Hands-on Activity 9.2 Customized Visualizations using Seaborn

Instructions:

Create a Python notebook to answer all shown procedures, exercises and analysis in this section.

Resources:

Download the following datasets: fb_stock_prices_2018.csv, Download earthquakes-1.csv Procedures:

- 9.4 Introduction to Seaborn
- 9.5 Formatting Plots
- 9.6 Customizing Visualizations

Data Analysis:

- 9.4 Introduction to Seaborn:

I learned that Seaborn is built on top of Matplotlib and it makes data visualization a lot easier and cleaner. It’s designed to work well with DataFrames, and the default styles already look professional. Using functions like sns.boxplot() and sns.histplot(), I can create good-looking plots with just a few lines.

- 9.5 Formatting Plots:

This part was all about improving the appearance of my graphs. I learned how to add proper titles, labels, and adjust things like rotation on tick labels to make the plot more readable. Even small formatting changes made a big difference in making the plots easier to understand.

- 9.6 Customizing Visualizations:

Here I got to explore things like choosing color palettes, adjusting transparency, and turning on grid lines. It helped me match the visual style of the plot to the type of data I was showing. Customizing the visuals made it more engaging and clearer for comparisons

Supplementary Activity

Using the CSV files provided and what we have learned so far in this module complete the following exercises:

1. Using seaborn, create a heatmap to visualize the correlation coefficients between earthquake magnitude and whether there was a tsunami with the magType of mb.
2. Create a box plot of Facebook volume traded and closing prices, and draw reference lines for the bounds of a Tukey fence with a multiplier of 1.5. The bounds will be at Q1 - 1.5 * IQR and Q3 + 1.5 * IQR. Be sure to use the quantile() method on the data to make this easier. (Pick whichever orientation you prefer for the plot, but make sure to use subplots.)
3. Fill in the area between the bounds in the plot from exercise #2.
4. Use axvspan() to shade a rectangle from '2018-07-25' to '2018-07-31', which marks the large decline in Facebook price on a line plot of the closing price.
5. Using the Facebook stock price data, annotate the following three events on a line plot of the closing price:
   - Disappointing user growth announced after close on July 25, 2018
   - Cambridge Analytica story breaks on March 19, 2018 (when it affected the market)
   - FTC launches investigation on March 20, 2018
6. Modify the reg_resid_plots() function to use a matplotlib colormap instead of cycling between two colors. Remember, for this use case, we should pick a qualitative colormap or make our own.

Summary/Conclusion:

Provide a summary of your learnings and the conclusion for this activity.

I learned how to use different types of plots to better understand data, like line plots, histograms, KDEs, and box plots. I also practiced using rolling and resample for time series data, and how to customize visuals with Matplotlib and Seaborn. It made it easier to see patterns and differences in the data. I might not remember all the details right away, but I have a better idea of how to start and what tools to use when analyzing data.

In [1]:

```
#Using the CSV files provided and what we have learned so far in this module comple

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
fb = pd.read_csv('data/fb_stock_prices_2018.csv', index_col='date', parse_dates=Tru
quakes = pd.read_csv('data/earthquakes.csv')
```

