PCA_Outline

Sammy, Marina, Bradley

2023-03-06

IMPORT LIBRARIES

We will import the *FactoMineR* and *factoextra* R packages. These packages are commonly used for principal component analysis. *factorextra* is an extension of the *FactorMineR* package that provides additional visualizations and analysis tools.

Principal Component Analysis (PCA) Introduction

Overview of PCA

Principal Component Analysis (PCA) is a technique used to reduce the dimensionality of high-dimensional data by identifying and extracting the most important features or patterns in the data. It does this by transforming the original data set into a new set of variables (principal components) that are linear combinations of the original variables and capture the most important sources of variation in the data. The number of principal components is less than or equal to the number of original variables in the data set.

Overall the goals of PCA are to:
* Extract the most important information from the data. * Compress the size of the data set by keeping only the most important information. * Simplify the description of the data set. * Analyze the structure of the observations and the variables.

Research Question

A pizza lover wants to take a deeper look into the measurements that capture the kind of things that make a pizza tasty in order to determine which pizza brand works best for him.

Import the Data

The pizza data set can be downloaded from:
https://data.world/sdhilip/pizza-datasets

The data set contains the id, brand, and nutritional content for 300 samples and 10 brands.

Variables: brand – Pizza brand (class label) id – Sample being analyzed mois – Amount of water per 100 grams in the sample prot – Amount of protein per 100 grams in the sample fat – Amount of fat per 100 grams in the sample ash – Amount of ash per 100 grams in the sample sodium – Amount of sodium per 100 grams in the sample carb – Amount of carbohydrates per 100 grams in the sample cal – Amount of calories per 100 grams in the sample

```r
# Loads the CSV file into memory. You may need to adapt this line to work on your computer
pizza <- read.csv("Pizza.csv", header=TRUE, sep=",")

# Peek at the first few rows
head(pizza)
```

```
##   brand    id  mois  prot   fat  ash sodium carb  cal
## 1     A 14069 27.82 21.43 44.87 5.11   1.77 0.77 4.93
## 2     A 14053 28.49 21.26 43.89 5.34   1.79 1.02 4.84
## 3     A 14025 28.35 19.99 45.78 5.08   1.63 0.80 4.95
## 4     A 14016 30.55 20.15 43.13 4.79   1.61 1.38 4.74
## 5     A 14005 30.49 21.28 41.65 4.82   1.64 1.76 4.67
## 6     A 14075 31.14 20.23 42.31 4.92   1.65 1.40 4.67
```

Data Requirements

For PCA to work correctly, the following requirements must be met:

1. **Data must be numeric** Because the data is being scaled, categorical variables wouldn't make sense for this type of analysis. During PCA, mathematical operations like matrix multiplication and eigenvalue decomposition are used which only work for numeric values.
2. **Data must have at least three features** You need multiple variables to analyze correlation between them. Even though you can do PCA with two variables it is generally more common to do it with high dimensional data to see patterns that you wouldn't see otherwise and compressing data sets with lots of features. It is also useful for compressing and getting rid of noise in data sets with a lot of features.
3. **Data must not contain missing values** Rows with missing values need to be ignored because you can't look at the relationship between features.
4. **Data must be correlated** Correlation is important because PCA helps understand correlations between features, this kind of analysis won't show much for uncorrelated data.
5. **Data must be linear** PCA uses linear combinations to analyze the data, using Pythagorean distance between points and the origin and this won't work as well with other kinds of correlation.

1 & 2. Numeric Data and Feature Check

We can check the data types and number of features in our data frame by using the **str()** function.

```r
str(pizza)
```

```
## 'data.frame':    300 obs. of  9 variables:
##  $ brand : chr  "A" "A" "A" "A" ...
##  $ id    : int  14069 14053 14025 14016 14005 14075 14082 14097 14117 14133 ...
##  $ mois  : num  27.8 28.5 28.4 30.6 30.5 ...
##  $ prot  : num  21.4 21.3 20 20.1 21.3 ...
##  $ fat   : num  44.9 43.9 45.8 43.1 41.6 ...
##  $ ash   : num  5.11 5.34 5.08 4.79 4.82 4.92 4.71 5.28 5.02 5.16 ...
##  $ sodium: num  1.77 1.79 1.63 1.61 1.64 1.65 1.58 1.75 1.71 1.66 ...
##  $ carb  : num  0.77 1.02 0.8 1.38 1.76 1.4 1.77 2.95 1.18 0.64 ...
##  $ cal   : num  4.93 4.84 4.95 4.74 4.67 4.67 4.63 4.72 4.93 4.95 ...
```

All of the features are numeric, except of the `brand`, which is categorical, thus should not be included in our analysis.

The `id` is a sequence of numbers and will also be removed for the PCA (it does not reflect "real" variation in the data set).

Note: we will keep our brand column in the data set for now, but will drop it during our PCA analysis; this is for plotting functionality later on

```
# remove the id column
pizza <- subset(pizza, select = -id)

# setting pizza brand as a factor for graphing later on
pizza$brand <- as.factor(pizza$brand)

# ensure id column is removed
str(pizza)
```

```
## 'data.frame':    300 obs. of  8 variables:
##  $ brand : Factor w/ 10 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ mois  : num  27.8 28.5 28.4 30.6 30.5 ...
##  $ prot  : num  21.4 21.3 20 20.1 21.3 ...
##  $ fat   : num  44.9 43.9 45.8 43.1 41.6 ...
##  $ ash   : num  5.11 5.34 5.08 4.79 4.82 4.92 4.71 5.28 5.02 5.16 ...
##  $ sodium: num  1.77 1.79 1.63 1.61 1.64 1.65 1.58 1.75 1.71 1.66 ...
##  $ carb  : num  0.77 1.02 0.8 1.38 1.76 1.4 1.77 2.95 1.18 0.64 ...
##  $ cal   : num  4.93 4.84 4.95 4.74 4.67 4.67 4.63 4.72 4.93 4.95 ...
```

The pizza data set has 9 variables (7 removing `id` and `brand`) and 300 observations (pizza samples). This exceeds our three feature minimum criteria.

3. Checking for missing data

We can check if there is any missing observations with the **is.na()** function.

```
# returning the number of missing values in each column
colSums(is.na(pizza))
```

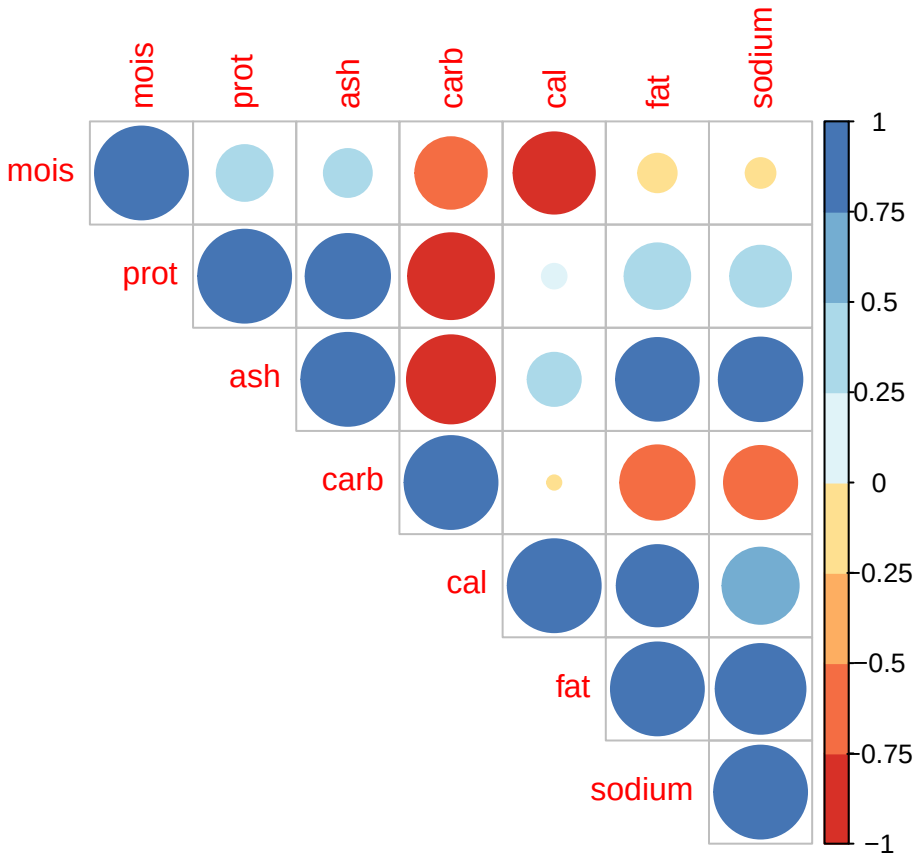
```
##  brand   mois   prot    fat    ash sodium   carb    cal
##      0      0      0      0      0      0      0      0
```

Here we can see, that none of the variable columns has missing values, thus none of the samples needs to be removed.

4. Correlation Check

To check for correlation we can create a correlation matrix and then use the **corrplot()** function to plot the features and their correlation to one another.

```
# create the correlation matrix
corr_matrix <- cor(pizza[,-1])

# plot the correlogram matrix
corrplot(corr_matrix, type="upper", order="hclust", col=brewer.pal(n=8, name="RdYlBu"))
```



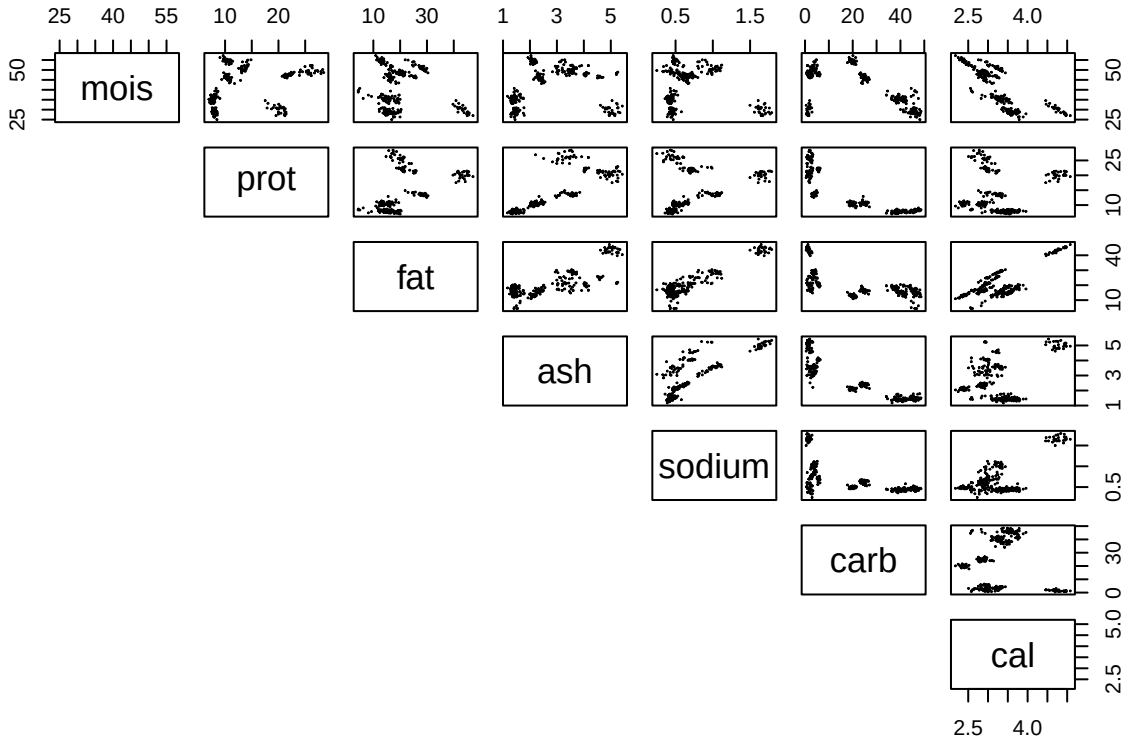
This Correlogram shows the most correlated variables in the data test. In this plot, the correlation coefficients are colored according to their value. Correlation between variables indicates that there is redundancy in the data. Due to this redundancy, PCA can be used to reduce the original variables into a smaller number of new variables (PCs) explaining most of the variance in the original variables.

In terms of this correlogram, there are features in the data set that have a quite high correlation (e.g. `protein` and `ash` or `sodium` and `ash`). There aren't many pairs of features that show low to no correlation; this lets us assume the data set is correlated enough for PCA to make sense.

5. Linearity Check

To check for linearity between features, we can create a pairwise plot of all features using the **pairs()** function.

```
# the pizza[,-1] drops the first column for this pairwise plot (brand)
# remove this because it will NOT be a part of the PCA
pairs(pizza[,-1], pch=19, cex=0.1,lower.panel=NULL)
```



Linearity is visible between some of the variables e.g.:
- `ash` and `sodium`
- `protein` and `ash`
- `fat` and `cal`
- `mois` and `carb`
- ....

Performing PCA()

Standardization

To perform principal component analysis, it is necessary to standardize the variables especially if they are measured in different scales, as failing to do so can significantly impact the PCA results. The purpose of scaling is to make the variables comparable by setting them to have a mean of zero and a standard deviation of one.

Note: By default the function PCA() [in FactoMineR] standardizes the data automatically during the PCA, so you don't need to before the PCA

Nevertheless, the below code chunk shows how to standardize the data beforehand.

The R base function 'scale()' can be used to standardize the data. It takes a numeric matrix as an input and performs scaling on the columns.

```
pizza.scaled<-scale(pizza[,-1])
pizza.scaled<-as.data.frame(pizza.scaled)
```

```
#head(pizza.scale)
```

PCA()

There are multiple ways to calculate PCA:
1. Eigendecomposition of the covariance matrix
2. Eigenvalue approximation via power iterative computation
3. Non-linear iterative partial least squares (NIPALS) computation 4. Singular value decomposition of the data matrix 5. ... and more.

There is different PCA functions from different packages that can be used. We will use PCA() from the FactoMineR package here, but there is others out there that work the same way. Some examples are:
- **stats**: prcomp() - **FactoMiner**: PCA()
- **ade4**: dudi.pca()
- **ExPosition**: epPCA()

The **PCA()** function utilizes the common **Singular Value Decomposition** (SVD) method which is what we will use in this demo!

For further information on the function, use the help function!

```
# use to get help for the PCA function
# uncomment the line below for help!
#?PCA
```

The function `PCA()` [FactoMineR package] can be used as follows:

```
PCA(X, scale.unit = TRUE, ncp = 7, graph = TRUE)

X: a data frame. The rows being individuals, and the columns being numeric values
scale.unit: If TRUE, the data are scaled to unit variance before analysis
ncp: number of dimensions kept in the final results
graph: If TRUE, a graph is displayed
```

We will first keep the *graph* variable as false, as we will re-perform this PCA function after the number of principal components are chosen.

```
# Make sure to set scale.unit to TRUE if your data isn't standardized!
pizza.PCA <- PCA(pizza[,-1], ncp = 7, scale.unit = TRUE, graph = FALSE)
# show output of PCA
pizza.PCA
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 300 individuals, described by 7 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
```

```
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

The above section shows the results of our PCA. We will utilize many of the objects from this output for our analysis.

The first object we will look at is the $*eig* object, in the next section *Determining the number of Principle Components (PC).*

Determining the number of Principal Components (PC)

The purpose of this analysis is to reduce the number of dimensions of the pizza data set while retaining all the information on what makes pizza tasty. We performed the PCA, but now we need to determine how many principal components (PCs) are necessary to accurately explain our pizza data set's overall variance. To do so, we will look towards the $*eig* object.

Eigenvalues Eigenvalues measure the amount of variation retained by each principal component. They indicate the relative importance of each principal component by accounting for the overall variability in the data, with larger eigenvalues indicating more important components. The sum of all eigenvalues equals the total variance in the original data, and the proportion of variance explained by each principal component is given by the ratio of its eigenvalue to the total variance.

Eigenvalues are larger for the first PCs and smaller for the subsequent PCs.

We can examine the eigenvalues to determine the number of principal components to be considered. The eigenvalues and the proportion of variance retained by the PCs can be extracted using the **get_eigenvalue()** function.

```
eig.val <- get_eigenvalue(pizza.PCA)
eig.val
```

```
##         eigenvalue variance.percent cumulative.variance.percent
## Dim.1 4.171782e+00     5.959688e+01                     59.59688
## Dim.2 2.290457e+00     3.272082e+01                     92.31770
## Dim.3 4.145623e-01     5.922319e+00                     98.24002
## Dim.4 9.517423e-02     1.359632e+00                     99.59966
## Dim.5 2.767702e-02     3.953860e-01                     99.99504
## Dim.6 3.376094e-04     4.822991e-03                     99.99986
## Dim.7 9.518780e-06     1.359826e-04                    100.00000
```

The rows in the table indicate each principal components (Dim #). The columns report the eigenvalue, variance percentage, and cumulative variance percentage.

Remember! Eigenvalues measure the amount of variation retained by each principal component.

You can see that the sum of all eigenvalues gives a total variance of 7.

The proportion of variation explained by each eigenvalue is given in the second column.

For example, $\frac{4.17}{7} = 0.59 = 59\ \%$ of the variation is explained by this first eigenvalue.

The cumulative percentage explained is obtained by adding the previous proportions of variations explained to a running total.

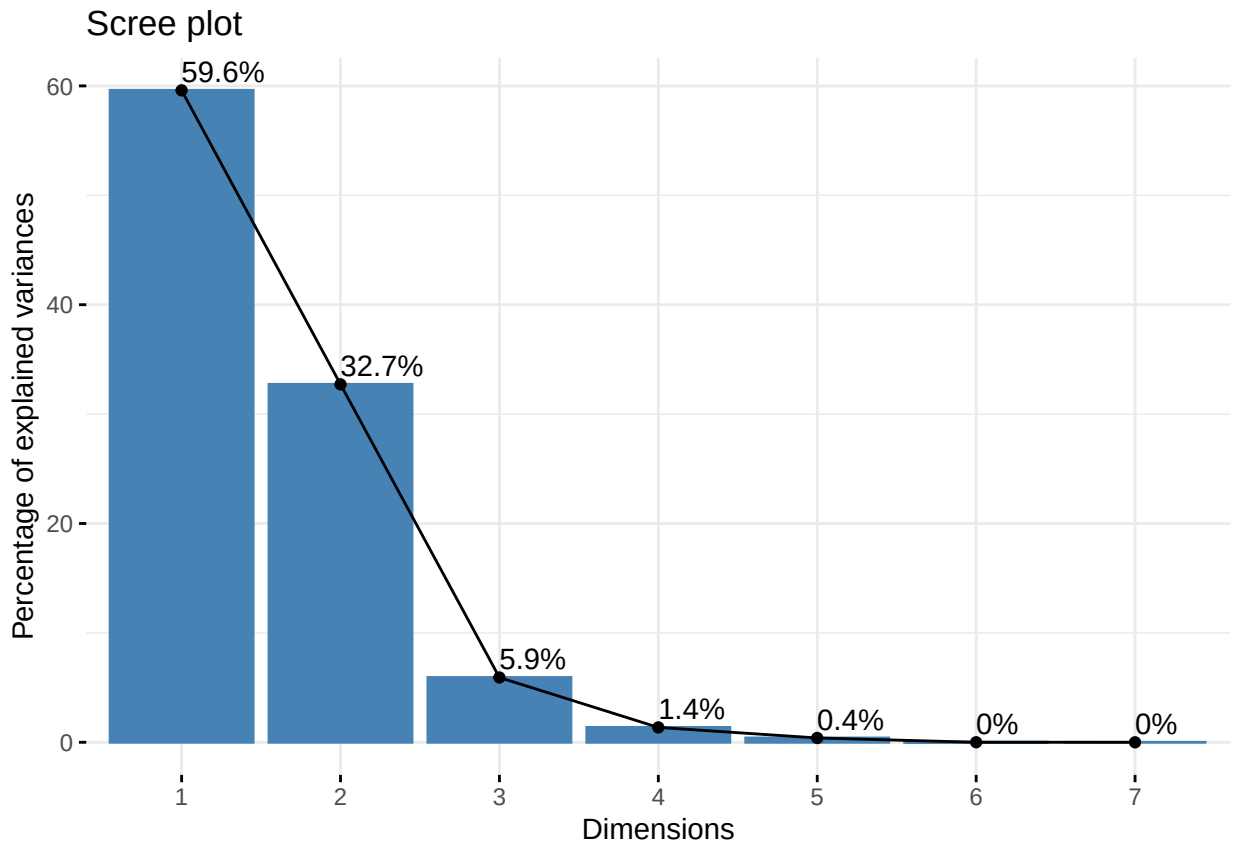
For example, the cumulative variance explained by PC1 (Dim1) and PC2 (Dim2) is $59.59 + 32.72 = \mathbf{92.3\%}$. Therefore, you can say: about **92.31%** of the variation is explained by the first two eigenvalues together.

These eigenvalues can be used to determine the number of principal components to keep in our analysis:

- An *eigenvalue* > 1 indicates that the principal component explains more variability in the data then the original variable would. This is commonly used as a cutoff point for which principal components are retained. This holds true only when the data is standardized!
- You can also use eigenvalues to limit the number of components by choosing an number that accounts for a certain fraction of the total variance. For example, if you are satisfied with 90% of the total variance explained then use the number of components to achieve that. In our case, we would use the first two PCs!

Scree Plot An alternative method to determine the number of PCs is to look at a Scree plot which is just a plot of the eigenvalues ordered from largest to smallest.
A Scree Plot can also be looked at as a diagnostic tool to check whether PCA works well on the data or not.

```
# the factoextra allows us to easily plot our eigenvalues!
scree <- fviz_eig(pizza.PCA, addlabels = TRUE)
scree
```



To analyze a Scree Plot, look for the "elbow" or point where the curve flattens for the optimal number of components to retain. For our pizza PCA, we can see an "elbow" point at 3 components (or dimensions).

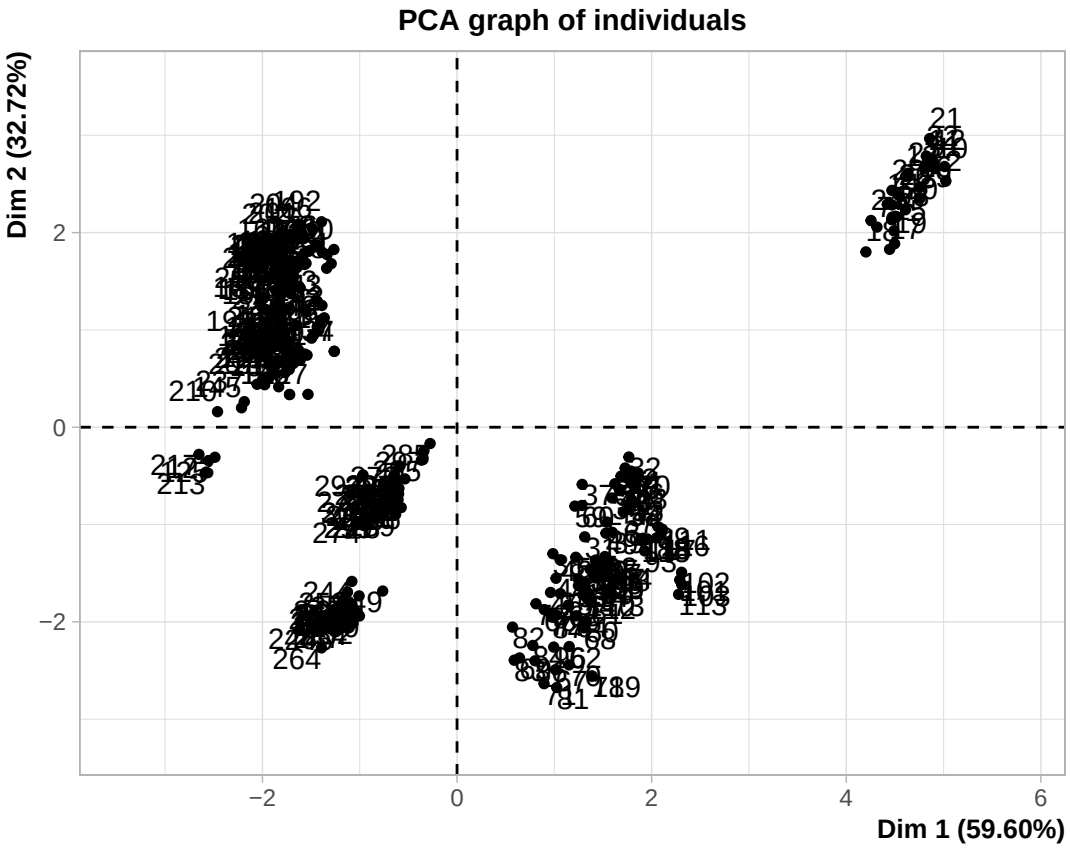
Looking back at our eigenvalue table, we can see that at three PCs, we can account for **98.24%** of the variation in the data set!

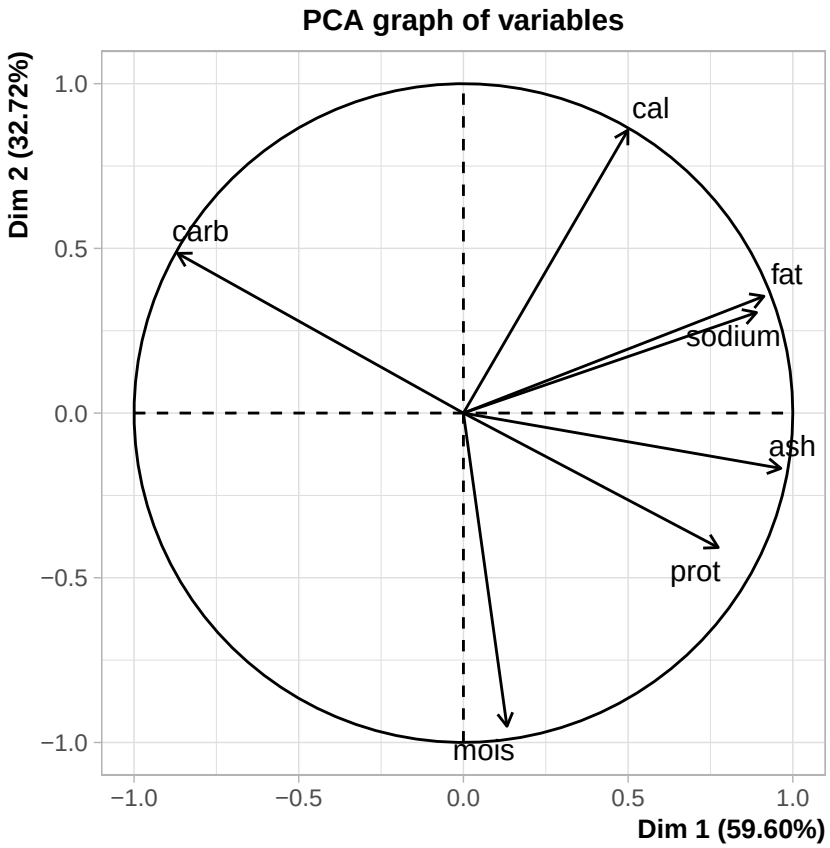
That is what we choose for our analysis in this demo!

Re-Performing PCA Now that we have an understanding of how many principal components are necessary to explain the ideal amount of variation in our data set, we can re-perform our PCA analysis, but this time specifying the number of dimensions (number of dimensions = number of principal components) to keep (*ncp*).

This time, we will set our graph to TRUE.

```
# Make sure to set scale.unit to TRUE if your data isn't standardized!
PCA <- PCA(pizza[,-1], ncp = 3, scale.unit = TRUE, graph = TRUE)
```


PCA graph of individuals
Dim 2 (32.72%)
Dim 1 (59.60%)
2
0
−2
−2
0
2
4
6




```
PCA
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 300 individuals, described by 7 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

PCA graph of variables

This graph is also known as a *Variables Factor Map* or a *Circle of Correlations*. This plot provides us with

information about the relationship between variables in the data. Specifically, it shows the strength and direction of the correlation between each variable and each factor (or principal component).

We can analyze the correlation of variables with the following:

- Positively correlated variables are grouped together.
- Negatively correlated variables are positioned on opposite sides of the plot origin (opposed quadrants).
- The angle between two variable vectors indicates the degree of correlation between those variables.
- The distance between variables and the origin measures the quality of variables on the factor map. Variables that are away from the origin (closer to the circumference) are well represented on the factor map while variables that are close to the origin are not well represented by the PCs.

In this graph we can see that:

- `sodium` and `fat` are positively correlated because the angle between the two vectors is very small (they are grouped together). Also `prot` and `ash` have a high positive correlation.
- `cal` and `carb` or `cal` and `prot` are most likely not correlated, because these pairs have 90° angles between the vectors.
- `prot` and `carb` are negatively correlated, as the vectors point into totally opposite directions (180°).
- `carb` and `cal` are the two variables that are best presented on the factor map by the PCs because they are closest to the circumference, `prot` seems to be the variable that is the least represented by the PCs (shortest vector length).

PCA graph of individuals

This graph is also known as a *Individuals Factor Map*. It provides us with information about the relationships between individuals and identifies patterns or clusters.

We can analyze the individuals within our reduced dimensional space with the following:

- Clusters of individuals that are close together may indicate they share similar characteristics.
- Outlier individuals are far away from other points on the map– this may indicate that they may have unusual characteristics.

In this graph we can see that:

- there are most likely 6 clusters in the data set.
- the data set does not contain any outliers with unusual characteristics far away from the majority of observations.

We will further analyze our PCA output in the next section.

PCA Analysis

Components of a Factor Map

We will break down which component goes into a Factor Map.

As seen above, we can create a factor map of *variables* and *individuals*. Below, we will explore how to get the components necessary to create a factor map, how to analyze the factor map, and how to customize it.

Note: Variables are also known as your columns or features; individuals are known as your rows or observations.

Variables We will start by creating the variables‘ factor map. Note that the components used for factor map of individuals and variables are the same type of components.

We can extract the results for the variables with the **get_pca_var()** function.

```
var <- get_pca_var(PCA)
var
```

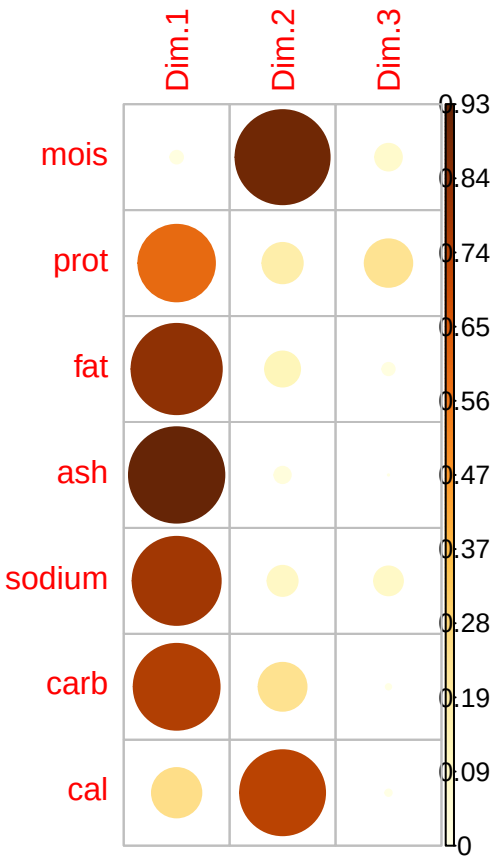
```
## Principal Component Analysis Results for variables
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates for the variables"
## 2 "$cor"     "Correlations between variables and dimensions"
## 3 "$cos2"    "Cos2 for the variables"
## 4 "$contrib" "contributions of the variables"
```

These components can be used in the plot of variables factor map where,
* var$coord$: $coordinates of variables to create a scatter plot$ * varcos2: represents the quality of representation for variables on the factor map. It's calculated as the squared coordinates: var.cos2 = var.coord * var.coord * var$contrib: contains the contributions (in percentage) of the variables to the principal components. The contribution of a variable (var) to a given principal component is (in percentage) : (var.cos2 * 100) / (total cos2 of the component)

All of the above variables will produce the same factor map but can be used to analyze our PCA in different ways.

Quality of Coordinates The *var$cos2* component tells us the quality of representation of each variable on each principal component–

```
# Get the cos2 - quality of coordinates on the factor map
#var$cos2

# correlation plot
corrplot(var$cos2, is.corr=FALSE)
```



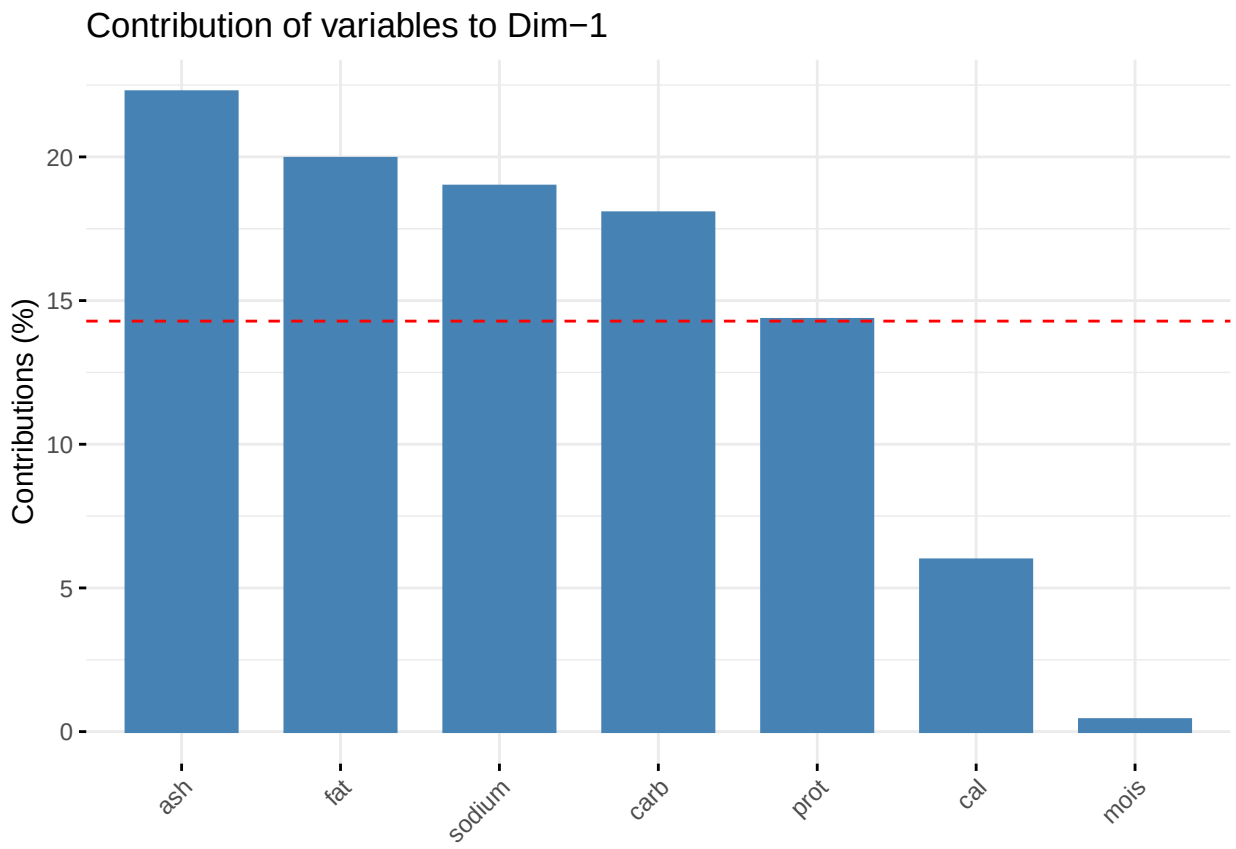
A high cos2 value indicates a good representation of the variable on the corresponding principal component, while a low cos2 indicates the variable is not perfectly represented by the principal component.

The plot shows for our data set that:
* `prot`, `fat`, `ash`, `sodium` and `carb` are very well represented by the first PC
* `mois` and `cal` are very well represented by the second PC
* and a little bit of the `prot` information is captured in the third PC

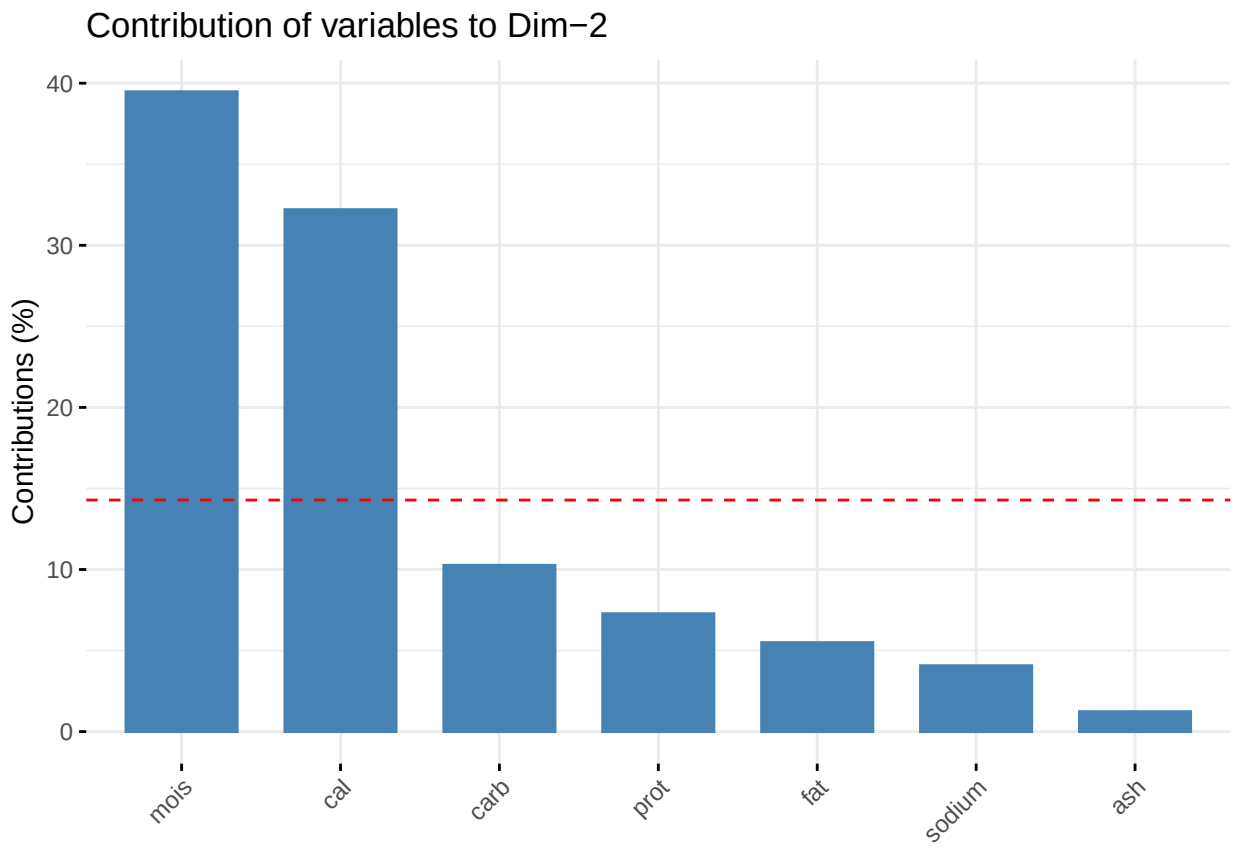
Contribution of Variables The *var$contrib* component tells us the importance of each variable in explaining the variation in the data captured by the principal components (or factors).

The variables that are correlated with PC1 (principal component 1) and PC2 (principal component 2) are the most important in explaining the variability in the data set.

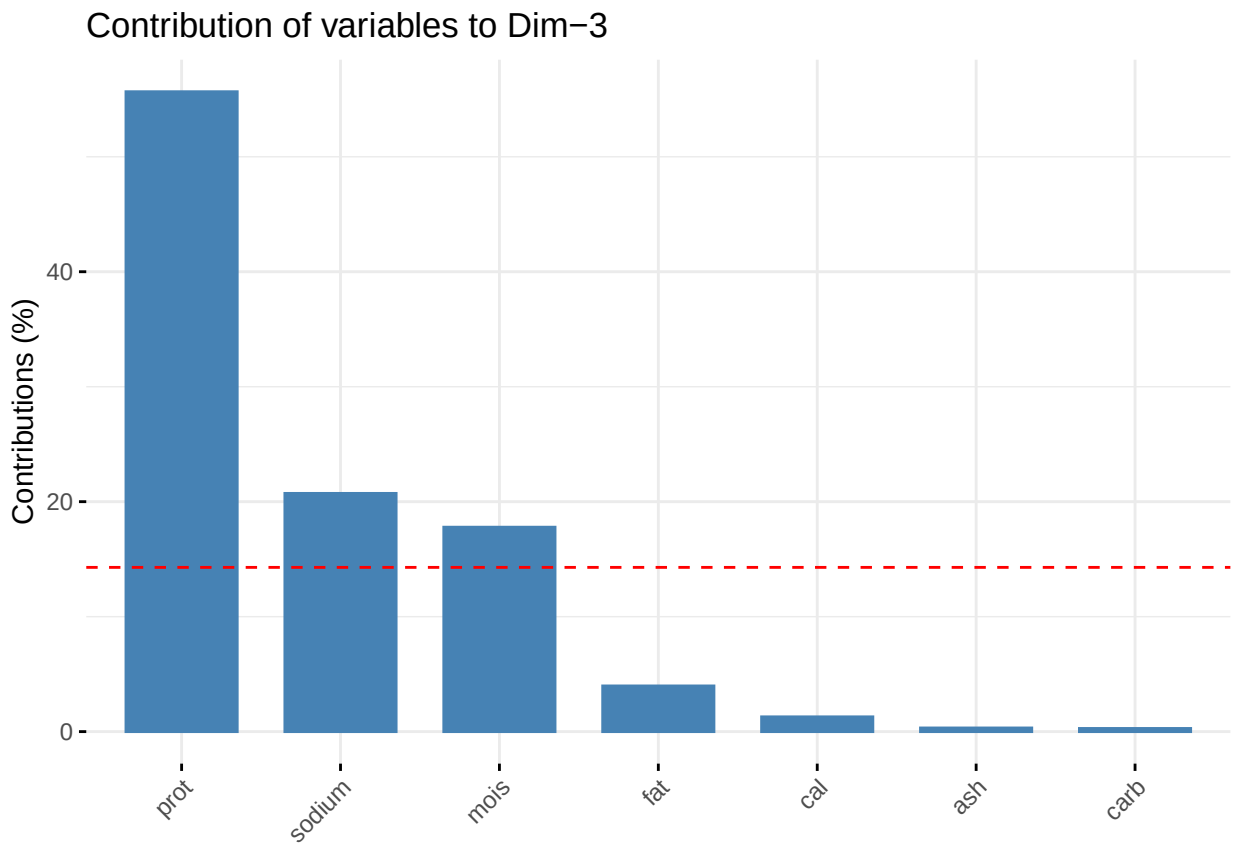
```
# Get the contrib - contribution of variable to the PCs
#var$contrib

# Contributions of variables to PC1
fviz_contrib(PCA, choice = "var", axes = 1)
```




```
# Contributions of variables to PC2
fviz_contrib(PCA, choice = "var", axes = 2)
```




```
# Contributions of variables to PC3
fviz_contrib(PCA, choice = "var", axes = 3)
```



The red dashed line on the graphs above indicates the expected average contribution. If the contribution of the variables were uniform, the expected value would be $\frac{1}{7} = \frac{1}{7} = \mathbf{14.29\%}$. For a given component, a variable with a contribution larger than this cutoff would be considered as important in contributing to the component, while components lower than this cutoff can be further examined to determine if the variable is worth keeping.

The graph with respect to PC1 (Dim1) shows:
* `ash`, `fat`, `sodium`, `carb` and `prot` have a high contribution towards PC1. We could already see this fact above in the cos2 correlation plot as well.

The graph with respect to PC2 (Dim2) shows: * `mois` and `cal` have a high contribution towards PC2 (as already mentioned i the cos2 correlation plot).

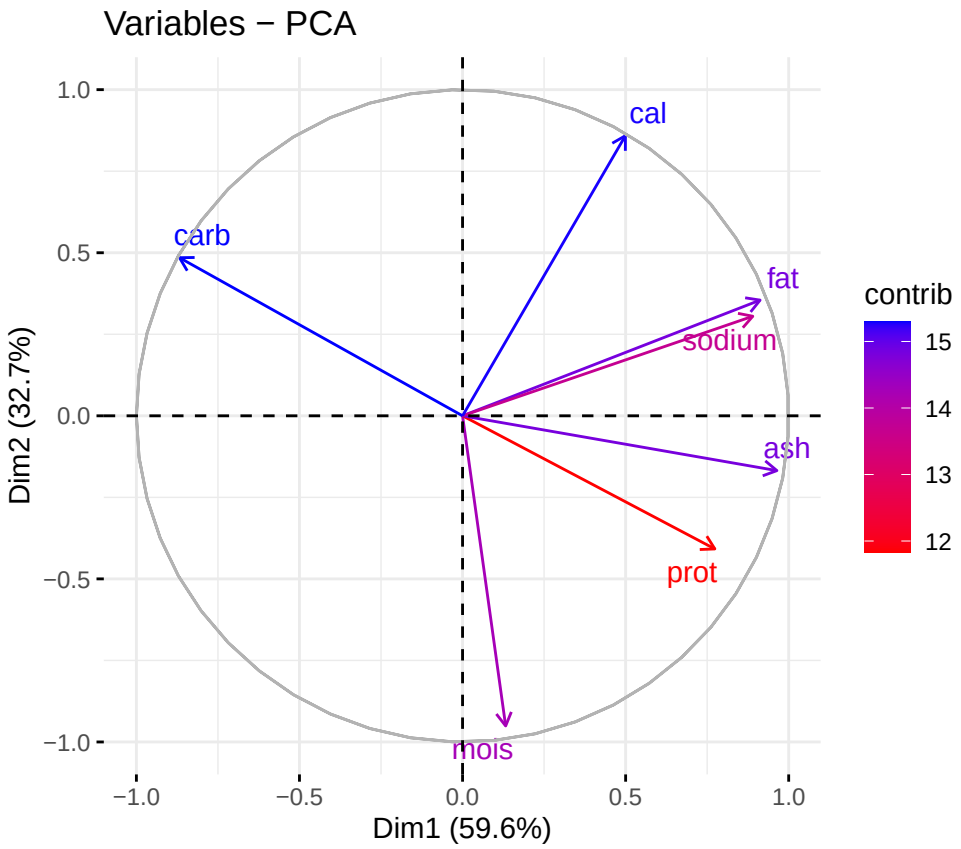
The graph with respect to PC3 (Dim3) shows: * `prot`, `sodium` and `mois` have a contribution towards PC3 that is higher than the expected one.

In general, these two plots tell us, that the pizza data is very well represented by 3 dimensions (3 Principal Components).

Variables Factor Map This time we are going to create a Factor Map where each variable is color-coded for the amount of contribution it has to the principal components. We can create multiple of these Factor Maps for different pairs of principal components and see how the relationship between variables change.

```
# plot the coordinates in relation to the PC
# axes specify which PC should be considered

# Factor Map of PC1 and PC2
fviz_pca_var(PCA, axes=c(1,2), col.var = "contrib",
```

```
             gradient.cols = c("red","blue"),
             repel=TRUE # Avoid text overlap
             )
```



This Variable Factor Map is the same as the one we saw in our output from our PCA() function, but this time it is colored by the contribution percentage. With reference to our analysis steps above, we can make insights on our pizza variables in our data set.

The graph above shows that:
* `carb` and `cal` overall have the highest contribution towards the first 2 PCs
* They are followed by `fat` and `ash` * Next highest are `mois` and `sodium`
* The lowest contribution towards PC1 and PC2 has `prot` (which makes sense because `prot` also has the highest contribution of all variables towards PC3).

Individuals We can extract the results for the individuals with the **get_pca_ind()** function. These are the same components as we saw above!

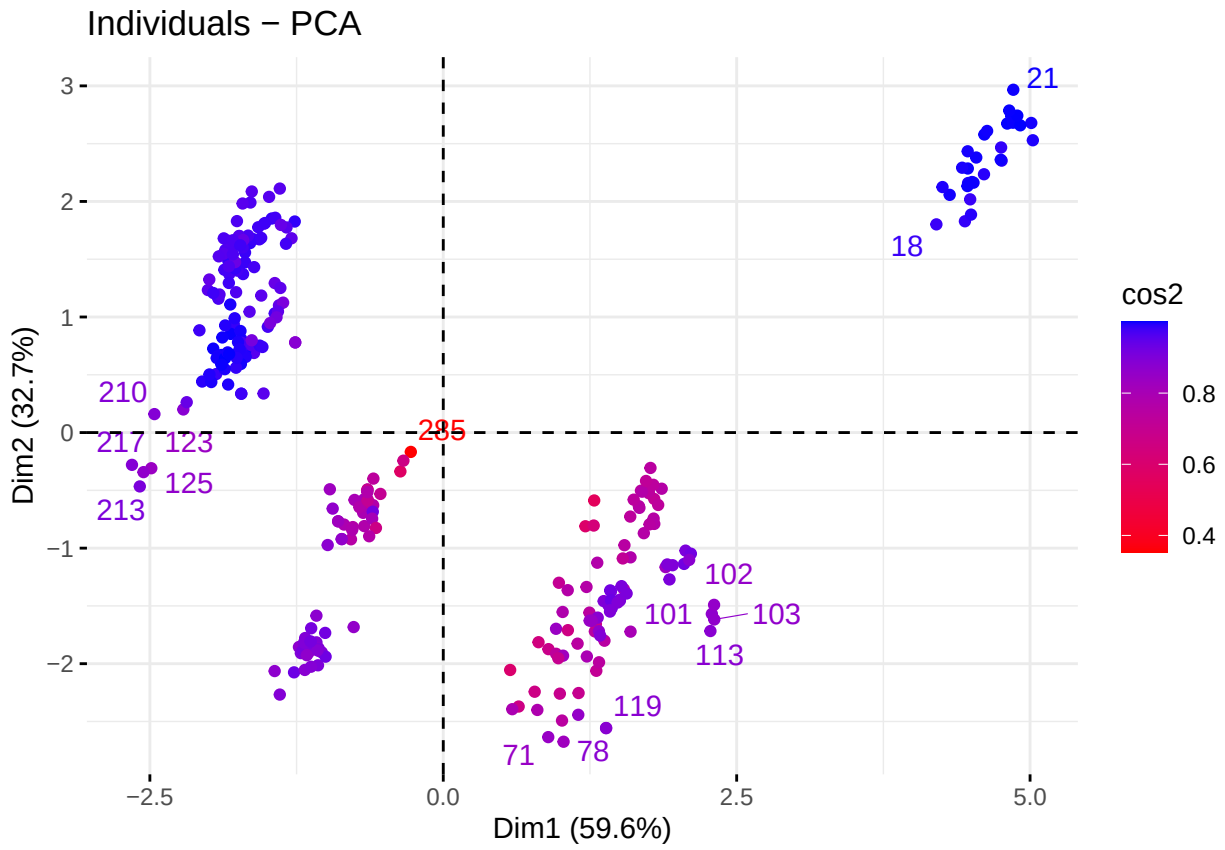
```
ind <- get_pca_ind(PCA)
ind
```

```
## Principal Component Analysis Results for individuals
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates for the individuals"
## 2 "$cos2"    "Cos2 for the individuals"
## 3 "$contrib" "contributions of the individuals"
```

These components can be used in the plot of variables (Factor Map) where, * ind*coord* : *coordinatesofindividualstocreateascatterplot* *ind*cos2: represents the quality of representation for individuals on the factor map. It's calculated as the squared coordinates: ind.cos2 = ind.coord * ind.coord * ind$contrib: contains the contributions (in percentage) of the individuals to the principal components. The contribution of a individual (ind) to a given principal component is (in percentage) : (ind.cos2 * 100) / (total cos2 of the component).

Like variables, we can visualize our coordinates on a factor map, and this time we will color individuals by their quality of representation (cos2):

```
# Factor Map of PC1 and PC2
fviz_pca_ind(PCA, axes=c(1,2), col.ind = "cos2",
             gradient.cols = c("red","blue"),
             repel=TRUE # Avoid text overlap
             )
```



This Individual Factor Map is the same as the one we saw in our output from our PCA() function, but this time it is colored by the quality of representation. With reference to our analysis steps above, we can make insights on our individuals of our pizza data set.

Remember: A high cos2 value indicates a good representation of the variable on the corresponding principal component, while a low cos2 indicates the variable is not perfectly represented by the principal component

In general we can see, that most of the pizza samples (individual observations) have a very high quality of representation by the first two principal components (blue colored dots). Only few observations (like for example 285) are relevant to justify more than 2 PCs.
In summary, one could conclude from this colored plot, that there are 5 or 6 clusters within the pizza data set. Let us find out if these clusters can be used to determine the brand of pizza.

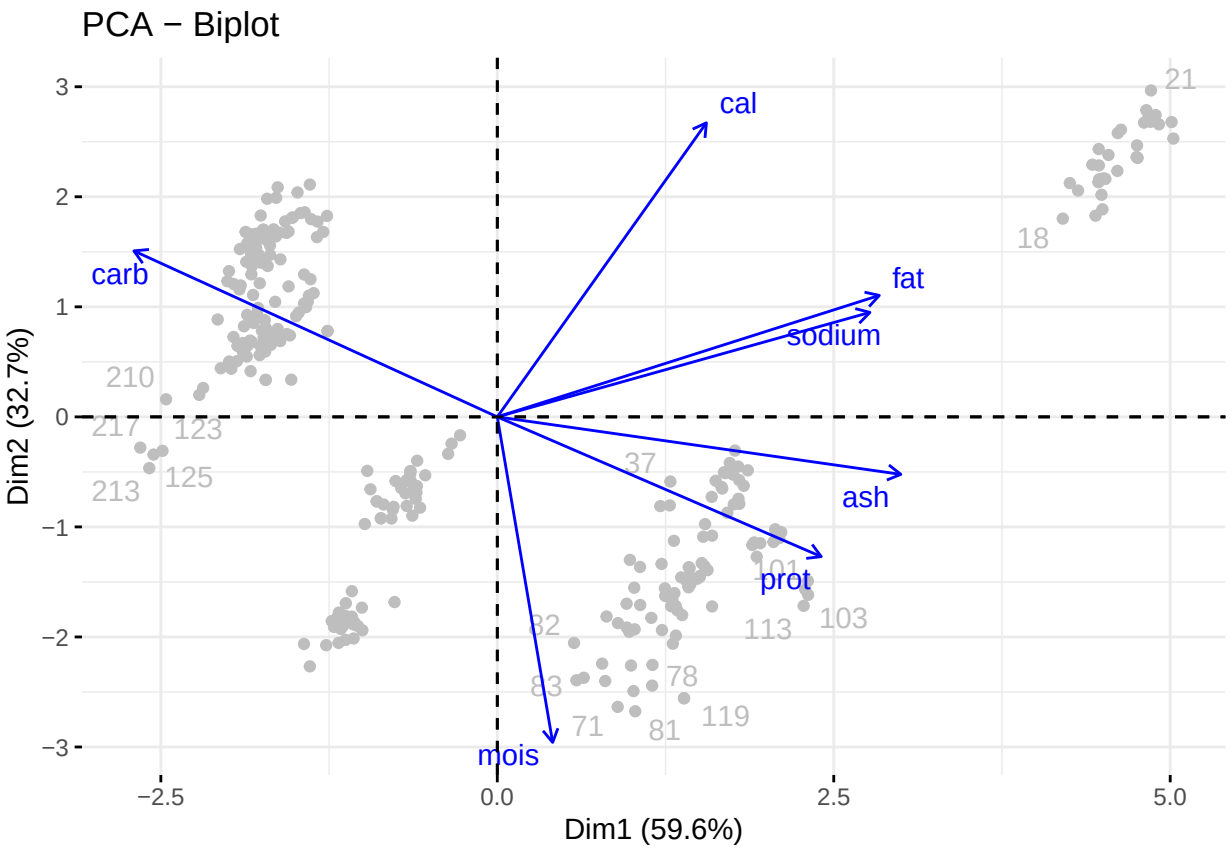
Bi-Plot

There will be instances where we don't want to look at only our variables or individuals. This is where the **Bi-Plot** comes in.

The Bi-Plot is a type of graph that is used to simultaneously visualize and analyze the relationship between variables and individuals against the principal components.

To create a Bi-Plot, we can use the $fviz_pca_biplot()$ function–

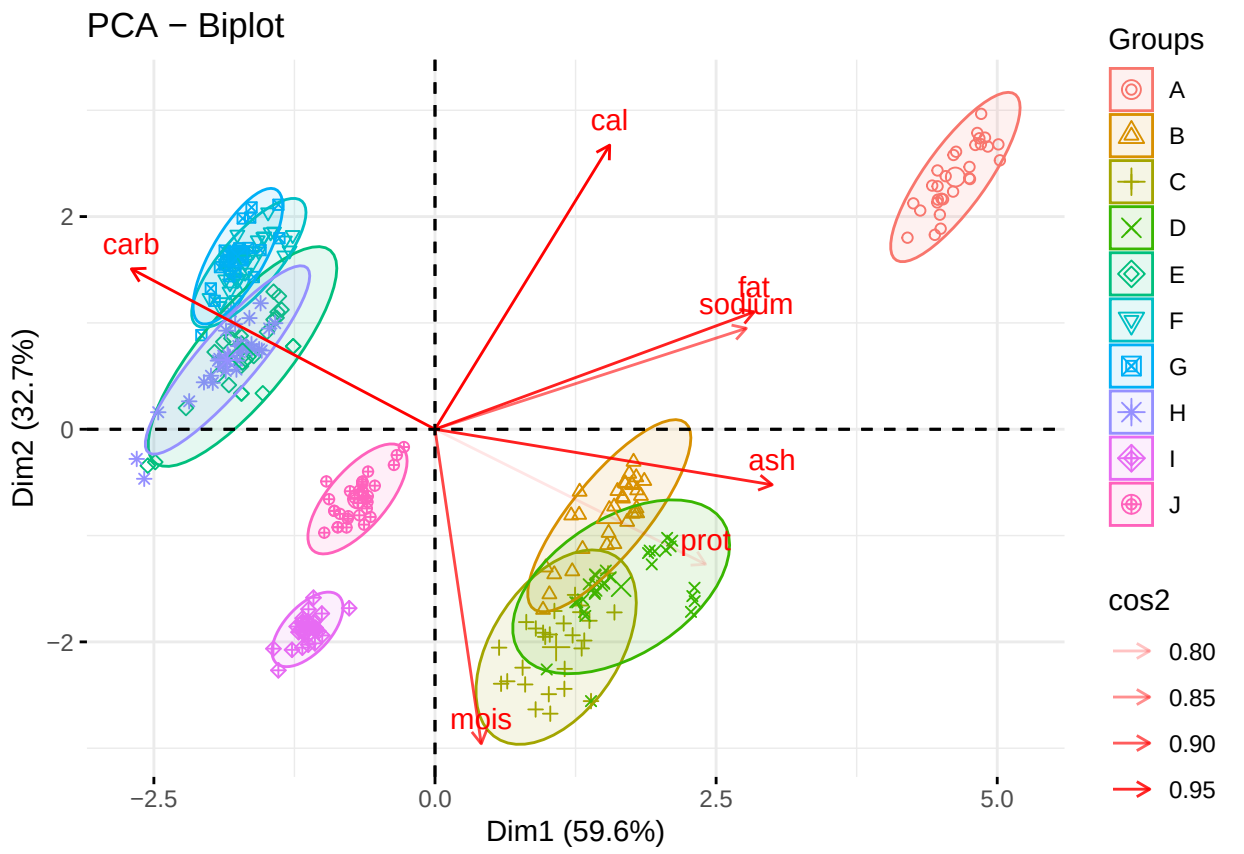
```
fviz_pca_biplot(PCA,  repel = TRUE,
                col.var = "blue", # variables color
                col.ind = "grey" #individual color
                  )
```



In a Bi-Plot, if our data has a column that represents a type of group, we can change the colors of our individuals to represent its group. We can also add ellipses to further visualize each group.

In our pizza data set, we had the `brand` column that we didn't include in our PCA(), but we can now use that data to show clusters of individuals that fall into a particular brand. We can also use the quality of representation by setting *alpha.var* variable to cos2.

```
# the habillage argument allows us to choose the column with the group data
# the habillage argument MUST be a factor type! (we did this at the beginning)
fviz_pca_biplot(PCA,
  habillage = pizza$brand, addEllipses = TRUE,
  col.var = "red", alpha.var ="cos2",
  label = "var")
```



This Bi-Plot above summarizes our pizza data set very well and contains almost all of the relevant information.

- It shows the single pizza samples / observations as shapes (circles, triangles,...) in the plane formed by two Principal Components.
- Like for any scatter plot, we can look for patterns, clusters and outliers and see, that the 10 different pizza brands (A to J), are nicely clustered within this plot (even though without color coding (unsupervised) we interpreted 5 to 6 clusters). The reason for that is, that some of these brand clusters are very close together in the plane.
- It is also visible that the data set does not contain any terrible outliers far away from its "actual" cluster. Almost all of the data points are within the cluster ellipses.
- Additionally, this Bi-Plot shows the original variables as vectors (red arrows). They begin at the origin and extend to coordinates given by their variation representation within the first two Principal Components.
- `cal` and `carb` are best represented by the first two PCs as their vector length is the biggest. This means most of the variability of these two variables is captured within PC1 and PC2.
- `fat` and `sodium` are the two highest positively correlated variables as they are grouped together very closely. `cal` and `carb` are barely correlated which can be seen by their 90° angle. `prot` and `carb` have the highest negative correlation (the vectors point in total opposite directions).
- Another topic that the Bi-Plot shows us is, that `ash` mostly contributes to PC1 (dimension 1) as it is

the variable that is the most 'parallel' towards the X-Axis (PC1). With respect to PC2 (dimension 2) that is `mois` because `mois` is the variable that stretches most along the Y-Axis (PC2).

Overall, we can say that PC1 refers to the richness in fat, sodium, ash and protein, and the lack of carbohydrates. PC2 refers to the richness in calories and the lack of moisture.

Conclusions

PCA is useful in any situation where you have many variables that are correlated to some degree, and you would like to reduce the dimensionality, that is, be able to use fewer variables. Reducing the dimensionality helps to identify patterns among the variables and to identify commonalities among the rows (or objects). This is true for our pizza data set, where we started with 7 numeric variables, and were able to explain ~98% of the variation in our data by using only 3 variables (3 PCs). With PCA, we could see which variables were correlated with one another in our Factor Map of Variables; though this was easy in our correlation plot, if we had more variables, it would have been much more difficult.

In summary, if we conduct PCA and end up with too many PCs (more than 3), PCA might not be the best way to visualize the data. Instead, other dimension reduction techniques such as t-SNE or MDS should be considered.

To answer our research question, which brand would be the best pizza brand looking at the nutritional information, we can conclude the following:

- If fatty and crispy pizza is your choice, brand A might be a good option.
- If soft and non-fat pizza is your choice, brand I might be a good option.
- If a good balance of nutrients is your choice, brand B and J might be good options. However brand B pizza might be more fat-based than brand J, on the other hand brand J might be more carbohydrate-based than brand B.