Homework4

October 24, 2024

0.1 Homework 4

0.1.1 Problem 1: ANOVA

For this problem, consider only differences between watershed 1 (WS1) and watershed 2 (WS2). These two watersheds are adjacent to each other in the HJ Andrews Experimental Forest. We want to test if there was a change in streamflow due to the forest within WS1 being completely clearcut (starting late 1962 and completed in 1966). Because the two watersheds are adjacent, we can expect that they experience the same storms leading to peak runoff (so we won't be considering any differences due to different precipitation amounts or timing). We want to know whether the four periods are statistically different from each other, and if so, which one or ones are statistically different from which other ones.

A. First, plot the timeseries of the streamflow measurements as a function of water year for both watershed 1 and watershed 2 on the same graph. Use vertical dashed lines (axvline in matplotlib) to indicate the different periods (put a vertical dashed line in 1963, in 1967, and in 1982).

B. It has been suggested that paired data such as this can be made to be closer to normally distributed by taking the log of each value before subtracting. Create two datasets: Q12 = streamflow1 - streamflow2 and Qlog12 = log(streamflow1)- log(streamflow2) and make graphs to demonstrate which is closer to normally distributed. Given that we want to use an ANOVA analysis, explain why is it important to check if a transformation might get the data closer to normally distributed?

C. State the null and the alternative hypothesis for the question of whether the four periods are statistically different from each other. State the type I error (alpha value) that you are willing to accept.

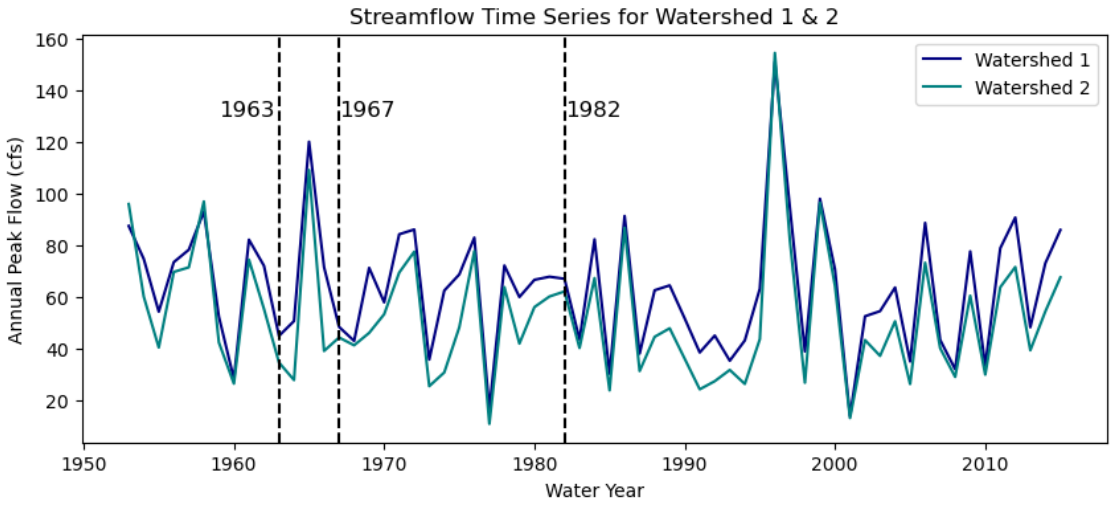
D. Perform an ANOVA test and discuss the results, related both to your hypothesis test listed above and to the more detailed question of which groups are statistically different from which other groups. Include graphs and/or tables that illustrate your results, and be sure to discuss what they mean. When using these ANOVA and other statistics functions, be sure that you understand what the code is doing (especially the defaults that different functions use) and outputting.

A.

[36]:
```
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
%matplotlib inline
```

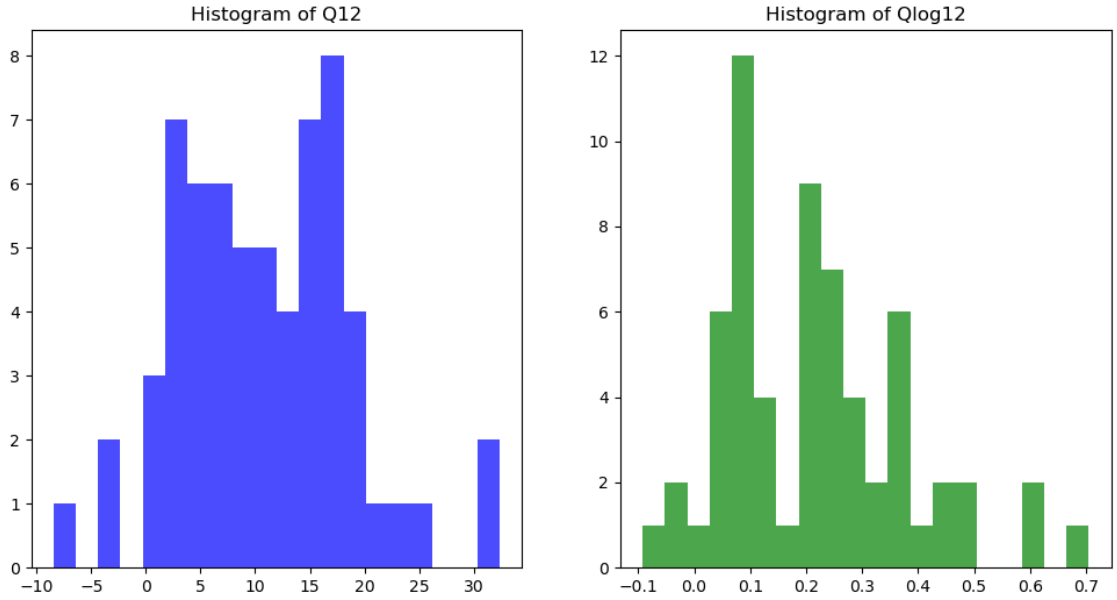
```
# Load the data
HJ_data_file = '../data/HJAndrews_peakflow_WS1_WS2_WS3.xlsx'
HJ_data = pd.read_excel(HJ_data_file, skiprows=2)

# Rename columns
HJ_data.columns = ['water year', 'WS1', 'WS2', 'WS3', 'Ind 1', 'Ind 2']

# Plotting time series for WS1 and WS2
fig, ax = plt.subplots(figsize=(10,4))


# Plot streamflow for both WS1 and WS2
ax.plot(HJ_data['water year'], HJ_data['WS1'], label='Watershed 1',␣
 ↪color='#000080')
ax.plot(HJ_data['water year'], HJ_data['WS2'], label='Watershed 2',␣
 ↪color='#008080')

# Add vertical dashed lines for the periods
ax.axvline(x=1963, color='k', linestyle='--')
plt.text(1959, 130, '1963',fontsize=12, color='k')
ax.axvline(x=1967, color='k', linestyle='--')
plt.text(1967, 130, '1967',fontsize=12, color='k')
ax.axvline(x=1982, color='k', linestyle='--')
plt.text(1982, 130, '1982',fontsize=12, color='k')

# Add titles and labels
ax.set_title('Streamflow Time Series for Watershed 1 & 2')
ax.set_xlabel('Water Year')
ax.set_ylabel('Annual Peak Flow (cfs)')
plt.legend(loc="best")
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/openpyxl/worksheet/_read_only.py:81:
UserWarning: Unknown extension is not supported and will be removed
  for idx, row in parser.parse():
```



B.

