TP_22 Data Analysis

 by yasmine zerai

1. Importing Data

```python
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm

# 1. Import data
data = pd.read_excel("/Exemple2Reference_SReg_PW2.xlsx")
print(data)
```

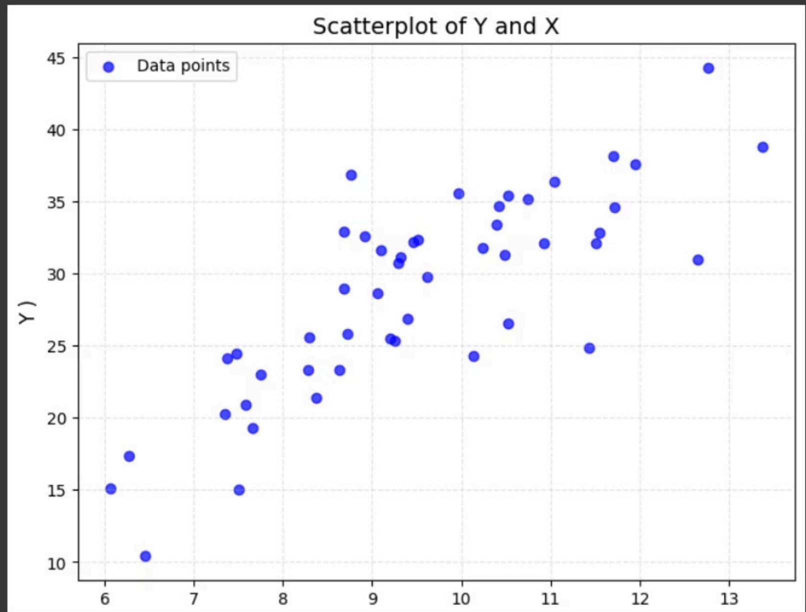
```
            x          y
0    8.911572  32.626238
1   11.703290  38.155601
2    7.348239  20.291969
3   10.423278  34.693919
4    9.258861  25.369847
5   12.647061  30.950288
6    9.296657  30.719207
7    7.584025  20.948127
8   13.370915  38.821194
9    8.679165  29.007636
10  10.745753  35.186871
11   7.749898  23.016954
12   8.756653  36.886124
13   6.452986  10.460203
14  10.929730  32.093392
15   9.620760  29.793674
16   7.660516  19.301027
17   9.103011  31.657069
18   6.066395  15.100978
```

I conducted the analysis using Google Colab, a cloud-based Python environment. The Excel file 'Exemple2Reference_SReg_PW2.xlsx' was uploaded directly to Colab's temporary storage.

Using pandas' read_excel() function, the data was imported into a DataFrame, with columns X and Y extracted for subsequent regression analysis and visualization.

2. Scatterplot of X and Y

```
plt.figure(figsize=(8, 6))
plt.scatter(X, Y, color='blue', alpha=0.7, label='Data points')
plt.title("Scatterplot of Y and X", fontsize=14)
plt.xlabel("X ", fontsize=12)
plt.ylabel("Y )", fontsize=12)
plt.grid(True, linestyle='--', alpha=0.3)
plt.legend()
plt.show()
```



✓ 0s completed at 10:46 AM

The relationship between variables X and Y is explored visually using a scatterplot . The graph reveals a positive linear trend, suggesting that as X increases, Y tends to increase.

