Bridging Theory and Practice: A Practical Exploration of Simple Linear Regression in Python

From statistical assumptions to model diagnostics with statsmodels.

Hernández Pérez Erick Fernando

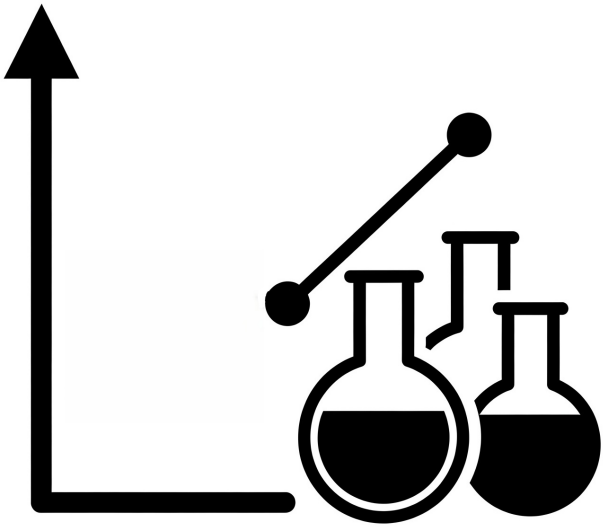

Figure 1

This article is inextricably linked to the theoretical groundwork laid out in 022. Unveiling the Statistical Foundations of Simple Linear Regression. That previous work focused on establishing the conceptual and mathematical foundations that support the linear regression model—its assumptions, derivations, and idealized properties. However, as is often the case in applied statistics, theory alone cannot carry the full burden of practical implementation.

There is a well-known adage that *"in theory, theory and practice are the same, but in practice, they are different."* This simple yet profound statement captures the essence of what we aim to explore in this piece: the transition from theoretical understanding to practical application. Theory outlines the ideal conditions under which a model performs optimally, but practice is where the real test lies—where data is noisy, assumptions are challenged, and the analytical process must be grounded in computational tools and critical evaluation.

With this perspective in mind, the purpose of this article is to revisit the essential elements of simple linear regression through a practical lens. Using Python's powerful `statsmodels` library, we will walk step-by-step through the process of building a regression model from real-world data. More importantly, we will explore how to interpret the output provided by the model and how to diagnose whether our assumptions—so carefully laid out in the preceding theoretical discussion—hold true in practice.

Specifically, we will review how to test the core assumptions of a simple linear regression model:

- **Exogeneity:** The errors should have a conditional mean of zero given the independent variable(s).
- **Homoscedasticity:** The variance of the errors should remain constant across all levels of the independent variable.
- **No autocorrelation:** The residuals should not be correlated with one another.
- **Normality of residuals:** For inference and hypothesis testing, it is assumed that the residuals are normally distributed.

Our Data

For this study, we use the publicly available dataset titled Salary Dataset - Simple Linear Regression, created by Allena Venkata Sai Abhishek and hosted on Kaggle (Sai, n.d.)[4]. This dataset has gained popularity in educational contexts and has been featured in the Machine Learning A to Z course series. It was last updated in 2023, ensuring its relevance for contemporary demonstrations.

The dataset contains two variables of interest: YearsExperience, representing the number of years of professional experience, and Salary, representing the corresponding annual salary in USD. In our analysis, Salary will serve as the dependent variable, while YearsExperience will act as the explanatory (independent) variable. This simple structure makes it ideal for illustrating the key concepts and assumptions of simple linear regression in a practical and intuitive way.

Our data

```
import kagglehub
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('seaborn-v0_8-darkgrid')

# Download latest version
path =
↪  kagglehub.dataset_download("abhishek14398/salary-dataset-simple-linear-regression")

print("Path to dataset files:", path)

df = pd.read_csv(path + '/Salary_dataset.csv', index_col=0)
df.describe()
```

Statistic	YearsExperience	Salary
Count	30.000000	30.000000
Mean	5.413333	76004.000000
Std	2.837888	27414.429785
Min	1.200000	37732.000000
25%	3.300000	56721.750000
50%	4.800000	65238.000000
75%	7.800000	100545.750000
Max	10.600000	122392.000000

Table 1: Descriptive statistics for the variables *YearsExperience* and *Salary*.

