How to Easily Extract P-Values from Linear Regression in R

Authored by

stats writer

November 21, 2025

The process of performing linear regression in R relies heavily on the built-in function `lm()`. While fitting the model is straightforward, the subsequent step often involves assessing the model's performance and determining the influence of individual predictors. This assessment critically depends on analyzing the associated P-values, which serve as crucial indicators of statistical significance. Understanding how to precisely extract these values--both the overall model P-value and the P-values for each individual coefficient--is fundamental for proper statistical reporting and model interpretation.

The standard practice for viewing these statistical metrics is by passing the fitted `lm` object to the `summary()` function. This output delivers a comprehensive table detailing the estimates, standard errors, t-statistics, and the corresponding P-values for every independent variable included in the model. These extracted P-values ultimately determine whether we can reject the null hypothesis, thereby confirming if a particular predictor has a statistically significant relationship with the dependent variable.

Understanding the Role of P-Values in Model Evaluation

In the context of regression analysis, P-values fulfill two primary roles: evaluating the overall fit of the model and assessing the contribution of individual predictors. The overall model P-value, derived from the F-statistic, tests the null hypothesis that all regression coefficients (excluding the intercept) are zero. If this P-value is below the chosen significance level (commonly 0.05), we conclude that the model, as a whole, is statistically significant and explains a meaningful amount of variance in the response variable.

Conversely, the individual P-values associated with each predictor test the null hypothesis that that specific predictor's coefficient is zero, meaning the variable has no linear relationship with the response when controlling for the other variables in the model. A low P-value here indicates that the variable is a significant contributor to the model. Because the `summary()` output provides a large amount of information, specialized techniques are often required to isolate only the P-values for use in automated pipelines, reports, or subsequent calculations.

There are two distinct methods we employ in R to efficiently extract these crucial metrics, depending on whether we need the global significance metric or the specific values for each regressor. Both methods leverage the structure of the data object returned by the `summary()` function, but they target different components of that complex structure.

Method 1: Extracting the Overall P-Value of the Regression Model

The overall P-value of the regression model is derived from the global F-test statistic. This statistic measures the ratio of explained variance to unexplained variance. While the `summary()` output

displays this value at the bottom, accessing it programmatically requires navigating the structured list object that the `summary()` function returns. Specifically, the information needed for this calculation is stored within the `fstatistic` element of the summary object.

To calculate the P-value from the F-statistic, we utilize the cumulative distribution function for the F-distribution, which is implemented in R as the `pf()` function. This function requires three primary inputs: the calculated F-statistic value, the numerator degrees of freedom, and the denominator degrees of freedom. By setting the argument `lower.tail=FALSE`, we calculate the area in the upper tail, which corresponds precisely to the desired P-value.

Due to the complexity of accessing and calculating this overall P-value directly, it is highly recommended to encapsulate this logic within a custom R function. This approach not only cleans up the code but also makes the extraction process repeatable and robust across different linear models fitted during an analysis session.

Implementing a Custom Function for Overall P-Value Extraction

The following custom function, named `overall_p`, is designed to take any fitted `lm` model object as input, calculate the overall P-value using the F-statistic components, and return a clean, scalar numeric value. This function abstracts away the need for manual navigation through the summary object every time this metric is required.

```
#define function to extract overall p-value of model
overall_p <- function(my_model) {
f <- summary(my_model)$fstatistic
p <- pf(f,f,f,lower.tail=F)
attributes(p) <- NULL
return(p)
}

#extract overall p-value of model
overall_p(model)
```

This implementation ensures that the output is a standard numeric vector, free of the auxiliary attributes that sometimes accompany statistical calculations in R, making it ideal for integration into larger scripts or automated reporting tools. The use of `summary(my_model)$fstatistic` specifically targets the necessary components for the F-test calculation.

Method 2: Extracting Individual P-Values for Regression Coefficients

The individual P-values correspond to the statistical tests performed on each regression coefficient (i.e., the slope associated with each predictor variable). These P-values are critical for determining which independent variables contribute significantly to the predictive power of the model. When viewing the `summary(model)` output, these values are listed in the final column of the "Coefficients" table.

Programmatically accessing these values is much simpler than calculating the overall P-value. The `summary()` function returns a list object, within which the coefficients table is stored as a matrix under the element named `coefficients`. This matrix typically has four columns: Estimate, Standard Error, t value, and finally, the P-value (Pr(>|t|)).

To isolate just the P-values, we can use standard matrix indexing in R. Since the P-values are consistently located in the fourth column of the `coefficients` matrix, we can select the entire matrix and then specify that we only want the values from the fourth column. This method is both efficient and highly reliable for extracting the P-values associated with the intercept and all predictor variables.

summary(model)$coefficients

The following comprehensive example demonstrates how to apply both of these extraction methods in a real-world scenario, ensuring clarity and practical application of these techniques.

Example: Extracting P-Values from `lm()` in R

Setting Up the Linear Regression Model

To provide a clear demonstration, we will first create a sample data frame and then fit a multiple linear regression model. Suppose we are analyzing factors influencing a player's rating based on their points scored, assists, and rebounds. The dependent variable is `rating`, and the independent variables are `points`, `assists`, and `rebounds`.

The process begins by structuring the data appropriately within an R data frame. Following data creation, we utilize the `lm()` function to define the linear relationship, specifying the formula `rating ~ points + assists + rebounds` and the source data frame `df`. This step generates the fitted model object, which serves as the input for all subsequent P-value extraction steps.

#create data frame
df <- data.frame(rating=c(67, 75, 79, 85, 90, 96, 97),

points=c(8, 12, 16, 15, 22, 28, 24),
assists=c(4, 6, 6, 5, 3, 8, 7),
rebounds=c(1, 4, 3, 3, 2, 6, 7))

#fit multiple linear regression model
model <- lm(rating ~ points + assists + rebounds, data=df)

Reviewing the Comprehensive Model Summary

Before extracting specific numeric values, it is always beneficial to view the complete output generated by the `summary()` function. This output provides context regarding the model fit, including residuals, coefficients, R-squared values, and the overall F-statistic and its corresponding P-value. This step confirms that the model was fitted correctly and allows for visual verification of the values we intend to extract programmatically.

The coefficient table is the most critical part of this output, detailing the estimates and the results of the t-tests for each predictor. Pay particular attention to the final column, labeled `Pr(>|t|)`, as these are the individual P-values we aim to isolate. Furthermore, note the information at the very bottom, which contains the F-statistic and the overall model P-value.

#view model summary
summary(model)

Call:
lm(formula = rating ~ points + assists + rebounds, data = df)

Residuals:
1 2 3 4 5 6 7
-1.5902 -1.7181 0.2413 4.8597 -1.0201 -0.6082 -0.1644

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 66.4355 6.6932 9.926 0.00218 **
points 1.2152 0.2788 4.359 0.02232 *
assists -2.5968 1.6263 -1.597 0.20860
rebounds 2.8202 1.6118 1.750 0.17847

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

Residual standard error: 3.193 on 3 degrees of freedom
Multiple R-squared: 0.9589, Adjusted R-squared: 0.9179

F-statistic: 23.35 on 3 and 3 DF, p-value: 0.01396

Programmatic Extraction of the Overall P-Value

Observing the output above, the overall P-value for the regression model is explicitly stated at the bottom as `0.01396`. Since this value is less than 0.05, we conclude that the model significantly predicts the player rating. While manual reading is possible, using the custom function defined earlier allows us to automate this extraction process seamlessly. This is particularly valuable when running simulations or analyzing hundreds of models simultaneously.

When we apply the `overall_p` function to our fitted `model` object, the function accesses the F-statistic components and uses the `pf()` function to calculate the precise P-value. This ensures that the extracted value is accurate and ready for numerical comparison or storage.

```
#define function to extract overall p-value of model
overall_p <- function(my_model) {
f <- summary(my_model)$fstatistic
p <- pf(f,f,f,lower.tail=F)
attributes(p) <- NULL
return(p)
}
```

```
#extract overall p-value of model
overall_p(model)
```

0.01395572

As confirmed by the output, the function successfully returns a P-value of approximately 0.01396, precisely matching the value observed in the detailed model summary. This demonstrates the robustness and utility of defining a dedicated function for this specific extraction task.

Extracting Individual Coefficient P-Values

The next critical step is isolating the P-values for the individual coefficients--Intercept, points, assists, and rebounds. These P-values determine which predictors are considered statistically significant contributors to the model when controlling for the effects of the other variables. We use matrix indexing on the `coefficients` element of the summary object to achieve this.

The code `summary(model)$coefficients` instructs R to retrieve the matrix of coefficients and

select all rows (indicated by the empty space before the comma) but only the fourth column (`4`), which contains the P-values (Pr(>|t|)). This returns a named numeric vector, where the names correspond to the predictors.

Analyzing the resulting vector shows that `points` (P-value = 0.022) and the `Intercept` (P-value = 0.002) are statistically significant at the 0.05 level, while `assists` (P-value = 0.208) and `rebounds` (P-value = 0.178) are not. This isolation technique is essential for targeted analysis and reporting.

#extract p-values for individual regression coefficients in model
summary(model)$coefficients

(Intercept) points assists rebounds
0.002175313 0.022315418 0.208600183 0.178471275

This syntax provides a quick and direct method to obtain the precise numeric P-values without needing to parse the full summary text output manually.