ResultMeasureValue'] = kansas_phospho['ResultMeasureValue'].
        ↪fillna(0)
```

```
[137]: fig, ax = plt.subplots(figsize = (8,6))
       kansas_escher.plot(column = 'ResultMeasureValue', ax= ax, cmap = 'Blues',␣
        ↪edgecolor = 'black', legend = True, legend_kwds={'label':'Phosphorus Level,␣
        ↪micrograms/L'})
       plt.xlabel('Longitude')
       plt.ylabel('Latitude')
       plt.title('Phosphorus Level by County')
       plt.show()
```

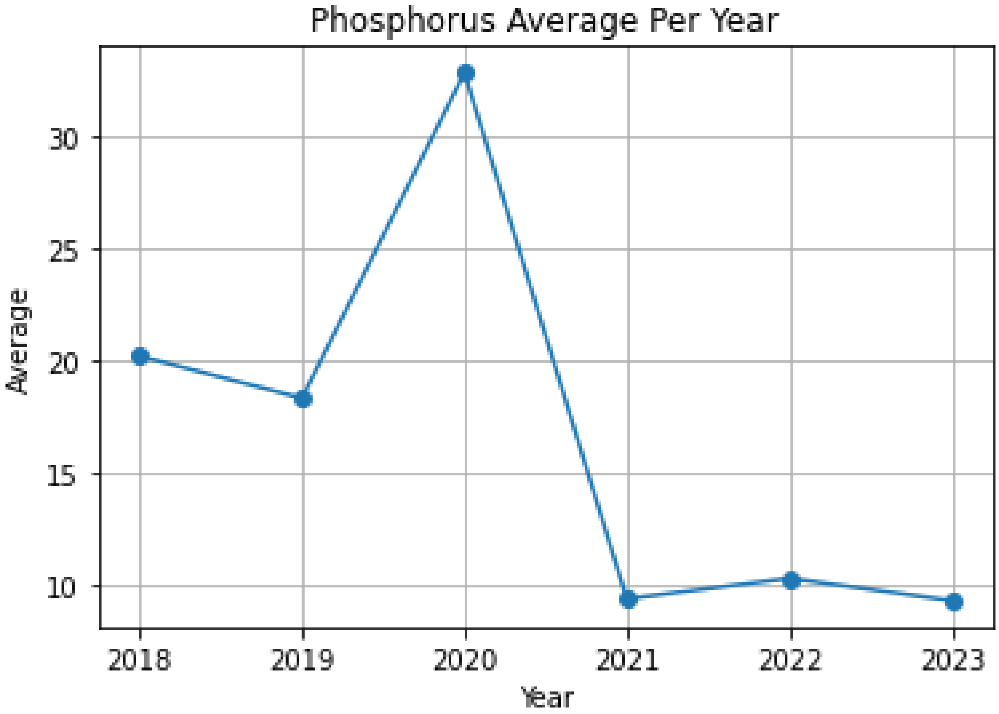


1.0.10 Working with Dissolved Oxygen

```
[138]: ox_df = water.loc[water['CharacteristicName'] == 'Dissolved oxygen (DO)']
```

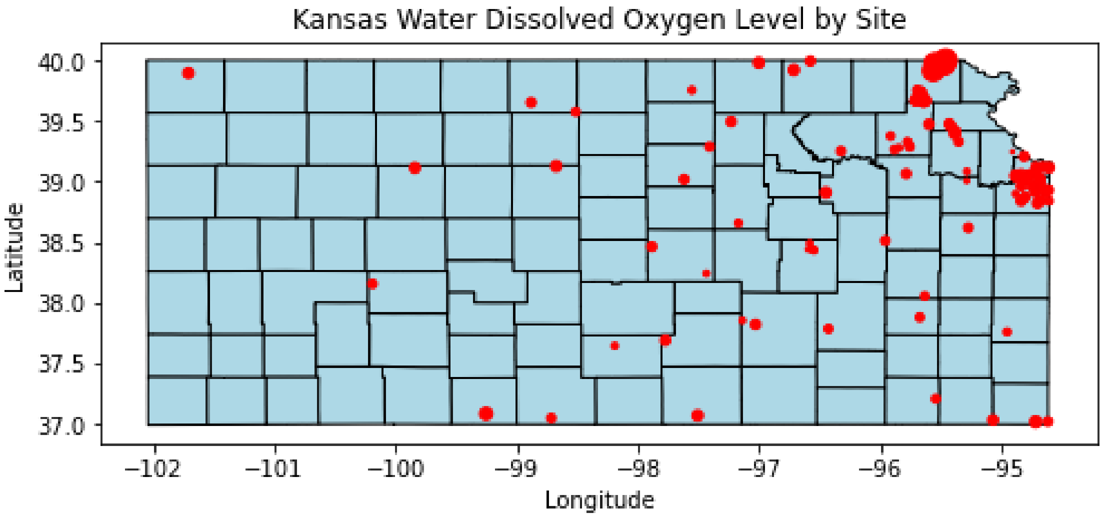
```
[139]: ox_year_avg = ox_df.groupby('Year')['ResultMeasureValue'].mean().reset_index()
```