[38]:
```
# Create Q12 and Qlog12 datasets
Q12 = HJ_data['WS1'] - HJ_data['WS2']
Qlog12 = np.log(HJ_data['WS1']) - np.log(HJ_data['WS2'])

# Plot histograms
fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# Q12 distribution
ax[0].hist(Q12, bins=20, color='blue', alpha=0.7)
ax[0].set_title('Histogram of Q12')

# Qlog12 distribution
ax[1].hist(Qlog12, bins=20, color='green', alpha=0.7)
ax[1].set_title('Histogram of Qlog12')

plt.show()
```



It is important to check if a transformation may get the data closer to normally distributed when using an ANOVA analysis because ANOVA uses the F-distribution to determine significant differences between group means. F-statistics, however, use the assumption that the groups are all normally distributed, so the ANOVA uses the same assumption. If there's a significant deviation from a normal distribution then you will increase your chances of getting Type-II error. For this data, we can use the non-log-transformed data beacasue the histograms are not very symetric around a mean.

C.

Null Hypothesis (H): There is no mean difference in peak flows between WS1 and WS2 (it is the same across all four groups):

$$H_0 : \mu_{\Delta,1} = \mu_{\Delta,2} = \mu_{\Delta,3} = \mu_{\Delta,4}$$

where (_{ ,i}) is the mean difference in peak flows between WS1 and WS2 during period (i) (for (i = 1, 2, 3, 4)).

Alternative Hypothesis (H): At least one of the periods has a significantly different mean difference in peak flows than the other groups:

$$H_1 : i, j \,\text{such that}\, \mu_{\Delta,i} \neq \mu_{\Delta,j} \quad \text{for}\, i, j \in \{1, 2, 3, 4\}$$

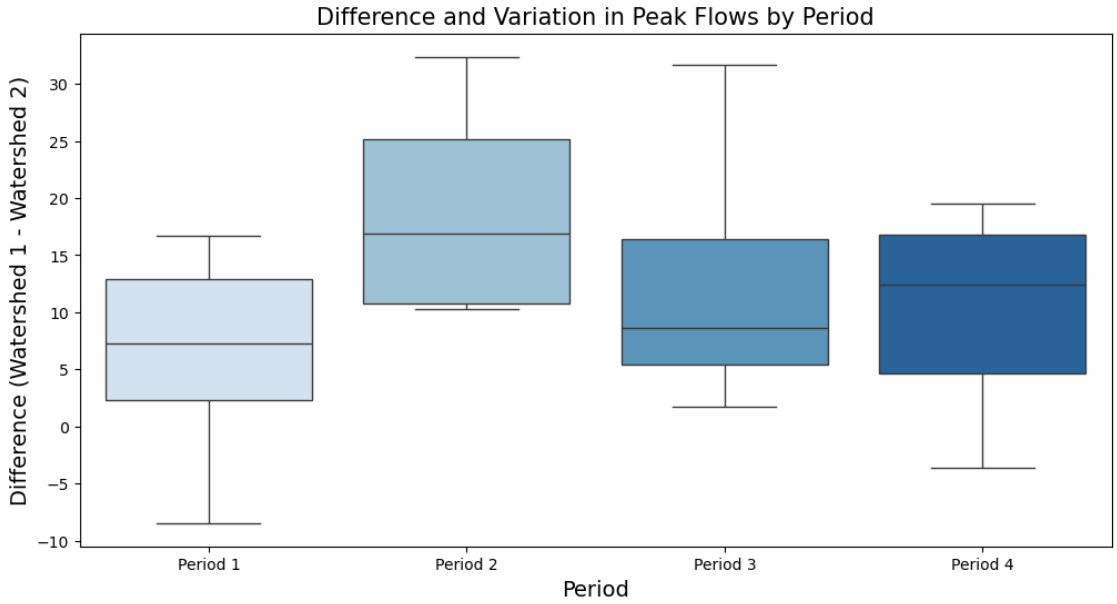
Type I Error (( )): Set (= 0.05). There is a 5% probability of rejecting the null hypothesis when it is actually true.

$$\alpha = 0.05$$

D.

[47]:
```
# Sort data into 4 groups based on the water year
HJ_data['Period'] = np.select(
    [(HJ_data['water year'] <= 1962),
     (HJ_data['water year'] >= 1963) & (HJ_data['water year'] <= 1966),
     (HJ_data['water year'] >= 1967) & (HJ_data['water year'] <= 1981),
     (HJ_data['water year'] >= 1982)],
    [1, 2, 3, 4])

# Calculate differences for each period (WS1 - WS2)
diff_period1 = pd.DataFrame({'Difference': HJ_data[HJ_data['Period'] ==␣
 ↪1]['WS1'] - HJ_data[HJ_data['Period'] == 1]['WS2']}).assign(Period='Period␣
 ↪1')
diff_period2 = pd.DataFrame({'Difference': HJ_data[HJ_data['Period'] ==␣
 ↪2]['WS1'] - HJ_data[HJ_data['Period'] == 2]['WS2']}).assign(Period='Period␣
 ↪2')
diff_period3 = pd.DataFrame({'Difference': HJ_data[HJ_data['Period'] ==␣
 ↪3]['WS1'] - HJ_data[HJ_data['Period'] == 3]['WS2']}).assign(Period='Period␣
 ↪3')
diff_period4 = pd.DataFrame({'Difference': HJ_data[HJ_data['Period'] ==␣
 ↪4]['WS1'] - HJ_data[HJ_data['Period'] == 4]['WS2']}).assign(Period='Period␣
 ↪4')

# Concatenate all differences into one DataFrame
df_diff = pd.concat([diff_period1, diff_period2, diff_period3, diff_period4],␣
 ↪ignore_index=True)

# Define a custom palette with different shades of blue
blue_palette = sns.color_palette("Blues", 4)

# Plot the differences using seaborn boxplot
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
sns.boxplot(x='Period', y='Difference', data=df_diff, palette=blue_palette)
plt.title('Difference and Variation in Peak Flows by Period', fontsize=15)
plt.xlabel('Period', fontsize=14)
plt.ylabel('Difference (Watershed 1 - Watershed 2)', fontsize=14)
plt.show()
```

```
/tmp/ipykernel_88/4089812771.py:26: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='Period', y='Difference', data=df_diff, palette=blue_palette)
```



[46]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd

# calculate ANOVA
# Group the data
period_1 = df_diff[df_diff['Period'] == 'Period 1']['Difference']
period_2 = df_diff[df_diff['Period'] == 'Period 2']['Difference']
period_3 = df_diff[df_diff['Period'] == 'Period 3']['Difference']
period_4 = df_diff[df_diff['Period'] == 'Period 4']['Difference']

fvalue, pvalue = stats.f_oneway(period_1, period_2, period_3, period_4)

print(f'F-value: {fvalue}')
print(f'P-value: {pvalue}')
```

```
F-value: 2.867860922868084
P-value: 0.04402714850976671
```

The p-value of 0.044 is less than the alpha threshold of 0.05, so we can reject the null hypothesis. This means there is a significant difference between mean peak flow differences in at least one of the groups.

[43]:
```
# perform multiple pairwise comparison (Tukey HSD)
m_comp = pairwise_tukeyhsd(endog=df_diff['Difference'],␣
  ↪groups=df_diff['Period'], alpha=0.05)

print(m_comp)
```

```
   Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1   group2  meandiff p-adj   lower    upper  reject
-----------------------------------------------------
Period 1 Period 2  12.8226 0.0292   0.9605 24.6847   True
Period 1 Period 3    5.498 0.2951  -2.6876 13.6836  False
Period 1 Period 4   4.3973   0.38  -2.8157 11.6103  False
Period 2 Period 3  -7.3246 0.3245 -18.6077  3.9585  False
Period 2 Period 4  -8.4253 0.1645 -19.0239  2.1734  False
Period 3 Period 4  -1.1007 0.9657  -7.3157  5.1143  False
-----------------------------------------------------
```

The Tukey HSD test demostrates that Period 1 & 2 have a significant mea difference with a p-value of 0.0292 which is less than the alpha threshold of 0.05. None of the other group differences had a signficant p-value. This identifies Period 1 & 2 as having the statistically significant change in the mean between the two periods with a confidence of 95%. So, we can reject the null hypothesis for this group. We cannot reject the null for any other groups.

0.1.2 Problem 2: Linear and Quantile Regression

Streamflow records for the Columbia River measured at The Dalles, Oregon, began in water year 1879 and continues to the present day (this is one of the longest continuous records of streamflow in the U.S.). Peak flow records extend back to 1858 (based on peak stage values recorded by railroad workers). Using the coincident peak flow records from 1879-1933 (also a period with no major storage dams on the Columbia), create models to predict annual flow for years 1858-1878:

A. Isolate the period of relevant overlap for annual and peak flow records (1879-1933) and plot a timeseries of these measurements. Then create a least-squares linear regression model to predict annual flow using peak flow as an explanatory variable.

B. How much of the true variance of annual flow is explained by the resulting model?

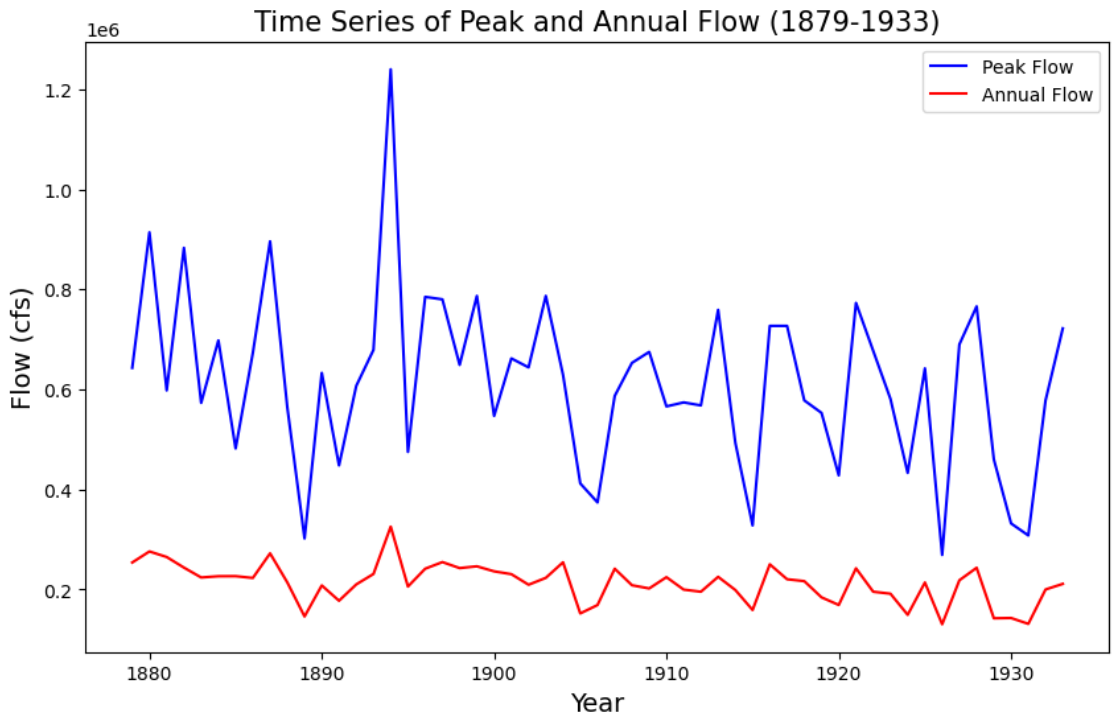
C. Now create a non-parametric, quantile-based regression model using the same data.

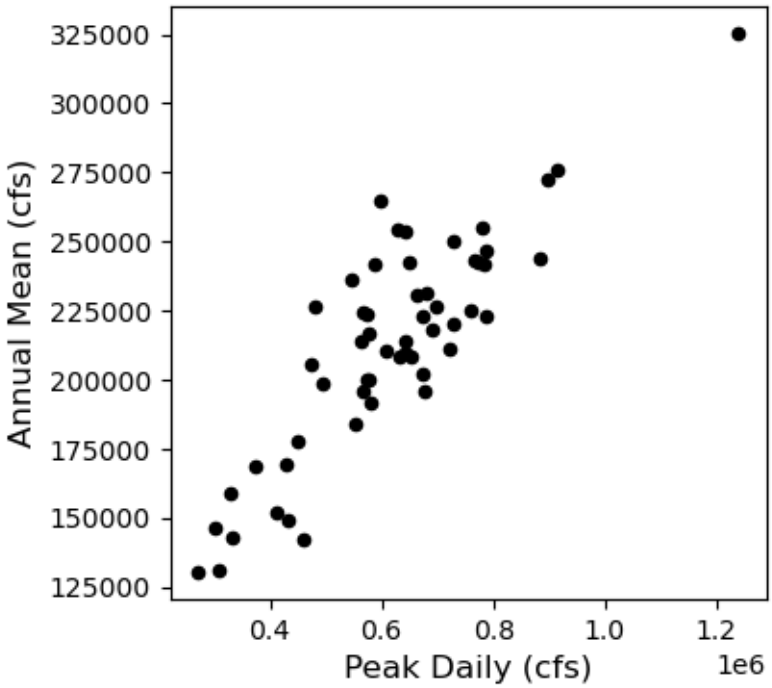
D. Plot the predictions and residuals for the two different prediction models for the training period (1879-1933), and plot the model predictions for the 1858-1878 data for the two different models. Is there a substantial difference between the two model formulations? Discuss any differences that you observe.

A.

[106]:
```
from sklearn.linear_model import LinearRegression

# Load the data
columbia_data_file='../data/dalles_flow.csv'
columbia_data=pd.read_csv(columbia_data_file, skiprows=2)

# Rename columns
columbia_data.columns = ['year', 'peak_flow', 'annual_flow']
```

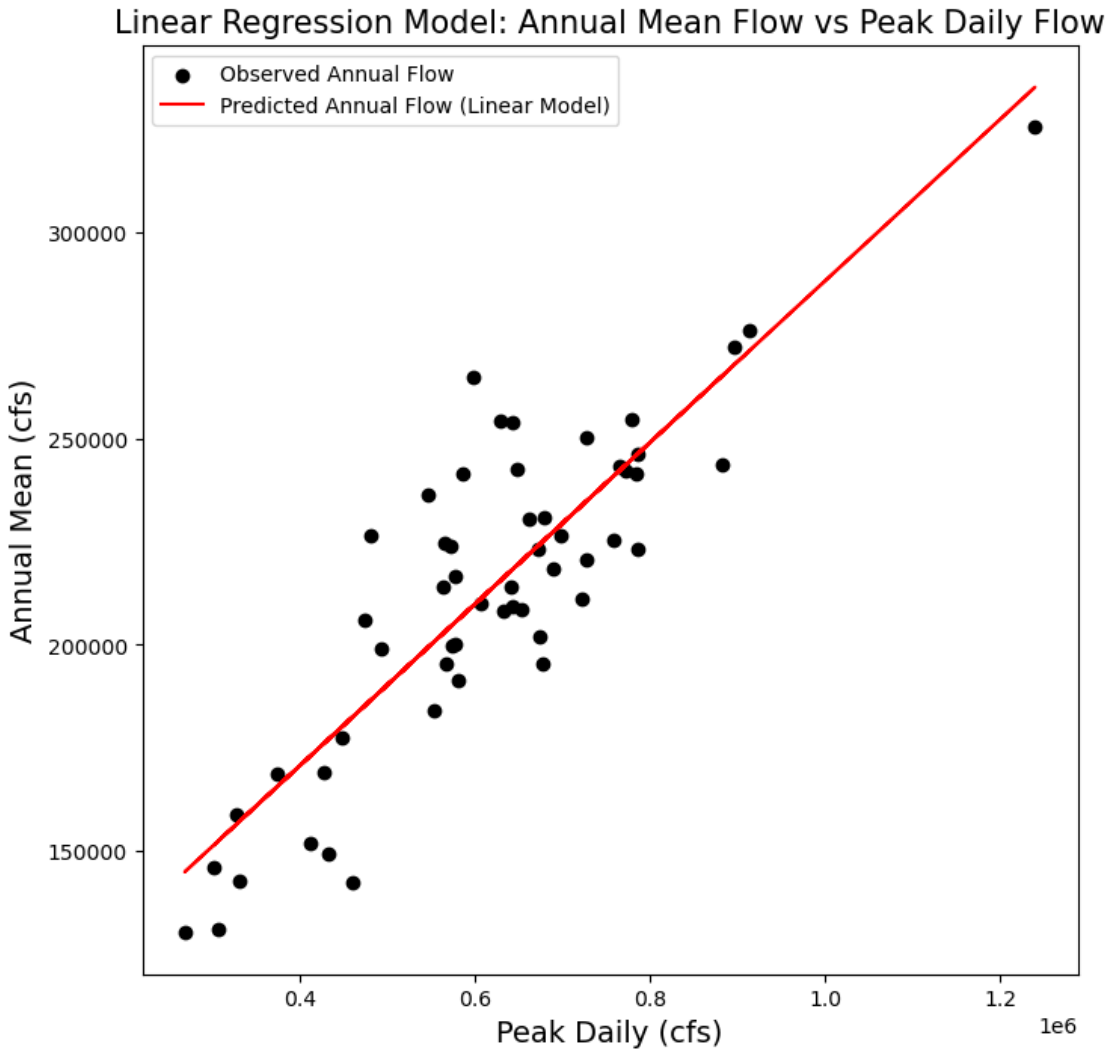
```
# Isolate the period from 1879-1933
data_overlap = columbia_data[(columbia_data['year'] >= 1879) &␣
 ↪(columbia_data['year'] <= 1933)]

# Plot time series of peak flow and annual flow for the overlap period
plt.figure(figsize=(10, 6))
plt.plot(data_overlap['year'], data_overlap['peak_flow'], label='Peak Flow',␣
 ↪color='blue')
plt.plot(data_overlap['year'], data_overlap['annual_flow'], label='Annual␣
 ↪Flow', color='red')
plt.title('Time Series of Peak and Annual Flow (1879-1933)', fontsize=15)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Flow (cfs)', fontsize=14)
plt.legend()
plt.show()

# Make a scatterplot
fig, ax = plt.subplots(figsize=(4, 4))
data_overlap.plot.scatter(x='peak_flow', y='annual_flow', c='k', ax=ax)
ax.set_xlabel('Peak Daily (cfs)', fontsize=12)
ax.set_ylabel('Annual Mean (cfs)', fontsize=12)
plt.show()
```





[120]:
```
# Create least-squares regression model
X = data_overlap[['peak_flow']]
y = data_overlap['annual_flow']

