Curve Fitting with R: A Practical Guide to Regression Analysis

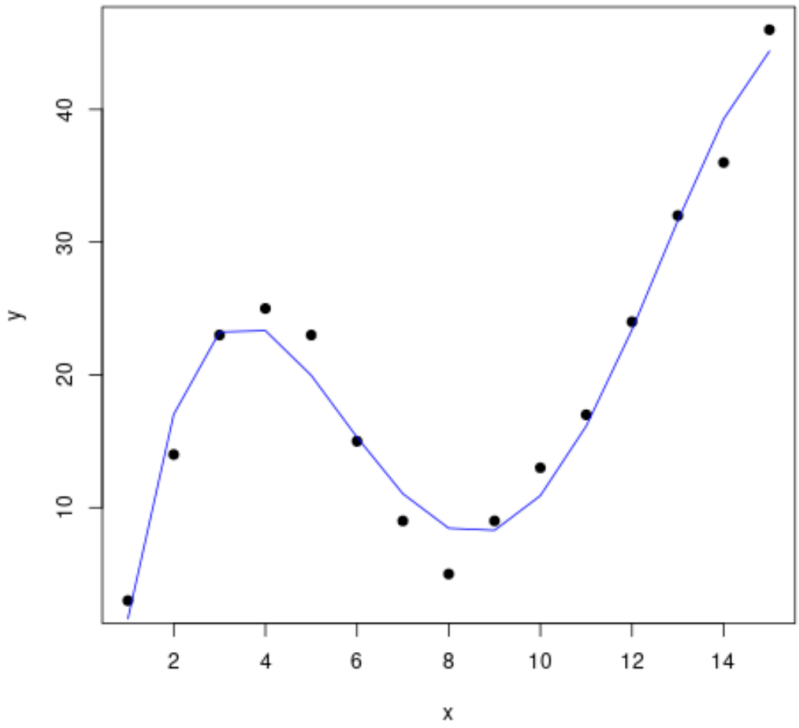
Authored by

Mohammed looti

November 5, 2025

In the realm of data analysis and statistical modeling, a fundamental requirement is the ability to determine the precise mathematical formula that governs the relationship between observed variables. This critical technique is known as curve fitting, a process indispensable for accurate prediction, sophisticated forecasting, and deep conceptual understanding of complex physical or social phenomena.

The core objective of curve fitting is to select an appropriate function--frequently a polynomial--whose graphical representation aligns as closely as possible with the observed set of data points. This article provides a comprehensive, step-by-step methodology for executing various curve fits in the statistical programming environment, R. We will utilize the powerful built-in functions, specifically **lm()** and the specialized poly() function, and introduce a rigorous statistical metric to objectively identify the optimal model.



The Role of Polynomial Regression in Curve Fitting

Curve fitting necessitates the development of a statistical model that accurately traces the underlying pattern evident in a scatterplot of the data. Because real-world observations rarely adhere perfectly to a simple straight line, practitioners often rely on **non-linear regression** techniques. Among these, polynomial regression is paramount, as it is highly effective at capturing the curvilinear relationships, bends, and inflection points observed between variables.

This method achieves flexibility by incorporating powers of the independent or predictor variable

(e.g., x, x^2, x^3, and so on) into the linear model framework. While this flexibility is crucial for high-quality fitting, it introduces a significant modeling challenge: selecting the correct degree for the polynomial. A model with a degree that is too low will result in high bias and poor fit (underfitting), whereas a degree that is excessively high risks capturing noise instead of signal, leading to the major statistical issue of **overfitting**.

Step 1: Preparing and Visualizing the Dataset

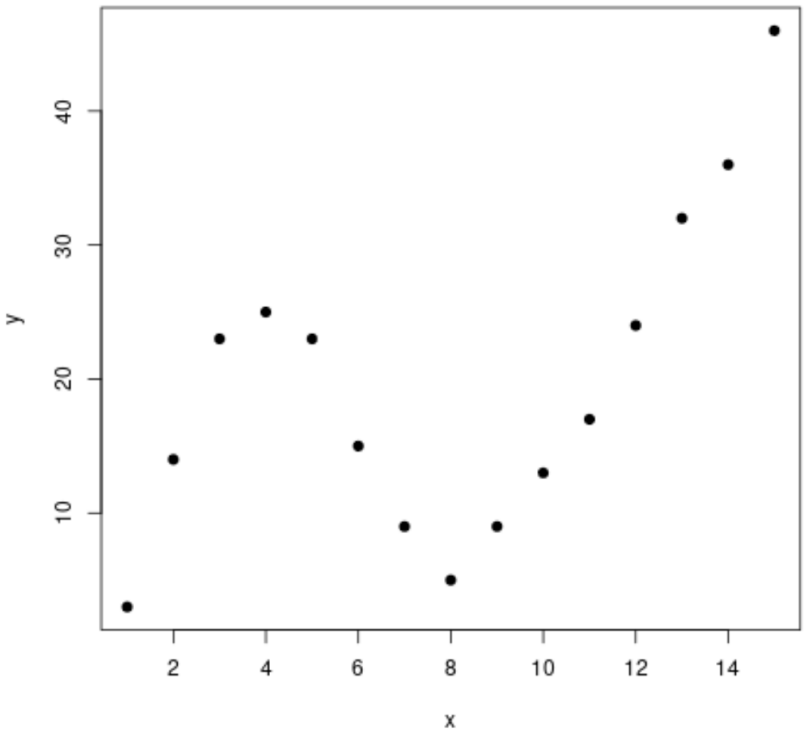
The modeling process must always begin with robust data preparation and an exploratory visual analysis of the dataset. Generating a scatterplot provides essential, immediate insight into the distribution and nature of the relationship, allowing the analyst to form an initial hypothesis about whether a simple linear model or a more complex polynomial structure will be required to describe the data effectively.

For demonstration purposes, we will first construct a synthetic data frame named `df`, comprising fifteen observations. Following the creation of this structure, we immediately generate the corresponding scatterplot to clearly visualize the relationship between the input variable (X) and the output variable (Y). This initial step dictates the subsequent modeling choices.

```
#create data frame
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
plot(df$x, df$y, pch=19, xlab='x', ylab='y')
```

The resulting graphical output clearly illustrates a pronounced non-linear trend--the data does not follow a straight path. This visual evidence confirms our initial suspicion that a higher-order polynomial model is necessary to achieve a statistically accurate and visually smooth curve fitting result.



Step 2: Fitting Multiple Polynomial Models for Comparison

Given the non-linear structure identified in the visualization stage, the next logical step is to fit a range of polynomial regression models. By testing models of varying complexity--specifically, ranging from degree 1 (a simple linear model) up to degree 5--we can systematically evaluate which degree best captures the variance in our data without overcomplicating the model structure. This comparative approach is essential for robust statistical practice.

In R, we leverage the fundamental **lm()** function (linear model) for this task. Crucially, we use the `poly(x, degree, raw=TRUE)` structure within **lm()** to define the polynomial terms. The argument `raw=TRUE` is vital; it ensures that the raw polynomial terms (x, x^2, etc.) are used, which significantly simplifies the subsequent extraction and interpretation of the final regression equation coefficients. We fit five distinct models, `fit1` through `fit5`, representing degrees 1 through 5, respectively.

After fitting the models, we overlay the predicted curves from each fit onto the original scatterplot. This provides an immediate, intuitive visual comparison of how well each model adheres to the observed data points, helping us narrow down the candidate models before applying formal statistical evaluation.

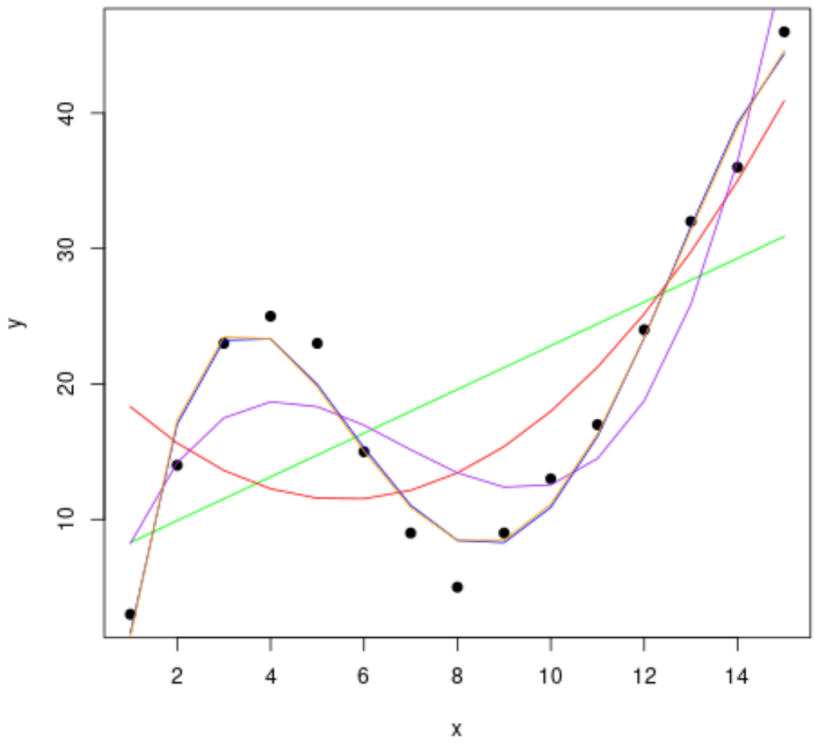
#fit polynomial regression models up to degree 5
fit1 <- lm(y~x, data=df)
fit2 <- lm(y~poly(x,2,raw=TRUE), data=df)

```
fit3 <- lm(y~poly(x,3,raw=TRUE), data=df)
fit4 <- lm(y~poly(x,4,raw=TRUE), data=df)
fit5 <- lm(y~poly(x,5,raw=TRUE), data=df)

#create a scatterplot of x vs. y
plot(df$x, df$y, pch=19, xlab='x', ylab='y')

#define x-axis values
x_axis <- seq(1, 15, length=15)

#add curve of each model to plot
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')
```



Selecting the Optimal Fit Using Adjusted R-squared

While visual comparison offers initial guidance, selecting the statistically superior model requires a rigorous quantitative metric. To achieve this, we rely on the Adjusted R-squared value for each

fitted model. This metric is crucial because it not only assesses the percentage of variance in the response variable (Y) explained by the predictor variable(s) (X), but it also applies a penalty for the inclusion of unnecessary predictors.

The inherent design of the Adjusted R-squared actively discourages the selection of overly complex models that contribute little explanatory power, thereby serving as a vital guard against overfitting. A higher value indicates a better balance between goodness of fit and model parsimony. We calculate and compare this metric across all five polynomial fits using the `summary()` function in R, specifically targeting the `adj.r.squared` component.

#calculated adjusted R-squared of each 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

The statistical output clearly reveals that the fourth-degree polynomial model, designated as `fit4`, yields the highest Adjusted R-squared score at **0.9590276**. Although the fifth-degree model (`fit5`) is structurally more complex, its resulting score (0.9549709) is marginally lower. This quantitative evidence confirms that the fourth-degree curve represents the optimal balance of complexity and predictive accuracy for this specific experimental dataset.

Step 3: Visualizing the Optimal Curve

Once the statistically superior model is identified using the Adjusted R-squared criterion, the final analytical step is to visualize this specific curve in isolation against the original data points. This process serves as a final, definitive confirmation that the model providing the best statistical performance also results in a visually intuitive, smooth, and highly accurate representation of the underlying data trend.

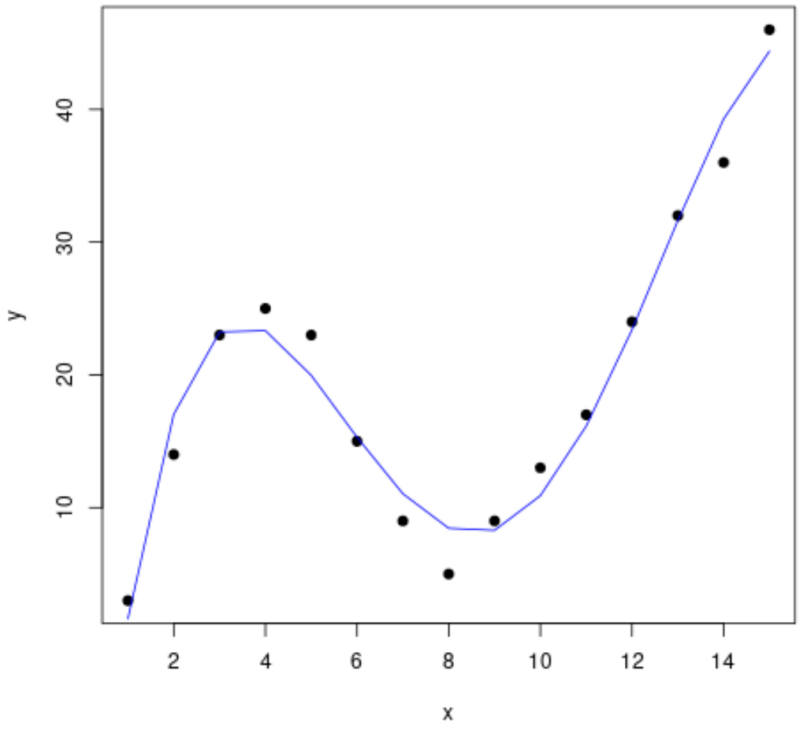
We regenerate the scatterplot, this time adding only the predicted line derived from the fourth-degree polynomial model (`fit4`). The resulting visualization powerfully demonstrates the highly accurate fit achieved, confirming that our methodological approach--combining visual inspection,

systematic modeling, and quantitative metric selection--has successfully identified the ideal curve fitting solution.

#create a scatterplot of x vs. y
plot(dfx, dfy, pch=19, xlab='x', ylab='y')

#define x-axis values
x_axis <- seq(1, 15, length=15)

#add curve of fourth-degree polynomial model
lines(x_axis, predict(fit4, data.frame(x=x_axis)), col='blue')



Extracting the Final Regression Equation and Prediction

The practical utility of curve fitting culminates in the derivation of the explicit mathematical equation, which allows for precise predictions outside the observed dataset. To obtain the necessary coefficients for our optimal model (`fit4`), we execute the **summary()** function on the model object. The resulting coefficients table provides the estimated values for the intercept and for every polynomial term included in the model (x, x^2, x^3, and x?).

The output below details the coefficients, standard errors, and significance tests for the fourth-degree model. These estimates are the building blocks of the final predictive equation:

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 coefficient estimates into the standard polynomial form, we construct the definitive equation for the fitted curve:

y = -0.0192x4 + 0.7081x3 - 8.3649x2 + 35.823x - 26.516

This precise equation allows us to accurately predict the value of the response variable (y) for any input value of the predictor variable (x). For instance, if we input the value $x = 4$ into the formula, the predicted value for y is calculated to be **23.34**, demonstrating the predictive power derived from effective statistical modeling.

y = -0.0192(4)4 + 0.7081(4)3 - 8.3649(4)2 + 35.823(4) - 26.516 = 23.34

Additional Resources for R Programming

To further advance your skills in statistical modeling and proficiency in the R programming environment, we recommend consulting the following related guides and documentation:

How to Use seq Function in R