
# Unsupervised Learning: Principle Component Analysis and Multidimensional Scaling

## STA5077Z - Unsupervised Learning

**Due: 2019/08/15**

Chris Maree - MRXCHR013

**Abstract**

This paper presents multidimensional scaling (MDS) and principle component analysis (PCA) as two methods for conducting unsupervised learning to identify patterns and visualize hidden information in higher dimensional data sets. MDS was performed on UN country data to identify different ways to quantify a countries level of development. A one dimensional solution was presented to rank countries aggregate development. Two dimensional representations were also performed which included a selection of different clustering algorithms to group countries. PCA was used to analyze national track record data. This process involved outlining a number of different visualizations to reveal hidden patterns within the dataset and represent the information in two dimensions. Additionally, a Cos2 analysis was performed to quantify the quality of the model. A detailed Code appendix can be found at the end of this paper which highlights all the techniques used. Source code can also be found on Github here.

## Contents

# 1 Introduction

Unsupervised learning aims to discover patterns and grouping associated with features which have no defined response variable. This is in contrast to supervised learning which aims to predict a response variable using a set of features. A resulting challenge with unsupervised learning is that as there is no defined response variable it is hard to quantify the quality of the model through test sets.

Some common use cases of unsupervised learning include identifying market segmentation to better target appropriate customers, identifying fraudulent banking behaviour, segmentation of images, gene clustering and deriving climate change indices based on measurements. This paper will examine analysis of demographic and athletic data to find structure and patterns in the underlying data.

A key part of unsupervised learning is dimensional reduction which is the process of reducing the number of variables under consideration within the model by obtaining a set of principle variables. Two prevalent techniques for preforming dimensional reduction are considered within this paper, namely Principle Component Analyses (PCA) and Multi Dimensional Scaling (MDS).

This report is broken into three main sections. First, the theory behind PCA and MDS is explored and the two techniques are compared and contrasted. Second, MDS is applied to a data set of national demographic factors from the United Nations (UN) to try and identify patterns and clusters within the data. In the third and final section PCA is applied to a data set of national women's track records to analyze patterns within the data.

# 2 Dimension Reduction Techniques

Data sets in high dimensions are difficult to visualize and are taxing to preform computation over when building machine learning models. Dimension reduction aims to transform high dimensional feature spaces to lower dimensional spaces while preserving as much information as possible. This process reduces the time and space required when working with datasets while removing multi-collinearity between variables. From a visualization perspective dimension reduction reveals patterns and relationships within data that are obscured in higher dimensions. For example a 10th dimensional data set can be transformed into a 2D or 3D graph through the extraction of principle components and then plotted revealing patterns within the data that were hard to visualize in the original higher dimension.

## 2.1 Principle Component Analysis

PCA is a dimensionality-reduction technique that is used to transform high dimension data sets into lower dimensional sets while preserving as much variance as possible. PCA uses variance within the data as a proxy for information. The decrease in dimension does result in a loss of information as some of the variance is left in principle components that are not selected. This is normally acceptable as the first few principle components contain the vast majority of the variance of the set, meaning that you can throw away many of the other principle components without loosing too much information.

PCA finds a low-dimensional representation of high dimensional data that contains as much variance as possible. This low-dimensional representation is a linear combination of the higher dimensional space. This low-dimensional representation makes up the principle components which are created such that they are uncorrelated with each other and as much of the variance of the original data is expressed within the first few components. To ensure maximum uncorrelation between each principle component each principle component is positioned perpendicular to all other principle components.

## 2.2 PCA Computation

The process of finding the principle components can be broken down into a number of key steps which are discussed below. When PCA is performed in R these steps are abstracted away from the user.

### PCA step 1: Data Preprocessing

Before PCA can be preformed the data needs to be standardized. This is vital for PCA as non-standardized data will have larger variance depending on the units which will in turn skew the principle components. Numerically this is done by subtracting the mean from each data point and then dividing by the standard deviation. This can be seen by the simple Equation 1 below.

$$x_{new} = \frac{x - \mu}{\sigma} \tag{1}$$

### PCA step 2: Generation of Covariance Matrix

Once the data points have been standardized the next step is to find the variance within the set by computing the covariance matrix. This reveals relationships between data points and ultimately shows values that are highly correlated in such a way that the points contain redundant information that are embedded within other values. The covariance matrix is a $p \times p$ matrix for $p$ predictors and stores covariances for each pair of variables in the set. Covariance of two variables $X$ and $Y$ can be calculated using Equation 2 below.

$$Cov(X, Y) = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{x})(Y_i - \bar{y}) \tag{2}$$

For a 3-dimensional data set with variables $x$,$y$ and $z$ the covariance matrix would be calculated as shown in Matrix 3 below with the covariance calculated between each variable pair.

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix} \tag{3}$$

As covariance of a variable with itself is variance and because covariance is commutative this matrix is symetric along the main diagonal. The polarity of the covariance values within this matrix show if the variables are positively or negatively correlated with one another and the magnitude represents the strength of this correlation.

### PCA step 3: Eigenvalue Decomposition

As the covariance matrix is diagonalizable, eigenvalue decomposition can be performed to generate eigenvectors and eigenvalues. These together are used to determine the *principle components* of the data. There will be one set of eigenvectors and eigenvalues for each input dimension. The eigenvectors of the covariance matrix define the direction of the axes where there is the most variance within the data. These eigenvectors are therefore the principle components of the data set. The eigenvalues are the coefficients attached to the eigenvectors and they provide the amount of variance that each principle component carries.

As a result of the eigenvalues representing the amount of variance expressed by each principle component, the principle components can be ordered by magnitude of eigenvalues to identify which principle components express the most variance of the original set. Through this ordering process a *scree* plot can be generated which in turn can be used to identify how many principle components should be included in the final model based off the desired amount of variance that should be explained in the lower dimensional space. This ability to express the majority of variance of a higher dimensional space within a few principle components is the key behind PCA dimensionality reduction.

From a more mathematical perspective, for a given square matrix $A$, a vector $v$ and a scalar $\lambda$ that satisfies the equation $Av = v$ then $\lambda$ is known as a eigenvalue assiciated with the eigenvector $v$ of A. From this the eigenvalues of A are found as the roots of the characteristic equation defined in equation **??** where $I$ is the identify matrix.

$$det(A - \lambda I) = 0 \tag{4}$$

**Step 4: Generating the Feature Vector**

As each principle component's contribution can now be quantified through the magnitude of its associated eigenvalue, a small subset of the principle components can be chosen which have the largest contribution to overall variance. These selected eigenvectors are combind together within a matrix of vectors to form a *feature vector* ($W$) which represents the principle components which were chosen and not discarded.

The number of principle components that are selected and included within the feature vector is dependent on how much variance one wants to express in the lower dimension and is normally chosen by examining the scree plot. This number is normally refered to as $k$.

**Step 5: Relocation of Data Along Principle Component**

Lastly the original data is recast to be represented along the principle component axes. This is done through a multiplication of the feature vector ($W$) and the original standardized data set ($\bar{A}$) after transposing the matrices. This can be seen in Equation 5 below.

$$\begin{aligned} FinalDataSet &= FeatureVector^T \times StandarizedOriginalDataSet \\ y &= W' \times \bar{A} \end{aligned} \tag{5}$$

At this point the initial data set has been reduced to a lower number of dimensions based on the number of principle components selected in the feature vector. The final data is a linear combination of the original variables.

## 2.3 Multidimensional Scaling

MDS is a visualization technique used to represent distances between objects in higher dimensions within a lower dimensional space. The goal in this transformation is to find a mapping wherein the distances between points in the final image map are as similar as posible to the distances of the corresponding data in the original higher dimensional space. Through this transformation process MDS acts as a form of non-linear dimensionality reduction.

Two main variants of MDS are *metric* and *non-metric* MDS. Metric MDS (also called classical MDS or principal coordinates analysis) aims to find a mapping which approximates the original inter-sample dissimilarities as closely as possible. This means that the fitted distances on the MDS map and the original distances from the higher dimentional space are in the same metric. Lastly, Non-metric MDS can handle qualitative data while metric can not.

Non-metric MDS (also called ordinal MDS) aims to find a mapping which shows the rank order of distances between samples, which match the rank order of the given dissimilarities in the higher dimensional space. In other words non-metric MDS considers the value of distance in relation to the distances between pairs of objects in the higher dimensional space. For example, if the distances between two objects are rank fourth in the higher dimensional space then they should be rank fourth in the lower dimensional space.

## 2.4 Multidimensional Scaling Computation

The process in preforming MDS is very similar to that of PCA in terms of using eigenvalue decomposition and the generation of a feature vector. Where MDS differs is at the input layer wherein MDS inputs the relative distances between samples instead of PCA's use of a correlation matrix. This relative distance is expressed through the use of a dissimilarity matrix which represents the distances between pairs of objects. The remaining steps outlined before can be used in almost the exact same way to compute a MDS transformation.

### Distance Computation

MDS considers relative distances between objects in a higher dimensional space. As a result, the method used to calculate this relative distance is important to consider. There are a number of different techniques available and the kind of distance equation used will differ based on the domain context of the problem. A number of popular techniques for continuous variables are; Euclidean distance shown in equation 6, Manhattan distance shown in equation 7 and log fold distance. MDS can also be applied to categorical variables by considering the proportional differences between objects or the Spearman Rank.

$$d_{euc} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{6}$$

$$d_{man} = \sum_{i=1}^{n}|(x_i - y_i)| \tag{7}$$

Other kinds of dissimilarity measurements can be used such as correlation-based distances which are used in gene expression data analyses. This method quantifies distance by subtracting the correlation coefficient from the euclidean distance. An example of this method is the Pearson correlation distance.

## 2.5 Principle Component Analysis vs. Multidimensional Scaling

Multi-dimensional scaling is very similar to principle component analysis, except that instead of converting correlations into a lower dimensional space, MDS converts distances among the samples in higher dimensions into a lower dimensional representation. Clustering based on minimizing the linear distances between points is the same as maximizing the linear correlations between points. As a result, preforming classical MDS with euclidean distance results in the same output as a PCA analysis.

Ultimately PCA's goal is to preserve variance through dimension reduction and MDS's goal is to preserve inter-point distances through dimension reduction. As a result, PCA is focused on dimensions themselves and seeks to maximize the explained variance while MDS is more focused on relations between scaled objects in higher dimensions.

# 3 Multidimensional Scaling Performed on UN Statistics Dataset

MDS was performed on the UN Statistics dataset. This dataset shows five economic and demographic factors for a sample of 25 countries and was generated in 1990. The key indicators for the dataset, their types and descriptions are shown in Table 1 below.

| **Variable Name** | **Variable Description** | **Type** |
|---|---|---|
| Country | Country name | factor |
| Increase | Annual percentage population growth rate | numeric |
| Life | Life expectancy in years | numeric |
| IMR | Infant mortality rate per 1000 | numeric |
| TFR | Total fertility rate | numeric |
| GDP | Gross Domestic Product per capita in US dollars | numeric |

Table 1: Provided data set variables and descriptions

## 3.1 Data Pre-Processing

The data set provided requires a bit of manipulation to be done on it before MDS can be considered. Firstly, it is converted into a matrix in R using the `as.matrix()` function. This conversion is done without the first column (country names) and then the names are added back to the matrix as `rownames`.

Before MDS can be performed the data needs to be normalized and scaled appropriately. This is important as if certain features in the data set have a larger range than other features, the distance metrics will be strongly dominated by the features with the larger ranges. This can be done easily in R using the `scale()` function applied to the matrix $X$. This was done in code in Appendix 1.1.

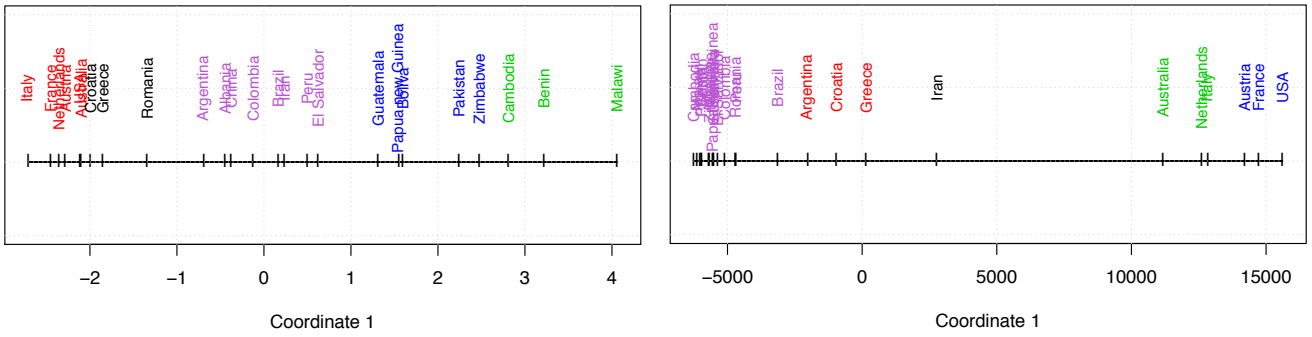
### 3.2 Multidimensional Scaling In R

MDS can be performed in R using a number of different functions and packages. `cmdscale` can be used to calculate classical (metric) MDS and is part of the `stats` package. `isoMDS` can be used to calculate Kruskals non-metric MDS and `sammon` can be used to calculate sammons non-linear mapping. Both `stats` and `sammon` are part of the `MASS` package. This report implements all three kinds of MDS to compare and contrast the outputs. This implementation can be seen in Appendix 1.2

### 3.3 One Dimensional Data Representation

The first analysis performed projects the 5th dimensional data set down to a one dimensional representation of the data by using MDS. MDS takes in a distance matrix as its input parameters. This is found using the `dist`function on the scaled input data. To achieve a one dimensional output in MDS the `cmdscale` function is used while setting `k=1` to instruct the function to construct a one dimensional representation of the data. This data can then be plotted by casting the results to a data frame and then plotting. In order to add visual separation to the data to make it easier to identify different groupings Hierarchical Clustering was also performed on the distance matrix and then grouped into 5 different clusters represented as colours. This implementation can be found in Appendix 1.4.

The output from this computation can be seen in the Figure 1 below. Figure 1a shows the normalized one dimensional output when the input has been normalized and scaled using the `scale` function. Figure 2b shows the same data set but without normalization. This second figure makes it clear to see how impactful large value ranges can be when conducting MDS with the overall ordering and distribution being heavily effected by the units in which the metrics are measured.




(a) One Dimensional MDS, Scaled

(b) One Dimensional MDS, Not Scaled

Figure 1: Output of One Dimensional MDS on US Statistics Dataset with 5 Colour Groupings

This one dimensional representation can be seen as a proxy for country development with highly developed countries grouped together on the extreme left hand side of the graph (red grouping) and lower developed countries grouped together on the extreme right of the graph (green grouping).

It is interesting that Italy is defined as the most developed country in the scaled version when considering the other developed countries on the list. This is due to the normalization of the data which results in the MDS acting to represent an aggregation of all higher dimensions in one dimension. Looking into the underlying values that make up the MDS, this positioning of Italy is due to Italy having a very low total fertility rate (TFR), negative population growth rate coupled with a high life expectancy rate results in the net positioning of Italy being higher than other developed countries. On the other hand Malawi is classified as the least developed country by far. This is due to them having the highest infant morality and fertility rate while having some of the lowest gross domestic product and life expectancy rates out of

all the countries.

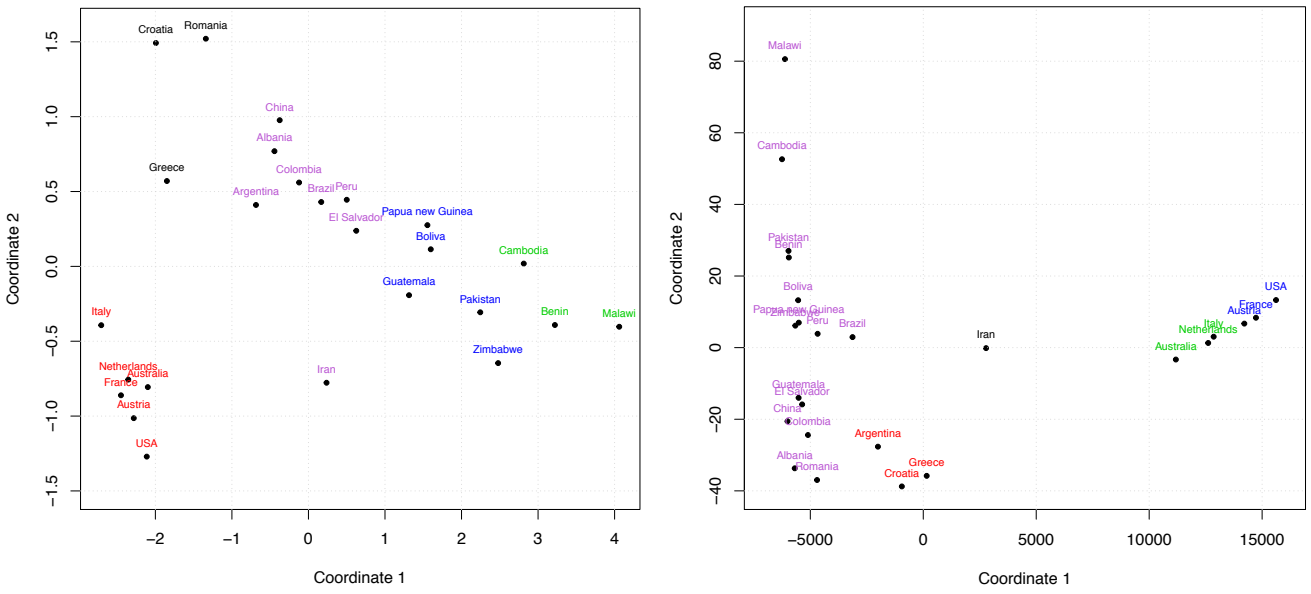
If the data set is not normalized then some variables dominate the overall representation of the countries due to the units. For example, the USA is then quantified as the most developed country. This is due to the USA having a higher GDP value than any other country. The less developed countries are now all grouped together and it becomes hard to distinguish between them. Again this is due to GDP having a large impact on the distribution of development.

Overall 1a acts as a good proxy for the development of a country as it considers all values within the set in a normalized way, meaning that values are equally considered. One can conclude that when all factors are equally taken into account countries Italy, France, USA and the Netherlands are quantified as the most developed. On the other hand the countries Malawi, Benin and Cambodia are the least developed.

It is important to note however that not all 5 variables included in this set should necessarily equally contribute to defining a countries development. For example, infant morality rate might be a better proxy for a countries development than say life expectancy. If this was the case a different scaling and standardization technique would need to be used.

### 3.4 Two Dimensional Data Representation

Next, MDS is performed in two dimensions as apposed to one. This is done in almost the exact same way as for one dimension, except that `cmdscale` now takes in a value of $k = 2$ to generate a two dimensional output. Again, clustering was performed using `hclust` to identify pattern groupings in the MDS output with 5 distinct groups identified. This implementation makes use of classical MDS scaling. This output can be seen in Figure 2a below. This implementation can be found in Appendix 1.3.




(a) Two dimensional MDS, Scaled

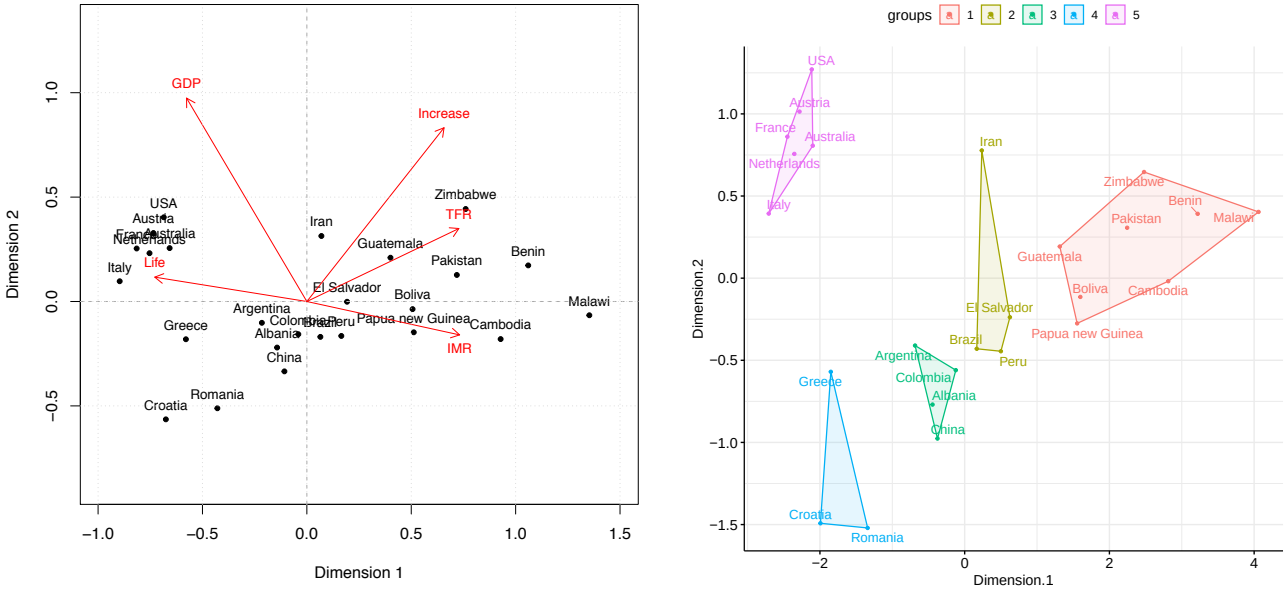
(b) Two dimensional MDS, Not Scaled

Figure 2: Output of Two Dimensional MDS on US Statistics Dataset with 5 Colour Groupings

Additional libraries were experimented with to compare and contract these results generated. One of which is the `smacof` package which provides a built in `mds` function. To use this function the same distance plot is fed in as an input parameter. This is a useful package when conducting MDS because it has a function `biplotmds` which enables the underlying variables to be super imposed on top of the two dimensional MDS plot. This can then be used to infer which variable contributed to each dimension in the MDS plot.

Interestingly the plot generated by the `smacof` was identical to that of the `cmdscale` function except that the dimension 2 is flipped. The underlying data represented is identical with the important informa-

tion within the relative distances being represented in the same way but just with a different orientation of the graph. This output can be seen in Figure 3a. The scaling in this plot however makes it quite hard to see the groupings of countries. Figure 3b shows the output of a MDS process performed using the `cmdscale` function as before, except this figure has had the second dimension flipped around to be consistent with the output of `smacof`. This figure also makes use of `kmeans` clustering as apposed to the hierarchical clustering used before. Lastly, `ggscatter` was used in the formation of this second figure to make the visualisation of the diffrent countries even clearer into the respective groupings. Through this grouping and visualisation Figure 3b enables clear distinctions between countries. This implementation can be found in Appendix 1.5 and Appendix 1.6.




(a) MDS plot including underlying variables

(b) MDS plot with kmeans cluster

Figure 3: Two Dimensional MDS plots following smacof dimension interpretation

The primary difference between Figure 3a and 3b is how the dimension scaling has been performed by the `cmd` function from `smacof` vs the `cmdscale` from the `stats` library. This is apparent when looking at the minimum and maximum magnitudes of the dimensions.

### 3.5 One Dimensional vs. Two Dimensional MDS Representations

The two dimensional representations of the country data, as seen in Figure 3b, does not show a direct comparison between different countries level of development as was the case with the one dimensional representation. Rather, this figure shows collective groupings bases on similarities (represented by distances in the higher dimensional space) between countries. Looking at the locations of countries relative to each other yields a number of interesting patterns which are explored below.

The relative position of countries along the first dimension in Figure 3b corresponds closely to the relative position in the one dimensional representation in Figure 1a. In Figure 1a the countries on the extreme left were Italy, France, Netherlands, Australia, Austria and the USA. These were classified as the most "developed" countries by the one dimensional representation. These countries can be seen grouped closely together in the purple group 5 kmeans cluster in Figure 3b. These countries have the largest negative value in the first dimension as well as a large magnitude in the second dimension. Following closely from these countries from the one dimensional representation are Croatia, Greece and Romania which were grouped in black in Figure 1a. These three countries have been grouped together in Figure 3b at the bottom left of the plot in the blue group 4. They have a large negative value of dimension 1 with a negative second dimension.

On the other side of the spectrum are the less developed countries. This can be seen as the red group

1 from Figure 3b. These countries corresponded to bottom end of the development spectrum in the one dimensional case in Figure 1a.

The first dimension in the 2D plot therefore represents the majority of the influence in the 1D plot. The second dimension in the 2D plot does have some influence but it is less profound than that of the first dimension. This can be more rigorously quantified by looking at the influence of each underlying variable, as is done in the next section.

### 3.6 MDS Dimension Interpretation

Figure 3a can be examined to understand the contribution to each dimension from the underlying variables. Specifically what each dimension is measuring and where the majority of it's influence originates from, can be determined by examining the angle and direction of each variable vector.

Dimension one is directly proportional to population growth, total fertility rate and infant mortality rate. It is inversely proportional to life expectancy and GDP. From these variables dimension 1 is mainly made up from total fertility rate, infant mortality rate and life expectancy as the angle between these variables and the horizontal line within the plot is lower when compared to GDP and population increase.

Dimension two, on the other hand, is directly proportional to all variables except for infant mortality rate. Dimension two is mainly made up from country GDP and population growth based off the angles of the underlying variables in Figure 3a.

Looking at Figure 3a's vectors and comparing them to the underlying data points shows the strong correlation represented in the MDS plot. For example the country with the highest TFR, IMR, high population increase and lowest life expectancy is Malawi. Malawi sits with the highest magnitude in the first dimension out of all countries. This corresponds to the significance of each underlying variable as expressed in this dimension. Strongly dominated direct relationships can also be identified with countries like Croatia which sit at the exact opposite direction and position to the population increase vector on the bottom left of Figure 3a. This suggests that Croatia should have a very low value of population increase and indeed when the underlying data is examined Croatia has by far the lowest value of population increase out of all countries represented.

## 4 Principle Component Analysis on National Track Records Dataset

In this section PCA is performed on the track records data set which shows the top national results for women in different distance races. There are a total of 55 rows in this data set with 8 predictor variables and one label variable. These can be seen in Table 4a below. As this section is examining PCA which considers variance within data, a correlation plot for the 8 predictors is also shown in Figure 4b to show how highly correlated the data set is. The top diagonal of this grid shows the correlation coefficients between each variable pair.

Figure 4b is useful in identifying how highly correlated the different countries are along times to finish each race. This is useful as it indicates that PCA will work effectively in this data set as the high correlation means that few principle components can be used to express much of the variance in higher dimensions.

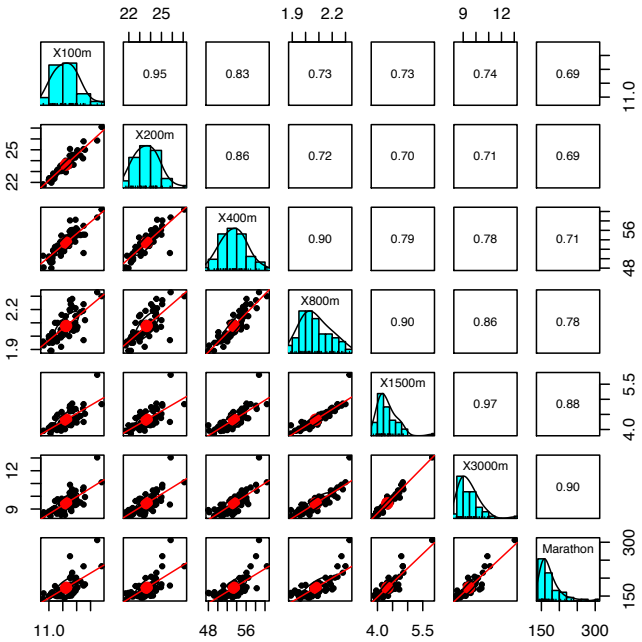
### 4.1 Data Pre-Processing

The data set provided has an observation index which was removed from the data before any processing was performed. Like with the MDS data set, the row names of the data are first stripped, then the data is converted into a matrix. Next, the matrix row names are assigned based off the original matrix.

Similarly to the process preformed when conducting MDS, the data must be scaled before PCA can be performed. This can be done easily using the `cor` parameter in the `princomp` function which, when

| Name | Variable Description | Type |
|---|---|---|
| OBS | Observation index. | integer |
| Country | Country name | factor |
| X100m | 100m race time in seconds | numeric |
| X200m | 200m race time in seconds | numeric |
| X400m | 400m race time in seconds | numeric |
| X800m | 800m race time in minutes | numeric |
| X1500m | 1500m race time in minutes | numeric |
| X3000m | 3000m race time in minutes | numeric |
| Marathon | Marathon race in minutes | numeric |

(a) Provided dataset variables and descriptions




(b) Correlation Plot for 8 Predictor Variables

Figure 4: Variable Descriptions and Correlation Plot For Track Record Data

set to true, first centres and scales the data set before the analysis. Note that if `prcomp` is used then the `scale` parameter can be set to normalize the data in the same way. This data pre-processing can be found in Appendix 2.1.

### 4.2 PCA In R

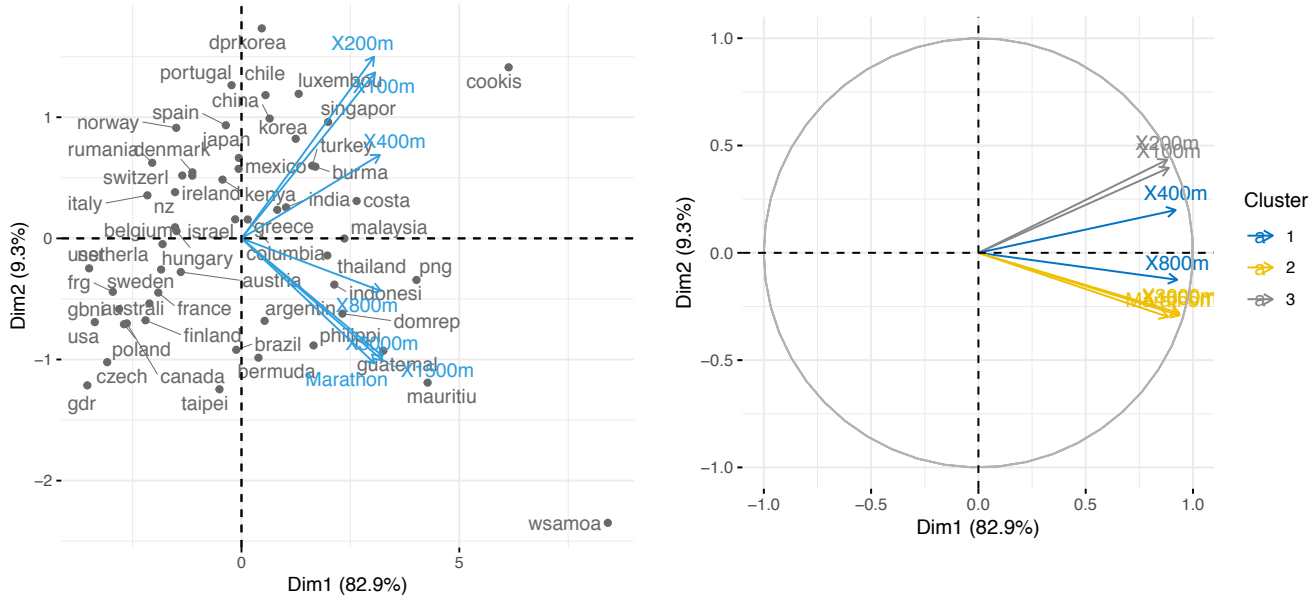
There are many libraries that can perform PCA in R. Some of the popular ones are `prcomp()` and `princomp()` which are both built-in R stats package, `PCA()` from the `FactoMineR` package, `dudi.pca()` from the ade4 package and `epPCA()` from the `ExPosition` package. This report mainly uses the `princomp` function to keep the implementation without the need for custom packages. This simple PCA implementation can be found in Appendix 2.2

### 4.3 Principle Component Generation

PCA can now be performed on the data set. A `biplot` can be created to represent both the principle component scores of samples and loadings of variables on the same visualization. This can be seen in Figure 5a below. An additional graph can be created to show the same principle components and the contribution of the underlying variables, but without the samples, to more clearly see the magnitude and direction of each variable with it's associated principle components. This can be seen in Figure 5b. This figure makes use of a `kmeans` clustering process to group the 6 underlying variables into 3 broad clusters. This implementation can be found in Appendix 2.3

Figure 5a clearly identifies two outliers based off their positioning along the principle components: `wsamoa` and `cookis`. These values are left in the data set. This figure also includes the percentage of explained variance between the first two principle components (82.9% and 9.3% respectively). More analysis is performed on this percentage variance as explained later in section 4.5.

Figure 5b reveals which variables contribute most to each principle component. From this figure it is clear that all variables are positively correlated with the first principle component. The second principle component is positively correlated with short distance races(X100m, X200m and X400m) and is inversely proportional to longer distance races (X800, X3000 and Marathon). This grouping is especially clear when




(a) PCA Biplot of Individuals and Variables

(b) Graph of variables with three groupings

Figure 5: PCA Biplot and Variable Grouping To Visualize Variable Contribution To Each Principle Component

the three cluster colours are considered between *short*, *medium* and *long* distance races corresponding to clusters 3, 1 and 2 respectively. This plot generation can be found in Appendix 2.4

### 4.4 Principle Component Contextual Interpretation

From the analysis above it was identified which principle components are proportional to which kind of races. However it is important to remember the context of the underlying numbers used in the analysis. The shorter the time taken to complete a race the better the country performed and so low values in the plots represent better country performances for that respective metric.

As dimension 1 is directly proportional to all metrics it is a proxy of overall country race times. As a result, countries with a negative value for dimension 1 (to the left of the 0 line) are above average in performance and countries to the right are below average in performance.

Dimension 2 is a proxy for a countries short or long distance prowess. Countries with a positive value for dimension two are worse off at short distances races and better off at long distance races. Likewise countries with a negative value for dimension two are better at short distance races and worse off at long distance races.

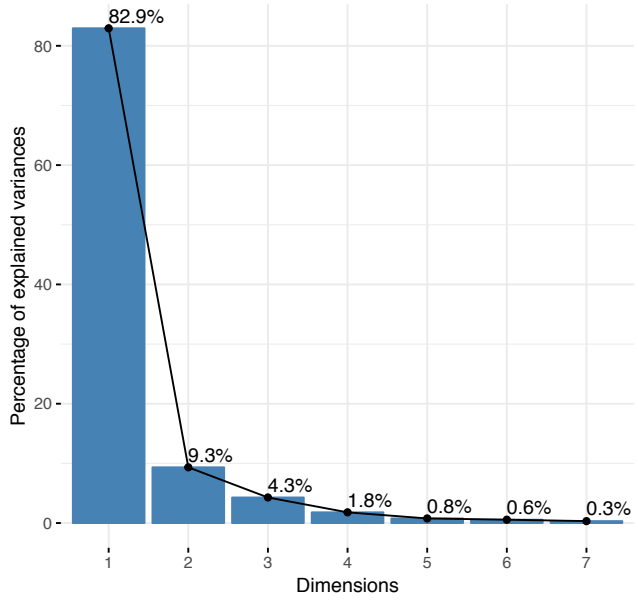
These relationships can be quantified by looking at the position of different countries and comparing their overall times for both short and long distances races. Countries with high values in dimension one are said to have worse overall performance. The two countries that clearly have the highest values in this dimension from Figure 5a are `cookis` and `wsamoa`. As is to be expected these two countries consistently come in the last 3 countries in all races, irrespective of distance. When looking at dimension two `cookis` should be better at long distance races and worse at short distance races when compared to `wsamoa` due to it's relative position in this dimension. Indeed this is the case when the underlying data is explored with `cookis` having the worst times overall for short distance races and `wsamoa` having worse times overall with long distance races.

The exact opposite behaviour can be examined with countries that perform well in the first dimension, represented by low relative Dim1 values. `gdr` for example has a very low value for Dim1 and a low value for Dim2. This indicates that `gdr` is overall a good performer with fast times specifically in the short race

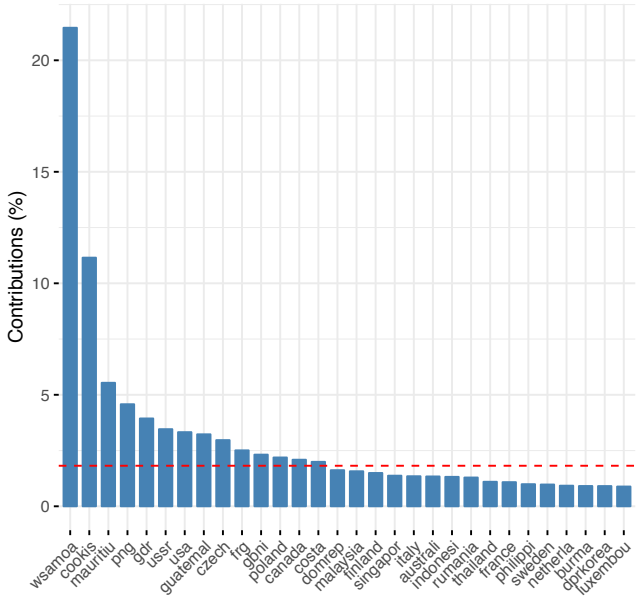
domain (due to negative Dim2). Indeed this is the case with `grd` placing in the top 2 for the X100m, X200m and X400m. However, the large negative value of Dim2 indicates that `grd` is not as fast at longer distance races. This is reflected in the underlying data where `grd` does not perform as consistently well placing in the top 10 in these longer races.

### 4.5 Principle Component Variance Analysis

Figure 6a is a scree plot which represents the total variance explained over each principle component within the set. From this figure one can identify the number of principle components that should be selected to form the total number of dimensions in order to express the desired amount of variance. From the scree plot it is clear that the vast majority of the variance is expressed in the first dimension (82.9%). This means that the 7th dimensional representation of the data can be expressed on one dimension, using only the first principle component, while still explaining 82.9% of the total variation from the higher dimension.




(a) PCA Scree Plot




(b) Variable Contribution to First Two Components

Figure 6: PCA Scree Plot and Variable Contribution Visualization

Figure 6b represents the contribution of the top 30 countries to the results of the PCA for the first two principle components. Impelemntation can be found in Appendix 2.5. This figure was generated using the `factoextra` library which provides powerful visualization for exploratory multivariate data analyses. From this figure one can see that `wsamoa` and `cookis` collectively contribute more than 35% to the first two principle components. The red dashed line corresponds to the expected value if the contribution were uniform between all countries. Countries with a contribution above this reference line are considered important in contributing to the first two dimensions.

The proportion of the total sample variance explained by the first two principal components can be identified by the scree plot in Figure 6a. Implementation can be found in Appendix 2.6. From this graph this value is $82.9\% + 9.3\% = 92.2\%$. Alternatively, a more accurate value can be found by calculating the variance of the principle components using the standard deviation of the output from the PCA computation. This is done by simply taking the square of the standard deviation of the PCA output. Then, the total variance over all principle components can be found and the fraction expressed by the first two can be calculated. Through this process the exact fraction of the first two principle components contribution to the overall variance is 92.27616%. This value corresponds with the rough percentage calculated by the Scree plot summation. This computation can be found in Appendix XXX.

## 4.6 Ranking Nations Based on Principle Components

As was discussed in Section 4.4, the first principle component can be seen as a proxy for overall countries performance. This component captures 82.9% of the total variation across all races and so it can be used as a mechanism to rank nations on their times across all races. Countries with a value above 0 are slower than the average and countries with a value below 0 are faster than the average. This visualization can be seen in Figure 7 below with implementation in Appendix 2.7.

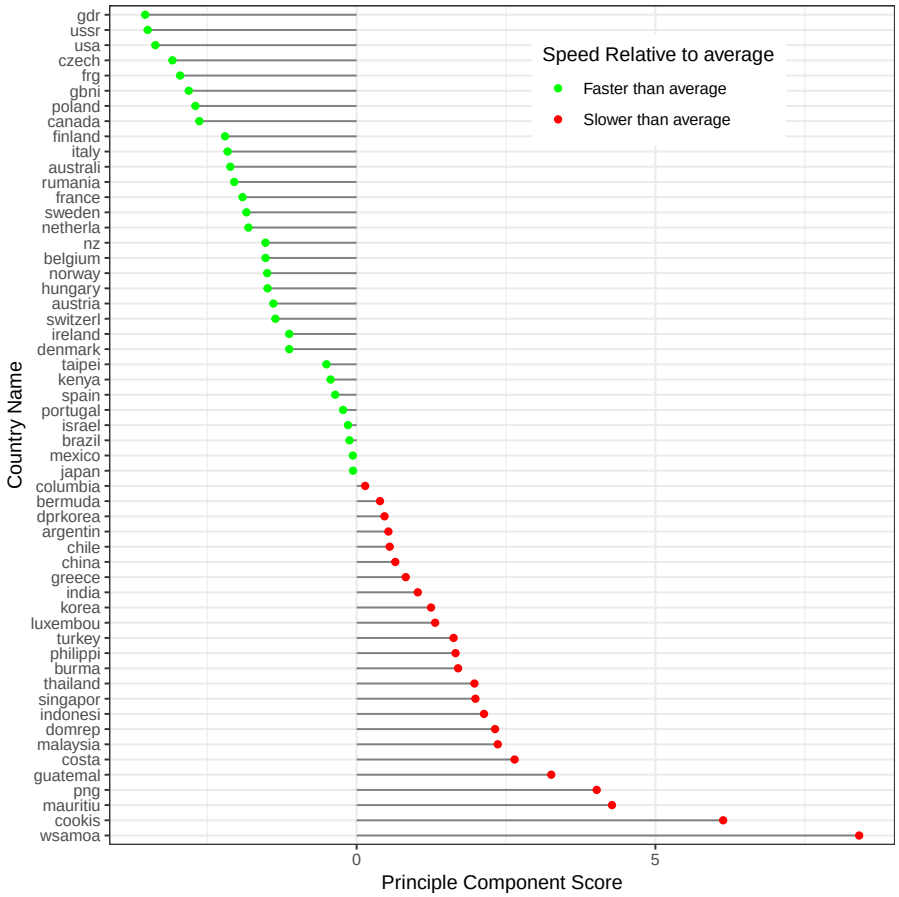



Figure 7: Variable Contribution to First Two Components

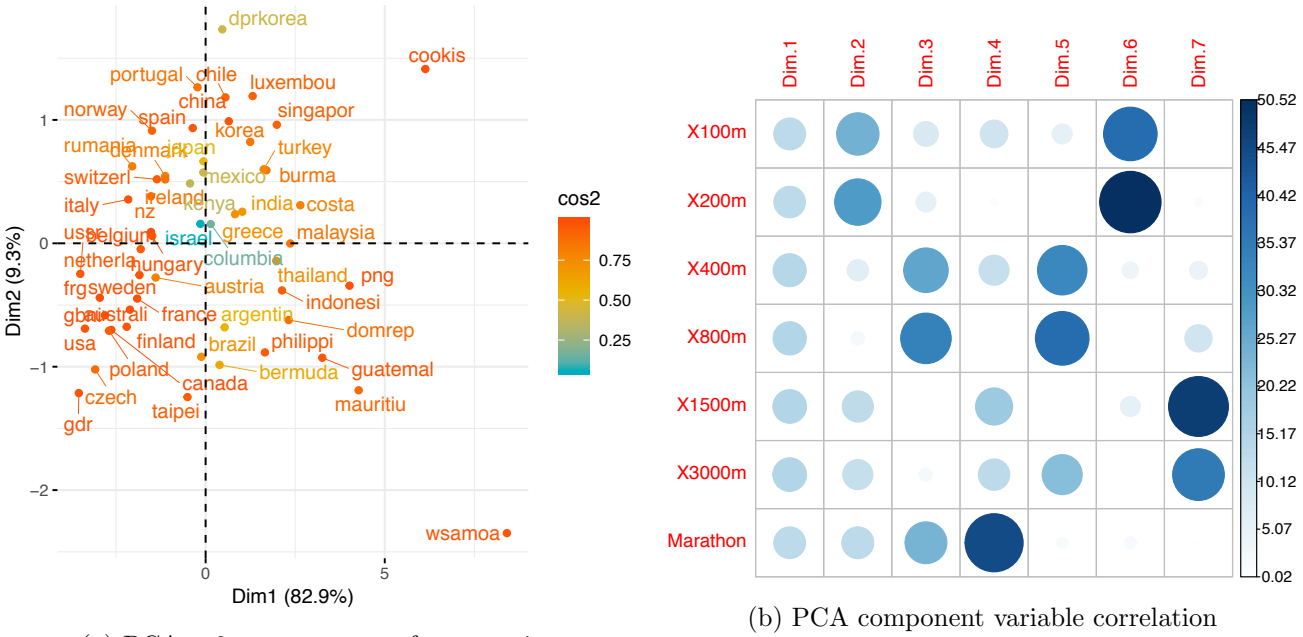
The ordering in this figure corresponds to intuition around which countries should be faster based on their athletic excellence with the top three countries being `gdr`, `ussr` and `usa`. Likewise, countries on the bottom end of the scale are `wsamoa` and `cookis` which corresponds to the analysis preformed thus far.

## 4.7 Quality of Principle Component Representation And Variable Contribution

The quality of a PCA match can be quantified through the use of the cos2 calculation. This is calculated as the squared loadings for variables. If a variable is perfectly represented through two components then the sum of the cos2 is equal to one. In this way the cos2 is used to estimate the quality of the lower dimensional representation. Figure 8a below shows the different countries along with the first two principle components and their associated cos2 values. Most countries have a high value of cos2, indicating that most of their data points have been expressed in the 2D plot. The exception to these are primarily values that sit along the y axis with a value close to 0 for Dim1. This indicates that there is information in higher dimensions (other than the first two principle components) that influences the relative positioning of these countries. Implementation can be found in Appendix 2.8.

Figure 8b shows the correlation between each variable and each principle component dimension. This shows how some of the higher dimensions are dominated by a few variables. For example dimension 4 is primarily defined by marathon times indicated by the large blue. The first dimension on the other hand has a very equal spread amounts all variables. This is expected as this dimension acts as a proxy over

all principle components and is not heavily influenced by one set variable. The second dimension shows influence from all races except for X400m and X800m. This is to be expected when contrasted to Figure 5b which shows these two races (X400m and X800m) having minimal correlation to the second principle component. Implementation can be found in Appendix 2.9.




(a) PCA co2 measurements for countries

(b) PCA component variable correlation

Figure 8: PCA quality measures and variable correlation to identify dimension contribution

# 5 Conclusion

This report examined the use of two unsupervised learning techniques in data exploration through dimensionality reduction. Multidimensional scaling was performed on UN statistics data set to revel different ways of quantifying countries development. Through this, different countries could be ordered into a level of development ranking. Secondly Principle component analysis was performed on female athletics data to reveal hidden patterns and structures within the data. A one dimensional ranking was also generates for these countries.

# Unsupervised Learning Assignment 1

This notebook contains all the code for the unsupervised learning assignment 1 implementation of Multidimensional Scaling on the UN_Statistics dataset.

Multidimensional scaling (MDS) is a unsupervised, multivariate data analysis technique that is used to visualize the similarity/dissimilarity between samples by plotting points in two dimensional plots.

MDS represents data in a lower-dimensional space, mapped from a higher dimentional space. In this notebook one and two dimentional representations are generated for a higher dimentional data set.

From an alorithmic perspective, MDS uses a dissimilarity matrix to represent the distances between pairs of objects. The input data to the MDS algorithm is this disimilarity matrix.

## Enviroment Setup and Import data files

This notebook is set up to make the results atained as reproducable as posible.

```
#setup work space, install packages and import libs
rm(list=ls())
suppressMessages(library(cluster))
suppressMessages(library(MASS))
suppressMessages(library(smacof))
suppressMessages(library(magrittr))
suppressMessages(library(dplyr))
suppressMessages(library(ggpubr))
suppressMessages(library(psych))

#import and sample data, apply seed and sample to get the set of 400 unique data points
statistics <- read.csv("UN_Statistics.csv")
X <- as.matrix(statistics[,-1])
rownames(X) <- statistics[,1]
```

#data exploration

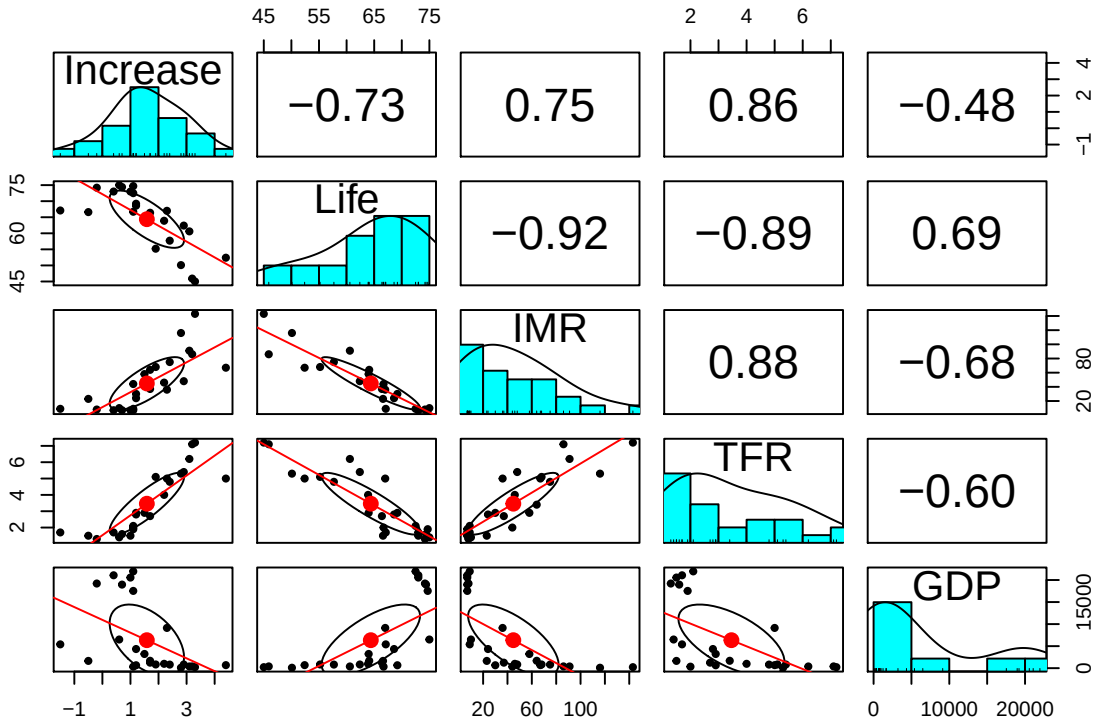
```
rownames(X)
```

```
##  [1] "Albania"          "Argentina"        "Australia"
##  [4] "Austria"          "Benin"            "Boliva"
##  [7] "Brazil"           "Cambodia"         "China"
## [10] "Colombia"         "Croatia"          "El Salvador"
## [13] "France"           "Greece"           "Guatemala"
## [16] "Iran"             "Italy"            "Malawi"
## [19] "Netherlands"      "Pakistan"         "Papua new Guinea"
## [22] "Peru"             "Romania"          "USA"
## [25] "Zimbabwe"
```

```
sapply(statistics,class)
```

```
##    Country   Increase       Life        IMR        TFR        GDP
##   "factor" "numeric" "numeric" "integer" "numeric" "numeric"
```

```
pairs.panels(statistics[,-1],cex=1,lm=TRUE)
```

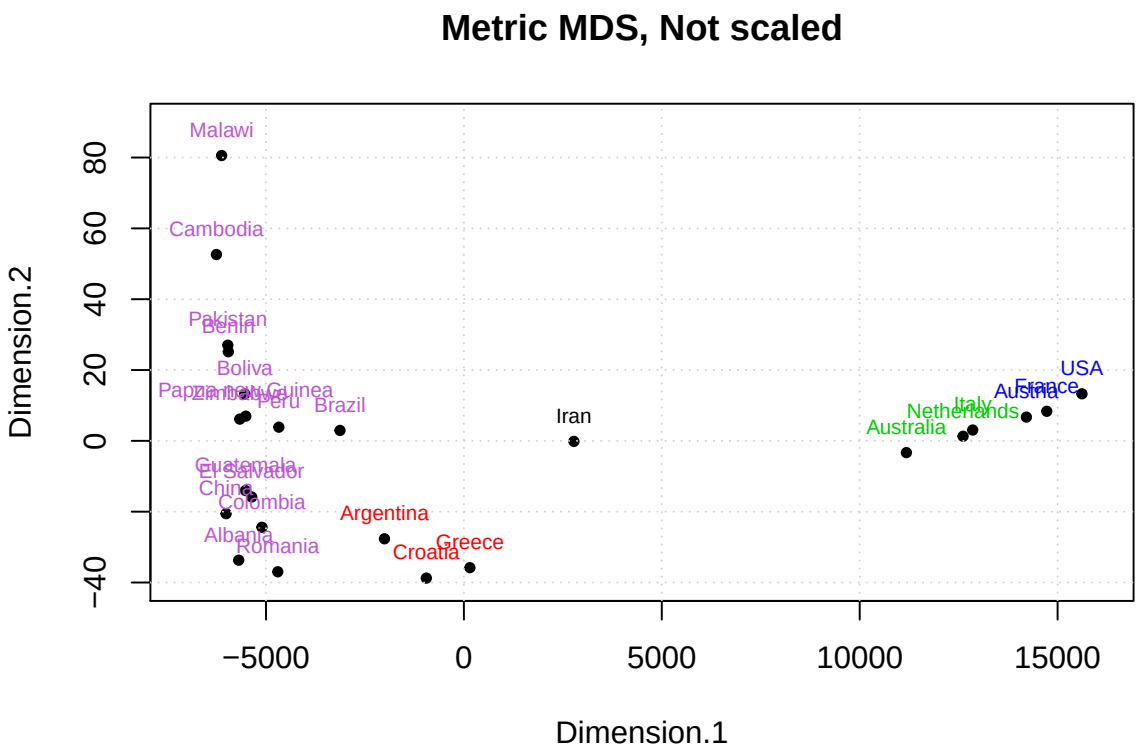



## Classical MDS Scaling

First let's plot the data before we normalize it.

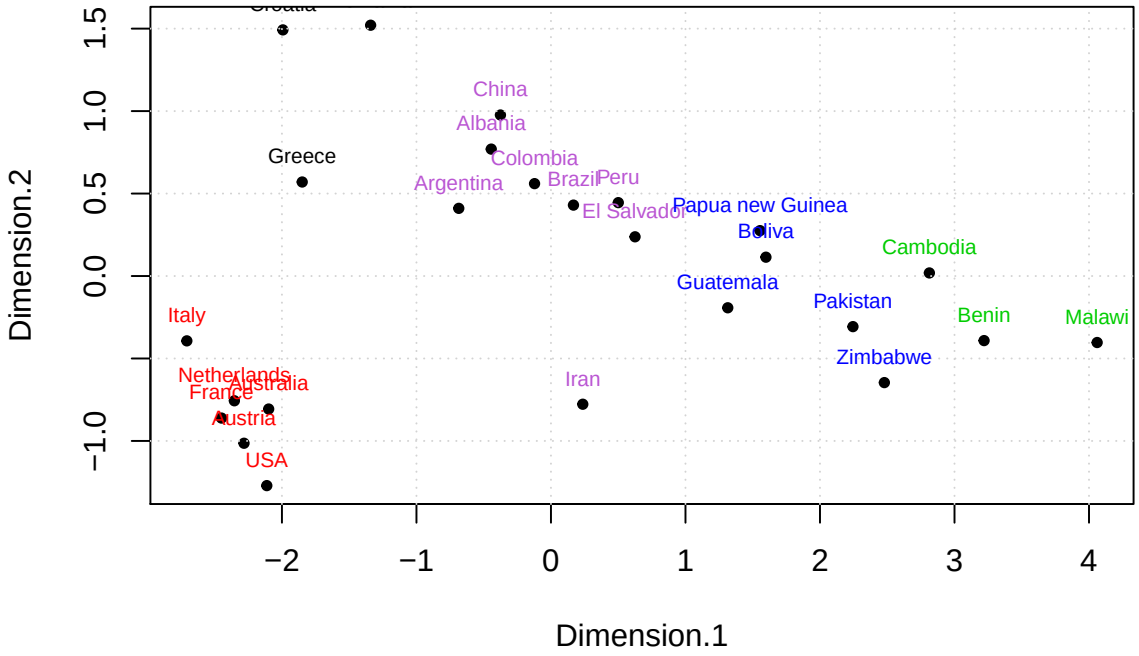
```
d <- dist(X) # euclidean distances between the rows

#preform clustering so we can allocate colours to the plot
cluster <- hclust(d,method="complete")
clusvec <- cutree(cluster, k=5)

scaledDistances <- cmdscale(d) # preform the multidimensional scalling

# create empty plot and then add text and colours. Colours added based on the clustor groups
plot (scaledDistances, xlab="Dimension.1", ylab="Dimension.2",
  main="Metric MDS, Not scaled", type="p", pch=20,ylim = c(-40,90), xlim = c(-7000,16000))
grid()
#ensure you list enough colours for the number of clusters
colvec <- c("mediumorchid",
            "red",
            "green3",
            "blue",
            "black",
            "gold",
            "indianred",
            "moccasin",
            "lightcyan",
            "skyblue")
```

```
for (i in 1:length(scaledDistances[,1]))
  text (scaledDistances[i,1],
        scaledDistances[i,2],
        rownames(X)[i],
        col=colvec[clusvec[i]],
        cex=0.7,
        pos = 3)
```




#Normalized MDS Next we can repeat the process but this time normalize the data scaling

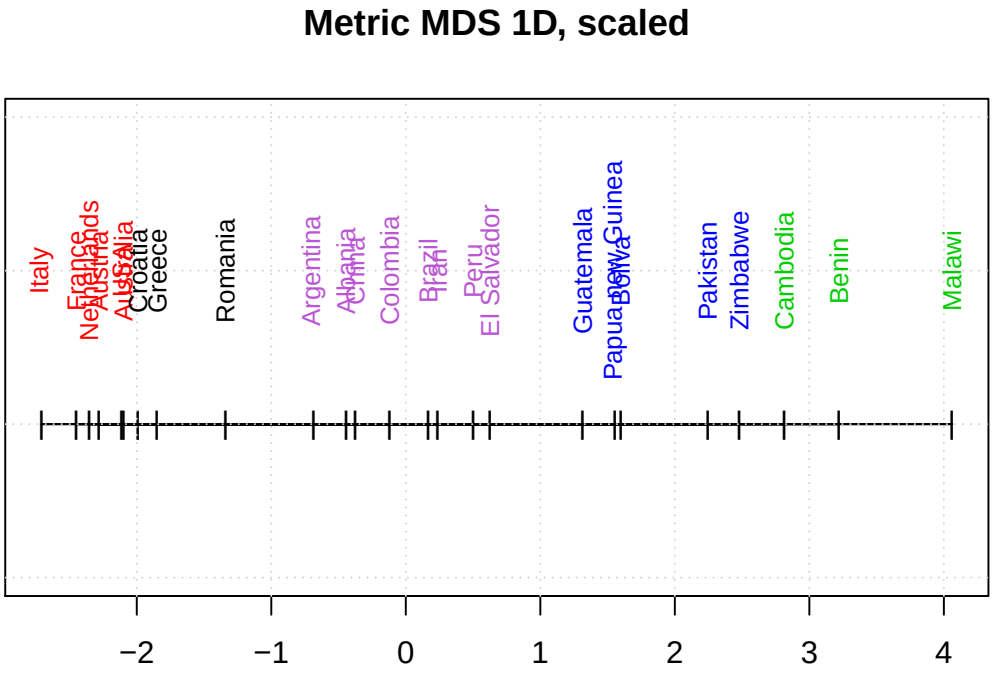
```
# euclidean distances between the rows
d <- dist(scale(X, center=TRUE, scale=TRUE), method="euclidean")

#preform clustering so we can allocate colours to the plot
cluster <- hclust(d,method="complete")
clusvec <- cutree(cluster, k=5)

scaledDistances <- cmdscale(d, k = 2) # preform the multidimensional scalling

# create empty plot and then add text and colours. Colours added based on the clustor groups
plot (scaledDistances, xlab="Dimension.1", ylab="Dimension.2",
  main="Metric MDS, Scaled", type="p", pch=20)
grid()
for (i in 1:length(scaledDistances[,1]))
  text (scaledDistances[i,1],
        scaledDistances[i,2],
        rownames(X)[i],
        col=colvec[clusvec[i]],
        cex=0.7,
        pos = 3)
```




#One

Dimentional Plot Generate a one dimentional representation of the data

```
d <- dist(scale(X)) # euclidean distances between the rows

#preform clustering so we can allocate colours to the plot
cluster <- hclust(d,method="complete")
clusvec <- cutree(cluster, k=5)

scaledDistances <- cmdscale(d, eig=TRUE, k=1) # preform the multidimensional scalling

x <- data.frame(scaledDistances$points[,1],1)

# create empty plot and then add text and colours. Colours added based on the clustor groups
plot (x, xlab="Dimension.1",
  main="Metric MDS 1D, scaled",
  type = 'o',
  pch = '|',
  ylab = '',
  yaxt='n',
  xlim=c(min(x),
         max(x)),
  ylim=c(0.95,1.1))
grid()

for (i in 1:length(x[,1]))
  text (x[i,1],
        1.05,
        rownames(X)[i],
        col=colvec[clusvec[i]],
        cex=0.8,
        srt=90)
```

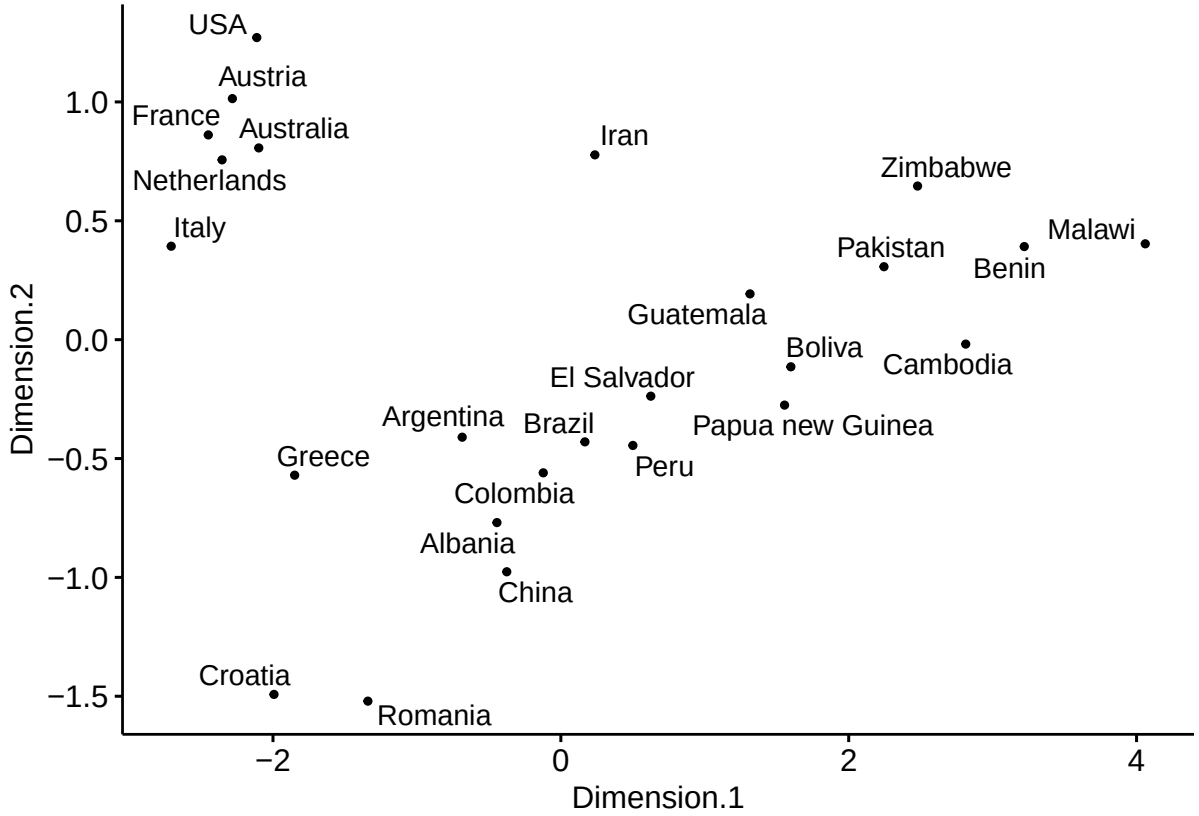



# Classical MDS

```
# Cmpute MDS
mds <- scale(X) %>%
  dist() %>%
  cmdscale() %>%
  as_tibble()
```

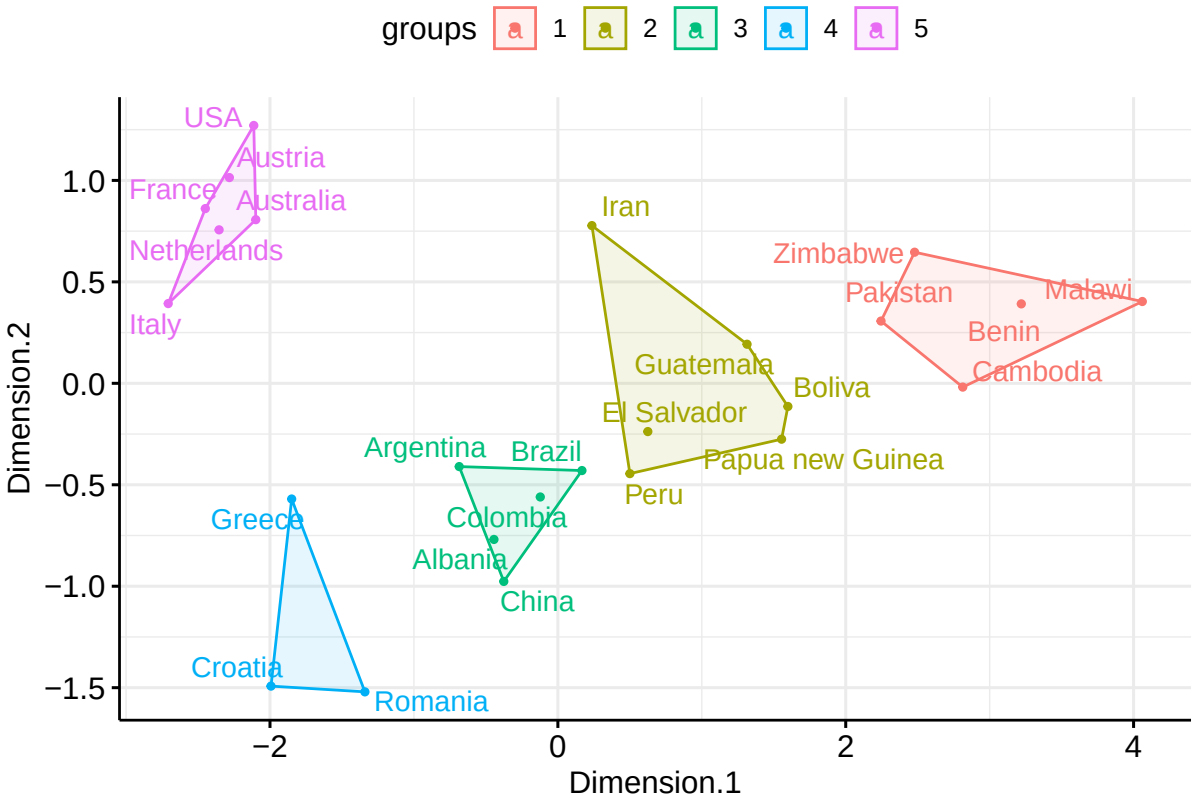
```
## Warning: `as_tibble.matrix()` requires a matrix with column names or a `.name_repair` argument. Using
## This warning is displayed once per session.
```

```
colnames(mds) <- c("Dimension.1", "Dimension.2")
# Plot MDS
mds[,2] <- -1 * mds[,2]
ggscatter(mds, x = "Dimension.1", y = "Dimension.2",
          label = rownames(X),
          size = 1,
          repel = TRUE)
```




We can add colours to the plots while clustering them together

```
# K-means clustering
clust <- kmeans(mds, 5)$cluster %>%
  as.factor()
mds <- mds %>%
  mutate(groups = clust)
# Plot and color by groups
ggscatter(mds, x = "Dimension.1", y = "Dimension.2",
          label = rownames(X),
          color = "groups",
          palette = "pal3",
          size = 1,
          ellipse = TRUE,
          ellipse.type = "convex",
          repel = TRUE) +
grids(axis = c("xy", "x", "y"), color = "grey92", size = NULL,
  linetype = NULL)
```



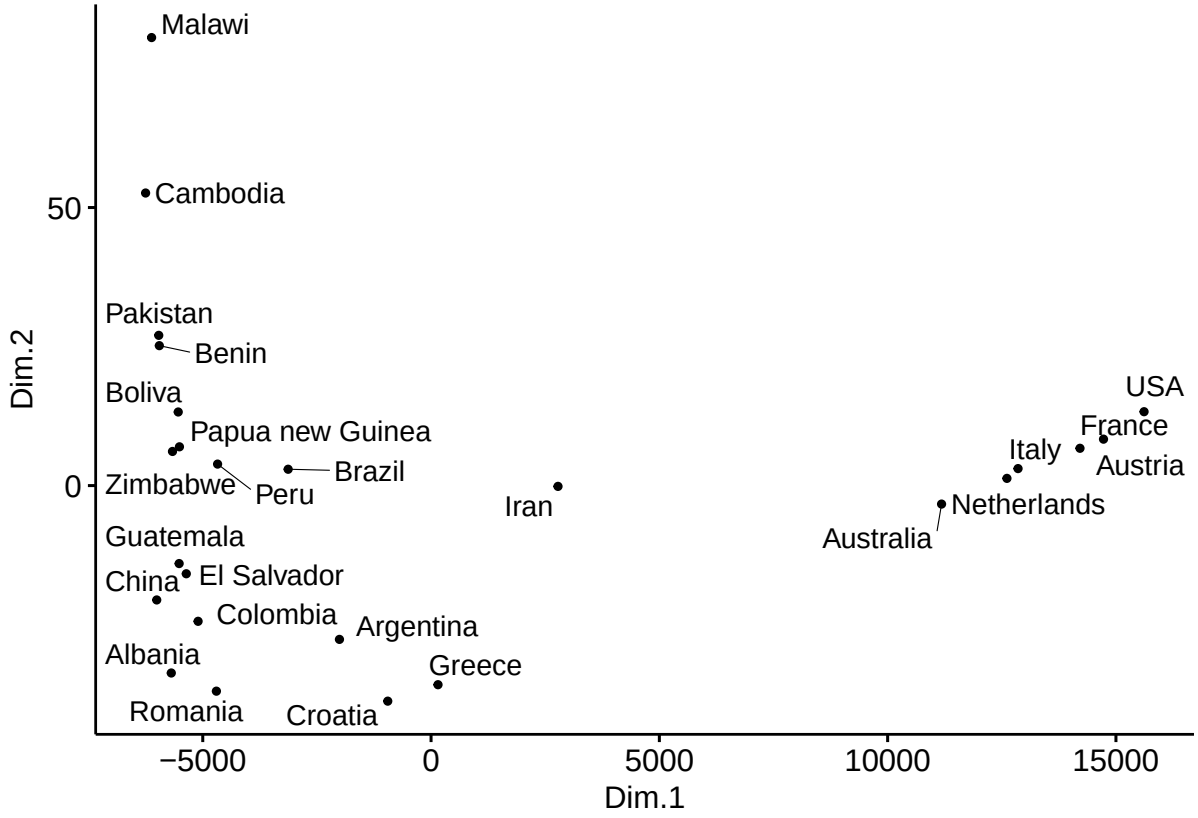

## Non-metric MDS

## Kruskal's non-metric multidimensional scaling

```
mds <- X %>%
  dist() %>%
  isoMDS() %>%
  .$points %>%
  as_tibble()
```

```
## initial  value 0.000168
## final  value 0.000168
## converged
```

```
colnames(mds) <- c("Dim.1", "Dim.2")
# Plot MDS
ggscatter(mds, x = "Dim.1", y = "Dim.2",
          label = rownames(X),
          size = 1,
          repel = TRUE)
```

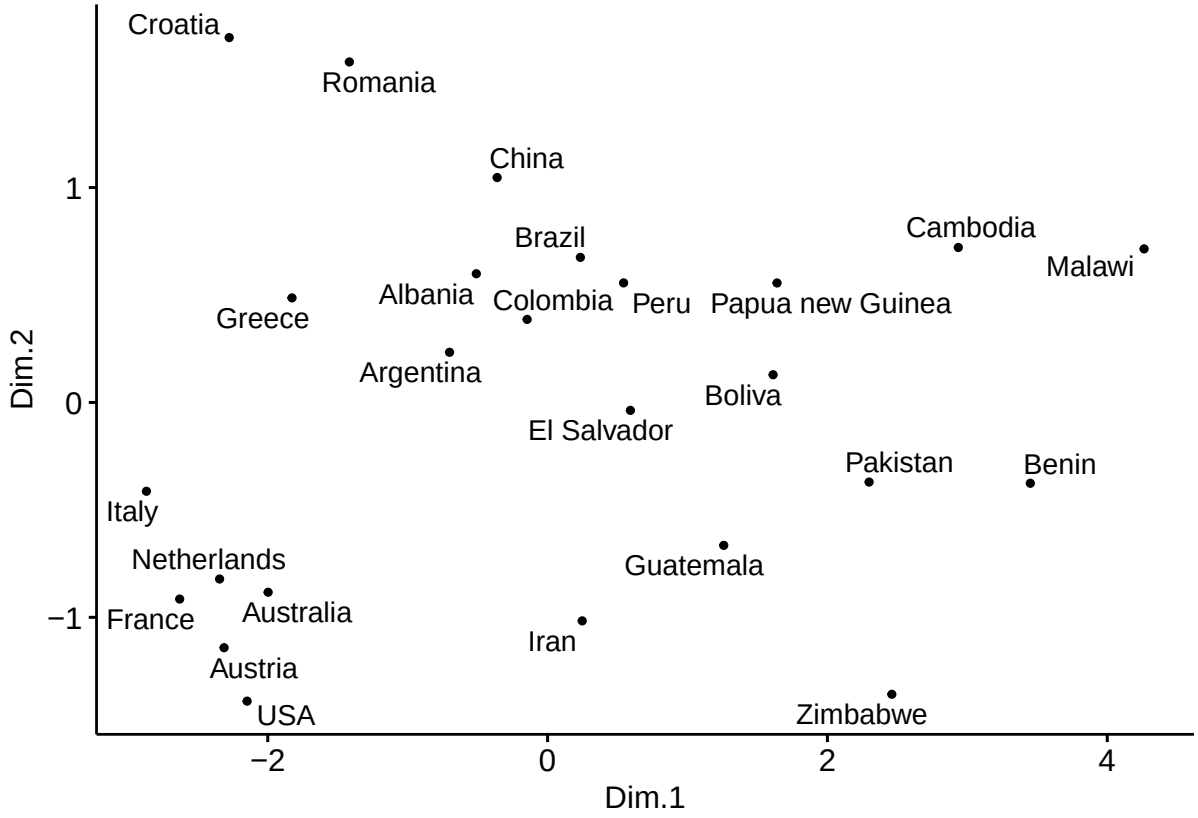



## Sammon's non-linear mapping:

```
mds <- scale(X) %>%
  dist() %>%
  sammon() %>%
  .$points %>%
  as_tibble()
```

```
## Initial stress        : 0.01485
## stress after  10 iters: 0.00425, magic = 0.500
## stress after  20 iters: 0.00421, magic = 0.500
```

```
colnames(mds) <- c("Dim.1", "Dim.2")
# Plot MDS
ggscatter(mds, x = "Dim.1", y = "Dim.2",
          label = rownames(X),
          size = 1,
          repel = TRUE)
```

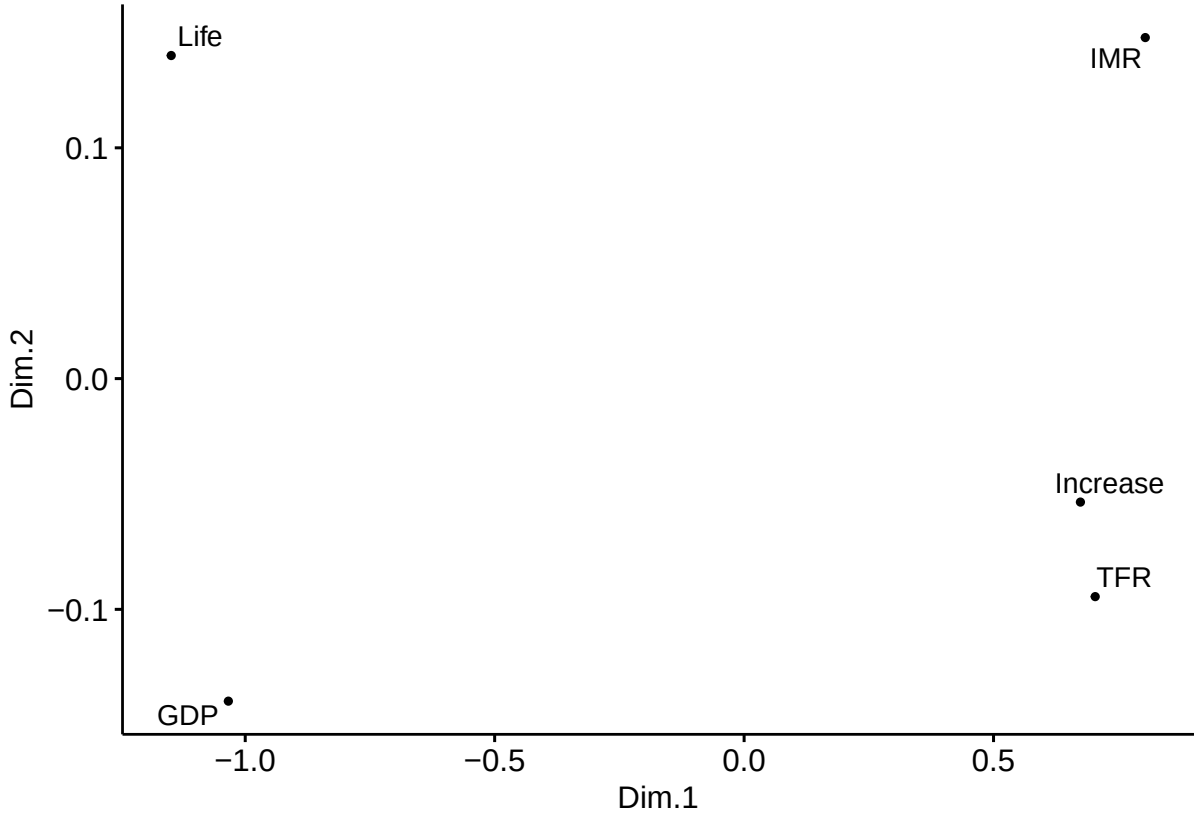



## Visualizing a correlation matrix using Multidimensional Scaling

MDS can be also used to reveal a hidden pattern in a correlation matrix.

Correlation actually measures similarity, but it is easy to transform it to a measure of dissimilarity.

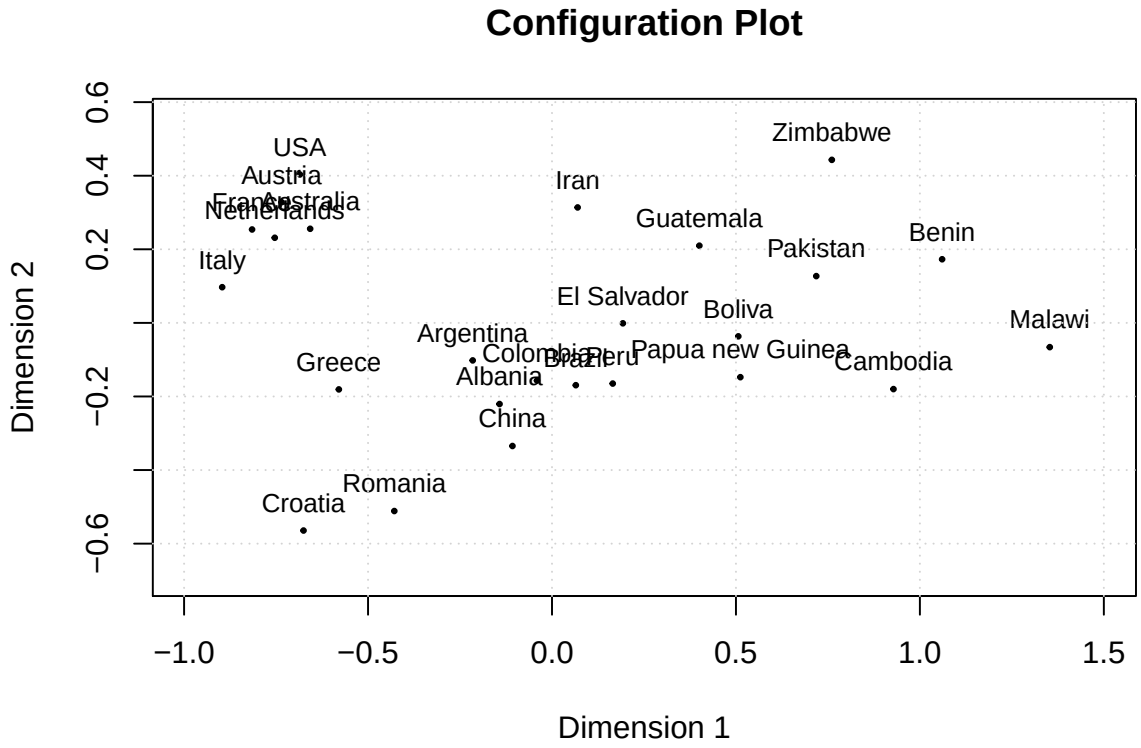
```
res.cor <- cor(X, method = "spearman")
mds.cor <- (1 - res.cor) %>%
  cmdscale() %>%
  as_tibble()
colnames(mds.cor) <- c("Dim.1", "Dim.2")
ggscatter(mds.cor, x = "Dim.1", y = "Dim.2",
          size = 1,
          label = colnames(res.cor),
          repel = TRUE)
```



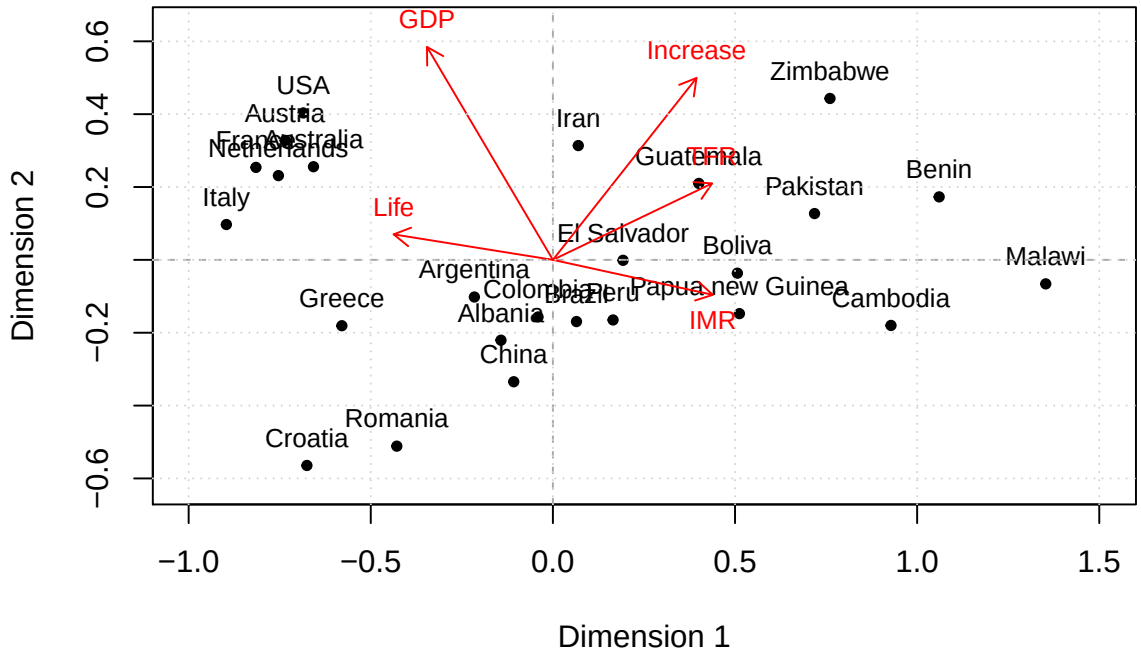

## MDS Biplot

Visualize underlying variables along with MDS plot

```
d <- dist(scale(X, center=TRUE, scale=TRUE), method="euclidean")

countriesMds <- mds(delta = d, ndim = 2)
plot(countriesMds)
grid()
```




```
biplot <- biplotmds(countriesMds, extvar = X)
plot(biplot,  vecscale = 0.3, vec.conf = list(col = "red", length = 0.1),main="")
grid()
```



# Unsupervised Learning Assignment 1

This notebook contains all the code for the unsupervised learning assignment 1 implementation of Principal Component Analysis(PCA) on the TrackRecords data set.

PCA enables the summarization and visualization of information in a dataset described by multiple inter-correlated quantitive variables. Through this process PCA acts to reduce the dimentionality of a dataset.

PCA's goal is to extract useful information from a multvariate data set and to express this useful information as a set of principle components. These principle components correspeond to a linear combination of the original variables.

Information in a dataset can be quantified by it's total variation. The goal of PCA is to identify the directions along which the variation in the data is maximum and therby extract the principle components.

As a result, PCA can be used to reduce the dimensionality of a multivarable data set. This enables easy visualization in two or three dimensions of a much higher dimentional dataset.

## Enviroment Setup and data import

Begin by adding the libraries that are needed and reading in the data. Drop the first two rows as these are row names and indexes. The row names are added back later.

```
#clean workspace
rm(list=ls())

#install librarys
suppressMessages(library(factoextra))
suppressMessages(library("corrplot"))
suppressMessages(require(ggplot2))
suppressMessages(library(psych))

#import data
trackRecords <- read.csv("TrackRecords.csv")
X <- as.matrix(trackRecords[,-1:-2]) #remove the row number and row name
rownames(X) <- trackRecords[,2] #set the row name as the matrix row name
```

#Basic data exploration

```
rownames(X)
```

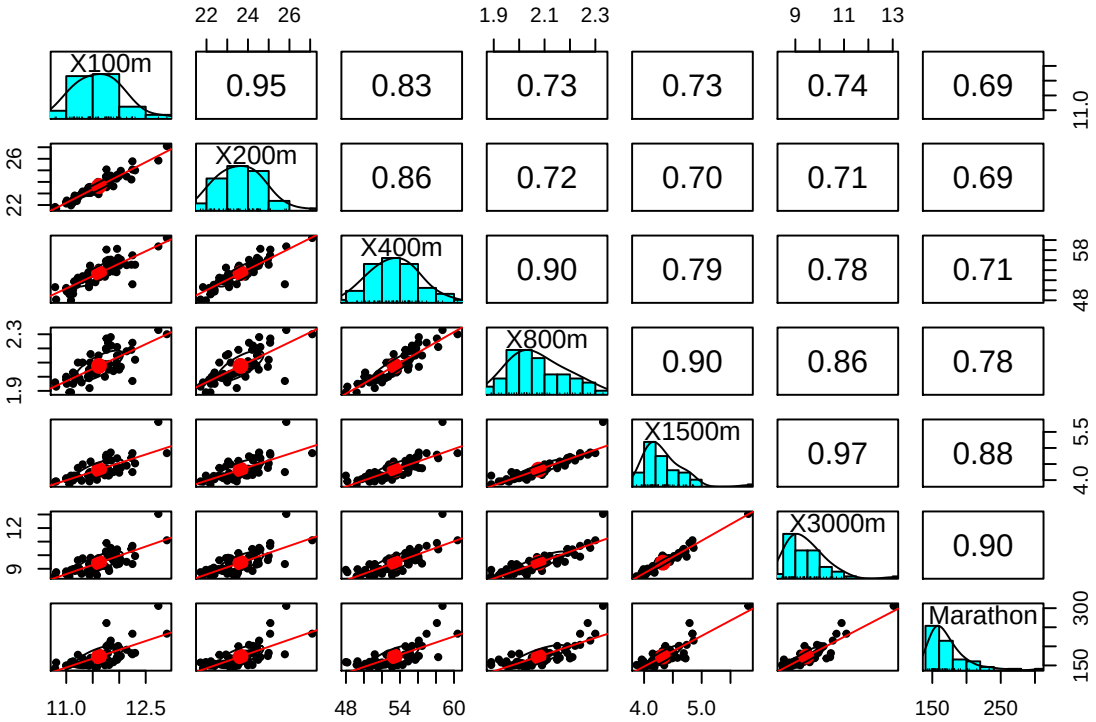
```
##  [1] "argentin" "australi" "austria"  "belgium"  "bermuda"  "brazil"
##  [7] "burma"    "canada"   "chile"    "china"    "columbia" "cookis"
## [13] "costa"    "czech"    "denmark"  "domrep"   "finland"  "france"
## [19] "gdr"      "frg"      "gbni"     "greece"   "guatemal" "hungary"
## [25] "india"    "indonesi" "ireland"  "israel"   "italy"    "japan"
## [31] "kenya"    "korea"    "dprkorea" "luxembou" "malaysia" "mauritiu"
## [37] "mexico"   "netherla" "nz"       "norway"   "png"      "philippi"
## [43] "poland"   "portugal" "rumania"  "singapor" "spain"    "sweden"
## [49] "switzerl" "taipei"   "thailand" "turkey"   "usa"      "ussr"
## [55] "wsamoa"
```

```
sapply(trackRecords,class)
```

```
##       OBS   COUNTRY     X100m     X200m     X400m     X800m    X1500m
```

```
## "integer"  "factor" "numeric" "numeric" "numeric" "numeric" "numeric"
##    X3000m  Marathon
## "numeric" "numeric"
```

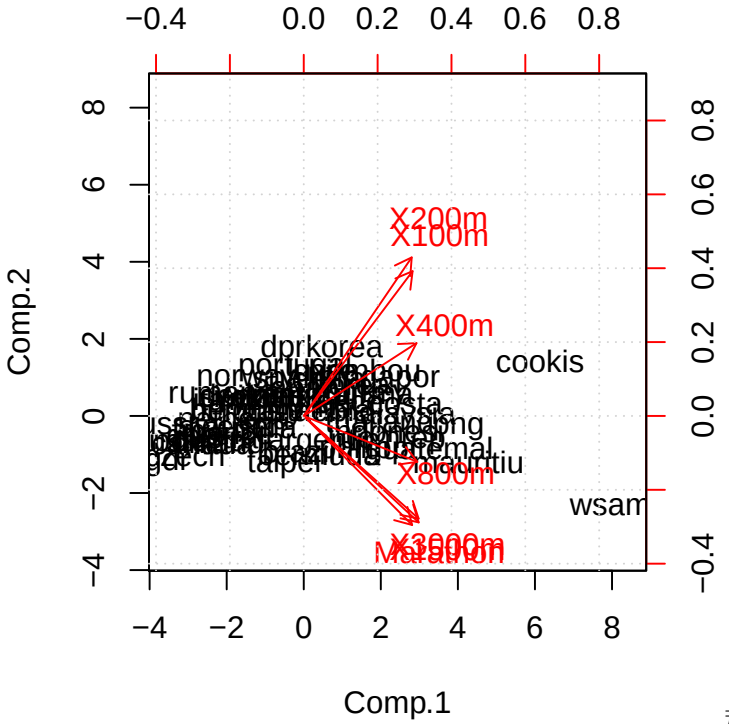
```
pairs.panels(trackRecords[,-1:-2],cex=1,lm=TRUE)
```




## Generation of PCA

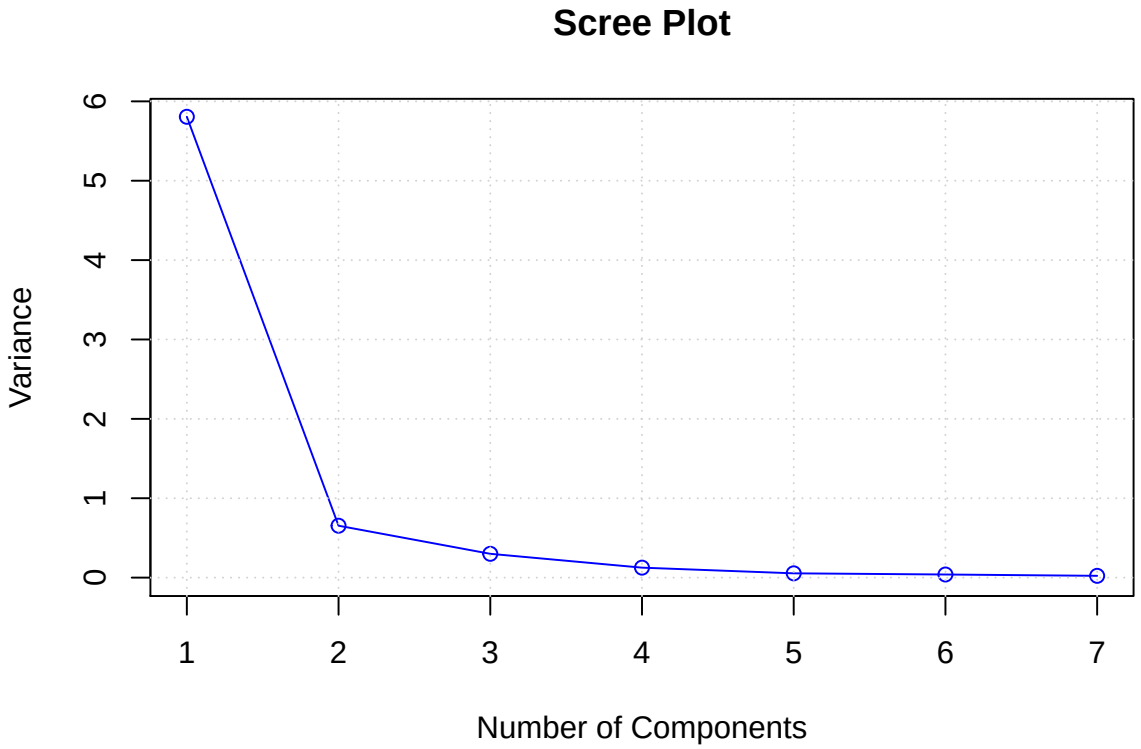
Next generate the basic PCA and create a simple biplot. This graph is refined later on.

```
#preform PCA
#Cor=TRUE: the data will be centered and scaled before the analysis
#scores=TRUE: the coordinates on each principal component are calculated
pca.out <- princomp(X, cor=TRUE, scores=TRUE)

#generate basic biplot
biplot(pca.out,scale=0)
grid()
```



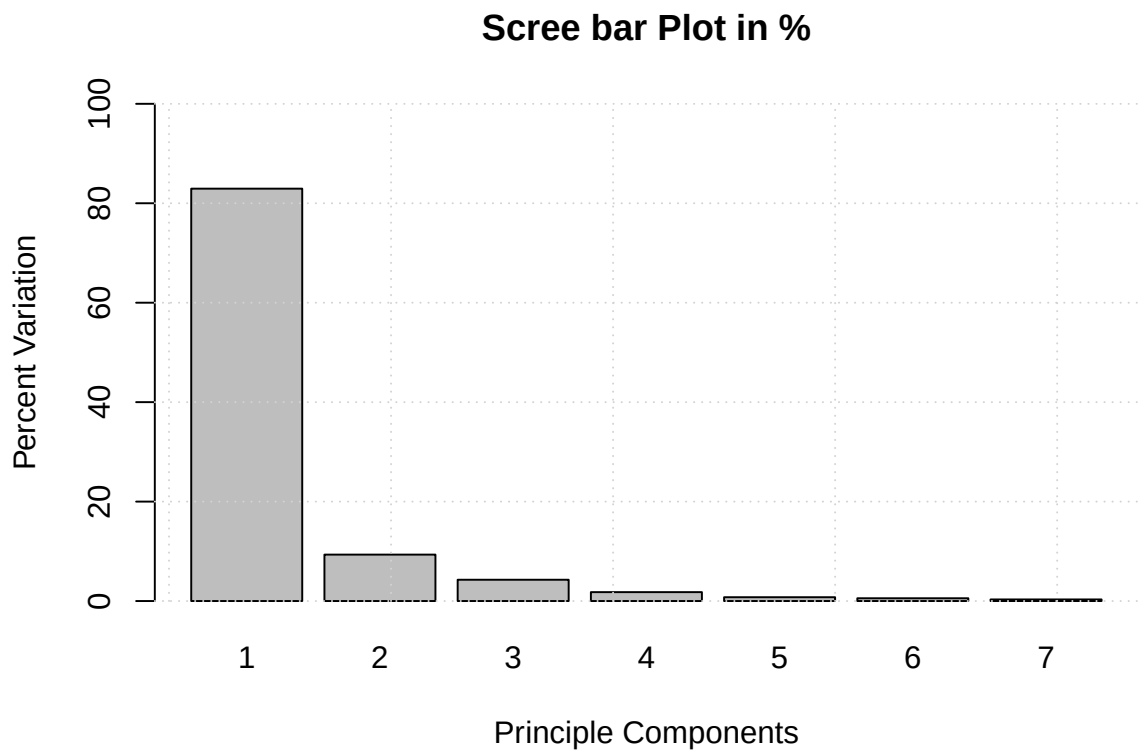

# Basic scree plot We can quantify the total variance expressed by the diffrent principle components using a scree plot. First, this is done using a line then a bar plot.

```
Variance<-(pca.out$sdev)^2 #calculate the standard deviation of this principle component
max_Var<-round(max(Variance),1)
Components<-c(1:length(Variance))
Components<-as.integer(Components)
plot(Components,
     Variance,
     main="Scree Plot",
     xlab="Number of Components",
     ylab="Variance",
     type="o",
     col="blue",
     ylim=c(0,max_Var))
grid()
```




# Bar plot-scree plot and variance explained Can express the total variance explain as the sum of all the variance expressed by each principle component. Can then plot this as a bar plot

```
totalVariance <- sum(Variance)
fractionOfVariance <- (Variance / totalVariance) * 100
barplot(fractionOfVariance,
        main = "Scree bar Plot in %",
        xlab = "Principle Components"
        , ylab="Percent Variation"
        , ylim = c(0,100)
        , names.arg = c(1:7))
grid()
```




```
first2PC <- sum(fractionOfVariance[1:2])
first2PC
```

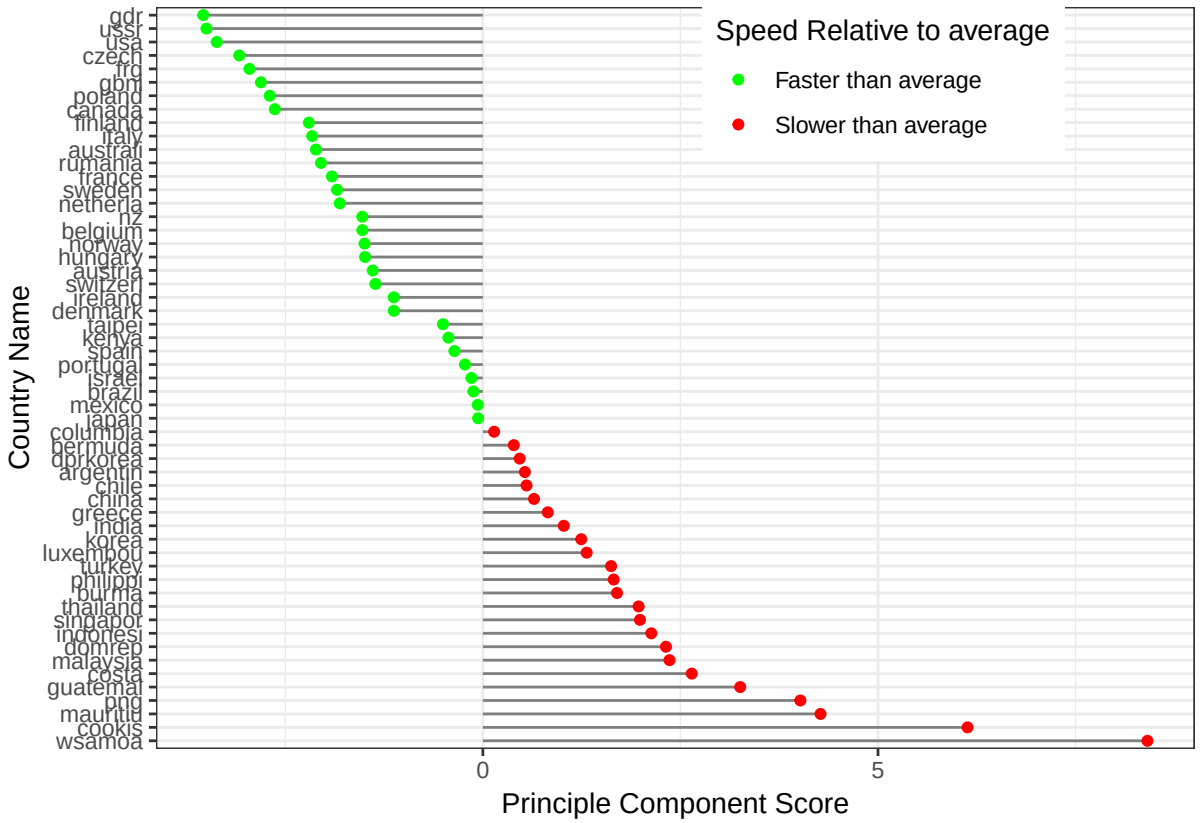
```
## [1] 92.27616
```

## Order countreis based on first component

Next, the nations are ranked based on the score of the first principle component. This plot shows

```
#order and process the data
pc1Ranked <- order(pca.out$scores[,1], decreasing = TRUE)
orderedCountries <- rownames(pca.out$scores)[pc1Ranked]
orderedValues <- pca.out$scores[pc1Ranked]
plotDataFrame <- cbind(orderedCountries,orderedValues)
plotDataFrame  <- as.data.frame(plotDataFrame)
plotDataFrame$orderedValues <- as.numeric(as.character(plotDataFrame$orderedValues))

#generate plot
plotDataFrame$orderedCountries <- factor(plotDataFrame$orderedCountries,
                                         levels = rownames(pca.out$scores)[pc1Ranked])
ggplot(plotDataFrame,
       aes(x=orderedValues,
           y=orderedCountries,
           color = orderedValues > 0)) +
  labs(x = "Principle Component Score",
       y = "Country Name",
       color = "Speed Relative to average") +
  scale_color_manual(labels = c("Faster than average",
                                "Slower than average"),
```

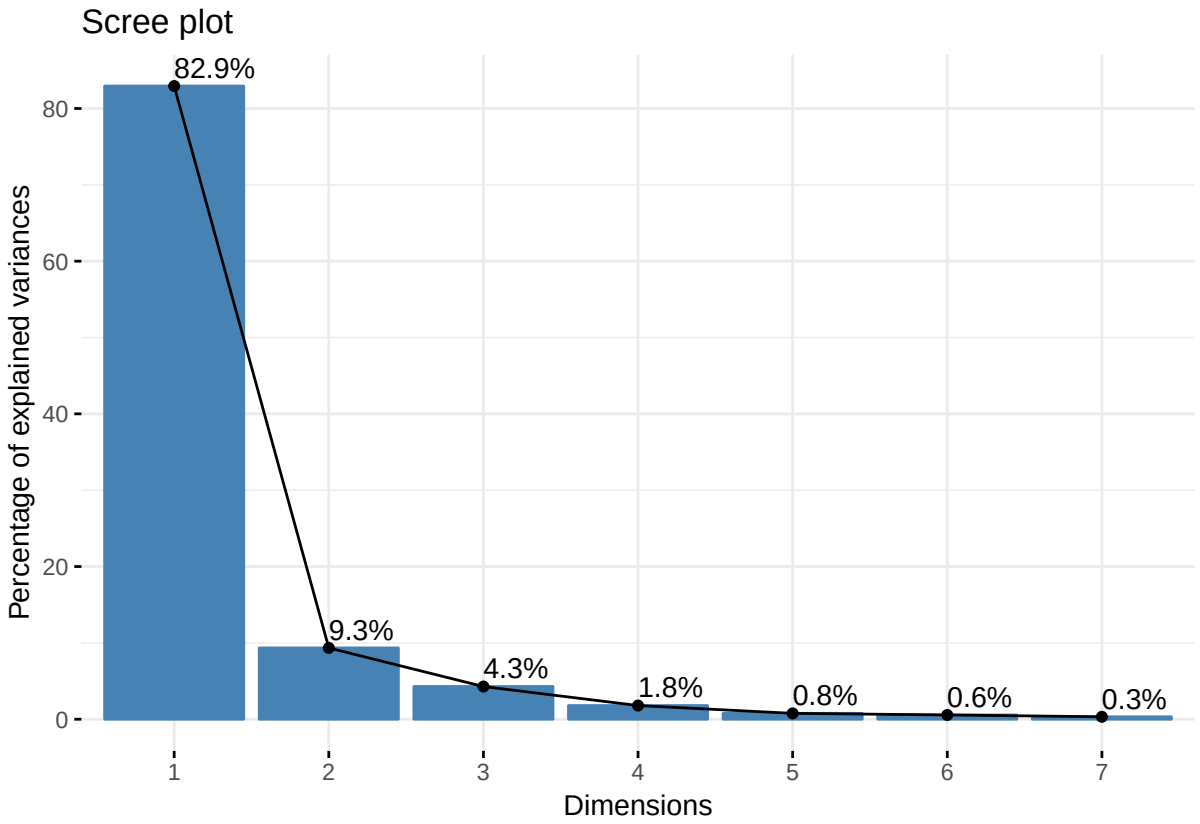
```
                    values = c("green", "red")) +
      geom_segment(aes(x = 0,
                       y = orderedCountries,
                       xend = orderedValues,
                       yend = orderedCountries),
                   color = "grey50") +
      geom_point() +
  theme_bw() +
  theme(legend.position=c(0.7,0.9))
```



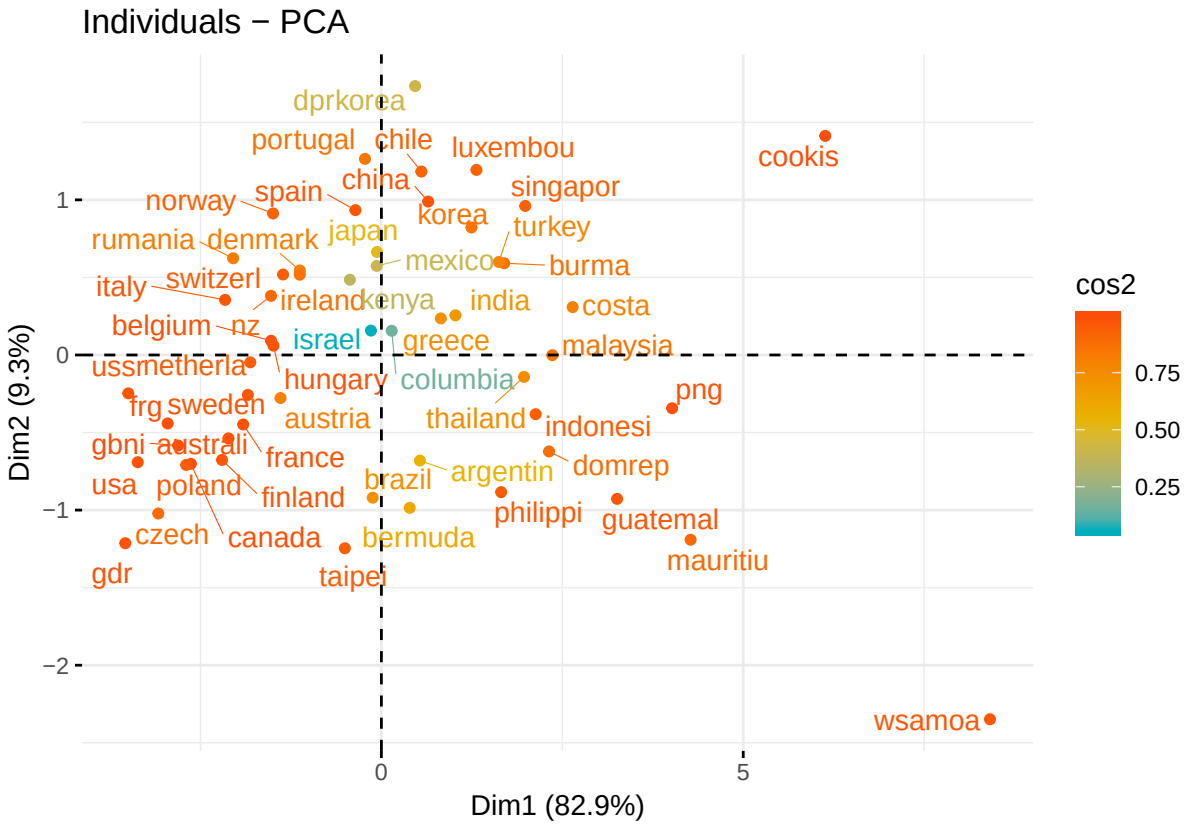

## Better visualization of Scree and Cos2 valuation

Next, there are a number of visualizations that we can perform on the PCA.

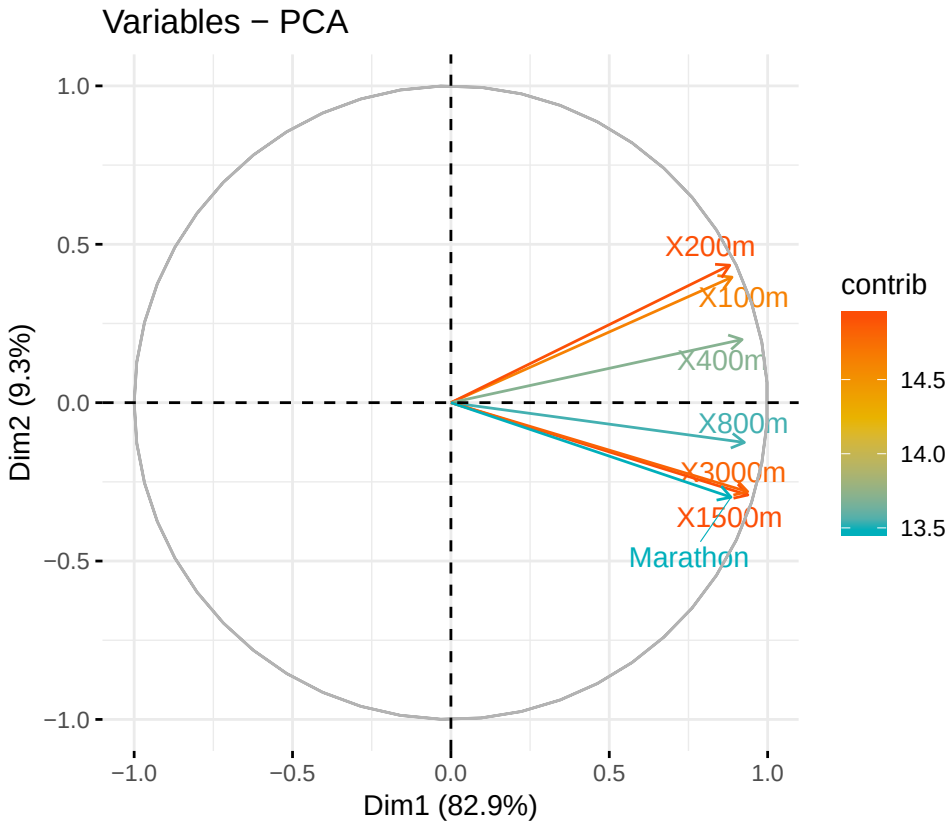
```
#Visualize eigenvalues (scree plot).
#Show the percentage of variances explained by each principal component.
fviz_eig(pca.out, addlabels = TRUE)
```



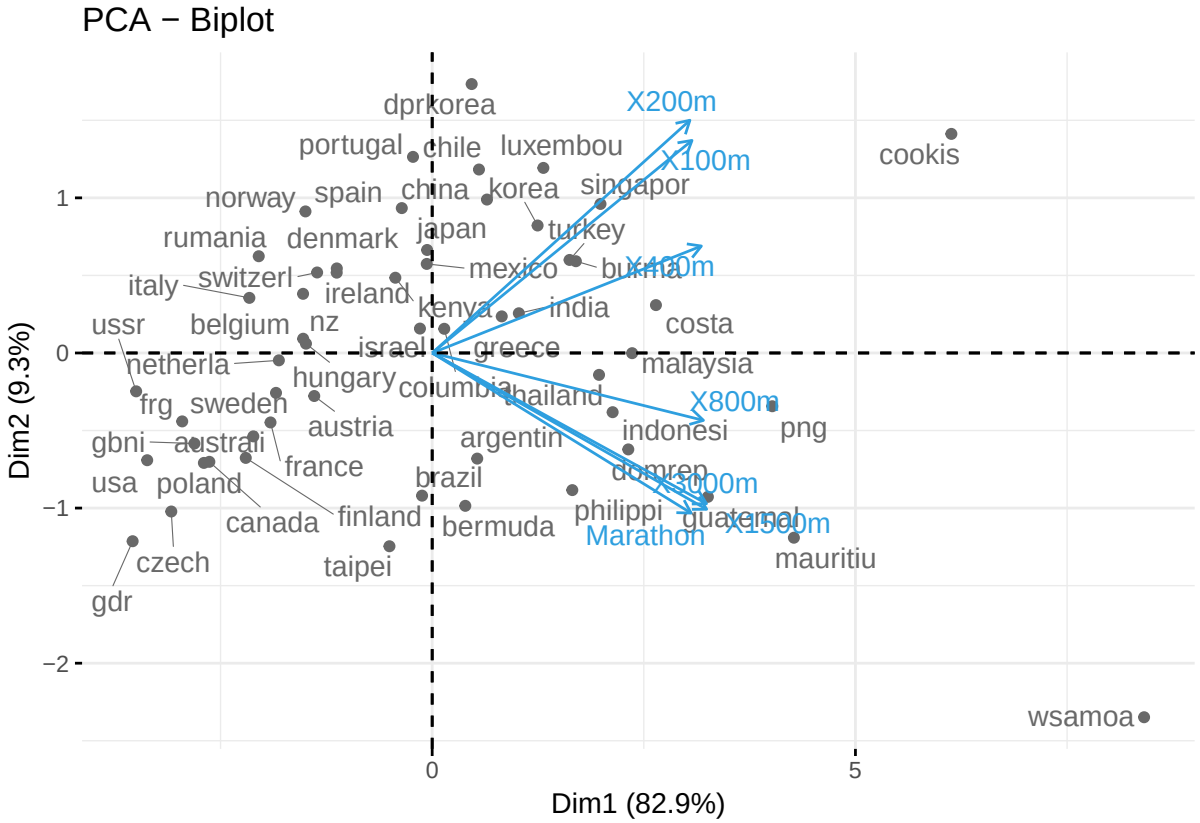

```
#Graph of individuals. Individuals with a similar profile are grouped together.
fviz_pca_ind(pca.out,
             col.ind = "cos2", # Color by the quality of representation
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE     # Avoid text overlapping
             )
```




```
#Graph of variables.
#Positive correlated variables point to the same side of the plot.
#Negative correlated variables point to opposite sides of the graph.
fviz_pca_var(pca.out,
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE     # Avoid text overlapping
             )
```



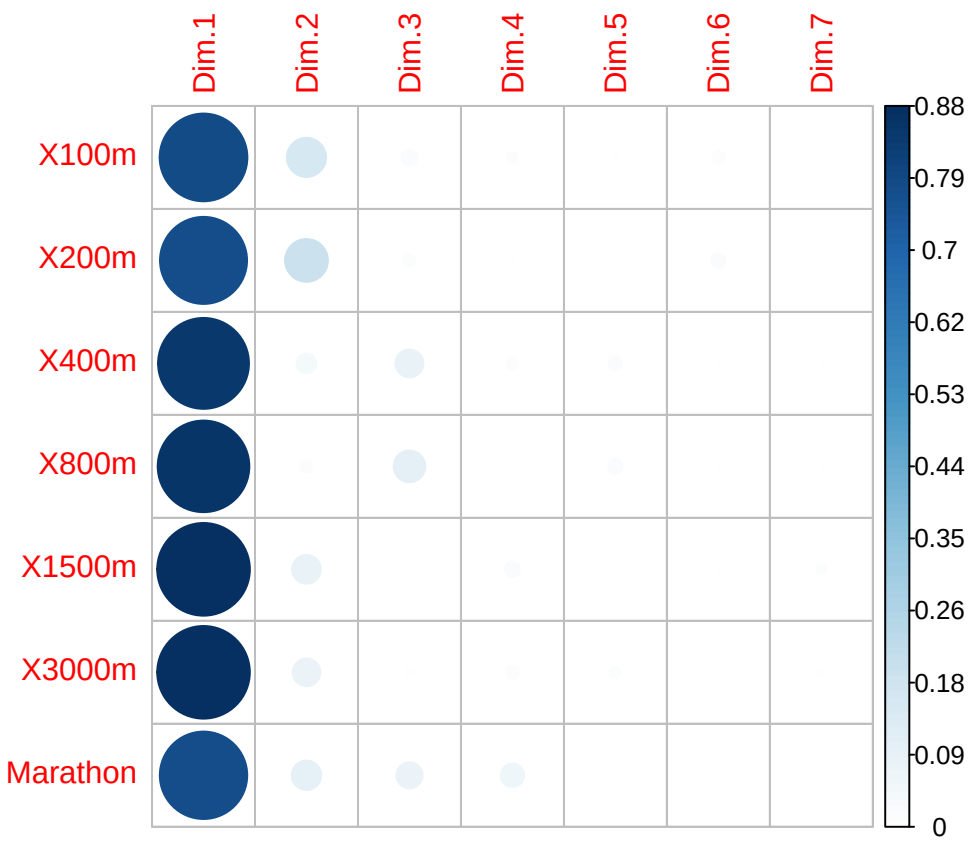

```
#Biplot of individuals and variables
fviz_pca_biplot(pca.out, repel = TRUE,
                col.var = "#2E9FDF", # Variables color
                col.ind = "#696969"  # Individuals color
                )
```



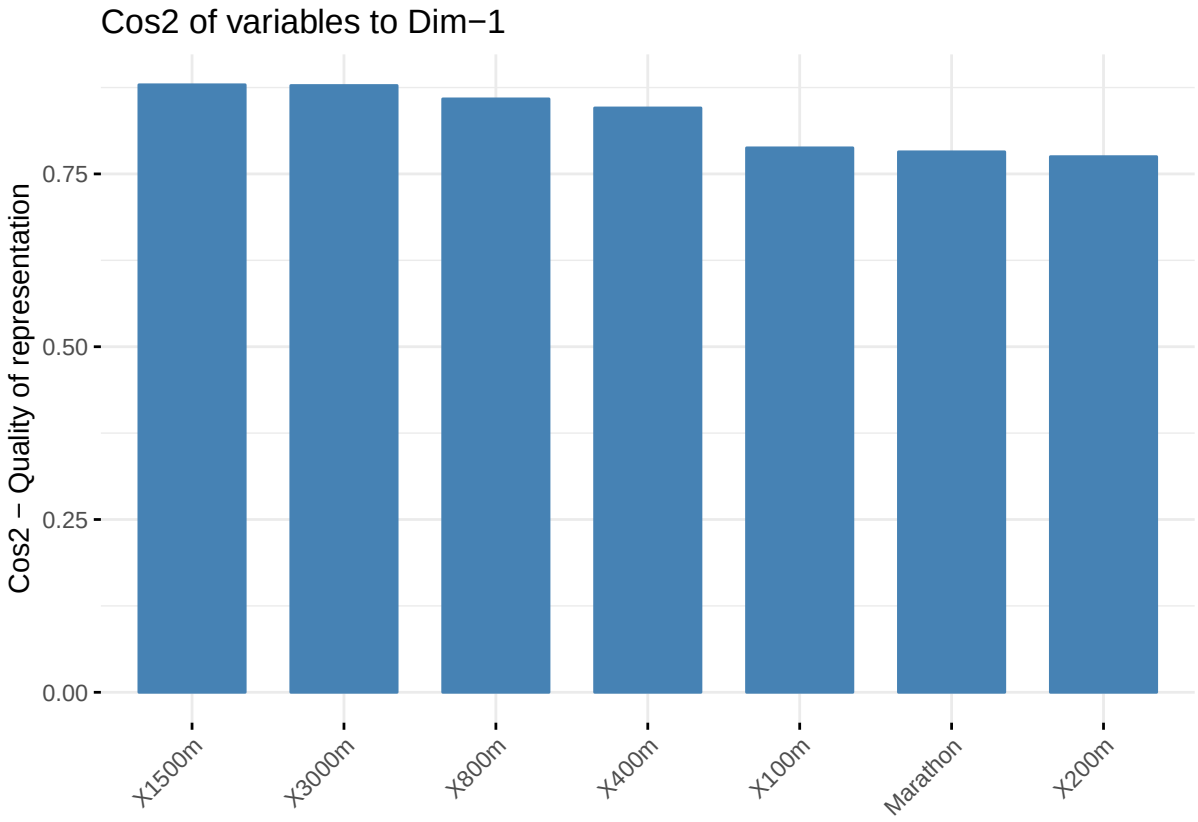

# Further Cos2 evalation A high cos2 indicates a good representation of the variable on the principal component. In this case the variable is positioned close to the circumference of the correlation circle.

A low cos2 indicates that the variable is not perfectly represented by the PCs. In this case the variable is close to the center of the circle.

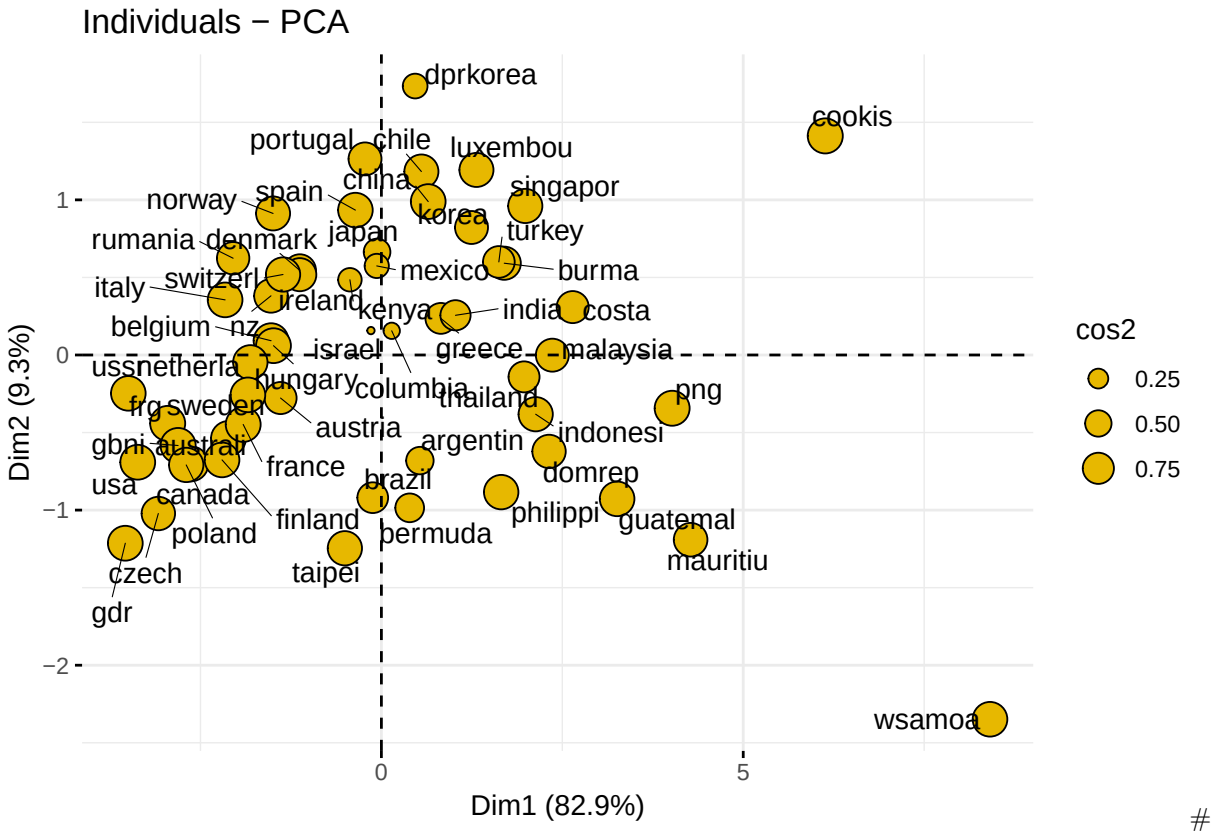
```
#We can look at the cos2 value.
#This represents the quality of representation for variables on the factor map.
# It's calculated as the squared coordinates: var.cos2 = var.coord * var.coord.
var <- get_pca_var(pca.out)
corrplot(var$cos2, is.corr=FALSE)
```




```
#It's also possible to create a bar plot of variables cos2
fviz_cos2(pca.out, choice = "var", axes = 1)
```



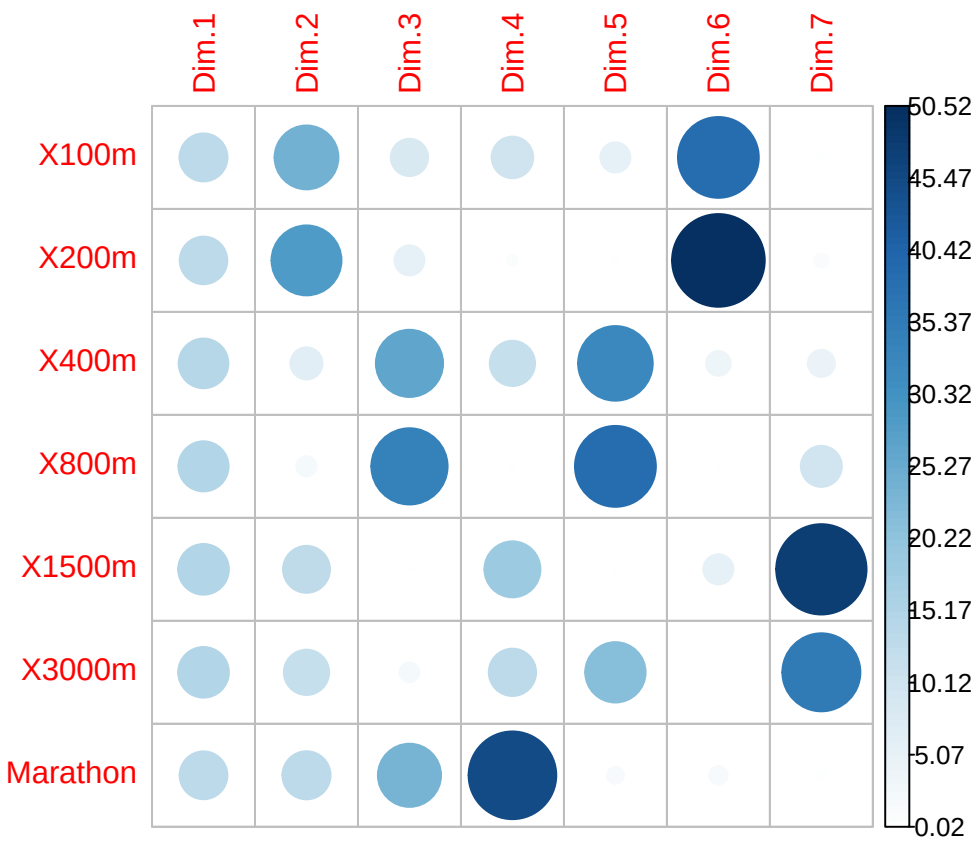

```
fviz_pca_ind(pca.out, pointsize = "cos2",
             pointshape = 21, fill = "#E7B800",
             repel = TRUE # Avoid text overlapping (slow if many points)
             )
```



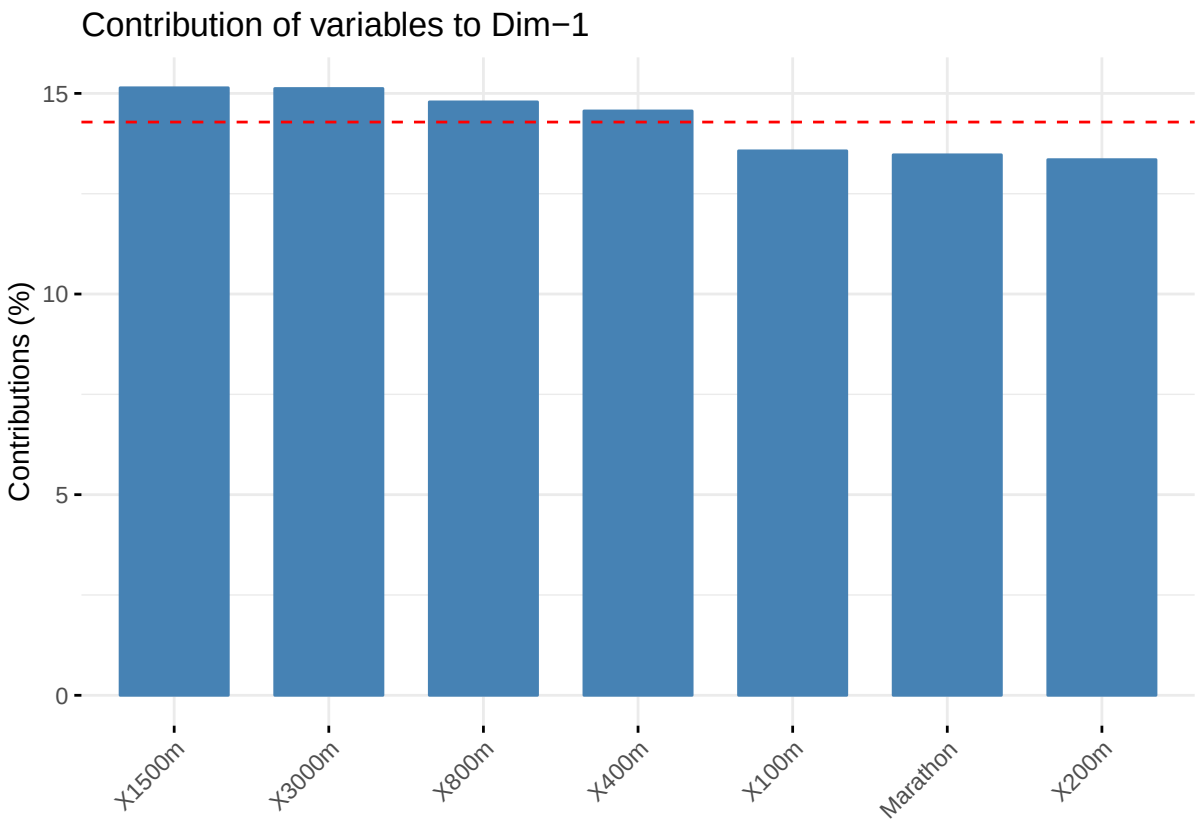

#

Dimension contribution Plots We can also look at the contribution to each principle component.

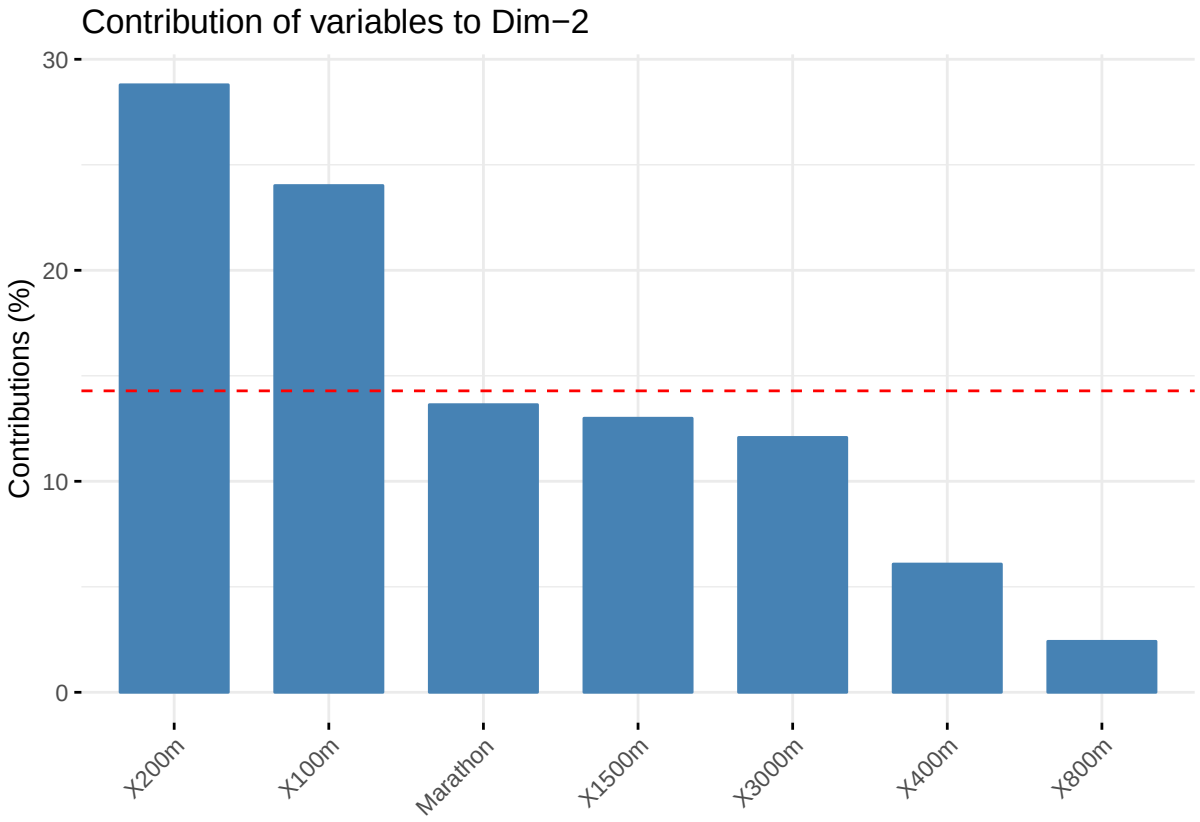
```
corrplot(var$contrib, is.corr=FALSE)
```



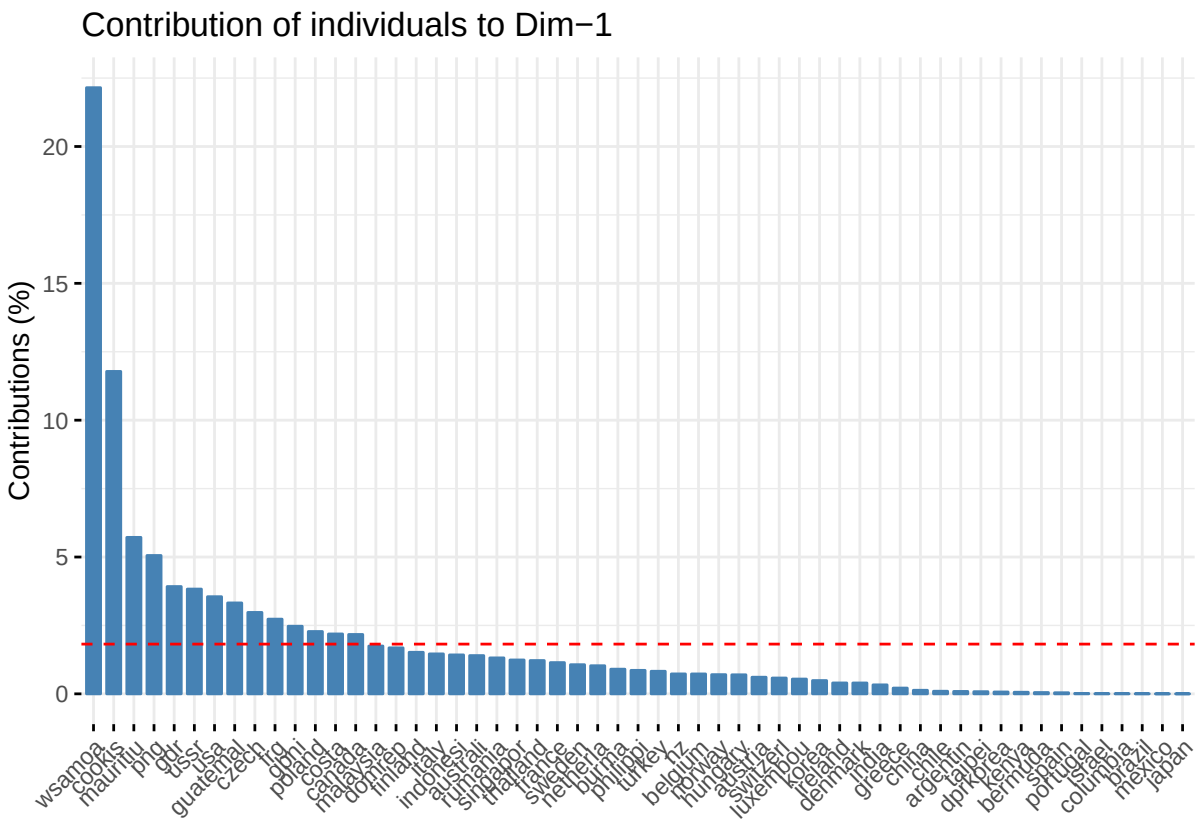

```
# Contributions of variables to PC1
fviz_contrib(pca.out, choice = "var", axes = 1, top = 7)
```



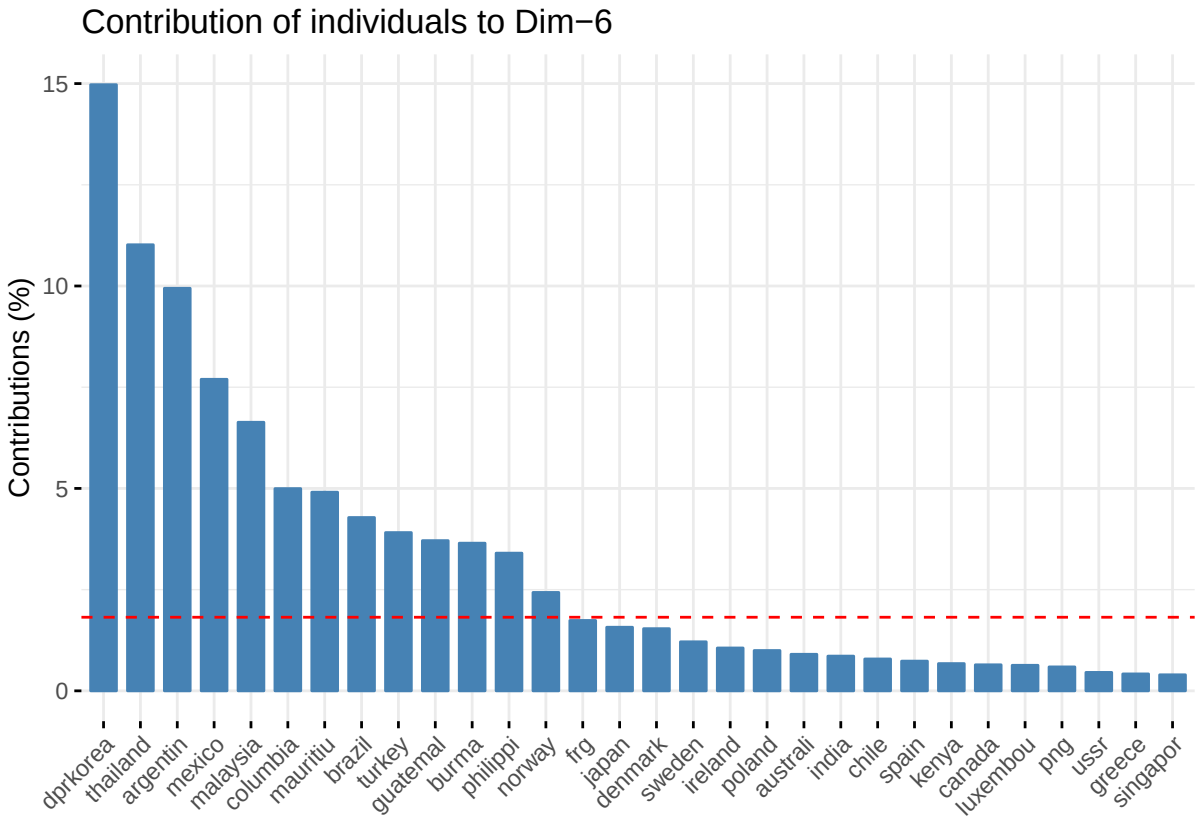

```
# Contributions of variables to PC2
fviz_contrib(pca.out, choice = "var", axes = 2, top = 7)
```




```
#contribution to first 2 principle components
fviz_contrib(pca.out, choice = "ind", axes = 1)
```

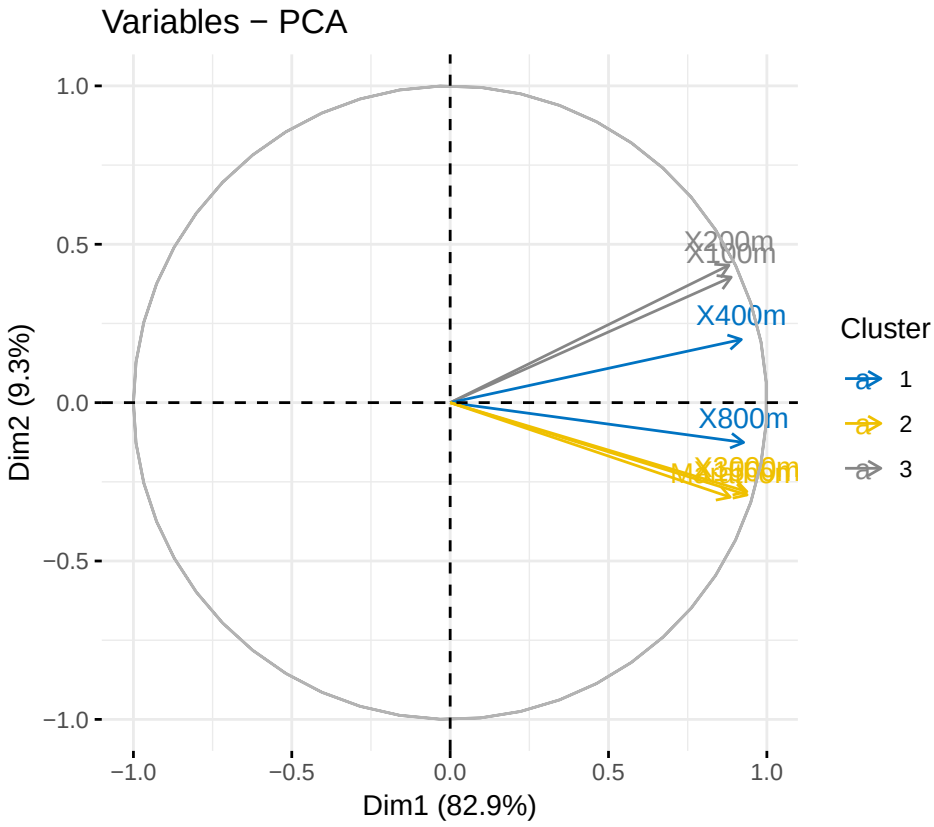



```
fviz_contrib(pca.out, choice = "ind", axes = 6, top = 30)
```




# Variable contribution plot We can also look at the contribution by each variable

```
# Create a grouping variable using kmeans
# Create 3 groups of variables (centers = 3)
set.seed(123)
res.km <- kmeans(var$coord, centers = 3, nstart = 25)
grp <- as.factor(res.km$cluster)
# Color variables by groups
fviz_pca_var(pca.out, col.var = grp,
             palette = c("#0073C2FF", "#EFC000FF", "#868686FF"),
             legend.title = "Cluster")
```


Variables − PCA
1.0
0.5
0.0
−0.5
−1.0
Dim2 (9.3%)
−1.0
−0.5
0.0
0.5
1.0
Dim1 (82.9%)
X200m
X400m
X800m
Cluster
1
2
3