SFkFkDeer

April 24, 2024

```
[ ]: import warnings
     warnings.filterwarnings("ignore", category=DeprecationWarning)
```

```
[ ]: import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
```

```
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
```

The following methods were performed to wrangle the data in Excel:

- Converting `Start Date` from `mm/dd/yy` to `yyyy-mm-dd`
- Converting `Time` from `hhmm` to `hh:mm`
- Changing one `Time` value `UNKNOWN` to `12:00` (middle of the day)
- Combining `Start Date` and `Time` into `DateTime` variable `yyyy-mm-dd hh:mm`
- Adding a `Season` variable based on `Start Date` variable
- Converting all blank values in `Value` to `1/2 * MDL RDQ Measure 2`

```
[ ]: # Load data
     df = pd.read_csv('SFkFkDeer.csv')
```

```
[ ]: # Convert DateTime to datetime data type (plot can interpret date/time values␣
      ↪in order)
     df['DateTime'] = pd.to_datetime(df['DateTime'])
```

```
[ ]: # Select relevant variables
     df = df.loc[:, df.columns.isin(['DateTime', 'Season', 'Characteristic',␣
      ↪'Value', 'Unit'])]
     df
```

```
[ ]:                DateTime  Season Characteristic    Value     Unit
     0   1998-12-16 09:15:00    Fall   Conductivity   60.000  umho/cm
     1   1999-03-24 11:45:00  Spring   Conductivity   65.000  umho/cm
     2   1999-06-09 09:50:00  Spring   Conductivity  127.000  umho/cm
     3   1999-09-28 09:50:00    Fall   Conductivity  110.000  umho/cm
     4   1999-12-01 11:20:00    Fall   Conductivity   97.000  umho/cm
     ..                  ...     ...            ...      ...      ...
     879 2022-06-29 10:38:00  Summer     Phosphorus    0.130     mg/l
     880 2022-09-27 10:55:00    Fall     Phosphorus    0.143     mg/l
     881 2022-12-13 10:50:00    Fall     Phosphorus    0.176     mg/l
     882 2023-03-09 10:10:00  Winter     Phosphorus    0.147     mg/l
     883 2023-10-24 10:56:00    Fall     Phosphorus    0.221     mg/l

     [884 rows x 5 columns]
```

```
[ ]: # Define boxplot season order and color palette
     order = ['Spring', 'Summer', 'Fall', 'Winter']
     palette = ['#7FFF00', '#FF6347', '#FFA500', '#4169E1']
```