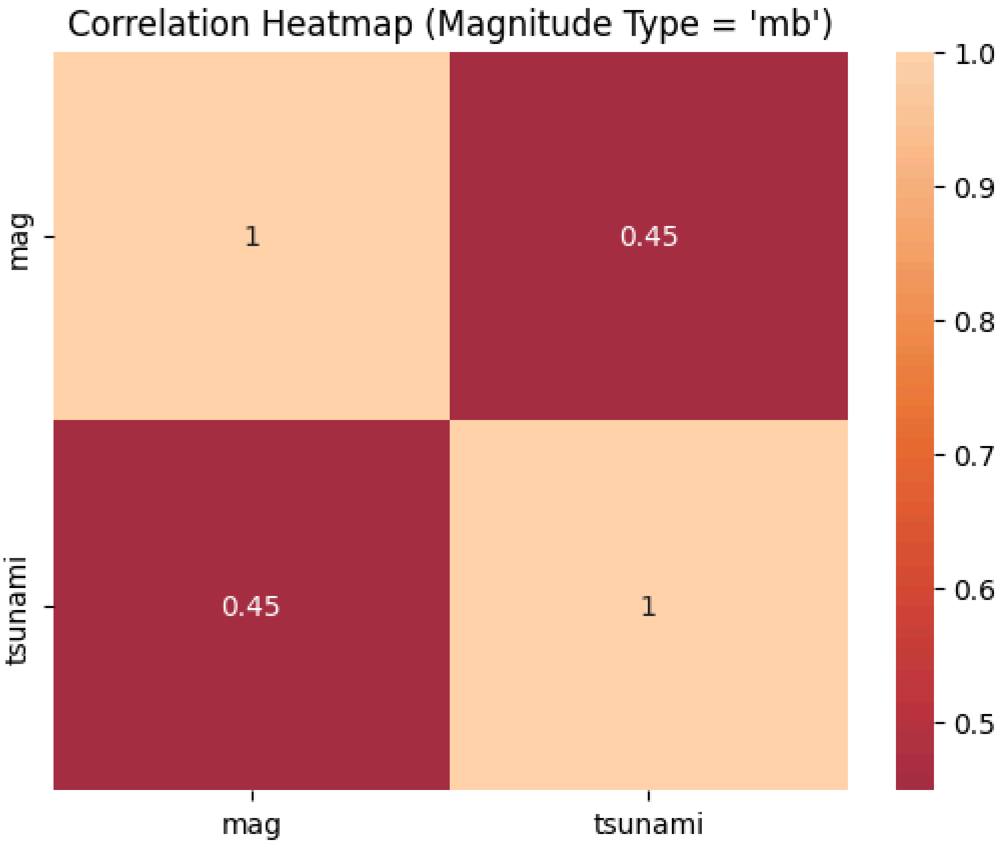
In [10]:

```
#Using seaborn, create a heatmap to visualize the correlation coefficients between

mb_filter = quakes.query(" magType == 'mb'") #filters out magtype to only choosing

correlations = mb_filter[['mag','tsunami']].corr() #computing the correlations of m
```

```
sns.heatmap(correlations, annot = True, center = 0)
plt.title("Correlation Heatmap (Magnitude Type = 'mb')")
```

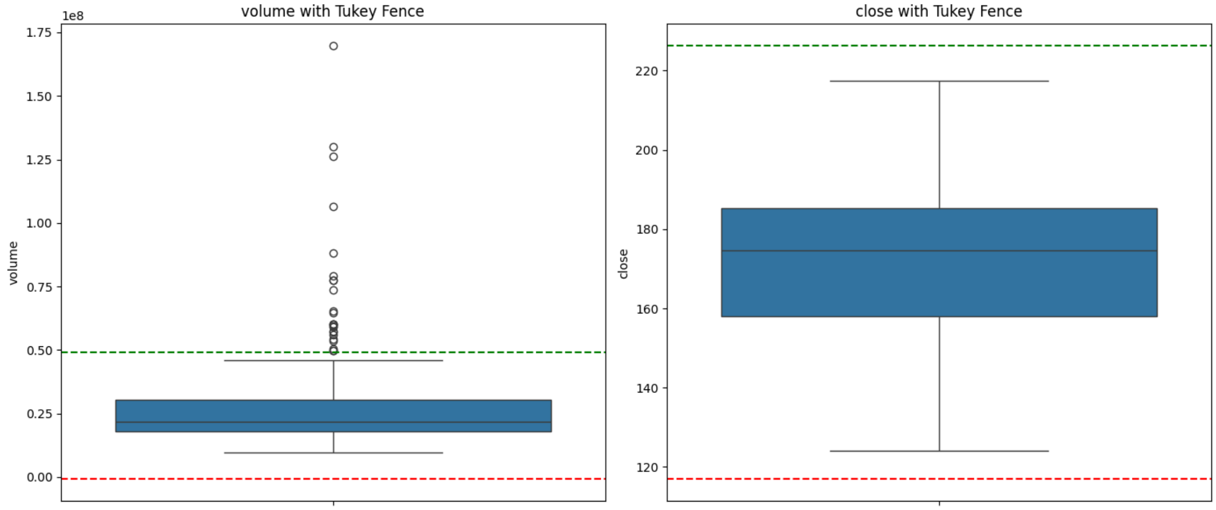
Out[10]:

```
Text(0.5, 1.0, "Correlation Heatmap (Magnitude Type = 'mb')")
```



