Principal Component Analysis in R Tutorial

In this tutorial, you'll learn how to use R PCA (Principal Component Analysis) to extract data with many variables and create visualizations to display that data.

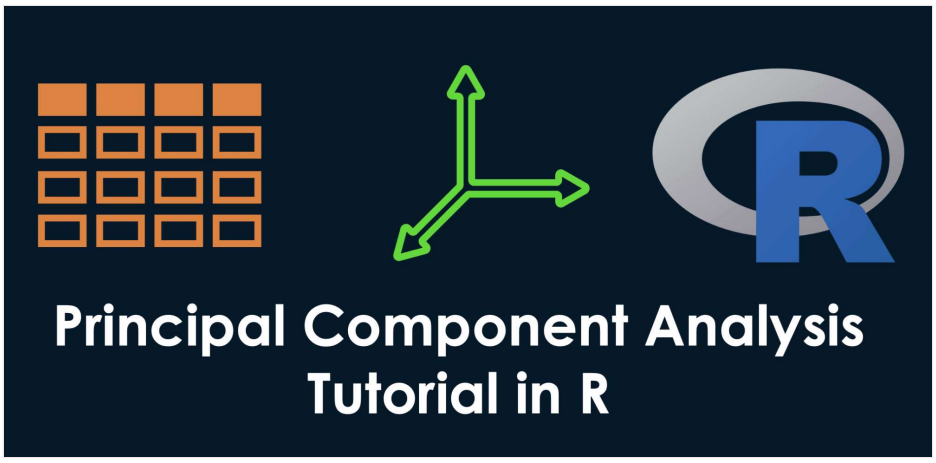
Updated Feb 13, 2023 · 15 min read

Zoumana Keita
A data scientist who likes to write and share knowledge with the data and IA community

TOPICS

R Programming

Data Analysis



Introduction to Principal Component Analysis (PCA)

As a data scientist in the retail industry, imagine that you are trying to understand **what makes a customer happy** from a dataset containing these five characteristics: monthly expense, age, gender, purchase frequency, and product rating. To better analyze and draw actionable conclusions, we need to understand the data set or, at the very least, visualize it. Human beings cannot easily visualize more than three dimensions, hence visualizing customer data with five characteristics (dimensions) is not straightforward. This is where principal component analysis (PCA for short) comes in.

"But, what is principal component analysis?"

It is a statistical approach that can be used to analyze high-dimensional data and capture the most important information from it. This is done by transforming the original data into a lower-dimensional space while collating highly correlated variables together. In our scenario,

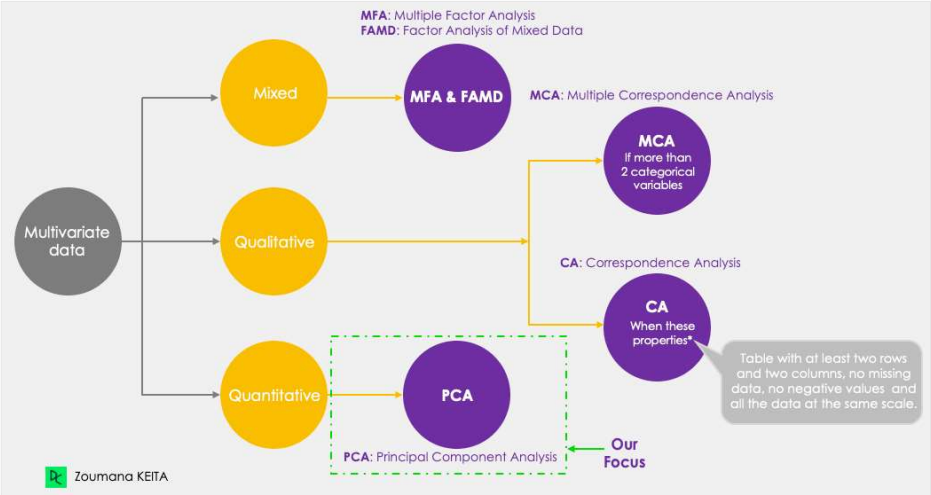
PCA would pick three characteristics such as monthly expense, purchase frequency, and product rating. This could make it easier to visualize and understand the data.