```
[140]: x_values = ox_year_avg['Year']
       y_values = ox_year_avg['ResultMeasureValue']

       plt.plot(x_values, y_values, marker = 'o')
       plt.title('Phosphorus Average Per Year')
       plt.xlabel('Year')
       plt.ylabel('Average')
       plt.grid(True)
       plt.show()
```



```
[141]: ox_site_avg = ox_df.
        ↪groupby(['MonitoringLocationIdentifier','LocationLatitudeMeasure',␣
        ↪'LocationLongitudeMeasure'])['ResultMeasureValue'].mean().reset_index()
```

```
[143]: latitude = list(ox_site_avg['LocationLatitudeMeasure'])
       longitude = list(ox_site_avg['LocationLongitudeMeasure'])
       measure = list(ox_site_avg['ResultMeasureValue'])

       plot_data = {'latitude': latitude,
                    'longitude': longitude,
                    'measure': measure}

       plot_df = pd.DataFrame(plot_data)

       gdf = gpd.GeoDataFrame(plot_df, geometry = gpd.points_from_xy(plot_df.
        ↪longitude, plot_df.latitude))
       fig, ax = plt.subplots(figsize = (8,6))
       kansas.plot(ax=ax, color = 'lightblue', edgecolor = 'black')
       gdf.plot(ax = ax, color = 'red', markersize = plot_df['measure']*2)
       plt.title('Kansas Water Dissolved Oxygen Level by Site')
       plt.xlabel('Longitude')
       plt.ylabel('Latitude')
       plt.show()
```

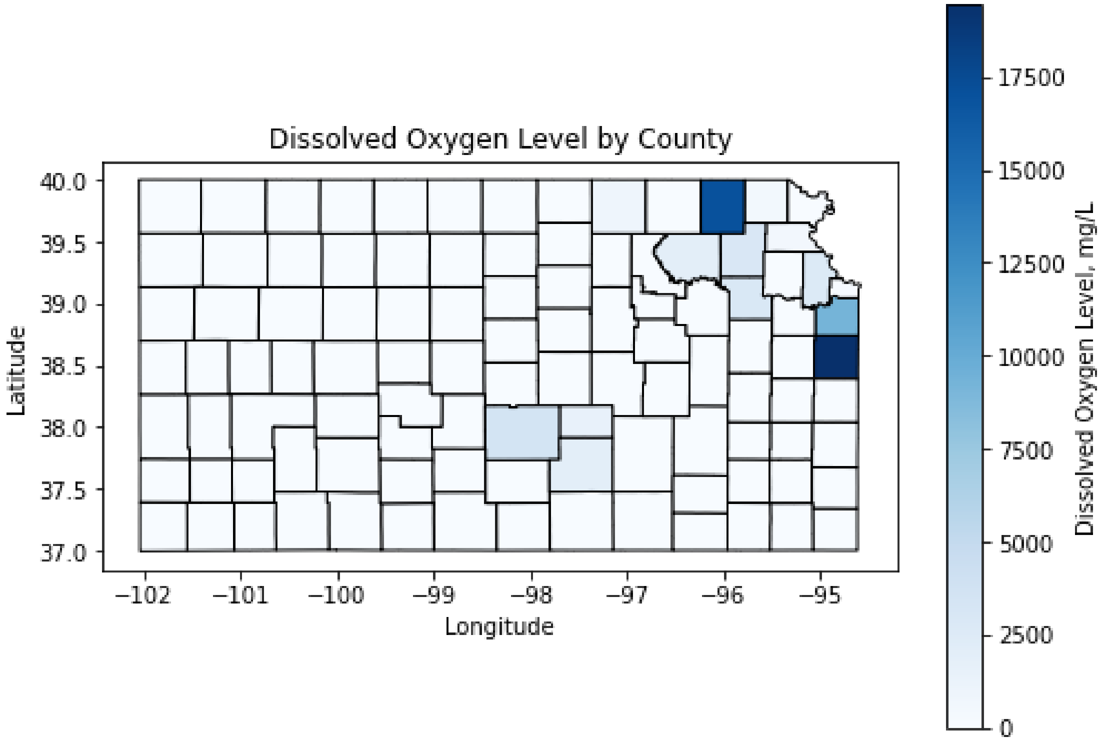


[144]:
```
