Anascombe Data Visualisation

October 9, 2025

1 Anascombe's Quartet: Exploratory Data Analysis

Date October 8, 2025
Author: Sanchit Khope (@sanchitkhope3)

1.1 1. Abstract

In this report we dive deeper into statistics and follow proper exploratory data visualization techniques in order to understand Anscombe's Quartet and the four datasets that come with it. We create several graphs, like scatterplot graphs, violin plots, box plots, and regression lines, in order to see how the datasets are seen on a visual scale. The goal of this report is to understand how simple summary statistics about topics such as mean, variance, standard deviation, covariance, correlation, regression coefficients, and R^2 do not show the entire picture.

Using several Python modules like matplotlib, seaborn, numpy, and pandas, we can clearly calculate both the visual graphs and the statistical summaries in order to gain full insights on what this data represents. Some datasets differ from others, with some with linear trends, some nonlinear, and some with no active trends found. This report lets us understand the importance of both aspects of understanding data, allowing analysts to find patterns, mistakes, and issues in large data sets through both summaries and a visual output through code, which can be easily altered to process a larger amount of data.

1.2 2. Introduction

Anscombe's Quartet is one of the most famous examples given in many statistics courses, and it represents one of the main issues that many analysts may face. Anscombe's Quartet is made up of 4 datasets, each being carefully made to present the same exact statistical summaries and values, meaning that the mean, variance, correlation, and regression coefficients are all the same. If an analyst only looked at the statistical values, without using any other methods to cross-check, they would never be able to find any anomalies, outliers, and discrepancies within the datasets, which would lead to issues in future projects.

This is where the exploratory data analysis part of the assignment comes in. Exploratory Data Analysis is a mixture of both summary statistics and data visualization, made to solve the exact issue that Anscombe's Quartet exemplifies. It finds statistical summaries and also helps to find outliers, understand variables, and see patterns within the distribution of the different datasets. Exploratory data analysis is used in both science- and business-based fields, allowing for greater understanding of variables with projects and the financial markets to reduce loss, process risks, and understand relationships that would be overlooked through traditional summaries. Combining both code and statistics, it makes understanding these relationships all the easier and provides fast understanding of the correlation of two or more variables in a dataset.

1.3 3. Methods

Statistical Summaries:

- Mean - the Average Values of X and Y
- Variance - How far apart the x and y values are
- Standard Deviation (SD) - How Far The Points Differ From the Average
- Covariance - The Similarities between the Movement of the X and Y values
- Correlation - How strongly the X and Y values move together
- Linear Regression - The Line of Best Fit between the points
- R^2 (Coefficient of Determination) - How the regression line fits the data given (on a scale from 0-1)

1.3.1 Mean Formula:

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i, \quad \bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

1.3.2 Variance Formula:

$$\mathrm{Var}(X) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2, \quad \mathrm{Var}(Y) = \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2$$

1.3.3 Standard Deviation (SD) Formula:

$$\sigma_X = \sqrt{\mathrm{Var}(X)}, \quad \sigma_Y = \sqrt{\mathrm{Var}(Y)}$$

1.3.4 Covariance Formula:

$$\mathrm{Cov}(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$$

1.3.5 Correlation Formula:

$$r = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y}$$

1.3.6 Linear Regression Formula:

$$Y = \beta_0 + \beta_1 X$$

1.3.7 R^2 (Coefficient of Determination) Formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

1.4 Visuals Used:

- Scatter Plots with Regression Lines - Used to Show the 4 Datasets With Regression Line to Plot Out the Correlation Between Them
- Residual Plots - To Understand Where the Clear Outliers are, and How Much they Differ from the Regression Line
- Box Plots - To Compare the Distribution Between X and Y
- Violin Plots - To Do the Same as a Box Plot but in a More Easy to Understand Manner
- Overlaid Scatter Plot - To Compare the Trends and Relations Between All 4 Datasets At Once

---

1.5 Imports and Data Path

[5]:
```python
import pandas as pd
import numpy as np
from pathlib import Path
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns


DATA_PATH = Path("Anscombe_quartet_data - Anscombe_quartet_data.csv") #creates␣
 ↪a path to the .csv file so no manual inputs

df = pd.read_csv('Anscombe_quartet_data - Anscombe_quartet_data.csv') #creates␣
 ↪a database with it

thedata = {
    'I': ('x123', 'y1'),
    'II': ('x123', 'y2'),
    'III': ('x123', 'y3'),
    'IV': ('x4', 'y4')
} #makes labels for the different columns
```

1.6 4. Statistical Summaries

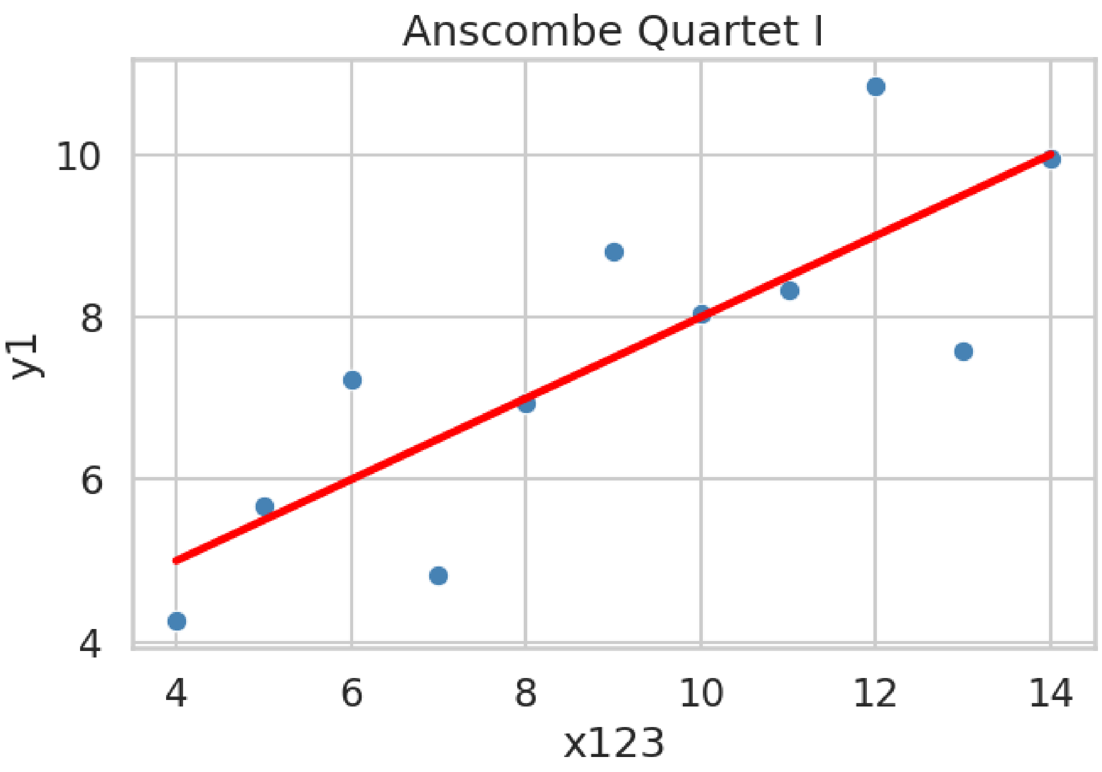
[6]:
```python
def stats_recap(df, x_col, y_col): #creates a function

    x = df[x_col]
    y = df[y_col]

    n = len(x)
    mean_x = x.mean() # Mean of x
    mean_y = y.mean() # Mean Of Y
    var_x = x.var(ddof=1) # Variance of X
    var_y = y.var(ddof=1) # Variance of Y
    std_x = x.std(ddof=1) # Standard Deviation of X
    std_y = y.std(ddof=1) # Standard Deviation of Y
    cov_xy = x.cov(y) # Covariance of X and Y
    corr_xy = x.corr(y) # Correlation of X and Y

    X = sm.add_constant(x)
    fit = sm.OLS(y, X).fit()
    rslope = fit.params[x_col]
    intercept = fit.params['const']
    rsquare = fit.rsquared # Coefficient of Determination

    return pd.Series({
        'n': n,
        'mean_x': mean_x, 'mean_y': mean_y,
        'var_x': var_x, 'var_y': var_y,
        'std_x': std_x, 'std_y': std_y,
        'cov_xy': cov_xy, 'corr_xy': corr_xy,
        'slope': rslope, 'intercept': intercept, 'rsquare': rsquare
        })

recap_table = pd.DataFrame({
    name: stats_recap(df, x_col, y_col)
    for name, (x_col, y_col) in thedata.items()
}).T

recap_rounded = recap_table.round(3)


print(recap_rounded)
```

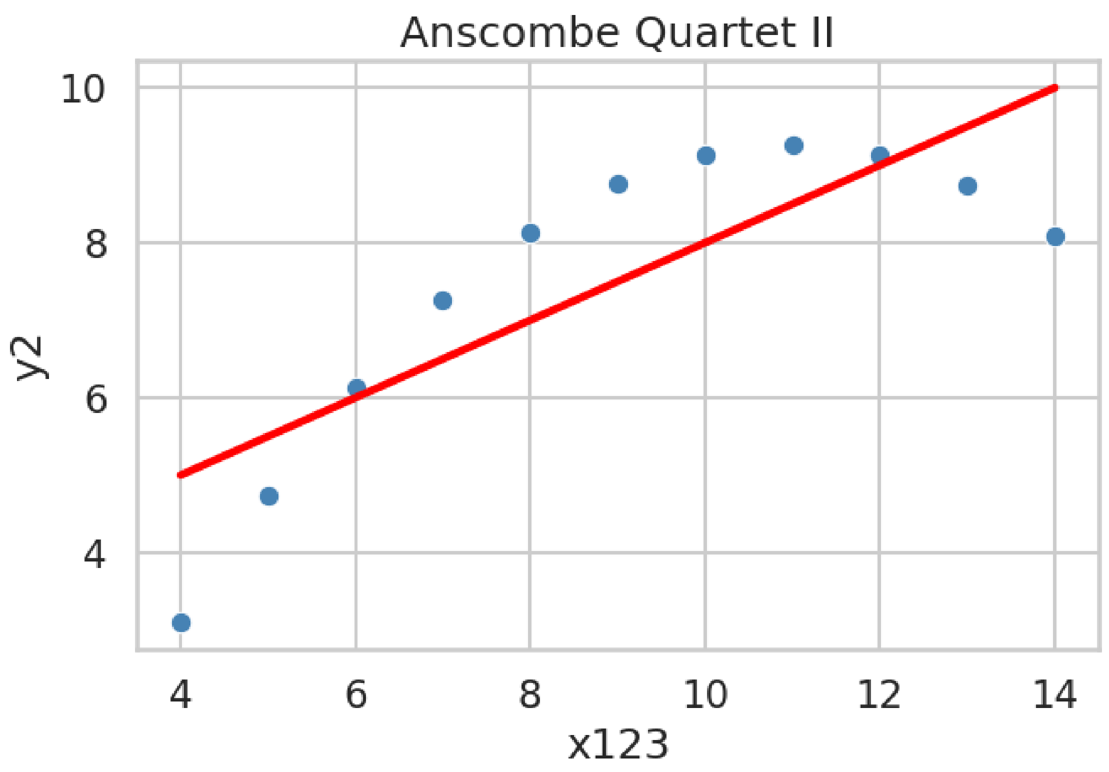
```
        n  mean_x  mean_y  var_x  var_y  std_x  std_y  cov_xy  corr_xy  slope  \
I    11.0     9.0   7.501   11.0  4.127  3.317  2.032   5.501    0.816    0.5
II   11.0     9.0   7.501   11.0  4.128  3.317  2.032   5.500    0.816    0.5
III  11.0     9.0   7.500   11.0  4.123  3.317  2.030   5.497    0.816    0.5
IV   11.0     9.0   7.501   11.0  4.123  3.317  2.031   5.499    0.817    0.5

     intercept  rsquare
I        3.000    0.667
II       3.001    0.666
III      3.002    0.666
IV       3.002    0.667
```

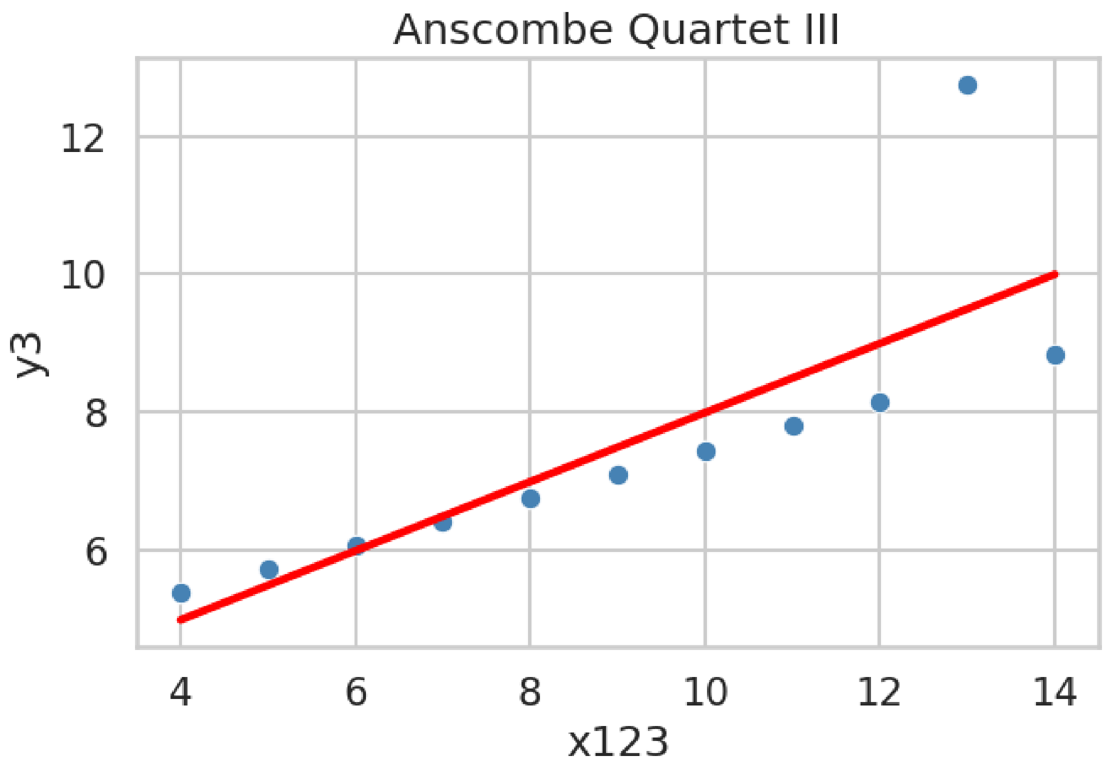
1.7 5. Visualisations and Code

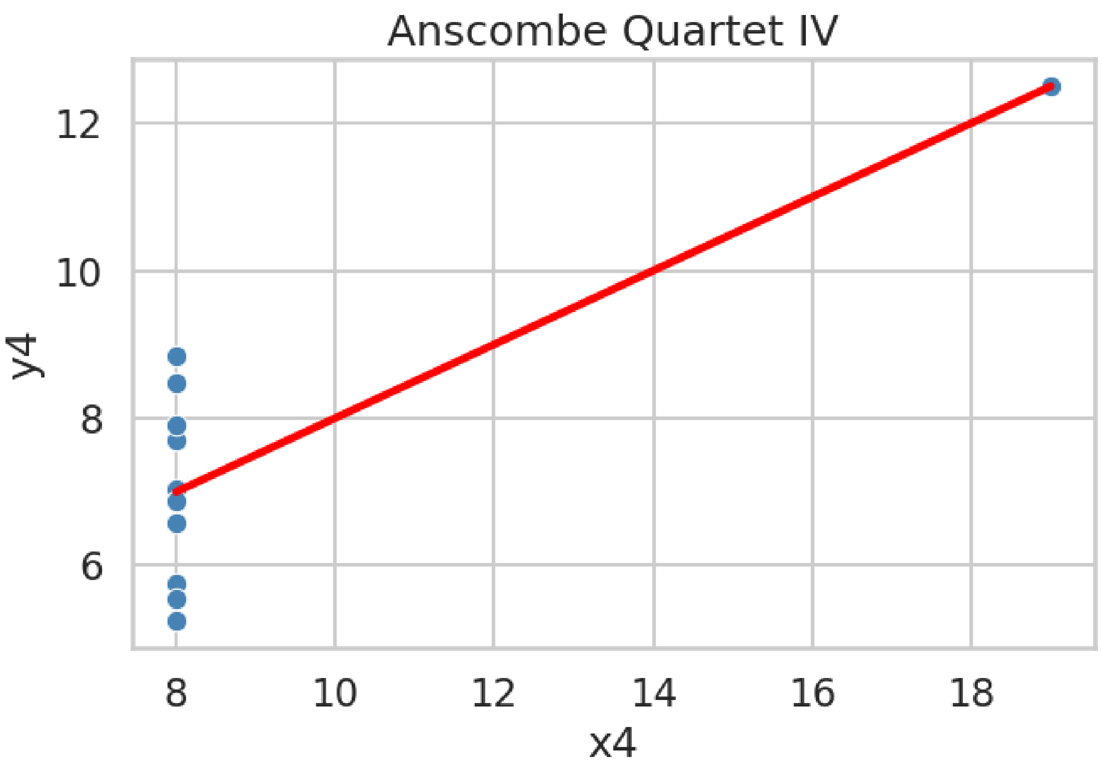
1.7.1 Individual Scatter Plots:

[24]:
```
def plottingtime(df, x_col, y_col, title): #creates function to plot them
    sns.set(style="whitegrid", context="talk")
    plt.figure(figsize=(7,5)) #adjusts size of the graphs
    sns.scatterplot(x=x_col,y=y_col, data=df, s=80, color= "steelblue")
    sns.regplot(x=x_col, y=y_col, data=df, scatter=False, color="red", ci=None)
    plt.title(f"Anscombe Quartet {name}")
    plt.xlabel(x_col)
    plt.ylabel(y_col)
    plt.tight_layout()
    plt.show()

for name, (x_col, y_col) in thedata.items():
    plottingtime(df, x_col, y_col, f"Anscombe Quartet {name}") #runs the␣
 ↪plotting time function
```



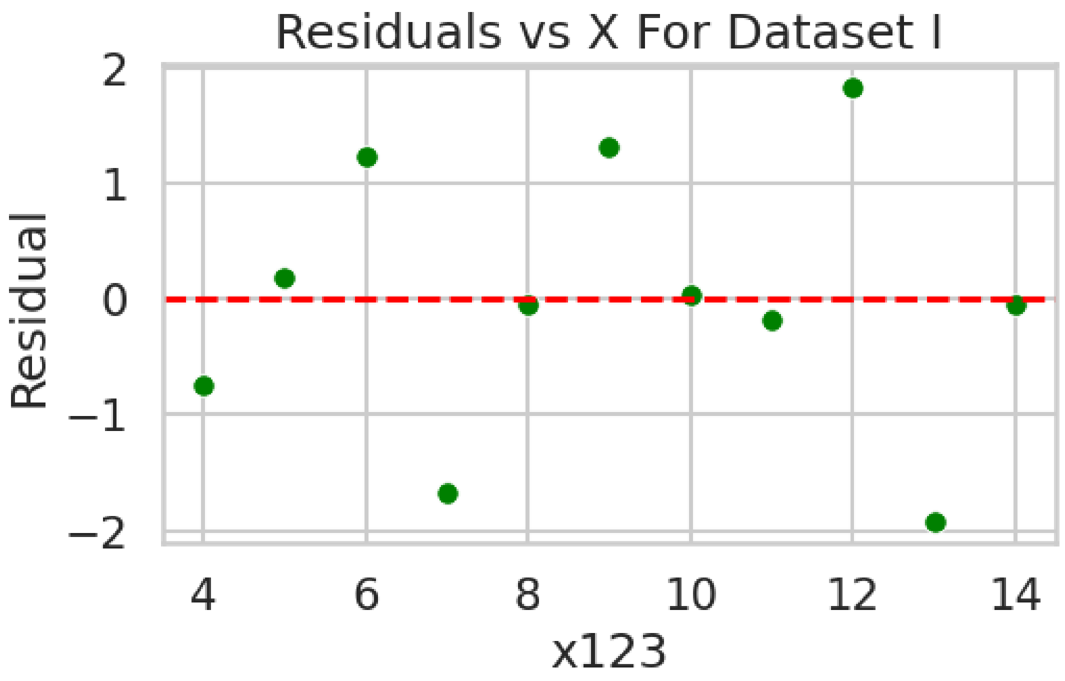

Anscombe Quartet II
y2
x123
10
8
6
4
4
6
8
10
12
14

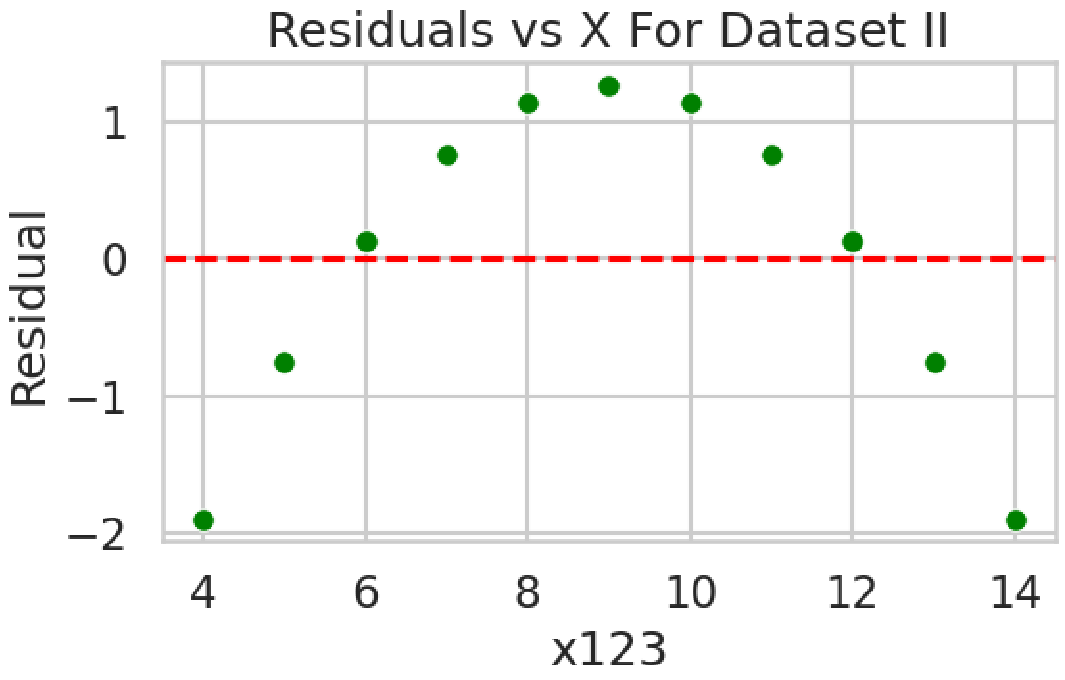

Anscombe Quartet III
y3
x123
12
10
8
6
4
6
8
10
12
14

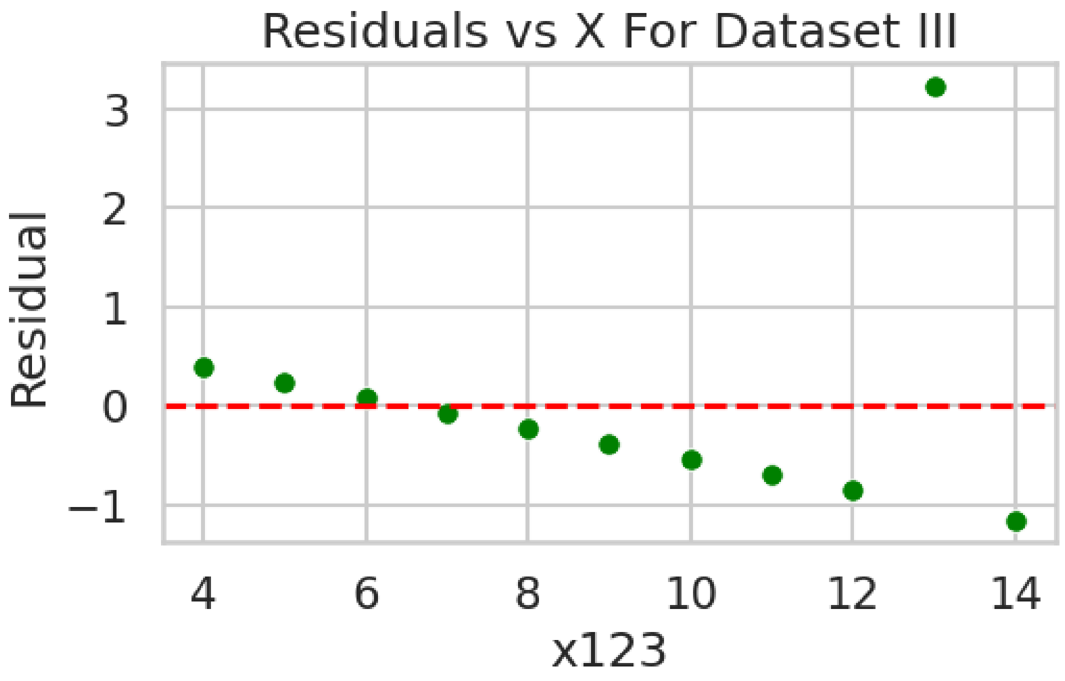


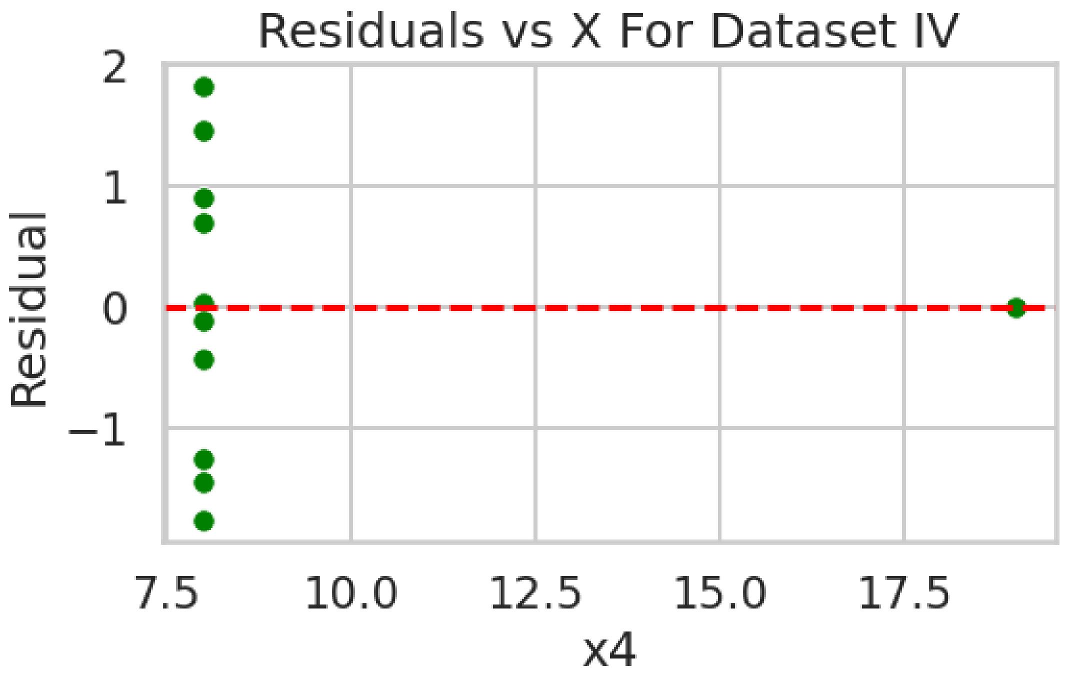
1.7.2 Residual Plots:

[14]:
```
for name, (x_col, y_col) in thedata.items():
    x = df[x_col]
    y = df[y_col]

    XD = sm.add_constant(x) #adds a constant line of reference
    residualplt = sm.OLS(y, XD).fit() # creates the line of regression, through␣
↪Ordinary Least Squares
    residualpredict = residualplt.predict(XD)

    residual = y - residualpredict # Calculates how far each point is from the␣
↪regression line

    plt.figure(figsize=(6,4))
    sns.scatterplot(x=x, y=residual, s=70, color="green")
    plt.axhline(0, color='red', linestyle='--')
    plt.title(f'Residuals vs X For Dataset {name}')
    plt.xlabel(x_col)
    plt.ylabel("Residual")
    plt.tight_layout() #formats the labels and titles
    plt.show()
    plt.close
```



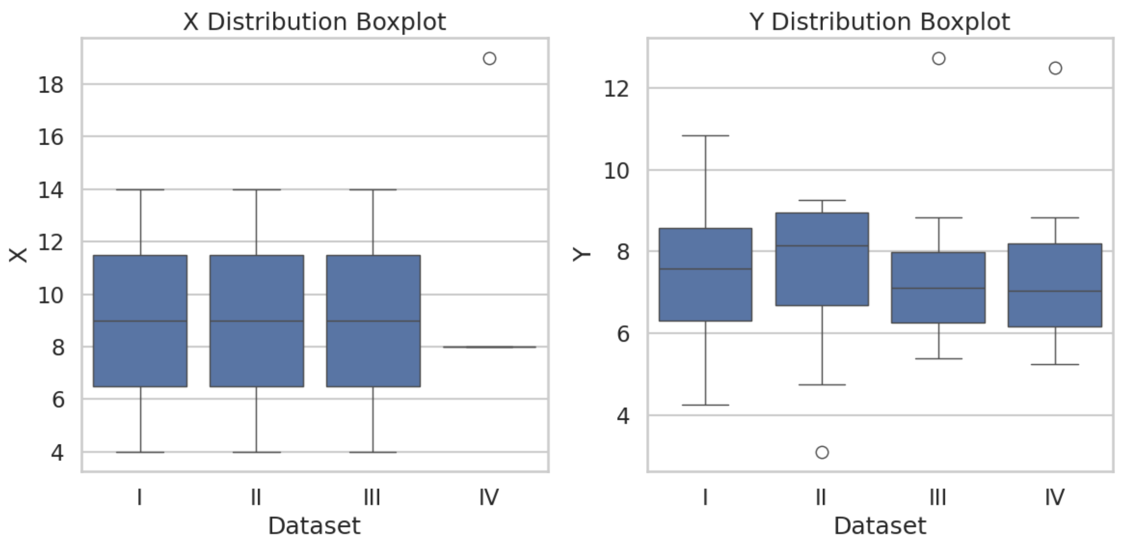



Residuals vs X For Dataset III
Residual
3
2
1
0
−1
4
6
8
10
12
14
x123


Residuals vs X For Dataset IV
Residual
2
1
0
−1
7.5
10.0
12.5
15.0
17.5
x4

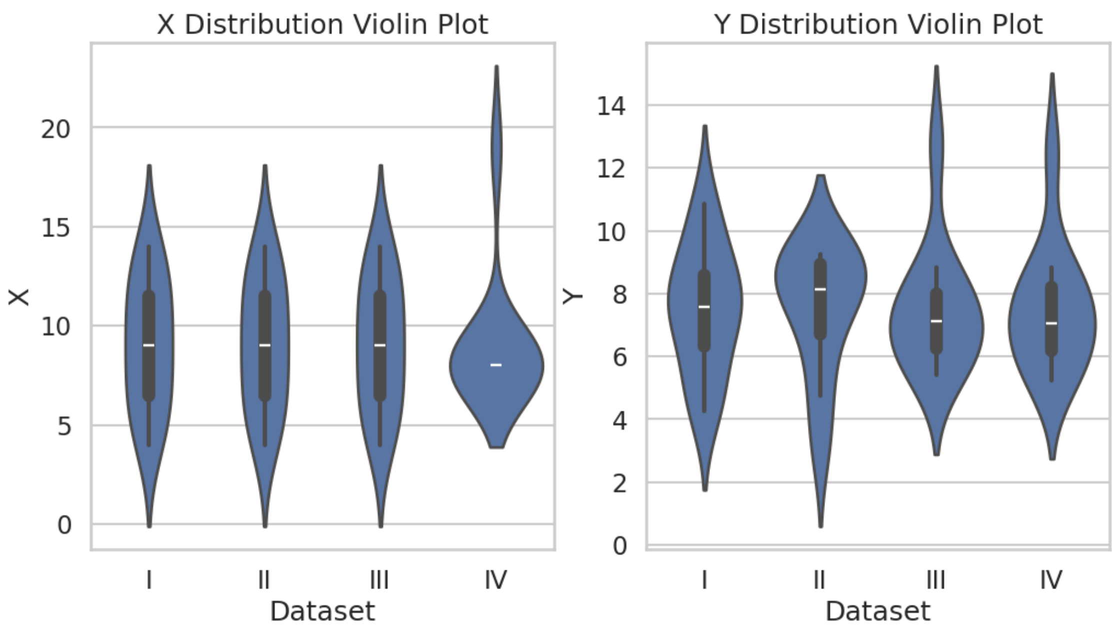
1.7.3 Box Plots:

[15]:
```python
bp_df = pd.DataFrame() #creates new dataframe to it easier for Seaborn to use

for name, (x_col, y_col) in thedata.items():
    bp_dataset = df[[x_col, y_col]].copy() #makes a copy of the data frame so␣
 ↪that the original dataframe isn't affected
    bp_dataset['dataset'] = name
    bp_dataset.rename(columns={x_col: 'x', y_col: 'y'}, inplace= True) #changes␣
 ↪the names to X and Y so that it is easier to plot the Box Plot, making sure␣
 ↪that it remains inside bp_dataset
    bp_df = pd.concat([bp_df, bp_dataset], ignore_index=True) #combines the␣
 ↪datasets made by bp_df and bp_dataset to use it as a master index and makes␣
 ↪it easier to extract the information

sns.set(style='whitegrid', context='talk')
fig, axes = plt.subplots(1, 2, figsize=(12,6))

sns.boxplot(x='dataset', y= 'x', data = bp_df, ax=axes[0]) #axes positions the␣
 ↪graphs so that they dont take up as much space next to each other
axes[0].set_title('X Distribution Boxplot')
axes[0].set_xlabel('Dataset')
axes[0].set_ylabel('X')

sns.boxplot(x='dataset', y='y', data = bp_df, ax=axes[1])# 0 is left, and 1 is␣
 ↪right for the axes
axes[1].set_title('Y Distribution Boxplot')
axes[1].set_xlabel('Dataset')
axes[1].set_ylabel('Y')

plt.tight_layout()
plt.show()
```



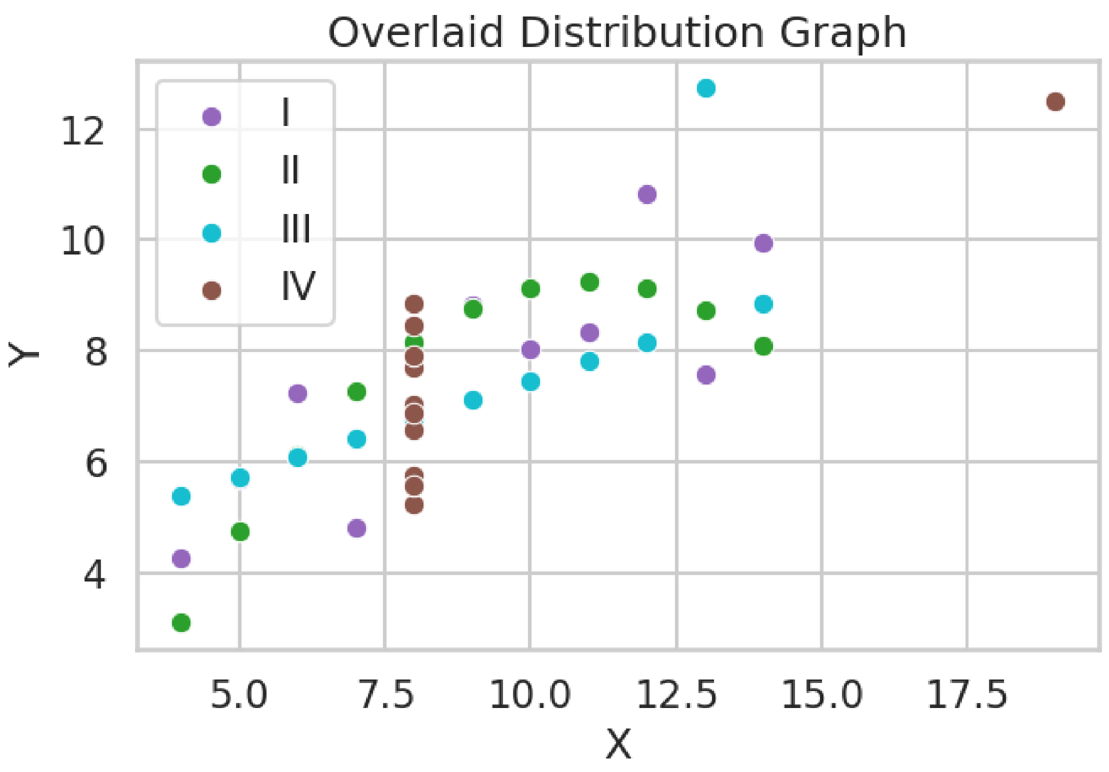
1.7.4 Violin Plots:

[17]:
```python
sns.set(style='whitegrid', context='talk')
fig, axes = plt.subplots(1, 2, figsize=(12,6))

sns.violinplot(x='dataset', y= 'x', data = bp_df, ax=axes[0])
axes[0].set_title('X Distribution Violin Plot')
axes[0].set_xlabel('Dataset')
axes[0].set_ylabel('X')

sns.violinplot(x='dataset', y='y', data = bp_df, ax=axes[1])
axes[1].set_title('Y Distribution Violin Plot')
axes[1].set_xlabel('Dataset')
axes[1].set_ylabel('Y')
```



1.7.5 Combined Scatter Plot:

[19]:
```
sns.set(style='whitegrid', context='talk')
graphcolors = ['tab:purple', 'tab:green', 'tab:cyan', 'tab:brown'] #assigns␣
↪colors to the different data sets so that it's easier to read
plt.figure(figsize=(7,5)) #resets the figure so that this graph doesn't␣
↪interfere with the violin plots

for i, (name, (x_col, y_col)) in enumerate(thedata.items()):
    sns.scatterplot(x=df[x_col], y=df[y_col], s=80, color=graphcolors[i],␣
↪label=name)
plt.title('Overlaid Distribution Graph')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.tight_layout()
plt.show()
```



1.8 6. Interpretation and Final Thoughts

Anscombe's Quartet is a great example of how summary statistics cannot always be relied on and how they may be holding out on information. Through the coding and graphing of these data points along the different graphs, we can see that they are very different. Though, when you look at the statistics summary given at the top, you can also see that the values are nearly identical, which could pose potential flaws and errors in the future if not addressed. In particular, many different parts can be noticed that the summary misses out on.

- Dataset II shows that it is a non-linear curve, which is quite different from Dataset I, which is fairly linear. Despite this, you can also see how the summary statistics fail to point this out.
- Dataset III has an outlier at (13, 12.74), which causes the regression line to grow steeper, which cannot be seen through the statistics either, even though the rest of it is extremely linear.
- Dataset IV mainly has an X value of 8; however, the statistics don't show this either.

These variations show how, even though the visuals are very different, the statistical summaries fail to show this. Relying purely on summarized portions of information like this is what could cause larger issues to proceed in the future, especially if this data is being used on a larger scale.

A real-life example of this is how statistics like this are used in getting census and income statistics for nations. When a developing nation has high-income outliers, the overall average and GDP of that nation would be higher than they actually are, which impacts income inequality and restricts funds from being donated to the specific country for aid. Statistics affect the lives of people in many small but meaningful ways, and having a truthful and complete understanding of these specific figures allows the right decisions to be made at the right time. Misinformation about important statistics could mean the difference between success and failure for many, as it did in the Challenger Disaster, where the right information wasn't presented or analyzed properly/accurately, leading to huge losses overall.

Through this report, the importance of truthful and proper data analysis came into perspective, as well as a crucial understanding of modern exploratory data analysis software. For future steps, it would be to apply this learning forward and understand what else we can do with it in fields like engineering, healthcare, and business, and how management of data and other visualization techniques could be used through modules like Seaborn and Matplotlib.