In [13]:

```
#2. Create a box plot of Facebook volume traded and closing prices, and draw refere
#The bounds will be at Q1 - 1.5 * IQR and Q3 + 1.5 * IQR. Be sure to use the quanti
#(Pick whichever orientation you prefer for the plot, but make sure to use subplots
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

for i, col in enumerate(['volume', 'close']):
    q1 = fb[col].quantile(0.25)
    q3 = fb[col].quantile(0.75)
    iqr = q3 - q1
    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr

    sns.boxplot(y=fb[col], ax=axes[i])
    axes[i].axhline(lower, color='red', linestyle='--')
    axes[i].axhline(upper, color='green', linestyle='--')
    axes[i].set_title(f'{col} with Tukey Fence')

plt.tight_layout()
```

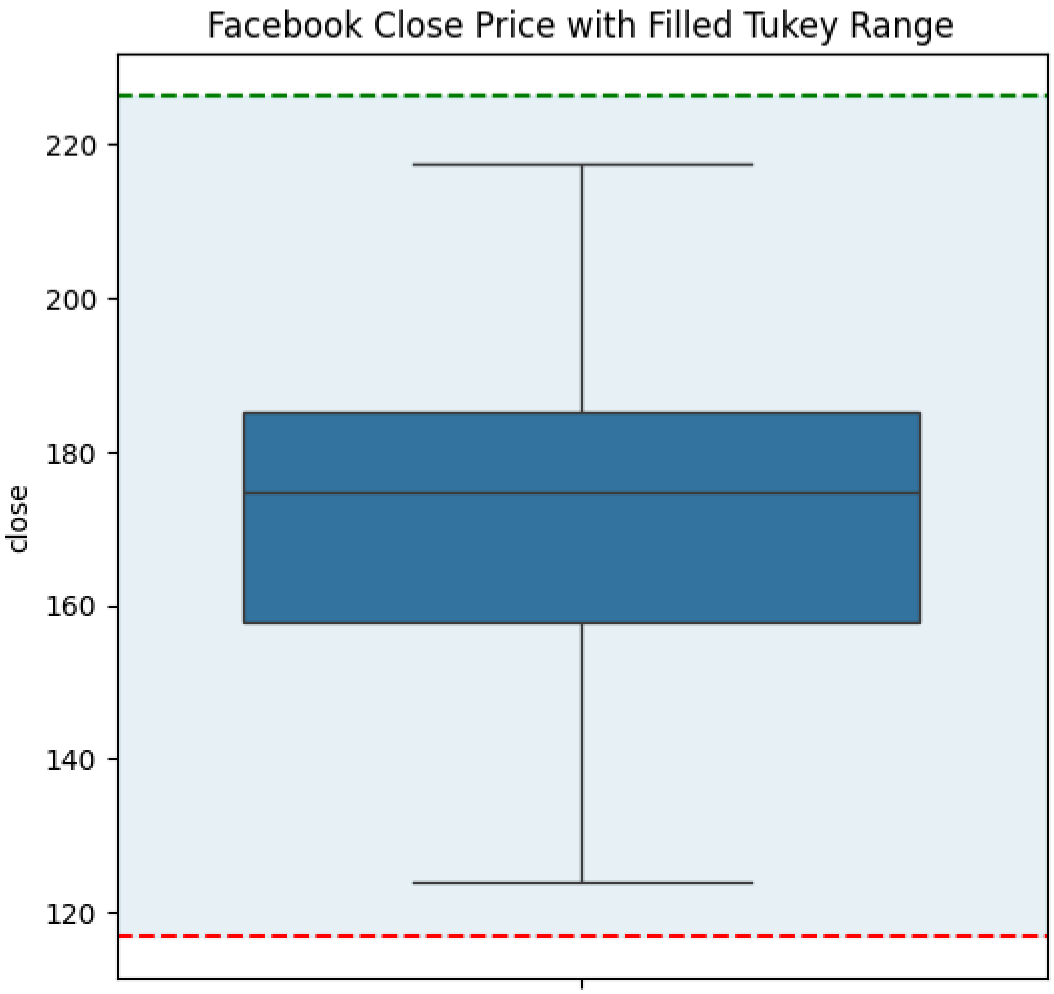


In [15]:

```
# 3. Fill in the area between the bounds in the plot from exercise #2.

fig, ax = plt.subplots(figsize=(6, 6))

q1 = fb[col].quantile(0.25)
q3 = fb[col].quantile(0.75)
iqr = q3 - q1
lower = q1 - 1.5 * iqr
upper = q3 + 1.5 * iqr

sns.boxplot(y=fb[col], ax=ax)
ax.axhline(lower, color='red', linestyle='--')
ax.axhline(upper, color='green', linestyle='--')
ax.fill_betweenx(y=[lower, upper], x1=0, x2=1, color='lightblue', alpha=0.3, transf

ax.set_title('Facebook Close Price with Filled Tukey Range')
```

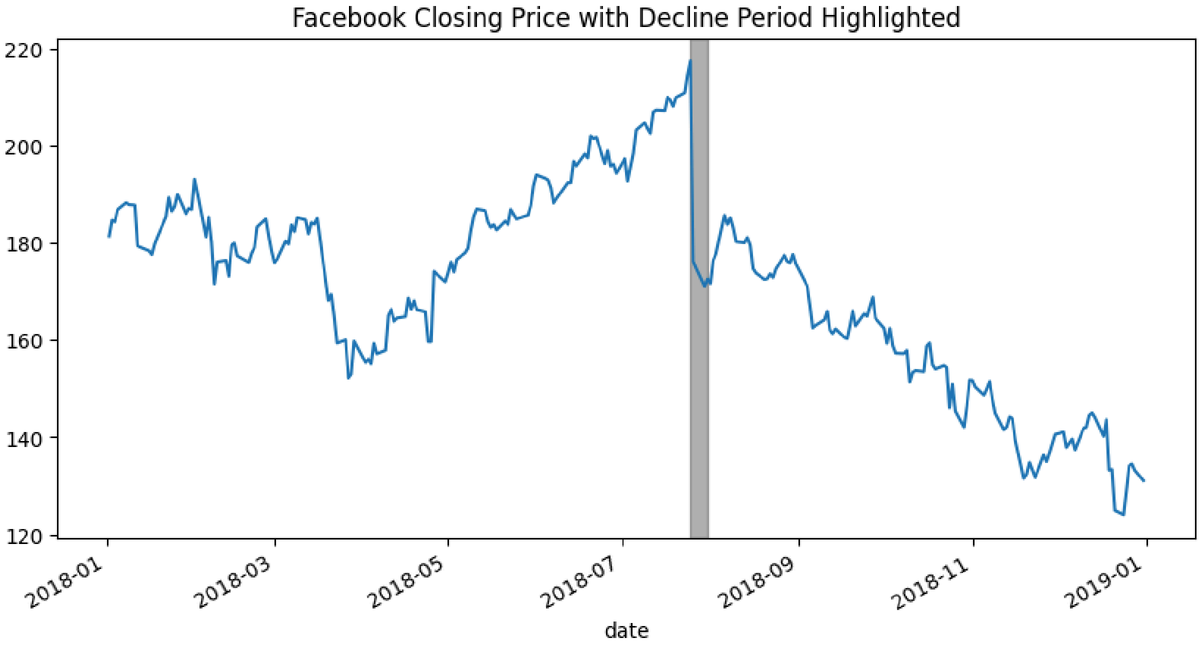
Out[15]:

```
Text(0.5, 1.0, 'Facebook Close Price with Filled Tukey Range')
```