# Fit the regression model
linear_model = LinearRegression()
linear_model.fit(X, y)

# Make predictions using the model
annual_flow_pred = linear_model.predict(X)

# Plot the regression line
plt.figure(figsize=(8, 8))
plt.scatter(X, y, color='black', label='Observed Annual Flow')
plt.plot(X, annual_flow_pred, color='red', label='Predicted Annual Flow (Linear␣
 ↪Model)')
plt.title('Linear Regression Model: Annual Mean Flow vs Peak Daily Flow',␣
 ↪fontsize=15)
plt.xlabel('Peak Daily (cfs)', fontsize=14)
plt.ylabel('Annual Mean (cfs)', fontsize=14)
plt.legend()
plt.show()

# Calculate coefficients
```

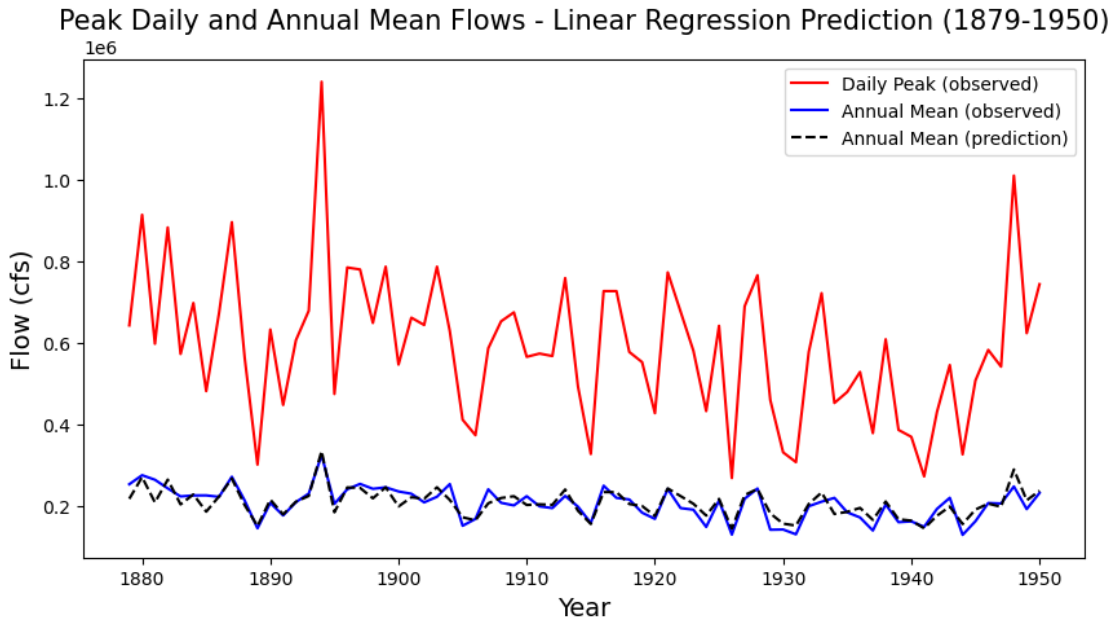
```
B1 = linear_model.coef_[0]
B0 = linear_model.intercept_

print('B0 : {}'.format(np.round(B0, 4)))
print('B1 : {}'.format(np.round(B1, 4)))

# Predicted values
y_predicted = B0 + B1 * data_overlap['peak_flow']
residuals = y - y_predicted

# SSE (Sum of Squared Errors)
sse = np.sum((y - y_predicted) ** 2)

# SST (Total Sum of Squares)
sst = np.sum((y - np.mean(y)) ** 2)

# R-squared
r_squared = 1 - sse / sst
r = np.sqrt(r_squared)

# Standard deviation of residuals
n = len(data_overlap)  # Length of our filtered dataset
sigma = np.sqrt(sse / (n - 2))

# Output results
print('SSE : {} cfs'.format(np.round(sse, 2)))
print('SST : {} cfs'.format(np.round(sst, 2)))
print('R^2 : {}'.format(np.round(r_squared, 3)))
print('R : {}'.format(np.round(r, 3)))
print('sigma : {}'.format(np.round(sigma, 3)))
```




```
B0 : 92295.1358
B1 : 0.1958
SSE : 20843038698.27 cfs
SST : 84348481248.51 cfs
R^2 : 0.753
R : 0.868
sigma : 19830.907
```

[135]:
```
# Linear regression prediction
pred_annual = B1 * columbia_data['peak_flow'] + B0

# Filter the data to include only the range from 1879 to 1950
filtered_data = columbia_data[(columbia_data['year'] >= 1879) &␣
 ↪(columbia_data['year'] <= 1950)]
```

```
# Create the plot
fig, ax = plt.subplots(figsize=(10, 5))

# Plot the filtered data
ax.plot(filtered_data['year'], filtered_data['peak_flow'], c='r', label='Daily␣
 ↪Peak (observed)')
ax.plot(filtered_data['year'], filtered_data['annual_flow'], c='b',␣
 ↪label='Annual Mean (observed)')
ax.plot(filtered_data['year'], B1 * filtered_data['peak_flow'] + B0, c='k',␣
 ↪linestyle='--', label='Annual Mean (prediction)')

ax.set_title('Peak Daily and Annual Mean Flows - Linear Regression Prediction␣
 ↪(1879-1950)', fontsize=15)
ax.set_xlabel('Year', fontsize=14)
ax.set_ylabel('Flow (cfs)', fontsize=14)
plt.legend()
plt.show()
```



B. 73.5% of the variance in the true annual flow can be explained by the linear regression model because our R^2-value was 0.735.

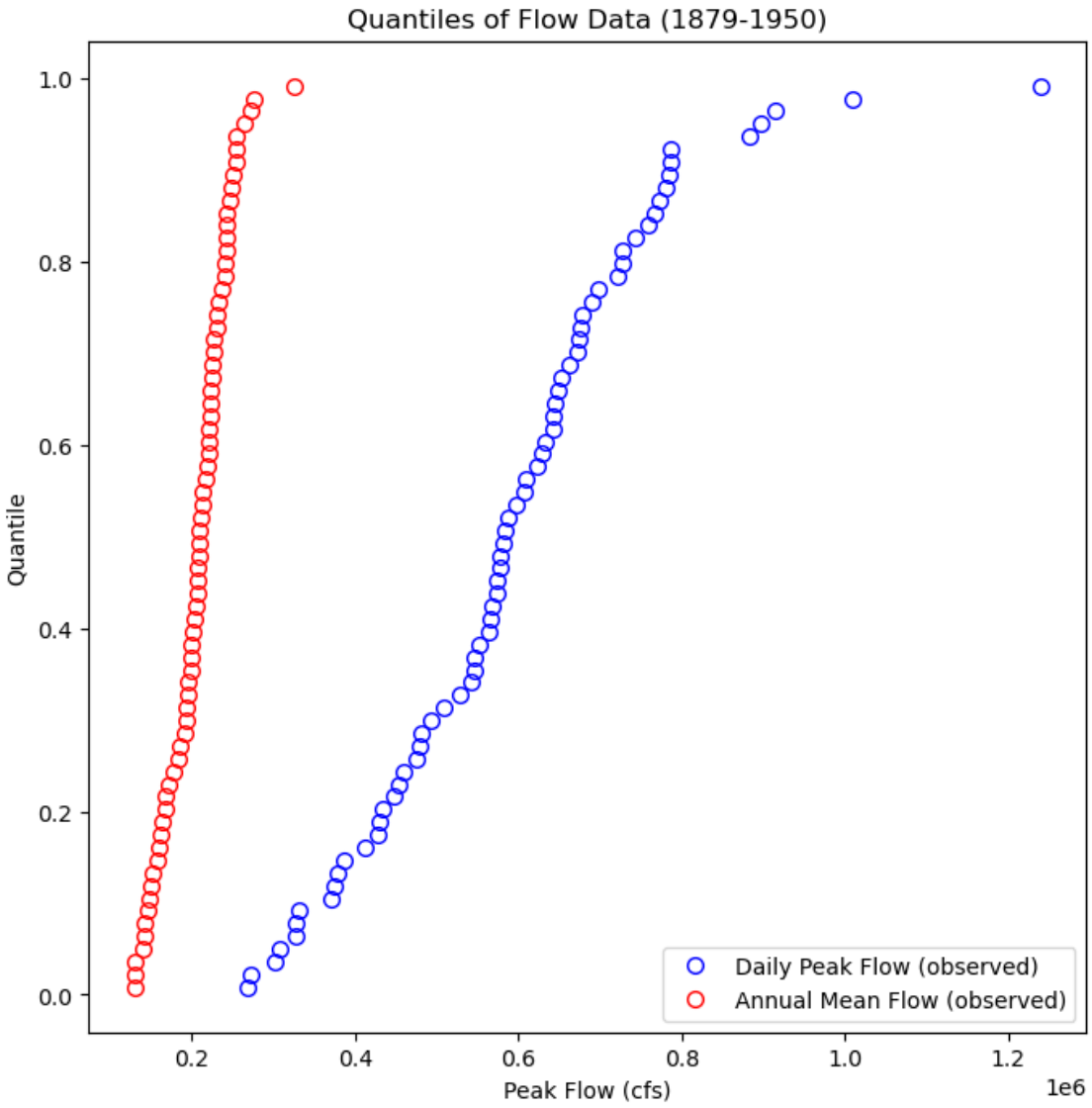
C.

[122]:

```
# This function should be able to accept any one-dimensional numpy array or␣
 ↪list, of numbers
# It returns two numpy arrays, one of the sorted numbers, the other of the␣
 ↪plotting position
def cunnane_quantile_array(numbers):
    '''This function also computes the Cunnane plotting position given an array␣
 ↪or list of numbers (rather than a pandas dataframe).
    It has two outputs, first the sorted numbers, second the Cunnane plotting␣
 ↪position for each of those numbers.
    [Steven Pestana, spestana@uw.edu, Oct. 2020]'''

    # 1) sort the data, using the numpy sort function (np.sort())
    sorted_numbers = np.sort(numbers)

    # length of the list of numbers
    n = len(sorted_numbers)

    # make an empty array, of the same length. below we will add the plotting␣
 ↪position values to this array
    cunnane_plotting_position = np.empty(n)

    # 2) compute the Cunnane plotting position for each number, using a for␣
 ↪loop and the enumerate function
    for rank, number in enumerate(sorted_numbers):
        cunnane_plotting_position[rank] = ( (rank+1) - (2/5) ) / ( n + (1/5) )

    return sorted_numbers, cunnane_plotting_position
```

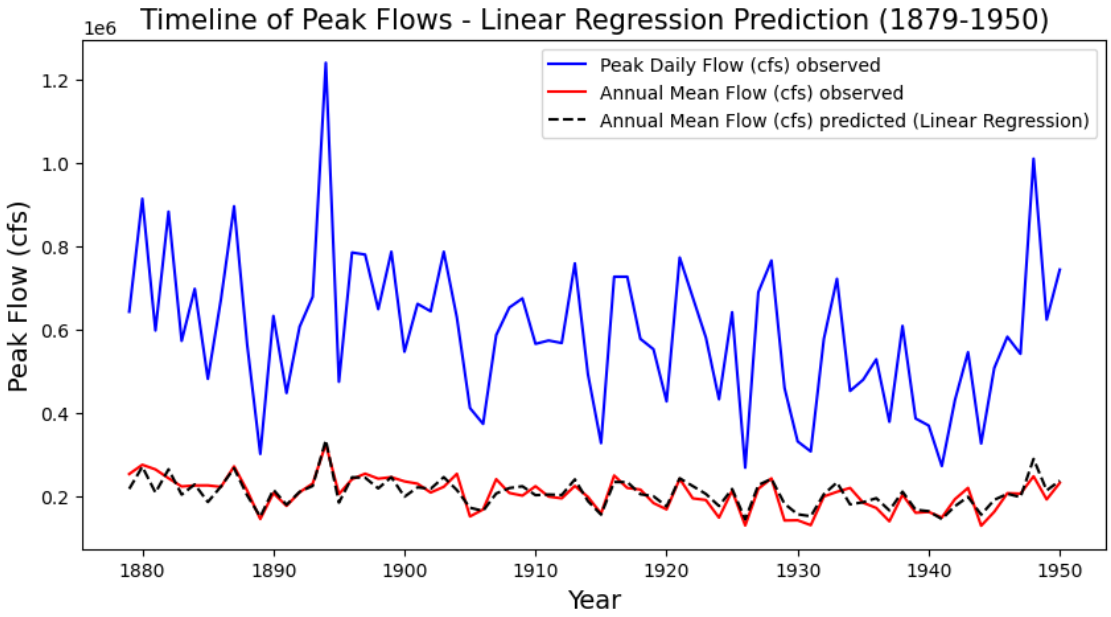
[132]:
```
# Filter data to include only the range from 1879 to 1950
filtered_data = columbia_data[(columbia_data['year'] >= 1879) &␣
 ↪(columbia_data['year'] <= 1950)]

plt.figure(figsize=(8, 8))

daily_ordered, daily_quantile =␣
 ↪cunnane_quantile_array(filtered_data['peak_flow'])
annual_ordered, annual_quantile =␣
 ↪cunnane_quantile_array(filtered_data['annual_flow'])
plt.plot(daily_ordered, daily_quantile, 'o', markeredgecolor='b',␣
 ↪markerfacecolor='None', markersize=7, label='Daily Peak Flow (observed)')
plt.plot(annual_ordered, annual_quantile, 'o', markeredgecolor='r',␣
 ↪markerfacecolor='None', markersize=7, label='Annual Mean Flow (observed)')

plt.ylabel('Quantile')
plt.xlabel('Peak Flow (cfs)')
plt.title('Quantiles of Flow Data (1879-1950)')
```

