Learn How to Calculate Sum of Squares (SST, SSR, SSE) for Regression Analysis in Python

Authored by

Mohammed looti

November 1, 2025

The Role of Sums of Squares in Regression Analysis

When conducting any form of regression analysis, the primary goal is to determine how effectively a set of predictor variables can explain the variability observed in a dependent variable. Evaluating model performance requires a standardized framework that allows us to quantify this explanatory power. The concept of decomposing total variance--breaking down the variation in the response data into components explained by the model and components left unexplained--is central to this evaluation process.

This decomposition is achieved through the calculation of the three core metrics known collectively as the **Sums of Squares**. These fundamental values provide a detailed, quantitative assessment of the quality of fit, revealing exactly how much of the total variation in the dependent variable is captured by the fitted regression line. Understanding these metrics is not merely academic; they are the bedrock upon which crucial statistical measures, such as the F-test statistic and the coefficient of determination (R-squared), are built.

The Sums of Squares metrics are integral components of the Analysis of Variance (ANOVA) framework when applied to regression. ANOVA provides a methodology for partitioning the total variability into assignable sources. By meticulously calculating these three sums--Total, Regression, and Error--we gain profound insights into the predictive strength and overall adequacy of our statistical model against the empirical data used for training. Our discussion below will define each metric rigorously and then demonstrate their precise calculation using powerful Python libraries like Pandas, Statsmodels, and NumPy.

Deconstructing Data Variation: Defining SST, SSR, and SSE

To accurately measure a model's effectiveness, we must first establish clear definitions for the sources of variation. The three essential metrics--SST, SSR, and SSE--each represent a distinct component of the data's variability, measured relative to either the overall mean or the model's predictions. These metrics are always calculated using squared deviations, which ensures that positive and negative differences do not cancel each other out, thereby providing a true measure of magnitude.

The following ordered list outlines the specific definitions and mathematical formulas for each of the three key sums of squares metrics, which are central to evaluating any statistical model:

Sum of Squares Total (SST)

The **SST** represents the total inherent variation present within the response variable (Y). Conceptually, it measures the variability of the observed data points around their own mean. If the model provided zero explanatory power, the SST would represent the benchmark error, as using

the mean of Y would be the simplest prediction. It is calculated by summing the squared differences between each individual observed data point (yi) and the overall mean of the response variable (y).

SST = Σ(yi - y)2

Sum of Squares Regression (SSR)

Also frequently referred to as the Sum of Squares Explained, the **SSR** quantifies the proportion of the total variation in the dependent variable that is successfully accounted for or explained by the regression model. Essentially, it measures how much better the model's predictions are compared to simply using the mean of the response variable. A larger SSR indicates that the model is doing a better job of capturing the underlying relationship between the predictors and the response. It is calculated as the sum of squared differences between the predicted data points (?i) and the mean of the response variable (y).

SSR = Σ(?i - y)2

Sum of Squares Error (SSE)

Also known as the Sum of Squares Residual, **SSE** is perhaps the most critical measure of the model's performance, representing the variation that remains unexplained by the regression line. This is the residual error, or the difference between what was observed and what the model predicted. Minimizing SSE is the core objective of the Ordinary Least Squares (OLS) fitting procedure. It is calculated as the sum of squared differences between the observed data points (yi) and their corresponding predicted data points (?i).

SSE = Σ(yi - ?i)2

It is important to note the distinction in the SSE formula here versus the SSR formula: SSE compares the observed value to the predicted value, reflecting the vertical distance from the data point to the line, whereas SSR compares the predicted value to the overall mean, reflecting the improvement provided by the model.

The Fundamental Relationship: SST = SSR + SSE

The three sums of squares are intrinsically linked by a powerful and fundamental identity: **SST = SSR + SSE**. This equation is not just a convenient identity; it is the mathematical expression of the decomposition of variance, confirming that the total variation in the data must equal the sum of the variation explained by the model plus the variation unexplained by the model.

This relationship is crucial because it forms the denominator and numerator components required

for calculating the coefficient of determination (R-squared). R-squared, often interpreted as the percentage of the response variable variance that is predictable from the predictor variables, is formally defined as: R^2 = SSR / SST. Since SSR and SSE must sum to SST, R^2 must always fall between 0 and 1 (or 0% and 100%), providing a normalized measure of model fit.

A high ratio of SSR to SSE indicates a strong model fit, meaning a large portion of the total variability is successfully captured by the regression equation. Conversely, if SSE approaches SST, the SSR must be near zero, signifying that the model's predictions are no better than using the simple mean of the response variable, implying a very poor fit. The example provided below, utilizing a simple linear regression model, will concretely demonstrate how this identity holds true in a computational setting, ensuring the reliability of our calculations.

Step 1: Preparing and Structuring the Dataset in Python

The first practical step in calculating these metrics within a computational environment is to establish a well-organized dataset. For complex data manipulation and statistical analysis in Python, the Pandas library is the standard tool. Pandas provides the DataFrame structure, which is ideal for storing and accessing structured data efficiently, allowing us to easily designate predictor and response variables.

For our demonstration, we utilize a hypothetical dataset comprising 20 observations related to student study habits, specifically tracking the number of hours studied and the corresponding exam score received. We organize this data into a Pandas DataFrame, ensuring that the variables are clearly defined and ready for statistical modeling. This setup allows for straightforward extraction of the observed values (yi) and the calculation of the required mean (y).

import pandas as pd

# Create pandas DataFrame representing hours studied and exam score
df = pd.DataFrame({'hours': ,
'score': })

# Display the initial structure of the DataFrame
df.head()

hours score
0 1 68
1 1 76
2 1 74
3 2 80
4 2 76

The resulting DataFrame clearly presents our independent variable, `hours`, and our dependent variable, `score`. This structured approach is essential for the next step, where we will use these columns to define our model specification and derive the predicted values needed for the Sums of Squares calculations.

Step 2: Implementing Ordinary Least Squares (OLS) with Statsmodels

Once the data is prepared, the next phase involves fitting the regression model itself. We choose the Statsmodels library, a robust Python package designed for statistical modeling and econometric analysis, utilizing its **OLS()** (Ordinary Least Squares) function. OLS is the standard method for estimating the parameters of linear regression models, aiming to minimize the Sum of Squares Error (SSE).

In our model definition, `score` serves as the response variable (Y), and `hours` is designated as the primary predictor variable (X). A critical procedural requirement when using Statsmodels is the explicit inclusion of an intercept term (constant) in the predictor matrix. Without this constant, the regression line would be forced through the origin (0,0), which is usually unrealistic. We achieve this necessary inclusion using the `sm.add_constant(x)` function before the model estimation.

import statsmodels.api as sm

# Define the response variable (y)
y = df

# Define the predictor variable (x)
x = df]

# Add the constant term (intercept) to the predictor variables
x = sm.add_constant(x)

# Fit the simple linear regression model using OLS
model = sm.OLS(y, x).fit()

The execution of the `model.fit()` method determines the coefficients (slope and intercept) that best fit the data according to the least squares criterion. Crucially, the resulting `model` object contains the necessary fitted values (?i), which are the predicted scores for each student based on the hours studied. These fitted values are indispensable, as they serve as the input for calculating both the SSR and the SSE in the subsequent step.

Step 3: Calculating Sums of Squares Metrics using NumPy

With the observed values (yi) and the predicted values (?i) readily available from our Pandas DataFrame and the fitted model, we can now proceed to calculate the three sums of squares metrics. We leverage the NumPy library for its efficiency in performing vectorized mathematical operations, which translates the statistical formulas directly into optimized Python code.

The calculation is executed in three distinct parts, mirroring the definitions established earlier. First, **SSE** is calculated by taking the square of the difference between the actual scores (observed) and the model's predictions (fitted values), and then summing these squared residuals. Second, **SSR** is calculated by squaring the difference between the fitted values and the overall mean of the observed scores, and summing these values. Finally, **SST** can be calculated either directly from the observed data and its mean, or more simply, by utilizing the fundamental identity: **SST = SSR + SSE**.

import numpy as np

```
# Calculate SSE (Sum of Squares Error: Observed - Predicted)
sse = np.sum((df.score - model.fittedvalues)**2)
print(sse)
```

331.07488479262696

```
# Calculate SSR (Sum of Squares Regression: Predicted - Mean)
ssr = np.sum((model.fittedvalues - df.score.mean())**2)
print(ssr)
```

917.4751152073725

```
# Calculate SST (Sum of Squares Total: SSR + SSE)
sst = ssr + sse
print(sst)
```

1248.5499999999995

This NumPy implementation is highly efficient and directly reflects the mathematical definitions of the sums of squares. We have successfully decomposed the total variability in the student scores into the portion explained by the hours studied (SSR) and the portion attributable to random error or unmodeled factors (SSE).

Verification and Interpretation of Results

The execution of the Python script yields precise numerical values for the three core sums of squares metrics. These figures quantify the model's overall explanatory power and provide the raw materials necessary for further statistical inference, such as calculating R-squared.

Sum of Squares Total (SST): 1248.55
Sum of Squares Regression (SSR): 917.4751
Sum of Squares Error (SSE): 331.0749

The most crucial step following calculation is verification, ensuring that the fundamental relationship holds. We confirm that the total variation (SST) is equal to the explained variation (SSR) plus the unexplained variation (SSE):

SST = SSR + SSE
1248.55 ≈ 917.4751 + 331.0749
1248.55 ≈ 1248.55

This verification confirms the integrity of our calculations and affirms that the variance decomposition is mathematically sound. Furthermore, the substantial difference between the **SSR** (917.4751) and the **SSE** (331.0749) suggests that the amount of variation explained by the predictor variable (hours studied) is significantly larger than the variation left as residual error. This indicates a strong fit for the simple linear regression model, a fact that would translate into a high R-squared value (approximately 917.4751 / 1248.55 ≈ 0.735, or 73.5%).

Additional Resources

While the step-by-step methodology using Python provides maximum transparency and control, alternative tools exist for quick calculations and deeper conceptual understanding. For further exploration, you may find the following resources helpful for calculating these metrics in other contexts or for utilizing automated calculation tools.

You can use the following calculators to automatically calculate SST, SSR, and SSE for any simple linear regression line:

The following tutorials explain how to calculate SST, SSR, and SSE in other statistical software: