**IDN0110. HA2. Report prepared by:** Lizaveta_Miasayedava (184361IVGM)

## Exercise 1. Cosine distance function

**Implementation:** *my_distfunc.R, my_text_processing.R, test_cosine.R*

Test text data (a list of 3 texts)*:*

```
text1 <- "word1 word1 word2", text2 <- "word2 word2", text3 <- "word1 word2"
```

*Data preparation:*

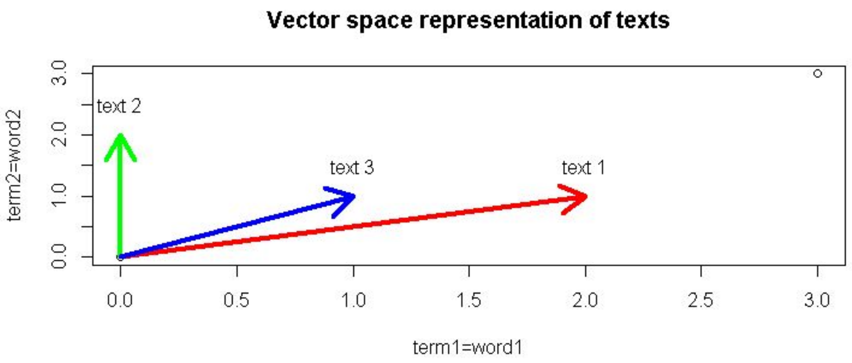
1) Tokenization (decomposing texts into tokens - distinct words/terms).

```
tokens from 3 documents
  text1 : [1] "word1" "word1" "word2"
  text2 : [1] "word2" "word2"
  text3 : [1] "word1" "word2"
```

2) Building a document-frequency matrix (DFM) to study document similarity: each row represents a document (text), each column - a distinct token, each cell - the frequency of occurrence of the token in the text. DFM = a Bag-Of-Words (BOW) model of the test data:

```
docs    word1 word2
 text1     2     1
 text2     0     2
 text3     1     1
```

*Interpretation:* Documents (texts) can be represented as vectors of numbers (frequencies, rows of DFM) that allows for the geometric interpretation of documents (vector space representation of texts).



By means of cosine distance function it’s possible to measure angles between documents/texts:

```
[1] "Cosine distance between text1 and text2:  0.447213595499958 , angle (degrees): 
63.434948822922"
[1] "Cosine distance between text1 and text3:  0.948683298050514 , angle (degrees): 
18.434948822922"
[1] "Cosine distance between text2 and text3:  0.707106781186547 , angle (degrees):  45"
```

The longer the cosine distance (the less the angle) between texts, the more similar texts are.

## Exercise 2 and 3. Text clustering and classification

Text dataset: https://www.kaggle.com/team-ai/spam-text-message-classification/version/1

Description of the dataset: text messages in a *spam.csv* file with columns for type (“spam” or “ham”) and the text of the message.

Goal: ham/spam classification.

**Implementation:** *test_text_mining.R*

**Workflow:**

1) Text preprocessing (cleaning, low casing, the removal of punctuation, numbers, symbols, stop words, stemming). Tokenization of preprocessed texts.

2) Normalization of texts (to make them length independent):

calculate Term Frequency (TF) - the proportion of the term t frequencies in the text d: *TF(t,d)=freq(t,d)/sum(freq(ti,d));*

since terms that appear in many texts may have a low predictive power, calculate Inverse Document Frequency (IDF): *IDF(t)=log(N/count(t)),* N - the number of texts in the corpus, count(t) - the number of texts in the corpus in which the term t is present;

calculate TF-IDF (the combination of TF and IDF to enhance DFM): *TF-IDF(t,d)=TF(t,d)*IDF(t).*

**Implementation:** *my_text_processing.R*

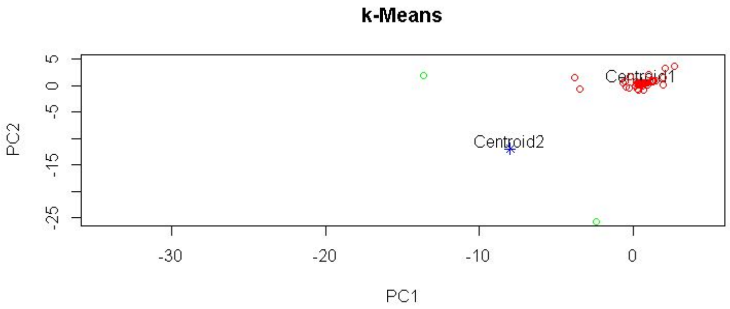
**Intermediate result:** multidimensional normalized single term (1-gram) DFM with TF-IDF values.

3) Principal Component Analysis to reduce dimensionality:

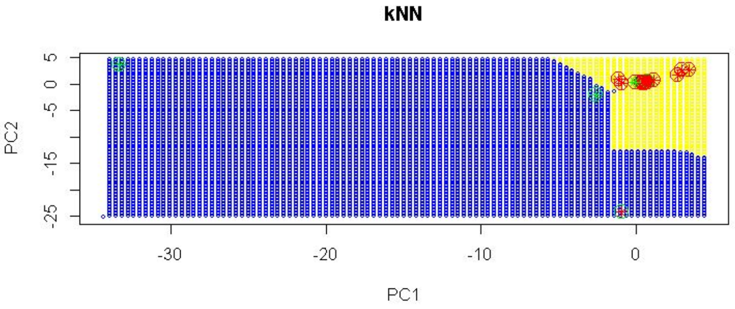
Assumption: since it's difficult to visualize a multi-dimensional hyperspace having more than 3 terms (dimensions) in DFM, it was decided to perform Principal Component Analysis (PCA) over the data to get a set of principal components (linear combinations of original vectors). Since PCA allows to find directions along which the variation in the data is maximal, it's possible to reduce the number of dimensions (considering that directions with the largest variances are more important) and the loss of information during interpretation projecting texts onto the principal components with the largest variation (measured by eigenvalues) in case of redundancy in the texts (high correlation).

4) Clustering (k-Means) and classification (kNN) over 2 principal components of the largest variation percentage.

**Implementation:** *my_knn.R, my_kmeans.R*



Visualization of clusters of messages and their centroids built with the k-means algorithm (k=2)



Visualization of classified messages built with the k-nearest neighbours algorithm (k=2) and classification decision boundary