Sentimental Analysis of Customer Review

Package Vignette - Rationale

Text clustering is one of the most used tool to organize the documents to a well structured format or to find out a meaningful information from the text document. However, the text documents would consume irrelevant, redundant and noisy data which creates barrier in providing an insight from the data. Similarly, many customer provides review on different products in the company's website, however, the company may not be able to recognize the sentiment behind the review. To mitigate and address the problem, we created a EmoReader package which contains the get_emotion function that provides 10 different sentiments from the document and scores them accordingly. Furthermore, we perform nmf_func and pca_func to find out which word tends to co-occur. The package would also include the cluster_kmeans and h_cluster to cluster different emotions and to find out which clustering provides accurate information.

Package Vignette - Significance/Innovation

The EmoReader package will help in the efficiency of the data synthesis to generate the emotions from the text and will help in discovering the pattern of how the sentiments are clustered together.

Aim 1 and Aim 2

The package addresses on generating the 10 different emotions which are then followed by performing two different dimension reduction function which is nmf_func and pca_func that helps to remove the redundant and noisy data. The sentiments that are generated from the get_emotion are then clustered using the cluster_kmeans and h_cluster. The aim of the package is to be able to get the emotions from the text and to be able to compare between two different clusters to find out which clustering method provides accurate information on the sentiments are clustered together.

Loading the EmoReader package

```
library(EmoReader)
```

Demo of how to use the functions

Loading the dataset

We will be analyzing the dataset of an Amazon's customer review made on Electronic Products and the function below loads and pre-process the data. The function separates and breaks the word such as I'm to I am making it a different words.

it requires para as dataframe or list typenot csv file type

```
df <- read_inbuilt_data(amazon_data)
str(df)
```

```
## 'data.frame':    500 obs. of  2 variables:
##  $ star_rating: int  5 5 5 4 2 3 3 5 4 5 ...
##  $ review     : chr  "As advertised. Everything works perfectly, I'm very happy with the camera. As 
```

The get_emotion function converts the review which are in the string format to 10 different emotions. Firstly, the function changes the capitalized words to a lower case and then the 10 different emotions are created and it scores each review to different emotions accordingly.The user need to pass a dataframe in the parameter. Here, we created a data frame named df of the data.

```
emo_mat <- get_emotion(df)
```

```
## Warning: `spread_()` was deprecated in tidyr 1.2.0.
## i Please use `spread()` instead.
## i The deprecated feature was likely used in the syuzhet package.
##   Please report the issue to the authors.
```

```
head(emo_mat)
```

```
##   anger anticipation disgust fear joy sadness surprise trust negative positive
## 1     0            1       0    0   1       0        0     2        0        1
## 2     0            0       0    0   0       0        0     0        0        0
## 3     0            0       0    1   0       1        0     1        1        0
## 4     0            0       0    1   0       0        0     0        1        0
## 5     1            1       0    0   1       0        0     1        1        3
## 6     0            1       0    0   1       0        1     1        0        3
```

The count_emotions function provides the frequency of each of the emotions with the frequency table. The function as for the parameter to be passed and we have passed our emo_mat parameter which consists of the 10 different emotion that has been generated from each reviews. we need to pass list or dataframe as parameter.

```
count_emotions(emo_mat)
```

```
##              count      emotion
## anger          164        anger
## anticipation   389 anticipation
## disgust         90      disgust
## fear           152         fear
## joy            348          joy
## sadness        194      sadness
## surprise       216     surprise
## trust          401        trust
## negative       350     negative
## positive       715     positive
```

The matrix_conversion function helps to convert the emotion table values into a matrix table which are required for performing dimension reduction and clustering of the generated emotions.

```
my_mat <-matrix_conversion(emo_mat)
head(my_mat)
```

```
##      anger anticipation disgust fear joy sadness surprise trust negative
## [1,]     0            1       0    0   1       0        0     2        0
## [2,]     0            0       0    0   0       0        0     0        0
## [3,]     0            0       0    1   0       1        0     1        1
## [4,]     0            0       0    1   0       0        0     0        1
## [5,]     1            1       0    0   1       0        0     1        1
## [6,]     0            1       0    0   1       0        1     1        0
##      positive
## [1,]        1
## [2,]        0
## [3,]        0
## [4,]        0
## [5,]        3
## [6,]        3
```

After converting to a matrix_conversion, the convert_to_sparse_matrix functions converts the basic matrix table into the sparsematrix format which consumes less memory than the dense basic matrix. This function would generate the dgc matrix which is the class of sparse matrix.

```
library(Matrix)
sparse_mat <- convert_to_sparse_matrix()
head(sparse_mat)
```

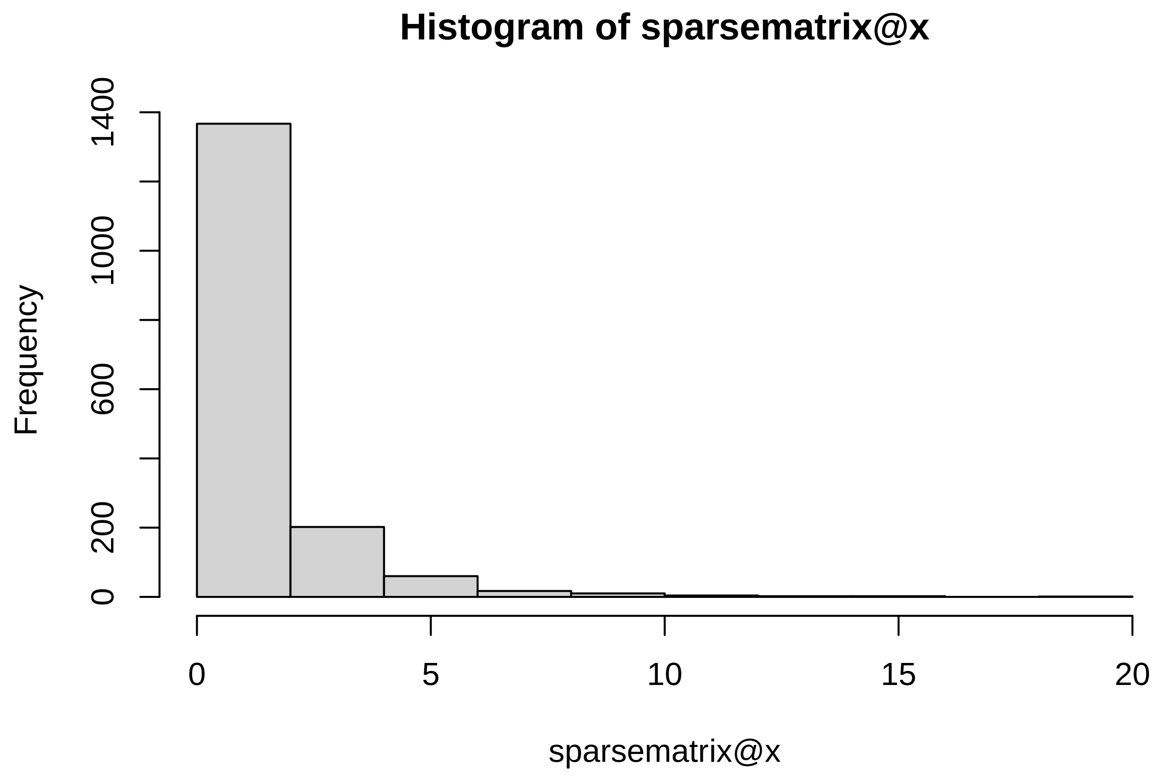
```
## 6 x 10 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 10 column names 'anger', 'anticipation', 'disgust' ... ]]

##
## [1,] . 1 . . 1 . . 2 . 1
## [2,] . . . . . . . . . .
## [3,] . . . 1 . 1 . 1 1 .
## [4,] . . . 1 . . . . 1 .
## [5,] 1 1 . . 1 . . 1 1 3
## [6,] . 1 . . 1 . 1 1 . 3
```

