How to do a Linear Discriminant Analysis in R (Step-by-Step)?

Authored by

stats writer

December 19, 2025

Linear Discriminant Analysis (LDA) is a powerful statistical technique widely utilized for classification tasks. Its primary purpose is to identify a linear combination of features that characterizes or separates two or more classes of objects or events. When implementing LDA in the statistical environment of R, the process involves several key computational steps.

Initially, the dataset must be structured appropriately, typically as a data frame object. Once prepared, the core computation is handled by the specialized `lda()` function, which calculates the optimal linear discriminants. These results are then used to formulate the complete linear discriminant model. Subsequently, standard R functions like `summary()` help interpret the model performance and structure, while the crucial `predict()` function allows the model to classify new, unseen data points, projecting their categorization based on the learned discriminants.

Linear Discriminant Analysis is ideally applied when you have multiple predictor variables (features) and your objective is to classify a categorical response variable into two or more distinct groups. This comprehensive tutorial will guide you through a detailed, step-by-step practical example of implementing LDA using R, ensuring clarity and reproducibility.

We will use a classic dataset, `iris`, to demonstrate data preparation, model fitting, prediction, and visualization of the results.

Step 1: Load Required R Packages

The first prerequisite for running the LDA is ensuring that the necessary computational packages are installed and loaded into the R environment. For this specific task, we require two essential libraries: the MASS package, which contains the core `lda()` function, and `ggplot2`, which we will use later for powerful visualization of the model's results.

Execute the following R commands to load these libraries:

```
library(MASS)
library(ggplot2)
```

If you do not have these packages installed, you would typically run `install.packages("MASS")` and `install.packages("ggplot2")` beforehand.

Step 2: Access and Examine the Sample Data

To illustrate the application of LDA, we will utilize the universally recognized **iris** dataset, which is conveniently built into the R environment. This dataset is a classic benchmark in machine learning and statistics, containing measurements for 150 iris flowers across three species. The commands

below demonstrate how to make the dataset accessible and inspect its fundamental structure using `str()`.

 # Attach the iris dataset to the R search path for easy column access
attach(iris)

# View the internal structure of the dataset
str(iris)

'data.frame': 150 obs. of 5 variables:
$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
$ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
$ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
$ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
$ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 ...

The output confirms that the dataset comprises 150 total observations and 5 variables. Our objective is to construct an LDA model capable of classifying an iris specimen based on its morphological measurements (the predictor variables) into one of the known species (the response variable).

The **predictor variables** used to train the model are quantitative measurements:

Sepal.Length
Sepal.Width
Petal.Length
Petal.Width

The categorical **response variable** we aim to predict is *Species*, which includes three distinct classes:

setosa
versicolor
virginica

Step 3: Standardize the Feature Data

A crucial prerequisite for applying Linear Discriminant Analysis is the assumption of homogeneity of variance, meaning the covariance matrices of the groups must be equal. While this assumption is often violated in practice, scaling the predictor variables is a common and effective step to normalize the input features, thereby stabilizing the statistical properties used in the separation

calculation.

We standardize the data by transforming each variable so that it has a mean of zero (centering) and a standard deviation of one (scaling). This process, known as Z-score normalization, ensures that variables measured on different scales contribute equally to the distance metrics used by the LDA model. The built-in R function `scale()` performs this transformation efficiently on the first four columns (the predictors) of the `iris` dataset.

```
# Apply Z-score standardization to the first four columns (predictor variables)
iris <- scale(iris)
```

To confirm the scaling operation was successful, we use the `apply()` function to calculate the mean and standard deviation of the standardized columns. The resulting means should be extremely close to zero (due to floating-point arithmetic), and the standard deviations should equal exactly one.

```
# Calculate the mean of each standardized predictor variable
apply(iris, 2, mean)
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width
-4.484318e-16 2.034094e-16 -2.895326e-17 -3.663049e-17
```

```
# Calculate the standard deviation of each standardized predictor variable
apply(iris, 2, sd)
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width
1 1 1 1
```

Step 4: Partition the Data into Training and Testing Sets

For robust model evaluation, it is standard practice in machine learning to partition the dataset into two distinct subsets: a **training set**, used to estimate the model parameters, and a **testing set**, used to assess the model's generalization ability on unseen data. This prevents overfitting and provides a reliable estimate of classification accuracy.

We will allocate approximately 70% of the total observations to the training set and reserve the remaining 30% for testing. To ensure that our random sampling process is reproducible across different sessions, we utilize the `set.seed()` function before executing the sampling logic.

```
# Set a seed for reproducibility of random sampling
set.seed(1)
```

# Create a logical vector (sample) where TRUE represents the training set (70% probability)
sample <- sample(c(TRUE, FALSE), nrow(iris), replace=TRUE, prob=c(0.7,0.3))

# Subset the data to create the training set (TRUE indices)
train <- iris

# Subset the data to create the testing set (FALSE indices)
test <- iris

Step 5: Training and Interpreting the LDA Model

With the data standardized and partitioned, we now proceed to fit the LDA model. We utilize the `lda()` function provided by the **MASS** package. The formula `Species~.` instructs R to model the response variable `Species` using all other variables (the four measurements) present in the specified `train` dataset.

 # Fit the Linear Discriminant Analysis model using the training data
model <- lda(Species~., data=train)

Display the detailed model output
model

Call:
lda(Species ~ ., data = train)

Prior probabilities of groups:
setosa versicolor virginica
0.3207547 0.3207547 0.3584906

Group means:
Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa -1.0397484 0.8131654 -1.2891006 -1.2570316
versicolor 0.1820921 -0.6038909 0.3403524 0.2208153
virginica 0.9582674 -0.1919146 1.0389776 1.1229172

Coefficients of linear discriminants:
LD1 LD2
Sepal.Length 0.7922820 0.5294210
Sepal.Width 0.5710586 0.7130743
Petal.Length -4.0762061 -2.7305131
Petal.Width -2.0602181 2.6326229

Proportion of trace:
LD1 LD2
0.9921 0.0079

The output generated provides critical insight into how the model differentiates between the species. We must carefully examine each section to understand the classification criteria:

Prior probabilities of groups: These values reflect the baseline frequency of each species within the training dataset. For instance, the *virginica* species accounted for approximately 35.8% of the observations used for training the model.
Group means: This table displays the mean value for each predictor variable, calculated separately for each species. Since we scaled the data, these means are in standardized units, highlighting which species tend to have higher or lower measurements relative to the overall dataset average.
Coefficients of linear discriminants: These coefficients define the linear combinations of predictors that maximize the separation between the groups. They form the core decision rule of the LDA model. Specifically, **LD1** and **LD2** are the discriminant functions (canonical variates) calculated.

The discriminant functions LD1 and LD2 are constructed using these coefficients:

LD1 Equation: $0.792 times text{Sepal.Length} + 0.571 times text{Sepal.Width} - 4.076 times text{Petal.Length} - 2.060 times text{Petal.Width}$
LD2 Equation: $0.529 times text{Sepal.Length} + 0.713 times text{Sepal.Width} - 2.731 times text{Petal.Length} + 2.633 times text{Petal.Width}$

The **Proportion of trace** indicates the relative importance of each linear discriminant function in separating the classes. Here, LD1 accounts for 99.21% of the total separation, demonstrating that virtually all the discriminatory power is contained within the first discriminant function.

Step 6: Predicting Classifications and Assessing Performance

The true measure of a classification model's effectiveness lies in its ability to classify data it has never encountered. We leverage the trained LDA model to generate classifications for the observations in our designated `test` set using the `predict()` function.

 # Use the fitted LDA model to generate predictions on the reserved test data
predicted <- predict(model, test)

# Check the structure of the prediction output
names(predicted)

"class" "posterior" "x"

The output of the `predict()` function is a list containing three critical components for evaluating the classification:

class: This is the final predicted category (species) for each test observation.
posterior: This provides the posterior probability--the calculated likelihood that a specific observation belongs to each of the possible classes, given the discriminant functions.
x: These are the coordinates of the test data points projected onto the linear discriminant axes (LD1 and LD2).

We can inspect the first six predicted observations to understand the model's classifications and certainty levels:

 # Display the predicted class for the first six observations in the test set
head(predicted$class)

setosa setosa setosa setosa setosa setosa
Levels: setosa versicolor virginica

# Display the posterior probabilities for the first six observations
head(predicted$posterior)

setosa versicolor virginica
4 1 2.425563e-17 1.341984e-35
6 1 1.400976e-21 4.482684e-40
7 1 3.345770e-19 1.511748e-37
15 1 6.389105e-31 7.361660e-53
17 1 1.193282e-25 2.238696e-45
18 1 6.445594e-22 4.894053e-41

# Display the linear discriminants for the first six observations in the test set
head(predicted$x)

LD1 LD2
4 7.150360 -0.7177382
6 7.961538 1.4839408
7 7.504033 0.2731178
15 10.170378 1.9859027
17 8.885168 2.1026494
18 8.113443 0.7563902

The posterior probabilities demonstrate extremely high confidence (approaching 1) that the first six observations belong to the *setosa* class, with negligible probability assigned to the other classes.

Finally, we calculate the model's accuracy by comparing the predicted classes against the actual species labels in the test data:

 # Calculate the accuracy (proportion of correctly classified observations)
mean(predicted$class==test$Species)

1

The model achieved a perfect classification accuracy of **100%** on the test dataset. While this result is exceptional, it is important to note that the `iris` dataset is known for its high separability, and achieving perfect accuracy is less common in complex, real-world data scenarios. This outcome confirms that the LDA successfully identified highly effective linear boundaries for separating these three species.

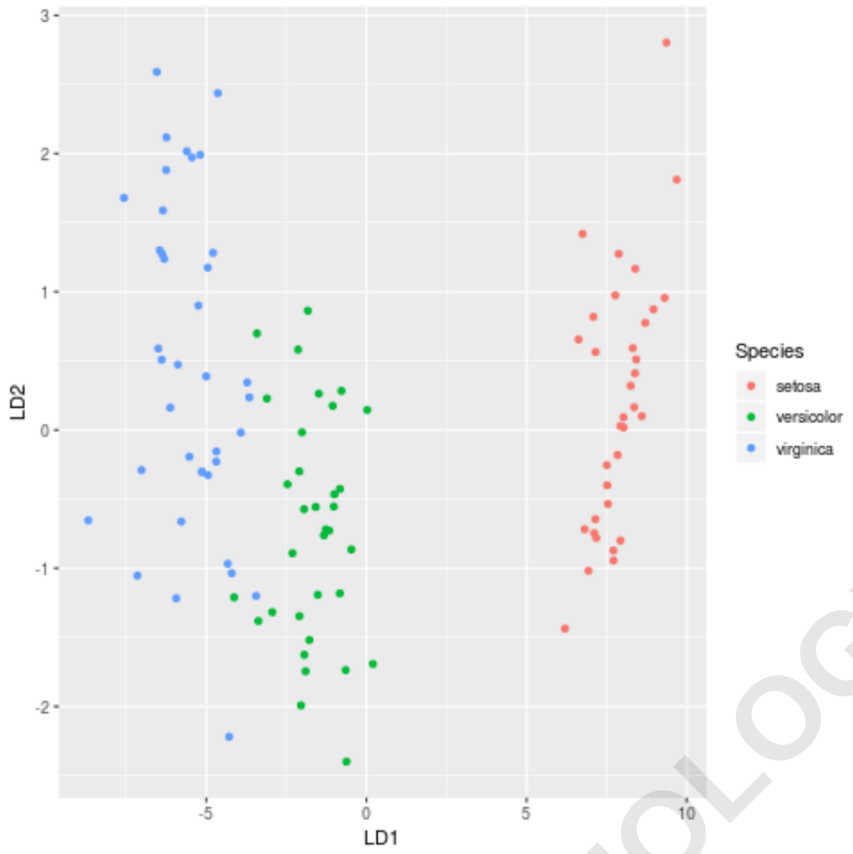
Step 7: Visualizing Group Separation using Linear Discriminants

One of the most valuable aspects of LDA is its ability to reduce dimensionality while maximizing class separation. By plotting the data points onto the calculated Linear Discriminant axes (LD1 and LD2), we can visually assess how distinct the groups are in this newly transformed space.

To generate the visualization, we first combine the training data with the computed discriminant scores (`predict(model)$x`). We then use the `ggplot2` package to create a scatter plot, mapping the LD1 scores to the X-axis and the LD2 scores to the Y-axis. The points are colored according to their true Species label, providing a clear map of the model's performance in separating the classes.

 # Combine the training data frame with the computed linear discriminant scores
lda_plot <- cbind(train, predict(model)$x)

Create the visualization using ggplot2
ggplot(lda_plot, aes(LD1, LD2)) +
geom_point(aes(color = Species))



The resulting plot clearly demonstrates the exceptional separation achieved by the LDA model. The *setosa* species (blue dots) is perfectly isolated along the LD1 axis, confirming why the model achieved 100% accuracy. The primary goal of LDA--finding the optimal projection axes for class discrimination--is visibly satisfied.

For those interested in reviewing the full implementation, the complete R script used throughout this tutorial is available for download here.