A Comprehensive Guide to Comparing Regression Models in R Using the mtable() Function

Authored by

Mohammed looti

November 12, 2025

In the demanding landscape of R statistical analysis, practitioners routinely face the task of estimating and comparing the outcomes from multiple regression analysis models simultaneously. Whether exploring different sets of predictor variables or comparing methodologies on a single dataset, fitting several models is standard procedure. However, retrieving and comparing the resulting coefficients, standard errors, and summary statistics using R's default output functions often results in fragmented, difficult-to-interpret views. This inherent challenge demands a specialized tool that can consolidate complex statistical results into a concise, comparative format.

The **mtable()** function provides an elegant and highly effective solution to this comparative challenge. This function, which is sourced from the robust memisc package, has been meticulously engineered to generate clean, professional-quality tables. These tables unify the core results of multiple statistical models, presenting them in a structured, side-by-side format. This capability is absolutely essential for data scientists and researchers who require transparent reporting and efficient model assessment, allowing them to quickly evaluate how variations in model specifications affect parameter estimates and overall goodness-of-fit.

To fully leverage this powerful utility, it is crucial to first establish a strong understanding of the fundamental syntax and customizable features that position **mtable()** as an indispensable tool for comparing statistical output within the R environment. We will explore its key arguments before demonstrating its use with a classic example.

Mastering the mtable() Function Syntax and Parameters

The **mtable()** function is designed for high flexibility, capable of accepting various model objects generated by standard R procedures, such as **lm()** (for linear models) or **glm()** (for generalized linear models). While its basic structure is straightforward, the optional parameters provide extensive control over the content, formatting, and appearance of the final output table, ensuring it meets publication standards.

The core syntax for invoking the function clearly illustrates its principal components:

mtable(..., summary.stats=TRUE, coef.style, ...)

Each argument plays a distinct and critical role in defining the structure and detailed presentation of the comparison table:

...: This represents the primary input--a named list of the statistical model objects you wish to compare (e.g., `"Model A"=model_a, "Model B"=model_b`). Users can supply any number of models, and **mtable()** will automatically align the resulting coefficient values for common predictors across all included models.
summary.stats: This is a logical argument that defaults to **TRUE**. When set to true, it ensures

that aggregate metrics essential for model evaluation--such as R-squared values, F-statistics, or residual standard errors--are included in a clean block at the bottom of the output table. Keeping this setting active is essential for a comprehensive comparative analysis.
coef.style: This critical argument accepts a character string that governs the exact formatting style for the estimated coefficients and their accompanying measures of precision (standard errors or p-values). Options are available to display standard errors in parentheses below the estimate or to present p-values alongside the estimates, allowing the table to conform to specific journalistic or academic reporting guidelines.

The resulting output from **mtable()** is a meticulously structured table that not only aligns predictor estimates across different models but also clearly annotates these estimates with measures of precision and indicators of statistical significance. This single, consolidated view drastically reduces the manual effort typically required to compile and interpret complex statistical results.

Preparing the Data: Utilizing the Classic mtcars Dataset

To effectively demonstrate the capability and convenience of **mtable()**, we will utilize the universally recognized **mtcars** dataset, which is conveniently pre-loaded into R. This dataset is a rich source of information, containing 11 different attributes for 32 automobiles manufactured in the 1970s, making it perfectly suited for developing and comparing multivariate regression analysis models.

Our primary objective is to build models that predict miles per gallon (**mpg**) using two distinct combinations of vehicle characteristics. Before fitting any models, however, a preliminary inspection of the data is necessary to understand the variables we will employ. We use the standard **head()** function to display the initial observations from the dataset:

#view head of mtcars dataset
head(mtcars)

mpg cyl disp hp drat wt qsec vs am gear carb
Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4
Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4
Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1
Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1
Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2
Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

The **mtcars** dataset provides measurements such as engine displacement (**disp**), gross

horsepower (**hp**), number of cylinders (**cyl**), and the number of carburetors (**carb**). We are interested in investigating how varying subsets of these influential variables affect our ability to predict **mpg**. We will proceed by fitting one model incorporating a relatively broad set of predictors and a second, more parsimonious model designed for simpler interpretation.

Fitting Two Distinct Regression Models

The next step involves defining and fitting two separate models using Ordinary Least Squares (OLS) regression via the standard **lm()** function in R. Both models share the same dependent variable, **mpg**, but they are fundamentally distinct in their inclusion of independent, or predictor, variables. Model 1 is established as a comprehensive model utilizing four predictors, while Model 2 simplifies the structure to include only two, allowing us to directly observe the impact of omitting predictors.

The following R code executes the fitting of these two distinct regression analysis specifications:

#fit first regression model
model1 <- lm(mpg ~ disp + carb + hp + cyl, data = mtcars)

#fit second regression model
model2 <- lm(mpg ~ disp + carb, data = mtcars)

With both models successfully fitted, the true power and efficiency of the **mtable()** function, sourced from the memisc package, can now be realized. We must first load the necessary library, and then we call **mtable()**, passing both model objects, Model 1 and Model 2, along with clear, descriptive names for labeling the columns in the output.

library(memisc)

#create table to compare coefficient values from both regression models
mtable("Model 1"=model1,"Model 2"=model2)

Calls:
Model 1: lm(formula = mpg ~ disp + carb + hp + cyl, data = mtcars)
Model 2: lm(formula = mpg ~ disp + carb, data = mtcars)

==========
Model 1 Model 2

(Intercept) 34.022*** 31.153***
(2.523) (1.264)

disp -0.027* -0.036***
(0.011) (0.005)
carb -0.927 -0.956*
(0.579) (0.359)
hp 0.009
(0.021)
cyl -1.049
(0.784)

R-squared 0.788 0.774
N 32 32
==========
Significance: *** = p < 0.001;
** = p < 0.01;
* = p < 0.05

The resulting output is a high-quality, text-based table that efficiently organizes the complex results. Each column neatly represents a model, and the rows correspond to the intercept and the specific predictor coefficients. This integrated structure facilitates the immediate and intuitive comparison of estimates across various model specifications, which would otherwise require tedious manual transcription.

Drawing Conclusions from the Comparative Output

The table generated by **mtable()** encapsulates a significant amount of information necessary for thoroughly assessing and comparing the performance of Model 1 against Model 2. A proper interpretation of each section is key to formulating valid statistical conclusions and making informed choices about model preference.

Focusing initially on the estimated coefficients, we can clearly observe how the inclusion or exclusion of certain predictors causes shifts in the estimates for the remaining variables:

Intercept Value Comparison: Model 1 exhibits an intercept of **34.022**, which is notably higher than the **31.153** intercept found in Model 2. The values provided in parentheses directly below these estimates represent the standard error (**2.523** for Model 1 and **1.264** for Model 2). It is important to note that both intercept values are highly statistically significant (indicated by ***).
Predictor Comparison (disp and carb): The estimated impacts for shared variables, such as **disp** and **carb**, demonstrate subtle but important shifts when moving from the four-predictor Model 1 to the two-predictor Model 2. For instance, the coefficient for **disp** changes from **-0.027** in Model 1 to **-0.036** in Model 2. This difference suggests that the estimated

influence of engine displacement on MPG is altered when the effects of **hp** and **cyl** are removed from the model specification.
Significance Levels: The asterisks placed next to the estimates indicate their level of statistical significance, as explicitly defined at the table's footer. In the more complex Model 1, **disp** is significant at the p < 0.05 level (*), whereas in Model 2, it achieves a higher significance level (p < 0.001, denoted by ***). Furthermore, variables exclusive to Model 1, such as **hp** and **cyl**, show very low significance within the context of that broader model.

Below the coefficient estimates, the table provides crucial summary statistics that quantify the overall fit and characteristics of each model:

R-squared (Coefficient of Determination): This metric quantifies the proportion of the variance in the response variable (**mpg**) that can be explained by the predictor variables included in the model. Model 1, with an R-squared of **0.788**, explains slightly more variation than Model 2 (**0.774**). This marginal increase suggests that the inclusion of the additional variables (**hp** and **cyl**) contributes only slightly to the overall explanatory power, which might lead a researcher to prefer the simpler Model 2 for reasons of parsimony.
N (Sample Size): This row confirms the total number of observations used to fit each model. In this specific comparison, **N=32** for both models, verifying that the entire **mtcars** dataset was utilized for fitting.

By meticulously organizing this complex statistical output into a single, comparative table, **mtable()** greatly simplifies informed decision-making regarding model selection and hypothesis testing in R, ensuring that statistical results are both accessible and easily digestible for any audience.

Expanding R Proficiency with Additional Resources

The ability to effectively compare and professionally report statistical models is a foundational skill for any professional data analyst or researcher. The **mtable()** function, along with the broader utility provided by the memisc package, establishes a robust framework for generating publication-quality comparisons. For those committed to expanding their proficiency in R and advanced data analysis techniques, the following curated resources and tutorials offer guidance on performing other common and essential statistical tasks:

Featured Posts

[5 Statistical Biases to Avoid]

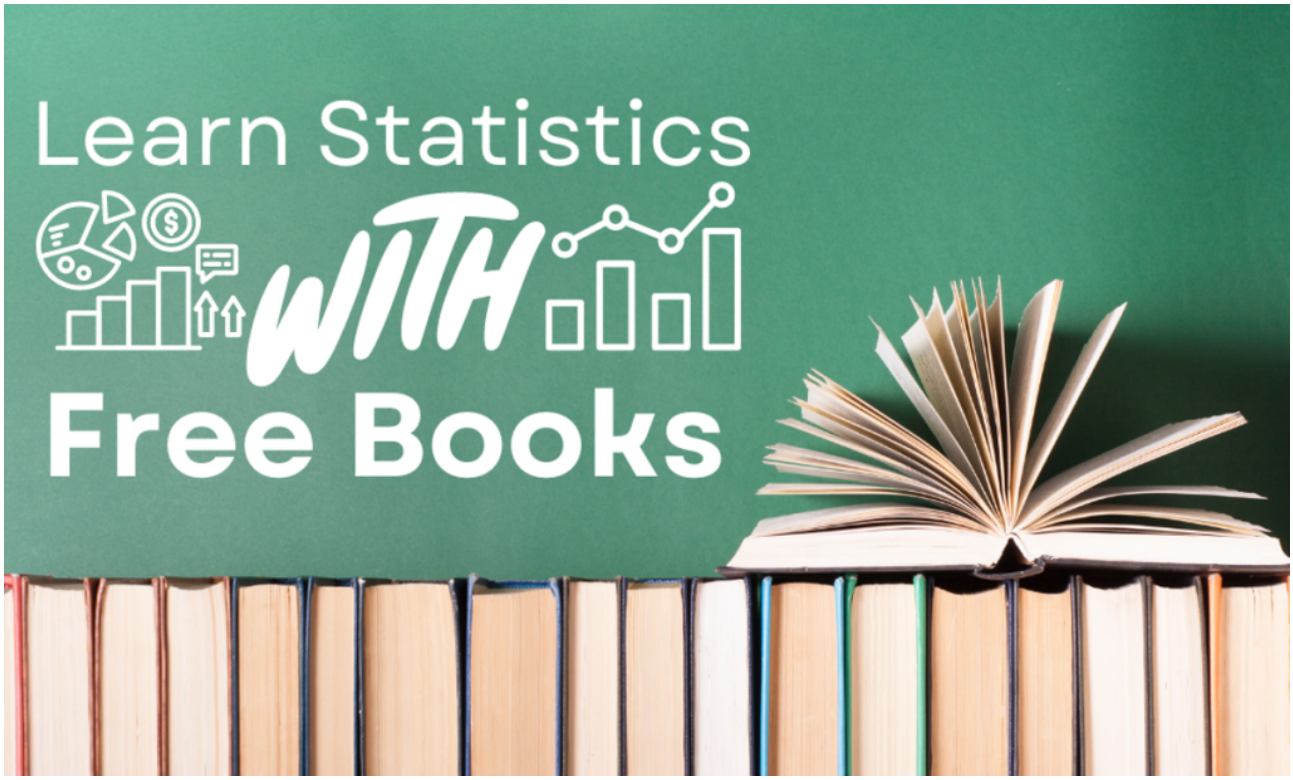
April 25, 2024

[5 Free Statistics Courses for Beginners]

April 19, 2024

5 MIT Statistics Courses That Are Free

April 18, 2024



5 Free Books to Learn Statistics

April 18, 2024

How to Use the info() Method in Pandas

April 12, 2024

How to Use pct_change() in Pandas

April 12, 2024