R Notebook

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

```
library(tidyverse) # metapackage of all tidyverse packages
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tokenizers)
library(stopwords)
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
##
## Attaching package: 'tm'
```

```
## The following object is masked from 'package:stopwords':
##
##     stopwords
```

```
library(text2vec)
```

```
library(NbClust)
library(cluster)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(dbscan)
```

Wczytanie i przetworzenie danych

```
data = read.csv("mbti_1.csv")

n <- 1000 #sample from data

data <- data[sample(nrow(data), n), ]
```

```
INTJ <- data[data$type=="INTJ",]
INTP <- data[data$type=="INTP",]
ENTJ <- data[data$type=="ENTJ",]
ENTP <- data[data$type=="ENTP",]
INFJ <- data[data$type=="INFJ",]
INFP <- data[data$type=="INFP",]
ENFJ <- data[data$type=="ENFJ",]
ENFP <- data[data$type=="ENFP",]
ISTJ <- data[data$type=="ISTJ",]
ISFJ <- data[data$type=="ISFJ",]
ESTJ <- data[data$type=="ESTJ",]
ESFJ <- data[data$type=="ESFJ",]
ISTP <- data[data$type=="ISTP",]
ISFP <- data[data$type=="ISFP",]
ESTP <- data[data$type=="ESTP",]
ESFP <- data[data$type=="ESFP",]
```

```
#replaces URLs with word "link"
data$posts <- gsub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', 'link', data$

#removes all noise from text
data$posts <- gsub('[^a-zA-Z]', " ", data$posts)

#removes more than 1 space
data$posts <- gsub('[ ]{2,}', " ", data$posts)

#word tokenization and stemming
data$posts <- tokenize_word_stems(data$posts, stopwords = stopwords::stopwords("en"))
```

```
#create dictionary
iterator = itoken(data$posts)
vocab = create_vocabulary(iterator)
pruned_vocab = prune_vocabulary(vocab,
                                 term_count_min = 5,
                                 doc_proportion_max = 0.7,
                                 doc_proportion_min = 0.1)
pruned_vocab
```

```
## Number of docs: 1000
## 0 stopwords:  ...
## ngram_min = 1; ngram_max = 1
## Vocabulary:
```

```
##          term term_count doc_count
##   1:   spent        106       100
##   2:    plus        108       100
##   3:    warm        109       100
##   4:    upon        110       100
##   5: brought        111       101
##  ---
## 831:   thank       1625       640
## 832:    intj       1750       489
## 833:    infp       1847       500
## 834:    infj       1857       466
## 835:    link       3527       678
```

Wektoryzacja

```
#document term matrix
vectorizer = vocab_vectorizer(pruned_vocab)
dtm = create_dtm(iterator, vectorizer)

#(Term Co-occurrence Matrix)
tcm = create_tcm(iterator, vectorizer, skip_grams_window = 5L)

#tf_idf
tf_idf = TfIdf$new()
# fit tf-idf to training data
dt_tfidf = fit_transform(dtm, tf_idf)

# apply pre-trained tf-idf transformation to testing data
#doc_term_test_tfidf  = transform(doc_term_test, tf_idf)

vectors.dtm <- dtm
vectors.tfidf <- dt_tfidf
dim(dtm)
```

```
## [1] 1000  835
```

```
##########
# glove #
##########

glove = GlobalVectors$new(rank = 50, x_max = 10)
wv_main = glove$fit_transform(tcm, n_iter = 100, convergence_tol = 0.01, n_threads = 8)
```

```
## INFO  [20:47:49.654] epoch 1, loss 0.1706
## INFO  [20:47:49.852] epoch 2, loss 0.1066
## INFO  [20:47:50.048] epoch 3, loss 0.0942
## INFO  [20:47:50.219] epoch 4, loss 0.0877
## INFO  [20:47:50.363] epoch 5, loss 0.0837
## INFO  [20:47:50.514] epoch 6, loss 0.0809
## INFO  [20:47:50.654] epoch 7, loss 0.0788
## INFO  [20:47:50.797] epoch 8, loss 0.0772
## INFO  [20:47:50.939] epoch 9, loss 0.0758
## INFO  [20:47:51.084] epoch 10, loss 0.0748
## INFO  [20:47:51.234] epoch 11, loss 0.0739
## INFO  [20:47:51.366] epoch 12, loss 0.0732
## INFO  [20:47:51.510] epoch 13, loss 0.0726
## INFO  [20:47:51.511] Success: early stopping. Improvement at iterartion 13 is less then convergence_
```

```
wv_context = glove$components
word_vectors = wv_main + t(wv_context)
#wv = glove$get_word_vectors()
#dim(wv)
#wv


#If your goal is to classify documents - I doubt any doc2vec approach will beat bag-of-words/ngrams.
#If you still want to try - common simple strategy short documents (< 20 words) is to represent documen
#test= word_vectors["link", , drop=F]
#cos_sim_rom = sim2(x = word_vectors, y = test, method = "cosine", norm = "l2")
#head(sort(cos_sim_rom[,1], decreasing = T), 10)

common_terms = intersect(colnames(dtm), rownames(word_vectors) )
dtm_averaged =  normalize(dtm[, common_terms], "l1")
# you can re-weight dtm above with tf-idf instead of "l1" norm
sentence_vectors = dtm_averaged %*% word_vectors[common_terms, ]
vectors.glove <- sentence_vectors
```

```
#split rows into single post not 50
library(tidyr)
library(dplyr)

data2 = read.csv("mbti_1.csv")
data2 <- data2[sample(nrow(data2), 50), ]# do 200 jakoś idzie

#replaces URLs with word "link"
data2$posts <- gsub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', 'link', data

data2 <-data2 %>%
    mutate(posts = strsplit(as.character(posts), "\\|\\|\\|")) %>%
    unnest(posts)

data2$posts <- gsub('[^a-zA-Z]', " ", data2$posts)

#removes more than 1 space
data2$posts <- gsub('[ ]{2,}', " ", data2$posts)

data2<-subset(data2, sapply(strsplit(posts, " "), length) >=8)

#word tokenization and stemming
data2$posts <- tokenize_word_stems(data2$posts, stopwords = stopwords::stopwords("en"))

#create dictionary
iterator2 = itoken(data2$posts)
vocab2 = create_vocabulary(iterator2)

pruned_vocab2 = prune_vocabulary(vocab2,
                                 term_count_min = 5,
                                 doc_proportion_max = 0.8,
                                 doc_proportion_min = 0.001)
```

```r
#document term matrix
vectorizer2 = vocab_vectorizer(pruned_vocab2)
dtm2 = create_dtm(iterator2, vectorizer2)
#(Term Co-occurrence Matrix)
tcm2 = create_tcm(iterator2, vectorizer2, skip_grams_window = 5L)



glove2 = GlobalVectors$new(rank = 50, x_max = 10)
wv_main2 = glove2$fit_transform(tcm2, n_iter = 100, convergence_tol = 0.01, n_threads = 8)
```

```
## INFO  [20:47:52.800] epoch 1, loss 0.1302
## INFO  [20:47:52.833] epoch 2, loss 0.0738
## INFO  [20:47:52.868] epoch 3, loss 0.0596
## INFO  [20:47:52.904] epoch 4, loss 0.0516
## INFO  [20:47:52.941] epoch 5, loss 0.0462
## INFO  [20:47:52.974] epoch 6, loss 0.0420
## INFO  [20:47:53.019] epoch 7, loss 0.0386
## INFO  [20:47:53.052] epoch 8, loss 0.0358
## INFO  [20:47:53.087] epoch 9, loss 0.0334
## INFO  [20:47:53.127] epoch 10, loss 0.0313
## INFO  [20:47:53.162] epoch 11, loss 0.0295
## INFO  [20:47:53.202] epoch 12, loss 0.0280
## INFO  [20:47:53.239] epoch 13, loss 0.0266
## INFO  [20:47:53.272] epoch 14, loss 0.0254
## INFO  [20:47:53.306] epoch 15, loss 0.0242
## INFO  [20:47:53.344] epoch 16, loss 0.0232
## INFO  [20:47:53.378] epoch 17, loss 0.0223
## INFO  [20:47:53.417] epoch 18, loss 0.0215
## INFO  [20:47:53.458] epoch 19, loss 0.0208
## INFO  [20:47:53.503] epoch 20, loss 0.0201
## INFO  [20:47:53.542] epoch 21, loss 0.0194
## INFO  [20:47:53.579] epoch 22, loss 0.0188
## INFO  [20:47:53.614] epoch 23, loss 0.0183
## INFO  [20:47:53.656] epoch 24, loss 0.0178
## INFO  [20:47:53.691] epoch 25, loss 0.0173
## INFO  [20:47:53.727] epoch 26, loss 0.0169
## INFO  [20:47:53.780] epoch 27, loss 0.0165
## INFO  [20:47:53.823] epoch 28, loss 0.0161
## INFO  [20:47:53.872] epoch 29, loss 0.0158
## INFO  [20:47:53.920] epoch 30, loss 0.0154
## INFO  [20:47:53.968] epoch 31, loss 0.0151
## INFO  [20:47:54.015] epoch 32, loss 0.0148
## INFO  [20:47:54.056] epoch 33, loss 0.0145
## INFO  [20:47:54.104] epoch 34, loss 0.0142
## INFO  [20:47:54.144] epoch 35, loss 0.0140
## INFO  [20:47:54.188] epoch 36, loss 0.0137
## INFO  [20:47:54.226] epoch 37, loss 0.0135
## INFO  [20:47:54.262] epoch 38, loss 0.0133
## INFO  [20:47:54.297] epoch 39, loss 0.0131
## INFO  [20:47:54.333] epoch 40, loss 0.0129
## INFO  [20:47:54.366] epoch 41, loss 0.0127
## INFO  [20:47:54.403] epoch 42, loss 0.0125
## INFO  [20:47:54.442] epoch 43, loss 0.0123
## INFO  [20:47:54.485] epoch 44, loss 0.0122
## INFO  [20:47:54.526] epoch 45, loss 0.0120
## INFO  [20:47:54.571] epoch 46, loss 0.0119
## INFO  [20:47:54.616] epoch 47, loss 0.0117
## INFO  [20:47:54.658] epoch 48, loss 0.0116
## INFO  [20:47:54.693] epoch 49, loss 0.0114
## INFO  [20:47:54.733] epoch 50, loss 0.0113
## INFO  [20:47:54.770] epoch 51, loss 0.0112
## INFO  [20:47:54.807] epoch 52, loss 0.0110
## INFO  [20:47:54.842] epoch 53, loss 0.0109
## INFO  [20:47:54.877] epoch 54, loss 0.0108
## INFO  [20:47:54.915] epoch 55, loss 0.0107
## INFO  [20:47:54.956] epoch 56, loss 0.0106
## INFO  [20:47:54.996] epoch 57, loss 0.0105
## INFO  [20:47:54.996] Success: early stopping. Improvement at iterartion 57 is less then convergence_
```

```
wv_context2 = glove2$components
word_vectors2 = wv_main2 + t(wv_context2)
#wv = glove$get_word_vectors()
#dim(wv)
#wv



#If your goal is to classify documents - I doubt any doc2vec approach will beat bag-of-words/ngrams.
#If you still want to try - common simple strategy short documents (< 20 words) is to represent documen
#test= word_vectors["link", , drop=F]
#cos_sim_rom = sim2(x = word_vectors, y = test, method = "cosine", norm = "l2")
#head(sort(cos_sim_rom[,1], decreasing = T), 10)

common_terms = intersect(colnames(dtm2), rownames(word_vectors2) )
dtm_averaged =  normalize(dtm2[, common_terms], "l1")
# you can re-weight dtm above with tf-idf instead of "l1" norm
sentence_vectors2 = dtm_averaged %*% word_vectors2[common_terms, ]
vectors.glove_post <- sentence_vectors2
```

Grupowanie

K-medoids

```
pam.dtm = pam(dtm, 16, metric = "euclidean", stand = FALSE)
pam.tfidf = pam(dt_tfidf, 16, metric = "euclidean", stand = FALSE)
pam.glove = pam(sentence_vectors, 16, metric = "euclidean")
pam.glove2 = pam(sentence_vectors2, 16, metric = "euclidean", stand = FALSE)
```

K-medoids DTM

```
pam_results_dtm <- data %>%
mutate(cluster = pam.dtm$clustering) %>%
group_by(cluster)

pam_results <- subset(pam_results_dtm, select = c(type, cluster))
t <-  table(pam_results)
print ("Wyniki ilosciowe grupowania dla dtm")
```

```
## [1] "Wyniki ilosciowe grupowania dla dtm"
```

```
print (t)
```

```
##       cluster
## type    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
##   ENFJ  8  6  0  6  0  1  0  1  0  0  0  4  3  1  0  0
##   ENFP 17 10  8 13  1  5  0  3  0  3  2  9  7  2  0  0
##   ENTJ 14  5  1  4  0  0  0  0  0  0  1 13  5  0  0  0
##   ENTP 16  8  8 27  1  3  0  0  1  0  0  3 13  1  1  1
##   ESFJ  1  1  1  0  1  0  0  0  0  0  0  2  0  0  0  0
##   ESFP  2  1  0  2  0  0  0  0  0  0  0  0  0  0  0  0
##   ESTJ  0  1  0  3  0  0  0  0  0  0  0  1  0  1  0  0
##   ESTP  2  1  1  4  0  1  0  0  0  0  0  1  2  1  0  0
##   INFJ 92  8  9 11  0  5  2  5  1  0  4  7 13  2  0  0
##   INFP 51 35  2 18  3  8  3  2  0  0 12 25 36  5  0  0
##   INTJ 34  6  2 20  1  1  1 17  2  0  1 15 15  2  0  0
##   INTP 31  3  3 29  3  3  0  0 15  0  3 15 28  2  0  0
##   ISFJ  2  7  4  2  0  2  1  0  1  0  1  8  2  0  0  0
##   ISFP  2  2  0  5  0  0  0  0  1  0  1  9  6  2  1  0
##   ISTJ  6  1  1  8  0  0  0  0  0  0  0  3  2  0  0  0
##   ISTP 18  3  1  3  0  2  2  0  1  0  3  4  5  1  0  0
```

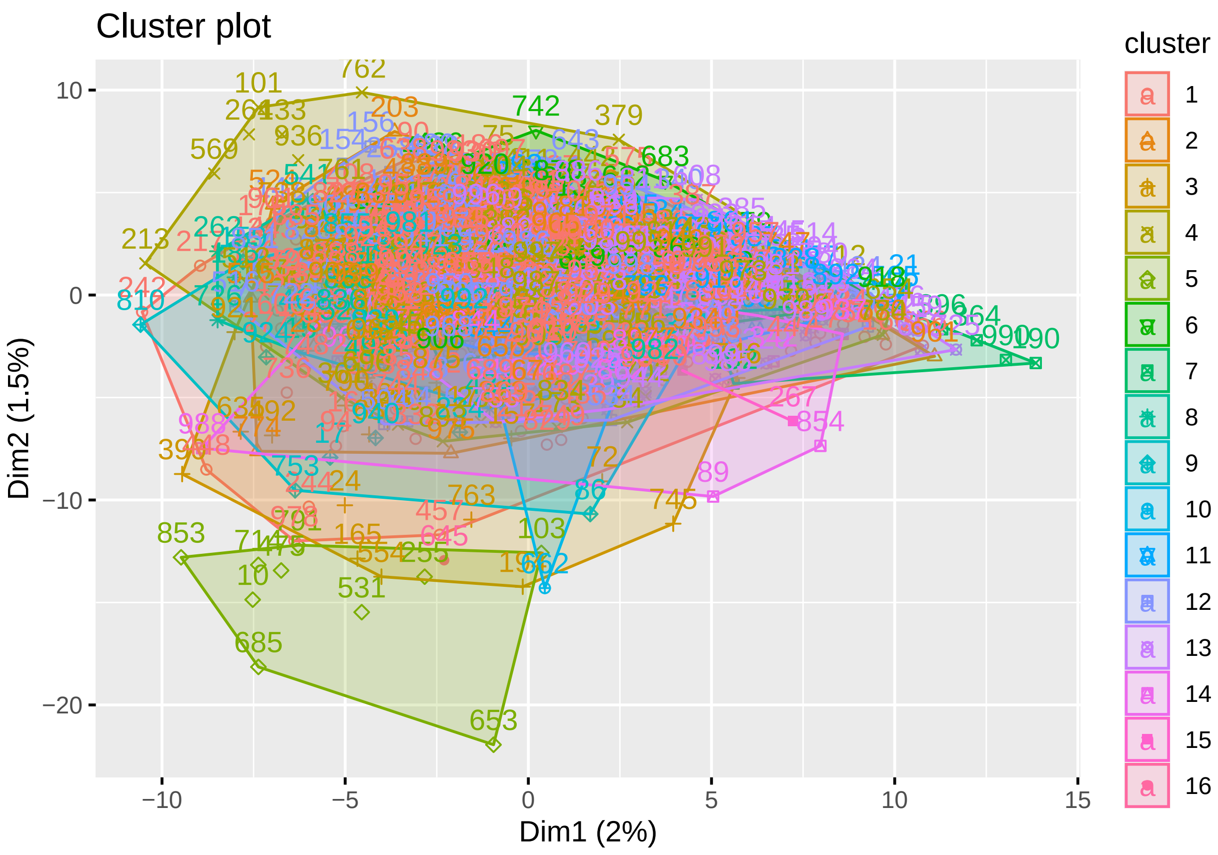
```
t2 <- round(t/rowSums(t)*100)
t2 <- cbind(t2,rowSums(t))
print ("Wyniki procentowe grupowania dla dtm")
```

```
## [1] "Wyniki procentowe grupowania dla dtm"
```

```
print (t2)
```

```
##       1  2  3  4  5 6 7  8  9 10 11 12 13 14 15 16
## ENFJ 27 20  0 20  0 3 0  3  0  0  0 13 10  3  0  0  30
## ENFP 21 12 10 16  1 6 0  4  0  4  2 11  9  2  0  0  80
## ENTJ 33 12  2  9  0 0 0  0  0  0  2 30 12  0  0  0  43
## ENTP 19 10 10 33  1 4 0  0  1  0  0  4 16  1  1  1  83
## ESFJ 17 17 17  0 17 0 0  0  0  0  0 33  0  0  0  0   6
## ESFP 40 20  0 40  0 0 0  0  0  0  0  0  0  0  0  0   5
## ESTJ  0 17  0 50  0 0 0  0  0  0  0 17  0 17  0  0   6
## ESTP 15  8  8 31  0 8 0  0  0  0  0  8 15  8  0  0  13
## INFJ 58  5  6  7  0 3 1  3  1  0  3  4  8  1  0  0 159
## INFP 26 18  1  9  2 4 2  1  0  0  6 12 18  2  0  0 200
## INTJ 29  5  2 17  1 1 1 15  2  0  1 13 13  2  0  0 117
## INTP 23  2  2 21  2 2 0  0 11  0  2 11 21  1  0  0 135
## ISFJ  7 23 13  7  0 7 3  0  3  0  3 27  7  0  0  0  30
## ISFP  7  7  0 17  0 0 0  0  3  0  3 31 21  7  3  0  29
## ISTJ 29  5  5 38  0 0 0  0  0  0  0 14 10  0  0  0  21
## ISTP 42  7  2  7  0 5 5  0  2  0  7  9 12  2  0  0  43
```

```
fviz_cluster(pam.dtm)
```



K-medoids TFIDF

```
pam_results_tfidf <- data %>%
mutate(cluster = pam.tfidf$clustering) %>%
group_by(cluster)

pam_results <- subset(pam_results_tfidf, select = c(type, cluster))
t <-  table(pam_results)
print ("Wyniki ilosciowe grupowania dla tf-idf")
```

```
## [1] "Wyniki ilosciowe grupowania dla tf-idf"
```

```
print (t)
```

```
##       cluster
## type    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
##   ENFJ  6  5  1  6  0  0  0  4  0  5  1  2  0  0  0  0
##   ENFP 18  5 22 12  0  0  0 10  0  8  3  2  0  0  0  0
##   ENTJ  7  2  9  7  0  1  0 10  0  7  0  0  0  0  0  0
##   ENTP 18  4  7 23  1  0  1 11  0 13  2  2  0  1  0  0
##   ESFJ  0  1  1  3  0  0  0  1  0  0  0  0  0  0  0  0
##   ESFP  1  0  2  0  0  0  0  0  0  2  0  0  0  0  0  0
##   ESTJ  1  0  0  3  0  0  0  0  0  1  1  0  0  0  0  0
##   ESTP  3  0  2  0  0  0  0  5  0  0  1  2  0  0  0  0
##   INFJ 24 15  9 33  0  1  6  7  1 55  1  6  0  0  1  0
##   INFP 42  9 16 41  0  2 18 13  0 49  4  5  0  0  1  0
##   INTJ 17  3 11 19  0  2  2 41  0 20  1  1  0  0  0  0
##   INTP 24  7 10 24  0  0  7 24  1 34  2  2  0  0  0  0
##   ISFJ  2 12  2  2  0  0  1  4  0  4  0  1  1  0  0  1
##   ISFP  4  4  6  4  0  0  2  3  0  4  0  1  0  1  0  0
##   ISTJ 14  1  3  1  0  0  0  2  0  0  0  0  0  0  0  0
##   ISTP 11  0  3  7  0  0  5  3  0 10  1  2  1  0  0  0
```

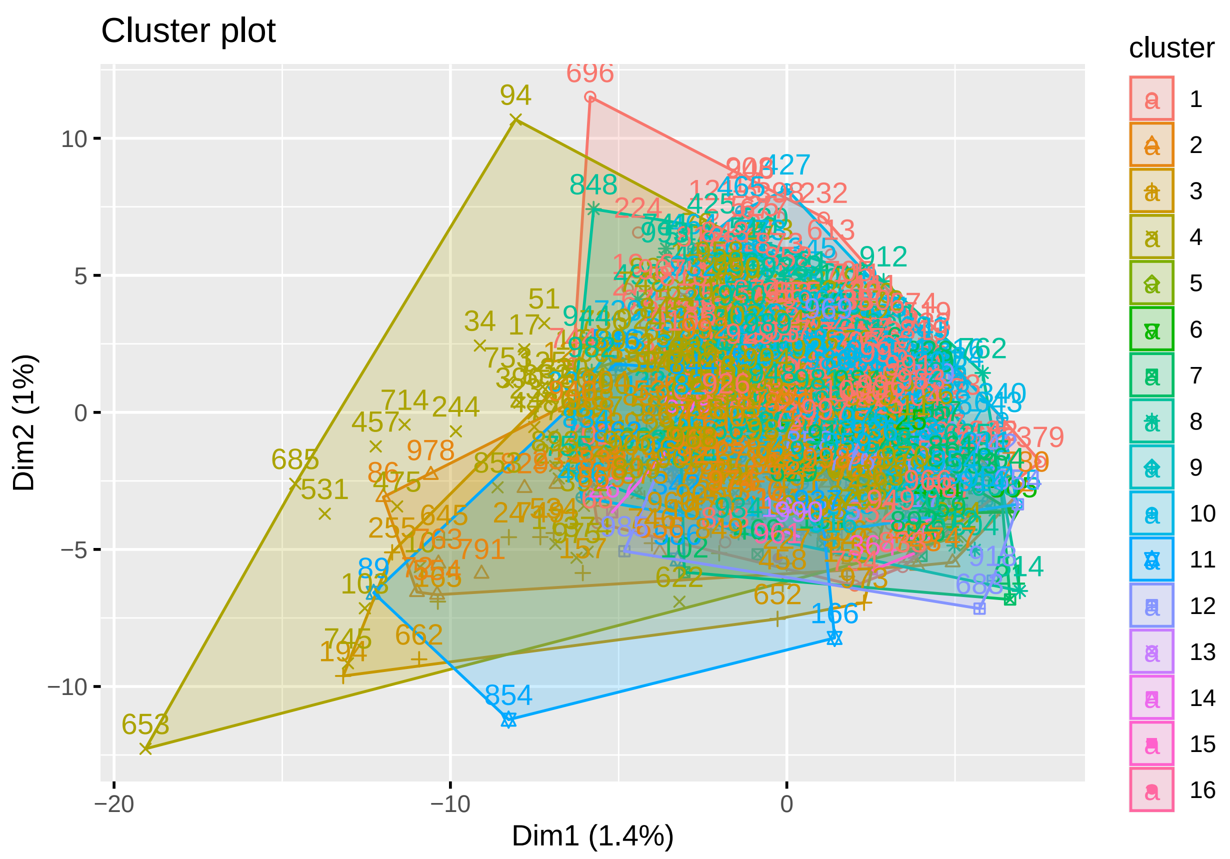
```
t2 <- round(t/rowSums(t)*100)
t2 <- cbind(t2,rowSums(t))
print ("Wyniki procentowe grupowania dla tf-idf")
```

```
## [1] "Wyniki procentowe grupowania dla tf-idf"
```

```
print (t2)
```

```
##       1  2  3  4 5 6  7  8 9 10 11 12 13 14 15 16    
## ENFJ 20 17  3 20 0 0  0 13 0 17  3  7  0  0  0  0  30
## ENFP 22  6 28 15 0 0  0 12 0 10  4  2  0  0  0  0  80
## ENTJ 16  5 21 16 0 2  0 23 0 16  0  0  0  0  0  0  43
## ENTP 22  5  8 28 1 0  1 13 0 16  2  2  0  1  0  0  83
## ESFJ  0 17 17 50 0 0  0 17 0  0  0  0  0  0  0  0   6
## ESFP 20  0 40  0 0 0  0  0 0 40  0  0  0  0  0  0   5
## ESTJ 17  0  0 50 0 0  0  0 0 17 17  0  0  0  0  0   6
## ESTP 23  0 15  0 0 0  0 38 0  0  8 15  0  0  0  0  13
## INFJ 15  9  6 21 0 1  4  4 1 35  1  4  0  0  1  0 159
## INFP 21  4  8 20 0 1  9  6 0 24  2  2  0  0  0  0 200
## INTJ 15  3  9 16 0 2  2 35 0 17  1  1  0  0  0  0 117
## INTP 18  5  7 18 0 0  5 18 1 25  1  1  0  0  0  0 135
## ISFJ  7 40  7  7 0 0  3 13 0 13  0  3  3  0  0  3  30
## ISFP 14 14 21 14 0 0  7 10 0 14  0  3  0  3  0  0  29
## ISTJ 67  5 14  5 0 0  0 10 0  0  0  0  0  0  0  0  21
## ISTP 26  0  7 16 0 0 12  7 0 23  2  5  2  0  0  0  43
```

```
fviz_cluster(pam.tfidf)
```



K-medoids Glove

```
pam_results_glove <- data %>%
mutate(cluster = pam.glove$clustering) %>%
group_by(cluster)

pam_results <- subset(pam_results_glove, select = c(type, cluster))
t <-  table(pam_results)
print ("Wyniki ilosciowe grupowania dla glove1")
```

```
## [1] "Wyniki ilosciowe grupowania dla glove1"
```

```
print (t)
```

```
##       cluster
## type    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
##   ENFJ  4  1  0  5  3  0  0  0  2  0  2  1  0  9  3  0
##   ENFP  8 12  4 22  5  3  1  0  1  3  3  2  0 10  6  0
##   ENTJ  9  3  1  8  4  1  0  1  3  0  3  4  0  4  2  0
##   ENTP 13  6  5 12  7  1  3  0  7  2  5  8  0  5  9  0
##   ESFJ  3  0  0  2  0  1  0  0  0  0  0  0  0  0  0  0
##   ESFP  0  3  1  0  0  0  0  0  1  0  0  0  0  0  0  0
##   ESTJ  0  1  0  1  3  0  0  0  0  1  0  0  0  0  0  0
##   ESTP  2  1  2  3  1  0  1  0  0  1  0  1  0  0  1  0
##   INFJ 26 17 18 24 17  0  6  2  8  0  6 11  1  7 16  0
##   INFP 17 11 29 31 14  3 27  6  7  2 12  9  0 14 18  0
##   INTJ 31  4 11 17 13  3  1  1  8  0  6  9  0  4  9  0
##   INTP 16  3 17 15 19  7  6  0  7  1  9 15  1  7 12  0
##   ISFJ  3  5  2  4  2  1  0  1  3  0  0  3  1  0  4  1
##   ISFP  3  5  2  4  0  0  0  0  1  1  1  5  0  2  5  0
##   ISTJ  3  4  1  1  4  1  0  0  0  0  2  1  0  2  2  0
##   ISTP  3  1  8  5  4  0  2  1  5  2  2  3  0  0  6  1
```

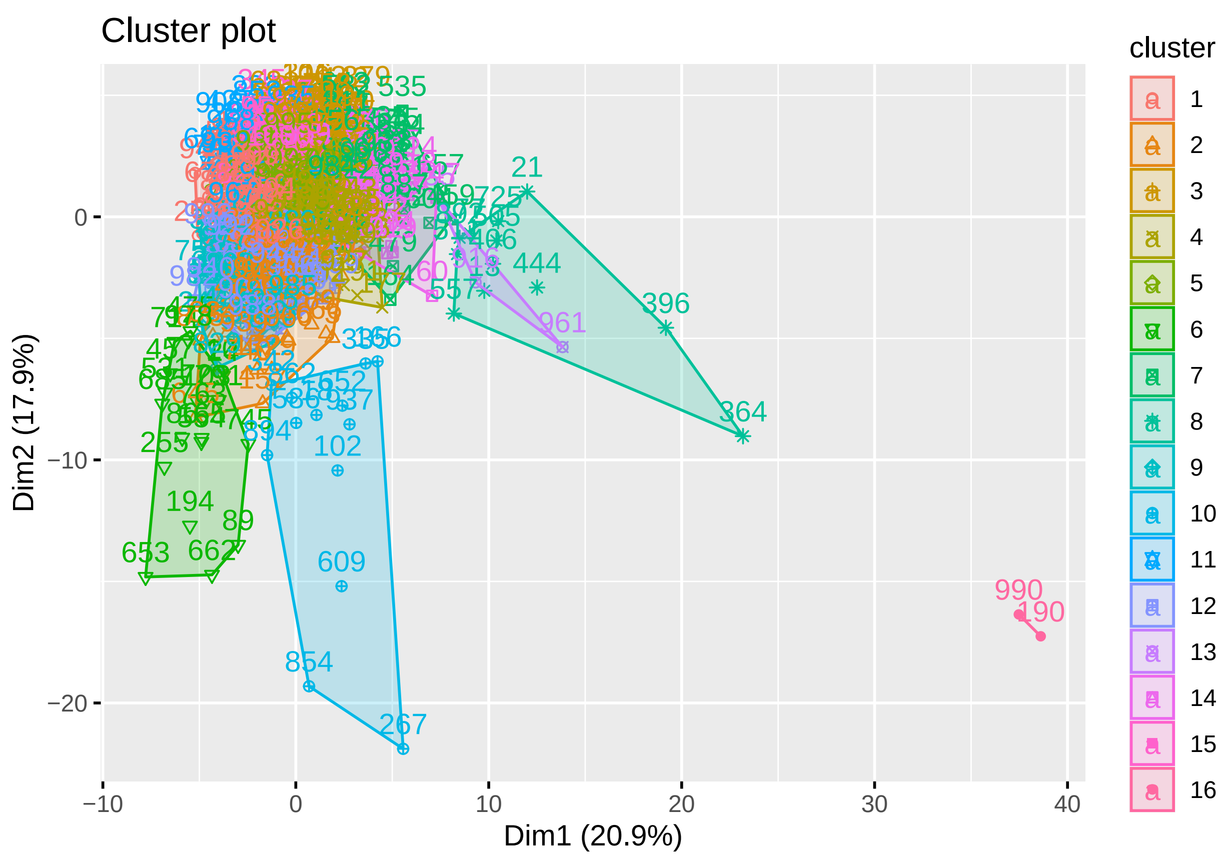
```
t2 <- round(t/rowSums(t)*100)
t2 <- cbind(t2,rowSums(t))
print ("Wyniki procentowe grupowania dla glove1")
```

```
## [1] "Wyniki procentowe grupowania dla glove1"
```

```
print (t2)
```

```
##       1  2  3  4  5  6  7 8  9 10 11 12 13 14 15 16
## ENFJ 13  3  0 17 10  0  0 0  7  0  7  3  0 30 10  0  30
## ENFP 10 15  5 28  6  4  1 0  1  4  4  2  0 12  8  0  80
## ENTJ 21  7  2 19  9  2  0 2  7  0  7  9  0  9  5  0  43
## ENTP 16  7  6 14  8  1  4 0  8  2  6 10  0  6 11  0  83
## ESFJ 50  0  0 33  0 17  0 0  0  0  0  0  0  0  0  0   6
## ESFP  0 60 20  0  0  0  0 0 20  0  0  0  0  0  0  0   5
## ESTJ  0 17  0 17 50  0  0 0  0 17  0  0  0  0  0  0   6
## ESTP 15  8 15 23  8  0  8 0  0  8  0  8  0  0  8  0  13
## INFJ 16 11 11 15 11  0  4 1  5  0  4  7  1  4 10  0 159
## INFP  8  6 14 16  7  2 14 3  4  1  6  4  0  7  9  0 200
## INTJ 26  3  9 15 11  3  1 1  7  0  5  8  0  3  8  0 117
## INTP 12  2 13 11 14  5  4 0  5  1  7 11  1  5  9  0 135
## ISFJ 10 17  7 13  7  3  0 3 10  0  0 10  3  0 13  3  30
## ISFP 10 17  7 14  0  0  0 0  3  3  3 17  0  7 17  0  29
## ISTJ 14 19  5  5 19  5  0 0  0  0 10  5  0 10 10  0  21
## ISTP  7  2 19 12  9  0  5 2 12  5  5  7  0  0 14  2  43
```

```
fviz_cluster(pam.glove)
```



K-medoids Glove(oddzielne posty)

```
pam_results_glove2 <- data2 %>%
mutate(cluster = pam.glove2$clustering) %>%
group_by(cluster)

pam_results <- subset(pam_results_glove2, select = c(type, cluster))
t <-  table(pam_results)
print ("Wyniki ilosciowe grupowania dla glove2")
```

```
## [1] "Wyniki ilosciowe grupowania dla glove2"
```

```
print (t)
```

```
##       cluster
## type     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
##   ENFJ   9  21  12   1   3   8   1   3   4   3   6  10   2   3   6   0
##   ENFP  12  28   3   2   0   8   7   6   5   2   2   8   4   1   1   1
##   ENTJ   8  11   2   0   1   1   2   2   3   2   0   5   3   2   1   0
##   ENTP  17  18   3   4   3   2   5   4   3   2   1   5   4   1   2   1
##   ESFP  10  11   5   1   0   2   3   2   0   3   2   2   3   0   0   0
##   ESTJ   7  18   4   0   0   3   3   4   0   1   1   4   2   2   0   0
##   INFJ  65 144  37  21  16  40  44  36  25  27  11  38  41  30  24   0
##   INFP  33 104  22  24  17  29  26  34  23  19  13  33  28   8  16   1
##   INTJ  34 121  15  22  12  19  21  23  18  10  13  33  13  12  15   3
##   INTP   8  25   5   5   2   9   5   5   2   2   1   8   7   3   4   0
##    ISFP   2   7   2   1   3   2   2   5   1   2   5   4   5   0   1   0
##    ISTJ   1   9   0   1   0   1   1   1   0   0   0   0   0   1   0   0
##    ISTP  12  40   7   6   3  13   7   9   3   3   1  10  11   3   4   0
```

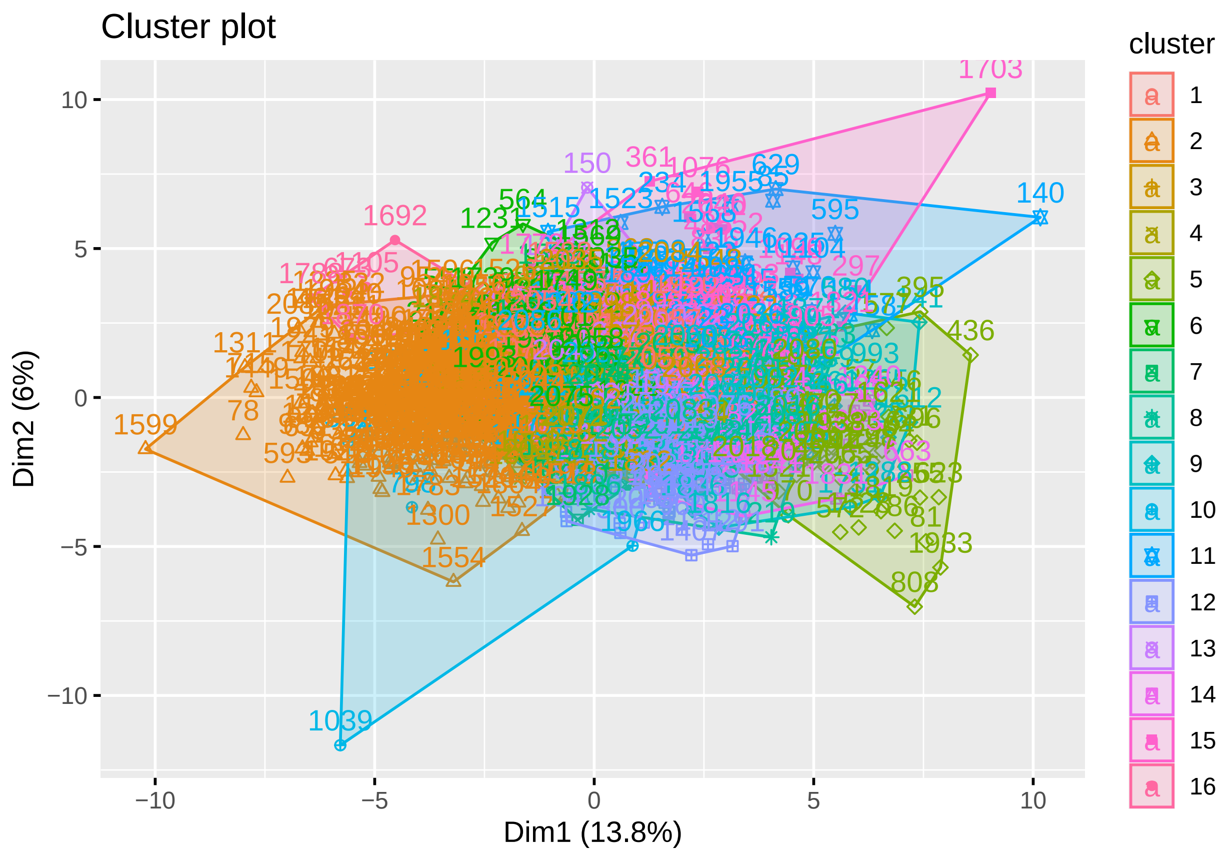
```
t2 <- round(t/rowSums(t)*100)
t2 <- cbind(t2,rowSums(t))
print ("Wyniki procentowe grupowania dla glove2")
```

```
## [1] "Wyniki procentowe grupowania dla glove2"
```

```
print (t2)
```

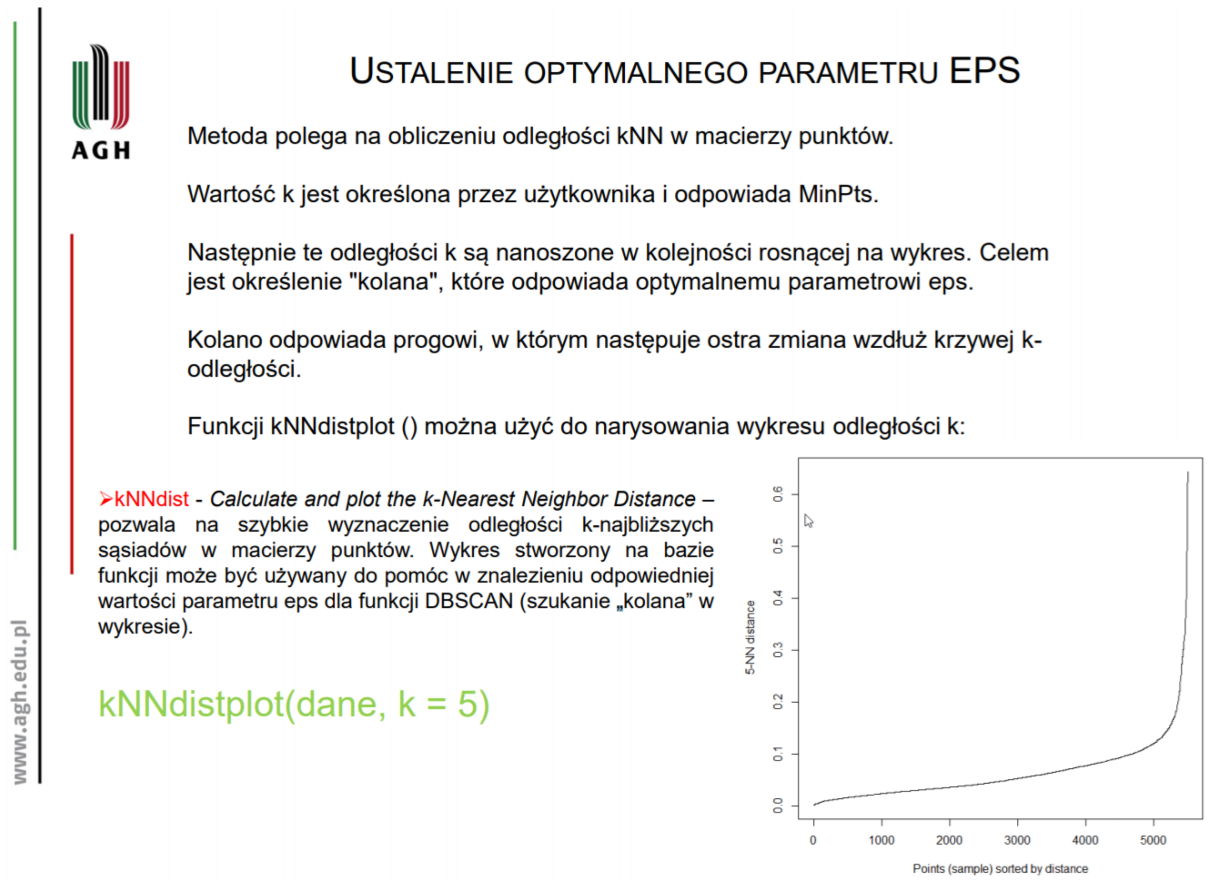
```
##       1  2  3 4 5  6 7  8 9 10 11 12 13 14 15 16
## ENFJ 10 23 13 1 3  9 1  3 4  3  7 11  2  3  7  0  92
## ENFP 13 31  3 2 0  9 8  7 6  2  2  9  4  1  1  1  90
## ENTJ 19 26  5 0 2  2 5  5 7  5  0 12  7  5  2  0  43
## ENTP 23 24  4 5 4  3 7  5 4  3  1  7  5  1  3  1  75
## ESFP 23 25 11 2 0  5 7  5 0  7  5  5  7  0  0  0  44
## ESTJ 14 37  8 0 0  6 6  8 0  2  2  8  4  4  0  0  49
## INFJ 11 24  6 4 3  7 7  6 4  5  2  6  7  5  4  0 599
## INFP  8 24  5 6 4  7 6  8 5  4  3  8  7  2  4  0 430
## INTJ  9 32  4 6 3  5 5  6 5  3  3  9  3  3  4  1 384
## INTP  9 27  5 5 2 10 5  5 2  2  1  9  8  3  4  0  91
## ISFP  5 17  5 2 7  5 5 12 2  5 12 10 12  0  2  0  42
## ISTJ  7 60  0 7 0  7 7  7 0  0  0  0  0  7  0  0  15
## ISTP  9 30  5 5 2 10 5  7 2  2  1  8  8  2  3  0 132
```

```
fviz_cluster(pam.glove2)
```



DBSCAN

```r
# metric = "euclidean", "manhattan", "gower"
dissimilarity.dtm <- daisy(as.matrix(vectors.dtm), metric = "euclidean")
dissimilarity.tfidf <- daisy(as.matrix(vectors.tfidf), metric = "euclidean")
dissimilarity.glove <- daisy(as.matrix(vectors.glove), metric = "euclidean")
dissimilarity.glove_post <- daisy(as.matrix(vectors.glove_post), metric = "euclidean")
```



Ustalenie optymalnego parametru EPS

Metoda polega na obliczeniu odległości kNN w macierzy punktów.

Wartość k jest określona przez użytkownika i odpowiada MinPts.

Następnie te odległości k są nanoszone w kolejności rosnącej na wykres. Celem jest określenie "kolana", które odpowiada optymalnemu parametrowi eps.

Kolano odpowiada progowi, w którym następuje ostra zmiana wzdłuż krzywej k-odległości.

Funkcji kNNdistplot () można użyć do narysowania wykresu odległości k:

➢kNNdist - *Calculate and plot the k-Nearest Neighbor Distance* – pozwala na szybkie wyznaczenie odległości k-najbliższych sąsiadów w macierzy punktów. Wykres stworzony na bazie funkcji może być używany do pomóc w znalezieniu odpowiedniej wartości parametru eps dla funkcji DBSCAN (szukanie „kolana” w wykresie).

kNNdistplot(dane, k = 5)

www.agh.edu.pl

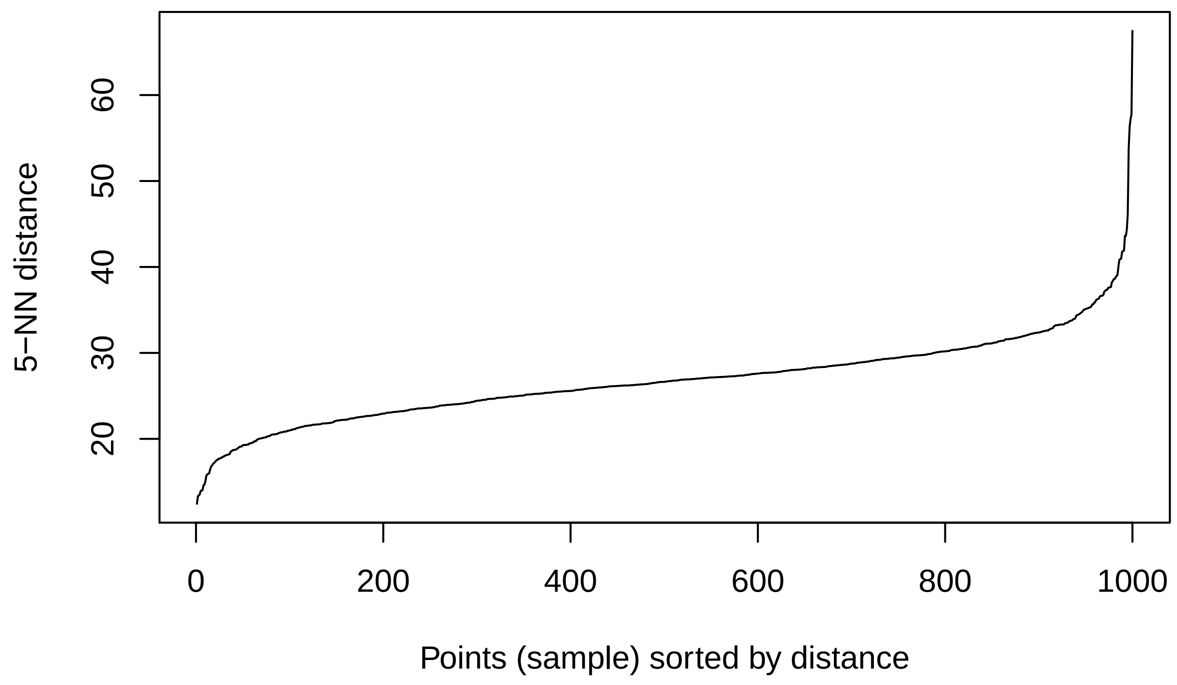
Figure 1: ESP

```r
plot_desity <- function(desity, dissimilarity, method) {
  points <- cmdscale(dissimilarity, k = 2)
  plot(points,
     main = method,
     col = as.factor(desity$cluster),
     mai = c(0, 0, 0, 0),
     mar = c(0, 0, 0, 0),
     xaxt = 'n', yaxt = 'n',
     xlab = '', ylab = '')
}
```

DTM

Należy odczytać gdzie na wykresie jest tak zwanne kolano, czyli punkt, po którym wykres zaczyna ustawiać się w pionie. Wartość tego punktu zostaje zastosowana jako eps.

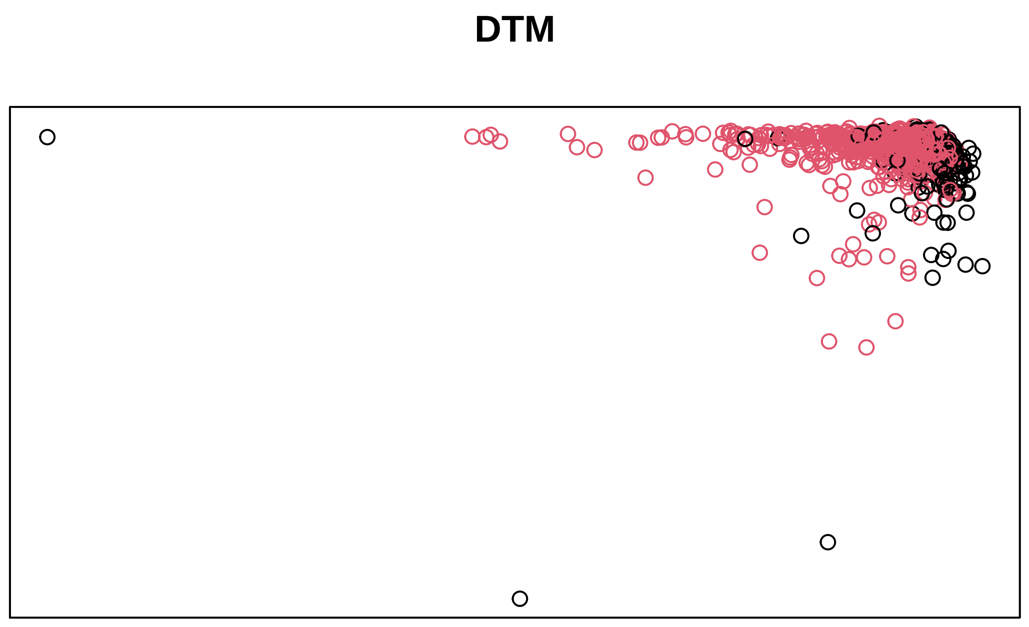
```
kNNdistplot(vectors.dtm, k = 5)
```



```
eps = 30
dbscan.dtm <- dbscan(vectors.dtm, eps = eps, minPts = 5)
dbscan.dtm
```

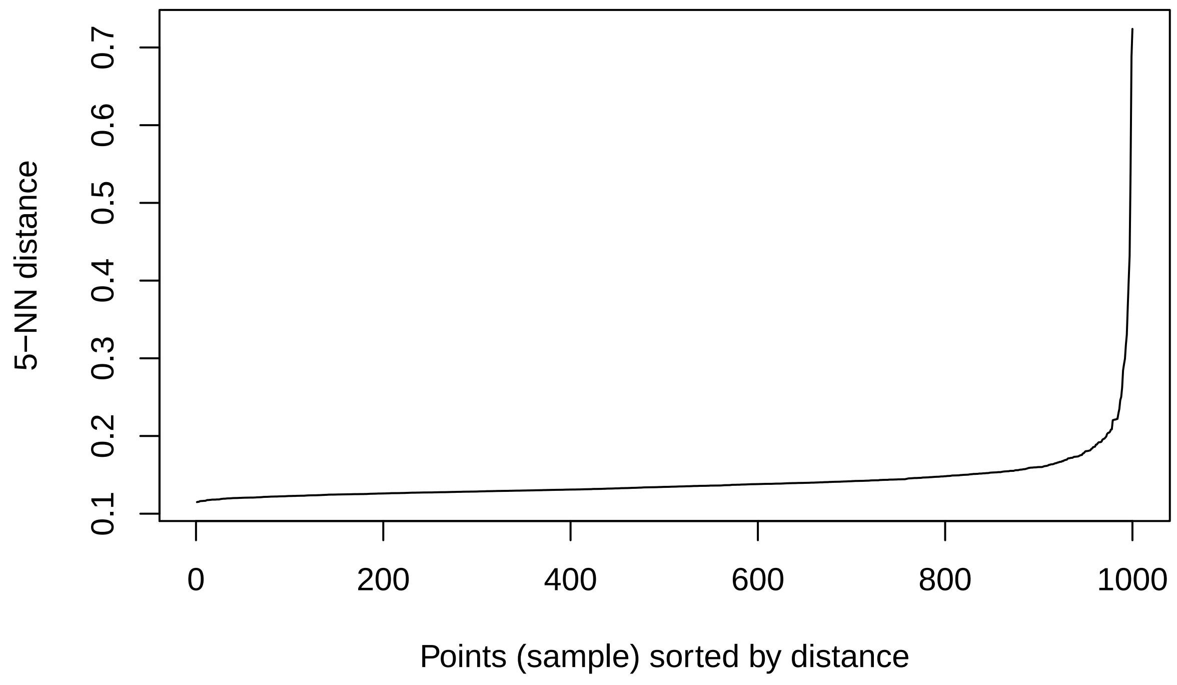
```
## DBSCAN clustering for 1000 objects.
## Parameters: eps = 30, minPts = 5
## The clustering contains 1 cluster(s) and 160 noise points.
##
##   0   1
## 160 840
##
## Available fields: cluster, eps, minPts
```

```
plot_desity(dbscan.dtm, dissimilarity.dtm, "DTM")
```



TFID

```
kNNdistplot(vectors.tfidf, k = 5)
```

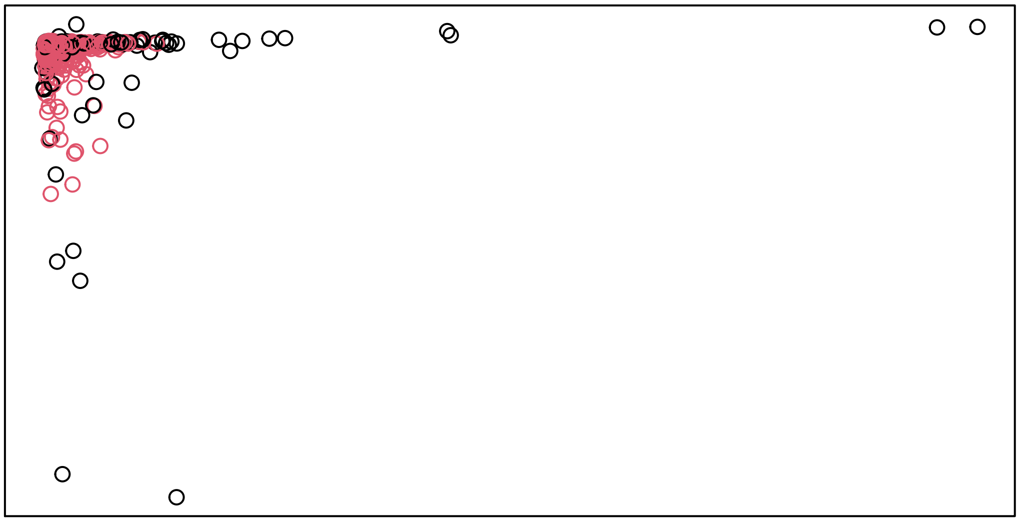


```
eps = 0.15
dbscan.tfidf <- dbscan(vectors.tfidf, eps = eps, minPts = 5)
dbscan.tfidf
```

```
## DBSCAN clustering for 1000 objects.
## Parameters: eps = 0.15, minPts = 5
## The clustering contains 1 cluster(s) and 133 noise points.
##
##   0   1
## 133 867
##
## Available fields: cluster, eps, minPts
```

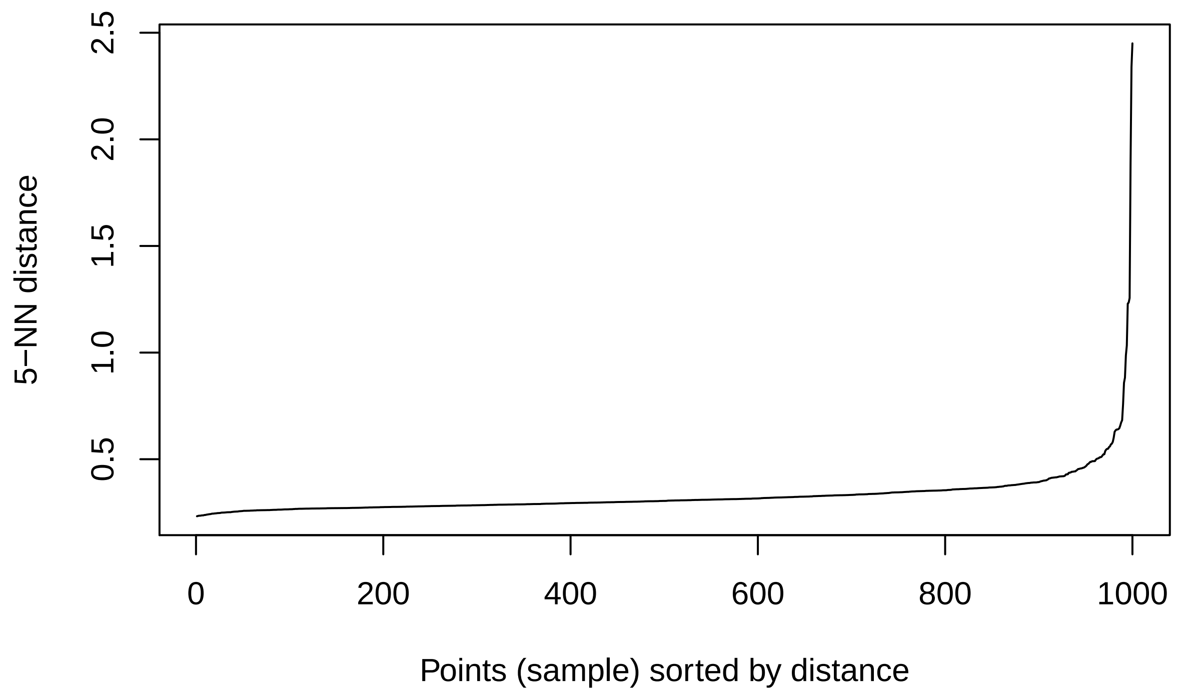
```
plot_desity(dbscan.tfidf, dissimilarity.tfidf, "TFIDF")
```

TFIDF



GLOVE

```
kNNdistplot(vectors.glove, k = 5)
```

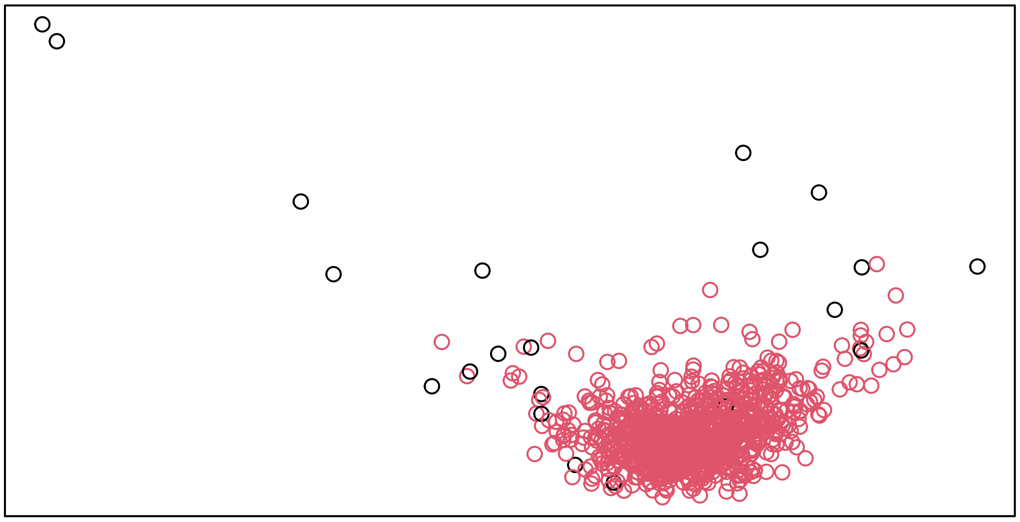


```
eps = 0.5
minPts = 5
dbscan.glove <- dbscan(vectors.glove, eps = eps, minPts = minPts)
dbscan.glove
```

```
## DBSCAN clustering for 1000 objects.
## Parameters: eps = 0.5, minPts = 5
## The clustering contains 1 cluster(s) and 21 noise points.
##
##    0   1
##   21 979
##
## Available fields: cluster, eps, minPts
```

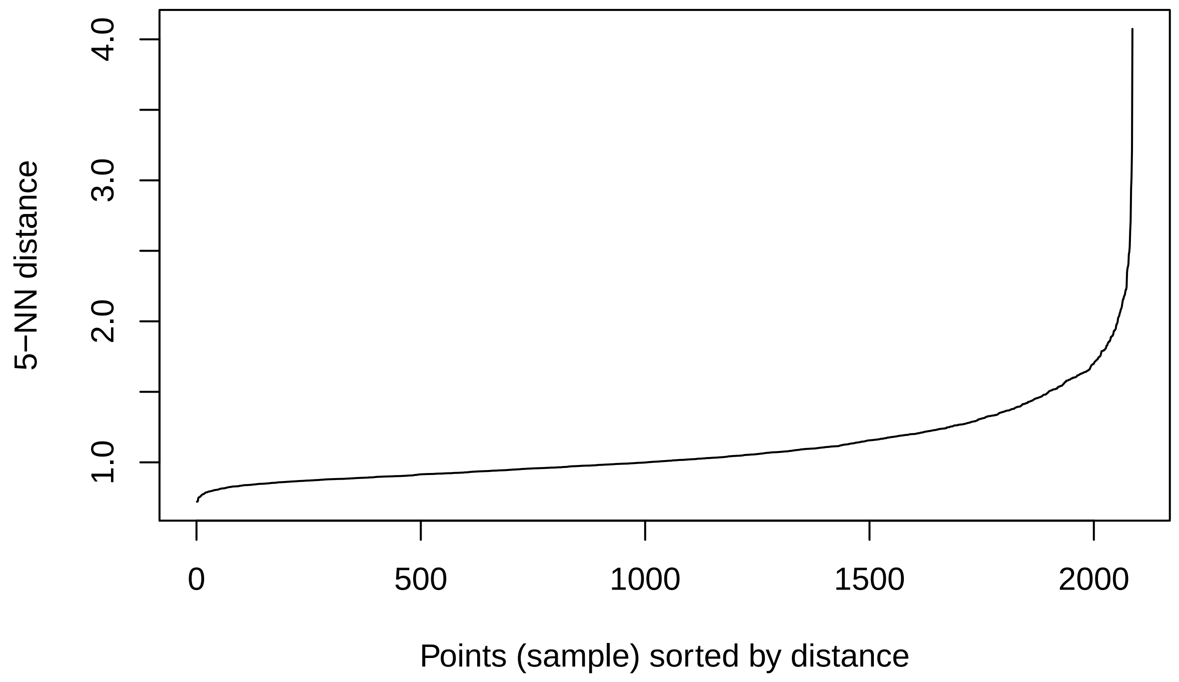
```
plot_desity(dbscan.glove, dissimilarity.glove, "GLOVE")
```

GLOVE



GLOVE POST

```
kNNdistplot(vectors.glove_post, k = 5)
```

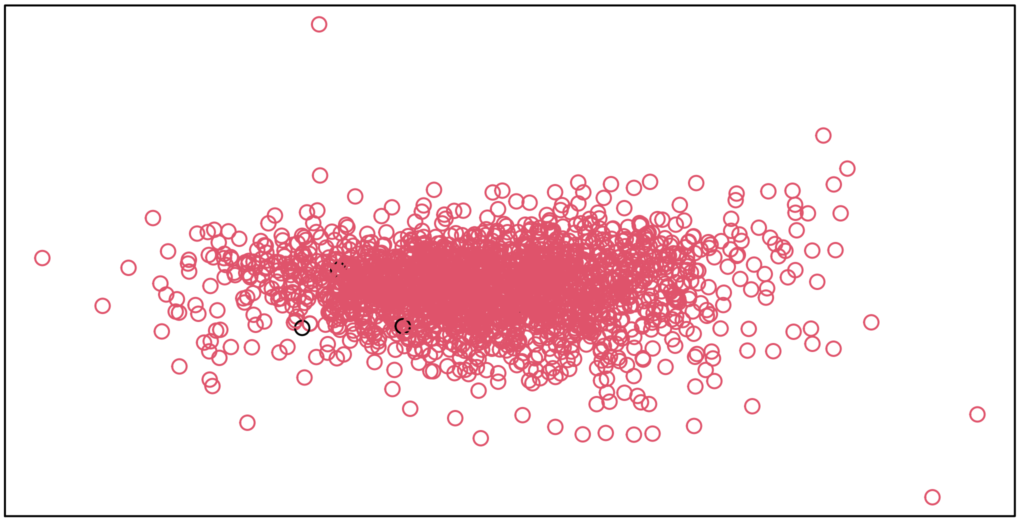


```
eps = 1.5
minPts = 5
dbscan.glove_post <- dbscan(vectors.glove, eps = eps, minPts = minPts)
dbscan.glove_post
```

```
## DBSCAN clustering for 1000 objects.
## Parameters: eps = 1.5, minPts = 5
## The clustering contains 1 cluster(s) and 2 noise points.
##
##   0   1
##   2 998
##
## Available fields: cluster, eps, minPts
```

```
plot_desity(dbscan.glove_post, dissimilarity.glove_post, "GLOVE POST")
```

GLOVE POST

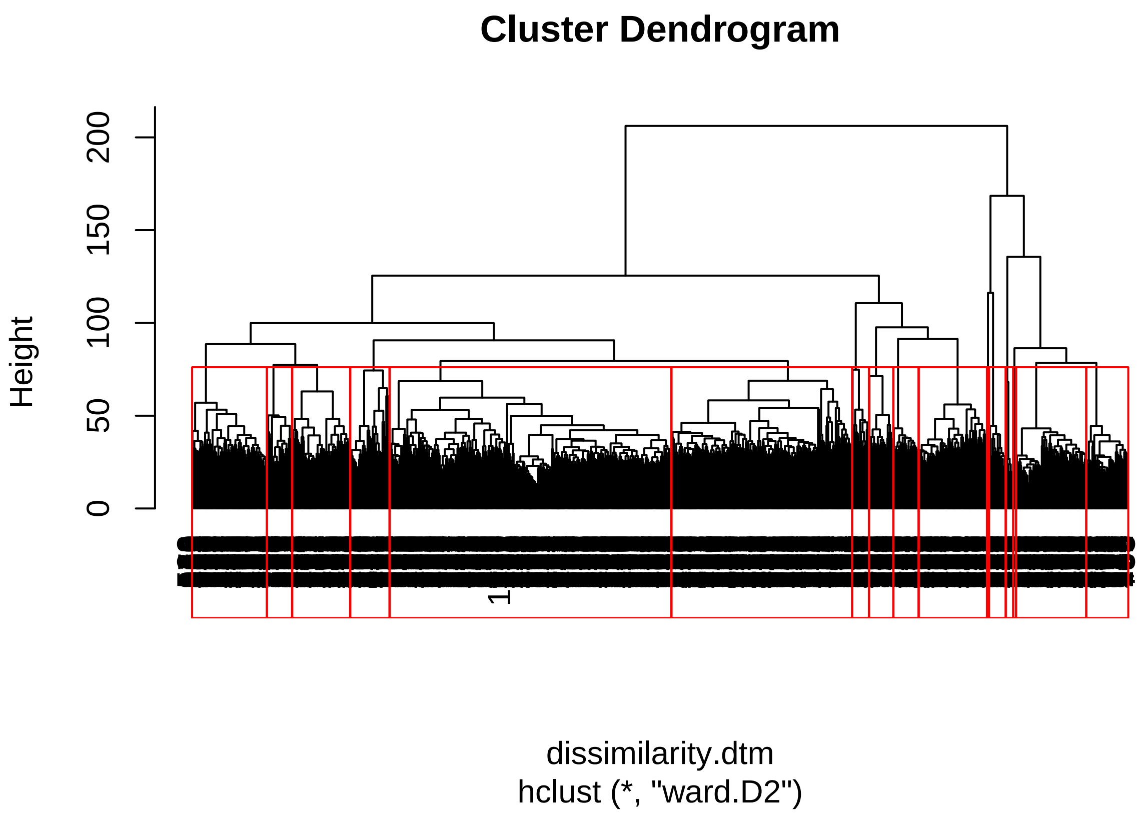


Grupowanie Hierarchiczne

```r
plot_hirarchical <- function(dissimilarity, hirarchical_result, method) {
  points <- cmdscale(dissimilarity, k = 2)
  res.hier <- cutree(hirarchical_result, k = 16)
  plot(points,
       main = method,
       col = res.hier,
       mai = c(0, 0, 0, 0),
       mar = c(0, 0, 0, 0),
       xaxt = 'n', yaxt = 'n',
       xlab = '', ylab = '')
}
```

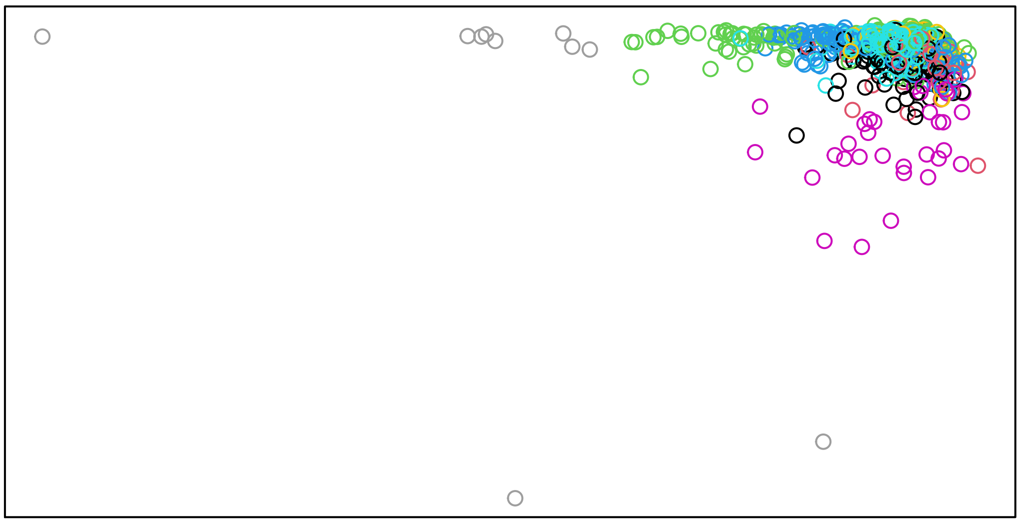
DTM

```r
# methods = "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", "centroid"
hc.dtm <- hclust(dissimilarity.dtm, method="ward.D2")
plot(hc.dtm, hang = -1)
rect.hclust(hc.dtm, k=16, border="red")
```



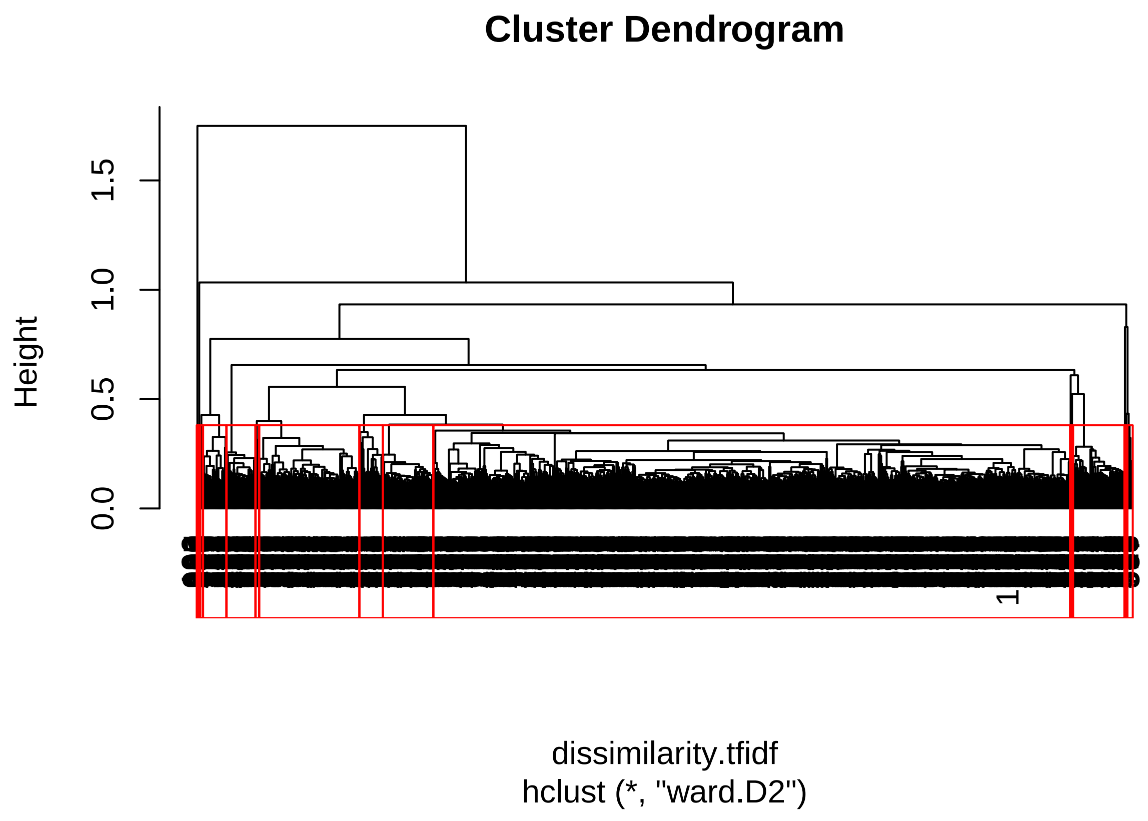
```
plot_hirarchical(dissimilarity.dtm, hc.dtm, "DTM")
```

DTM



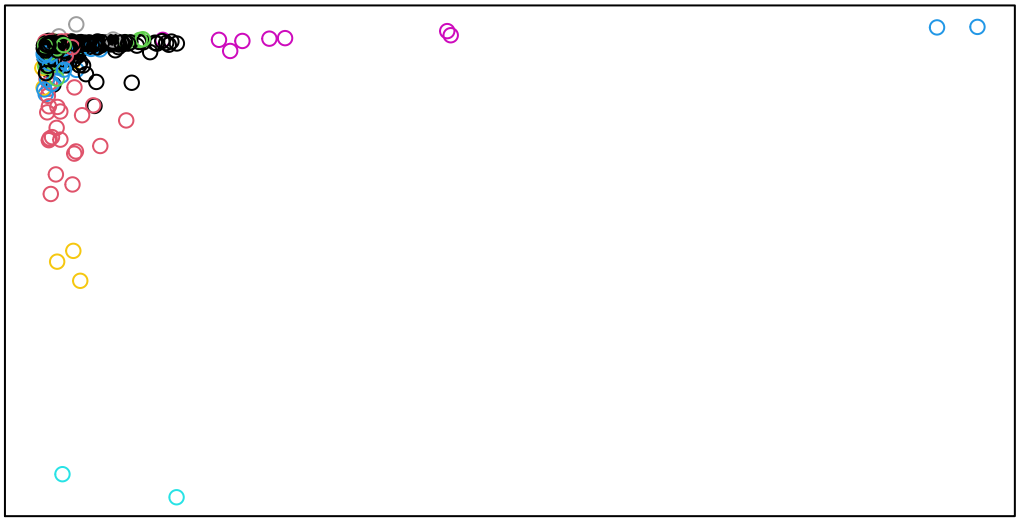
TfIDF

```
hc.tfidf <- hclust(dissimilarity.tfidf, method="ward.D2")
plot(hc.tfidf, hang = -1)
rect.hclust(hc.tfidf, k=16, border="red")
```



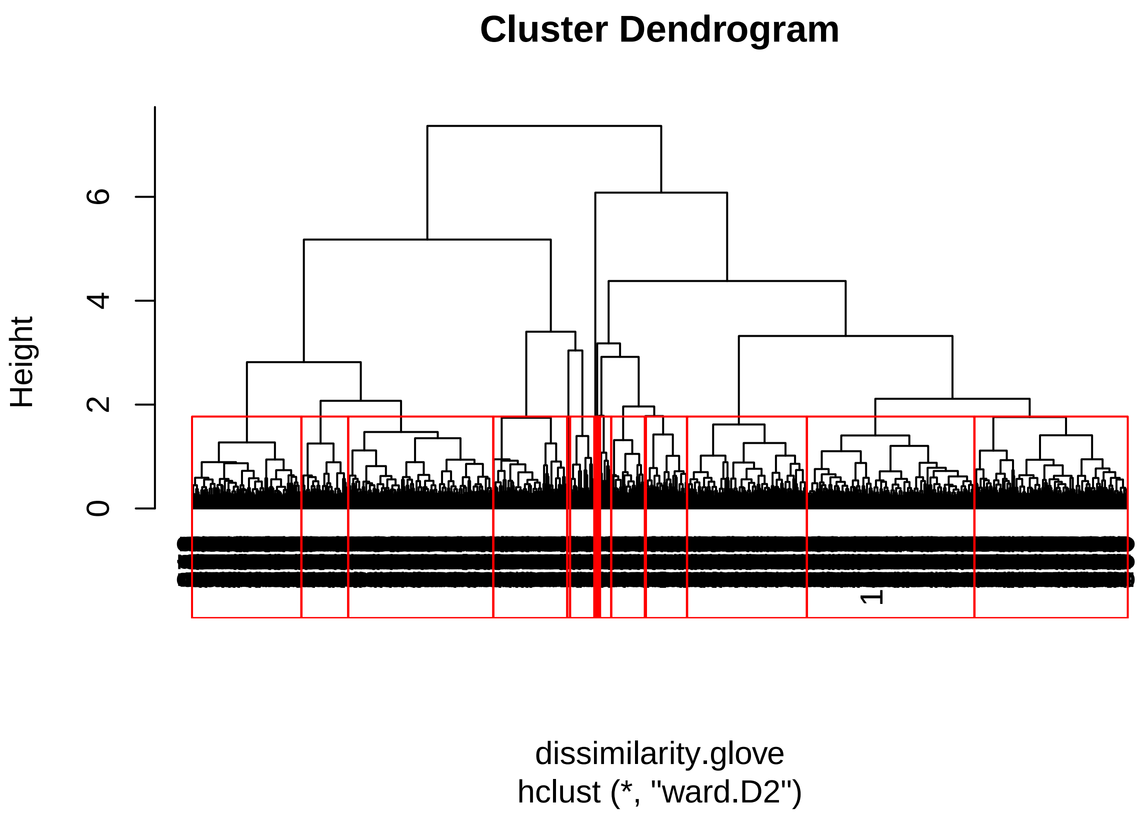
```
plot_hirarchical(dissimilarity.tfidf, hc.tfidf, "TFIDF")
```

TFIDF



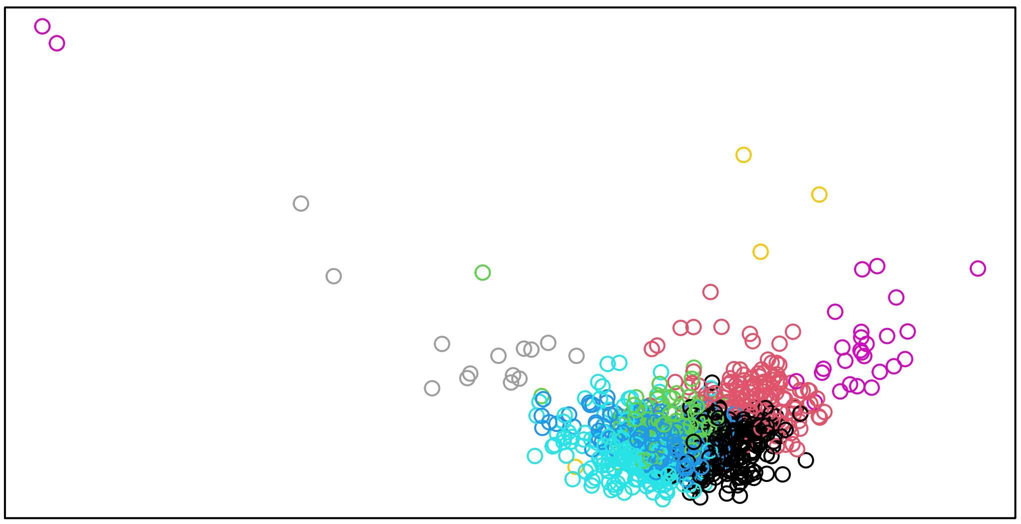
GLOVE

```
hc.glove <- hclust(dissimilarity.glove, method="ward.D2")
plot(hc.glove, hang = -1)
rect.hclust(hc.glove, k=16, border="red")
```



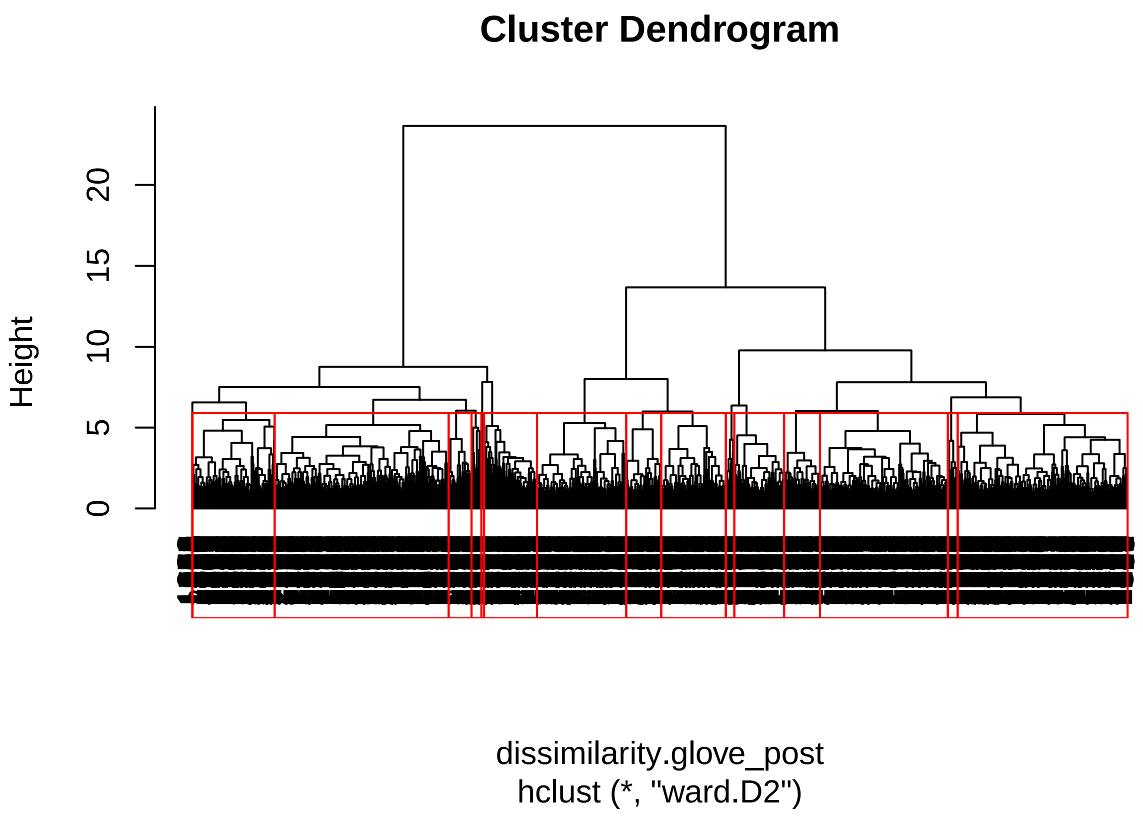
```r
plot_hirarchical(dissimilarity.glove, hc.glove, "GLOVE")
```

GLOVE



GLOVE Posts

```
hc.glove_post <- hclust(dissimilarity.glove_post, method="ward.D2")
plot(hc.glove_post, hang = -1)
rect.hclust(hc.glove_post, k=16, border="red")
```



```
plot_hirarchical(dissimilarity.glove_post, hc.glove_post, "GLOVE POSTS")
```

GLOVE POSTS