How to Easily Perform Power Regression in R

Authored by

stats writer

December 6, 2025

Performing Power regression in **R** requires a specialized approach, as standard power models are inherently non-linear regression and cannot be directly solved using the basic linear least squares method. However, by applying a logarithmic transformation, the power relationship can be converted into a linear form. This crucial step allows us to utilize the powerful and versatile **R** built-in function, lm() function, which is designed for fitting **linear models**.

The core strategy involves transforming both the response variable (y) and the predictor variable (x) using the natural logarithm. Once linearized, the model is specified within the formula argument of lm() function. This process efficiently fits the data and generates essential regression coefficients, along with vital summary statistics such as the R-squared and the **F-statistic**. Understanding this log-linear transformation is key to successfully modeling exponential growth, decay, or other scaling phenomena where the relationship follows a power law.

Understanding the Power Regression Model

The application of Power regression is fundamental in many scientific and engineering disciplines where proportional scaling relationships are observed. Unlike simple linear models, which assume an additive relationship, the power model captures multiplicative effects, meaning that a constant percentage change in the predictor variable often results in a constant percentage change in the response variable. This makes it an ideal choice for phenomena like allometric scaling in biology or cost-to-scale modeling in economics.

The standard mathematical representation of a power regression model takes the form of:

y = axb

In this equation:

y: The response variable
x: The predictor variable
a, b: The regression coefficients that describe the relationship between *x* and *y*

Since **x** is raised to a power **b**, this structure is inherently curvilinear, complicating direct estimation using standard ordinary least squares (OLS) methods. To bypass the complexities of true non-linear regression, we employ a technique known as **linearization**. This method relies on the mathematical property of logarithms, converting the multiplicative form into an additive form.

By taking the natural logarithm (ln) of both sides of the power equation, we achieve a model structure that is linear in terms of the transformed variables, allowing us to leverage the simplicity and robustness of the lm() function. Specifically, applying the natural logarithm yields the following transformation, which is the key to running Power regression in **R**:

ln(y) = ln(a) + b * ln(x)

Step 1: Preparing the Data for Analysis

The initial stage of any statistical analysis involves setting up the dataset. For this example, we will simulate a small dataset in **R** that is designed to exhibit a clear power relationship. Generating artificial data allows us to precisely demonstrate the necessary steps without the distraction of real-world data cleaning complexities.

We define our predictor variable, **x**, as a sequence of integers from 1 to 20. The response variable, **y**, is then explicitly defined to show a non-linear, increasing trend consistent with a power function. A crucial prerequisite for applying log transformation is that all values in both **x** and **y** must be strictly positive, as the logarithm of zero or a negative number is undefined in the real domain.

The following **R** code snippet creates these two vectors, initiating our power regression study:

```
# Create synthetic data vectors for demonstration
x=1:20
y=c(1, 8, 5, 7, 6, 20, 15, 19, 23, 37, 33, 38, 49, 50, 56, 52, 70, 89, 97, 115)
```

This dataset, comprising 20 paired observations, forms the foundation for our model fitting. A preliminary review of the **y** values shows an accelerating rate of increase, which immediately suggests that a model capable of handling curvilinear relationships, such as Power regression, will be far more appropriate than a standard linear model.

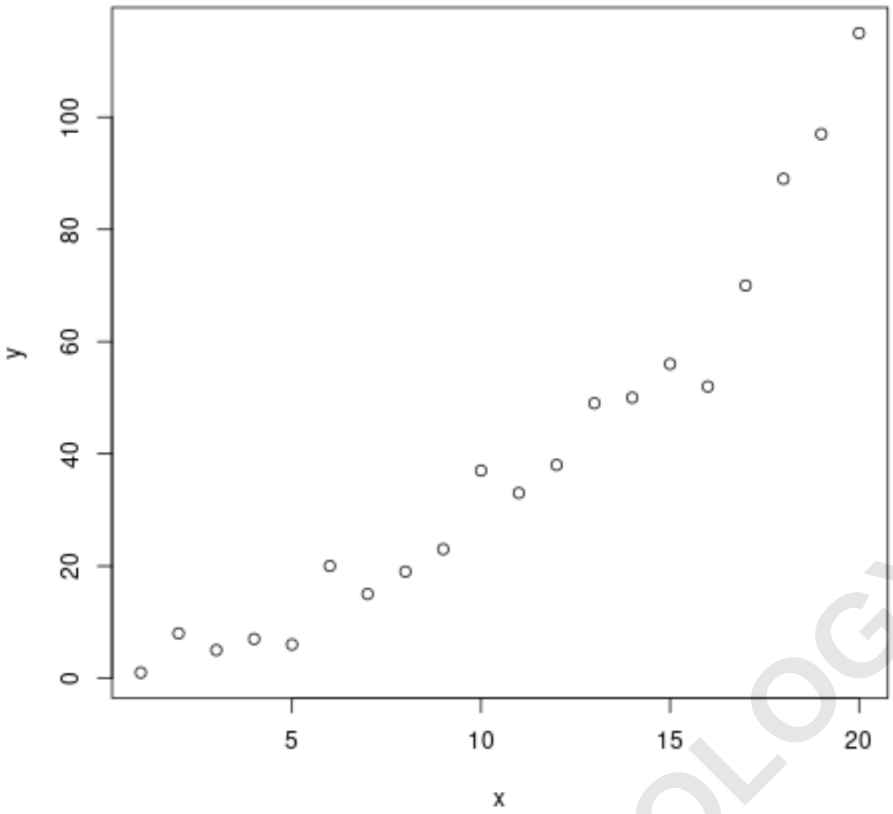
Step 2: Visual Confirmation of Relationship Type

Before fitting any statistical model, it is a non-negotiable step to visualize the data. A scatterplot offers immediate insight into the form of the relationship between the predictor variable **x** and the response variable **y**, helping us confirm that a power model is the correct choice.

We use the fundamental `plot()` function in **R** to generate this visualization. The graphical output serves as the empirical justification for our methodological choice, demonstrating whether the data follows a linear path, an exponential curve, or a power curve.

```
# Generate a scatterplot to visualize the relationship between x and y
plot(x, y)
```

The resulting plot is displayed below:



As clearly illustrated by the visual evidence, the relationship is distinctly non-linear and curvilinear, with the slope growing increasingly steeper as **x** increases. This visual pattern strongly supports the use of a power model. Attempting to fit a simple linear regression would result in residuals that are heteroscedastic and patterned, indicating a poor model specification. This visualization confirms the necessity of the Log transformation approach that defines Power regression using **R**'s linear tools.

Step 3: Linearizing the Model using Log Transformation

To perform power regression using the **lm()** function, we must convert the original multiplicative relationship (y = axb) into its additive, linear counterpart (ln(y) = ln(a) + b * ln(x)). This procedure, known as the **double-log transformation**, is executed seamlessly within the lm() function call.

By specifying `log(y) ~ log(x)` in the model formula, **R** computes the natural logarithm of every data point in both vectors internally before executing the regression. The primary advantage of this Log transformation is that it stabilizes the variance and often normalizes the error distribution, fulfilling the underlying assumptions required for Ordinary Least Squares (OLS) regression, which the **lm()** function performs.

It is crucial to remember that we are fitting a linear model to the *transformed* data. Therefore, the

resulting coefficients relate to the log-transformed variables, not the original variables. The intercept will be the estimate for **ln(a)**, and the slope will be the estimate for the exponent **b**. This foundational step bridges the gap between non-linear regression theory and linear modeling execution in **R**.

Step 4: Fitting the Power Regression Model using lm()

We now proceed to fit the linearized model. We store the output in an object named `model` and subsequently call the `summary()` function to examine the statistical results comprehensively. This summary provides detailed information on the fitted parameters, their standard errors, measures of statistical significance (t-values and p-values), and overall model fit statistics.

The code below executes the regression using the natural log of **y** as the dependent variable and the natural log of **x** as the independent variable.

 # Fit the linearized power model using the lm() function
model <- lm(log(y)~ log(x))

# View the detailed statistical output of the model
summary(model)

Call:
lm(formula = log(y) ~ log(x))

Residuals:
Min 1Q Median 3Q Max
-0.67014 -0.17190 -0.05341 0.16343 0.93186

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.15333 0.20332 0.754 0.461
log(x) 1.43439 0.08996 15.945 4.62e-12 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3187 on 18 degrees of freedom
Multiple R-squared: 0.9339, Adjusted R-squared: 0.9302
F-statistic: 254.2 on 1 and 18 DF, p-value: 4.619e-12

The output confirms a successful model fit to the logarithmic data. The subsequent step involves meticulously interpreting the results, particularly focusing on the coefficient estimates and the

overall metrics of model performance.

Step 5: Detailed Interpretation of Model Coefficients and Fit

A careful examination of the **Coefficients** table is necessary to extract the parameters for our final power equation. The table provides two essential regression coefficients from the log-linear model **ln(y) = ln(a) + b * ln(x)**.

Intercept (0.15333): This is the estimate for **ln(a)**. Although its p-value (0.461) suggests the intercept itself is not statistically different from zero in the linearized model, this value is vital for calculating the scaling factor **a** in the original equation.

log(x) Estimate (1.43439): This value is the estimate for the exponent **b**. Because the coefficient is highly significant (p-value = 4.62e-12), we can confidently state that the transformed predictor variable significantly influences the transformed response variable.

The **Multiple R-squared (0.9339)** statistic is highly encouraging, indicating that 93.39% of the variability in the log-transformed response is successfully accounted for by the log-transformed predictor. Such a high R-squared value confirms that the log-linear model provides an excellent fit to the transformed data. The highly significant **F-statistic** further reinforces the notion that the model is statistically useful.

We can now write the fitted equation in its linearized form:

ln(y) = 0.15333 + 1.43439 * ln(x)

Step 6: Converting Back to the Original Power Form

The linearized model is not directly usable for prediction in the original domain (x and y); thus, we must reverse the Log transformation. We apply the exponential function (e) to both sides of the log-linear equation to return to the form **y = axb**.

Starting with the linearized equation, we apply **e**:

y = e (0.15333 + 1.43439 * ln(x))

Using properties of exponents, we separate the terms:

y = e 0.15333 * e 1.43439 * ln(x)

The coefficient **a** is calculated as **e0.15333**, resulting in approximately 1.1657. The exponent **b** is directly given by the slope coefficient, 1.43439.

The final, estimated Power regression equation is:

y = 1.1657x1.43439

This equation now perfectly models the curvilinear relationship observed in the original scatterplot, allowing us to proceed to practical application and prediction based on the values of the predictor variable.

Step 7: Utilizing the Power Equation for Prediction

The derived non-linear equation, **y = 1.1657x1.43439**, is the final product of our analysis and is ready to be used for prediction. We can use this equation to estimate the response variable **y** for any given value of **x** within the scope of our data range.

For demonstration, let us predict the value of **y** when **x** equals 12. We substitute 12 into our formula:

y = 1.1657(12)1.43439

The calculation proceeds as follows: 12 raised to the power of 1.43439 is approximately 35.316. Multiplying this by the scaling coefficient (1.1657) yields:

y = 1.1657 * 35.316 ≈ **41.167**

The predicted value for **y** when **x = 12** is **41.167**. This exercise validates the successful use of the lm() function and the log transformation methodology to solve complex non-linear regression problems efficiently in **R**.

Bonus Resource: Feel free to use this online calculator to automatically compute the power regression equation for a given predictor and response variable, which is useful for verifying manual calculations or results from **R**.

Conclusion

The methodology for performing Power regression in **R** is a powerful technique for modeling multiplicative relationships often found in natural and economic sciences. By employing the double-log transformation, we successfully linearize the power equation, enabling us to utilize the robust and reliable lm() function. The resulting model, validated by a high R-squared value, provides precise regression coefficients that translate directly back into a usable non-linear formula for accurate prediction. Mastery of this transformation process is essential for statistical practitioners seeking to model scaling relationships effectively.