Partial Least Squares Regression in R: A Step-by-Step Guide to Handling Multicollinearity

Authored by

Mohammed looti

November 6, 2025

A persistent and significant challenge in statistical modeling and regression analysis is dealing with **multicollinearity**. This condition arises when two or more predictor variables within a chosen dataset exhibit high linear correlation with one another. When predictors are tightly linked, the model struggles to isolate the unique effect of each variable on the outcome.

The severe presence of multicollinearity often results in extremely unstable and highly variable regression coefficients. While a model trained under these conditions might achieve excellent performance on the specific training data, it frequently suffers from poor generalization capability when introduced to new, unseen observations. This failure to generalize effectively is a classic sign of overfitting, which compromises the reliability and practical utility of the resulting statistical model.

To successfully address these complications, analysts frequently rely on specialized dimension reduction techniques. Among the most powerful is Partial Least Squares (PLS) regression. PLS is designed not only to reduce the number of variables but also to identify underlying latent variables that effectively explain the covariance present in both the predictor matrix and the response variable, ensuring a more robust and parsimonious model structure.

The Core Methodology of Partial Least Squares Regression

The operational procedure for implementing Partial Least Squares involves a sequence of specific steps. These steps are meticulously engineered to maximize the correlation structure between the set of predictors and the response variables while simultaneously achieving effective dimensionality reduction. This makes PLS particularly valuable in fields like chemometrics and economics where data redundancy is common.

Data Preparation: It is standard practice to standardize both the **predictor** and **response variables**. This crucial preprocessing step involves centering the variables to a mean of zero and scaling them to a standard deviation of one, ensuring that variables measured in different units contribute equitably to the component derivation process.
Component Extraction: The algorithm calculates *M* linear combinations, often termed "PLS components" or latent vectors, from the original *p* predictor variables. Unlike Principal Components Analysis, these components are specifically optimized to capture the maximum variance that is correlated with the response variable, rather than just the maximum total variance in the predictors.
Model Fitting: A standard linear regression model is subsequently fitted. This model uses the derived PLS components as the new, orthogonal set of predictors, employing the powerful statistical method of **least squares** estimation to determine the regression coefficients.
Optimization and Selection: Robust model optimization is performed, typically utilizing techniques such as k-fold cross-validation. The primary goal of this optimization is to empirically determine the optimal number of PLS components that must be retained to maximize predictive performance while avoiding model complexity.

This comprehensive guide provides a practical, step-by-step tutorial illustrating how to successfully implement and rigorously interpret Partial Least Squares regression. We will utilize the extensive statistical capabilities provided by the **R programming language**, focusing on reproducibility and clear interpretation of the results.

Step 1: Loading Essential R Packages for PLS

The most efficient and widely accepted method for performing Partial Least Squares analysis within the R environment involves leveraging the functionality contained within the powerful **pls** package. This package is recognized as the authoritative standard for implementing PLS methods, particularly within the domain of chemometrics and multivariate statistical analysis.

Before proceeding with model fitting, the necessary package must be installed and loaded into the current R session. The following code snippet details the commands required to prepare the environment. If the package has been previously installed on your system, you only need to execute the `library()` command.

 # Install the pls package (execute only if the package is not yet available)
install.packages("pls")

# Load the required pls package into the current session
library(pls)

Step 2: Fitting the Partial Least Squares Model in R

To provide a concrete and easily replicable example, we will utilize the well-known, built-in R dataset, **mtcars**. This dataset offers valuable comparative data, summarizing performance characteristics for 32 distinct models of automobiles, including metrics like miles per gallon, weight, and horsepower.

We begin by inspecting the structure of the data to confirm its layout and variable types. Displaying the initial six rows provides a quick overview of the variables that will be used in our predictive model.

 # Display the initial six rows of the mtcars dataset for inspection
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

Our objective is to fit a Partial Least Squares model aimed at predicting `hp` (horsepower), which we designate as our primary response variable. We select a set of five key mechanical specifications to act as the predictors for this analysis:

mpg (Miles per gallon)
disp (Engine displacement)
drat (Rear axle ratio)
wt (Vehicle weight)
qsec (1/4 mile time)

The subsequent R code demonstrates the application of the `plsr()` function to fit the model. It is essential to correctly set the two critical arguments that govern the PLS component extraction and validation process:

scale=TRUE: This parameter is absolutely critical in PLS, ensuring that all variables are standardized (mean zero, unit variance). This standardization prevents variables with naturally larger numerical ranges (like displacement) from disproportionately influencing the derived PLS components.
validation="CV": By setting this argument, we instruct the function to utilize k-fold cross-validation (CV) for robustly estimating the model's predictive error. The default setting employs 10 folds, which is typically sufficient for stable results. Alternatively, specifying `"LOOCV"` would implement the more computationally intensive leave-one-out cross-validation.

 # Set a seed to ensure the results are exactly reproducible across different runs
set.seed(1)

# Fit the PLS model, specifying scaling and cross-validation for validation
model <- plsr(hp~mpg+disp+drat+wt+qsec, data=mtcars, scale=TRUE, validation="CV")

Step 3: Determining the Optimal Number of PLS Components

A foundational task in PLS analysis is identifying the optimal number of components to retain. Retaining too few components sacrifices predictive power, while retaining too many risks incorporating noise and causing overfitting. This selection process is driven by minimizing the cross-validated error metric, specifically the **Root Mean Squared Error of Prediction (RMSEP)**. We analyze the model results using the `summary()` function.

 # Generate a summary of the model fitting results
summary(model)

Data: X dimension: 32 5
Y dimension: 32 1
Fit method: kernelpls
Number of components considered: 5

VALIDATION: RMSEP
Cross-validated using 10 random segments.
(Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps
CV 69.66 40.57 35.48 36.22 36.74 36.67
adjCV 69.66 40.41 35.12 35.80 36.27 36.20

TRAINING: % variance explained
1 comps 2 comps 3 comps 4 comps 5 comps
X 68.66 89.27 95.82 97.94 100.00
hp 71.84 81.74 82.00 82.02 82.03

The output provides two essential tables for diagnostic assessment. The first table, **VALIDATION: RMSEP**, shows the cross-validated error (CV RMSE), which we aim to minimize. By examining these results, we can observe the following critical performance points:

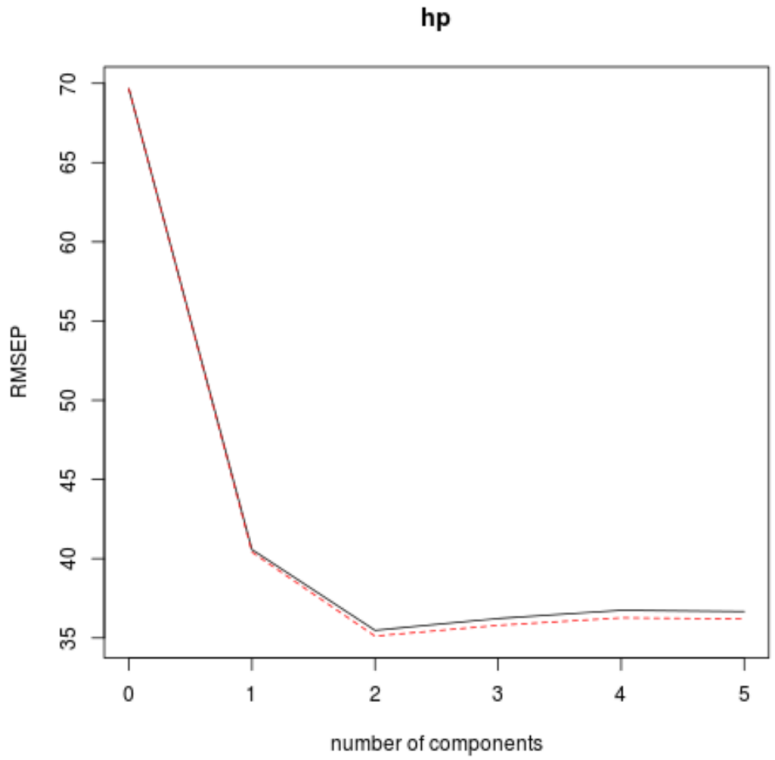
The baseline error (using only the intercept) is substantial at **69.66**.
The introduction of the first PLS component dramatically lowers the test RMSE to **40.57**.
Including the second PLS component achieves the minimum observed error of **35.48**.

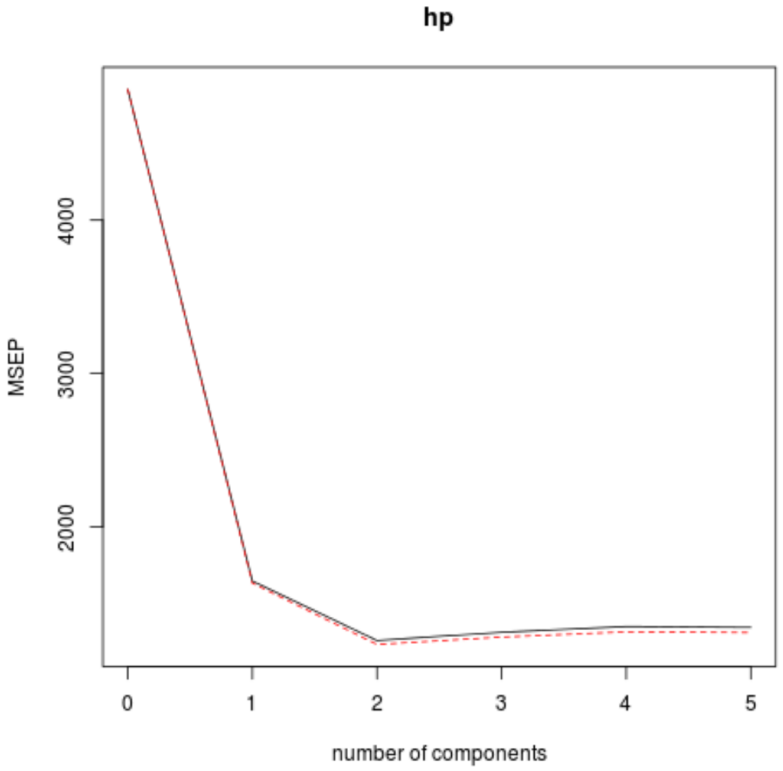
Crucially, we note that the RMSE begins a slight increase when we incorporate the third, fourth, and fifth components. This pattern strongly suggests that these additional components primarily model noise rather than meaningful signal. Therefore, based on the statistical goal of error minimization and adherence to the principle of parsimony, the optimal model specification requires only **two PLS components**.

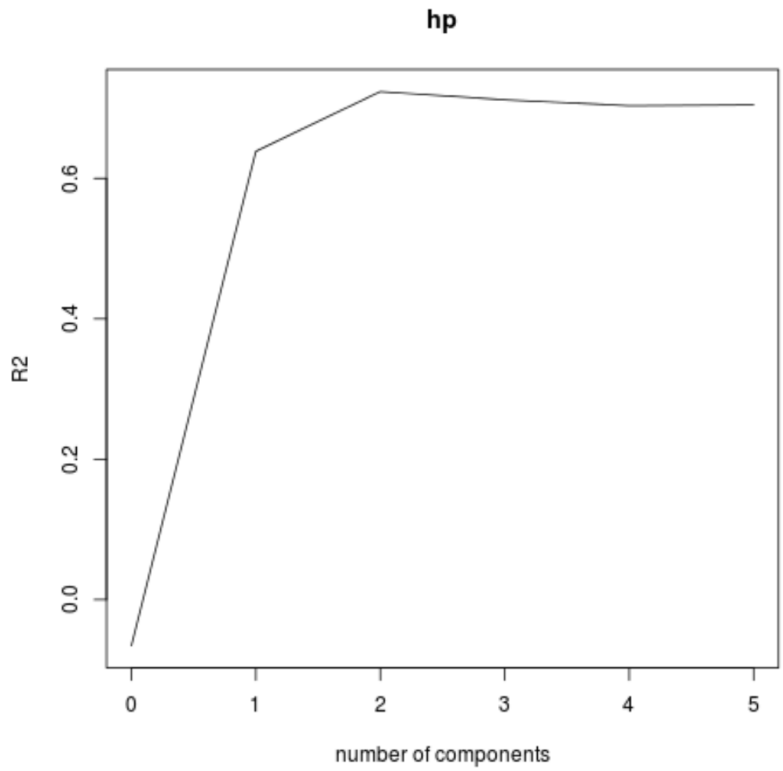
The second table, **TRAINING: % variance explained**, confirms the efficiency of the chosen components by reporting the cumulative percentage of variance accounted for in the response variable (`hp`). With the first PLS component alone, we account for **71.84%** of the variance in horsepower. By adding the second PLS component, the cumulative explained variance increases significantly to **81.74%**. Subsequent components offer only marginal improvements (e.g., component 3 adds a negligible 0.26%), further solidifying the choice of a two-component model as sufficient to capture the majority of the predictive structure.

For an intuitive and visual confirmation of this component selection process, we execute the `validationplot()` function. This command generates graphical representations showing how key predictive metrics--such as RMSEP, Mean Squared Error of Prediction (MSEP), and R-squared--evolve as the number of retained components increases.

 # Visualize the cross-validation metrics
validationplot(model)
validationplot(model, val.type="MSEP")
validationplot(model, val.type="R2")







As clearly demonstrated across all three generated visualizations, the predictive performance of

the model consistently improves up to the point where two PLS components are included. Beyond this threshold, the performance metrics either level off or begin to slightly worsen, providing strong graphical evidence that the selection of the two-component model is indeed optimal for this specific predictive task.

Step 4: Utilizing the Final Model for Generating Predictions

Having successfully identified the optimal model complexity (ncomp=2), the next logical step is to use this finalized PLS structure to generate predictions on an independent subset of data. This procedure is fundamental for rigorously evaluating the model's true ability to generalize beyond the data used for component derivation and selection.

To simulate a real-world scenario, we first segment the original **mtcars** dataset into distinct training and testing partitions. We will train the PLS model exclusively on the training data and then apply the optimized two-component model to predict the horsepower values within the reserved testing set.

 # Define training set (first 25 observations) and testing set (remaining observations)
train <- mtcars
y_test <- mtcars
test <- mtcars

# Re-fit the PLS model using only the training data
model <- plsr(hp~mpg+disp+drat+wt+qsec, data=train, scale=TRUE, validation="CV")

# Use the trained model to predict values on the test set, specifying ncomp=2
pcr_pred <- predict(model, test, ncomp=2)

# Calculate the test RMSE (Root Mean Squared Error)
sqrt(mean((pcr_pred - y_test)^2))

54.89609

The calculated test RMSE for our optimized two-component PLS model is **54.89609**. This metric quantifies the average magnitude of the errors between the horsepower values predicted by our model and the actual observed horsepower values within the dedicated testing set. Lower RMSE values indicate higher predictive precision.

For comparison, an equivalent Principal Components Regression (PCR) model, also utilizing two components, achieved a test RMSE of 56.86549 on the same split data. In the context of this specific dataset, the Partial Least Squares methodology demonstrated slightly superior predictive

accuracy compared to its PCR counterpart, reinforcing its utility in scenarios where predictors are highly correlated.

The complete, executable R code utilized throughout this detailed step-by-step analysis is conveniently available for download and reference on GitHub.