After this tutorial, you will have a better understanding of the principal component analysis and how to apply it to real-life scenarios using the famous *corrr* package in R.

Watch and learn more about Principal Component Analysis in R in this video from our course.

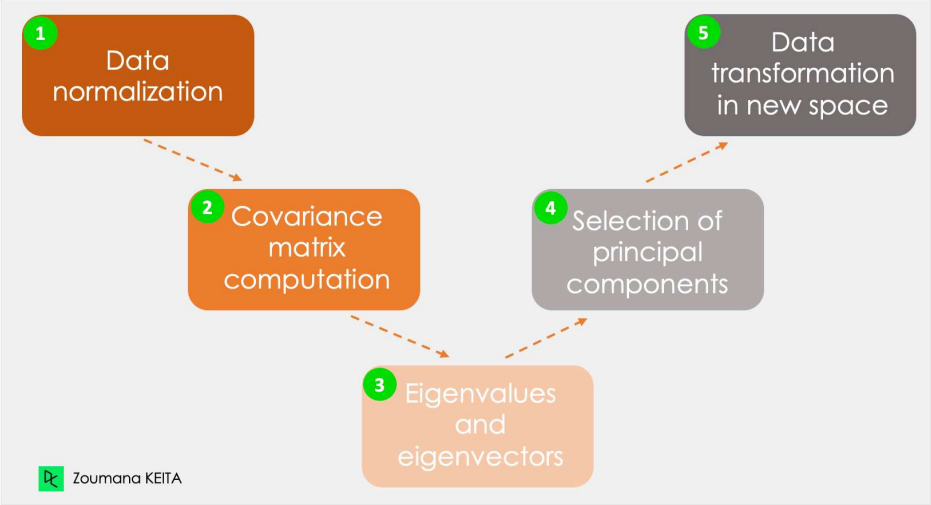
How Does PCA Work? A 5-Step Guide

Even though our focus is PCA, let's keep in mind the following five main principal component techniques that aim to summarize and visualize multivariate data. PCA, as opposed to the other techniques, only works with quantitative variables.



Principal component methods

We won't go into the explanation of the mathematical concept, which can be somewhat complex. However, understanding the following five steps can give a better idea of how to compute the PCA.



The five main steps for computing principal components

Step 1 - Data normalization

By considering the example in the introduction, let's consider, for instance, the following information for a given client.

- Monthly expenses: $300
- Age: 27
- Rating: 4.5

This information has different scales and performing PCA using such data will lead to a biased result. This is where data normalization comes in. It ensures that each attribute has the same level of contribution, preventing one variable from dominating others. For each variable, normalization is done by subtracting its mean and dividing by its standard deviation.

Step 2 - Covariance matrix

As the same suggests, this step is about computing the **covariance matrix** from the normalized data. This is a symmetric matrix, and each element (i, j) corresponds to the covariance between variables i and j.

Step 3 - Eigenvectors and eigenvalues

Geometrically, an **eigenvector** represents a direction such as "vertical" or "90 degrees". An **eigenvalue**, on the other hand, is a number representing the amount of variance present in the data for a given direction. Each eigenvector has its corresponding eigenvalue.

Step 4 - Selection of principal components

There are as many pairs of eigenvectors and eigenvalues as the number of variables in the data. In the data with only monthly expenses, age, and rate, there will be three pairs. Not all the pairs are relevant. So, the eigenvector with the highest eigenvalue corresponds to the first principal component. The second principal component is the eigenvector with the second highest eigenvalue, and so on.

Step 5 - Data transformation in new dimensional space

This step involves re-orienting the original data onto a new subspace defined by the principal components This reorientation is done by multiplying the original data by the previously computed eigenvectors.

It is important to remember that this transformation does not modify the original data itself but instead provides a new perspective to better represent the data.

Applications of Principal Component Analysis

Principal component analysis has a variety of applications in our day-to-day life, including (but by no means limited to) finance, image processing, healthcare, and security.

Finance

Forecasting stock prices from past prices is a notion used in research for years. PCA can be used for dimensionality reduction and analyzing the data to help experts find relevant components that account for most of the data's variability. You can learn more about dimensionality reduction in R in our dedicated course.

Image processing

An image is made of multiple features. PCA is mainly applied in image compression to retain the essential details of a given image while reducing the number of dimensions. In addition, PCA can be used for more complicated tasks such as image recognition.

Healthcare

In the same logic of image compression. PCA is used in magnetic resonance imaging (MRI) scans to reduce the dimensionality of the images for better visualization and medical analysis. It can also be integrated into medical technologies used, for instance, to recognize a given disease from image scans.

Security

Biometric systems used for fingerprint recognition can integrate technologies leveraging principal component analysis to extract the most relevant features, such as the texture of the fingerprint and additional information.

Real-World Example of PCA in R

Now that you understand the underlying theory of PCA, you are finally ready to see it in action.

This section covers all the steps from installing the relevant packages, loading and preparing the data applying principal component analysis in R, and interpreting the results.

The source code is available from DataCamp's workspace.

Setting up the environment

To successfully perform this tutorial, you'll need the following libraries, and each one requires two main steps to be used efficiently:

- Install the library to access all the functions.
- Load to be able to use all the functions.

corrr package in R

This is an R package for correlation analysis. It mainly focuses on creating and handling R data frames. Below are the steps to install and load the library.

```
install.packages("corrr")
library('corrr')
```

Explain code

POWERED BY datalab

ggcorrplot package in R

The ggcorrplot package provides multiple functions but is not limited to the ggplot2 function that makes it easy to visualize correlation matrix. Similarly to the above instruction, the installation is straightforward.

```
install.packages("ggcorrplot")
library(ggcorrplot)
```

Explain code POWERED BY datalab

FactoMineR package in R

Mainly used for multivariate exploratory data analysis; the factoMineR package gives access to the PCA module to perform principal component analysis.

```
install.packages("FactoMineR")
library("FactoMineR")
```

Explain code POWERED BY datalab

factoextra package in R

This last package provides all the relevant functions to visualize the outputs of the principal component analysis. These functions include but are not limited to scree plot, biplot, only to mention two of the visualization techniques covered later in the article.

Exploring the data

Before loading the data and performing any further exploration, it is good to understand and have the basic information related to the data you will be working with.

Protein data

The protein data set is a real-valued multivariate data set describing the average protein consumption by citizens of 25 European countries.

For each country, there are ten columns. The first eight correspond to the different types of proteins. The last one corresponds to the total value of the average values of proteins.

Let's have a quick overview of the data.

First, we load the data using the `read.csv()` function, then `str()` which gives the image below.

```
protein_data <- read.csv("protein.csv")
str(protein_data)
```

Explain code POWERED BY datalab

We can see that the data set has 25 observations and 11 columns, and each variable is numerical, except the **Country** column, which is a text.

```
'data.frame':   25 obs. of  11 variables:
 $ Country             : chr  "Albania" "Austria" "Belgium" "Bulgaria" ...
 $ Red_Meat            : int  10 9 14 8 10 11 8 10 18 10 ...
 $ White_Meat          : int  1 14 9 6 11 11 12 5 10 3 ...
 $ Eggs                : int  1 4 4 2 3 4 4 3 3 3 ...
 $ Milk                : int  9 20 18 8 13 25 11 34 20 18 ...
 $ Fish                : int  0 2 5 1 2 10 5 6 6 6 ...
 $ Cereals             : int  42 28 27 57 34 22 25 26 28 42 ...
 $ Starchy_Foods       : int  1 4 6 1 5 5 7 5 5 2 ...
 $ Pulses_nuts_oilseeds: int  6 1 2 4 1 1 1 1 2 8 ...
 $ Fruits_Vegetables   : int  2 4 4 4 4 2 4 1 7 7 ...
 $ Total               : int  72 86 89 91 83 91 77 91 99 99 ...
```

Description of the protein data

Check for null values

The presence of missing values can bias the result of PCA. Therefore, it is highly recommended to perform the appropriate approach to tackle those values. Our **Top Techniques to Handle Missing Values Every Data Scientist Should Know** tutorial can help you make the right choice.

```
colSums(is.na(protein_data))
```

Explain code POWERED BY datalab

The colSums() function combined with the is.na() returns the number of missing values in each column. As we can see below, none of the columns have missing values.

```
Country:       0 Red_Meat:       0 White_Meat:       0 Eggs:       0 Milk:       0 Fish:       0
 Cereals:       0 Starchy_Foods:       0 Pulses_nuts_oilseeds:       0 Fruits_Vegetables:
      0 Total:       0
```

Number of missing values in each column

Normalizing the data

As stated early in the article, PCA only works with numerical values. So, we need to get rid of the **Country** column. Also, the **Total** column is not relevant to the analysis since it is the linear combination of the remaining numerical variables.

The code below creates new data with only numeric columns.

```
numerical_data <- protein_data[,2:10]

head(numerical_data)
```

Explain code POWERED BY datalab

	Red_Meat	White_Meat	Eggs	Milk	Fish	Cereals
1	10	1	1	9	0	42
2	9	14	4	20	2	28
3	14	9	4	18	5	27
4	8	6	2	8	1	57
5	10	11	3	13	2	34
6	11	11	4	25	10	22

Before the normalization of the data (only the first five columns are shown)

Now, the normalization can be applied using the scale() function.

```
data_normalized <- scale(numerical_data)
head(data_normalized)
```

Explain code POWERED BY datalab

A matrix: 6 × 9 of type dbl

Red_Meat	White_Meat	Eggs	Milk	Fish	Cereals
0.05876425	-1.8498883	-1.86538958	-1.1665829	-1.2333048	0.8791769
-0.23505701	1.6253354	0.82507616	0.3832253	-0.6569941	-0.3923599
1.23404931	0.2887109	0.82507616	0.1014420	0.2074718	-0.4831840
-0.52887828	-0.5132638	-0.96856767	-1.3074746	-0.9451495	2.2415378
0.05876425	0.8233607	-0.07174575	-0.6030163	-0.6569941	0.1525844
0.35258552	0.8233607	0.82507616	1.0876836	1.6482484	-0.9373043

Normalized data (only first five columns shown)

Applying PCA

Now, all the resources are available to conduct the PCA analysis. First, the princomp() computes the PCA, and summary() function shows the result.

```
data.pca <- princomp(data_normalized)
summary(data.pca)
```

POWERED BY datalab

```
Importance of components:
                          Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
Standard deviation     1.306476 0.5182247 0.3610639 0.27323842 0.14161042
Proportion of Variance 0.768188 0.1208652 0.0586723 0.03360071 0.00902517
Cumulative Proportion  0.768188 0.8890531 0.9477255 0.98132616 0.99035133
                           Comp.6       Comp.7       Comp.8 Comp.9
Standard deviation     0.114308469 0.081225268 0.042129867      0
Proportion of Variance 0.005880602 0.002969253 0.000798813      0
Cumulative Proportion  0.996231934 0.999201187 1.000000000      1
```

Focus here

R PCA summary

From the previous screenshot, we notice that nine principal components have been generated (Comp.1 to Comp.9), which also correspond to the number of variables in the data.

Each component explains a percentage of the total variance in the data set. In the **Cumulative Proportion** section, the first principal component explains almost 77% of the total variance. This implies that almost two-thirds of the data in the set of 9 variables can be represented by just the first principal component. The second one explains 12.08% of the total variance.

The cumulative proportion of Comp.1 and Comp.2 explains nearly 89% of the total variance. This means that the first two principal components can accurately represent the data.

It's great to have the first two components, but what do they really mean?

This can be answered by exploring how they relate to each column using the loadings of each principal component.

```
data.pca$loadings[, 1:2]
```

Explain code

POWERED BY datalab

A matrix: 9 × 2 of type dbl

	Comp.1	Comp.2
Red_Meat	0.2993407	0.10651363
White_Meat	0.3193363	0.22312711
Eggs	0.4134492	0.11960563
Milk	0.3837089	0.15175036
Fish	0.1137789	-0.68417466
Cereals	-0.4246411	0.28573531
Starchy_Foods	0.2807581	-0.40862948
Pulses_nuts_oilseeds	-0.4375650	-0.07772646
Fruits_Vegetables	-0.1633793	-0.42281956

Loading matrix of the first two principal components

The loading matrix shows that the first principal component has high positive values for both red meat, white meat, eggs, and milk. However, the values for cereals, pulses, nuts and oilseeds, and fruits and vegetables are relatively negative. This suggests that countries with a higher intake of animal protein are in excess, while countries with a lower intake are in deficit.

When it comes to the second principal component, it has high negative values for fish, starchy foods, and fruits and vegetables. This implies that the underlying countries' diets are highly influenced by their location, such as coastal regions for fish, and inland regions for a diet rich in vegetables and potatoes.

Visualization of the principal components

The previous analysis of the loading matrix gave a good understanding of the relationship between each of the first two principal components and the attributes in the data. However, it might not be visually appealing.

There are a couple of standard visualization strategies that can help the user glean insight into the data, and this section aims to cover some of those approaches, starting with the scree plot.

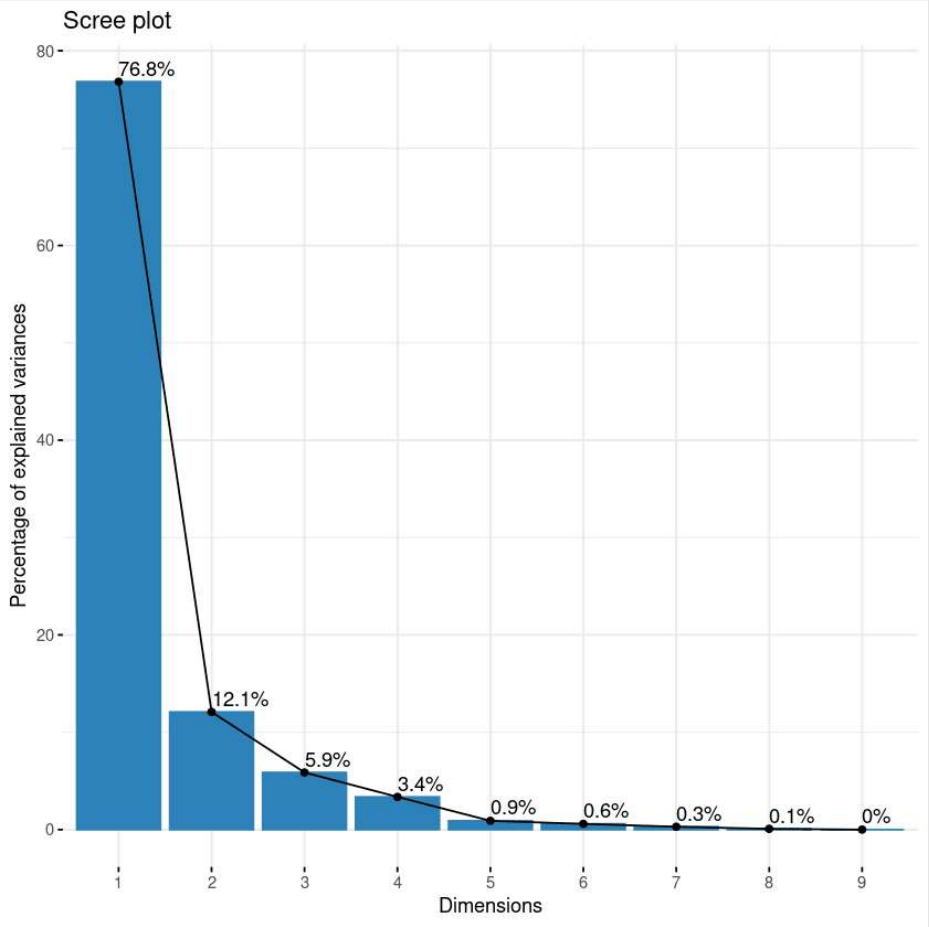
Scree Plot

The first approach of the list is the scree plot. It is used to visualize the importance of each principal component and can be used to determine the number of principal components to retain. The scree plot can be generated using the fviz_eig() function.

```
fviz_eig(data.pca, addlabels = TRUE)
```

Explain code

POWERED BY datalab



Scree plot of the components

This plot shows the eigenvalues in a downward curve, from highest to lowest. The first two components can be considered to be the most significant since they contain almost 89% of the total information of the data.

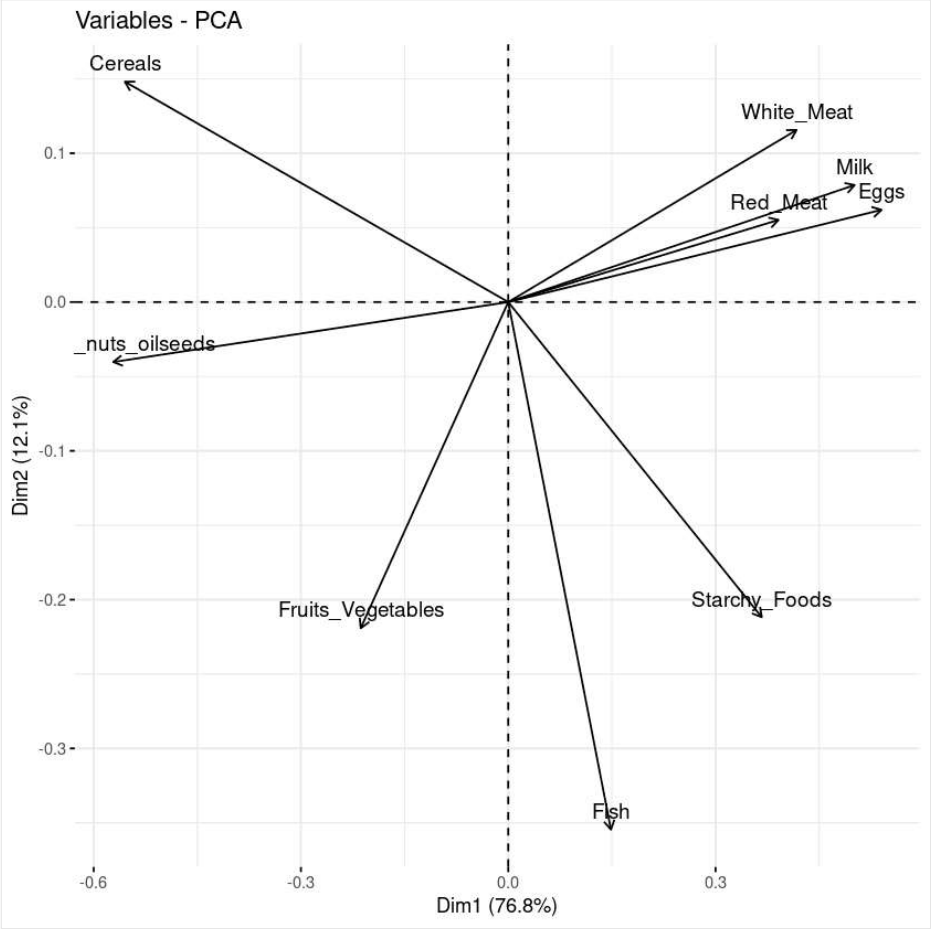
Biplot of the attributes

With the biplot, it is possible to visualize the similarities and dissimilarities between the samples, and further shows the impact of each attribute on each of the principal components.

```
# Graph of the variables
fviz_pca_var(data.pca, col.var = "black")
```

Explain code

POWERED BY datalab



Biplot of the variables with respect to the principal components

Three main pieces of information can be observed from the previous plot.

- First, all the variables that are grouped together are positively correlated to each other, and that is the case for instance for white/red meat, milk, and eggs have a positive correlation to each. This result is surprising because they have the highest values in the loading matrix with respect to the first principal component.
- Then, the higher the distance between the variable and the origin, the better represented that variable is. From the biplot, eggs, milk, and white meat have higher magnitude compared to red meat, and hence are well represented compared to red meat.
- Finally, variables that are negatively correlated are displayed to the opposite sides of the biplot's origin.

Contribution of each variable

The goal of the third visualization is to determine how much each variable is represented in a given component. Such a quality of representation is called the Cos2 and corresponds to the square cosine, and it is computed using the fviz_cos2 function.

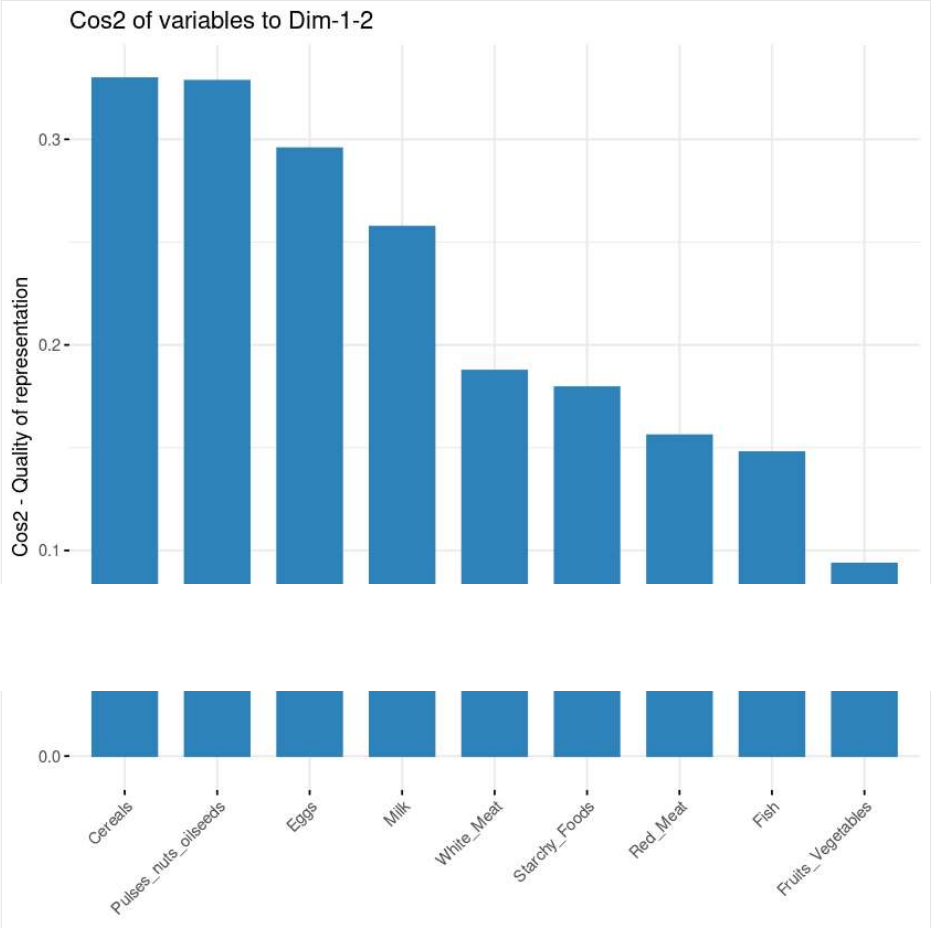
- A low value means that the variable is not perfectly represented by that component.
- A high value, on the other hand, means a good representation of the variable on that component.

```
fviz_cos2(data.pca, choice = "var", axes = 1:2)
```

Explain code

The code above computed the square cosine value for each variable with respect to the first two principal components.

From the illustration below, cereals, pulse nut oilseeds, eggs, and milk are the top four variables with the highest cos2, hence contributing the most to PC1 and PC2.



Variables' contribution to principal components

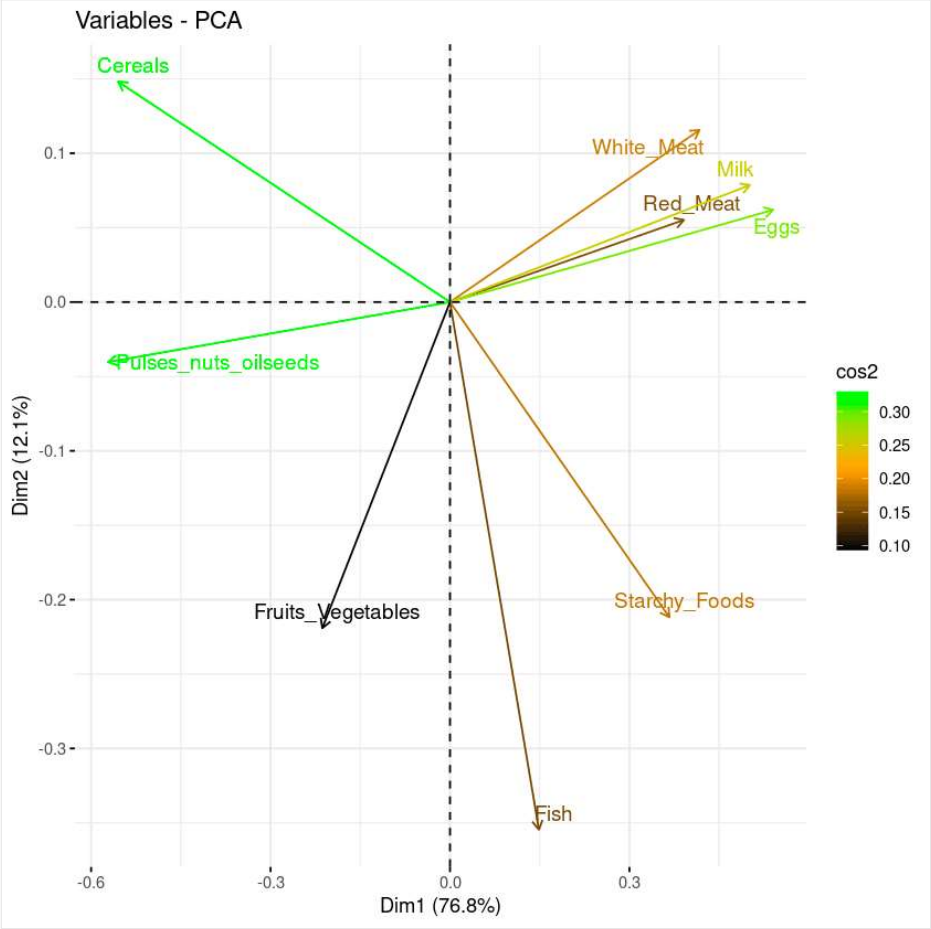
Biplot combined with cos2

The last two visualization approaches: biplot and attributes importance can be combined to create a single biplot, where attributes with similar cos2 scores will have similar colors. This is achieved by fine-tuning the fviz_pca_var function as follows:

```
fviz_pca_var(data.pca, col.var = "cos2",
             gradient.cols = c("black", "orange", "green"),
             repel = TRUE)
```


From the biplot below:

- High cos2 attributes are colored in green: Cereals, pulses, oilseeds, eggs, and milk.
- Mid cos2 attributes have an orange color: white meat, starchy food, fish, and red meat.
- Finally, low cos2 attributes have a black color: fruits and vegetables,



Combination of biplot and cos2 score

Conclusion

This article has covered what principal component analysis is and its importance in data analytics using the correlation matrix from the corrr package. In addition to covering some real world applications, it has also walked you through a PCA example with different visualization strategies from using the existing function to fine-tuning them using the combination of biplot and cos2 for better understanding and visualization of the relationship between pca analysis in r and the attributes.

We hope it provides you with the relevant skills to efficiently visualize and understand the hidden insights from your data.

To further your learning about principal component analysis, consider **Principal Component Analysis in Python tutorial**. It illustrated the use of PCA with Python on both tabular and image data sets. Our **Introduction to R** course is a good next step to master the basics of data analysis in R, including vectors, lists, and data frames, and practice with R with real data sets.

PCA Analysis FAQ

Is PCA feature extraction or selection?

PCA leverages an unsupervised linear transformation to perform feature extraction and dimensionality reduction.

When should you use PCA analysis?

What are the limitations of PCA?

What is the main advantage of PCA?

What is PC1 and PC2 in principal component analysis?

What are the assumptions of principal component analysis?

How to do PCA in R?

TOPICS

R Programming Data Analysis

Courses for R

COURSE

Introduction to R

4 hr 2.8M

Master the basics of data analysis in R, including vectors, lists, and data frames, and practice R with real data sets.

See Details → Start Course

See More →

Related

BLOG

R Correlation Tutorial

TUTORIAL

How to Do Principal Component Analysis (PCA) in Python

TUTORIAL

PCH in R Tutorial

See More →