How to do a Principal Components Analysis in R: Step-by-Step Example

Authored by

stats writer

December 18, 2025

Principal Components Analysis (PCA) is one of the most fundamental and powerful techniques in modern statistical learning. It serves as a crucial tool for **dimensionality reduction**, allowing analysts to simplify complex datasets while preserving the majority of the original information structure. The primary goal of PCA is to transform a large set of variables into a smaller set of composite variables, known as **principal components**. These components are created such that they are uncorrelated and ordered by the amount of **variance** they explain in the data.

Executing PCA is streamlined within the R statistical environment, primarily utilizing functions like `prcomp()` or `princomp()`. The process is systematic: data must be imported, appropriately **scaled** (standardized), and then subjected to the PCA algorithm. The resulting principal components provide a low-dimensional representation of the dataset, which is invaluable for visualization, noise reduction, and subsequent modeling efforts, such as regression or clustering.

The Core Concept of Principal Components Analysis (PCA)

Principal Components Analysis, often abbreviated PCA, is classified as an unsupervised machine learning technique. Unlike supervised methods that use labeled data, PCA works by identifying underlying patterns and structure within unlabeled data. Its mathematical objective is to find **principal components**--which are linear combinations of the original predictor variables--that maximize the amount of variance captured in the dataset. This reduction is vital when dealing with high-dimensional data where direct interpretation or visualization becomes impractical.

Consider a dataset containing *p* variables. If *p* is large (e.g., 15), the number of pairwise scatterplots required to visualize all relationships is calculated as p(p-1)/2. For 15 predictors, this results in 105 distinct scatterplots! This sheer volume makes exploratory data analysis challenging and often overwhelming. PCA elegantly solves this problem by projecting the data onto a lower-dimensional subspace (typically 2 or 3 dimensions) that retains the maximal possible variance.

By achieving this low-dimensional representation, we can effectively summarize the dataset's variability. If we are successful in capturing, say, 80% of the total variability using just the first two principal components, we can project all observations onto a simple, insightful **scatterplot**. This allows us to observe clusters, outliers, and trends that would otherwise be obscured in a high-dimensional space.

Mathematical Derivation of Principal Components

The core mechanism for finding these components involves calculating sequential linear combinations of the original predictors. Given a dataset with *p* predictors: X1, X2, … , Xp, we calculate a set of *M* principal components: Z1, … , ZM, where M ≤ p.

Each component Zm is derived as a weighted sum of the original variables, represented by the equation: Zm = ΣΦjmXj. The constants Φ1m, Φ2m, …, Φpm are known as the **loadings**, which determine the direction and magnitude of the component in the original variable space. The calculation follows a strict sequence:

Z1 is the linear combination of the predictors that captures the **most variance possible**. It defines the direction in the data space where the observations are most spread out.

Z2 is the next linear combination of the predictors that captures the most variance while being **orthogonal** (i.e., uncorrelated) to Z1.

Z3 is then the next linear combination of the predictors that captures the most variance while being orthogonal to all previously defined components.

And so on, up to M components.

In practice, this process relies heavily on linear algebra, specifically the decomposition of the covariance matrix (or correlation matrix, if data is scaled).

The procedural steps for calculating these linear combinations are typically:

Scaling: Each of the variables is standardized to have a mean of 0 and a standard deviation of 1.

Covariance Calculation: The covariance matrix is computed for the scaled variables.

Eigen-Decomposition: We calculate the eigenvalues and eigenvectors of the covariance matrix.

Using linear algebra, it can be shown that the eigenvector that corresponds to the largest eigenvalue is the first principal component. In other words, this particular combination of the predictors explains the most variance in the data. The eigenvector corresponding to the second largest eigenvalue is the second principal component, and so on. This tutorial provides a step-by-step example of how to perform this essential process in R.

Step 1: Preparing the R Environment and Loading Data

First we'll load the **tidyverse** package, which contains several useful functions for visualizing and manipulating data:

```
library(tidyverse)
```

For this example we'll use the *USArrests* dataset built into R, which contains the number of arrests per 100,000 residents in each U.S. state in 1973 for *Murder*, *Assault*, and *Rape*. It also includes

the percentage of the population in each state living in urban areas, *UrbanPop*.