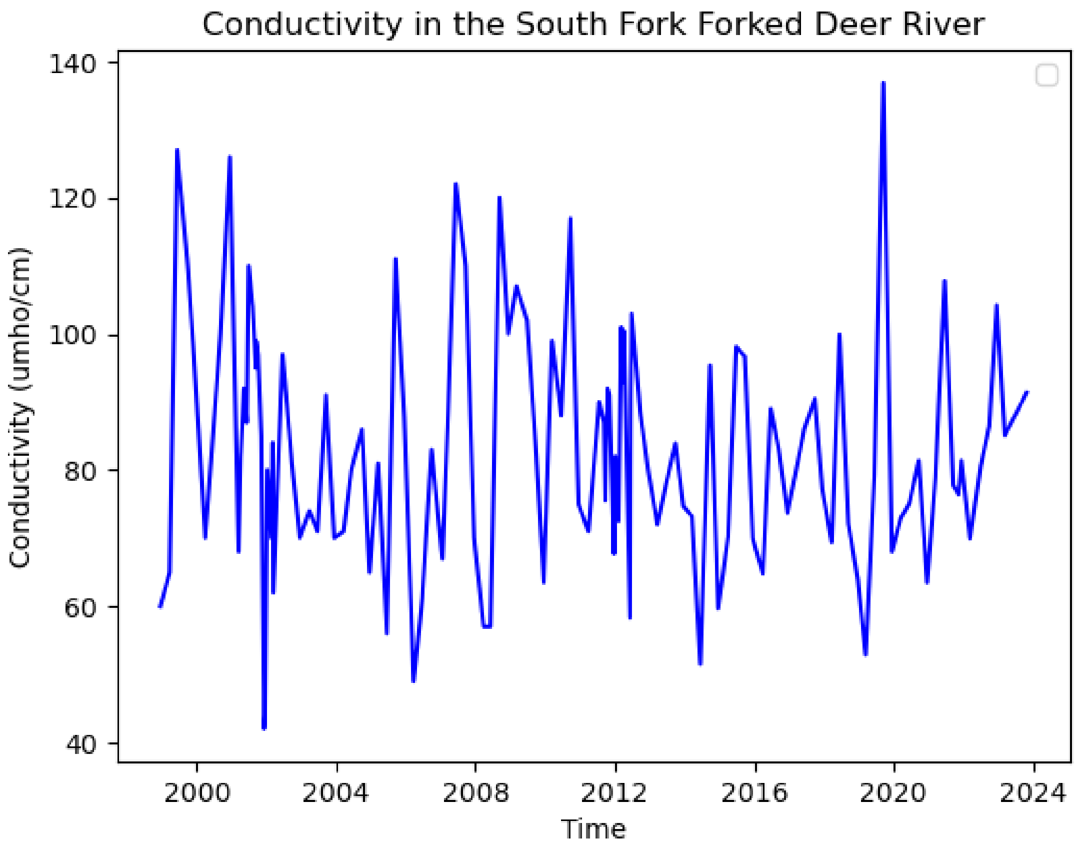
1 Conductivity Analysis

1.1 Temporal

```
[ ]: df_conductivity = df[df['Characteristic'] == 'Conductivity']
```

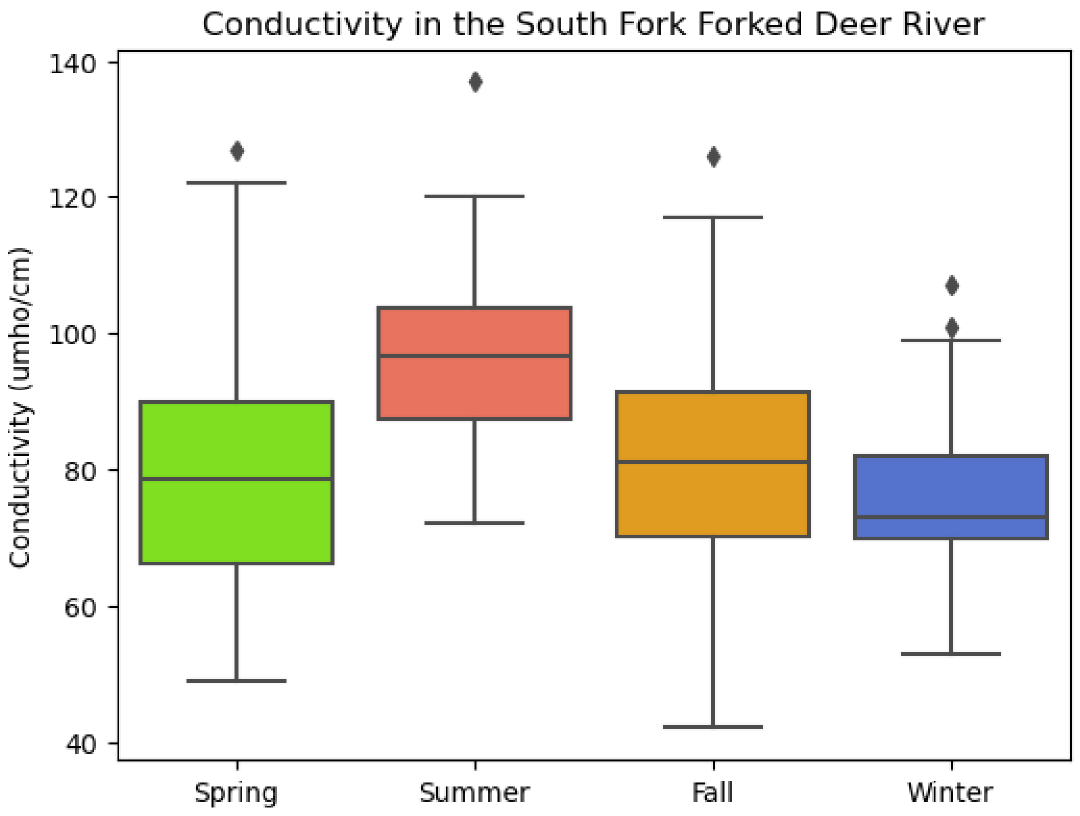
```
[ ]: sns.lineplot(x='DateTime', y='Value', data=df_conductivity, color='blue')
     plt.title('Conductivity in the South Fork Forked Deer River')
     plt.xlabel('Time')
     plt.ylabel('Conductivity (umho/cm)')
     plt.legend()
     plt.show()
```

```
/Users/brookestevens/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Users/brookestevens/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



1.2 Seasonal

```
[ ]: warnings.filterwarnings('ignore')
     sns.boxplot(x='Season', y='Value', data=df_conductivity, order=order,␣
      ↪palette=palette)
     plt.title('Conductivity in the South Fork Forked Deer River')
     plt.ylabel('Conductivity (umho/cm)')
     plt.xlabel('')
     plt.show()
```



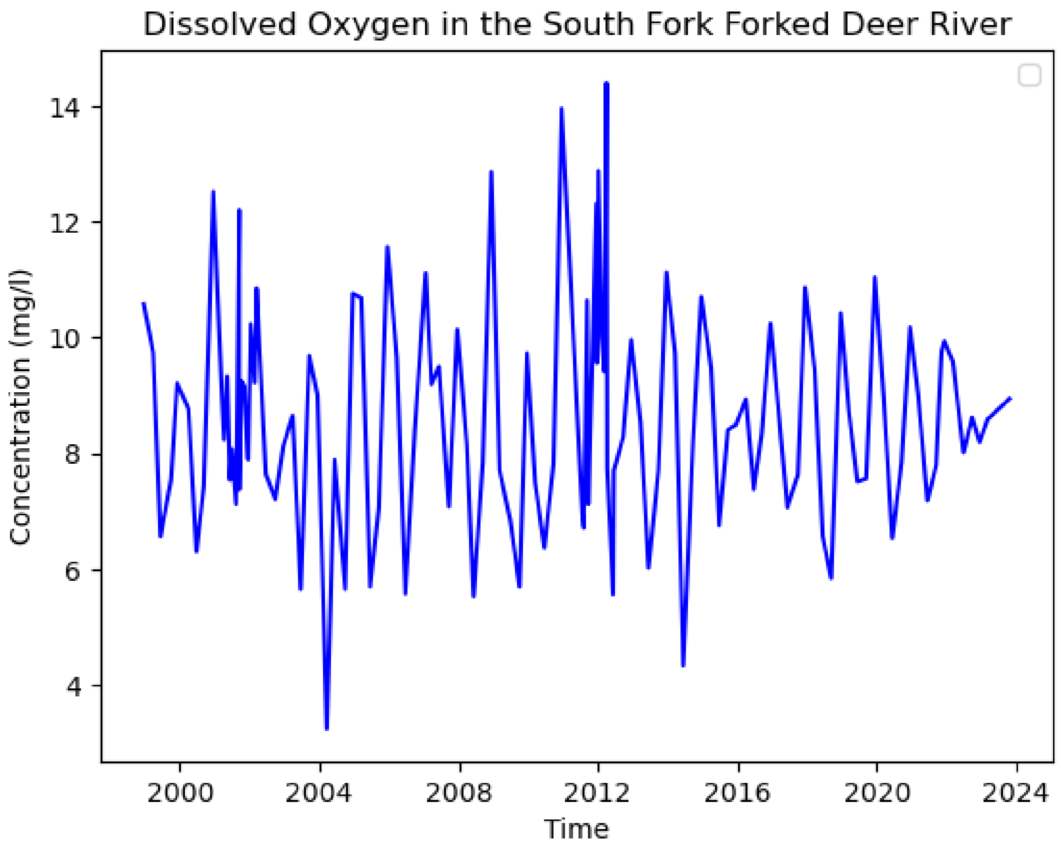
2 Dissolved Oxygen Analysis

2.1 Temporal

```
[ ]: df_do = df[df['Characteristic'] == 'Dissolved oxygen (DO)']
```

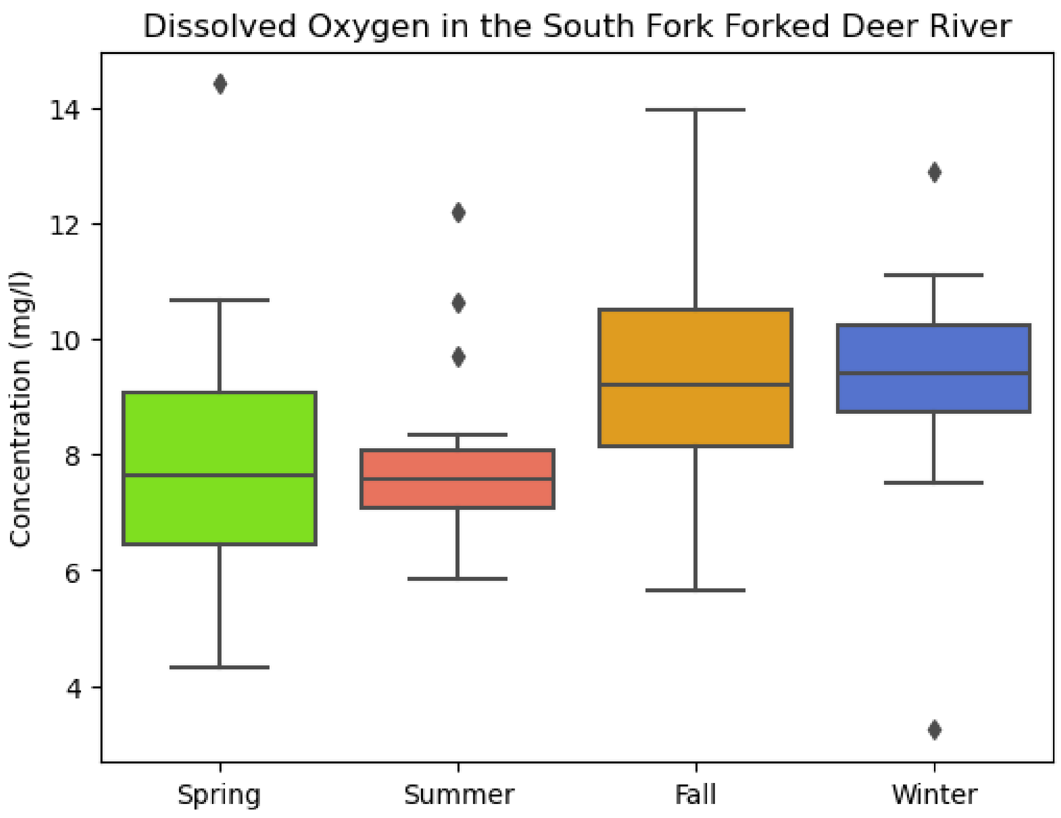
```
[ ]: sns.lineplot(x='DateTime', y='Value', data=df_do, color='blue')
     plt.title('Dissolved Oxygen in the South Fork Forked Deer River')
     plt.xlabel('Time')
     plt.ylabel('Concentration (mg/l)')
     plt.legend()
     plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



2.2 Seasonal

```
[ ]: warnings.filterwarnings('ignore')
     sns.boxplot(x='Season', y='Value', data=df_do, order=order, palette=palette)
     plt.title('Dissolved Oxygen in the South Fork Forked Deer River')
     plt.ylabel('Concentration (mg/l)')
     plt.xlabel('')
     plt.show()
```

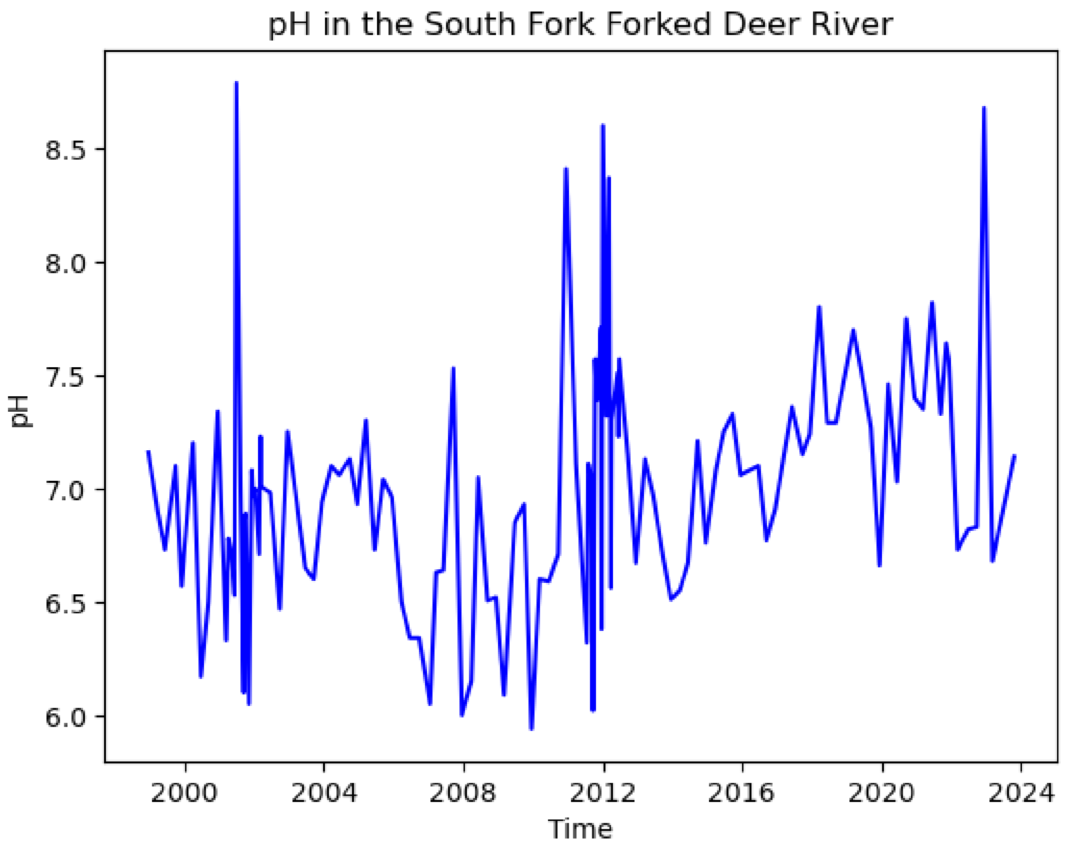


3 pH Analysis

3.1 Temporal

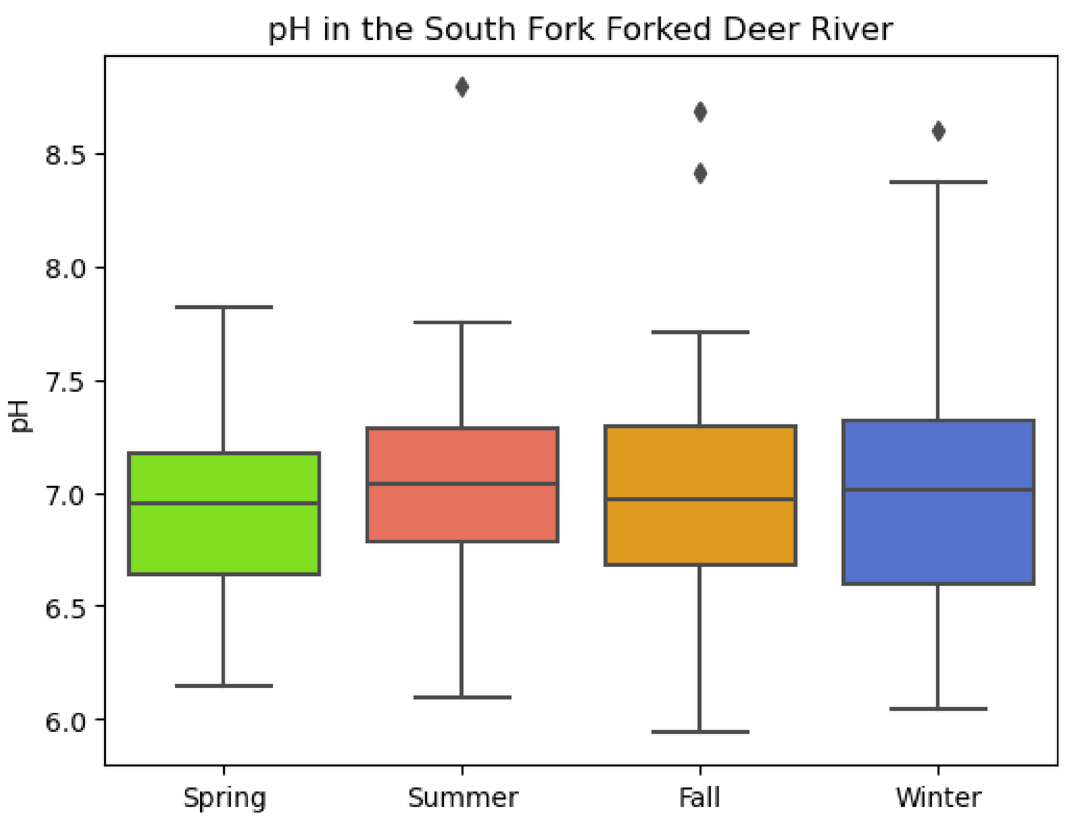
```
[ ]: df_ph = df[df['Characteristic'] == 'pH']
```

```
[ ]: sns.lineplot(x='DateTime', y='Value', data=df_ph, color='blue')
     plt.title('pH in the South Fork Forked Deer River')
     plt.xlabel('Time')
     plt.ylabel('pH')
     plt.show()
```



3.2 Seasonal

```
[ ]: warnings.filterwarnings('ignore')
     sns.boxplot(x='Season', y='Value', data=df_ph, order=order, palette=palette)
     plt.title('pH in the South Fork Forked Deer River')
     plt.ylabel('pH')
     plt.xlabel('')
     plt.show()
```



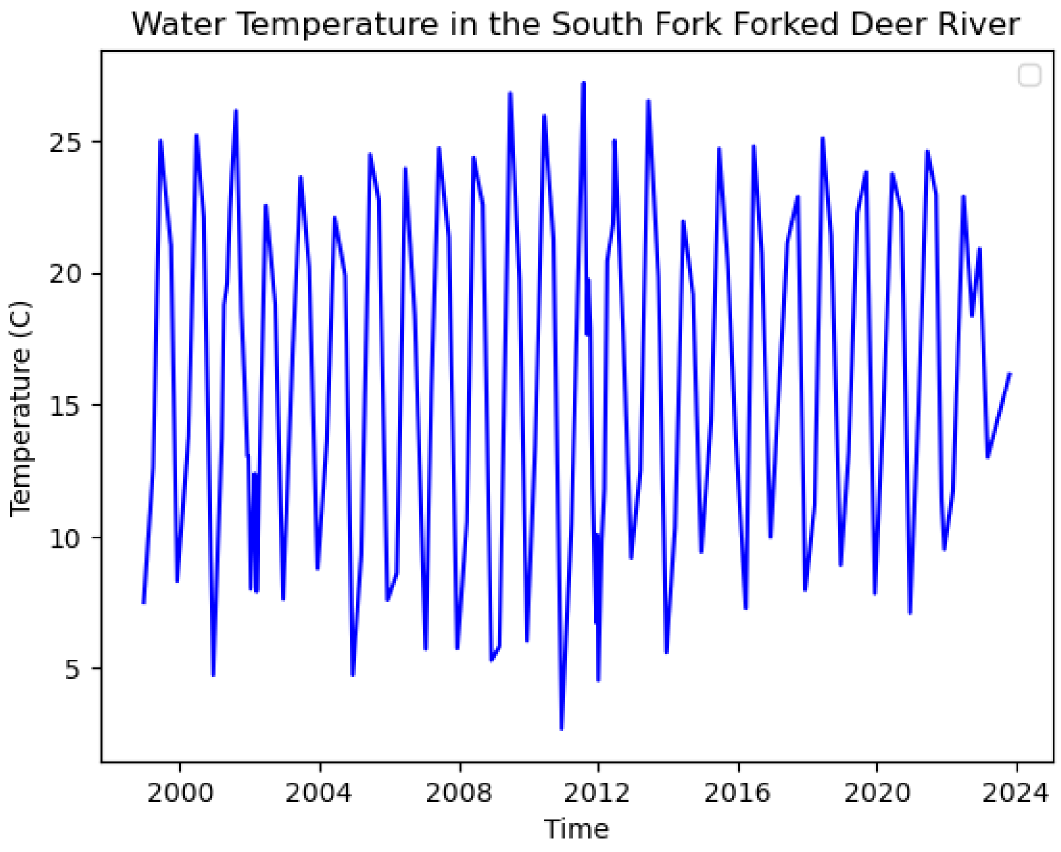
4 Water Temperature Analysis

4.1 Temporal

```
[ ]: df_watertemp = df[df['Characteristic'] == 'Temperature, water']
```

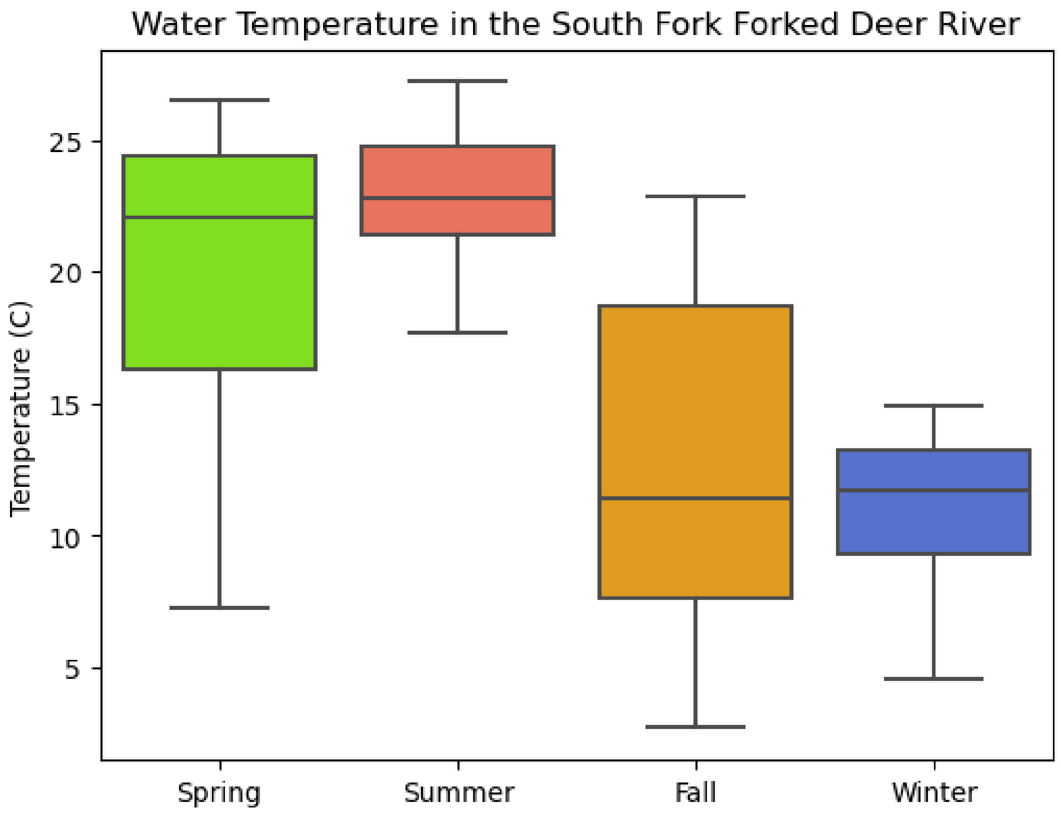
```
[ ]: sns.lineplot(x='DateTime', y='Value', data=df_watertemp, color='blue')
     plt.title('Water Temperature in the South Fork Forked Deer River')
     plt.xlabel('Time')
     plt.ylabel('Temperature (C)')
     plt.legend()
     plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



4.2 Seasonal

```
[ ]: warnings.filterwarnings('ignore')
     sns.boxplot(x='Season', y='Value', data=df_watertemp, order=order,␣
       ↪palette=palette)
     plt.title('Water Temperature in the South Fork Forked Deer River')
     plt.ylabel('Temperature (C)')
     plt.xlabel('')
     plt.show()
```



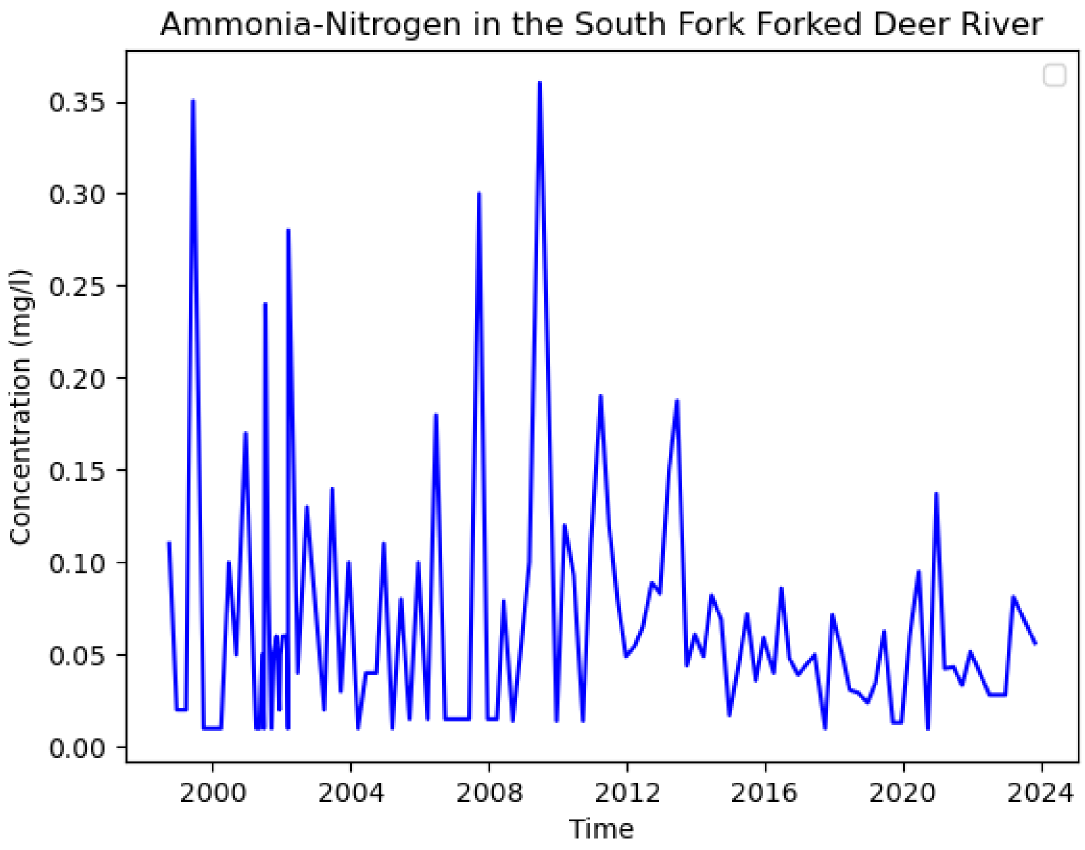
5 Ammonia-Nitrogen Analysis

5.1 Temporal

```
[ ]: df_ammnit = df[df['Characteristic'] == 'Ammonia-nitrogen']
```

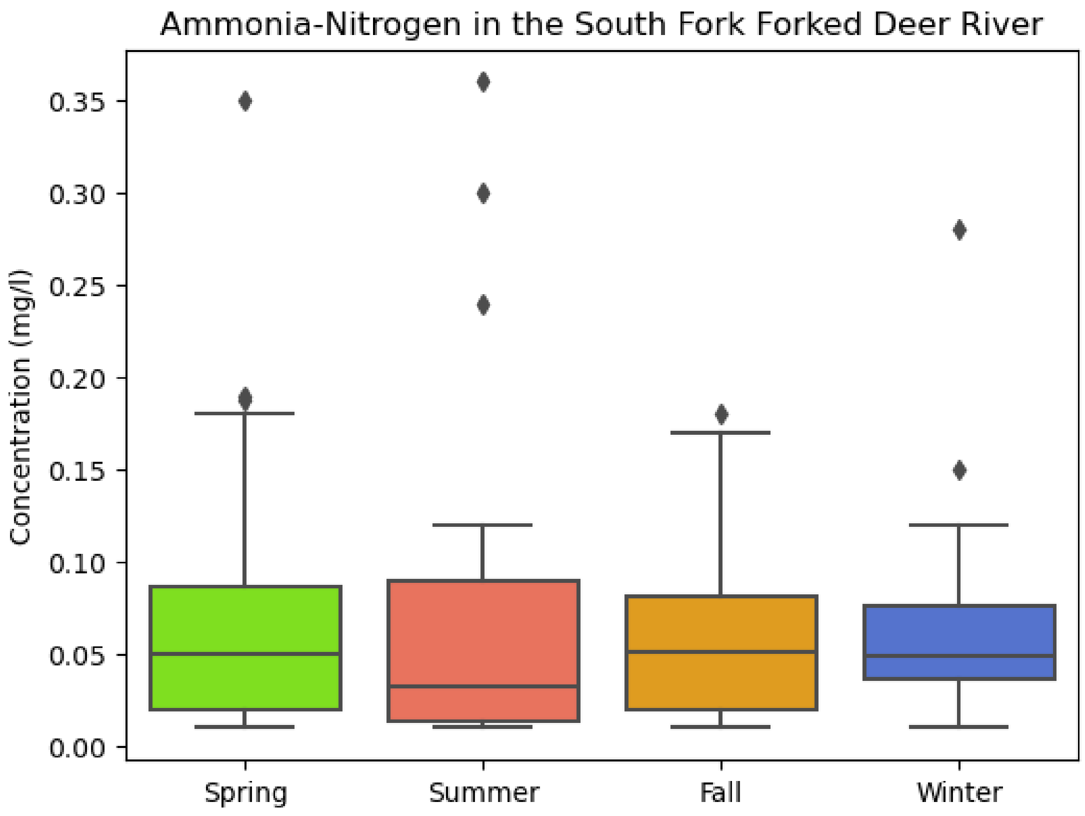
```
[ ]: sns.lineplot(x='DateTime', y='Value', data=df_ammnit, color='blue')
     plt.title('Ammonia-Nitrogen in the South Fork Forked Deer River')
     plt.xlabel('Time')
     plt.ylabel('Concentration (mg/l)')
     plt.legend()
     plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



5.2 Seasonal

```
[ ]: warnings.filterwarnings('ignore')
     sns.boxplot(x='Season', y='Value', data=df_ammnit, order=order, palette=palette)
     plt.title('Ammonia-Nitrogen in the South Fork Forked Deer River')
     plt.ylabel('Concentration (mg/l)')
     plt.xlabel('')
     plt.show()
```



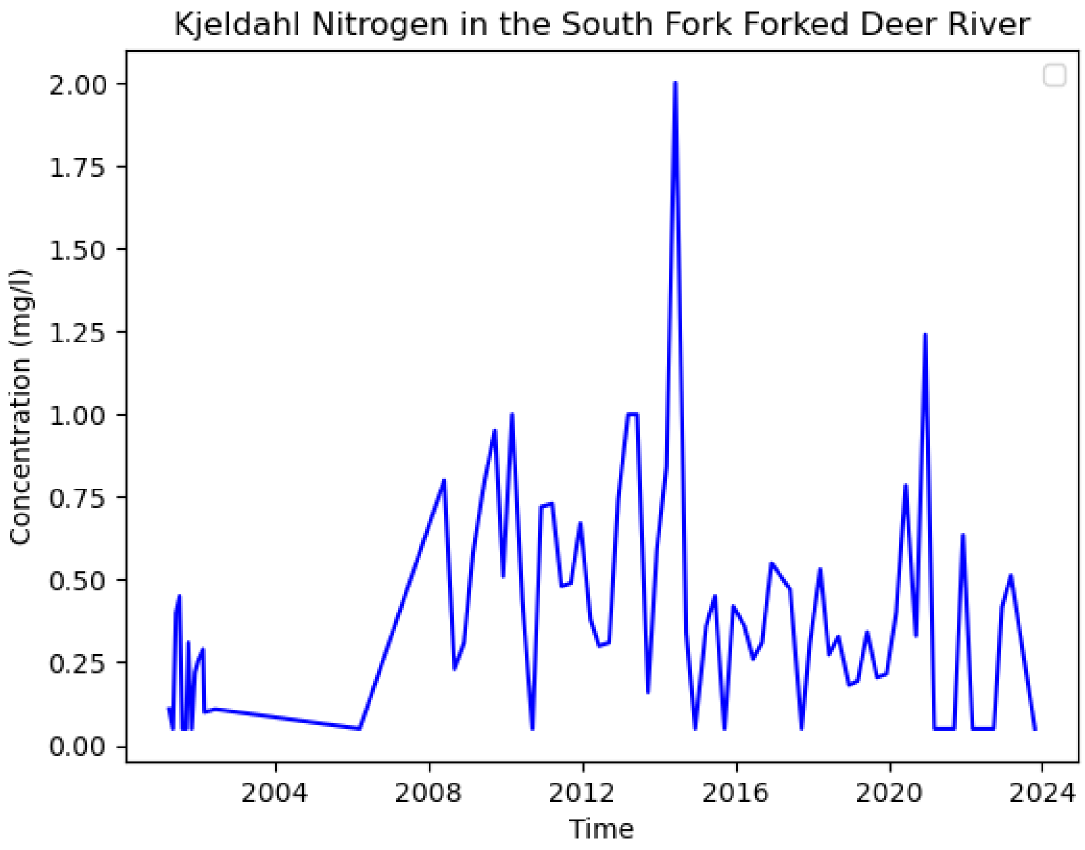
6 Kjeldahl Nitrogen Analysis

6.1 Temporal

[ ]:
```
df_kn = df[df['Characteristic'] == 'Kjeldahl nitrogen']
```

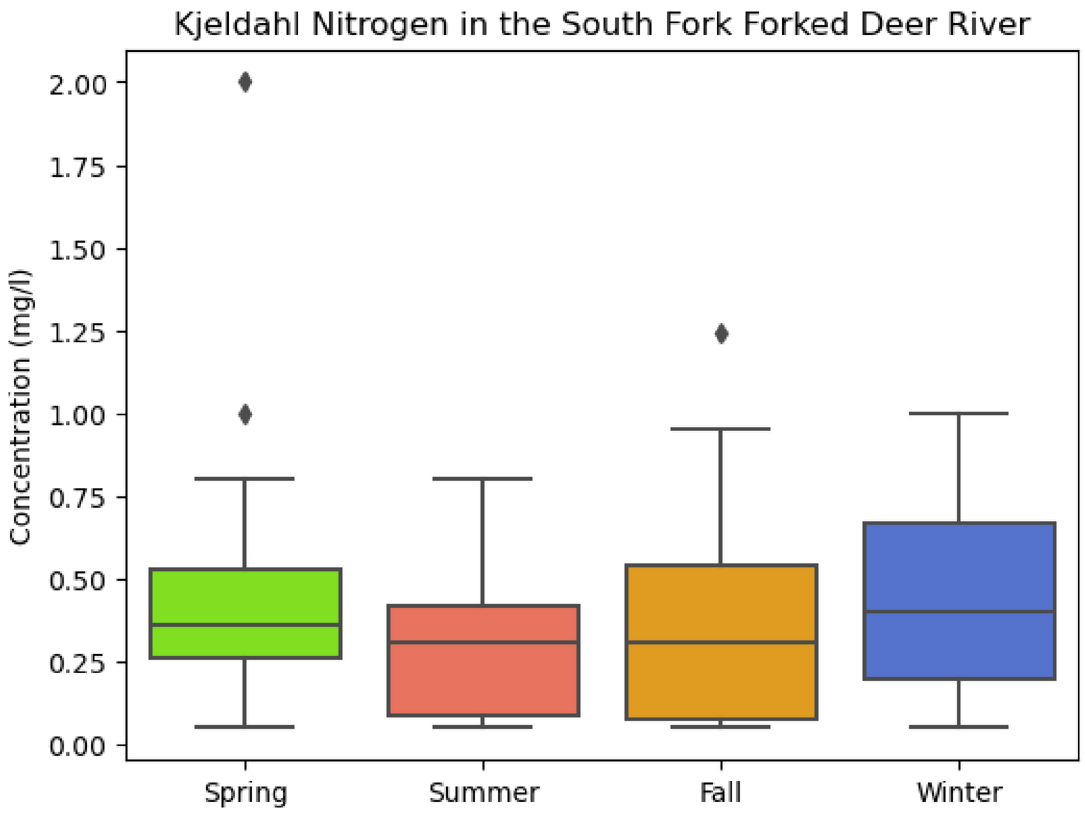
[ ]:
```
sns.lineplot(x='DateTime', y='Value', data=df_kn, color='blue')
plt.title('Kjeldahl Nitrogen in the South Fork Forked Deer River')
plt.xlabel('Time')
plt.ylabel('Concentration (mg/l)')
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



6.2 Seasonal

```
[ ]: warnings.filterwarnings('ignore')
     sns.boxplot(x='Season', y='Value', data=df_kn, order=order, palette=palette)
     plt.title('Kjeldahl Nitrogen in the South Fork Forked Deer River')
     plt.ylabel('Concentration (mg/l)')
     plt.xlabel('')
     plt.show()
```



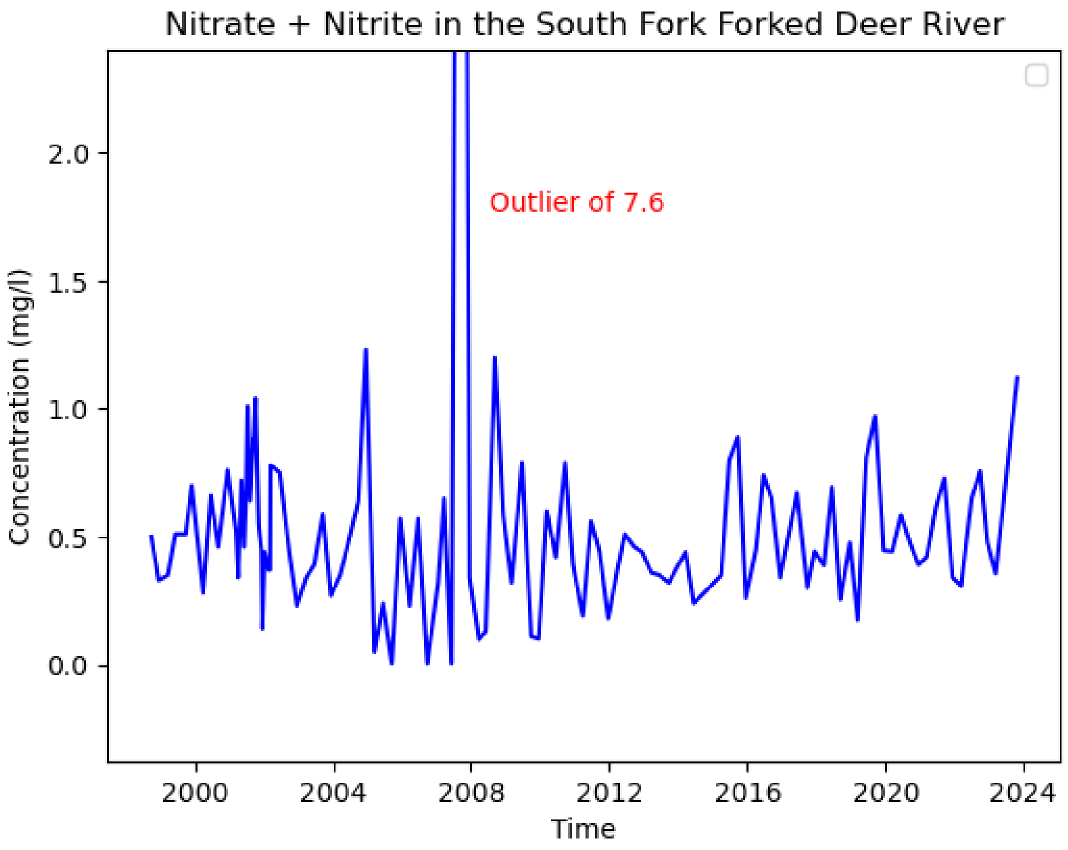
7 Nitrate + Nitrite Analysis

7.1 Temporal

```
[ ]: df_nn = df[df['Characteristic'] == 'Nitrate + Nitrite']
```

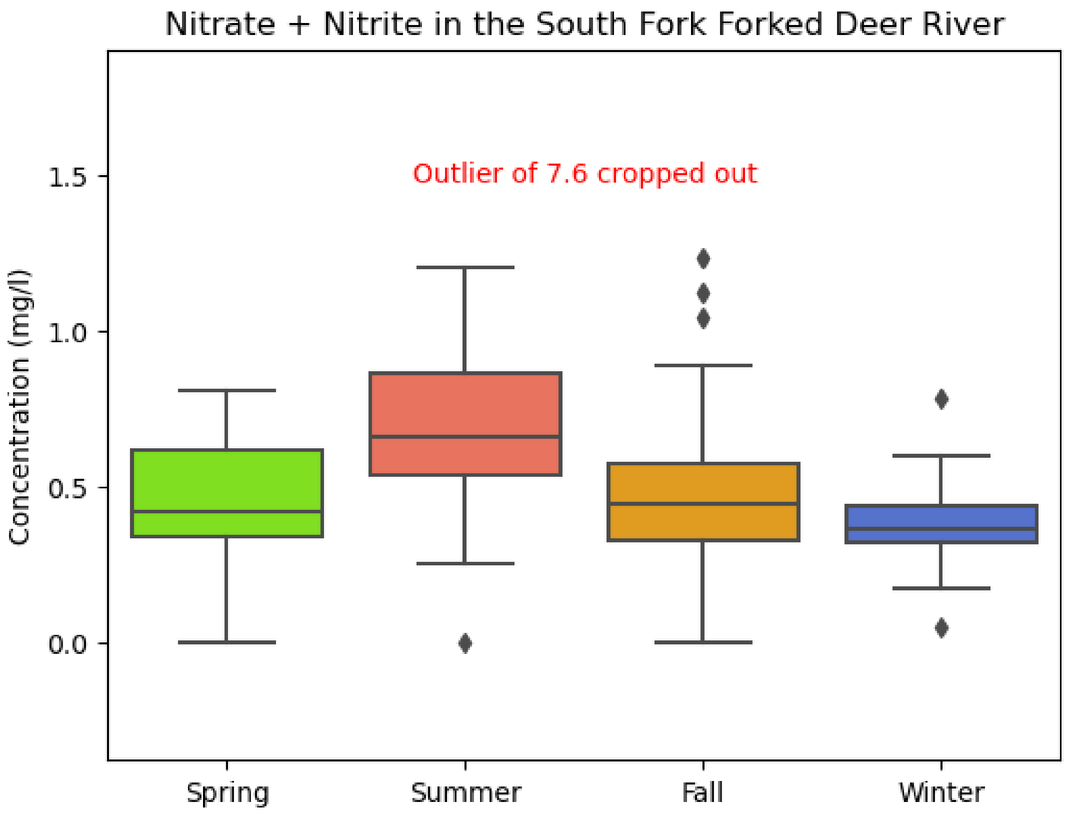
```
[ ]: sns.lineplot(x='DateTime', y='Value', data=df_nn, color='blue')
     plt.title('Nitrate + Nitrite in the South Fork Forked Deer River')
     plt.xlabel('Time')
     plt.ylabel('Concentration (mg/l)')
     plt.ylim(None, 2.4)
     plt.text(15000, 1.8, 'Outlier of 7.6', fontsize=10, color='red', ha='center',␣
      ↪va='center')
     plt.legend()
     plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



7.2 Seasonal

[ ]:
```
warnings.filterwarnings('ignore')
sns.boxplot(x='Season', y='Value', data=df_nn, order=order, palette=palette)
plt.title('Nitrate + Nitrite in the South Fork Forked Deer River')
plt.ylim(None, 1.9)
plt.text(1.5, 1.5, 'Outlier of 7.6 cropped out', fontsize=10, color='red',␣
  ↪ha='center', va='center')
plt.ylabel('Concentration (mg/l)')
plt.xlabel('')
plt.show()
```



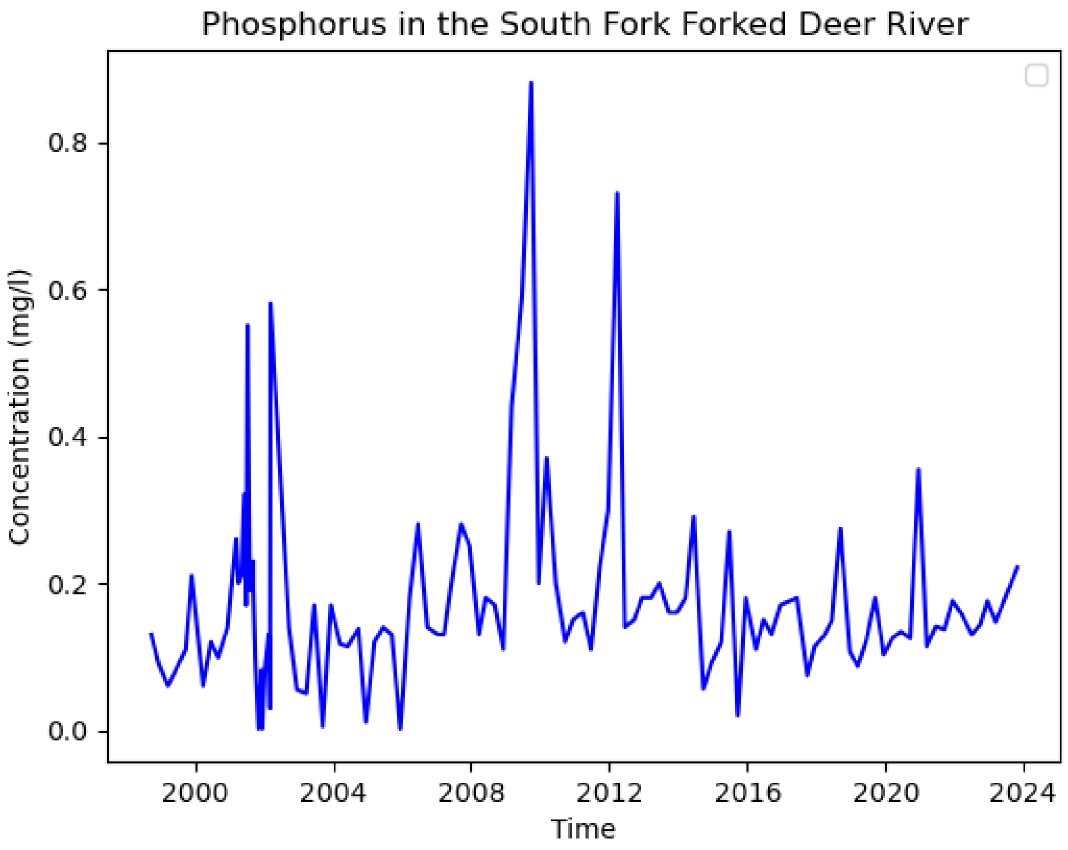
8 Phosphorus Analysis

8.1 Temporal

```
[ ]: df_phosphorus = df[df['Characteristic'] == 'Phosphorus']
```

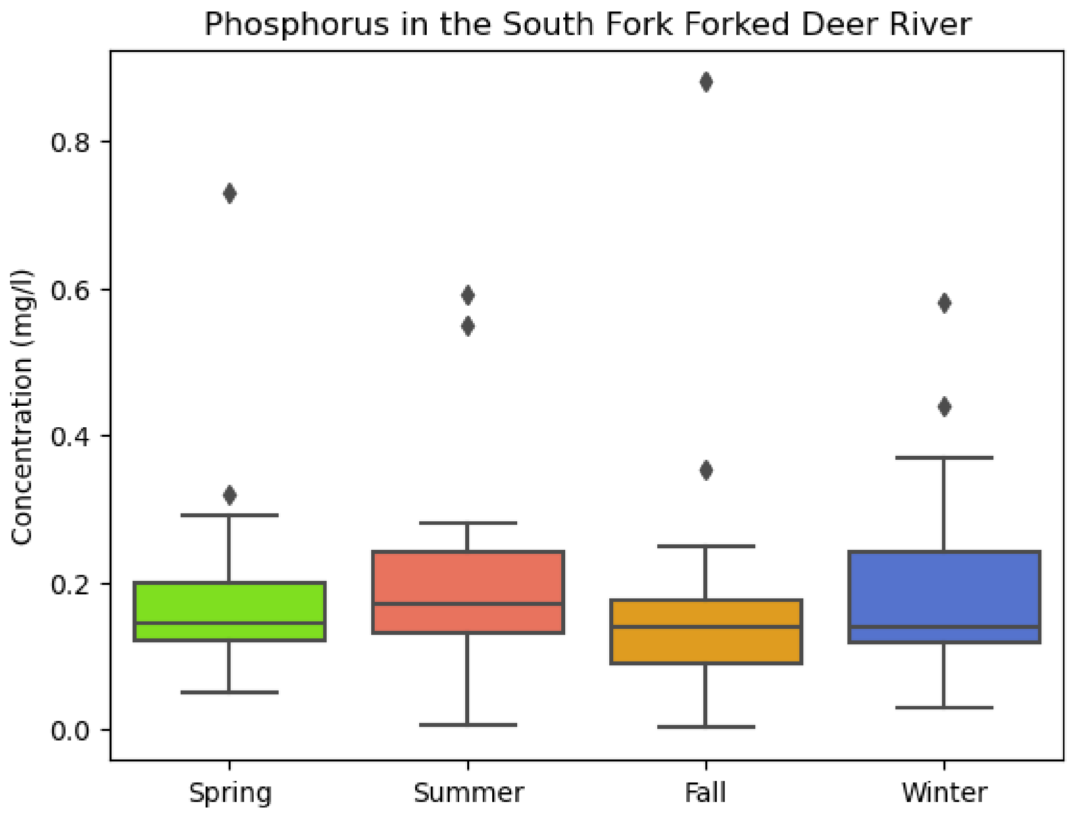
```
[ ]: sns.lineplot(x='DateTime', y='Value', data=df_phosphorus, color='blue')
     plt.title('Phosphorus in the South Fork Forked Deer River')
     plt.xlabel('Time')
     plt.ylabel('Concentration (mg/l)')
     plt.legend()
     plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



8.2 Seasonal

```
[ ]: warnings.filterwarnings('ignore')
     sns.boxplot(x='Season', y='Value', data=df_phosphorus, order=order,␣
       ↪palette=palette)
     plt.title('Phosphorus in the South Fork Forked Deer River')
     plt.ylabel('Concentration (mg/l)')
     plt.xlabel('')
     plt.show()
```


Phosphorus in the South Fork Forked Deer River
Concentration (mg/l)
0.0
0.2
0.4
0.6
0.8
Spring
Summer
Fall
Winter