ox_county_avg = ox_df.groupby('COUNTYFP')['ResultMeasureValue'].mean().
 ↪reset_index()
```

[145]:
```
kansas_ox = pd.merge(kansas2, ox_county_avg, on = "COUNTYFP", how = 'left')
```

[146]:
```
kansas_ox['ResultMeasureValue'] = kansas_ox['ResultMeasureValue'].fillna(0)
```

[147]:
```
fig, ax = plt.subplots(figsize = (8,6))
kansas_escher.plot(column = 'ResultMeasureValue', ax= ax, cmap = 'Blues',␣
 ↪edgecolor = 'black', legend = True, legend_kwds={'label':'Dissolved Oxygen␣
 ↪Level, mg/L'})
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Dissolved Oxygen Level by County')
plt.show()
```



1.0.11 Working with the Average Gross Income Level Per County

```
[155]: county_income_avg = water.groupby('COUNTYFP')['AvgGrossIncome_County'].mean().
        ↪reset_index()
```

```
[156]: county_income_avg.info()
```

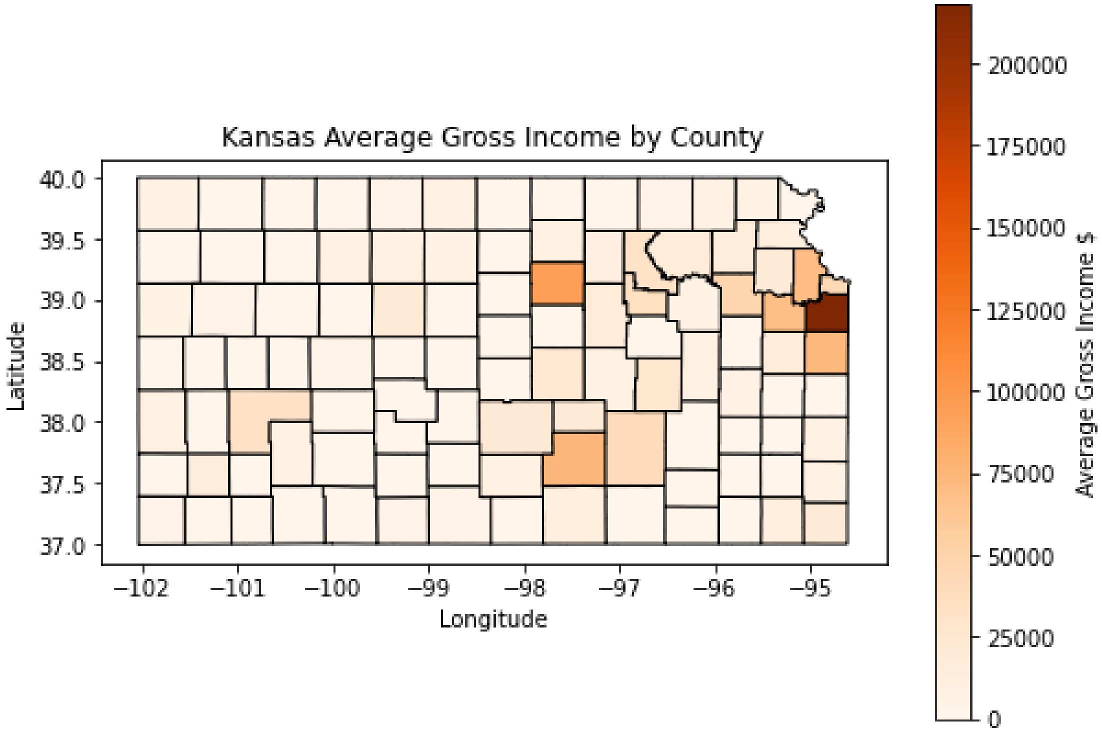
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61 entries, 0 to 60
Data columns (total 2 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   COUNTYFP               61 non-null     int64
 1   AvgGrossIncome_County  61 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 1.1 KB
```

```
[157]: kansas_county_income = pd.merge(kansas2, county_income_avg, on = "COUNTYFP",␣
        ↪how = 'left')
```

```
[160]: kansas_county_income['AvgGrossIncome_County'] =␣
        ↪kansas_county_income['AvgGrossIncome_County'].fillna(0)
```