The following code show how to load and view the first few rows of the dataset:

```
#load data
data("USArrests")

#view first six rows of data
head(USArrests)
```

Murder Assault UrbanPop Rape
Alabama 13.2 236 58 21.2
Alaska 10.0 263 48 44.5
Arizona 8.1 294 80 31.0
Arkansas 8.8 190 50 19.5
California 9.0 276 91 40.6
Colorado 7.9 204 78 38.7

Step 2: Calculating Principal Components

After loading the data, we can use the R built-in function **prcomp()** to calculate the principal components of the dataset. This function efficiently performs the necessary scaling and eigen-decomposition.

Be sure to specify **scale = TRUE** so that each of the variables in the dataset are scaled to have a mean of 0 and a standard deviation of 1 before calculating the principal components. Also note that eigenvectors in R sometimes point in the negative direction by default, so we'll multiply the rotation matrix by -1 to reverse the signs for easier interpretation of the loadings.

```
#calculate principal components
results <- prcomp(USArrests, scale = TRUE)

#reverse the signs of the rotation (loadings)
results$rotation <- -1*results$rotation

#display principal components (the loadings matrix)
results$rotation
```

PC1 PC2 PC3 PC4
Murder 0.5358995 -0.4181809 0.3412327 -0.64922780
Assault 0.5831836 -0.1879856 0.2681484 0.74340748
UrbanPop 0.2781909 0.8728062 0.3780158 -0.13387773

Rape 0.5434321 0.1673186 -0.8177779 -0.08902432

We can see that the first principal component (PC1) has high positive loadings for Murder, Assault, and Rape, which indicates that this principal component describes the most variation in these interconnected crime variables. PC1 can therefore be interpreted as a general measure of **violent crime rate**.

We can also see that the second principal component (PC2) has a high positive loading for UrbanPop (0.873) and a negative loading for Murder. This structure suggests that PC2 captures variability related to urbanization levels, contrasting urban areas with regions having high rates of certain crimes.

Note that the principal components scores for each state are stored in **results$x**. We will also multiply these scores by -1 to reverse the signs, ensuring consistency with the orientation of the loadings:

#reverse the signs of the scores
results$x <- -1*results$x

#display the first six scores
head(results$x)

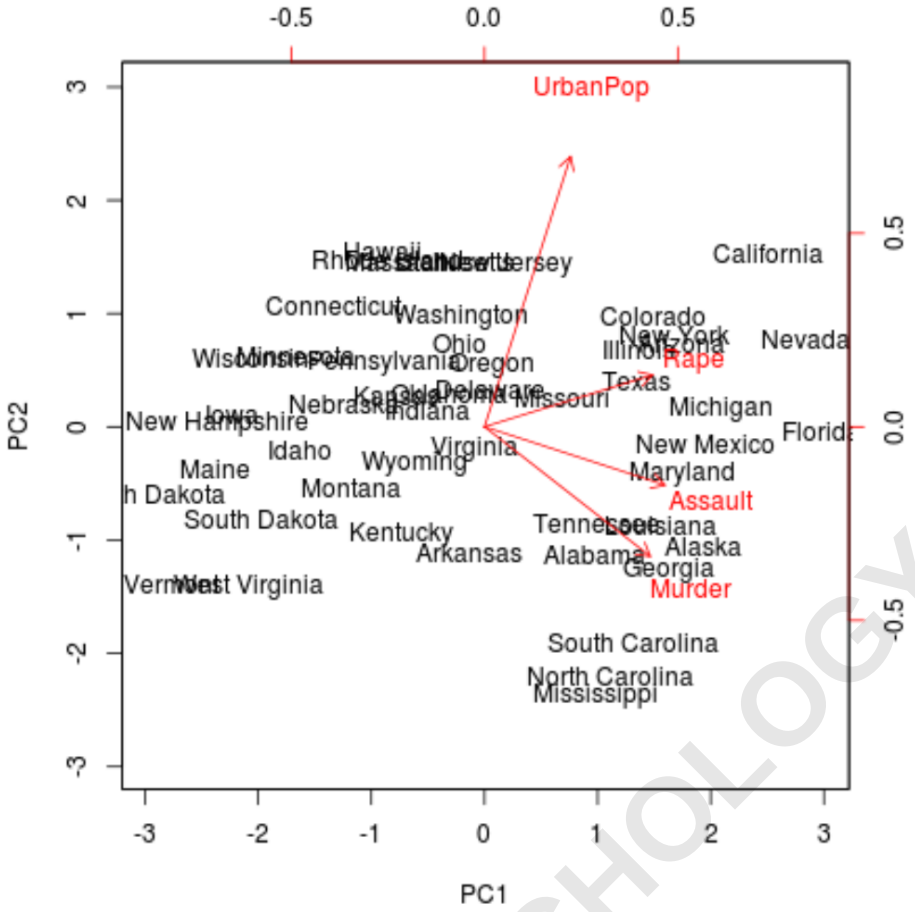
PC1 PC2 PC3 PC4
Alabama 0.9756604 -1.1220012 0.43980366 -0.154696581
Alaska 1.9305379 -1.0624269 -2.01950027 0.434175454
Arizona 1.7454429 0.7384595 -0.05423025 0.826264240
Arkansas -0.1399989 -1.1085423 -0.11342217 0.180973554
California 2.4986128 1.5274267 -0.59254100 0.338559240
Colorado 1.4993407 0.9776297 -1.08400162 -0.001450164

Step 3: Visualizing the Results with a Biplot

Next, we can create a **biplot** - a critical visualization tool that projects both the observations (states) and the variable loadings (arrows) onto a scatterplot defined by the first and second principal components:

Note that **scale = 0** ensures that the arrows in the plot are scaled appropriately to represent the loadings relative to the observation scores, facilitating accurate interpretation.

biplot(results, scale = 0)



From the plot we can see each of the 50 states represented in a simple two-dimensional space. The states that are close to each other on the plot have similar data patterns in regards to the variables in the original dataset. For instance, states clustered in the top-right quadrant have high scores on both PC1 (high violent crime) and PC2 (high urbanization).

We can also see that certain states are more highly associated with certain crimes than others. The length and direction of the arrows indicate the influence of the original variables. States that lie closest to the direction of a variable's arrow are those that score highest on that variable. For example, Georgia is the state closest to the variable *Murder* in the plot, confirming that it is an outlier with a high murder rate relative to the group structure.

If we take a look at the states with the highest murder rates in the original dataset, we can see that Georgia is indeed at the top of the list:

#display states with highest murder rates in original dataset
head(USArrests)

Murder Assault UrbanPop Rape

Georgia 17.4 211 60 25.8
Mississippi 16.1 259 44 17.1
Florida 15.4 335 80 31.9
Louisiana 15.4 249 66 22.2
South Carolina 14.4 279 48 22.5
Alabama 13.2 236 58 21.2

Step 4: Finding Variance Explained by Each Principal Component

To determine the quality of our low-dimensional representation, we must quantify the total variance explained by each principal component. This calculation involves taking the square of the standard deviations (`results$sdev^2`), which are proportional to the eigenvalues, and dividing them by the sum of all variances (the total variance in the system):

#calculate total variance explained by each principal component
results$sdev^2 / sum(results$sdev^2)

0.62006039 0.24744129 0.08914080 0.04335752

From the results we can observe the following breakdown of the total variability:

The first principal component (PC1) explains approximately **62.0%** of the total variance in the dataset.

The second principal component (PC2) explains an additional **24.7%** of the total variance.

The third principal component (PC3) explains **8.9%** of the total variance.

The fourth principal component (PC4) explains **4.3%** of the total variance.

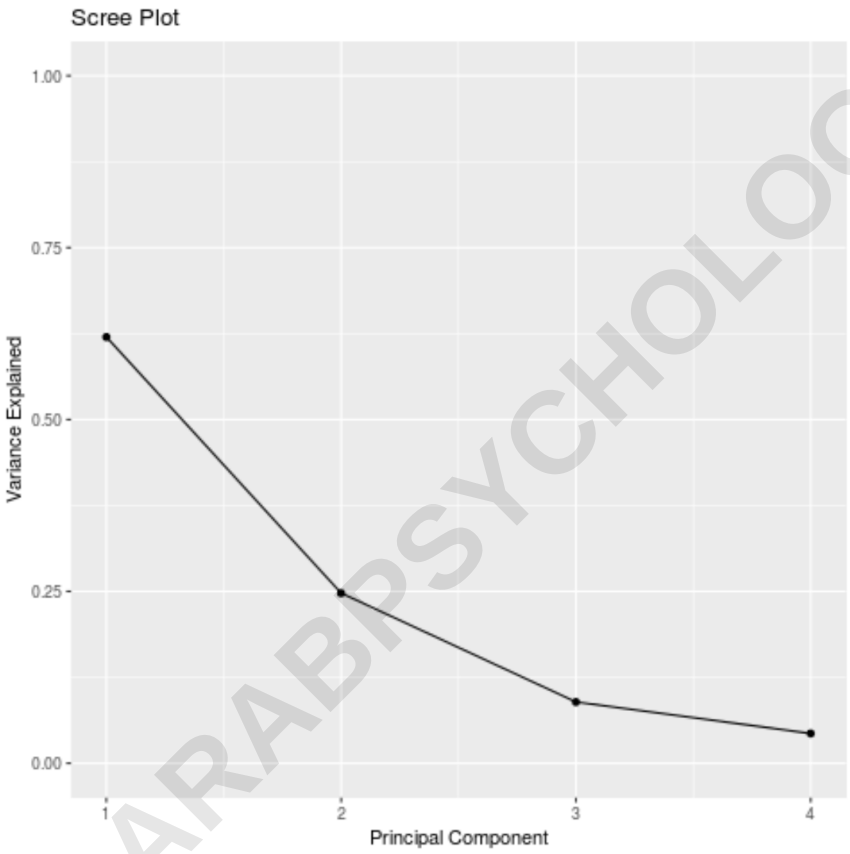
Thus, the first two principal components explain a highly significant majority (86.7%) of the total variance in the data. This confirms the validity of using the two-dimensional biplot for identifying patterns and similarities between the states, as nearly all of the original signal is retained within this reduced space.

Step 5: Visualizing Variance with a Scree Plot

We can also create a **scree plot** - a plot that displays the total variance explained by each principal component - to visually assess the components' importance and help determine how many components should be retained.

```
#calculate total variance explained by each principal component
var_explained = results$sdev^2 / sum(results$sdev^2)

#create scree plot
qplot(c(1:4), var_explained) +
geom_line() +
xlab("Principal Component") +
ylab("Variance Explained") +
ggtitle("Scree Plot") +
ylim(0, 1)
```



The plot vividly illustrates the rapid decrease in explanatory power after the second component. The "elbow" typically marks the point after which the components contribute negligibly to the overall variance. This visual evidence supports the mathematical finding that retaining the first two principal components is sufficient to capture the essential characteristics of the *USArrests* dataset.

Principal Components Analysis in Practice

In practice, PCA is utilized most often for two critical purposes in data science and statistics:

1. Exploratory Data Analysis and Feature Engineering: PCA is invaluable when we're first exploring a dataset and need to understand which observations are most similar to each other or when we seek to reduce noise in high-dimensional features. The resulting principal components can be used as new, uncorrelated features for machine learning models, leading to more robust results and reduced model complexity.

2. Principal Components Regression (PCR) and Handling Collinearity: We can also use PCA to calculate principal components that can then be used in Principal Components Regression. This technique is often employed when multicollinearity--high correlation between predictor variables--exists in a standard regression dataset. By replacing the collinear predictors with their uncorrelated principal components, PCR stabilizes the regression estimates and improves the model's reliability.

The complete R code used in this tutorial can be found here.