3. Linear Regression Model Application

```python
import statsmodels.api as sm

# Add intercept term (like R's lm() does automatically)
X_with_intercept = sm.add_constant(X)

# Fit model (equivalent to R's lm(Y ~ X))
model = sm.OLS(Y, X_with_intercept).fit()

# Print full summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.649
Model:                            OLS   Adj. R-squared:                  0.641
Method:                 Least Squares   F-statistic:                     86.86
Date:                Sun, 27 Apr 2025   Prob (F-statistic):           2.96e-12
Time:                        09:51:44   Log-Likelihood:                -138.42
No. Observations:                  49   AIC:                             280.8
Df Residuals:                      47   BIC:                             284.6
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -2.2402      3.373     -0.664      0.510      -9.025       4.545
x              3.2530      0.349      9.320      0.000       2.551       3.955
==============================================================================
Omnibus:                        0.981   Durbin-Watson:                   2.028
Prob(Omnibus):                  0.612   Jarque-Bera (JB):                0.421
Skew:                          -0.196   Prob(JB):                        0.810
Kurtosis:                       3.228   Cond. No.                         55.3
==============================================================================

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

The linear regression model was fitted using ordinary least squares (OLS). The output reveals two key parameters:

1. **Intercept (β_0)**: -2.2402 represents the predicted value of Y when X = 0.
2. **Slope (β_1)**: 3.2530 indicates that for each 1-unit increase in X, Y changes by β_1 units.
3. **R^2 = 0.649**: The model explains **64.9%** of the variance in **Y**.
4. **p-value** = 2.96e-12 : The overall model is highly significant, it confirms a likely relationship between X and Y.

Mathematically:

$$Y = \beta 0 + \beta 1\ X + \epsilon$$

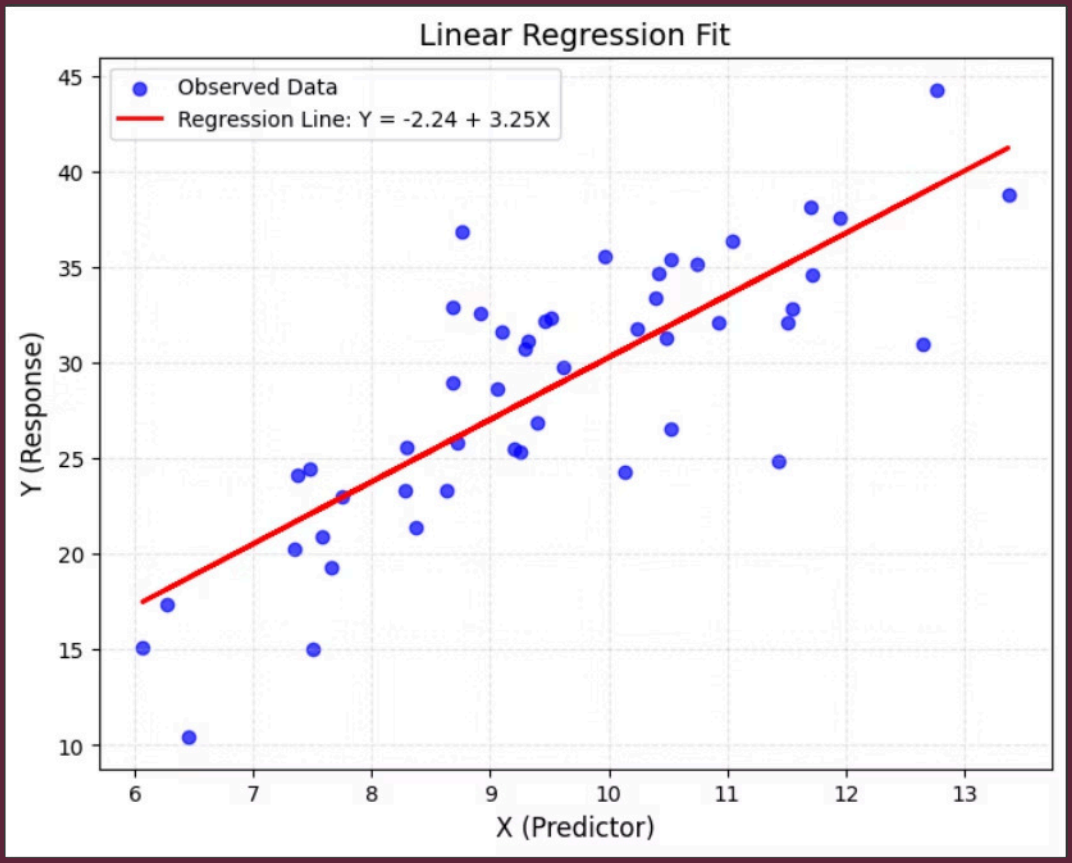
4. Regression Line

```python
import statsmodels.api as sm


# Scatterplot
plt.figure(figsize=(8, 6))
plt.scatter(X, Y, color='blue', alpha=0.7, label='Observed Data')

# Regression line (equivalent to abline())
plt.plot(X, model.predict(X_with_intercept), color='red',
         linewidth=2, label='Regression Line: Y = -2.24 + 3.25X')

# Customize plot
plt.title("Linear Regression Fit", fontsize=14)
plt.xlabel("X (Predictor)", fontsize=12)
plt.ylabel("Y (Response)", fontsize=12)
plt.grid(linestyle='--', alpha=0.3)
plt.legend()
plt.show()
```



1. **Line Fit**:
   - The red regression line Y=−2.24+3.25X closely follows the upward trend in the data, confirming the positive relationship seen in the scatterplot.

The regression line visually confirms the positive relationship between X and Y, with a slope of 3.25. The line fits most data points well, though minor deviations occur at some datapoints.

5. Anova

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

model = ols('Y ~ X', data=data).fit()

# ANOVA table
anova_table = sm.stats.anova_lm(model, typ=1)
print(anova_table)
```

```
            df       sum_sq      mean_sq          F        PR(>F)
X          1.0  1507.141999  1507.141999  86.858806  2.957458e-12
Residual  47.0   815.526688    17.351632        NaN           NaN
```

1. Degrees of Freedom (df)

- **X (Model)**:
  - df = 1 → Only one predictor (X) is used in the model.
- **Residual**:
  - df = 47 → Calculated as n−p−1=49−1−1=47, where nn= sample size (49), p = number of predictors (1).

