Dimension Reduction

Principal Components Analysis (PCA)

Dimension Reduction

Decrease the number of the variables (dimensions)

$$\mathbf{X} = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(p)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(p)} \\ \cdots & \cdots & \cdots & \cdots \\ x_n^{(1)} & x_n^{(2)} & \cdots & x_n^{(p)} \end{pmatrix} \rightarrow \mathbf{Z} = \begin{pmatrix} z_1^{(1)} & z_1^{(2)} & \cdots & z_1^{(p')} \\ z_2^{(1)} & z_2^{(2)} & \cdots & z_2^{(p')} \\ \cdots & \cdots & \cdots & \cdots \\ z_n^{(1)} & z_n^{(2)} & \cdots & z_n^{(p')} \end{pmatrix}$$

$$p' \ll p$$

Why?

- Find latent/hidden variables that are not/cannot be directly measured
- Reveal the hidden structure of the data
- Transform the feature space into variables that are not correlated, thus new variables can be used in different machine learning techniques

Principal Components Analysis (PCA)

Principal Components Analysis (PCA)

- PCA is a technique that can be used to simplify a dataset.
- It is a linear transformation that chooses a new coordinate system for the data set such that greatest variance by any projection of the data set comes to lie on the first axis (called the first principal component), the second greatest variance on the second axis, and so on.
- PCA can be used for reducing dimensionality by eliminating the later principal components.

Principal Components Analysis (PCA)

Each new Component/factor is a linear combination of all variables.

Imagine we have p variables ($X_j, j = 1, 2, ..., p$)

$$Z_i = \sum_{i=1}^{p} w_{ij} X_j$$

$$Z_i = w_{i1}X_1 + w_{i2}X_2 + w_{i3}X_3 + ... + w_{ip}X_p$$

where each pair of Z's are orthogonal (correlation $= 0$)!
After estimating weights w, Z's are ordered by their variance, with Z_1 having the largest variance and Z_p having the smallest variance.

Estimating principal components weights

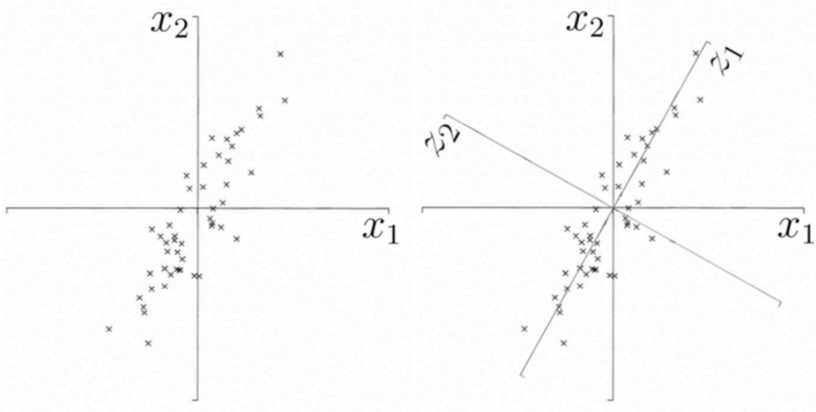
$$Z_i = w_i^{'} X$$

where $w_k^{'}$ is *eigenvector* of Σ (covariance matrix of initial variables $X_j, j = 1, 2, ..., p$) corresponding to its kth largest *eigenvalue* λ_k.
Furthermore, if w_k is chosen to have unit length ($w_k^{'} w_k = 1$), then $var(Z_k) = \lambda_k$.

A standard approach to stimate the unknown weights is the technique of Lagrange multipliers.

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq ... \geq \lambda_p$$

Geometric picture of principal components



- The 1st PC Z_1 is a minimum distance fit to a line in X space
- The 2nd PC Z_2 is a minimum distance fit to a line in the plane perpendicular to the 1st PC

PCA in R: Example

Annual food consumption in Armenia

```
consumption<-read.csv("consumption.csv")
str(consumption)
```

```
## 'data.frame':    4000 obs. of  6 variables:
##  $ bread_1_flour_white.kg: num  100 133.89 0.5 1.25 50 ...
##  $ bread_2_cereal.kg     : num  3.33 1.67 1.5 7.5 1.25 25 1.25 2.5 1.5 3 ...
##  $ bread_3_rice.kg       : num  1.67 1.33 1.5 3.75 2.5 10 1.25 1.67 1.5 3.5
##  $ bread_4_beans.kg      : num  1.33 1 0.5 1.25 2.5 5 0.75 1 2.5 2 ...
##  $ bread_5_macaroni.kg   : num  6 6.67 5 3.75 7.5 30 2.5 5 10 8.75 ...
##  $ bread_6_lavash.kg     : num  155 166.7 50 195 58.8 ...
```

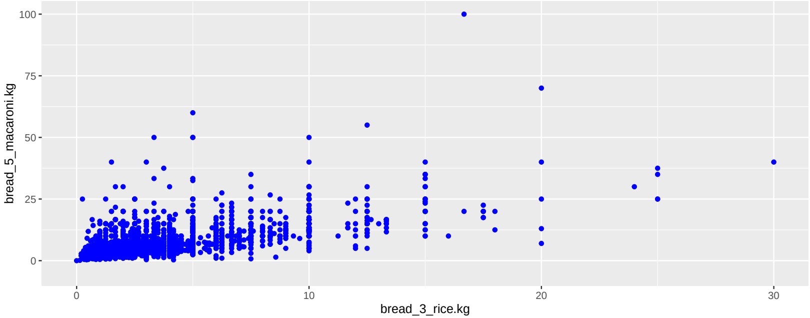
Descriptive Analysis

```r
round(data.frame(
  mean=sapply(consumption, mean),
  sd=sapply(consumption, sd),
  min=sapply(consumption, min),
  max=sapply(consumption, max),
  median=sapply(consumption, median)),3)
```

```
##                            mean     sd min    max median
## bread_1_flour_white.kg   38.569 70.198   0 550.00      5
## bread_2_cereal.kg         3.016  5.182   0 187.50      2
## bread_3_rice.kg           3.682  2.911   0  30.00      3
## bread_4_beans.kg          2.772  2.602   0  30.00      2
## bread_5_macaroni.kg       7.353  5.932   0 106.67      6
## bread_6_lavash.kg       115.343 79.294   0 636.00    100
```

Let's take two products: rice and macaroni

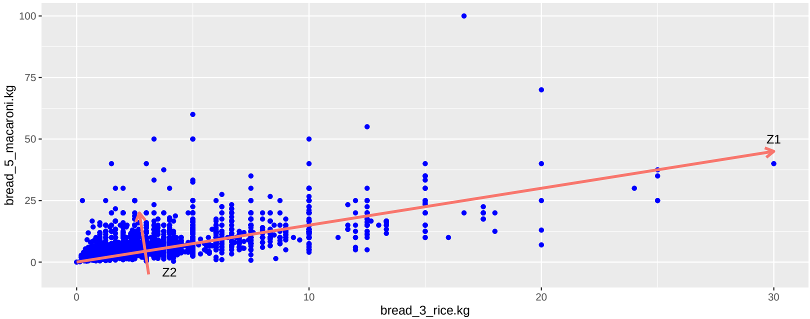
```
ggplot(consumption, aes(x=bread_3_rice.kg, y=bread_5_macaroni.kg)) +
  geom_point(color="blue", size=1.5)+xlim(0,30)+ylim(-5,100)
```




```
cor(consumption$bread_3_rice.kg, consumption$bread_5_macaroni.kg)
```

```
## [1] 0.603148
```

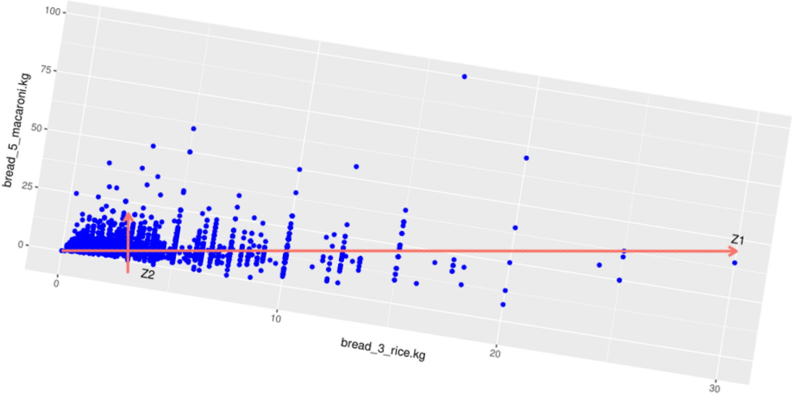
Let's take two products: rice and macaroni



- Z_1 accounts for the highest variance
- Z_1 and Z_2 are orthogonal, so the correlation is zero

This approach is called varimax orthogonal rotation, meaning that factors are orthogonal.

Rotated Space



Other considerations

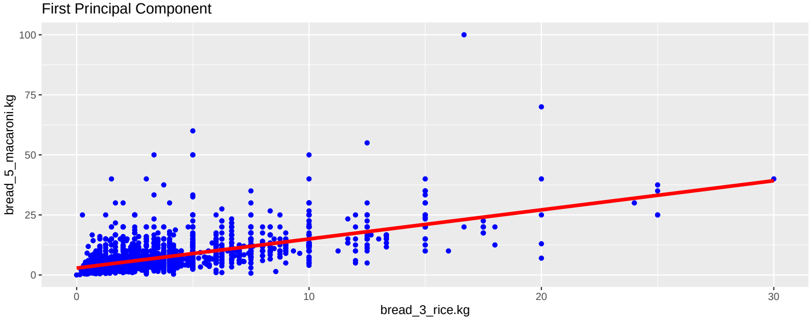
- Principal component analysis is otherwise called Factor Analysis
- Is done only with numeric variables
- PCA for categorical variables is called Correspondence analysis

Base R uses 2 methods of PCA,

- *princomp* uses covariance matrix
- *prcomp* uses Singular Value Decomposition
- The results are usually very similar

First Principal Component

```r
ggplot(consumption, aes(x=bread_3_rice.kg, y=bread_5_macaroni.kg)) +
  geom_point(color="blue", size=1.5)+xlim(0,30)+ylim(0,100)+
  geom_smooth(method = "lm", se=F, size=1.5, col="red")+
  ggtitle("First Principal Component")
```



Run PCA using base R **prcomp** function

```
p_comp <- prcomp(consumption[,c("bread_3_rice.kg", "bread_5_macaroni.kg")])
names(p_comp)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

Principal components score for each observation

```
head(p_comp$x)
```

```
##             PC1       PC2
## [1,]   1.9338107  1.463289
## [2,]   1.4107266  2.002625
## [3,]   2.9347612  1.298978
## [4,]   3.3854453 -1.235166
## [5,]   0.2454652  1.165985
## [6,] -23.4705820  1.387580
```

Summary

```r
summary(p_comp)
```

```
## Importance of components:
##                           PC1    PC2
## Standard deviation     6.2264 2.2119
## Proportion of Variance 0.8879 0.1121
## Cumulative Proportion  0.8879 1.0000
```

Proportion of variance explained by each factor/component

- First component (PC1) explains 89% of the variance
- Second component explains 11% of the variance

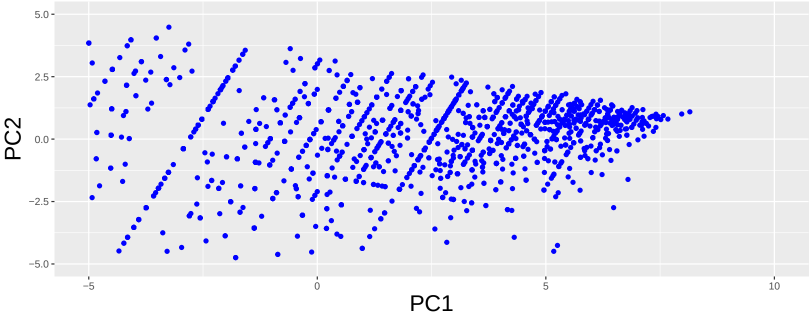
How to compute proportions of explained variance with **standard deviations**

```r
(6.2264)^2/((6.2264)^2+(2.2119)^2)
```

```
## [1] 0.8879423
```

Plot components

```
ggplot(as.data.frame(p_comp$x), aes(x = PC1, y = PC2)) +
  geom_point(color = "blue", size = 1.5) + xlim(-5, 10) + ylim(-5, 5) +
  theme(axis.title = element_text(size = 20))
```



Calculate PCA scores by hand

Lets Create a dataframe with original variables and the PCA

```
d1<-data.frame(consumption[,c("bread_3_rice.kg", "bread_5_macaroni.kg")], p_com
summary(d1)
```

```
##  bread_3_rice.kg  bread_5_macaroni.kg      PC1                 PC2
##  Min.   : 0.000   Min.   :  0.000     Min.   :-95.976   Min.   :-15.5461
##  1st Qu.: 1.670   1st Qu.:  3.750     1st Qu.: -2.281   1st Qu.: -0.8732
##  Median : 3.000   Median :  6.000     Median :  1.501   Median :  0.2868
##  Mean   : 3.682   Mean   :  7.353     Mean   :  0.000   Mean   :  0.0000
##  3rd Qu.: 5.000   3rd Qu.: 10.000     3rd Qu.:  3.882   3rd Qu.:  1.1382
##  Max.   :30.000   Max.   :106.670     Max.   :  8.151   Max.   : 26.3113
```

You can see that the means for components is equal to zero (because of scalling)

Now lets claculate the PC scores manually

Mean for rice = 3.682

Mean for macaroni = 7.353

Rotation provides loadings

```
p_comp$rotation
```

```
##                            PC1        PC2
## bread_3_rice.kg     -0.3250778 -0.9456873
## bread_5_macaroni.kg -0.9456873  0.3250778
```

```
head(d1, n=2)
```

```
##   bread_3_rice.kg bread_5_macaroni.kg      PC1      PC2
## 1            1.67                6.00 1.933811 1.463289
## 2            1.33                6.67 1.410727 2.002625
```

Lets calculate the score of the PC1 for the first case and PC2 for teh second case

```
(1.67-3.682)*(-0.3250778) + (6.00-7.353)*(-0.9456873)
```

```
## [1] 1.933571
```

```
(1.33-3.682)*(-0.9456873) + (6.67-7.353)*0.3250778
```

```
## [1] 2.002228
```

Correlations

```
round(cor(d1),4)
```

```
##                    bread_3_rice.kg bread_5_macaroni.kg     PC1     PC2
## bread_3_rice.kg             1.0000              0.6031 -0.6954 -0.7186
## bread_5_macaroni.kg         0.6031              1.0000 -0.9926  0.1212
## PC1                        -0.6954             -0.9926  1.0000  0.0000
## PC2                        -0.7186              0.1212  0.0000  1.0000
```

Lets go back to variance

Calculate total variance of the original data

```
cov(d1[,1:2])
```

```
##                    bread_3_rice.kg bread_5_macaroni.kg
## bread_3_rice.kg            8.47237            10.41425
## bread_5_macaroni.kg       10.41425            35.18871
```

Variance for rice = 8.47237; Variance for macaroni = 35.18871

Covariance = 10.41425

Total Variance=8.47237+35.18871=43.66108

% of variance explained by macaroni 35.18871/43.66108=80%

If we would like to do dimensionality reduction and keep only macaroni (highest variance) we would be able to keep only 80% of original information (lose 20%)

If we take PC1 we will be able to keep 89% of the original information

Run PCA for all bread products

Note: scale the initial variable to have a **unit variance**

```
colnames(consumption)
```

```
## [1] "bread_1_flour_white.kg" "bread_2_cereal.kg"
## [3] "bread_3_rice.kg"        "bread_4_beans.kg"
## [5] "bread_5_macaroni.kg"    "bread_6_lavash.kg"
```

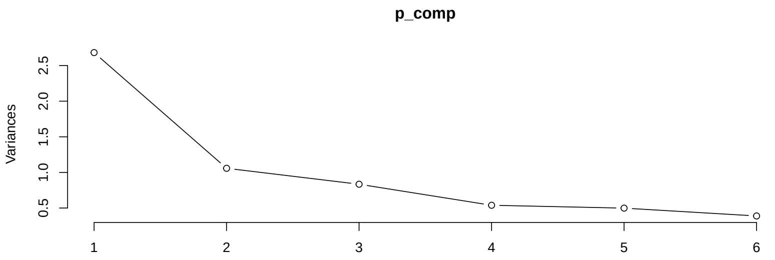
```
p_comp <- prcomp(consumption,  scale=T)
summary(p_comp)
```

```
## Importance of components:
##                          PC1    PC2    PC3    PC4     PC5     PC6
## Standard deviation     1.638 1.0285 0.9132 0.7340 0.70605 0.62345
## Proportion of Variance 0.447 0.1763 0.1390 0.0898 0.08308 0.06478
## Cumulative Proportion  0.447 0.6233 0.7623 0.8521 0.93522 1.00000
```

By default the number of extracted components is equal to the number of variables in the model

- Now, how many components to take? Remember we are doing dimensionality reduction
- There are extracted as many components as many variables we have.
- When standardized, each variable has a variance/stdev of 1
- If the component has variance < 1, then the component explains less than 1 variable from original dataset
- So take only components that have variance (also called eigenvalues) of greater than 1.
- Use Screeplot

```r
screeplot(p_comp, type="l")
```



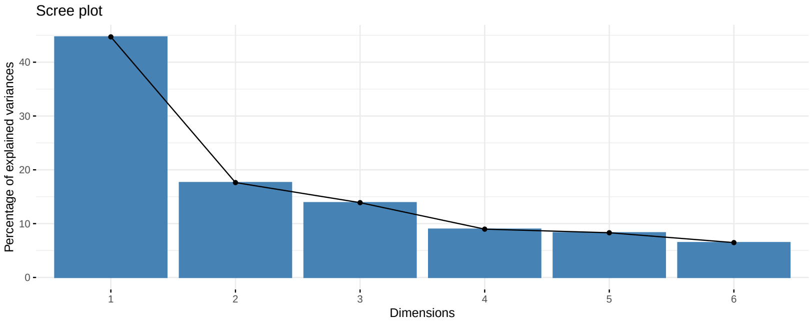

```r
# Eigenvalues
p_comp$sdev^2
```

```
## [1] 2.6821930 1.0578477 0.8339716 0.5387955 0.4985022 0.3886898
```

First 3 components together explain the biggest part of the variance (67%), we can keep them, however eigenvalues suggest that 2 components are enough (eigenvalues > 1)

Percentage of explained variances

```
fviz_eig(p_comp)
```



Correlation matrix

```
df <- data.frame(consumption, p_comp$x[, 1:2])
cor_mat<-cor(df)
```

Lets get the correlation matrix in a way that by columns we will have PC components only and by the rows we will get the correlations

```
cor_mat <- cor_mat[! rownames(cor_mat)  %in% c("PC1", "PC2"),
                 colnames(cor_mat) %in% c("PC1", "PC2")]
print(cor_mat, digits=3)
```

```
##                            PC1     PC2
## bread_1_flour_white.kg -0.385  0.8332
## bread_2_cereal.kg      -0.496  0.0870
## bread_3_rice.kg        -0.823 -0.0540
## bread_4_beans.kg       -0.763  0.1348
## bread_5_macaroni.kg    -0.800 -0.0825
## bread_6_lavash.kg      -0.623 -0.5728
```

```r
print(cor_mat, digits=3)
```

```
##                              PC1     PC2
## bread_1_flour_white.kg -0.385  0.8332
## bread_2_cereal.kg      -0.496  0.0870
## bread_3_rice.kg        -0.823 -0.0540
## bread_4_beans.kg       -0.763  0.1348
## bread_5_macaroni.kg    -0.800 -0.0825
## bread_6_lavash.kg      -0.623 -0.5728
```

Lets see which variables are highly correlated with which factors, lets take a threshold of 0.6 for correlation coefficient

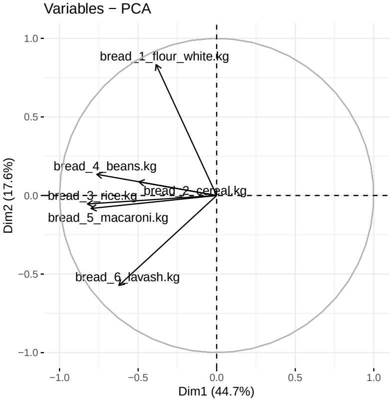
PC1 is described by rice, macaroni and beans

PC2 is described by flour_white

Visualization of the variables on the factor map

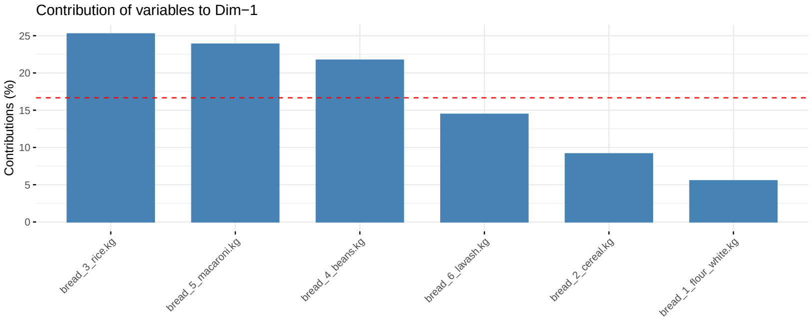
Correlation circle can help to visualize the most correlated variables

```
fviz_pca_var(p_comp, repel = TRUE)      # Avoid text overlapping
```



Contributions of variables on PC1

```
fviz_contrib(p_comp, choice = "var", axes = 1)
```



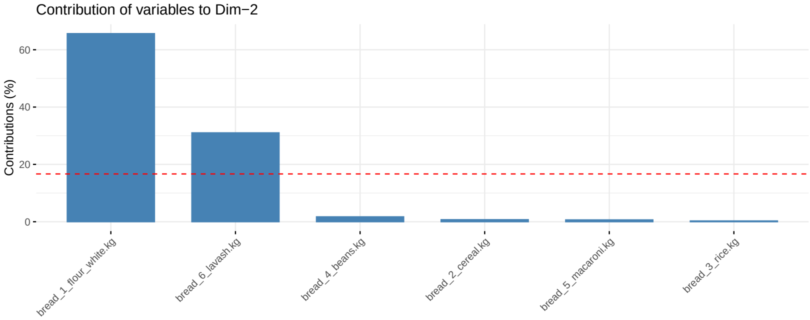
What means the red line on the graph?

If the contribution of the variables were uniform, the expected value would be $1/\text{length(variables)} = 1/6 = 16.67\%$.

The red dashed line on the graph above indicates the expected average contribution. For a given component, a variable with a contribution larger than this cutoff could be considered as important in contributing to the component.

Contributions of variables on PC2

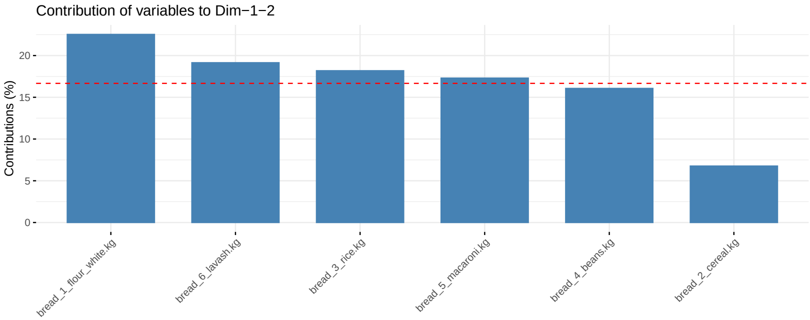
```r
fviz_contrib(p_comp, choice = "var", axes = 2)
```



Total contribution on PC1 and PC2

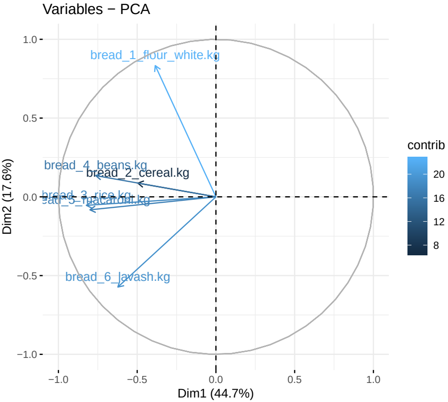
```r
fviz_pca_contrib(p_comp, choice = "var", axes = 1:2)
```

```
## Warning in fviz_pca_contrib(p_comp, choice = "var", axes = 1:2): The
## function fviz_pca_contrib() is deprecated. Please use the function
## fviz_contrib() which can handle outputs of PCA, CA and MCA functions.
```



Control variable colors using their contributions

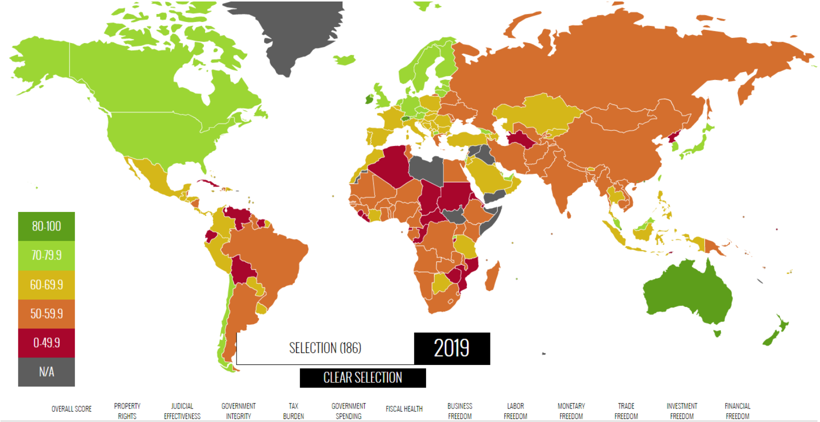
```
fviz_pca_var(p_comp, col.var="contrib")
```



Construct Economic Freedom Index for countries with PCA

The Heritage Foundation measures an Economic Freedom Index of countries

The Index covers 12 freedoms – from property rights to financial freedom – in 186 countries.



COnsrtuct an Economic Freedom Index with PCA based on 6 freedom indices:

- Business.Freedom
- Labor.Freedom
- Monetary.Freedom
- Trade.Freedom
- Investment.Freedom
- Financial.Freedom

For such problems only first principal component could be used (ignoring other components)

```r
df <- read.csv("Countries.csv")
df1 <- data.frame(Country = df[,2], df[,grepl(".Freedom", colnames(df))])
df1 <- df1[complete.cases(df1), ]
rownames(df1) <- df1$Country
head(df1)
```

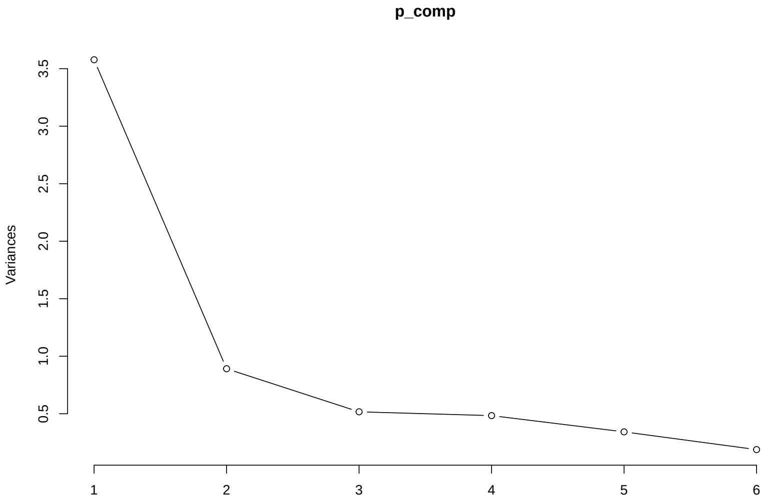
```
##                 Country Business.Freedom Labor.Freedom Monetary.Freedom
## Afghanistan Afghanistan             54.2          59.9             69.3
## Albania         Albania             79.3          50.7             81.4
## Algeria         Algeria             62.1          49.5             67.0
## Angola           Angola             58.5          40.4             70.6
## Argentina     Argentina             57.3          46.1             50.9
## Armenia         Armenia             78.5          72.4             72.8
##             Trade.Freedom Investment.Freedom Financial.Freedom
## Afghanistan          66.0                  0                 0
## Albania              87.7                 70                70
## Algeria              63.3                 35                30
## Angola               56.7                 30                40
## Argentina            66.7                 50                50
## Armenia              80.2                 80                70
```

```r
p_comp <- prcomp(df1[, -1], scale. = T)
summary(p_comp)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     1.8914 0.9440 0.71885 0.69545 0.58525 0.43396
## Proportion of Variance 0.5963 0.1485 0.08613 0.08061 0.05709 0.03139
## Cumulative Proportion  0.5963 0.7448 0.83092 0.91153 0.96861 1.00000
```

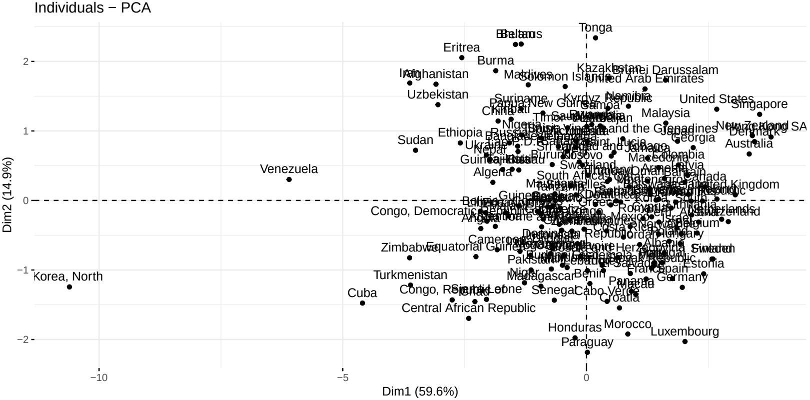
First principal component explain 60% of total variation!

```r
screeplot(p_comp, type="l")
```



Visualization of the individuals using factoextra package

```r
fviz_pca_ind(p_comp)
```



Run PCA one more time and extract only the first component

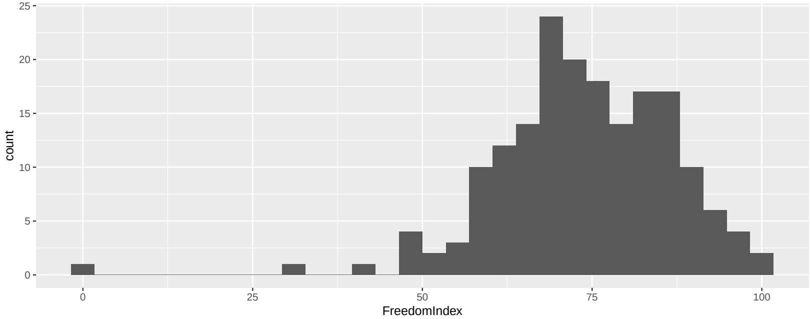
```r
p_comp<-prcomp(df1[, -1], scale. = T, rank. = 1)
head(p_comp$x)
```

```
##                    PC1
## Afghanistan -3.088722
## Albania      1.592204
## Algeria     -1.922102
## Angola      -2.169042
## Argentina   -1.872372
## Armenia      1.598681
```

Construct Economic Freedom Index with Min-Max normalization

```
df1$Score <- as.numeric(p_comp$x)
df1$FreedomIndex <- (df1$Score - min(df1$Score))/(max(df1$Score)-min(df1$Score)

ggplot(df1, aes(x = FreedomIndex)) + geom_histogram()
```



Correlation of Index and individual components

```
cor <- cor(df1[,c(colnames(df1)[grepl(".Freedom", colnames(df1))],
                  "FreedomIndex")])
data.frame(cor = cor[-nrow(cor), "FreedomIndex"])
```

```
##                           cor
## Business.Freedom   0.7833618
## Labor.Freedom      0.5722339
## Monetary.Freedom   0.7680385
## Trade.Freedom      0.8125706
## Investment.Freedom 0.8136312
## Financial.Freedom  0.8510606
```

Rank countries by economic Freedom Index

Top 10 countries

```
df1$FreedomIndexRank <- rank(df1$FreedomIndex)
df1$FreedomIndexRank <- nrow(df1) - df1$FreedomIndexRank +1
df1 <- df1[order(df1$FreedomIndexRank), ]
rownames(df1) <- NULL

head(df1[, c("Country", "Score", "FreedomIndex", "FreedomIndexRank")], 10)
```

```
##           Country    Score FreedomIndex FreedomIndexRank
## 1   Hong Kong SAR 3.784657    100.00000                1
## 2       Singapore 3.549703     98.36754                2
## 3         Denmark 3.445438     97.64310                3
## 4     New Zealand 3.400626     97.33175                4
## 5       Australia 3.336062     96.88315                5
## 6  United Kingdom 3.051521     94.90616                6
## 7     Switzerland 2.907249     93.90376                7
## 8     Netherlands 2.771262     92.95892                8
## 9         Ireland 2.674441     92.28621                9
## 10  United States 2.670496     92.25879               10
```

Rank countries by economic Freedom Index

Bottom 10 countries

```
tail(df1[, c("Country", "Score", "FreedomIndex", "FreedomIndexRank")], 10)
```

```
##                Country     Score FreedomIndex FreedomIndexRank
## 171 Congo, Republic of  -2.756953     54.54880              171
## 172         Uzbekistan  -3.048349     52.52417              172
## 173        Afghanistan  -3.088722     52.24366              173
## 174              Sudan  -3.507688     49.33267              174
## 175       Turkmenistan  -3.611616     48.61058              175
## 176               Iran  -3.626171     48.50945              176
## 177           Zimbabwe  -3.629881     48.48367              177
## 178               Cuba  -4.597460     41.76093              178
## 179          Venezuela  -6.101804     31.30872              179
## 180       Korea, North -10.607942      0.00000              180
```

t-Distributed Stochastic Neighbor Embedding (t-SNE)

PCA vs t-NSE

PCA is a **linear** feature extraction technique. It performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized.

t-SNE is a **non-linear** technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. It is extensively applied in image processing, NLP, genomic data and speech processing.

t-SNE

The algorithms starts by calculating the probability of similarity of points in high-dimensional space and calculating the probability of similarity of points in the corresponding low-dimensional space. The similarity of points is calculated as the conditional probability that a point i would choose point j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian (normal distribution) centered at i.

$$p_{j|i} = \frac{exp\frac{-||x_i - x_j||^2}{2\sigma_i^2}}{\sum_{k \neq i}(exp\frac{-||x_i - x_k||^2}{2\sigma_k^2})}$$

where σ_i is the variance of the Gaussian that is centered on datapoint x_i
Similar conditional probabilities are defined for the low-dimensional counterparts y_i and y_j of x_i and x_j

$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i}(exp(-||y_i - y_k||^2)}$$

Since we are only interested in modeling pairwise similarities, we set $p_{i|i} = q_{i|i} = 0$

t-SNE

It then tries to minimize the difference between these conditional probabilities (or similarities) in higher-dimensional and lower-dimensional space for a perfect representation of data points ($p_{j|i} = q_{j|i} = 0$) in lower-dimensional space (2D or 3D space).

In simpler terms, t-SNE minimizes the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding.

In this way, t-SNE maps the multi-dimensional data to a lower dimensional space and attempts to find patterns in the data by identifying observed clusters based on similarity of data points with multiple features.

However!!! after this process, the **input features are no longer identifiable**, and you cannot make any inference based only on the output of t-SNE. Hence it is mainly a **data exploration and visualization technique**.

t-SNE in R

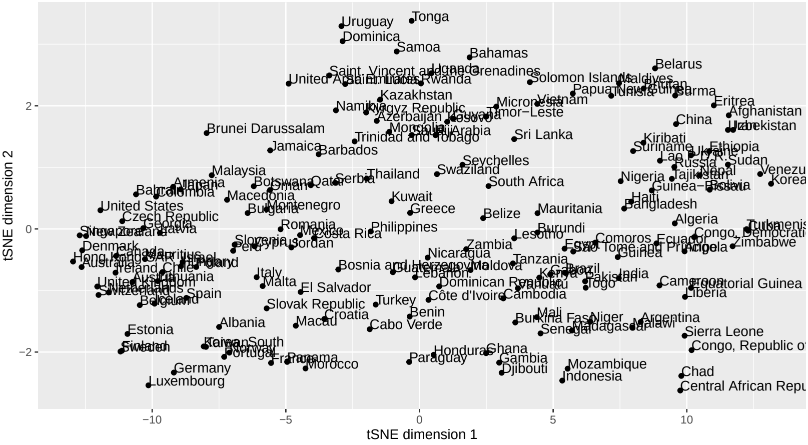
```
#library(Rtsne)
set.seed(1)
tsne <- Rtsne(df1[, -1], dims = 2)
dims <- data.frame(Country = df1$Country, tsne$Y)
head(dims)
```

```
##             Country        X1         X2
## 1   Hong Kong SAR -12.96044 -0.5297066
## 2       Singapore -12.71198 -0.1047265
## 3         Denmark -12.61775 -0.3477270
## 4     New Zealand -12.47197 -0.1160881
## 5       Australia -12.64009 -0.6194472
## 6 United Kingdom -12.05516 -0.9358272
```

The object **tsne$Y** contains coordinates of countries in 2D space

Visualizing t-SNE

```r
ggplot(dims, aes(x = X1, y = X2, label = Country)) +
  geom_point() + geom_text(aes(label = Country), hjust = 0, vjust = 0) +
  xlab("tSNE dimension 1") + ylab("tSNE dimension 2")
```



Thank You!

Questions?