Our data

```
sns.scatterplot(df,x='YearsExperience', y='Salary')
```

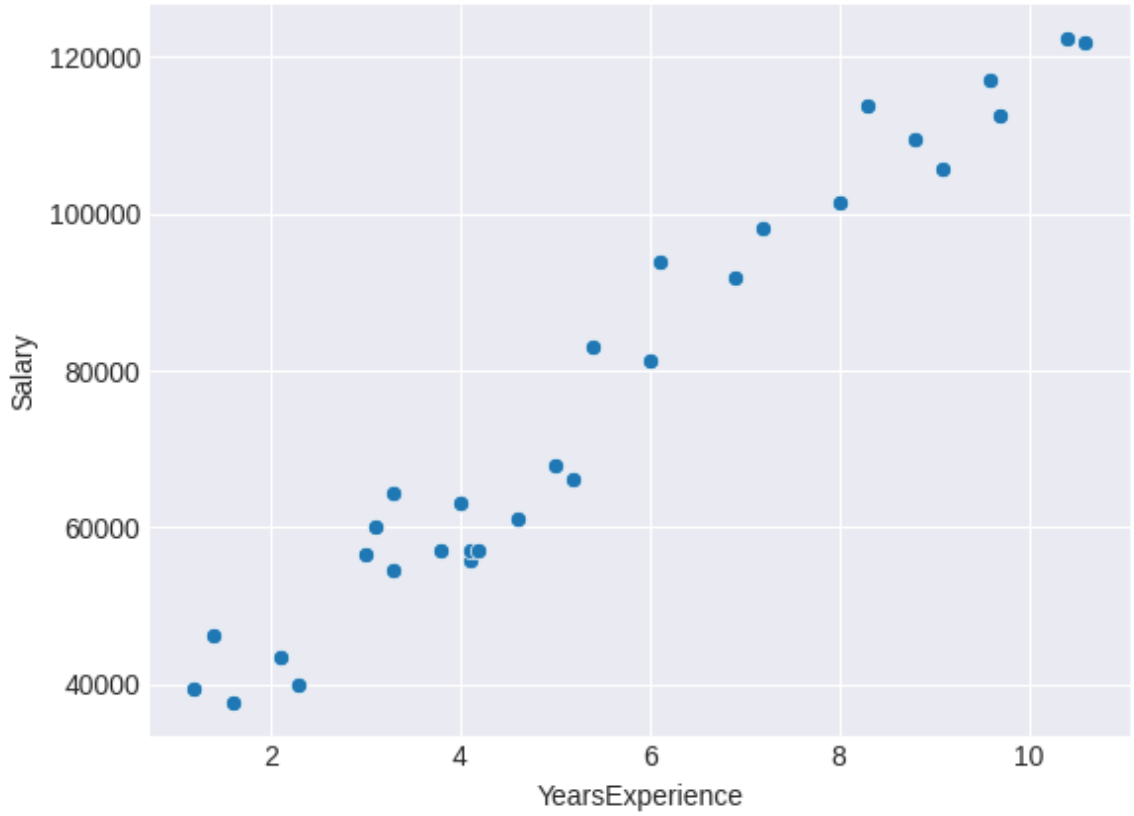

Figure 2

The Model

In the Python ecosystem, the statsmodels library stands out as a powerful tool for statistical modeling. It offers a comprehensive suite of methods and tests that allow researchers and data analysts to perform rigorous statistical analyses with ease. Unlike other machine learning libraries that prioritize prediction accuracy, statsmodels focuses on providing detailed statistical output, making it ideal for inferential tasks where understanding the relationships between variables is crucial.

Among the many features of statsmodels, one of the most fundamental is the implementation of Ordinary Least Squares (OLS) regression. OLS is a classic method used to model the linear relationship between a dependent variable and one or more explanatory variables. It estimates the coefficients by minimizing the sum of squared residuals — the differences between the observed and predicted values. This method not only allows us to construct a predictive model, but also provides rich statistical insight, including coefficient significance, goodness-of-fit metrics, and diagnostic information.

In the following code, we will demonstrate how to perform a simple linear regression using statsmodels by modeling Salary as a function of YearsExperience. This process includes preparing the data, fitting the OLS model, and printing a summary of the regression results:

The model

```
import statsmodels.api as sm
import matplotlib.pyplot as plt
import scipy.stats as stats
from statsmodels.stats.stattools import jarque_bera, durbin_watson
from statsmodels.stats.diagnostic import het_breuschpagan

# Prepare the data
```

```
X = sm.add_constant(df['YearsExperience'])
Y = df['Salary']

# Fit the OLS model
model = sm.OLS(Y, X).fit()
print(model.summary())
```

Table 2: OLS Regression Results

Dependent variable:	Salary
Model:	OLS (Ordinary Least Squares)
Method:	Least Squares
Date:	Tue, Apr 22 2025
Time:	22:11:27
Number of observations:	30
Degrees of freedom (model):	1
Degrees of freedom (residuals):	28
F-statistic:	622.5
Prob(F-statistic):	1.14×10^{-20}
R-squared:	0.957
Adjusted R-squared:	0.955
Log-likelihood:	-301.44
AIC:	606.9
BIC:	609.7

Variable	**Coef.**	**Std. Err.**	**t**	**P > |t|**	**[0.025, 0.975]**
const	24850	2306.654	10.772	0.000	[20100, 29600]
YearsExperience	9449.9623	378.755	24.950	0.000	[8674.119, 10200]

Omnibus:	2.140
Prob(Omnibus):	0.343
Jarque-Bera (JB):	1.569
Prob(JB):	0.456
Skew:	0.363
Kurtosis:	2.147
Durbin-Watson:	1.648
Cond. No.:	13.6

Assessing the model's general performance

Before moving forward with additional analyses, let us first break down the information provided by the summary output. As can be seen, it consists of three main sections. The upper section refers to the overall model and helps us answer key questions such as: Is the model statistically significant? (F-statistic), How much of the variability in the dependent variable does it explain? (R-squared), and How is the inclusion of additional variables penalized? (Adjusted R-squared) — which, in this case, remains unaffected since we only have one explanatory variable.

In addition, we see metrics like the Log-likelihood, AIC, and BIC, which offer deeper insights into model fit. The Log-likelihood measures how likely it is to observe the data given the model's parameters — the higher, the better. However, on its own, it isn't ideal for model comparison. That's where AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) come in. Both penalize model complexity to prevent overfitting. While AIC emphasizes predictive accuracy, BIC imposes a stricter penalty for models with more parameters, favoring simpler models. Lower values of AIC and BIC generally indicate a better-fitting model.

When evaluating the general performance of the model, the results are remarkably strong for a simple linear regression. The **F-statistic** is **622.5**, accompanied by a **p-value of** 1.14×10^{-20}, which is far below any

conventional significance level (e.g., 0.05). This indicates that the model, as a whole, is statistically significant — in other words, *YearsExperience* is a meaningful predictor of *Salary*.

The **R-squared** value of **0.957** suggests that approximately **95.7% of the variability in salary can be explained by years of experience**. This is exceptionally high for real-world data, particularly in a univariate model. The **Adjusted R-squared**, which accounts for the number of predictors, is also high at **0.955**, and, as expected, remains very close to the unadjusted R-squared due to the presence of a single independent variable.

Regarding model fit metrics, the **log-likelihood** is **-301.44**. While this value by itself doesn't carry direct interpretation, it is essential for computing other comparative criteria such as AIC and BIC. The **Akaike Information Criterion (AIC)** is **606.9**, and the **Bayesian Information Criterion (BIC)** is **609.7**. Both values are useful when comparing competing models — typically, *the lower, the better*. Although we are not comparing alternative models here, these values give a benchmark for future extensions (e.g., adding more variables or testing non-linear specifications).

Overall, these statistics suggest an exceptionally good fit for a simple model, and they provide a solid foundation for further diagnostic testing and interpretation.

Coefficient estimates and significance

The second part of the summary refers to the coefficients estimated using the OLS method: the intercept (the constant term, denoted as $\hat{\beta}_1$) and the slope for *YearsExperience* (denoted as $\hat{\beta}_2$). For each of these coefficients, a t-test is performed to assess whether they are statistically significant.

In particular, the null hypothesis in each case is that the coefficient is equal to zero — meaning it has no effect on the dependent variable. We look for **p-values** smaller than our chosen significance level (in this article, we use **0.05**) to reject the null hypothesis. When the p-value is below this threshold, we conclude that the corresponding variable makes a meaningful contribution to the model.

If you wish to embrace the full rigor of statistical verbiage, then we might say: *"There is statistically significant evidence indicating that the regression coefficients are meaningful contributors to the model."* Of course, that's just the polished way of saying — yes, the model seems to be doing a good job!

- **Intercept (const = 24,850):** Represents the predicted salary when YearsExperience is 0. Though not always realistic, it is necessary for defining the regression line.
- **YearsExperience Coefficient ($\approx 9,450$):** Indicates that each additional year of experience increases the predicted salary by approximately \$9,450, suggesting a strong positive linear relationship.
- **Statistical Significance:** Both coefficients have very low p-values (< 0.0001), implying strong evidence that they are different from zero and statistically significant at the 5% level.
- **Confidence Intervals (95%):**
  - Intercept: [20,100, 29,600]
  - YearsExperience: [8,674.12, 10,200]

  These intervals do not include zero, supporting the significance and reliability of the estimates.

Model diagnostics and residual analysis

The last section of the OLS summary provides diagnostic statistics that help evaluate how well the model assumptions are met:

- **Omnibus and Jarque-Bera (JB) Tests:** These are tests for normality of the residuals. The p-values (0.343 and 0.456 respectively) are both greater than 0.05, suggesting no strong evidence to reject the null hypothesis of normality.
- **Skew (0.363):** Measures asymmetry in the distribution of residuals. A value close to zero indicates near symmetry.
- **Kurtosis (2.147):** Indicates the "tailedness" of the residual distribution. A value close to 3 is considered normal; here, the value suggests a slightly platykurtic distribution (flatter than normal).
- **Durbin-Watson (1.648):** Tests for autocorrelation in residuals.
- **Condition Number (13.6):** Measures multicollinearity and numerical stability of the regression. Values below 30 are generally considered safe, so multicollinearity is not a concern here and because we only use YearsExperience.

Detecting autocorrelation in residuals

The Durbin-Watson statistic tests for the presence of autocorrelation in the residuals from a regression analysis. Specifically, it detects whether residuals from one observation are correlated with those from the next. Values close to 2 suggest no autocorrelation; values substantially below 2 indicate positive autocorrelation, while values above 2 suggest negative autocorrelation. It is based on the hypothesis that the errors of the regression model are generated by a first-order autoregressive process, observed at equally spaced time intervals (Montgomery et al., 2006)[3].

$$d = \frac{\sum_{t=2}^{n}(e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2}$$

If the Durbin-Watson statistic is below 1.0, it could signal a potential issue, suggesting that the error terms are positively linked (as a rule of thumb). On the other hand, values greater than 2 indicate negative correlation between consecutive errors. This may lead to inaccurate conclusions about the statistical significance in regression analysis.

To assess positive autocorrelation at significance level α, the test statistic d is compared with the lower and upper critical values ($d_{L,\alpha}$ and $d_{U,\alpha}$):

- If $d < d_{L,\alpha}$, there is statistical evidence indicating that the error terms are positively autocorrelated.
- If $d > d_{U,\alpha}$, there is no statistical evidence suggesting positive autocorrelation of the error terms.
- If $d_{L,\alpha} < d < d_{U,\alpha}$, the result is inconclusive.

Positive serial correlation occurs when a positive error for one observation increases the likelihood of a positive error for another observation.

To check for negative autocorrelation at significance level α, the test statistic $(4 - d)$ is compared with the lower and upper critical values ($d_{L,\alpha}$ and $d_{U,\alpha}$):

- If $(4-d) < d_{L,\alpha}$, there is statistical evidence indicating that the error terms are negatively autocorrelated.
- If $(4 - d) > d_{U,\alpha}$, there is no statistical evidence suggesting negative autocorrelation of the error terms.
- If $d_{L,\alpha} < (4 - d) < d_{U,\alpha}$, the result is inconclusive.

To obtain the critical values for the Durbin-Watson statistic, specialized tables are used. These tables provide the lower and upper critical values for different significance levels and degrees of freedom. In this case, the critical values used were obtained from the tables available at Durbin-Watson Tables. These values are essential for interpreting the Durbin-Watson statistic in terms of positive and negative autocorrelation in the residuals of a regression model.

Durbin-Watson

```
dw = durbin_watson(model.resid)
print("Durbin-Watson:", dw) # same as summary
```

$$\begin{aligned} d &= 1.648 \\ d_{L,0.05} &= 1.352 \\ d_{U,0.05} &= 1.489 \\ (4 - d) &= 2.352 \end{aligned}$$

Therefore, there are no positive and negative autocorrelation. (✓)

Testing for exogeneity

In regression analysis, testing for exogeneity is essential to validate that the independent variables are not correlated with the error term. If the predictor is exogenous, it ensures that the estimation of coefficients remains unbiased and consistent.

To test for exogeneity in a simple linear regression, we can check the correlation between the predictor variable and the residuals. The following Python code performs this check:

Exogeneity

```
predictor = df['YearsExperience']
correlation, p_value = stats.pearsonr(predictor, residuals)
print(f"Correlation: {correlation:.4f}, p-value: {p_value:.4f}")
```

This code computes the Pearson correlation coefficient between `YearsExperience` (the independent variable) and the residuals of the regression model. The `pearsonr` function from the `scipy.stats` library returns both the correlation value and its associated p-value.

- A correlation close to zero and a high p-value (greater than 0.05) suggest no significant linear relationship between the predictor and the residuals, which supports the assumption of exogeneity.
- A significant correlation (low p-value) would imply endogeneity, indicating that the predictor is not independent of the error term, potentially biasing the regression estimates.

```
Correlation:  0.0000, p-value:  1.0000
```

Therefore, there is exogeneity. (✓)

Testing for homoscedasticity

Homoscedasticity refers to the assumption that the variance of the residuals is constant across all levels of the independent variable. In other words, the spread of the residuals should remain roughly the same regardless of the predictor values. This assumption is fundamental for the validity of many regression-based inferential statistics.

To assess homoscedasticity, we can begin with a visual inspection by plotting the residuals against the fitted values. If the residuals appear randomly scattered with no discernible pattern or funnel shape, this suggests that the assumption is met.

Visual inspection

```
residuals = model.resid
predictions = model.fittedvalues

plt.scatter(predictions, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Residuals vs. Predicted Values')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.show()
```

As a more formal approach, we can use the **Breusch-Pagan test**, which evaluates whether the variance of the residuals depends on the independent variables. A high p-value in this test suggests that the residuals exhibit constant variance (homoscedasticity), while a low p-value indicates heteroscedasticity, i.e., a violation of the assumption.

$$\hat{\epsilon}^2 = \gamma_0 + \gamma_1 x + \varepsilon$$

Bresuch-Pagan

```
bp_test = het_breuschpagan(residuals, model.model.exog)
bp_stat, bp_pvalue, _, _ = bp_test
print(f"Breusch-Pagan: Statistic={bp_stat}, p-value={bp_pvalue}")
```

```
Breusch-Pagan:  Statistic=0.3990532776650091, p-value=0.5275785890839393
```

Therefore, there is homoscedasticity. (✓)

Both the graphical and statistical approaches should be used complementarily to draw more robust conclusions.

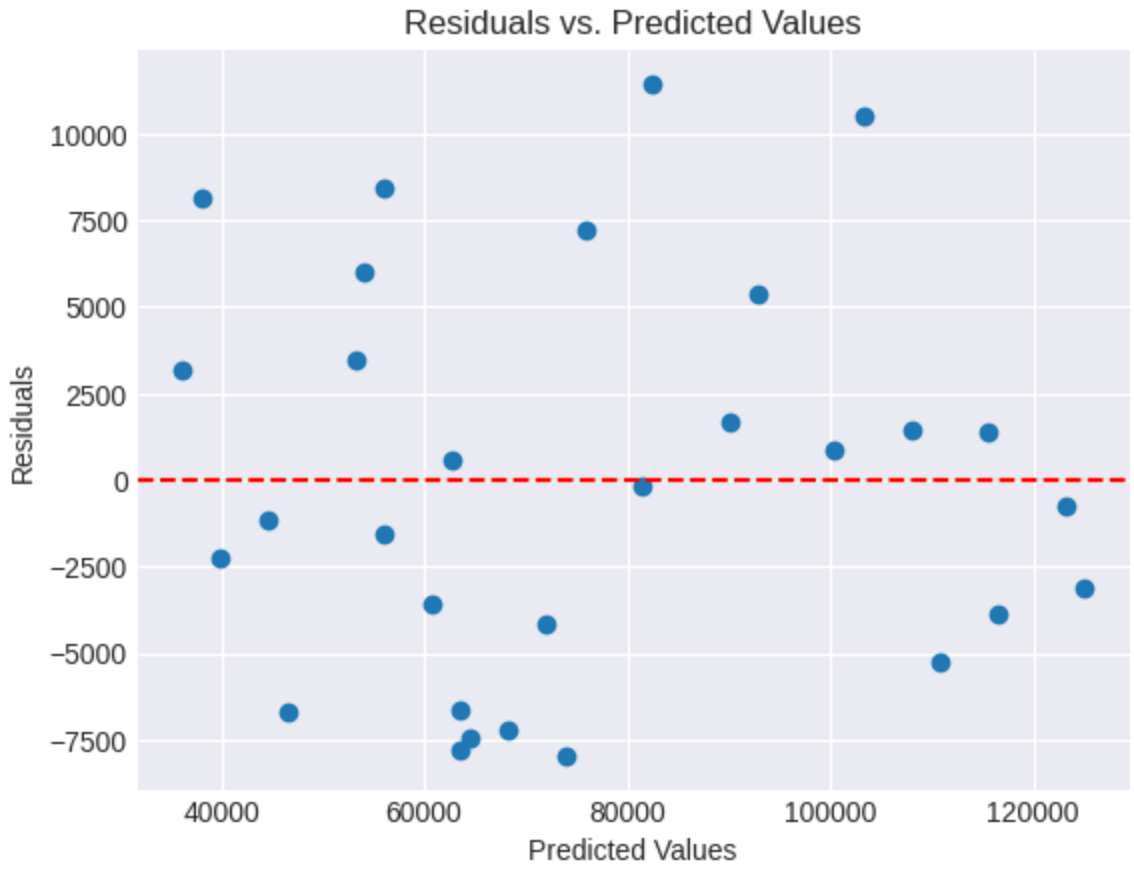

Figure 3: Visual inspection

Testing for normality of residuals

In linear regression analysis, one key assumption is that the residuals (errors) are normally distributed. This assumption is essential when constructing confidence intervals and conducting hypothesis tests on the regression coefficients.

A common way to visually assess the normality of residuals is through a **Q-Q plot** (quantile-quantile plot). In this plot, the quantiles of the residuals are compared against the quantiles of a standard normal distribution. If the points closely follow the diagonal reference line, this suggests that the residuals are approximately normally distributed. Deviations from the line, especially systematic ones (such as an S-shape), can indicate non-normality.

Visual inspection

```
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot of Residuals')
plt.show()
```

For a more formal assessment, we can use normality tests. While several tests exist, such as the *Shapiro-Wilk test*, in this case we apply the **Jarque-Bera test**. The Jarque-Bera test evaluates whether the skewness and kurtosis of the residuals match those of a normal distribution. Specifically:

- A high p-value (greater than the significance level, typically 0.05) indicates that we do not reject the null hypothesis of normality.
- A low p-value suggests that the residuals deviate significantly from a normal distribution.

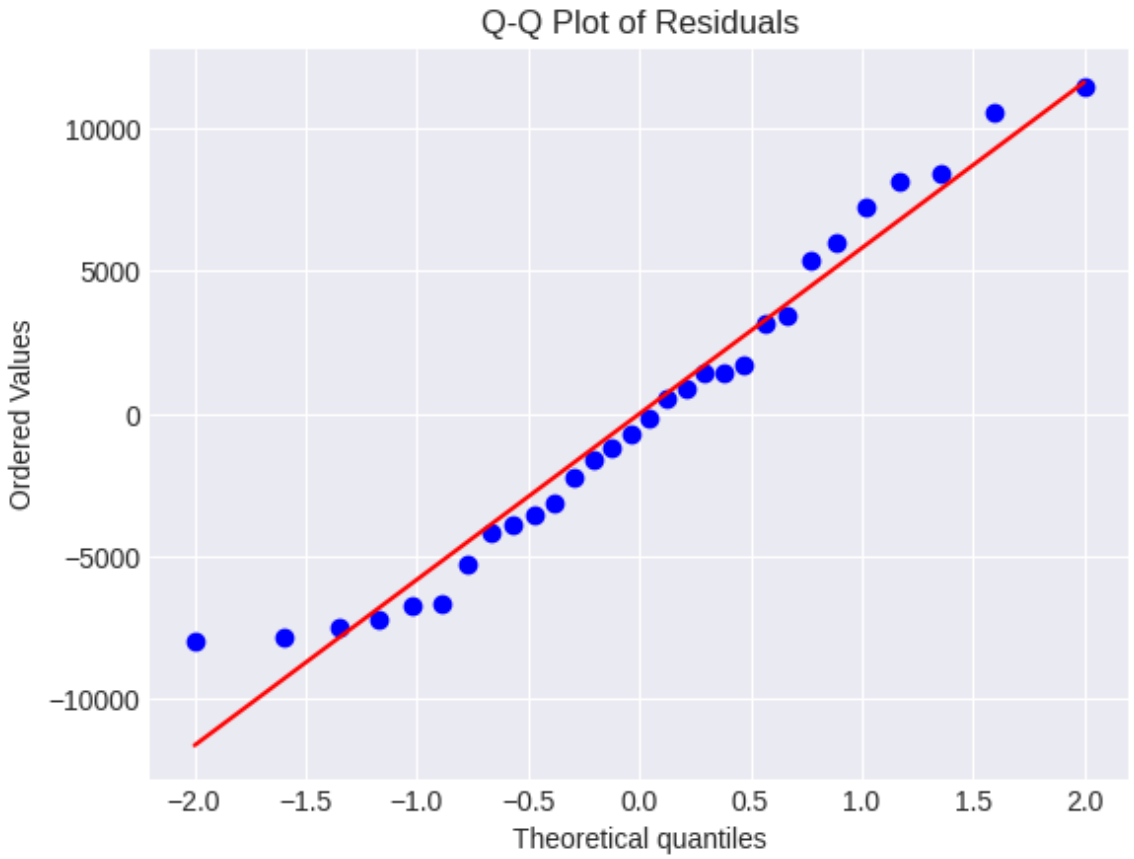

Figure 4: Visual inspection

$$JB = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right)$$

- n is the number of observations.
- S is the sample skewness of the residuals.
- K is the sample kurtosis of the residuals.

Jarque-Bera

```python
jb_stat, jb_pvalue, skew, kurtosis = jarque_bera(model.resid)

print(f"Jarque-Bera: Statistic={jb_stat}, p-value={jb_pvalue}")
print(f"Skewness: {skew}")
print(f"Kurtosis: {kurtosis}")
```

```
Jarque-Bera:  Statistic=1.5691038618767827, p-value=0.45632412070587647
                      Skewness:  0.3629411354039388
                      Kurtosis:  2.1465493462254392
```

Therefore, the residuals follow a normal distribution. (✓)

Combining graphical and statistical evidence allows us to make a more informed judgment about the normality assumption in our regression model.

After carefully conducting each of our diagnostic tests, we can confidently say that our model meets the fundamental assumptions of linear regression. We are, therefore, well within the comfort zone of theory.

We can now proudly present our simple linear regression model — though, as we've seen, there's nothing truly simple about it. (✓)

The model

```
X_series = df['YearsExperience']
reg = model.params.iloc[0] + model.params.iloc[1] * X_series

# Convert to NumPy and sort
X_sorted = np.array(X_series)
reg_sorted = np.array(reg)

sorted_idx = np.argsort(X_sorted)
X_sorted = X_sorted[sorted_idx]
reg_sorted = reg_sorted[sorted_idx]


fig, ax = plt.subplots(figsize=(10, 6))

# Simple scatterplot
sns.scatterplot(data=df, x='YearsExperience', y='Salary', ax=ax,
                color="#1f77b4", edgecolor="black", s=80, label=None)

# Regression line with sorted values
ax.plot(X_sorted, reg_sorted,
        label=f'y = {model.params.iloc[0]:.3f} + {model.params.iloc[1]:.3f}x',
        color="#ff7f0e", linewidth=2.5)

# Aesthetic settings of the plot
ax.set_title('Relationship between Experience and Salary', fontsize=16, pad=15)
ax.set_xlabel('Years of Experience', fontsize=14)
ax.set_ylabel('Salary (\$)', fontsize=14)
ax.legend(frameon=True, facecolor='white', edgecolor='black', fontsize=12)
ax.tick_params(axis='both', labelsize=12)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

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

Figure 5: The model

References

[1] Bruce, P., Bruce, A. & Gedeck, P. (2020). *Practical Statistics for Data Scientists*. O'Reilly.

[2] DeGroot, M. (1988). *Probabilidad y estadística*. Addison-Wesley Iberoamericana.

[3] Montgomery, D., Peck, E. & Vining, G. (2006). *Introducción al análisis de regresión lineal* [Introduction to linear regression analysis]. CECSA.

[4] Sai, A. (n.d.). *Salary Dataset - Simple Linear Regression* [Data set]. Kaggle. Available at https://www.kaggle.com/datasets/abhishek14398/salary-dataset-simple-linear-regression