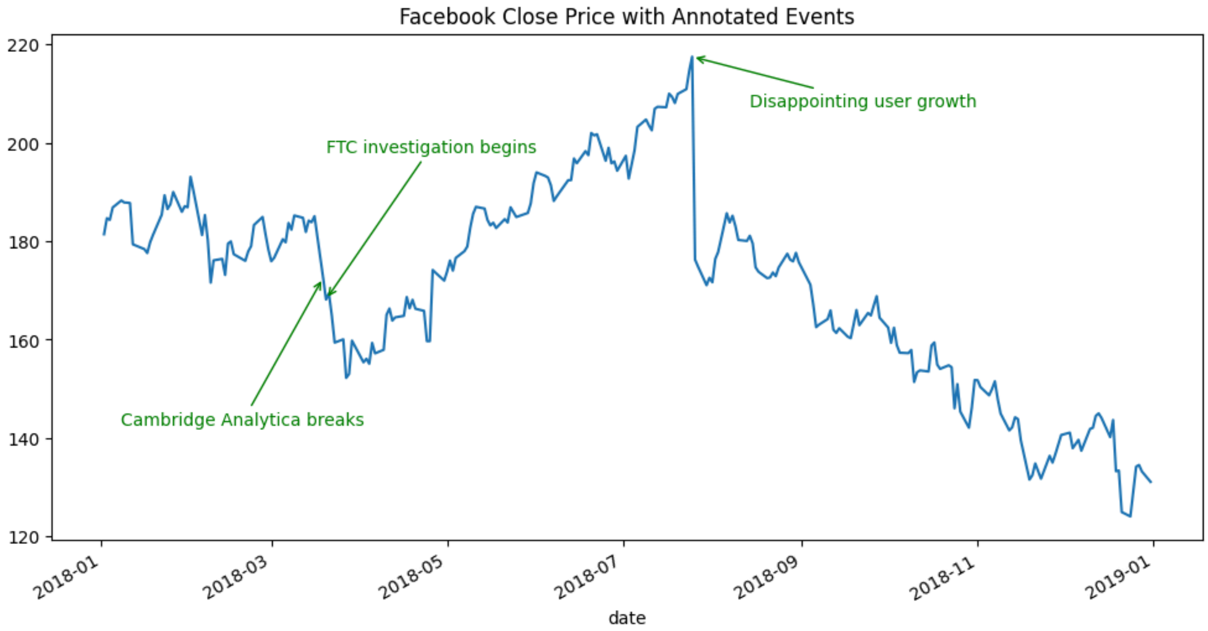
```
In [17]: # 4. Use axvspan() to shade a rectangle from '2018-07-25' to '2018-07-31', which ma

         fig, ax = plt.subplots(figsize=(10, 5))

         fb['close'].plot(ax=ax)
         ax.axvspan(pd.to_datetime('2018-07-25'), pd.to_datetime('2018-07-31'), color='black
         ax.set_title("Facebook Closing Price with Decline Period Highlighted")
```

```
Out[17]: Text(0.5, 1.0, 'Facebook Closing Price with Decline Period Highlighted')
```



In [23]:

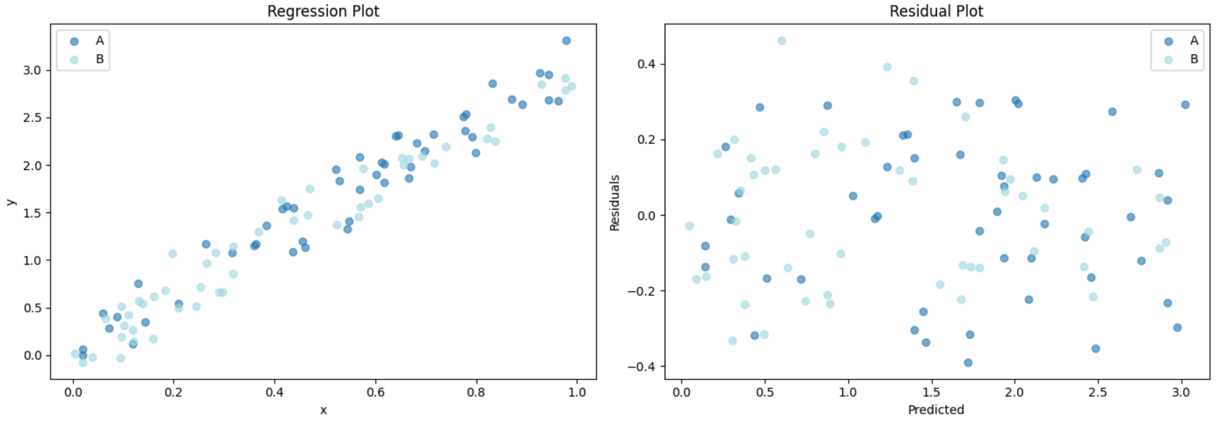
```
#Using the Facebook stock price data, annotate the following three events on a line
#   - Disappointing user growth announced after close on July 25, 2018
#   - Cambridge Analytica story breaks on March 19, 2018 (when it affected the mark
#   - FTC launches investigation on March 20, 2018

fig, ax = plt.subplots(figsize=(12, 6))

fb['close'].plot(ax=ax)
events = {
    '2018-07-25': 'Disappointing user growth',
    '2018-03-19': 'Cambridge Analytica breaks',
    '2018-03-20': 'FTC investigation begins'
}

# Define (x, y) offset for each annotation to get diagonal arrows
offsets = [
    (20, -10),  # Left-up
    (-70, -30),   # Right-up
    (0, 30)  # Left-down
]

for (date, label), (dx, dy) in zip(events.items(), offsets):
    x = pd.to_datetime(date)
    y = fb.loc[date, 'close']

    ax.annotate(label,
                xy=(x, y),
                xytext=(x + pd.Timedelta(days=dx), y + dy),
                arrowprops=dict(arrowstyle="->", color='green'),
                fontsize=10,
                color='green')

ax.set_title("Facebook Close Price with Annotated Events")
plt.show()
```



In [28]:

```
# 6. Modify the reg_resid_plots() function to use a matplotlib colormap instead of

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import to_rgba
import matplotlib.cm as cm

def reg_resid_plots(x, y, group=None):
    import statsmodels.api as sm  # Importing statsmodels for regression

    # Setting up two subplots  one for regression, one for residuals
    fig, axs = plt.subplots(1, 2, figsize=(14, 5))


    # Get unique group names and create a colormap with enough distinct colors
    unique_groups = list(pd.Series(group).unique())
    cmap = cm.get_cmap('tab20', len(unique_groups))
    for i, grp in enumerate(unique_groups):
        # Mask for the current group
        mask = (pd.Series(group) == grp)
        x_vals = x[mask]
        y_vals = y[mask]

        # Scatter plot for regression
        axs[0].scatter(x_vals, y_vals, color=cmap(i), label=str(grp), alpha=0.6)

        # Linear regression model using statsmodels
        X_sm = sm.add_constant(x_vals)  # Adds intercept
        model = sm.OLS(y_vals, X_sm).fit()
        pred_vals = model.predict(X_sm)  # Predictions

        # Residuals plot
        axs[1].scatter(pred_vals, model.resid, color=cmap(i), label=str(grp), alpha

    # Set titles and axis labels for both plots
    axs[0].set_title('Regression Plot')
    axs[0].set_xlabel('x')
    axs[0].set_ylabel('y')

    axs[1].set_title('Residual Plot')
    axs[1].set_xlabel('Predicted')
    axs[1].set_ylabel('Residuals')

    # Show legend for both plots
    for ax in axs:
        ax.legend()

    # Make layout cleaner
    plt.tight_layout()

# Random test data for demonstration
np.random.seed(0)
x = pd.Series(np.random.rand(100))
y = 3 * x + np.random.normal(0, 0.2, size=100)
group = ['A' if i < 50 else 'B' for i in range(100)]

# Run the function with sample data
reg_resid_plots(x, y, group)
```

```
<ipython-input-28-eb173abef410>:18: MatplotlibDeprecationWarning: The get_cmap funct
ion was deprecated in Matplotlib 3.7 and will be removed in 3.11. Use ``matplotlib.c
olormaps[name]`` or ``matplotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` ins
tead.
  cmap = cm.get_cmap('tab20', len(unique_groups))
```



In [ ]: