

How to Easily Fit Curves in R Using lm, nls, and glm

Authored by
stats writer

December 5, 2025

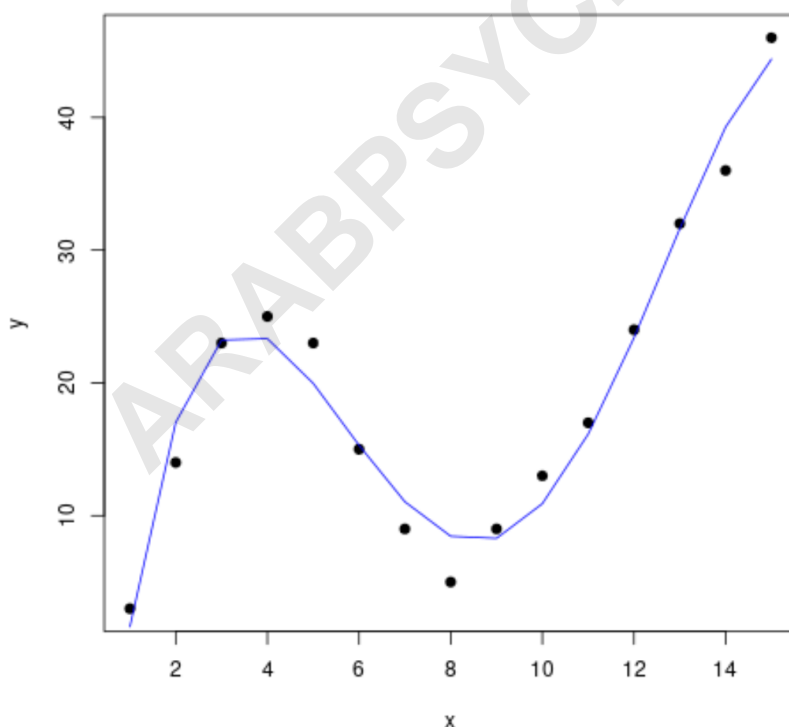
RECOMMENDED CITATION

stats writer (2025). *How to Easily Fit Curves in R Using lm, nls, and glm*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=106006>

Curve fitting is a fundamental technique in data science and statistics used to construct a mathematical function that best represents the relationship between a set of data points. In the context of R, this process allows practitioners to transform raw data into predictive statistical models. The goal of curve fitting is often to capture the underlying trend or pattern in the data, enabling accurate interpolation and extrapolation, which is crucial for forecasting and understanding complex phenomena.

In R, curve fitting is accomplished using robust functions available both in the base package and specialized libraries like 'stats' and 'graphics'. The most commonly employed functions for fitting various relationships include **lm()** (for linear models and standard polynomial fitting), **nls()** (for dedicated nonlinear least squares fitting), and **glm()** (for generalized linear models). These tools allow analysts to move beyond simple straight lines to fit complex curves, including high-degree polynomial regression, ensuring the chosen model parameters optimally describe the observed variability in the dataset.

This comprehensive guide details a step-by-step process for performing effective curve fitting using R. We will specifically focus on utilizing the **lm()** function in conjunction with the powerful poly() function to explore different polynomial degrees. Furthermore, we will establish a rigorous method for determining which curve provides the best fit by evaluating key diagnostic statistics, ultimately leading to the selection of a robust and parsimonious model.



Often, the initial visual inspection of the data suggests that a simple linear relationship is

inadequate, necessitating the exploration of curved or nonlinear mathematical expressions to accurately model the relationship between variables.

Understanding Polynomial Regression in Curve Fitting

When the relationship between the predictor variable (X) and the response variable (Y) is distinctly curvilinear, simple linear models fail to capture the true structure efficiently. This is precisely where polynomial regression becomes invaluable. Polynomial regression is technically a form of linear regression where the relationship between the independent variable and the dependent variable is modeled as an Nth-degree polynomial. The degree of the polynomial--be it quadratic (degree 2), cubic (degree 3), or higher--determines the complexity and curvature of the fitted line, allowing the model to bend and turn to match the data's inherent shape.

Selecting the correct polynomial degree is a critical analytical decision. A model that is too simple (a low degree, such as linear) may suffer severely from **underfitting**, failing to capture essential variations and resulting in high bias. Conversely, a model that is unnecessarily complex (a high degree, such as 5 or more) risks **overfitting**, where the curve fits the random noise or specific idiosyncrasies of the training data rather than the generalized underlying signal. Overfitting leads to models that perform poorly when exposed to new, unseen observations. Our systematic methodology is designed to test multiple polynomial degrees to find the ideal balance between complexity and explanatory power.

In the R environment, the **poly()** function is specifically engineered to generate the necessary orthogonal or raw polynomial terms for seamless use within the **lm()** framework. By specifying the argument `raw=TRUE`, as we will demonstrate in the following steps, we ensure that the generated coefficients directly correspond to the standard algebraic polynomial equation form ($\beta_n x^n$), making the interpretation and derivation of the final mathematical expression straightforward and precise.

Step 1: Preparing and Visualizing the Data

The foundation of any successful statistical analysis is the rigorous preparation and initial visualization of the dataset. Before attempting to fit any complex statistical model, we must first define our dataset structure and examine its distribution visually using a scatterplot. This preliminary step is crucial because the visual representation offers immediate, qualitative insights into the shape of the relationship between variables, thus guiding our selection of potential model types and complexity.

For this demonstration of curve fitting, we will construct a synthetic data frame in R. This frame, named `df`, contains fifteen observations for two variables: a continuous independent variable, `x` (ranging sequentially from 1 to 15), and a dependent variable, `y`, which has been intentionally

structured to exhibit a distinct curvilinear pattern. The subsequent execution of the plotting command visualizes this relationship.

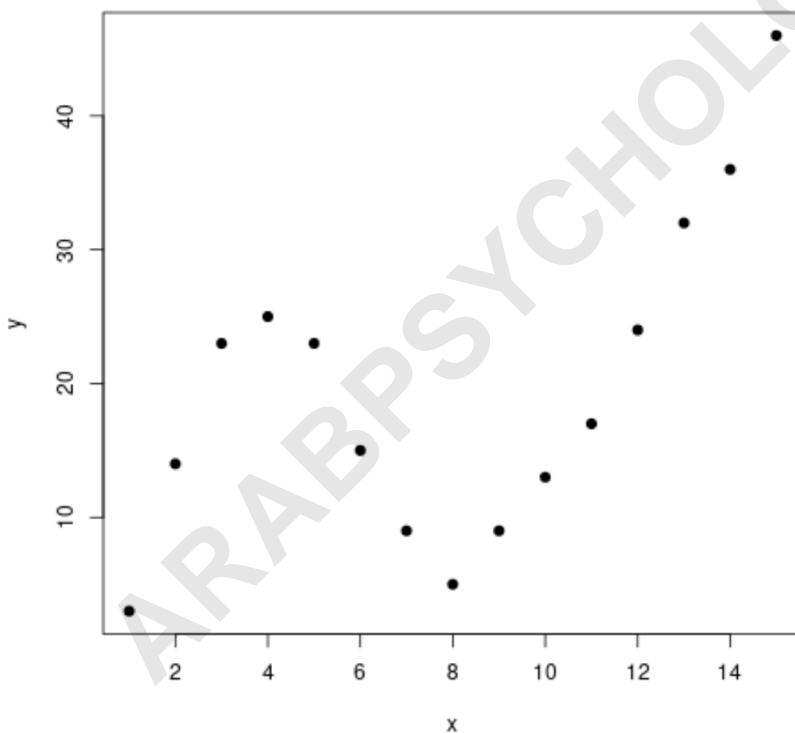
```
# Create the sample data frame named 'df'
```

```
df <- data.frame(x=1:15,  
y=c(3, 14, 23, 25, 23, 15, 9, 5, 9, 13, 17, 24, 32, 36, 46))
```

```
# Create a scatterplot of x vs. y to visualize the inherent relationship
```

```
plot(df$x, df$y, pch=19, xlab='x', ylab='y')
```

As illustrated by the output plot, the relationship between x and y is manifestly non-linear. The initial data points suggest rapid growth, followed by a local peak and trough, before accelerating again steeply towards the end of the observed range. This complex, undulating curve shape confirms that a simple first-degree linear model will be inadequate and dictates that we must investigate polynomial terms of degree two or higher to achieve an acceptable fit.



Step 2: Implementing Multiple Polynomial Regression Models

Following the visual confirmation of a non-linear trend, the next crucial step is to fit and evaluate a comprehensive suite of candidate models. We leverage the **lm()** function in R, which handles multiple regression, in tandem with the poly() function, systematically iterating through polynomial degrees from 1 (the simplest) up to 5 (a relatively high degree). By fitting five distinct models, we

create a spectrum of complexity to evaluate.

We create five distinct model objects--`fit1` through `fit5`--each incorporating an increasing degree of the polynomial term. This systematic experimentation allows for a direct comparison of how the inclusion of higher-order terms affects the goodness-of-fit. The linear model (`fit1`) serves as the necessary minimum baseline against which the predictive capability and statistical significance of all higher-degree polynomial regression models are benchmarked.

The subsequent code block not only executes the model fitting but also proceeds immediately to visualize the predicted curves of all five competing models overlaid onto the same original scatterplot. This provides an immediate, tangible assessment of the fit, illustrating which curve visually tracks the data points most accurately without exhibiting erratic behavior or excessive curvature (a sign of potential overfitting).

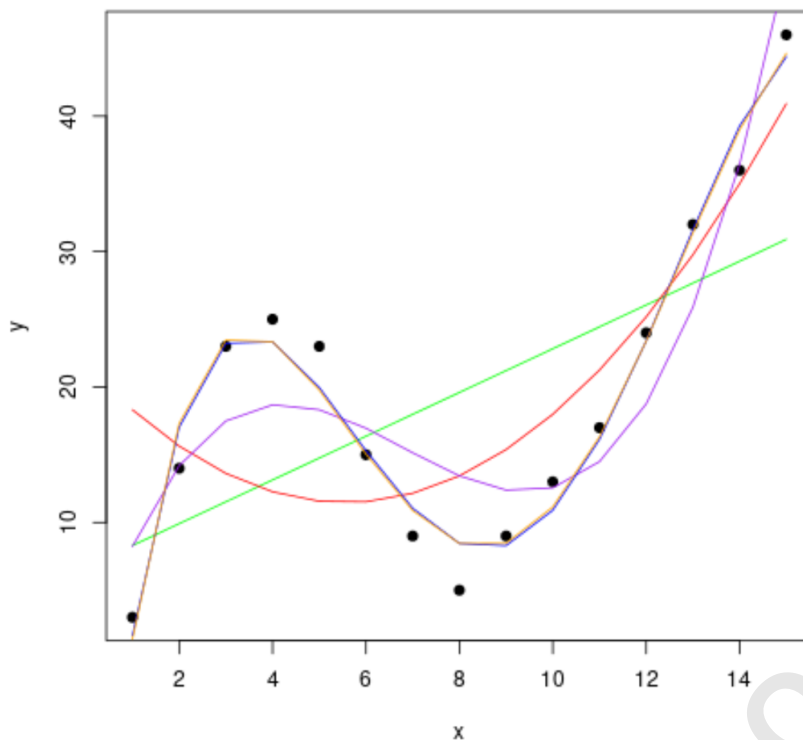
Fit polynomial regression models up to degree 5. raw=TRUE ensures standard polynomial coefficients.

```
fit1 <- lm(y~x, data=df) # Degree 1 (Linear model)
fit2 <- lm(y~poly(x,2,raw=TRUE), data=df) # Degree 2 (Quadratic model)
fit3 <- lm(y~poly(x,3,raw=TRUE), data=df) # Degree 3 (Cubic model)
fit4 <- lm(y~poly(x,4,raw=TRUE), data=df) # Degree 4 (Quartic model)
fit5 <- lm(y~poly(x,5,raw=TRUE), data=df) # Degree 5 (Quintic model)
```

```
# Re-create the initial scatterplot of observed data points
plot(df$x, df$y, pch=19, xlab='x', ylab='y')
```

```
# Define a sequence of x-values for smooth curve prediction
x_axis <- seq(1, 15, length=15)
```

```
# Add the predicted curve of each model to the plot for visual comparison
lines(x_axis, predict(fit1, data.frame(x=x_axis)), col='green')
lines(x_axis, predict(fit2, data.frame(x=x_axis)), col='red')
lines(x_axis, predict(fit3, data.frame(x=x_axis)), col='purple')
lines(x_axis, predict(fit4, data.frame(x=x_axis)), col='blue')
lines(x_axis, predict(fit5, data.frame(x=x_axis)), col='orange')
```



Step 3: Quantitative Comparison Using Adjusted R-squared

While visual confirmation from the previous step is insightful, making an objective decision about the optimal model requires a robust quantitative metric. For comparing multiple statistical models that inherently use a different number of predictor variables, the standard R-squared value is misleading because it always increases when more variables are added, even if those variables are statistically insignificant. This bias favors overly complex models.

To counteract this, we utilize the Adjusted R-squared (R^2_{adj}) statistic. This metric measures the proportion of variance in the response variable explained by the predictors, but it crucially incorporates a penalty for each additional predictor variable introduced into the model. Consequently, the R^2_{adj} only increases if the new term significantly improves the model's explanatory power more than would be expected by chance.

A higher Adjusted R-squared value consistently indicates a better fit, especially when evaluating models of differing complexity (different polynomial degrees). We calculate this statistic for all five fitted models using the **summary()** function in R and extracting the relevant summary component.

Calculate Adjusted R-squared of each fitted model

```
summary(fit1)$adj.r.squared
```

```
summary(fit2)$adj.r.squared
```

```
summary(fit3)$adj.r.squared
```

```
summary(fit4)$adj.r.squared
```

```
summary(fit5)$adj.r.squared
```

```
0.3144819
```

```
0.5186706
```

```
0.7842864
```

```
0.9590276
```

```
0.9549709
```

Analyzing the computed values, we observe a substantial improvement in fit as we move from the linear (0.314) to the quartic (0.959) model. Crucially, the fifth-degree polynomial model (`fit5`) yields a slightly decreased R^2_{adj} of 0.9549709. This drop confirms that the additional complexity introduced by the fifth-degree term did not justify its inclusion based on the explained variance, indicating the start of potential overfitting. Consequently, the model with the maximum Adjusted R-squared, `fit4` (the fourth-degree polynomial), is identified as the optimal choice.

Step 4: Isolating and Visualizing the Final Curve

With the fourth-degree polynomial model (`fit4`) quantitatively confirmed as the best fit, the next operational step is to create a refined visualization focused exclusively on this optimal result. This visualization serves as the definitive graphical output of the curve fitting process, clearly demonstrating the final relationship identified.

We generate a new scatterplot containing only the original data points and overlay the predicted curve derived from the `fit4` model. This clear presentation is vital for confirming the model's accuracy visually and communicating the findings effectively to stakeholders.

```
# Create a clean scatterplot of x vs. y
```

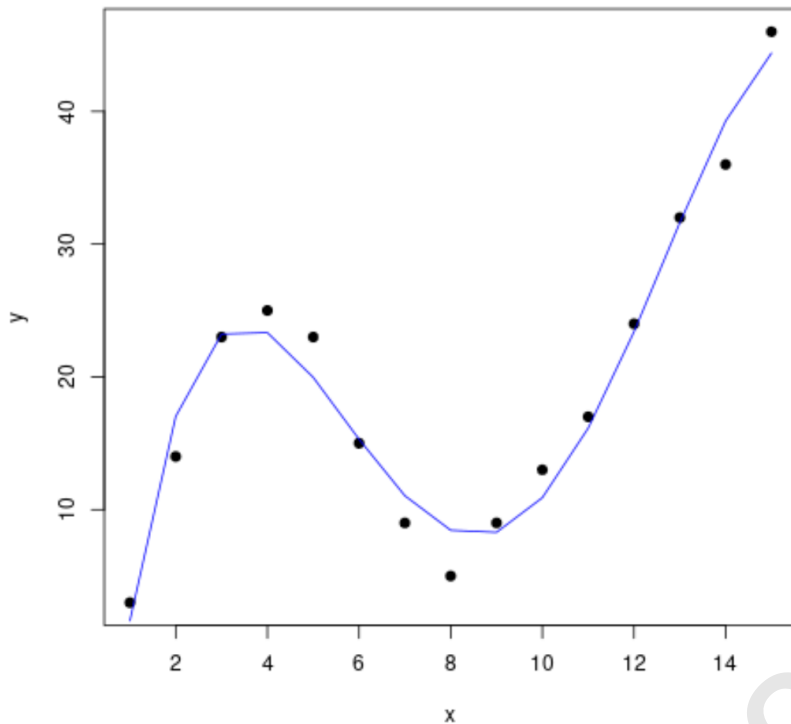
```
plot(df$x, df$y, pch=19, xlab='x', ylab='y')
```

```
# Define x-axis values for smooth curve plotting
```

```
x_axis <- seq(1, 15, length=15)
```

```
# Add the curve of the optimal fourth-degree polynomial model (fit4)
```

```
lines(x_axis, predict(fit4, data.frame(x=x_axis)), col='blue')
```



The resulting plot confirms that the quartic function (the blue line) successfully and smoothly maps the complex, undulating trend within the observed data points. This visual congruence strongly supports the statistical conclusion drawn from the R^2_{adj} analysis, confirming that the fourth-degree model is robust and suitable for predictive use.

Step 5: Deriving the Final Model Equation

For rigorous application, the fitted curve must be expressed as a precise mathematical equation. This equation is essential for implementation in other software or for manual calculation of predictions. We achieve this by examining the detailed output provided by the **summary()** function applied to our selected model, `fit4`.

The core information required is found within the 'Coefficients' table. Since we used `raw=TRUE` with the `poly()` function, the 'Estimate' values listed correspond directly to the intercept (β_0) and the coefficients for the X terms: X^1 , X^2 , X^3 , and X^4 . For example, the estimate for '(Intercept)' is β_0 , and the estimate for `'poly(x, 4, raw = TRUE)4'` is β_4 .

summary(fit4)

Call:

```
lm(formula = y ~ poly(x, 4, raw = TRUE), data = df)
```

Residuals:

```
Min 1Q Median 3Q Max
-3.4490 -1.1732 0.6023 1.4899 3.0351
```

Coefficients:

```
Estimate Std. Error t value Pr(>|t|)
(Intercept) -26.51615 4.94555 -5.362 0.000318 ***
poly(x, 4, raw = TRUE)1 35.82311 3.98204 8.996 4.15e-06 ***
poly(x, 4, raw = TRUE)2 -8.36486 0.96791 -8.642 5.95e-06 ***
poly(x, 4, raw = TRUE)3 0.70812 0.08954 7.908 1.30e-05 ***
poly(x, 4, raw = TRUE)4 -0.01924 0.00278 -6.922 4.08e-05 ***
```

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

Residual standard error: 2.424 on 10 degrees of freedom

Multiple R-squared: 0.9707, Adjusted R-squared: 0.959

F-statistic: 82.92 on 4 and 10 DF, p-value: 1.257e-07

By substituting the coefficients from the 'Estimate' column into the general polynomial formula, $Y = \text{beta}_0 + \text{beta}_1 X + \text{beta}_2 X^2 + \text{beta}_3 X^3 + \text{beta}_4 X^4$, we obtain the following precise mathematical equation for the optimal fourth-degree curve:

$$y = -0.0192x^4 + 0.7081x^3 - 8.3649x^2 + 35.823x - 26.516$$

Predictive Application of the Model

Once the equation is derived, the model shifts from being a descriptive tool to a powerful predictive instrument. The primary objective of successful curve fitting is the ability to use the resulting mathematical relationship to predict unknown or future values of the response variable (Y) based on new inputs for the predictor variable (X). This functionality is essential for making forecasts and simulating outcomes under different conditions.

For example, we can use this established fourth-degree relationship to predict the expected value of y at a specific point, such as when $x = 4$. This involves simple substitution into the derived polynomial equation.

Let's calculate the predicted y value for $x=4$ using the derived equation:

$$y = -0.0192(4)^4 + 0.7081(4)^3 - 8.3649(4)^2 + 35.823(4) - 26.516$$

The calculation yields a predicted value of $y = 23.34$. This confirms the practical utility of the polynomial regression model, allowing for accurate estimations across the domain of the fitted

curve.

Conclusion: Mastering Curve Fitting in R

The process of mastering curve fitting in R is vital for any analyst encountering non-linear data patterns. By following a systematic, multi-step approach--from data visualization and fitting multiple models using **lm()** and the poly() function, to critical evaluation using the Adjusted R-squared statistic--we ensure the selection of a statistically robust and parsimonious model.

This structured methodology minimizes the risk of both underfitting and overfitting, resulting in a curve that accurately reflects the data's underlying trend. Ultimately, the ability to derive and apply the final mathematical equation enables precise predictive analysis, turning raw data relationships into powerful, actionable insights across various research and industry domains.

How to Use seq Function in R