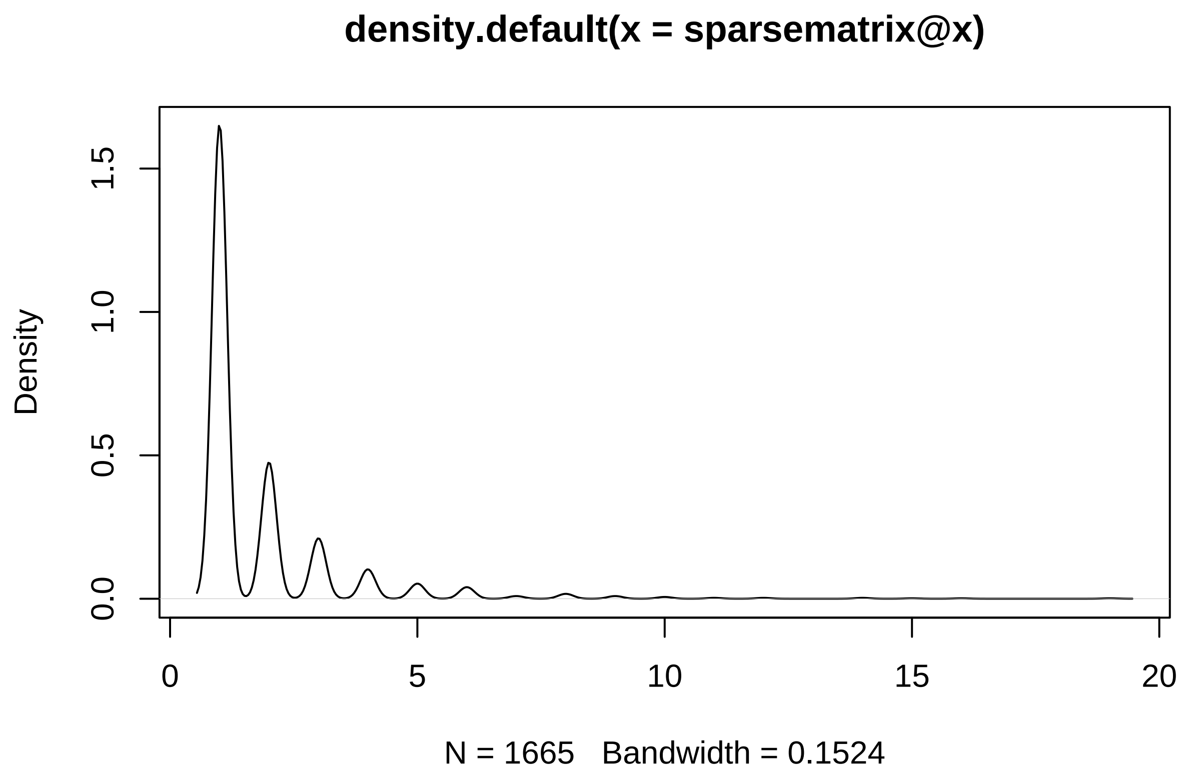
Aim 1: Classifying the emotions from the review performing NMF dimension reduction and then clustering the emotion using K-mean clustering.

The nmf_func takes the sparse matrix data which was converted from a basic matrix of the emotion. The nmf_func minimizes the dimension of the data using the Euclidean distance. Running the nmf_func provides the rank and need to pass the number of rank in the parameter.

```
library(singlet)
nmf_func(3)
```

```
##
## iter |      tol
## ---------------
##    1 | 9.24e-01
##    2 | 9.96e-02
##    3 | 1.71e-02
##    4 | 5.37e-03
##    5 | 2.95e-03
##    6 | 1.85e-03
##    7 | 1.06e-03
##    8 | 6.79e-04
##    9 | 4.95e-04
##   10 | 3.43e-04
##   11 | 2.46e-04
##   12 | 1.78e-04
##   13 | 1.32e-04
##   14 | 8.49e-05
```





```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x119033120>
## <environment: namespace:base>
```

The norm_sparse_matrix function normalizes the data. For this function, the Seurat package has been used and by default the LogNormalize function is used from the Seurat to normalize the feature expressions.

```
norm <- norm_sparse_matrix()

head(norm)
```

```
## 6 x 10 sparse Matrix of class "dgCMatrix"
```

```
##    [[ suppressing 10 column names 'anger', 'anticipation', 'disgust' ... ]]
```

```
##
## [1,] .        3.284923 . .        3.392346 .        .        3.929378 .
## [2,] .        .        . .        .        .        .        .        .
## [3,] .        .        . 4.201545 .        3.961696 .        3.255696 3.386809
## [4,] .        .        . 4.201545 .        .        .        .        3.386809
## [5,] 4.126741 3.284923 . .        3.392346 .        .        3.255696 3.386809
## [6,] .        3.284923 . .        3.392346 .        3.856432 3.255696 .
##
```

```
## [1,] 2.707117
## [2,] .
## [3,] .
## [4,] .
## [5,] 3.760224
## [6,] 3.760224
```

The modeling_nmf function takes the parameter from the user on the rank that has been generated after running the nmf_func. The nmf rank for our dataset is 3, so rank = 3 has been passed in the function.

```
emo_nmf <- modeling_nmf(3)
```

```
##
## iter |      tol
## ---------------
##    1 | 8.97e-01
##    2 | 8.19e-02
##    3 | 4.10e-02
##    4 | 1.72e-02
##    5 | 5.39e-03
##    6 | 2.38e-03
##    7 | 1.48e-03
##    8 | 9.84e-04
##    9 | 6.22e-04
##   10 | 4.10e-04
##   11 | 2.78e-04
##   12 | 1.91e-04
##   13 | 1.36e-04
##   14 | 7.59e-05
```

```
head(emo_nmf$w)
```

```
##              [,1]         [,2]         [,3]
## [1,] 2.304353e-03 0.000000e+00 2.753788e-03
## [2,] 1.929239e-44 2.532808e-43 1.243184e-42
## [3,] 0.000000e+00 3.511916e-03 2.228532e-04
## [4,] 0.000000e+00 2.233713e-03 0.000000e+00
## [5,] 4.426734e-03 2.096278e-03 1.318194e-03
## [6,] 4.480567e-03 0.000000e+00 2.426271e-03
```

```
head(emo_nmf$d)
```

```
## [1] 1120.1779 1055.9234  908.4289
```

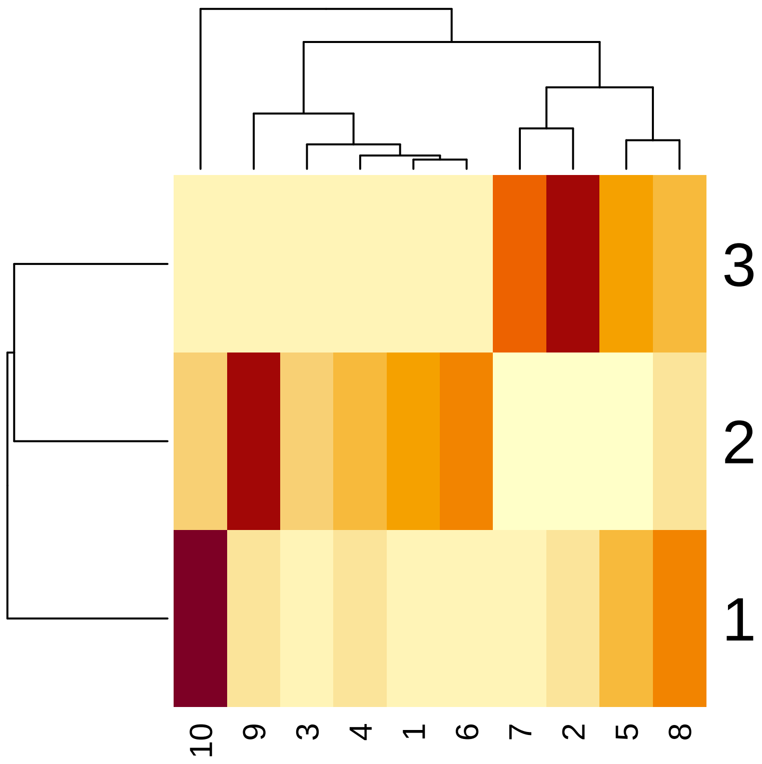
```
head(emo_nmf$h)
```

```
##            [,1]        [,2]       [,3]       [,4]      [,5]      [,6]
## [1,] 0.00000000 0.050183865 0.00000000 0.02078463 0.1595332 0.0000000
## [2,] 0.15218283 0.004814419 0.08704686 0.13200251 0.0000000 0.1782877
## [3,] 0.02419359 0.362228437 0.01309887 0.00000000 0.1861993 0.0000000
##            [,7]       [,8]        [,9]      [,10]
## [1,] 0.00000000 0.22012002 0.024054973 0.52532329
## [2,] 0.01104422 0.06239651 0.290376598 0.08184831
## [3,] 0.22549591 0.13751817 0.006954175 0.04431160
```

The heatmap_visualize function plots the heatmap which are colored differently according to their type. The grid type visualization is generated from the heatmap_visualize function which takes into the rank of 3 and cluster it as per hierarchical clustering showing the dendogram.

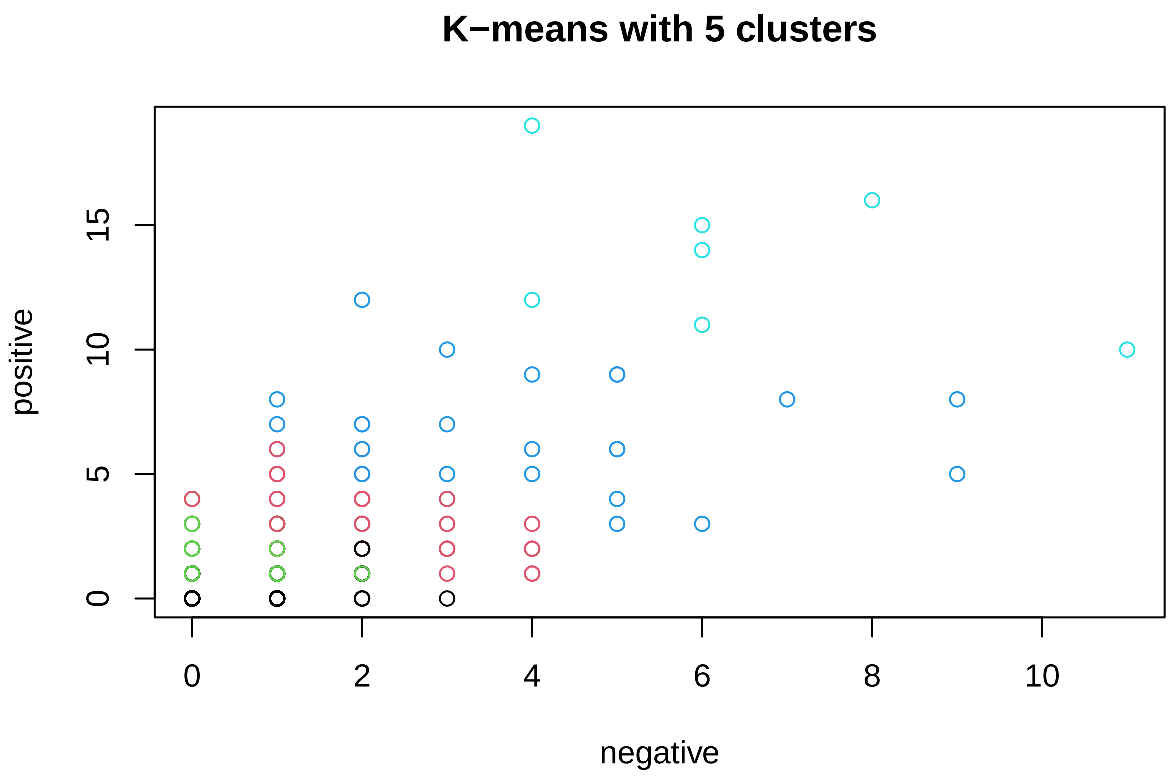
```
heatmap_visualize()
```

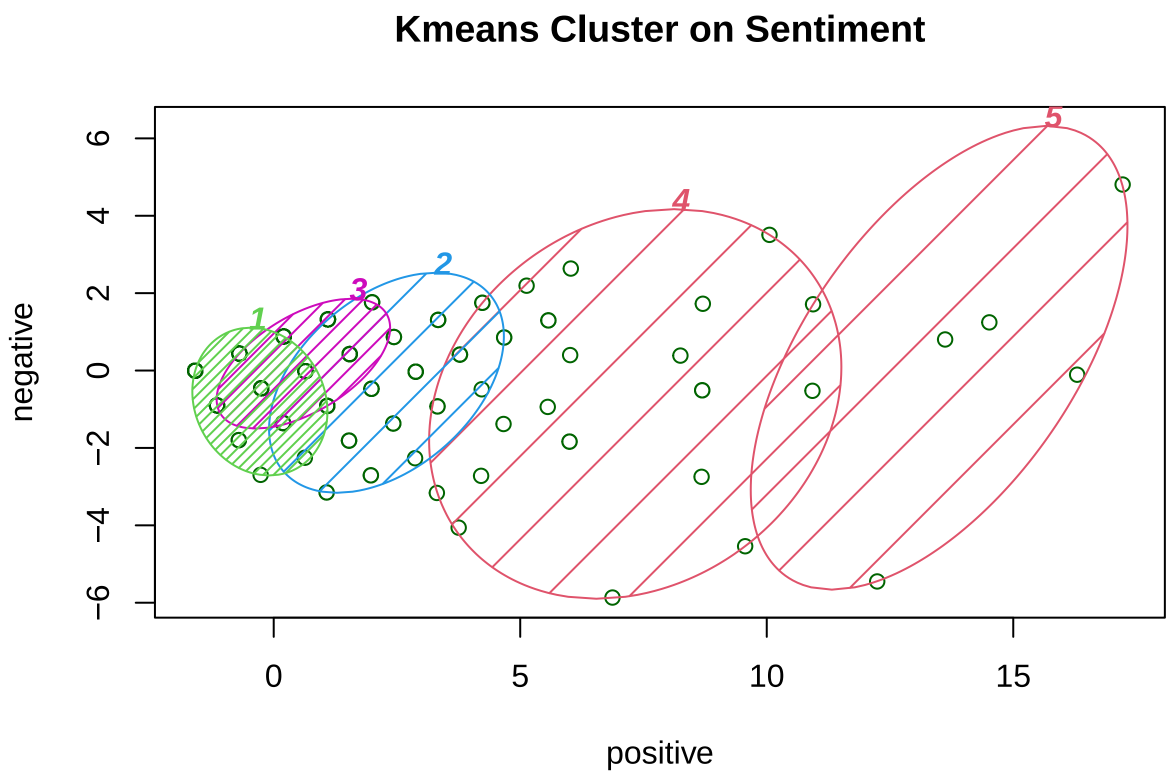
```
##
## iter |      tol
## ---------------
##    1 | 8.99e-01
##    2 | 5.25e-02
##    3 | 2.64e-02
##    4 | 2.25e-02
##    5 | 1.22e-02
##    6 | 4.34e-03
##    7 | 1.69e-03
##    8 | 8.32e-04
##    9 | 4.94e-04
##   10 | 3.34e-04
##   11 | 2.47e-04
##   12 | 1.85e-04
##   13 | 1.40e-04
##   14 | 8.42e-05
```



The cluster_kmeans function is used to cluster the group that have as per their similarity. Here, the cluster_kmeans clusters the emotion as per the similarity. Also, this function further cluster the emotion as per the negative and positive words.

```
library(cluster)
set.seed(1234) # Setting seed

cluster_kmeans(emo_mat)
```


K−means with 5 clusters
positive
negative
0
5
10
15
0
2
4
6
8
10



These two components explain 100 % of the point variability.

Aim 2: Classifying the emotion from the review of the customers performing PCA dimension reduction and h-clustering.

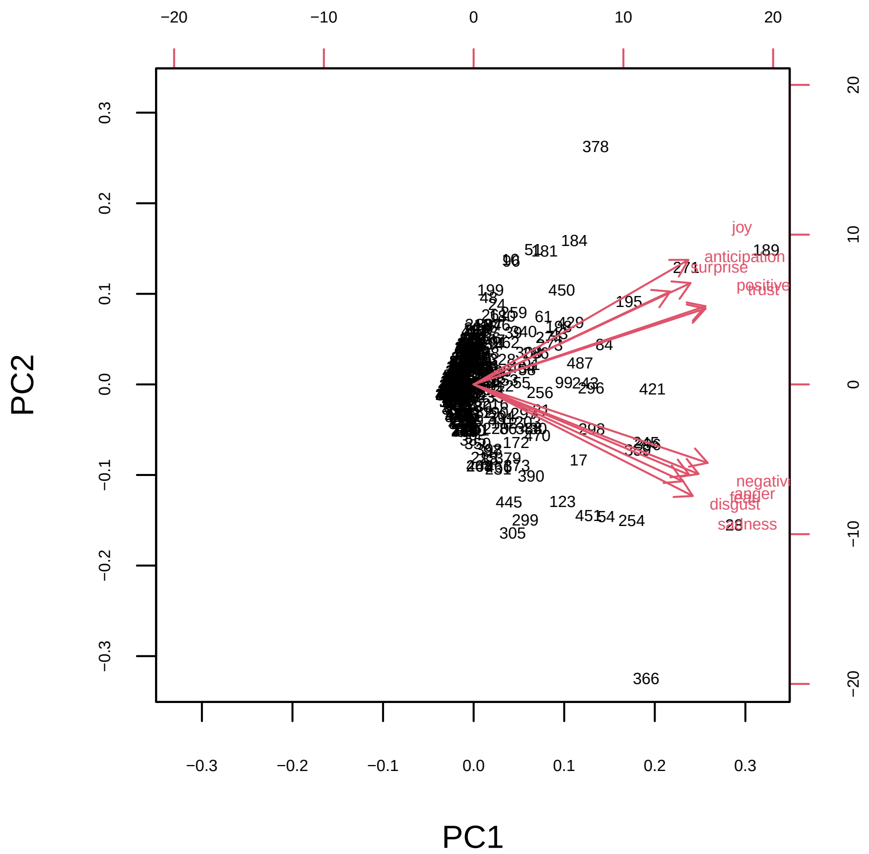
The pca_func is used to find the patterns in the high dimensional data while retaining the trends of the data. This function takes the data as input from the emotions and standardizes the input data so that it has zero mean and variance and the pca dimension reduction can be performed in the data.

```
pca_func()
```

```
## Standard deviations (1, .., p=10):
##  [1] 2.5945100 1.1228086 0.7330796 0.5672571 0.5114865 0.4983543 0.4535137
##  [8] 0.4419342 0.3873355 0.2960536
##
## Rotation (n x k) = (10 x 10):
##                   PC1        PC2         PC3          PC4         PC5
## anger       0.3236745 -0.2977592  0.21426209 -0.215536122  0.20437691
## anticipation 0.3121019  0.3368847  0.08772952 -0.061736248  0.59731907
## disgust     0.3008836 -0.3218567  0.42330705 -0.466379489 -0.40744682
## fear        0.3103829 -0.3008527 -0.35312541  0.540515353 -0.06476802
## joy         0.3089416  0.4140697 -0.11419183 -0.006658705 -0.41967135
## sadness     0.3152507 -0.3709132 -0.15361759  0.245455365 -0.16459797
## surprise    0.2829491  0.3088680  0.64478325  0.529086803 -0.09288049
## trust       0.3337320  0.2523409 -0.23694254 -0.210367907  0.03751979
## negative    0.3367815 -0.2608832 -0.03771510 -0.048247418  0.43112922
## positive     0.3335743  0.2599582 -0.36408942 -0.230777277 -0.17918910
##                     PC6        PC7         PC8         PC9        PC10
## anger       -0.34909265  0.6692503 -0.04252620  0.11822569  0.29477597
## anticipation  0.62969943  0.0609427  0.09695995  0.08599279  0.06141512
## disgust       0.30178223 -0.1982751  0.29558994 -0.06693195 -0.14466133
## fear          0.09577146  0.1808970  0.56497471 -0.13918956 -0.10786456
## joy           0.11236173  0.4503463 -0.34706848 -0.16302745 -0.42593628
## sadness       0.26925187 -0.2028546 -0.55445924  0.43207747  0.21056730
## surprise     -0.24977479 -0.1930900  0.01967747 -0.02582820  0.13967750
## trust        -0.39680278 -0.2324728  0.24946187  0.61308658 -0.27118343
## negative     -0.25688233 -0.3102916 -0.30014333 -0.46984122 -0.39470157
## positive     -0.10206913 -0.2116205  0.04042815 -0.37975307  0.63417171
```

The visualizing_pca function provides a visualization of any pattern or trends that can be shown from the dataset.

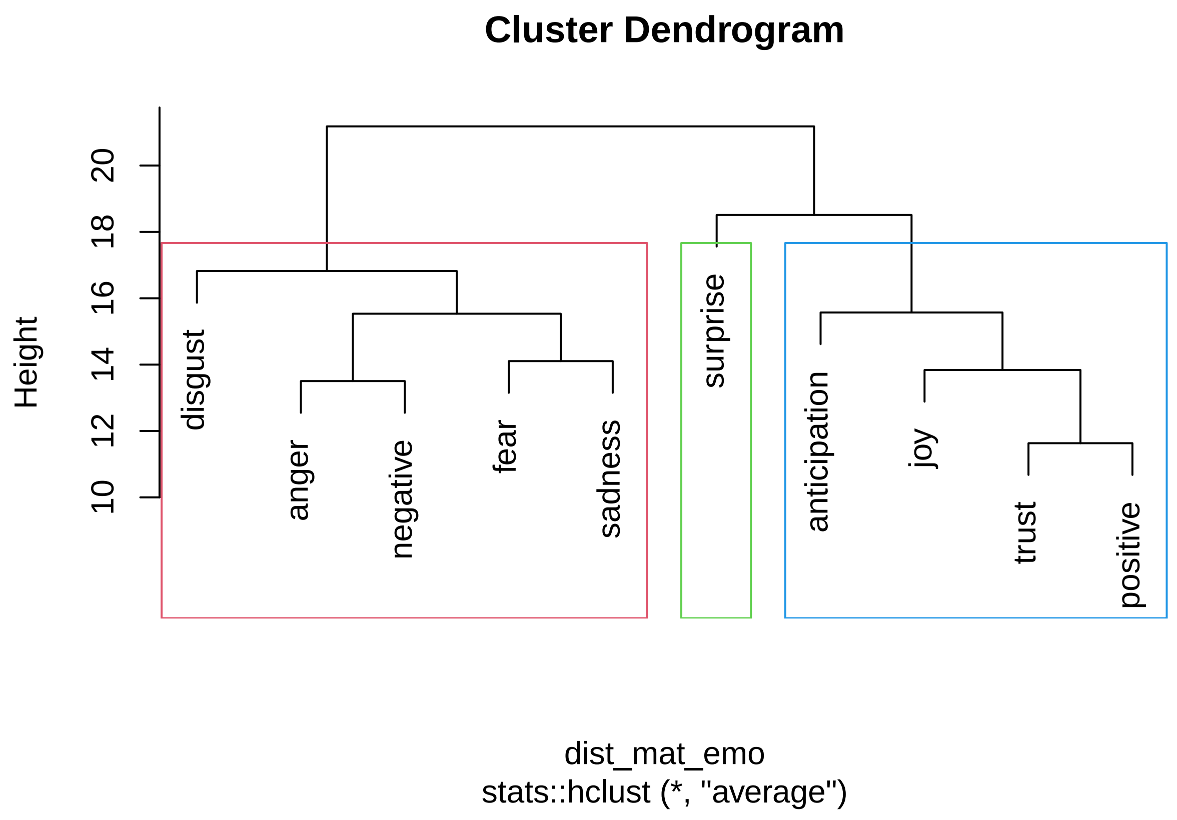
```
visualizing_pca()
```



```
## function (x, ...)
## UseMethod("biplot")
## <bytecode: 0x11b5652b0>
## <environment: namespace:stats>
```

The h_cluster function is used to perform clustering by grouping the data into a tree of clusters. First, through this function each cluster are generated and then it identifies the clusters that are closed together and is further showed in the tree as per the cluster. Here, the emotion are clustered in the dendrogram showing the cluster of emotions that shows similarity.

```
h_cluster(emo_mat)
```



Package Vignette: Real-life Examples

We also tried running using our package in the Flipkart's Customer Review dataset. The below function loads the data.

```
library(readr)
data <- read.csv('Flipkart_Customer_Review.csv')
data <- data.frame(star_rating = data$rating, review = data$review)
```

```
df_flip <- read_inbuilt_data(data)
str(df_flip)
```

```
## 'data.frame':    1000 obs. of  2 variables:
##  $ star_rating: int  5 5 4 5 5 5 4 4 5 5 ...
##  $ review     : chr  "It was nice produt. I like it's design a lot.  It's easy to carry. And.   Look
```

The get_emotion is the function to convert the review into emotion value of table. We need to pass list or dataframe as parameter in the get_emotion function.

```
emo_mat <- get_emotion(df_flip)
head(emo_mat)
```

```
##   anger anticipation disgust fear joy sadness surprise trust negative positive
## 1     0            0       0    0   0       0        0     0        0        0
## 2     0            2       0    0   2       0        1     2        0        2
## 3     1            3       0    1   1       0        1     3        1        4
## 4     0            2       0    1   3       1        1     4        2        5
## 5     0            2       0    0   1       0        1     1        1        2
## 6     2            4       1    1   5       1        0     4        1        6
```

The count_emotions function provides the count table with their frequency for the emotions. We need to pass list or dataframe as a parameter in the count_emotion function.

```
count_emotions(emo_mat)
```

```
##              count      emotion
## anger          580        anger
## anticipation  1537 anticipation
## disgust        176      disgust
## fear           327         fear
## joy           1552          joy
## sadness        450      sadness
## surprise       988     surprise
## trust         1710        trust
## negative       930     negative
## positive      2688     positive
```

Matrix conversion function helps to convert the emotion table values into matrix which is then converted to sparse matrix.

```
my_mat <-matrix_conversion(emo_mat)
head(my_mat)
```

```
##      anger anticipation disgust fear joy sadness surprise trust negative
## [1,]     0            0       0    0   0       0        0     0        0
## [2,]     0            2       0    0   2       0        1     2        0
## [3,]     1            3       0    1   1       0        1     3        1
## [4,]     0            2       0    1   3       1        1     4        2
## [5,]     0            2       0    0   1       0        1     1        1
## [6,]     2            4       1    1   5       1        0     4        1
##      positive
## [1,]        0
## [2,]        2
## [3,]        4
## [4,]        5
## [5,]        2
## [6,]        6
```

```
library(Matrix)
sparse_mat <- convert_to_sparse_matrix()
head(sparse_mat)
```

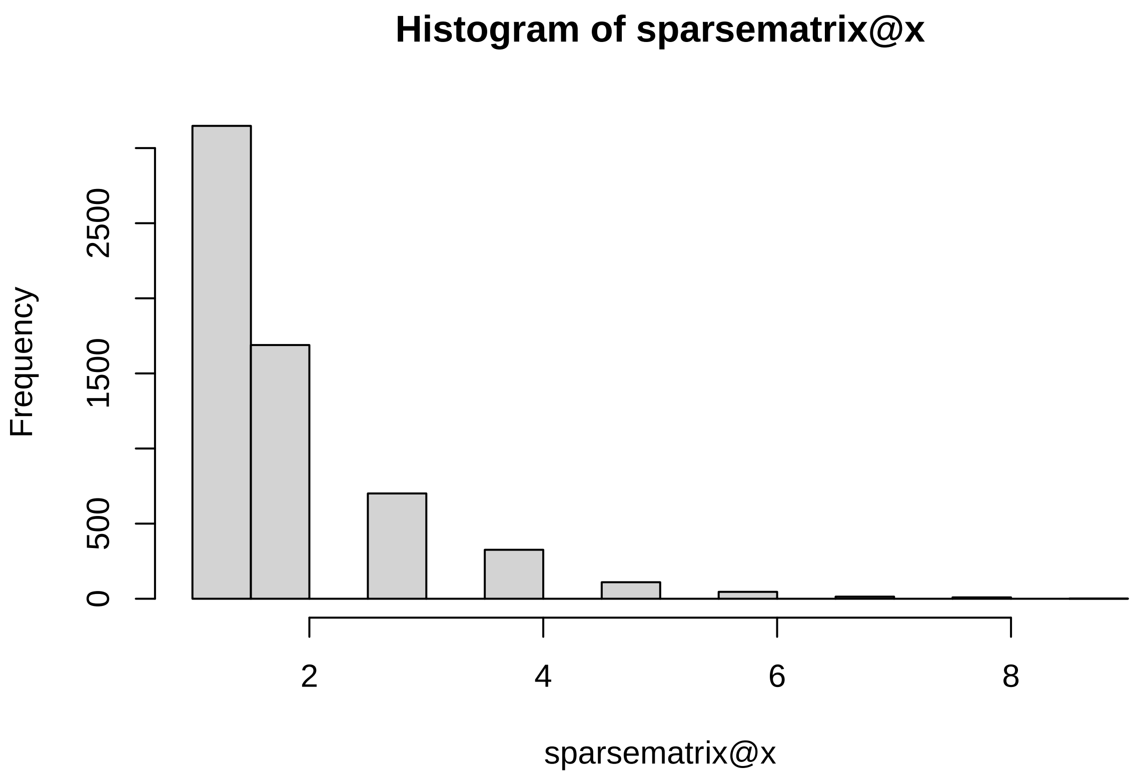
```
## 6 x 10 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 10 column names 'anger', 'anticipation', 'disgust' ... ]]

##
## [1,] . . . . . . . . . .
## [2,] . 2 . . 2 . 1 2 . 2
## [3,] 1 3 . 1 1 . 1 3 1 4
## [4,] . 2 . 1 3 1 1 4 2 5
## [5,] . 2 . . 1 . 1 1 1 2
## [6,] 2 4 1 1 5 1 . 4 1 6
```

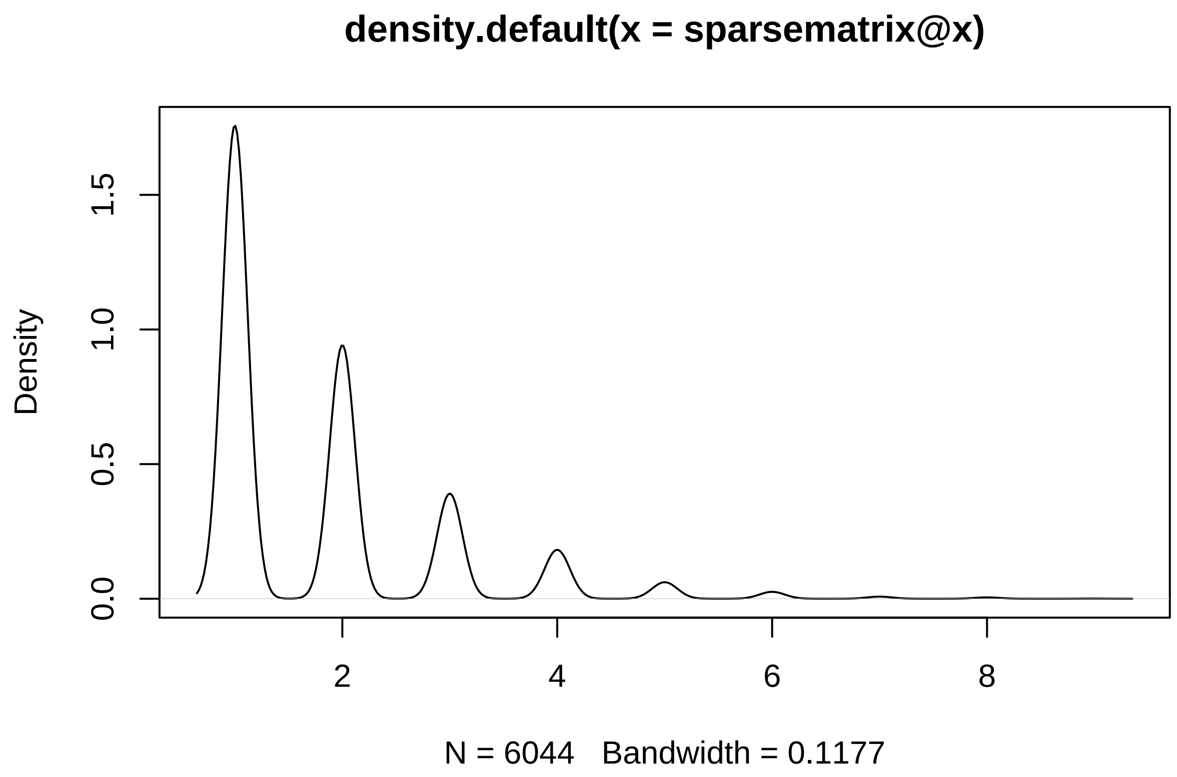
Aim 1: Classifying the emotions from the review performing NMF dimension reduction and then clustering the emotion using K-mean clustering.

The nmf_function provides the rank and the user needs to the rank to the parameter.

```
library(singlet)
nmf_func(3)
```

```
##
## iter |      tol
## ---------------
##    1 | 9.32e-01
##    2 | 9.08e-02
##    3 | 2.51e-02
##    4 | 1.08e-02
##    5 | 4.80e-03
##    6 | 2.30e-03
##    7 | 1.16e-03
##    8 | 6.46e-04
##    9 | 3.75e-04
##   10 | 1.39e-04
##   11 | 9.45e-05
```


Histogram of sparsematrix@x
Frequency
0
500
1500
2500
2
4
6
8
sparsematrix@x



```
## function (x, y, ...)
## UseMethod("plot")
## <bytecode: 0x119033120>
## <environment: namespace:base>
```

```
norm <- norm_sparse_matrix()

head(norm)
```

```
## 6 x 10 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 10 column names 'anger', 'anticipation', 'disgust' ... ]]

##
## [1,] .        .        .        .        .        .       .        .
## [2,] .        2.639940 .        .        2.630924 .       2.408876 2.541280
## [3,] 2.903693 3.021329 .        3.452557 2.007314 .       2.408876 2.920139
## [4,] .        2.639940 .        3.452557 3.012093 3.14511 2.408876 3.194248
## [5,] .        2.639940 .        .        2.007314 .       2.408876 1.923950
## [6,] 3.569047 3.296752 4.057303 3.452557 3.503047 3.14511 .        3.194248
##
## [1,] .        .
## [2,] .        2.133039
## [3,] 2.464082 2.765120
## [4,] 3.113754 2.975590
## [5,] 2.464082 2.133039
## [6,] 2.464082 3.149373
```

```
emo_nmf <- modeling_nmf(3)
```

```
##
## iter |      tol
## ---------------
##    1 | 9.38e-01
##    2 | 1.04e-01
##    3 | 1.67e-02
##    4 | 8.37e-03
##    5 | 5.37e-03
##    6 | 4.07e-03
##    7 | 3.29e-03
##    8 | 2.40e-03
##    9 | 1.92e-03
##   10 | 1.44e-03
##   11 | 9.87e-04
##   12 | 6.57e-04
##   13 | 4.52e-04
##   14 | 3.13e-04
##   15 | 2.21e-04
##   16 | 1.61e-04
##   17 | 1.21e-04
##   18 | 9.10e-05
```

```
head(emo_nmf$w)
```

```
##              [,1]         [,2]         [,3]
## [1,] 2.563253e-67 3.656817e-66 1.306936e-62
## [2,] 8.711689e-04 1.332461e-03 0.000000e+00
## [3,] 1.400340e-03 1.535768e-03 1.199074e-03
## [4,] 2.001073e-03 1.544590e-03 1.753631e-03
## [5,] 6.623152e-04 9.952473e-04 6.352714e-04
## [6,] 2.480339e-03 2.197646e-03 1.761689e-03
```

```
head(emo_nmf$d)
```

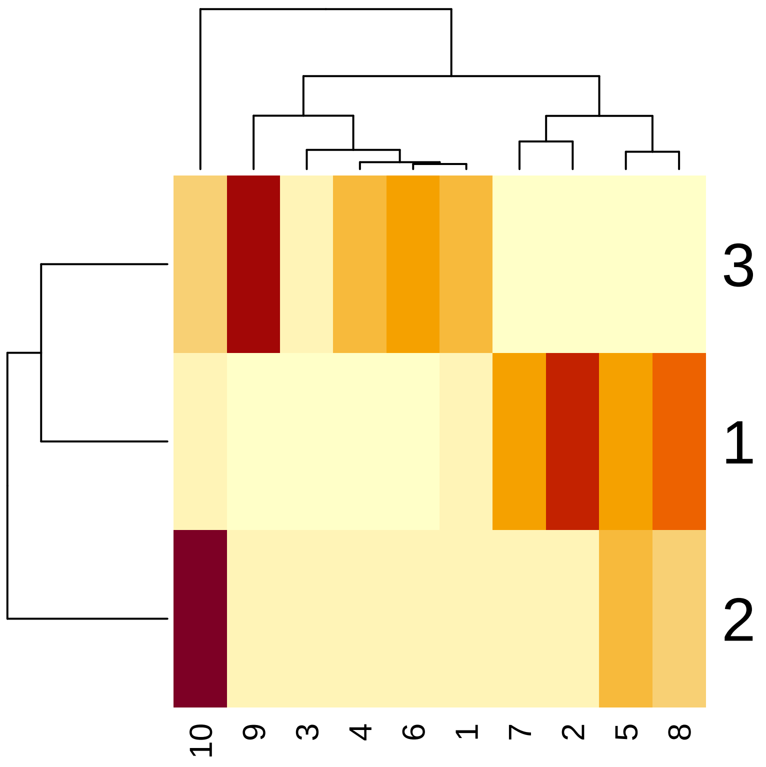
```
## [1] 4216.775 4149.774 2618.878
```

```
head(emo_nmf$h)
```

```
##            [,1]       [,2]       [,3]      [,4]      [,5]        [,6]
## [1,] 0.01531692 0.03191325 0.00000000 0.0000000 0.1944774 0.008435992
## [2,] 0.02148774 0.34003837 0.00000000 0.0000000 0.1787761 0.000000000
## [3,] 0.15709459 0.00000000 0.07381152 0.1408966 0.0000000 0.170963574
##            [,7]        [,8]        [,9]     [,10]
## [1,] 0.02144749 0.162239984 0.000000000 0.5661689
## [2,] 0.20819709 0.248607385 0.002893324 0.0000000
## [3,] 0.00000000 0.005790185 0.342520195 0.1089233
```

```
heatmap_visualize()
```

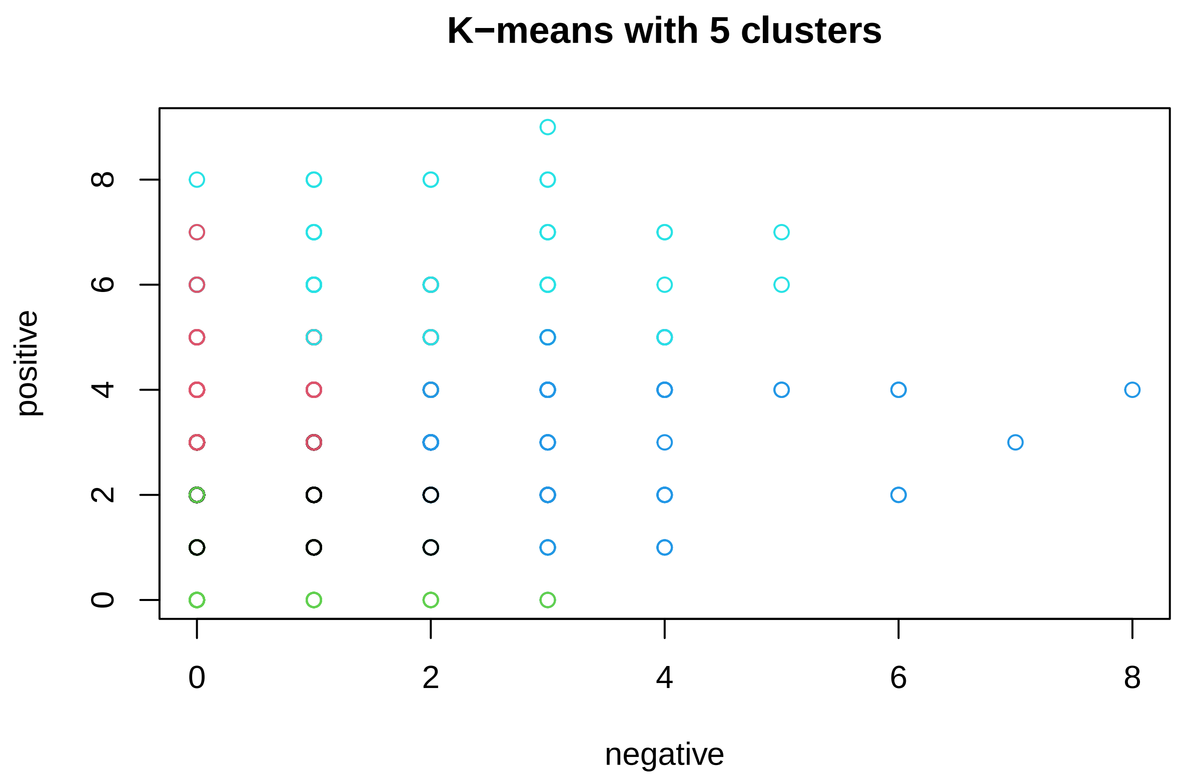
```
##
## iter |       tol
## ---------------
##    1 | 9.66e-01
##    2 | 1.16e-01
##    3 | 1.80e-02
##    4 | 9.17e-03
##    5 | 5.02e-03
##    6 | 2.75e-03
##    7 | 1.52e-03
##    8 | 8.73e-04
##    9 | 5.20e-04
##   10 | 3.34e-04
##   11 | 2.28e-04
##   12 | 1.61e-04
##   13 | 1.19e-04
##   14 | 8.07e-05
```



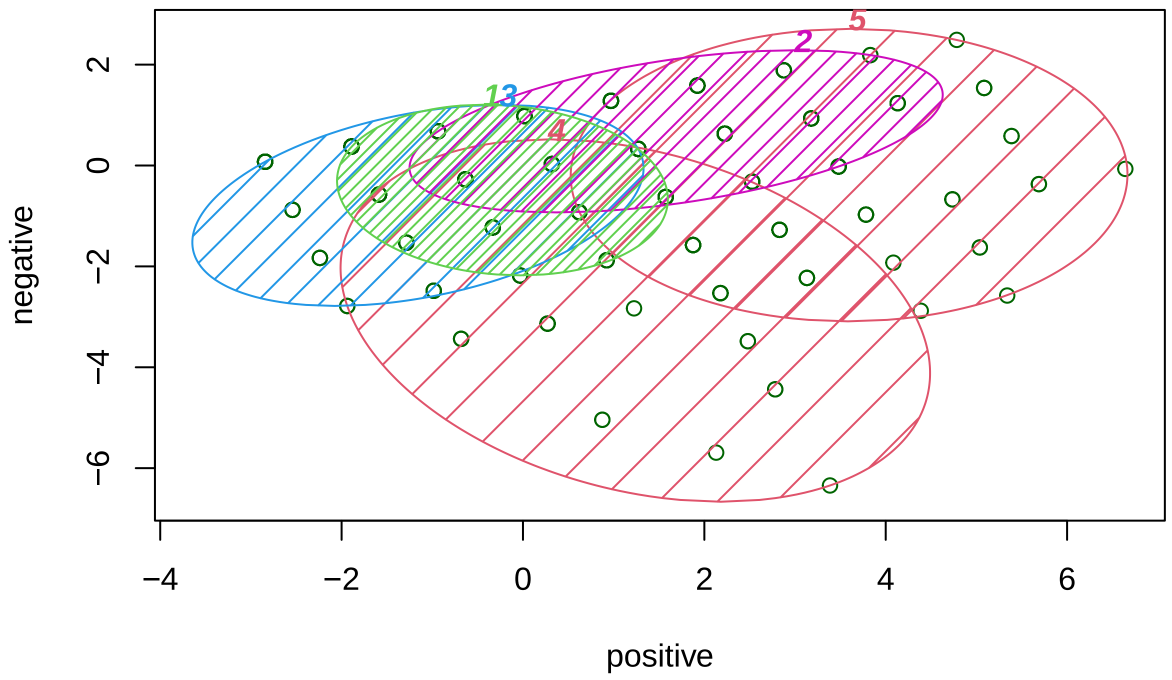
After running the NMF, the clustering is done for the emotions using cluster_kmeans function which clusters the emotion in comparison with positive and negative words too.

```
library(cluster)
set.seed(1234) # Setting seed
```

```
cluster_kmeans(emo_mat)
```



Kmeans Cluster on Sentiment



These two components explain 100 % of the point variability.

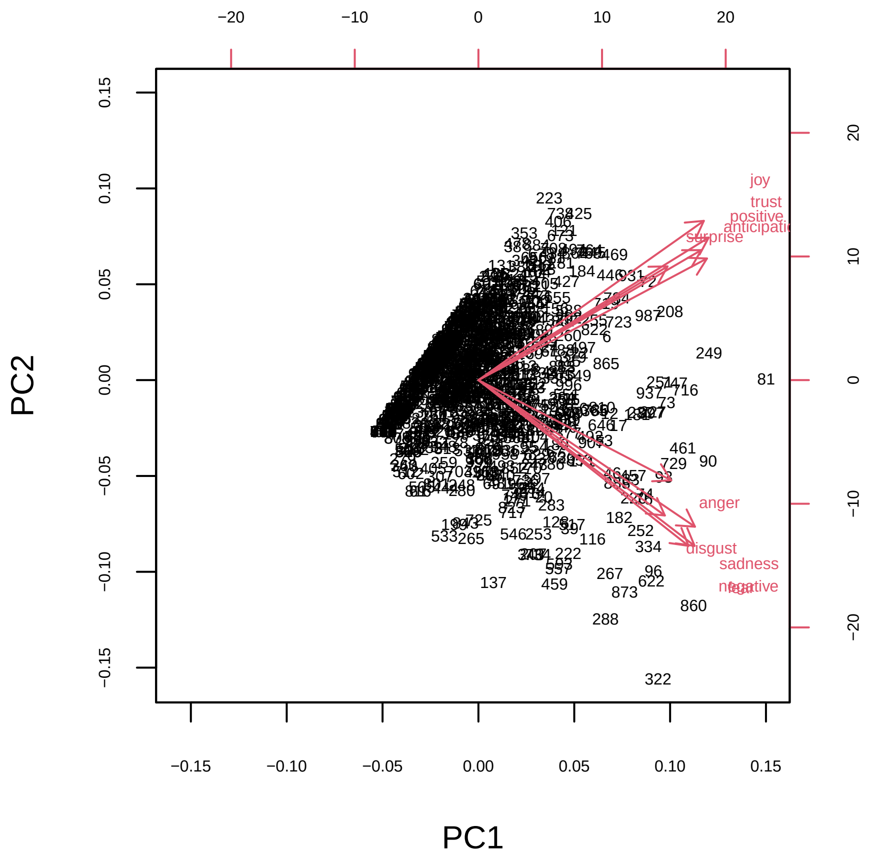
Aim 2: Classifying the emotion from the review of the customers performing PCA dimension reduction and h-clustering.

The pca_func is used to remove the redundant and noisy data from the dataset. The dimensions of the emotions is reduced.

```
pca_func()
```

```
## Standard deviations (1, .., p=10):
##  [1] 2.1475499 1.4120940 0.8857884 0.7919496 0.7217610 0.6334597 0.5652765
##  [8] 0.5547694 0.5150372 0.4091883
##
## Rotation (n x k) = (10 x 10):
##                    PC1        PC2        PC3          PC4         PC5
## anger        0.2870551 -0.2270588  0.6232659 -0.402949189  0.36051264
## anticipation 0.3402524  0.2753595  0.1480555  0.178998839 -0.28734976
## disgust      0.2775645 -0.3065634 -0.2180522  0.489377173  0.69436922
## fear         0.3127706 -0.3751627 -0.1279885  0.153102169 -0.32620906
## joy          0.3356144  0.3605531 -0.2361540 -0.122069856  0.10785055
## sadness      0.3224066 -0.3319065 -0.3586022 -0.001459149 -0.29854331
## surprise     0.2816223  0.2574477  0.4950456  0.560343278 -0.15194724
## trust        0.3427654  0.3226013 -0.1713431 -0.114211881  0.09780741
## negative     0.3218353 -0.3752166  0.1623216 -0.281074366 -0.21959555
## positive     0.3316622  0.2942693 -0.2013708 -0.348892756  0.13139053
##                     PC6         PC7         PC8        PC9        PC10
## anger        -0.05206324  0.03198238  0.06187405 -0.3274579 -0.26543427
## anticipation  0.54010230  0.51154350 -0.28997656 -0.1518110 -0.10655648
## disgust       0.12504768  0.11363607 -0.11510342  0.1066877  0.08130367
## fear          0.06933707  0.09408117  0.72912385 -0.2285984  0.13287613
## joy          -0.12601332 -0.26322594 -0.14857884 -0.5697930  0.49310847
## sadness      -0.37222472 -0.07797155 -0.41012699 -0.1259128 -0.48936417
## surprise     -0.44014807 -0.20546179  0.03366361  0.1850247  0.03089706
## trust         0.38484227 -0.53129473  0.24545116  0.2630751 -0.40821402
## negative      0.18501946 -0.20116293 -0.28066076  0.4544759  0.49197935
## positive     -0.39323997  0.52460779  0.19179421  0.3944314  0.04066655
```

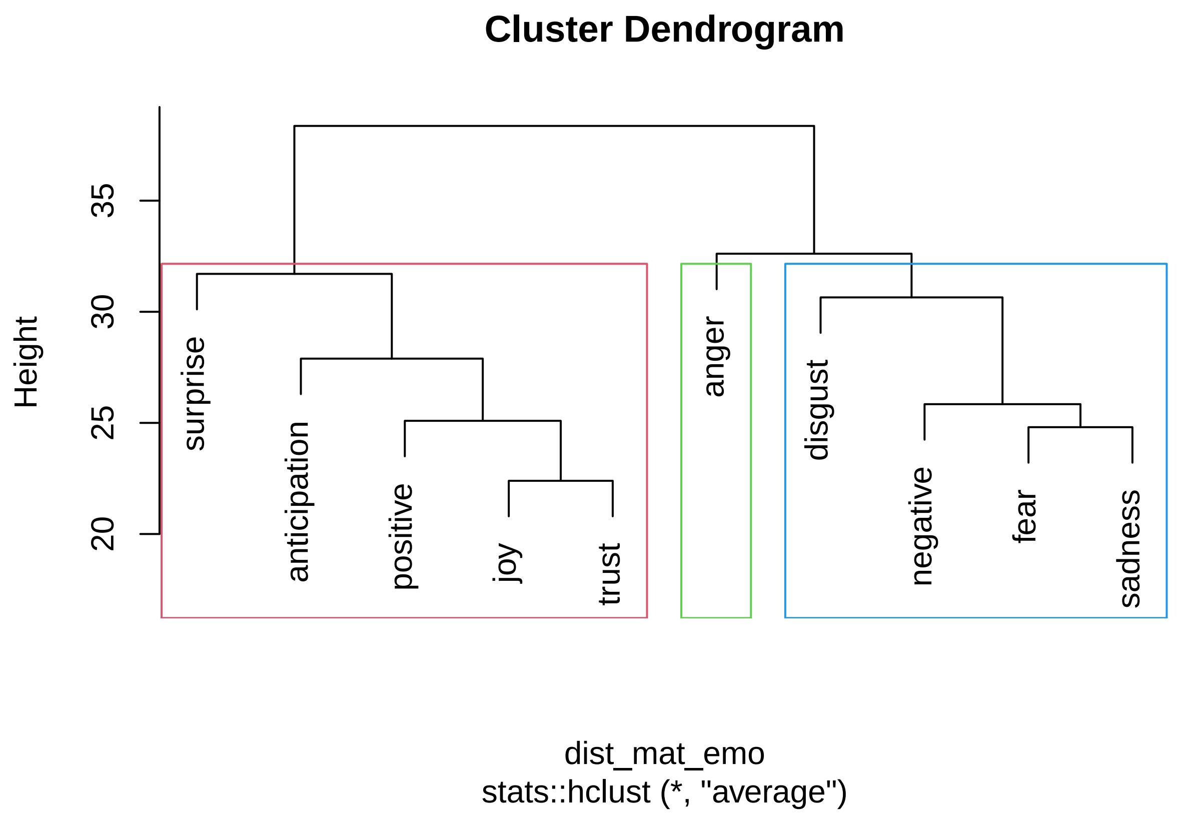
```
visualizing_pca()
```



```
## function (x, ...)
## UseMethod("biplot")
## <bytecode: 0x11b5652b0>
## <environment: namespace:stats>
```

After running the pca_func and reducing the dimension, the h_cluster function is performed to classify the emotions.

```
h_cluster(emo_mat)
```



Conclusion

From running the package, we came to find out that both the clustering method was good at classifying the emotions, however, for the dimension reduction NMF provided a meaningful information whereas the PCA didn't provided information as expected.

References

1. https://medium.com/swlh/exploring-sentiment-analysis-a6b53b026131
2. https://statisticsglobe.com/convert-data-frame-to-matrix-in-r
3. https://satijalab.org/seurat/articles/dim_reduction_vignette.html
4. https://rpubs.com/JanpuHou/278584
5. https://www.geeksforgeeks.org/k-means-clustering-in-r-programming/
6. https://www.datacamp.com/tutorial/hierarchical-clustering-R
7. https://r-pkgs.org/man.html