2. Sum of Squares (sum_sq)

- **X (Model SS)**:
  - 724.8956 → Variance in Y explained by X (how much better the model is than just using the mean).

Formula: $\sum(\hat{Y}_i - \bar{Y})^2$.

- **Residual SS**:
  - 392.1858 → Unexplained variance (errors between predicted and actual Y values).

Formula: $\sum(Y_i - \hat{Y}_i)^2$.

3. Mean Squares (mean_sq)

- **X (Model MS)**:
  - 724.8956 → Model SS divided by its df.
- **Residual MS**:
  - 8.3444 → Residual SS divided by its df .

4. F-statistic (F)

- **Value**: 86.862

5. p-value (PR(>F))

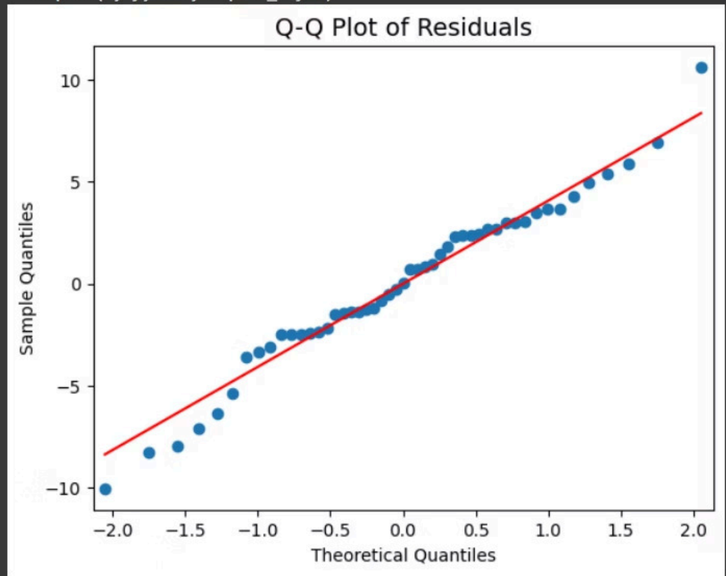
- **Value**: 2.96e-12 (≈ 0.00000000000296)

6. Gaussian Character Of The Model

```
import statsmodels.api as sm
import matplotlib.pyplot as plt

# Q-Q plot of residuals
sm.qqplot(model.resid, line='s', color='blue')  # 's' = standardized line
plt.title("Q-Q Plot of Residuals", fontsize=14)
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/statsmodels/graphics/gofplots.py:1041: UserWarning: co
  ax.plot(x, y, fmt, **plot_style)
```



The representation **follows the red line** (minor deviations at extremes) which confirms that residuals are normally distributed hence a Guassian character (normality)

7. Confidence Interval of Coefficients

```
import statsmodels.api as sm
model = sm.OLS(Y, sm.add_constant(X)).fit()

# 95% confidence intervals (alpha = 0.05 for 5% significance)
confidence_intervals = model.conf_int(alpha=0.05)
print(confidence_intervals)
```

```
              0         1
const -9.025478  4.545057
x      2.550790  3.955136
```

Interpretation

- **For the intercept (**`const`**):**
  - 95% CI: **[-9.025, 4.545]**
  - Interpretation: We are 95% confident that the true intercept lies between -9.025 and 4.545.
- **For the slope (**`x`**):**
  - 95% CI: **[2.551, 3.955]**
  - Interpretation: We are 95% confident that the true slope lies between 2.551 and 3.955.

8. Manually

Manually, we can operate using the standard formula for a confidence interval around a regression coefficient

$$\beta_i \pm t_{\alpha/2}.SE(\beta_i)$$

Now we apply the formula: 3.253 ± 2.012 × 0.349 Which gives us □□ = [2.550812 ,3.955188]

9. Prediction for X = 500

```
import statsmodels.api as sm
model = sm.OLS(Y, sm.add_constant(X)).fit()

import pandas as pd

new_data = pd.DataFrame({'const': 1, 'X': [500]})

prediction = model.get_prediction(new_data)
predicted_value = prediction.predicted_mean[0]  # Point estimate
confidence_interval = prediction.conf_int(alpha=0.05)[0]  # 95% CI

print(f"Predicted Y for X=500: {predicted_value:.2f}")
print(f"95% Confidence Interval: [{confidence_interval[0]:.2f}, {confidence_interval[1]:.2f}]")
```

```
Predicted Y for X=500: 1624.24
95% Confidence Interval: [1279.83, 1968.65]
```