[163]:

```
fig, ax = plt.subplots(figsize = (8,6))
kansas_county_income.plot(column = 'AvgGrossIncome_County', ax= ax, cmap =␣
 ↪'Oranges', edgecolor = 'black', legend = True, legend_kwds={'label':'Average␣
 ↪Gross Income $'})
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Kansas Average Gross Income by County')
plt.show()
```



1.0.12 Combining Data to form a Correlation Matrix

[ ]:

```
```

[165]:

```
nitrogen_county_avg = nitrogen_county_avg.rename(columns =␣
 ↪{'ResultMeasureValue':'AverageNitrogen'})
```

[169]:

```
nitrogen_county_avg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
```

```
 0   COUNTYFP         21 non-null     int64
 1   AverageNitrogen  21 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 464.0 bytes
```

[166]:

```
phospho_county_avg = phospho_county_avg.rename(columns = {'ResultMeasureValue':
 ↪'AveragePhosphorus'})
```

[167]:

```
escher_county_avg = escher_county_avg.rename(columns = {'ResultMeasureValue':
 ↪'AverageEscherColi'})
```

[168]:

```
ox_county_avg = ox_county_avg.rename(columns = {'ResultMeasureValue':
 ↪'AverageDissolvedOxygen'})
```

[174]:

```
characteristic_df = pd.merge(nitrogen_county_avg, phospho_county_avg, on =␣
 ↪'COUNTYFP', how = 'outer').merge(escher_county_avg, on = 'COUNTYFP', how =␣
 ↪'outer').merge(ox_county_avg, on = 'COUNTYFP', how = 'outer').
 ↪merge(county_income_avg, on = 'COUNTYFP', how = 'outer')
```

[175]:

```
characteristic_df.info()
```

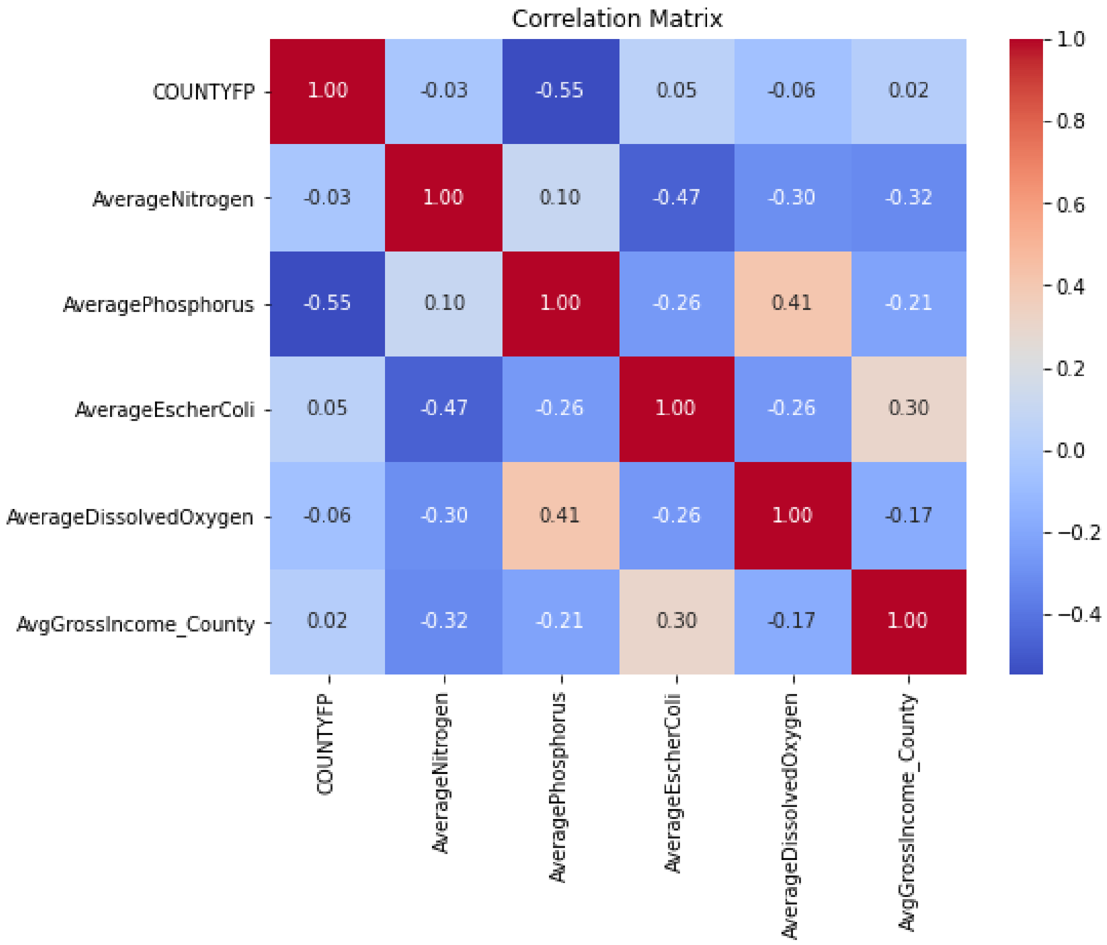
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 61 entries, 0 to 60
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   COUNTYFP                61 non-null     int64
 1   AverageNitrogen         21 non-null     float64
 2   AveragePhosphorus       18 non-null     float64
 3   AverageEscherColi       15 non-null     float64
 4   AverageDissolvedOxygen  38 non-null     float64
 5   AvgGrossIncome_County   61 non-null     float64
dtypes: float64(5), int64(1)
memory usage: 3.3 KB
```

[176]:

```
corr_matrix = characteristic_df.corr()
```

[180]:

```
import seaborn as sns

plt.figure(figsize = (8,6))
sns.heatmap(corr_matrix, annot = True, cmap = 'coolwarm', fmt = ".2f")
plt.title('Correlation Matrix')
plt.show()
```


Correlation Matrix
COUNTYFP 1.00 -0.03 -0.55 0.05 -0.06 0.02
AverageNitrogen -0.03 1.00 0.10 -0.47 -0.30 -0.32
AveragePhosphorus -0.55 0.10 1.00 -0.26 0.41 -0.21
AverageEscherColi 0.05 -0.47 -0.26 1.00 -0.26 0.30
AverageDissolvedOxygen -0.06 -0.30 0.41 -0.26 1.00 -0.17
AvgGrossIncome_County 0.02 -0.32 -0.21 0.30 -0.17 1.00
COUNTYFP
AverageNitrogen
AveragePhosphorus
AverageEscherColi
AverageDissolvedOxygen
AvgGrossIncome_County
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

```
[ ]:
```