```
plt.legend(loc="lower right")
plt.show()
```



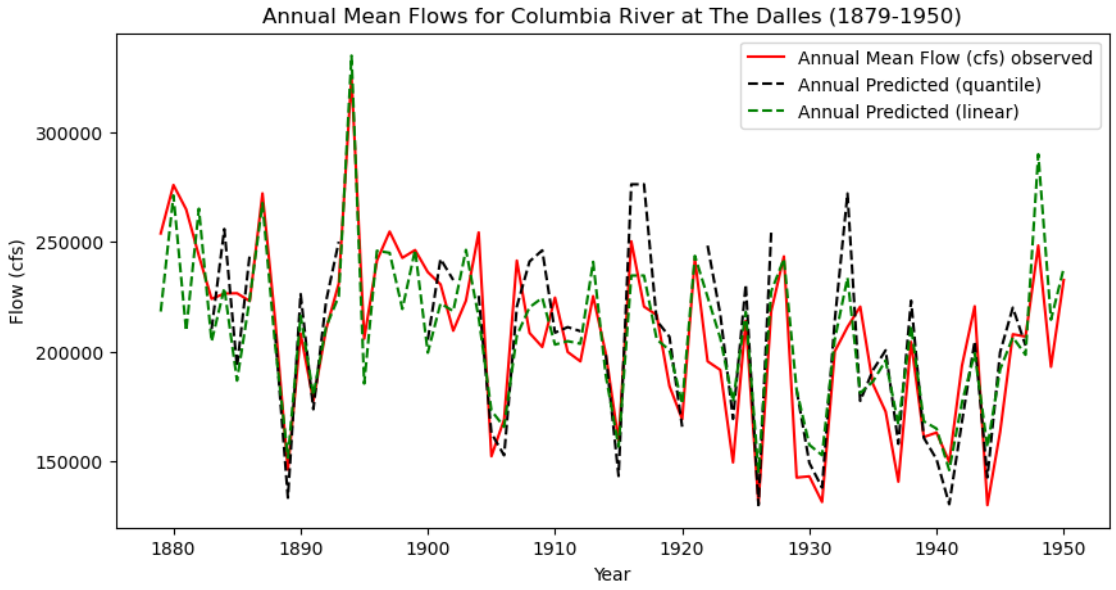
[168]:
```
# Define quantiles
quantiles = np.linspace(0, 1, 100)

# Compute empirical quantiles for annual and daily flows
annual_ordered = stats.mstats.mquantiles(columbia_data['annual_flow'],␣
 ↪quantiles)
daily_ordered = stats.mstats.mquantiles(columbia_data['peak_flow'], quantiles)

# Interpolating functions for daily and annual data
```

```
f_daily = interp1d(daily_ordered, quantiles, bounds_error=False,␣
 ↪fill_value="extrapolate")
g_annual = interp1d(quantiles, annual_ordered, bounds_error=False,␣
 ↪fill_value="extrapolate")

# Filter data
filtered_data = columbia_data[(columbia_data['year'] >= 1879) &␣
 ↪(columbia_data['year'] <= 1950)]

# Predicted annual flow using quantile regression
annual_predicted = g_annual(f_daily(filtered_data['peak_flow']))
residuals = (filtered_data['annual_flow'] - annual_predicted)

# Alternatively, use linear regression prediction
pred_annual_linear = B1 * filtered_data['peak_flow'] + B0  # Linear regression␣
 ↪prediction

# Plotting the original data and predictions
plt.figure(figsize=(10, 5))
plt.plot(filtered_data['year'], filtered_data['peak_flow'], 'b-', label='Peak␣
 ↪Daily Flow (cfs) observed')
plt.plot(filtered_data['year'], filtered_data['annual_flow'], 'r-',␣
 ↪label='Annual Mean Flow (cfs) observed')
plt.plot(filtered_data['year'], pred_annual_linear, 'k--', label='Annual Mean␣
 ↪Flow (cfs) predicted (Linear Regression)')
plt.legend()
plt.title('Timeline of Peak Flows - Linear Regression Prediction (1879-1950)',␣
 ↪fontsize=15)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Peak Flow (cfs)', fontsize=14)
plt.show()
```



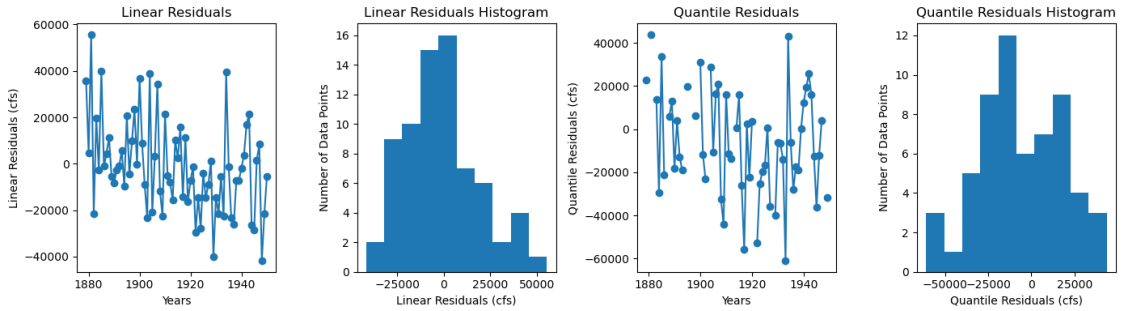
D.

[181]:
```
# Filter data to include only the range from 1879 to 1950
filtered_data = columbia_data[(columbia_data['year'] >= 1879) &␣
 ↪(columbia_data['year'] <= 1950)]

# Plot the annual mean flow and predictions
plt.figure(figsize=(10, 5))
plt.plot(filtered_data['year'], filtered_data['annual_flow'], 'r-',␣
 ↪label='Annual Mean Flow (cfs) observed')
plt.plot(filtered_data['year'], annual_predicted, 'k--', label='Annual␣
 ↪Predicted (quantile)')
plt.plot(filtered_data['year'], pred_annual[filtered_data.index], c='g',␣
 ↪linestyle='--', label='Annual Predicted (linear)')  # Aligning indices

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Flow (cfs)')
plt.title('Annual Mean Flows for Columbia River at The Dalles (1879-1950)')
plt.legend()
plt.show()
```



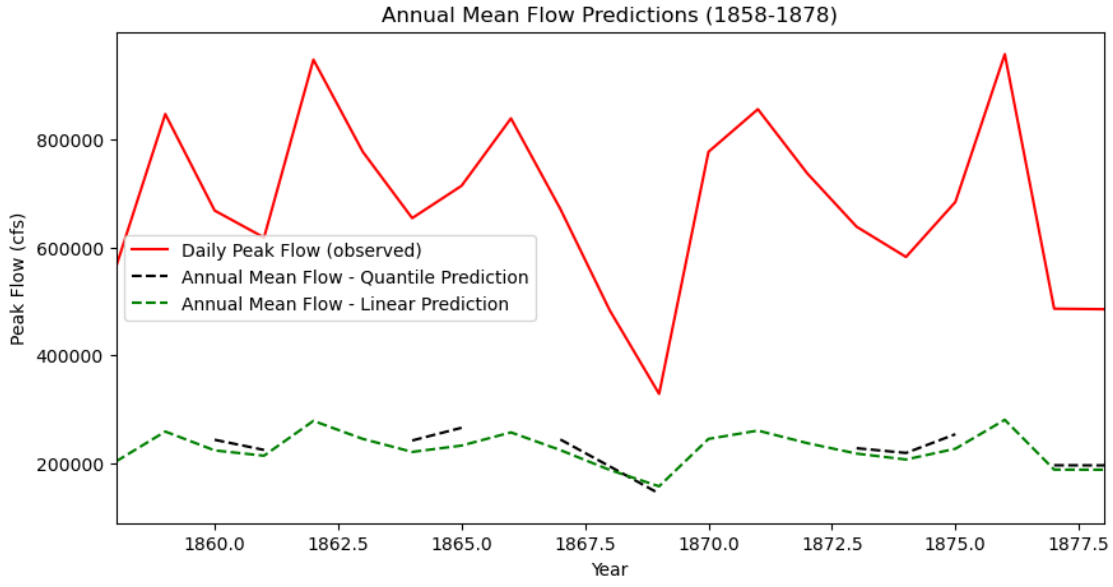
[154]:
```
# Filter data to include only the range from 1879 to 1950
filtered_data = columbia_data[(columbia_data['year'] >= 1879) &␣
 ↪(columbia_data['year'] <= 1950)]

# Linear regression residuals
lin_res = filtered_data['annual_flow'] - (B1 * filtered_data['peak_flow'] + B0)␣
 ↪ # Assuming B1 and B0 are defined

# Create subplots for residuals
fig, (ax1, ax2, ax3, ax4) = plt.subplots(nrows=1, ncols=4, figsize=(14, 4),␣
 ↪tight_layout=True)

# Linear regression residuals
ax1.plot(filtered_data['year'], lin_res, '-o')
ax1.set_title('Linear Residuals')
ax1.set_xlabel('Years')
ax1.set_ylabel('Linear Residuals (cfs)')

# Linear regression histogram
ax2.hist(lin_res, bins=10)
ax2.set_title('Linear Residuals Histogram')
ax2.set_xlabel('Linear Residuals (cfs)')
ax2.set_ylabel('Number of Data Points')

# Quantile regression residuals
quantile_predicted = g_annual(f_daily(filtered_data['peak_flow']))
residuals = filtered_data['annual_flow'] - quantile_predicted
```

```
ax3.plot(filtered_data['year'], residuals, '-o')
ax3.set_title('Quantile Residuals')
ax3.set_xlabel('Years')
ax3.set_ylabel('Quantile Residuals (cfs)')

# Quantile residuals histogram
ax4.hist(residuals, bins=10)
ax4.set_title('Quantile Residuals Histogram')
ax4.set_xlabel('Quantile Residuals (cfs)')
ax4.set_ylabel('Number of Data Points')

plt.show()
```



[184]:
```
B0 = 92295.1358
B1 = 0.1958
f_daily = interp1d(daily_ordered, quantiles, bounds_error=False,␣
 ↪fill_value="extrapolate")
g_annual = interp1d(quantiles, annual_ordered, bounds_error=False,␣
 ↪fill_value="extrapolate")


# Generate quantile predictions for the entire dataset
quant_annual_pred = g_annual(f_daily(columbia_data['peak_flow']))  # Apply␣
 ↪quantile regression on the entire dataset
lin_annual_pred = B1 * columbia_data['peak_flow'] + B0  # Linear predictions␣
 ↪for the entire dataset

# Filter data for plotting: focus on the 1858-1878 range for observed data
data_pred = columbia_data[(columbia_data['year'] >= 1858) &␣
 ↪(columbia_data['year'] <= 1878)]

# Plot observed daily peak flow and predictions
plt.figure(figsize=(10, 5))
```

```
# Plot observed daily peak flow for the specified range (1858-1878)
plt.plot(data_pred['year'], data_pred['peak_flow'], 'r-', label='Daily Peak␣
 ↪Flow (observed)')

# Plot quantile regression predictions for the entire dataset
plt.plot(columbia_data['year'], quant_annual_pred, 'k--', label='Annual Mean␣
 ↪Flow - Quantile Prediction')

# Plot linear regression predictions for the entire dataset
plt.plot(columbia_data['year'], lin_annual_pred, 'g--', label='Annual Mean Flow␣
 ↪- Linear Prediction')

# Add labels and title
plt.title('Annual Mean Flow Predictions (1858-1878)')
plt.xlabel('Year')
plt.ylabel('Peak Flow (cfs)')
plt.legend()

# Set x-axis limits to focus on the specified range (1858-1878)
plt.xlim(1858, 1878)

# Display the plot
plt.show()
```



Having issues getting the quatile to plot correctly again, however, where visible, there isn't a huge difference in the two prediction models, they follow a very similar path. Linear appears to have dampened slopes and peak values compared the the slightly heighened or more dramatic slopes and values on the quantile. Both appear to follow the curve of the Daily Peak flows overall, though.

Code adapted from labs 3-3, 4-1, 4-2, Stack Overflow, and assistance from Abby Crandall.

[ ]: