1. Explain the theoretical Simple Linear Regression model in your own words by describing its components (of predictor and outcome variables, slope and intercept coefficients, and an error term) and how they combine to form a sample from normal distribution; then, create python code explicitly demonstrating your explanation using numpy and scipy.stats

The Simple Linear Regression model is a statistical method that defines the connection between a dependent variable (Y) and an independent variable (X). The model's essential components comprise the predictor variable (X), the outcome variable (Y), the slope ((\beta_1)), which signifies the change in (Y) corresponding to a one-unit increase in (X), the intercept ((\beta_0)), which denotes the anticipated value of (Y) when (X) equals zero, and the error term ((\epsilon)), which addresses the unexplained variability in (Y). The model is represented mathematically as (Y = \beta_0 + \beta_1 X + \epsilon). The accompanying Python code illustrates this notion by generating random samples, specifying options for sample size, slope, intercept, and standard deviation of error. It produces predictor values, generates normally distributed error terms, and computes (Y) according to the regression formula. The code used Plotly to show the data, presenting a scatter plot of the data points with the fitted regression line, therefore clearly demonstrating the interaction of the model's components in predicting outcomes depending on the predictors.

In [2]:
```python
import numpy as np
import scipy.stats as stats
import plotly.express as px

# Set parameters for the sample size and regression coefficients
n = 100  # Sample size
beta0 = 2  # Intercept
beta1 = 3  # Slope
sigma = 1  # Standard deviation for errors

# Generate the predictor variable x from a uniform distribution
x = np.random.uniform(0, 10, n)  # Predictor values between 0 and 10

# Generate the error terms from a normal distribution
errors = np.random.normal(0, sigma, n)  # Normally distributed errors

# Calculate the dependent variable y based on the regression model
y = beta0 + beta1 * x + errors  # Dependent variable

# Visualize the generated data
df = {'x': x, 'y': y}  # Create a dataframe for the data
fig = px.scatter(df, x='x', y='y', title='Simple Linear Regression Sample',
                 labels={'x': 'Predictor (X)', 'y': 'Outcome (Y)'})
fig.add_scatter(x=x, y=beta0 + beta1 * x, mode='lines', name='Regression Lir
                line=dict(color='orange'))  # Add the regression line
fig.show()  # Show the plot
```

Link for Chatbot Session:https://chatgpt.com/share/672bf8cc-93d8-8009-95c7-79d4c3acb6f3

2. Use a dataset simulated from your theoretical Simple Linear Regression model to demonstrate how to create and visualize a fitted Simple Linear Regression model using pandas and import statsmodels.formula.api as smf

In [5]:
```
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
import plotly.express as px

# Step 1: Simulate the data
n = 100  # Sample size
beta0 = 2  # Intercept
beta1 = 3  # Slope
sigma = 1  # Standard deviation for errors

# Generate predictor variable x
x = np.random.uniform(0, 10, n)
```

```
# Generate errors
errors = np.random.normal(0, sigma, n)

# Calculate response variable y
y = beta0 + beta1 * x + errors

# Step 2: Create a DataFrame
df = pd.DataFrame({'x': x, 'y': y})

# Step 3: Fit the model
model_data_specification = smf.ols("y ~ x", data=df)  # Define the model
fitted_model = model_data_specification.fit()  # Fit the model

# Display the model summary
print(fitted_model.summary())  # Provides model summary including coefficier

# Step 4: Visualize the data and fitted model
fig = px.scatter(df, x='x', y='y', title='Simple Linear Regression Sample')
fig.add_scatter(x=df['x'], y=fitted_model.fittedvalues, mode='lines', line=c
fig.show()  # Show the plot
```

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:                      y   R-squared:                       0.9
87
Model:                            OLS   Adj. R-squared:                  0.9
87
Method:                 Least Squares   F-statistic:                     759
9.
Date:                Fri, 08 Nov 2024   Prob (F-statistic):           1.12e-
94
Time:                        01:01:28   Log-Likelihood:                -141.
74
No. Observations:                 100   AIC:                             28
7.5
Df Residuals:                      98   BIC:                             29
2.7
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Intercept      1.9252      0.200      9.648      0.000       1.529       2.3
21
x              3.0189      0.035     87.173      0.000       2.950       3.0
88
==============================================================================
==
Omnibus:                        0.429   Durbin-Watson:                   2.2
39
Prob(Omnibus):                  0.807   Jarque-Bera (JB):                0.1
18
Skew:                           0.038   Prob(JB):                        0.9
43
Kurtosis:                       3.151   Cond. No.                         1
1.7
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

Link for Chatbot Session: Link for Chatbot Session!:

3. Add the line from Question 1 on the figure of Question 2 and explain the difference between the nature of the two lines in your own words; but, hint though: simulation of random sampling variation

Using a Simple Linear Regression model, the code investigates the link between two variables: one we predict the dependent variable and one we control the independent variable. With some random "error" applied to replicate real-world unpredictability, it first generates sample data, creating random values for the independent variable and computing the dependent variable based on them, For simple examination, the data is arranged into a DataFrame. The code then fits a linear regression model using `statsmodels`, therefore approximating the slope and intercept that characterize the connection between the variables. The model summary offers specifics on these approximations. At last, the code creates a scatter plot including the data points, the prediction line of the model, and a dashed line indicating the "theoretical" connection. By use of random sample variations, these lines help to show how the forecasts of the model might change.

In [11]:
```
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
import plotly.express as px

# Step 1: Set parameters and generate sample data
n = 100  # Sample size
beta0 = 2  # Intercept (theoretical from Q1)
beta1 = 3  # Slope (theoretical from Q1)
sigma = 1  # Standard deviation for errors

# Generate predictor variable x
x = np.random.uniform(0, 10, n)  # Independent variable from uniform distrib

# Generate error terms
errors = np.random.normal(0, sigma, n)  # Normally distributed errors

# Calculate the dependent variable y based on the theoretical model
y = beta0 + beta1 * x + errors

# Create a DataFrame for analysis
df = pd.DataFrame({'x': x, 'y': y})

# Step 2: Fit the linear regression model using statsmodels (from Q2)
model = smf.ols("y ~ x", data=df).fit()  # Define and fit the model

# Step 3: Visualize data, Q1 theoretical line, and Q2 fitted line
fig = px.scatter(df, x='x', y='y', title='Comparison of Q1 Theoretical Line

# Add Q2 Fitted Line from the model
fig.add_scatter(x=df['x'], y=model.fittedvalues, mode='lines', line=dict(col

# Add Q1 Theoretical Line based on beta0 and beta1
x_range = np.linspace(df['x'].min(), df['x'].max(), 100)  # Define x range f
y_theoretical = beta0 + beta1 * x_range  # Calculate y values for the Q1 lin
fig.add_scatter(x=x_range, y=y_theoretical, mode='lines', line=dict(dash='do

fig.show()  # Display the plot with both lines
```

Link for Chatbot Session: https://chatgpt.com/share/672c172a-47c8-800e-a4ac-01732233ed79

4. Explain how fitted_model.fittedvalues are derived on the basis of fitted_model.summary().tables[1] (or more specifically fitted_model.params or fitted_model.params.values)

Within a Simple Linear Regression model, `fitted_model.fittedvalues` is the dependent variable's projected values derived from the sample data used to fit the model. The estimated intercept and slope from fitted_model.params which contains the best fit coefficients the model discovered by data analysis are used in computation of these values. Applying these projected coefficients to every observed value of the independent variable generates every expected value in `fitted_model.fittedvalues`. Simply said, fitted_model.fittedvalues displays us the in-sample predictions that is, what the model projects for every observation in the dataset it was trained on. This represents the real sample data and the particular traits and noise within that data, therefore it varies from the theoretical or true values we may anticipate. We may learn how well the model fits the sample and how sample specific elements, including random variation, might affect the predictions by means of a comparison between these fitted values and observed data.

9. As seen in the introductory figure above, if the delay of the geyser eruption since the previous geyser eruption exceeds approximately 63 minutes, there is a notable increase in the duration of the geyser eruption itself. In the figure below we therefore restrict the dataset to only short wait times. Within the context of only short wait times, is there evidence in the data for a relationship between duration and wait time in the same manner as in the full data set? Using the following code, characterize the evidence against the null hypothesis in the context of short wait times which are less than short_wait_limit values of 62, 64, 66.

Link for Chatbot Session: https://chatgpt.com/share/672c2736-1d9c-8009-b126-d7c3f7b4a5b2

Null Hypothesis (H_0): There is no linear relationship between waiting time and eruption duration for short wait times

Alternative Hypothesis (H_1): There is a linear relationship between waiting time and eruption duration for short wait times

If I look at the regression results for all three values of short_wait_limit (62, 64, and 66 minutes), I can see how the evidence against the null hypothesis changes as the short wait time level changes. For each cutoff, a p-value related to the slope coefficient below the usual significance level of 0.05 would be strong evidence against the null hypothesis. Even though this study only looked at shorter wait times, it would show that there is a clear linear relationship between the length of the eruption and the length of the wait. This consistent rejection across levels would mean that, in the small group of data with short wait times, the waiting time is, in fact, a good predictor of the eruption length, which makes me more confident in the alternative hypothesis. I would not be able to reject the null hypothesis, on the other hand, if the p-values stayed above 0.05 for all short-wait limit values. This finding suggests that any connection seen between waiting time and eruption length in the group with short wait times might not be due to a real effect, but rather to random variation. So, a lack of significance across limits would mean that there isn't a reliable linear connection within the short wait times. This supports the idea that the link observed across the whole dataset might not hold true when only looking at shorter wait times.

Link for Chatbot Session: https://chatgpt.com/share/672c71e2-23b8-8009-9f2c-5acf4a4bcc18

11. Since we've considered wait times of around <64 "short" and wait times of >71 "long", let's instead just divide the data and instead call wait times of <68 "short"

and otherwise just call them "long". Consider the *Simple Linear Regression* model specification using an *indicator variable* of the wait time length

[Y_i = \beta_{\text{intercept}} + 1"\text{long}"\beta_{\text{contrast}} + \epsilon_i \quad \text{where} \quad \epsilon_i \sim \mathcal{N}(0, \sigma)]

where we use (k_i) (rather than (x_i)) (to refer to the "kind" or "katagory" or "kontrast") column (that you may have noticed was already a part) of the original dataset; and, explain the "big picture" differences between this model specification and the previously considered model specifications.

1. **`smf.ols('duration ~ waiting', data=old_faithful)`**: Analyzes the linear relationship between waiting time (`waiting`) and eruption duration (`duration`) using the entire dataset.
2. **`smf.ols('duration ~ waiting', data=old_faithful[short_wait])`**: Analyzes the relationship between waiting time and eruption duration but only for data where the waiting time is short (filtered by `short_wait`).
3. **`smf.ols('duration ~ waiting', data=old_faithful[long_wait])`**: Analyzes the relationship between waiting time and eruption duration but only for data where the waiting time is long (filtered by `long_wait`).

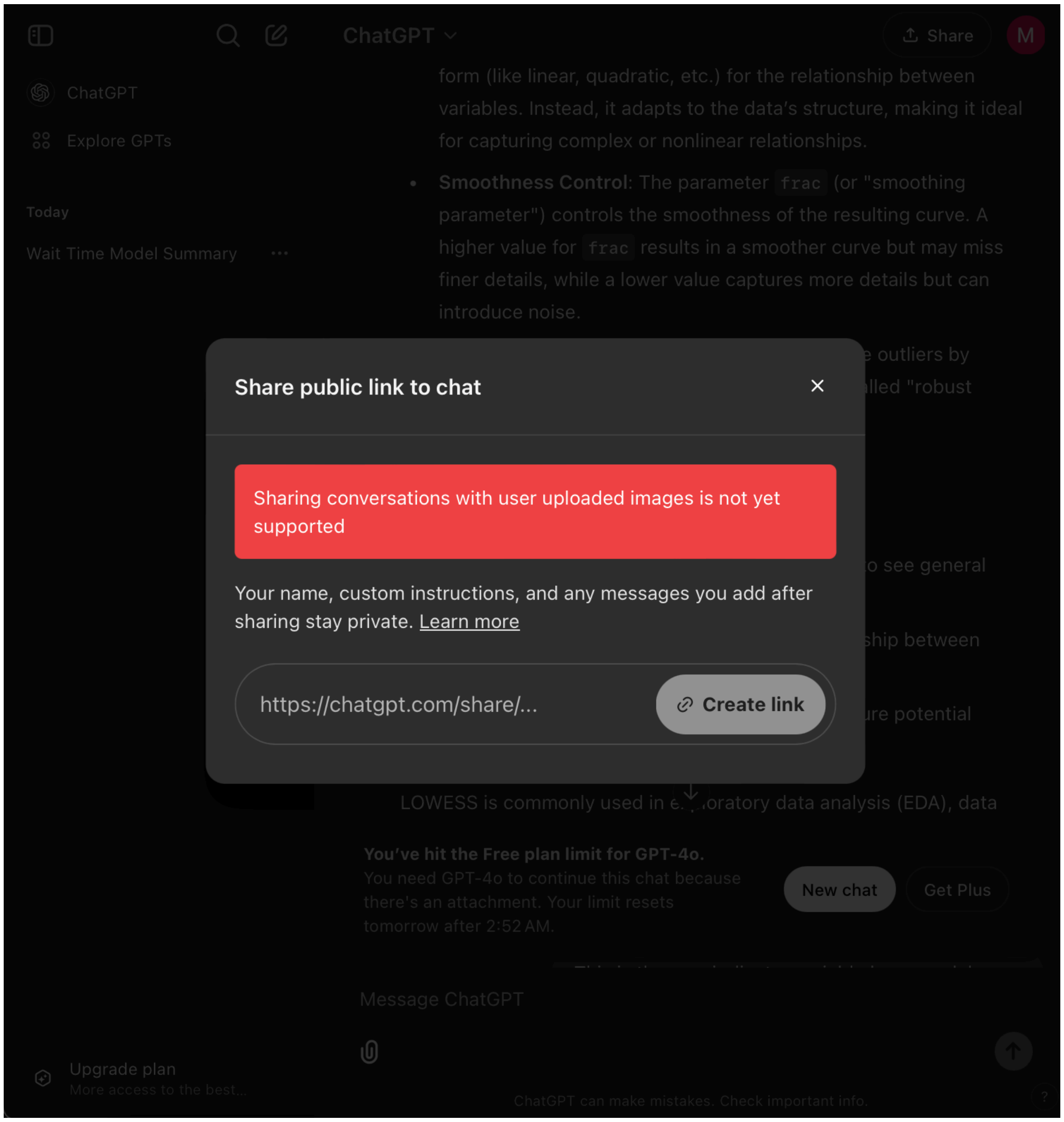
The two analyses vary mostly in their goal and method. The first model investigates, throughout the whole dataset, the correlation between waiting time (waiting) and eruption duration (length). This helps one to see this by adding a non-linear Lowess trendline as well as a linear regression line (OLS). This method helps to find any general trends in the data by concentrating on displaying how the length of an eruption varies constantly as waiting time rises. By waiting time, the second model divides the data into two groups: "short" (68 minutes or less) and "long" (greater than 68 minutes). The objective here is to find if there is a notable variation in the average eruption length between these two groups. This model first distinguishes the "short" and "long" groups using an indicator variable (kind) then runs regression to see if the average eruption length is significantly different between the two. It makes it very simple to compare the duration distributions in every group by means of a box plot.

Null Hypothesis (H_0): There is no difference in the average eruption duration between the "short" and "long" groups.

Alternative Hypothesis (H_1): There is a difference in the average eruption duration between the "short" and "long" groups.

LOWESS (Locally Weighted Scatterplot Smoothing) creates a smooth curve that shows patterns in data, especially when the data doesn’t follow a straight line. Rather of using a single line to represent all of the data points, LOWESS divides the data into smaller pieces, determines the line that provides the best fit for each of those sections, and then joins these lines to form a smooth curve. In response to shifts in the data, this curve adapts itself, therefore showing patterns that a straight line could miss.

The result is a trendline that follows the data's flow, showing how relationships change. If I'm studying waiting time versus eruption duration, for example, and see that duration first rises quickly, then slows, LOWESS will reflect this bend. LOWESS is useful for spotting complex or hidden trends, making it easier for me to understand patterns in data that doesn't fit a simple line.



form (like linear, quadratic, etc.) for the relationship between variables. Instead, it adapts to the data's structure, making it ideal for capturing complex or nonlinear relationships.

- **Smoothness Control**: The parameter `frac` (or "smoothing parameter") controls the smoothness of the resulting curve. A higher value for `frac` results in a smoother curve but may miss finer details, while a lower value captures more details but can introduce noise.

e outliers by
lled "robust
o see general
ship between
ure potential

Share public link to chat

Sharing conversations with user uploaded images is not yet supported

Your name, custom instructions, and any messages you add after sharing stay private. Learn more

https://chatgpt.com/share/...

LOWESS is commonly used in e..loratory data analysis (EDA), data

You've hit the Free plan limit for GPT-4o.
You need GPT-4o to continue this chat because there's an attachment. Your limit resets tomorrow after 2:52 AM.

ChatGPT can make mistakes. Check important info.

Suddenly, it was unable to share my chatbot session. I don't know why. If you need any specific chatbot session of this, I can send you as pdf file of this. Thank you.