**0-intro.R**

Exercise Code:

## exercise: select 2x2 subsection from the "bottom left" of matrix mm

## [your code here]

mm[c(3,4),c(1,2)]

Console Output:

> ## exercise: select 2x2 subsection from the "bottom left" of matrix mm

>

> ## [your code here]

>

> mm[c(3,4),c(1,2)]

[,1] [,2]

[1,] 3 7

[2,] 4 8

**1-data.R**

Exercise Code:

## EXERCISE

## obtain this data view from "df":

## X Grad.Rate

## 1 James Madison University 98

## 2 Incarnate Word College 95

## 3 Johns Hopkins University 90

## 4 John Carroll University 89

## 5 Kenyon College 88

## 6 King's College 87

## 7 La Salle University 84

## 8 Illinois Wesleyan University 83

## 9 Juniata College 80

## [your code here]

dy <- filter(df, Grad.Rate>=80)

dy

dz<-arrange(dy,desc(Grad.Rate))

select(dz,X,Grad.Rate)

## EXERCISE

## find max and min tuition ("Outstate") grouped by private/public

## school, in dataset 'df' and 'college'

college

## DF:

##

## Private max min

## 1 No 9766 3946

## 2 Yes 19240 6398

## college:

##

## Private max min

## 1 No 15732 2580

## 2 Yes 21700 2340

## [your code here]

dfx <- group\_by(df, Private)

dfx <-summarise(dfx,max=max(Outstate),min=min(Outstate))

dfx

dfy <- group\_by(college, Private)

dfy <-summarise(dfy,max=max(Outstate),min=min(Outstate))

dfy

Console Output:

## [your code here]

> dy <- filter(df, Grad.Rate>=80)

> dy

X Private Apps Accept Enroll Top10perc Top25perc F.Undergrad

1 Illinois Wesleyan University Yes 3050 1342 471 55 86 1818

2 Incarnate Word College Yes 1163 927 386 16 49 1685

3 James Madison University No 11223 5285 2082 32 72 9652

4 John Carroll University Yes 2421 2109 820 27 57 3168

5 Johns Hopkins University Yes 8474 3446 911 75 94 3566

6 Juniata College Yes 1005 859 298 36 55 1075

7 Kenyon College Yes 2212 1538 408 44 75 1445

8 King's College Yes 1456 1053 381 20 45 500

9 La Salle University Yes 2929 1834 622 20 56 2738

P.Undergrad Outstate Room.Board Books Personal PhD Terminal S.F.Ratio perc.alumni Expend

1 23 14360 4090 400 650 77 92 12.9 34 9605

2 556 8840 4689 750 2775 67 69 11.4 21 6095

3 742 7994 4544 500 732 77 81 17.9 29 5212

4 392 11700 5550 600 450 89 90 14.5 28 7738

5 1569 18800 6740 500 1040 96 97 3.3 38 56233

6 43 14850 4460 450 420 97 97 12.7 37 12067

7 1 19240 3690 750 480 95 95 11.1 46 14067

8 541 10910 5160 400 1795 66 72 15.6 37 7649

9 1662 12600 5610 450 3160 90 90 15.1 9 9084

Grad.Rate

1 83

2 95

3 98

4 89

5 90

6 80

7 88

8 87

9 84

> dz<-arrange(dy,desc(Grad.Rate))

> select(dz,X,Grad.Rate)

X Grad.Rate

1 James Madison University 98

2 Incarnate Word College 95

3 Johns Hopkins University 90

4 John Carroll University 89

5 Kenyon College 88

6 King's College 87

7 La Salle University 84

8 Illinois Wesleyan University 83

9 Juniata College 80

> dfx <- group\_by(df, Private)

> dfx <-summarise(dfx,max=max(Outstate),min=min(Outstate))

> dfx

# A tibble: 2 x 3

Private max min

*<fct>* *<int>* *<int>*

1 No 9766 3946

2 Yes 19240 6398

> dfy <- group\_by(college, Private)

> dfy <-summarise(dfy,max=max(Outstate),min=min(Outstate))

> dfy

# A tibble: 2 x 3

Private max min

*<fct>* *<int>* *<int>*

1 No 15732 2580

2 Yes 21700 2340

8-ml.R

Now, for the important part: your homework! Repeat this lab with a different dataset. You are free to download your own, or to use CRAN-built-in datasets, such as for example: data(diamonds)

library(caret)

data(CO2)

# rename the dataset

dataset <- CO2

#You now have the iris data loaded in R and accessible via the dataset variable.

# define the filename

#filename <- "CO2.csv"

# load the CSV file from the local directory

#dataset <- read.csv(filename, header=FALSE)

# set the column names in the dataset

colnames(dataset) <- c("Plant","Type","Treatment","conc","uptake")

# create a list of 80% of the rows in the original dataset we can use for training

validation\_index <- createDataPartition(dataset$Treatment, p=0.80, list=FALSE)

# select 20% of the data for validation

validation <- dataset[-validation\_index,]

# use the remaining 80% of data to training and testing the models

dataset <- dataset[validation\_index,]

# dimensions of dataset

dim(dataset)

# list types for each attribute

sapply(dataset, class)

# take a peek at the first 5 rows of the data

head(dataset)

# list the levels for the class

levels(dataset$Treatment)

# summarize the class distribution

percentage <- prop.table(table(dataset$Treatment)) \* 100

cbind(freq=table(dataset$Treatment), percentage=percentage)

# summarize attribute distributions

summary(dataset)

## ============================================================================

## 4. Visualize dataset

## ============================================================================

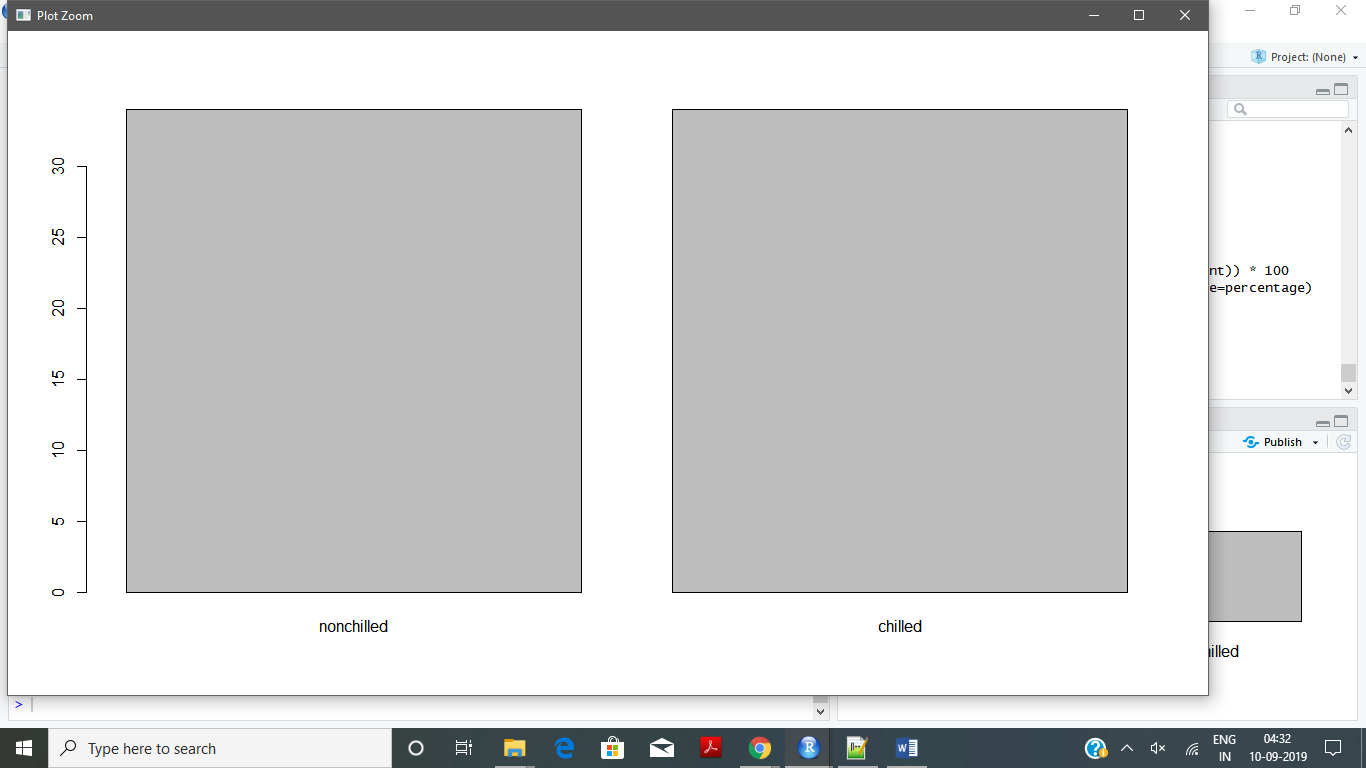
# split input and output

x <- dataset[,c(1,2,4,5)]

y <- dataset[,3]

# barplot for class breakdown

plot(y)



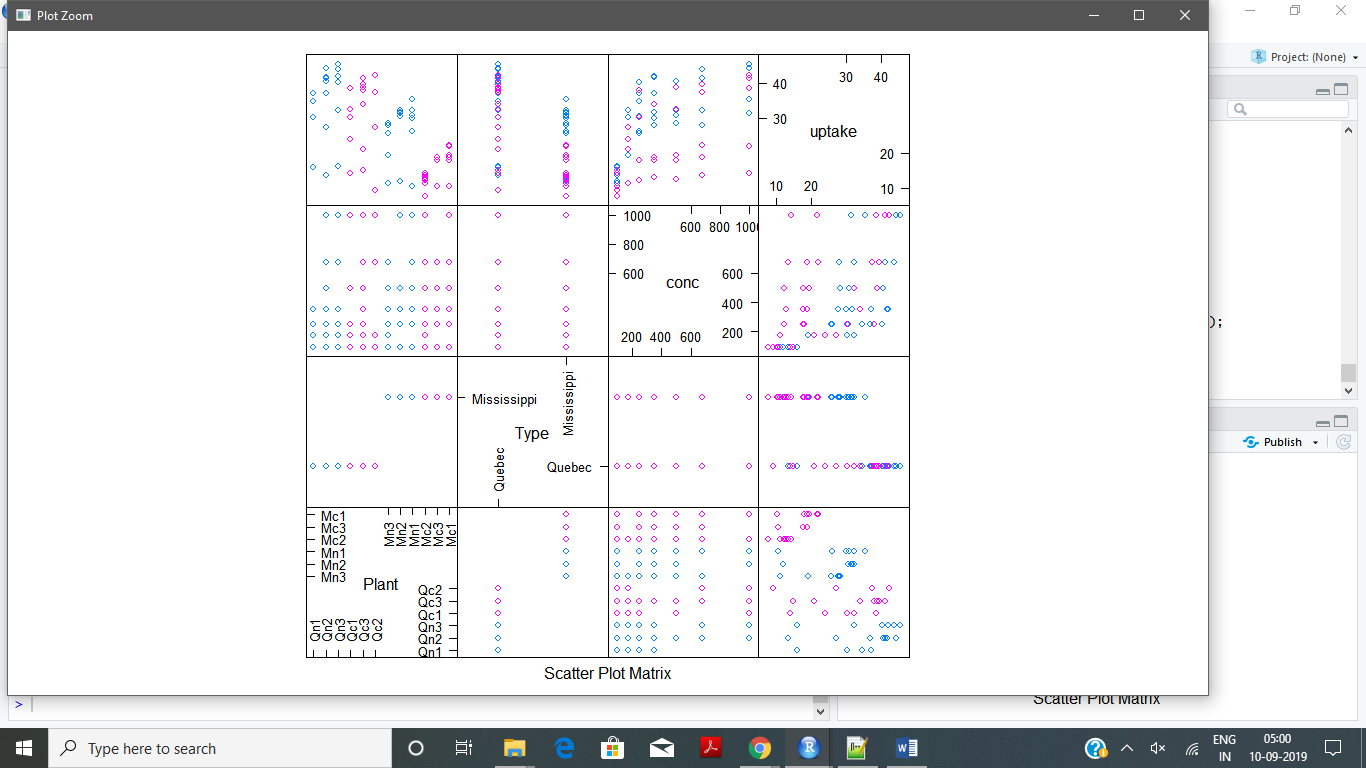
# scatterplot matrix

#install.packages("ISLR")

#install.packages("stringi")

library(ISLR); library(ggplot2); library(caret);

featurePlot(x=x, y=y, plot="pairs")



# box and whisker plots for each attribute

featurePlot(x=x, y=y, plot="box")

# density plots for each attribute by class value

#scales <- list(x=list(relation="free"), y=list(relation="free"))

#featurePlot(x=x, y=y, plot="density", scales=scales)

## ============================================================================

## 4. Evaluate ML algorithms

## ============================================================================

#We will use 10-fold crossvalidation to estimate accuracy.

# Run algorithms using 10-fold cross validation

control <- trainControl(method="cv", number=20)

metric <- "Accuracy"

# a) linear algorithms

#lda

library(e1071)

set.seed(7)

fit.lda <- train(Treatment~., data=dataset, method="lda", metric=metric, trControl=control)

# b) nonlinear algorithms

# kNN

set.seed(7)

fit.knn <- train(Treatment~., data=dataset, method="knn", metric=metric, trControl=control)

# c) SVMLinear2 algorithm

set.seed(7)

fit.svnLinear2 <-train(Treatment~., data=dataset, method="svmLinear2", metric=metric, trControl=control)

# d) Random Forest Algorithm

set.seed(7)

fit.rf <-train(Treatment~., data=dataset, method="rf", metric=metric, trControl=control)

# e) Ada Boost Classification Algortihm

set.seed(7)

fit.adaboost <- train(Treatment~., data=dataset, method="adaboost", metric=metric, trControl=control)

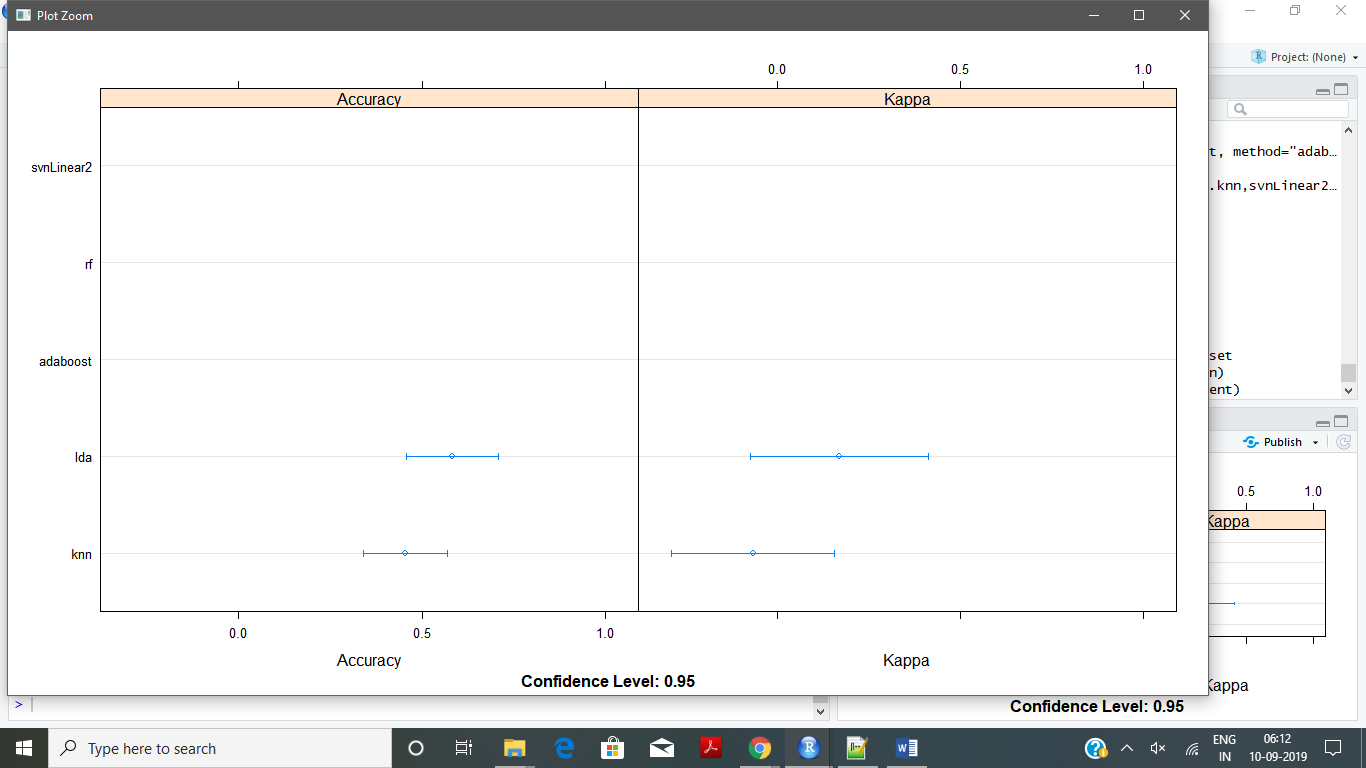
# summarize accuracy of models

results <- resamples(list(lda=fit.lda, knn=fit.knn,svnLinear2=fit.svnLinear2,rf=fit.rf,adaboost=fit.adaboost))

summary(results)

# compare accuracy of models

dotplot(results)



# summarize Best Model

print(fit.lda)

print(fit.adaboost)

print(fit.knn)

print(fit.svnLinear2)

print(fit.rf)

# estimate skill of LDA on the validation dataset

predictions <- predict(fit.adaboost, validation)

confusionMatrix(predictions, validation$Treatment)

Console Output:

> library(caret)

>

> data(CO2)

>

> # rename the dataset

> dataset <- CO2

>

> #You now have the iris data loaded in R and accessible via the dataset variable.

>

> # define the filename

> #filename <- "CO2.csv"

>

> # load the CSV file from the local directory

> #dataset <- read.csv(filename, header=FALSE)

>

> # set the column names in the dataset

> colnames(dataset) <- c("Plant","Type","Treatment","conc","uptake")

>

> # create a list of 80% of the rows in the original dataset we can use for training

> validation\_index <- createDataPartition(dataset$Treatment, p=0.80, list=FALSE)

>

> # select 20% of the data for validation

> validation <- dataset[-validation\_index,]

>

> # use the remaining 80% of data to training and testing the models

> dataset <- dataset[validation\_index,]

>

> # dimensions of dataset

> dim(dataset)

[1] 68 5

>

> # list types for each attribute

> sapply(dataset, class)

$`Plant`

[1] "ordered" "factor"

$Type

[1] "factor"

$Treatment

[1] "factor"

$conc

[1] "numeric"

$uptake

[1] "numeric"

>

> # take a peek at the first 5 rows of the data

> head(dataset)

Grouped Data: uptake ~ conc | Plant

Plant Type Treatment conc uptake

1 Qn1 Quebec nonchilled 95 16.0

2 Qn1 Quebec nonchilled 175 30.4

3 Qn1 Quebec nonchilled 250 34.8

4 Qn1 Quebec nonchilled 350 37.2

6 Qn1 Quebec nonchilled 675 39.2

8 Qn2 Quebec nonchilled 95 13.6

>

> # list the levels for the class

> levels(dataset$Treatment)

[1] "nonchilled" "chilled"

>

> # summarize the class distribution

> percentage <- prop.table(table(dataset$Treatment)) \* 100

> cbind(freq=table(dataset$Treatment), percentage=percentage)

freq percentage

nonchilled 34 50

chilled 34 50

>

> # summarize attribute distributions

> summary(dataset)

Plant Type Treatment conc uptake

Qc1 : 7 Quebec :34 nonchilled:34 Min. : 95.0 Min. : 7.70

Qn2 : 6 Mississippi:34 chilled :34 1st Qu.: 175.0 1st Qu.:17.98

Qc2 : 6 Median : 350.0 Median :28.30

Mn3 : 6 Mean : 424.6 Mean :26.99

Mn2 : 6 3rd Qu.: 675.0 3rd Qu.:35.92

Mn1 : 6 Max. :1000.0 Max. :44.30

(Other):31

>

> ## ============================================================================

> ## 4. Visualize dataset

> ## ============================================================================

>

> # split input and output

> x <- dataset[,c(1,2,4,5)]

> y <- dataset[,3]

>

> # barplot for class breakdown

> plot(y)

>

> # scatterplot matrix

> #install.packages("ISLR")

> #install.packages("stringi")

>

> library(ISLR); library(ggplot2); library(caret);

> featurePlot(x=x, y=y, plot="pairs")

>

> # box and whisker plots for each attribute

> featurePlot(x=x, y=y, plot="box")

Error in `$<-.data.frame`(`\*tmp\*`, ".y", value = c(1L, 1L, 1L, 1L, 1L, :

replacement has 272 rows, data has 136

In addition: Warning message:

In stack.data.frame(x) : non-vector columns will be ignored

>

>

> # density plots for each attribute by class value

> scales <- list(x=list(relation="free"), y=list(relation="free"))

> featurePlot(x=x, y=y, plot="density", scales=scales)

Error in `$<-.data.frame`(`\*tmp\*`, ".y", value = c(1L, 1L, 1L, 1L, 1L, :

replacement has 272 rows, data has 136

In addition: Warning message:

In stack.data.frame(x) : non-vector columns will be ignored

>

> ## ============================================================================

> ## 4. Evaluate ML algorithms

> ## ============================================================================

>

> #We will use 10-fold crossvalidation to estimate accuracy.

>

> # Run algorithms using 10-fold cross validation

> control <- trainControl(method="cv", number=20)

> metric <- "Accuracy"

>

> # a) linear algorithms

> #lda

> library(e1071)

> set.seed(7)

> fit.lda <- train(Treatment~., data=dataset, method="lda", metric=metric, trControl=control)

There were 21 warnings (use warnings() to see them)

>

>

> # b) nonlinear algorithms

> # kNN

> set.seed(7)

> fit.knn <- train(Treatment~., data=dataset, method="knn", metric=metric, trControl=control)

>

> # c) SVMLinear2 algorithm

> set.seed(7)

> fit.svnLinear2 <-train(Treatment~., data=dataset, method="svmLinear2", metric=metric, trControl=control)

>

> # d) Random Forest Algorithm

> set.seed(7)

> fit.rf <-train(Treatment~., data=dataset, method="rf", metric=metric, trControl=control)

>

> # e) Ada Boost Classification Algortihm

> set.seed(7)

> fit.adaboost <- train(Treatment~., data=dataset, method="adaboost", metric=metric, trControl=control)

>

> # summarize accuracy of models

> results <- resamples(list(lda=fit.lda, knn=fit.knn,svnLinear2=fit.svnLinear2,rf=fit.rf,adaboost=fit.adaboost))

> summary(results)

Call:

summary.resamples(object = results)

Models: lda, knn, svnLinear2, rf, adaboost

Number of resamples: 20

Accuracy

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

lda 0.25 0.3333333 0.5 0.5833333 0.75 1 0

knn 0.00 0.3333333 0.5 0.4541667 0.50 1 0

svnLinear2 1.00 1.0000000 1.0 1.0000000 1.00 1 0

rf 1.00 1.0000000 1.0 1.0000000 1.00 1 0

adaboost 1.00 1.0000000 1.0 1.0000000 1.00 1 0

Kappa

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

lda -0.5 0.0 0 0.170 0.5 1 0

knn -1.0 -0.5 0 -0.065 0.0 1 0

svnLinear2 1.0 1.0 1 1.000 1.0 1 0

rf 1.0 1.0 1 1.000 1.0 1 0

adaboost 1.0 1.0 1 1.000 1.0 1 0

>

>

> # compare accuracy of models

> dotplot(results)

>

> # summarize Best Model

> print(fit.lda)

Linear Discriminant Analysis

68 samples

4 predictor

2 classes: 'nonchilled', 'chilled'

No pre-processing

Resampling: Cross-Validated (20 fold)

Summary of sample sizes: 64, 64, 64, 65, 66, 64, ...

Resampling results:

Accuracy Kappa

0.5833333 0.17

> print(fit.adaboost)

AdaBoost Classification Trees

68 samples

4 predictor

2 classes: 'nonchilled', 'chilled'

No pre-processing

Resampling: Cross-Validated (20 fold)

Summary of sample sizes: 64, 64, 64, 65, 66, 64, ...

Resampling results across tuning parameters:

nIter method Accuracy Kappa

50 Adaboost.M1 1 1

50 Real adaboost 1 1

100 Adaboost.M1 1 1

100 Real adaboost 1 1

150 Adaboost.M1 1 1

150 Real adaboost 1 1

Accuracy was used to select the optimal model using the largest value.

The final values used for the model were nIter = 50 and method = Adaboost.M1.

> print(fit.knn)

k-Nearest Neighbors

68 samples

4 predictor

2 classes: 'nonchilled', 'chilled'

No pre-processing

Resampling: Cross-Validated (20 fold)

Summary of sample sizes: 64, 64, 64, 65, 66, 64, ...

Resampling results across tuning parameters:

k Accuracy Kappa

5 0.3333333 -0.270

7 0.4541667 -0.065

9 0.4250000 -0.130

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was k = 7.

> print(fit.svnLinear2)

Support Vector Machines with Linear Kernel

68 samples

4 predictor

2 classes: 'nonchilled', 'chilled'

No pre-processing

Resampling: Cross-Validated (20 fold)

Summary of sample sizes: 64, 64, 64, 65, 66, 64, ...

Resampling results across tuning parameters:

cost Accuracy Kappa

0.25 1 1

0.50 1 1

1.00 1 1

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was cost = 0.25.

> print(fit.rf)

Random Forest

68 samples

4 predictor

2 classes: 'nonchilled', 'chilled'

No pre-processing

Resampling: Cross-Validated (20 fold)

Summary of sample sizes: 64, 64, 64, 65, 66, 64, ...

Resampling results across tuning parameters:

mtry Accuracy Kappa

2 1 1

8 1 1

14 1 1

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was mtry = 2.

>

>

> # estimate skill of LDA on the validation dataset

> predictions <- predict(fit.adaboost, validation)

> confusionMatrix(predictions, validation$Treatment)

Confusion Matrix and Statistics

Reference

Prediction nonchilled chilled

nonchilled 8 0

chilled 0 8

Accuracy : 1

95% CI : (0.7941, 1)

No Information Rate : 0.5

P-Value [Acc > NIR] : 1.526e-05

Kappa : 1

Mcnemar's Test P-Value : NA

Sensitivity : 1.0

Specificity : 1.0

Pos Pred Value : 1.0

Neg Pred Value : 1.0

Prevalence : 0.5

Detection Rate : 0.5

Detection Prevalence : 0.5

Balanced Accuracy : 1.0

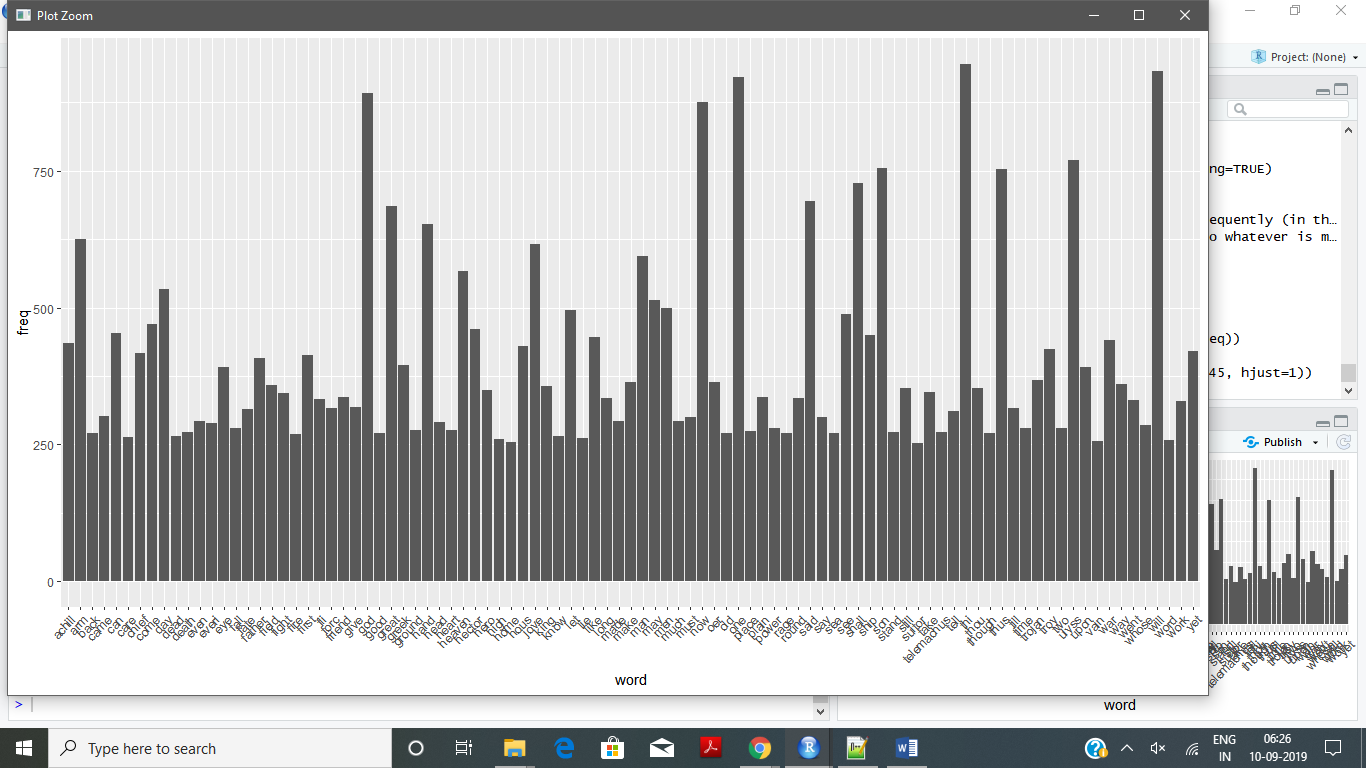
'Positive' Class : nonchilled

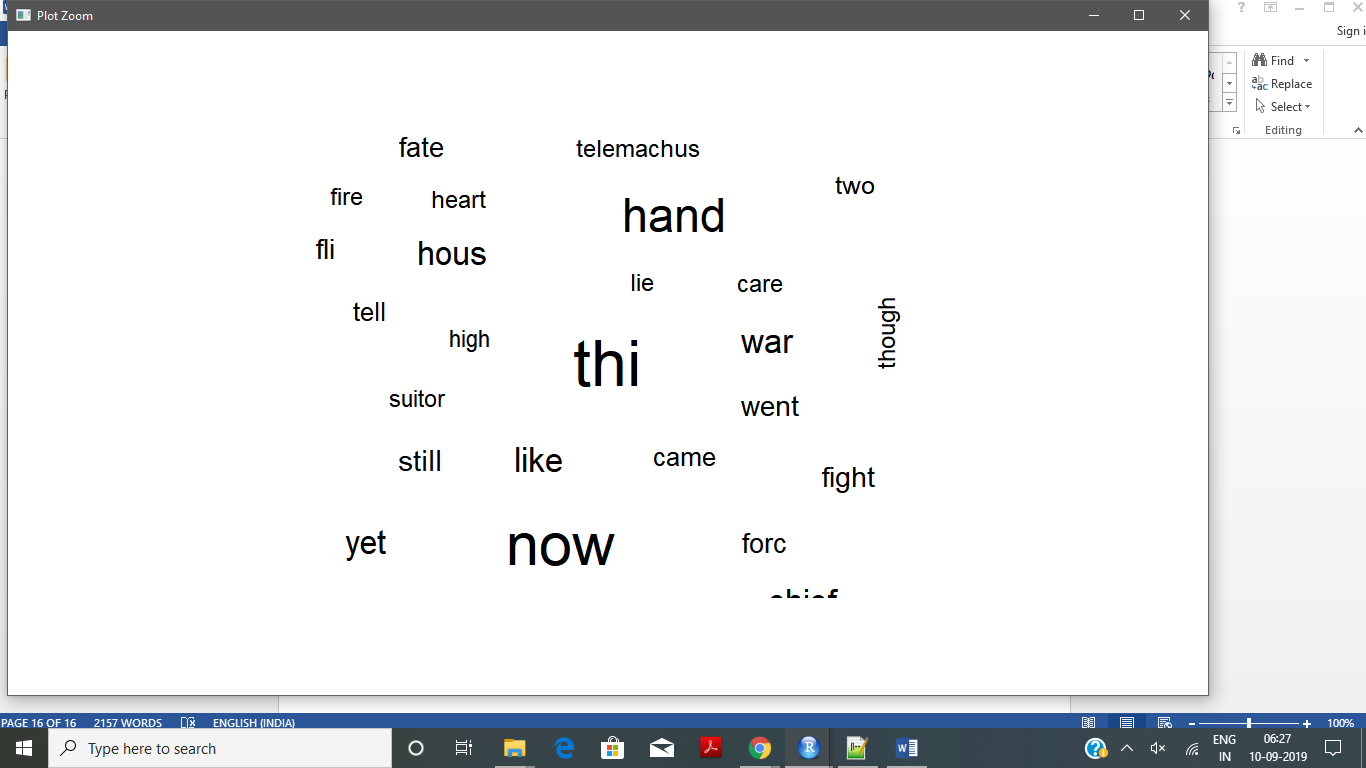
**4-textmining-cloud.R**

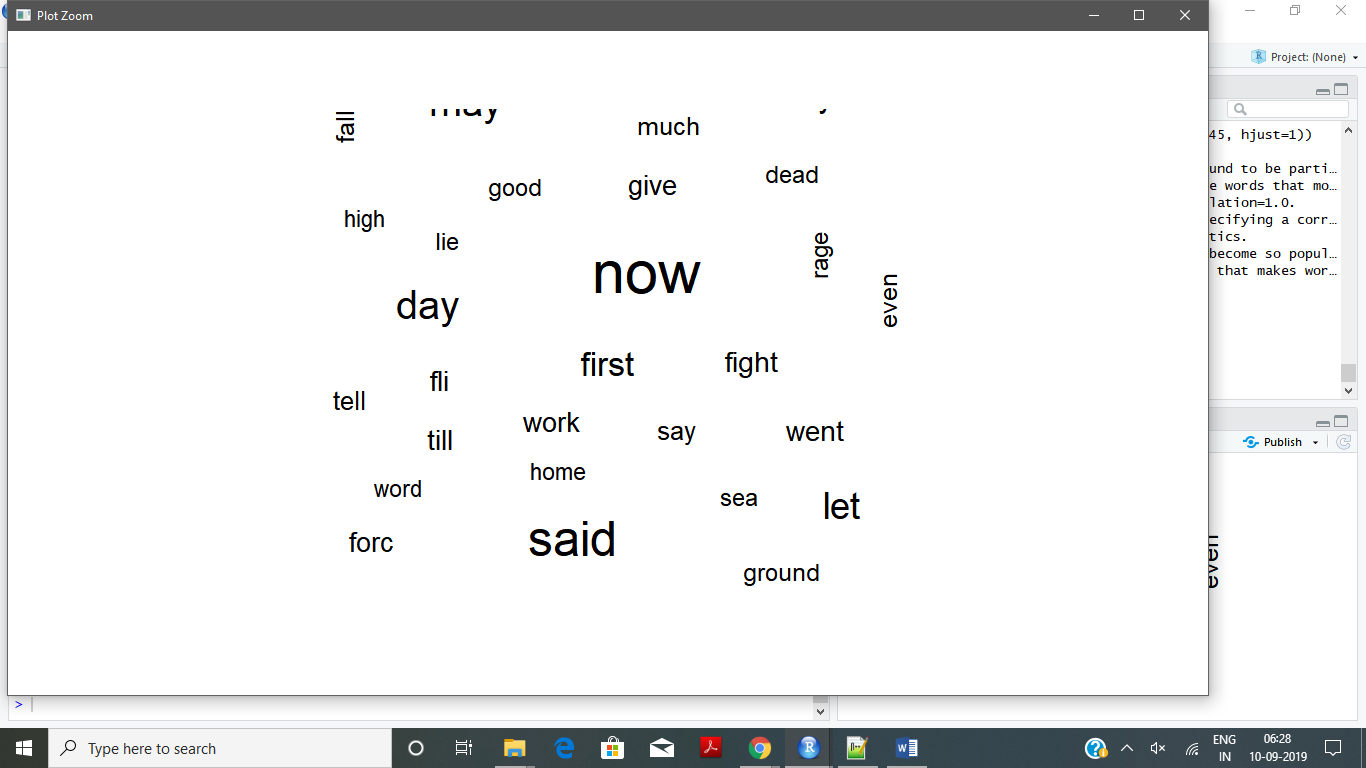
Exercise Code with the Console Output:

Ref 4-textmining-cloud.R

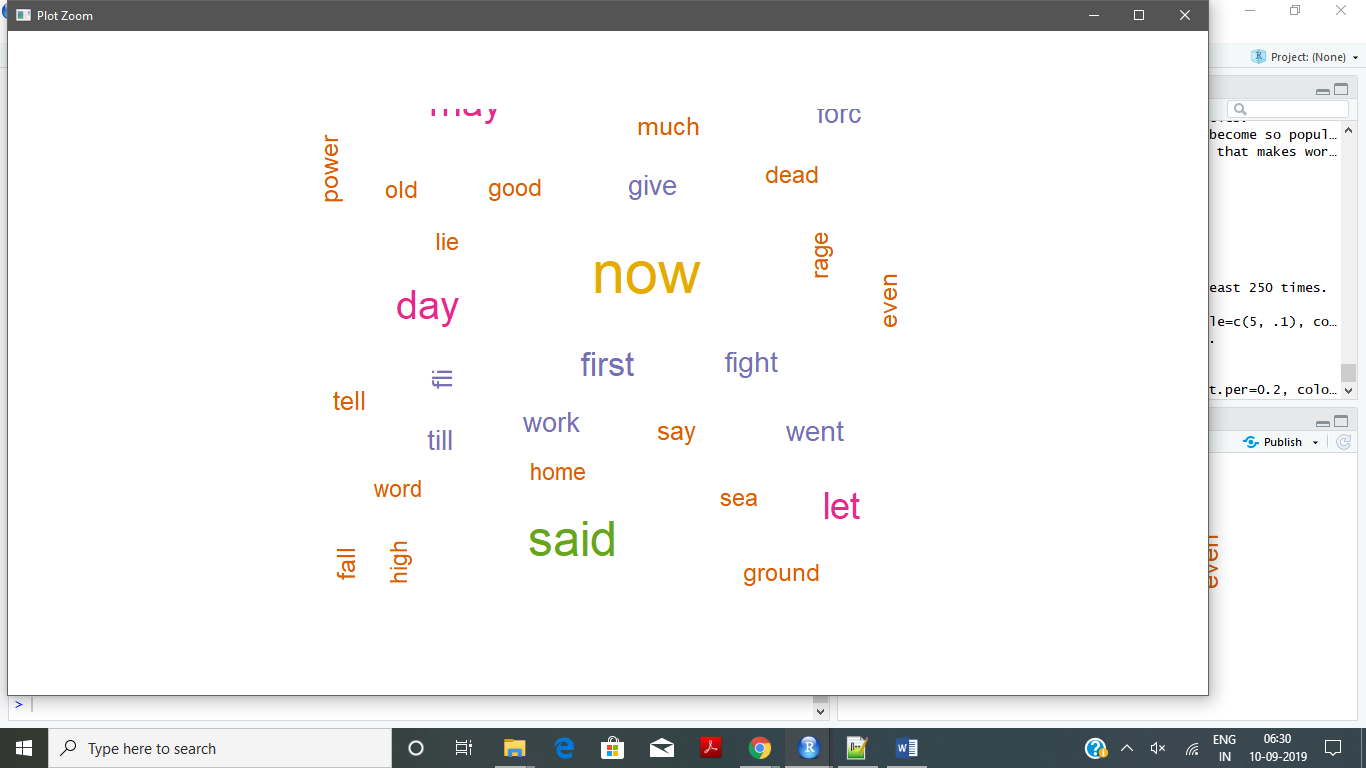
Plots are attached here:

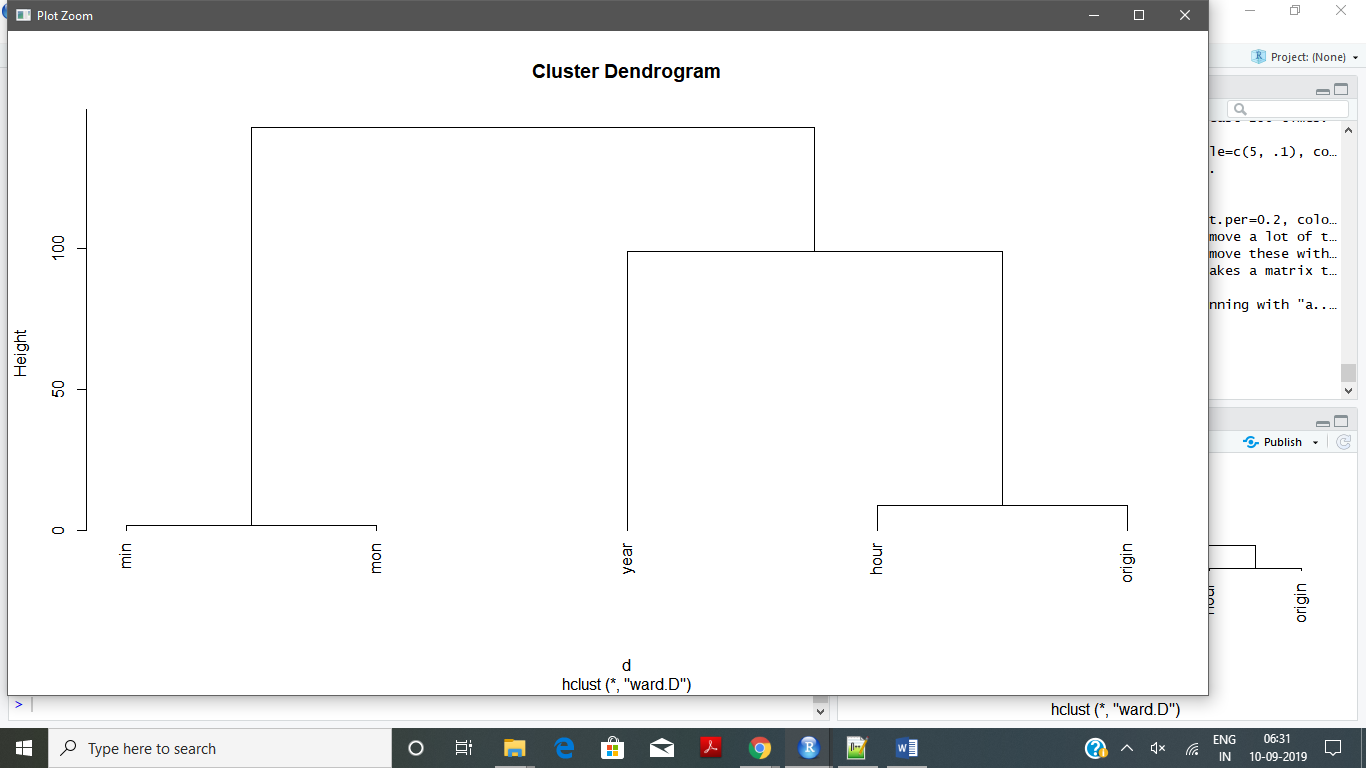


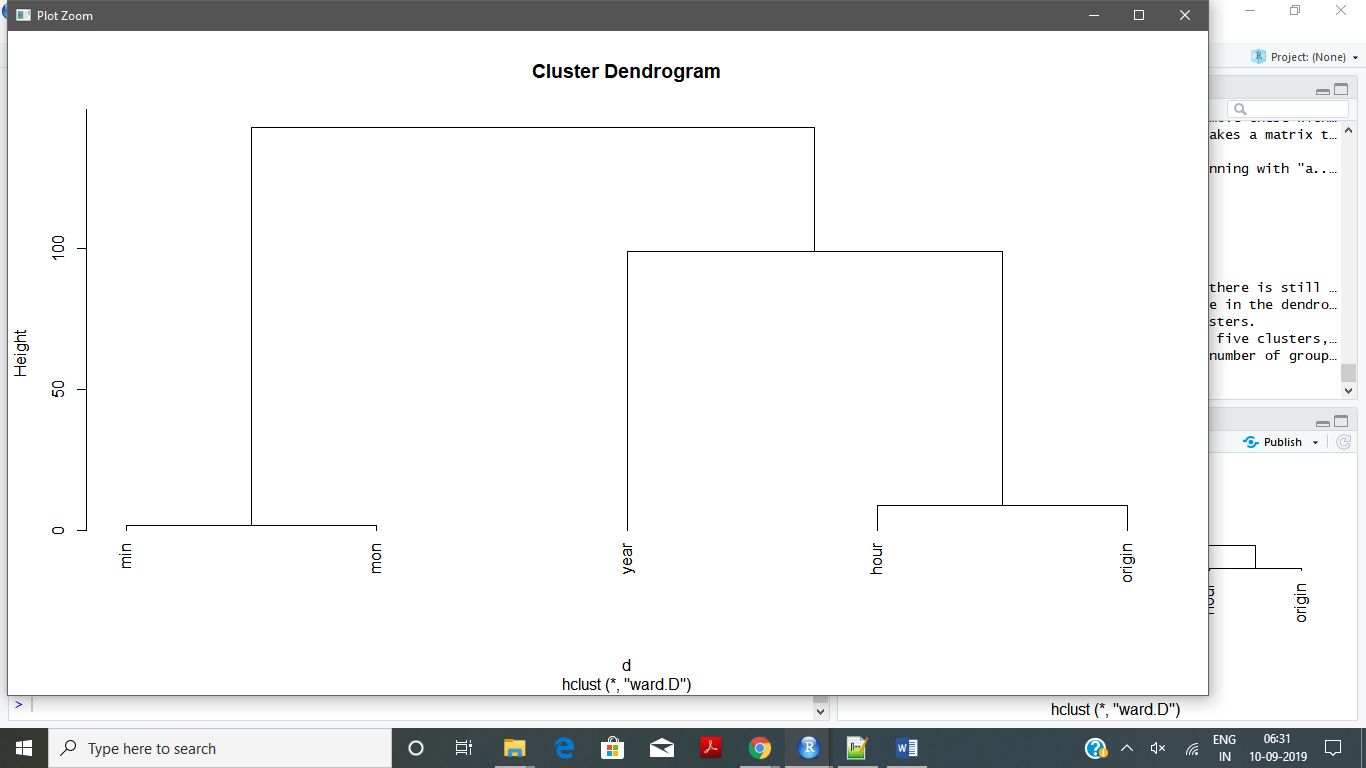


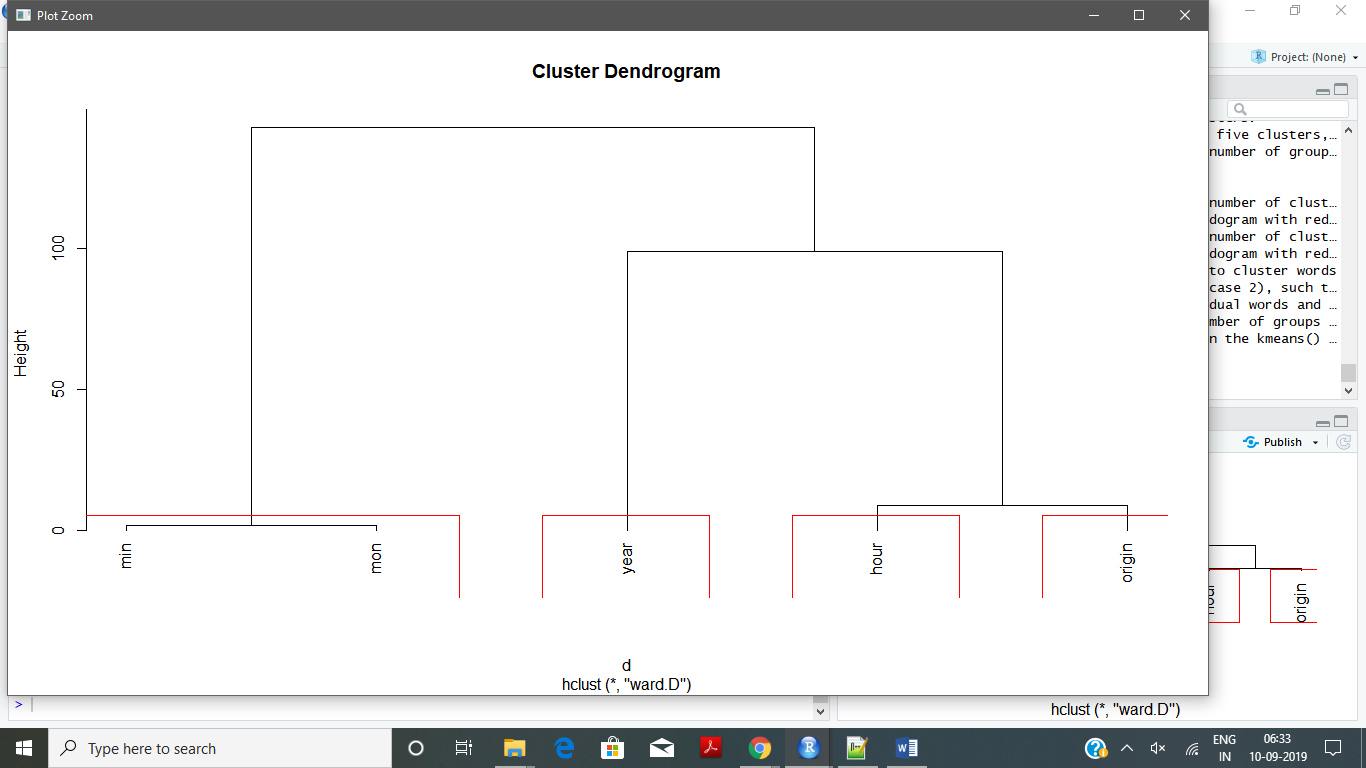












**Textmining-cloud-with-spanish.R**

Spanish Dataset Named Bailando.txt has been used for this textmining-assignment as we were unable to run the same using the hindi dataset.

Execution Code and Console Output:

> ## bnary packages:

> Needed <- c("tm", "SnowballCC", "RColorBrewer", "ggplot2", "wordcloud", "biclust", "cluster", "igraph", "fpc")

> ## On a Mac, save the folder to your desktop and use the following code chunk:

> cname <- file.path("~", "Desktop", "texts")

> cname

[1] "~/Desktop/texts"

> ## Use this to check to see that your texts have loaded.

> dir(cname)

character(0)

> ## On a PC, save the folder to your C: drive and use the following code chunk:

> cname <- file.path("M:/NEU folder/Data Science/Lecture1/RLab6105", "spanish-dataset")

> cname

[1] "M:/NEU folder/Data Science/Lecture1/RLab6105/spanish-dataset"

> dir(cname)

[1] "Bailando.txt"

> ## Load the R package for text mining and then load your texts into R.

> library(tm)

> docs <- Corpus(DirSource(cname))

> summary(docs)

Length Class Mode

Bailando.txt 2 PlainTextDocument list

> ## If you so desire, you can read your documents in the R terminal

> ## using inspect(docs). Or, if you prefer to look at only one of

> ## the documents you loaded, then you can specify which one using something like:

> inspect(docs[1])

<<SimpleCorpus>>

Metadata: corpus specific: 1, document level (indexed): 0

Content: documents: 1

Bailando.txt

Yo te miro, se me corta la respiración\nCuando tú me miras se me sube el corazón\n(Me palpita lento el corazón)\nY en silencio tu mirada dice mil palabras\nLa noche en la que te suplico que no salga el sol\n\n(Bailando, bailando, bailando, bailando)\nTu cuerpo y el mío llenando el vacío\nSubiendo y bajando (subiendo y bajando)\n(Bailando, bailando, bailando, bailando)\nEse fuego por dentro me va enloqueciendo\nMe va saturando\n\nCon tu física y tu química también tu anatomía\nLa cerveza y el tequila y tu boca con la mía\nYa no puedo más (ya no puedo más) Ya no puedo más (ya no puedo más)\nCon esta melodía, tu color, tu fantasía\nCon tu filosofía mi cabeza está vacía\nY ya no puedo más (ya no puedo más)\nYa no puedo más (ya no puedo más)\n\nYo quiero estar contigo, vivir contigo\nBailar contigo, tener contigo\nUna noche loca (una noche loca)\nAy besar tu boca (y besar tu boca)\nYo quiero estar contigo, vivir contigo\nBailar contigo, tener contigo una noche loca\nCon tremenda nota\n(Ooooh, ooooh, ooooh, ooooh)\n\nTú me miras y me llevas a otra dimensión\n\n(Estoy en otra dimensión)\nTus latidos aceleran a mi corazón\n(Tus latidos aceleran a mi corazón)\nQue ironía del destino no poder tocarte\nAbrazarte y sentir la magia de tu olor\n\n(Bailando, bailando, bailando, bailando)\nTú cuerpo y el mío llenando el vacío\nSubiendo y bajando (subiendo y bajando)\n(Bailando, bailando, bailando, bailando)\nEse fuego por dentro me va enloqueciendo\nMe va saturando\n\nCon tu física y tu química también tu anatomía\nLa cerveza y el tequila y tu boca con la mía\nYa no puedo más (ya no puedo más)\nYa no puedo más (ya no puedo más)\nCon esta melodía, tu color, tu fantasía\nCon tu filosofía mi cabeza está vacía\nY ya no puedo más (ya no puedo más)\nYa no puedo más (ya no puedo más)\n\nYo quiero estar contigo, vivir contigo\nBailar contigo, tener contigo\nUna noche loca (una noche loca)\nAy besar tu boca (y besar tu boca)\nYo quiero estar contigo, vivir contigo\nBailar contigo, tener contigo una noche loca\nCon tremenda nota\n\n(Ooooh, ooooh, ooooh, ooooh\nOoooh, ooooh, ooooh, ooooh\nOoooh bailando amor ooooh\nBailando amor ooooh es que se me va el dolor\nOoooh).

> ## Removing punctuation:

> ## Your computer cannot actually read. Punctuation and other special characters only look like more words to your computer and R. Use the following to methods to remove them from your text

> docs <- tm\_map(docs, removePunctuation)

> ## If necessary, such as when working with emails, you can remove special characters.

> ## This list has been customized to remove punctuation that you commonly

> ## find in emails. You can customize what is removed by changing them as you see fit, to meet your own unique needs.

> for(j in seq(docs))

+ {

+ docs[[j]] <- gsub("/", " ", docs[[j]])

+ docs[[j]] <- gsub("@", " ", docs[[j]])

+ docs[[j]] <- gsub("\\|", " ", docs[[j]])

+ }

> ## Removing numbers:

> docs <- tm\_map(docs, removeNumbers)

> ## Converting to lowercase:

> ## As before, we want a word to appear exactly the same every time it appears. We therefore change everything to lowercase.

> docs <- tm\_map(docs, tolower)

> ## Removing "stopwords" (common words) that usually have no analytic value.

> ## In every text, there are a lot of common, and uninteresting words (a, and, also, the, etc.). Such words are frequent by their nature, and will confound your analysis if they remain in the text.

> # For a list of the stopwords, see:

> # length(stopwords("english"))

> # stopwords("english")

> docs <- tm\_map(docs, removeWords, stopwords("english"))

> ## Removing particular words:

> ## If you find that a particular word or two appear in the output, but are not of value to your particular analysis. You can remove them, specifically, from the text.

> docs <- tm\_map(docs, removeWords, c("department", "email"))

> ## Combining words that should stay together

> # If you wish to preserve a concept is only apparent as a collection of two or more words, then you can combine them or reduce them to a meaningful acronym before you begin the analysis.

> ## Here, I am using examples that are particular to Homer, and some that are not ;-)

> for (j in seq(docs))

+ {

+ docs[[j]] <- gsub("Peloponnesian War", "PW", docs[[j]])

+ docs[[j]] <- gsub("World War I", "WWI", docs[[j]])

+ docs[[j]] <- gsub("World War II", "WWII", docs[[j]])

+ }

> ## Removing common word endings (e.g., "ing", "es", "s")

> ## This is referred to as "stemming" documents.

> ## We stem the documents so that a word will be recognizable to the computer,

> ## whether or not it may have a variety of possible endings in the original text.

> library(SnowballC)

> docs <- tm\_map(docs, stemDocument)

> ## Stripping unnecesary whitespace from your documents:

> ## The above preprocessing will leave the documents with a lot of "white space". White space is the result of all the left over spaces that were not removed along with the words that were deleted. The white space can, and should, be removed.

> docs <- tm\_map(docs, stripWhitespace)

> inspect(docs[1]) # Check to see if it worked.

<<SimpleCorpus>>

Metadata: corpus specific: 1, document level (indexed): 0

Content: documents: 1

Bailando.txt

yo te miro se corta la respiración cuando tú mira se sube el corazón palpita lento el corazón y en silencio tu mirada dice mil palabra la noch en la que te suplico que salga el sol bailando bailando bailando bailando tu cuerpo y el mío llenando el vacío subiendo y bajando subiendo y bajando bailando bailando bailando bailando ese fuego por dentro va enloqueciendo va saturando con tu física y tu química también tu anatomía la cerveza y el tequila y tu boca con la mía ya puedo más ya puedo más ya puedo más ya puedo más con esta melodía tu color tu fantasía con tu filosofía mi cabeza está vacía y ya puedo más ya puedo más ya puedo más ya puedo más yo quiero estar contigo vivir contigo bailar contigo tener contigo una noch loca una noch loca ay besar tu boca y besar tu boca yo quiero estar contigo vivir contigo bailar contigo tener contigo una noch loca con tremenda nota ooooh ooooh ooooh ooooh tú mira y lleva otra dimensión estoy en otra dimensión tus latido aceleran mi corazón tus latido aceleran mi corazón que ironía del destino poder tocart abrazart y sentir la magia de tu olor bailando bailando bailando bailando tú cuerpo y el mío llenando el vacío subiendo y bajando subiendo y bajando bailando bailando bailando bailando ese fuego por dentro va enloqueciendo va saturando con tu física y tu química también tu anatomía la cerveza y el tequila y tu boca con la mía ya puedo más ya puedo más ya puedo más ya puedo más con esta melodía tu color tu fantasía con tu filosofía mi cabeza está vacía y ya puedo más ya puedo más ya puedo más ya puedo más yo quiero estar contigo vivir contigo bailar contigo tener contigo una noch loca una noch loca ay besar tu boca y besar tu boca yo quiero estar contigo vivir contigo bailar contigo tener contigo una noch loca con tremenda nota ooooh ooooh ooooh ooooh ooooh ooooh ooooh ooooh ooooh bailando amor ooooh bailando amor ooooh es que se va el dolor ooooh

> docs <- tm\_map(docs, PlainTextDocument)

Warning message:

In tm\_map.SimpleCorpus(docs, PlainTextDocument) :

transformation drops documents

> ## create a document term matrix.

> ## This is what you will be using from this point on.

> dtm <- DocumentTermMatrix(docs[1:2])

> dtm

<<DocumentTermMatrix (documents: 2, terms: 105)>>

Non-/sparse entries: 105/105

Sparsity : 50%

Maximal term length: 17

Weighting : term frequency (tf)

> ## To inspect, you can use: inspect(dtm)

> ## This will, however, fill up your terminal quickly. So you may prefer to view a subset:

> inspect(dtm[1:2, 1:20]) #view first 2 docs & first 20 terms - modify as you like

<<DocumentTermMatrix (documents: 2, terms: 20)>>

Non-/sparse entries: 20/20

Sparsity : 50%

Maximal term length: 8

Weighting : term frequency (tf)

Sample :

Terms

Docs aceleran amor bailando bailar bajando besar boca con contigo corazón

content 2 2 18 4 4 4 6 10 16 4

meta 0 0 0 0 0 0 0 0 0 0

> dim(dtm) #This will display the number of documents & terms (in that order)

[1] 2 105

> ## You'll also need a transpose of this matrix. Create it using:

> tdm <- TermDocumentMatrix(docs[1:2])

> tdm

<<TermDocumentMatrix (terms: 105, documents: 2)>>

Non-/sparse entries: 105/105

Sparsity : 50%

Maximal term length: 17

Weighting : term frequency (tf)

> ## Organize terms by their frequency:

> freq <- colSums(as.matrix(dtm))

> length(freq)

[1] 105

> ord <- order(freq)

> ##Export the matrix to Excel:

> m <- as.matrix(dtm)

> dim(m)

[1] 2 105

> write.csv(m, file="dtm.csv")

> ## Have you ever seen an Excel file with tha tmany columns??

> # Try this instead":

> mt <- as.matrix(tdm)

> dim(mt)

[1] 105 2

> write.csv(mt, file="tdm.csv")

> ## Let's focus on just the interesting stuff.

> # Start by removing sparse terms:

> dtms <- removeSparseTerms(dtm, 0.1) # This makes a matrix that is 10% empty space, maximum.

> inspect(dtms)

<<DocumentTermMatrix (documents: 2, terms: 0)>>

Non-/sparse entries: 0/0

Sparsity : 100%

Maximal term length: 0

Weighting : term frequency (tf)

Sample :

Terms

Docs

content

meta

> ## There are a lot of terms, so for now, just check out some of the most and least frequently occurring words.

> freq[head(ord)]

abrazart corta cuando del destino dice

1 1 1 1 1 1

> freq[tail(ord)]

con contigo más ooooh puedo bailando

10 16 16 16 16 18

> ## Check out the frequency of frequencies.

> head(table(freq), 20)

freq

1 2 3 4 5 6 7 10 16 18

49 34 1 10 1 3 1 1 4 1

> ## The resulting output is two rows of numbers.

> ## The top number is the frequency with which words appear

> ## and the bottom number reflects how many words appear that frequently.

> ## Here, considering only the 20 lowest word frequencies, we can see that

> ## 7407 terms appear only once and 2562 terms appear twice.

> ## There are also a lot of others that appear very infrequently:

> tail(table(freq), 20)

freq

1 2 3 4 5 6 7 10 16 18

49 34 1 10 1 3 1 1 4 1

> ## For a less, fine-grained look at term freqency we can view a table of the terms we selected when we removed sparse terms, above. (Look just under the word "Focus".)

> freq <- colSums(as.matrix(dtms))

> freq

numeric(0)

> ## Or:

> freq <- sort(colSums(as.matrix(dtm)), decreasing=TRUE)

> head(freq, 14)

bailando contigo más ooooh puedo con

18 16 16 16 16 10

noch boca loca una character(0), bailar

7 6 6 6 5 4

bajando besar

4 4

> ## An alternate view of term frequency:

> ## This will identify all terms that appear frequently (in this case, 50 or more times).

> findFreqTerms(dtm, lowfreq=50) # Change "50" to whatever is most appropriate for your text data.

character(0)

> ## another way to do this:

> wf <- data.frame(word=names(freq), freq=freq)

> head(wf)

word freq

bailando bailando 18

contigo contigo 16

más más 16

ooooh ooooh 16

puedo puedo 16

con con 10

> ## Plot words that appear at least 250 times.

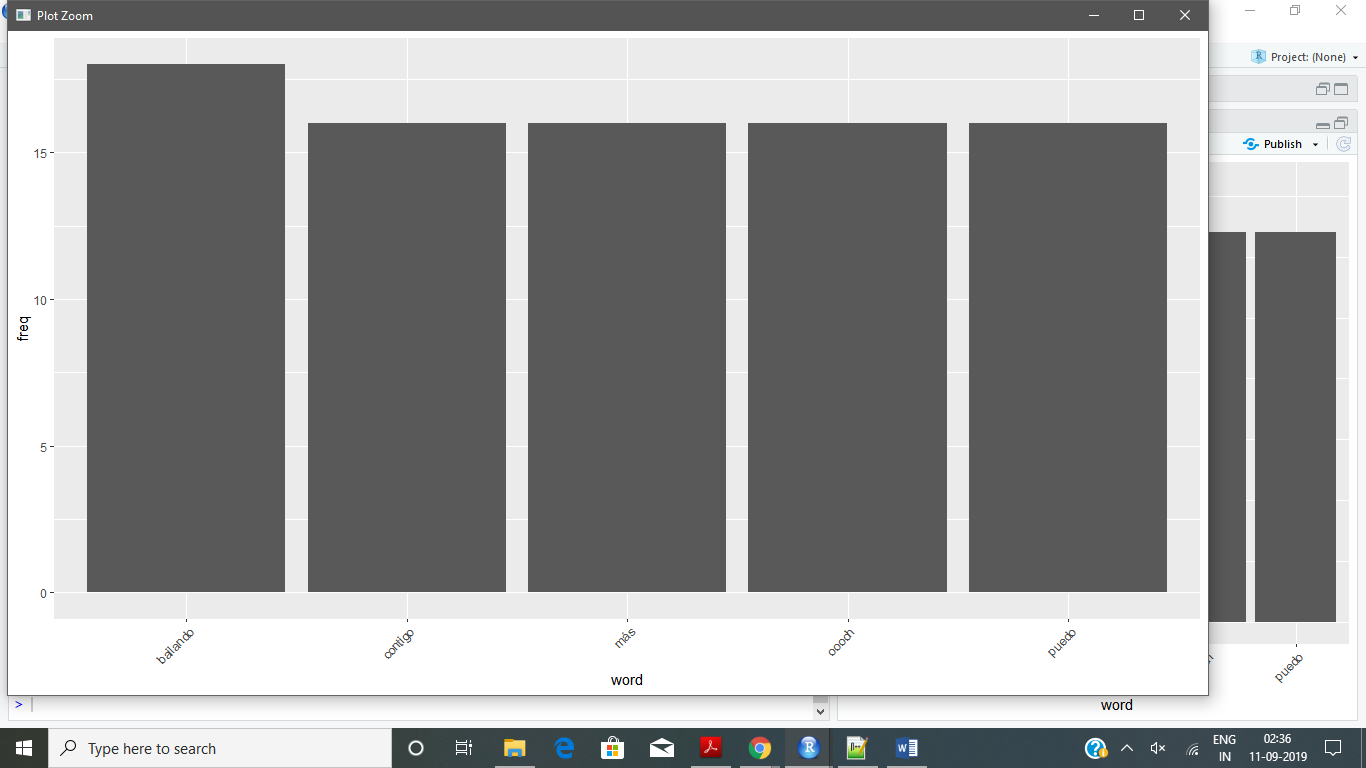
> library(ggplot2)

> p <- ggplot(subset(wf, freq>10), aes(word, freq))

> p <- p + geom\_bar(stat="identity")

> p <- p + theme(axis.text.x=element\_text(angle=45, hjust=1))

> p



> ## If you have a term in mind that you have found to be particularly meaningful to your analysis,

> ## then you may find it helpful to identify the words that most highly correlate with that term.

> ## If words always appear together, then correlation=1.0.

> findAssocs(dtms, "vessel", corlimit=0.98) # specifying a correlation limit of 0.999

$`vessel`

numeric(0)

> ## Humans are generally strong at visual analytics.

> ## That is part of the reason that these have become so popular. What follows are a variety of alternatives for constructing word clouds with your text.

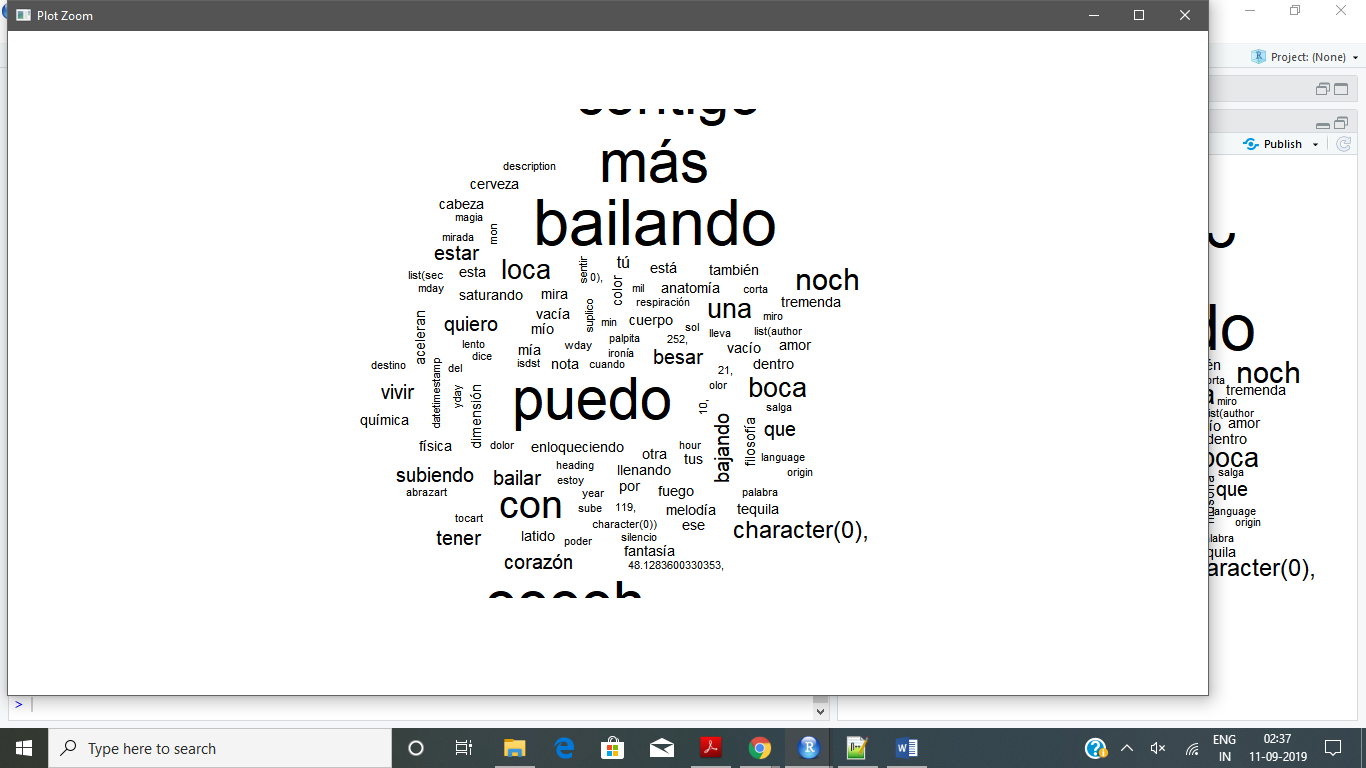
> ## But first you will need to load the package that makes word clouds in R.

> library(wordcloud)

> ##Plot words that occur at least 250 times.

> set.seed(142)

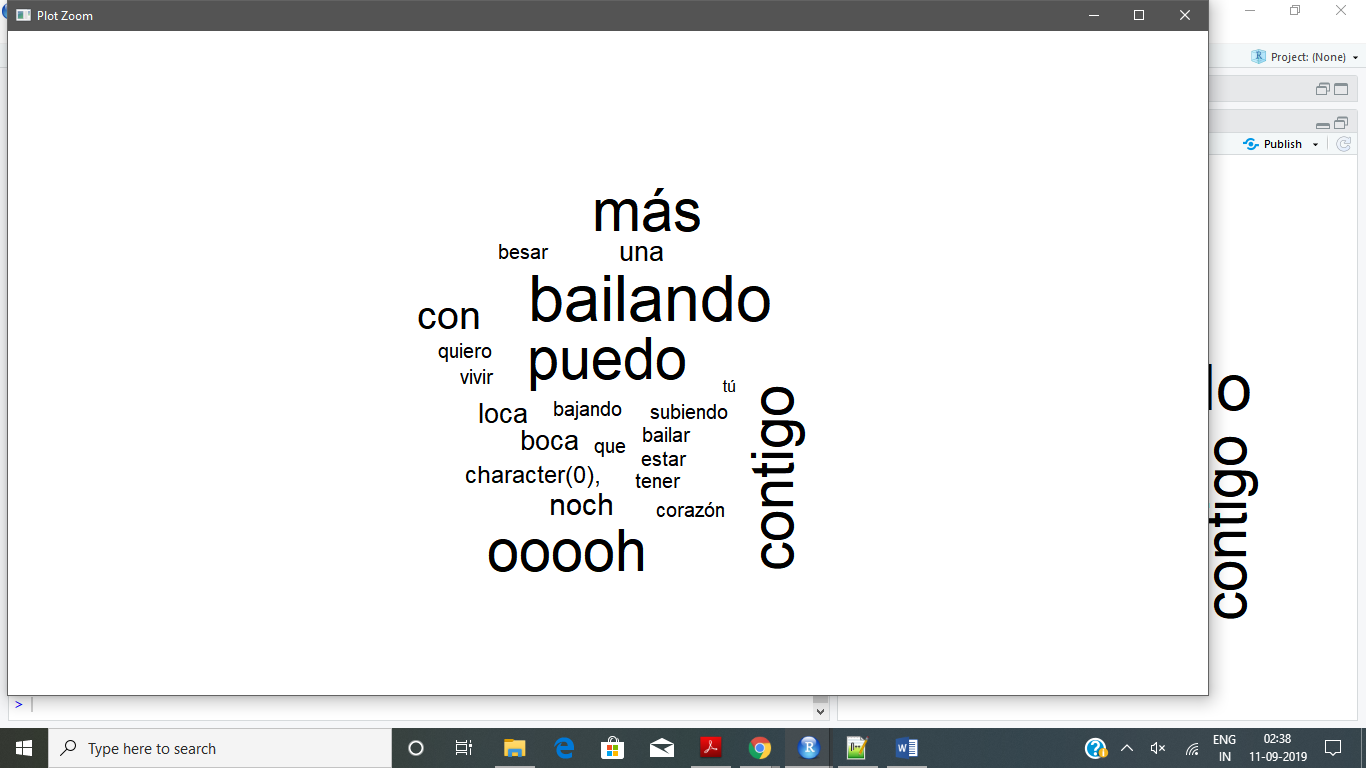
> wordcloud(names(freq), freq, min.freq=200)



> ## Plot the 100 most frequently used words.

> set.seed(142)

> wordcloud(names(freq), freq, max.words=100)



> ##Add some color and plot words occurring at least 250 times.

> set.seed(142)

> wordcloud(names(freq), freq, min.freq=200, scale=c(5, .1), colors=brewer.pal(6, "Dark2"))

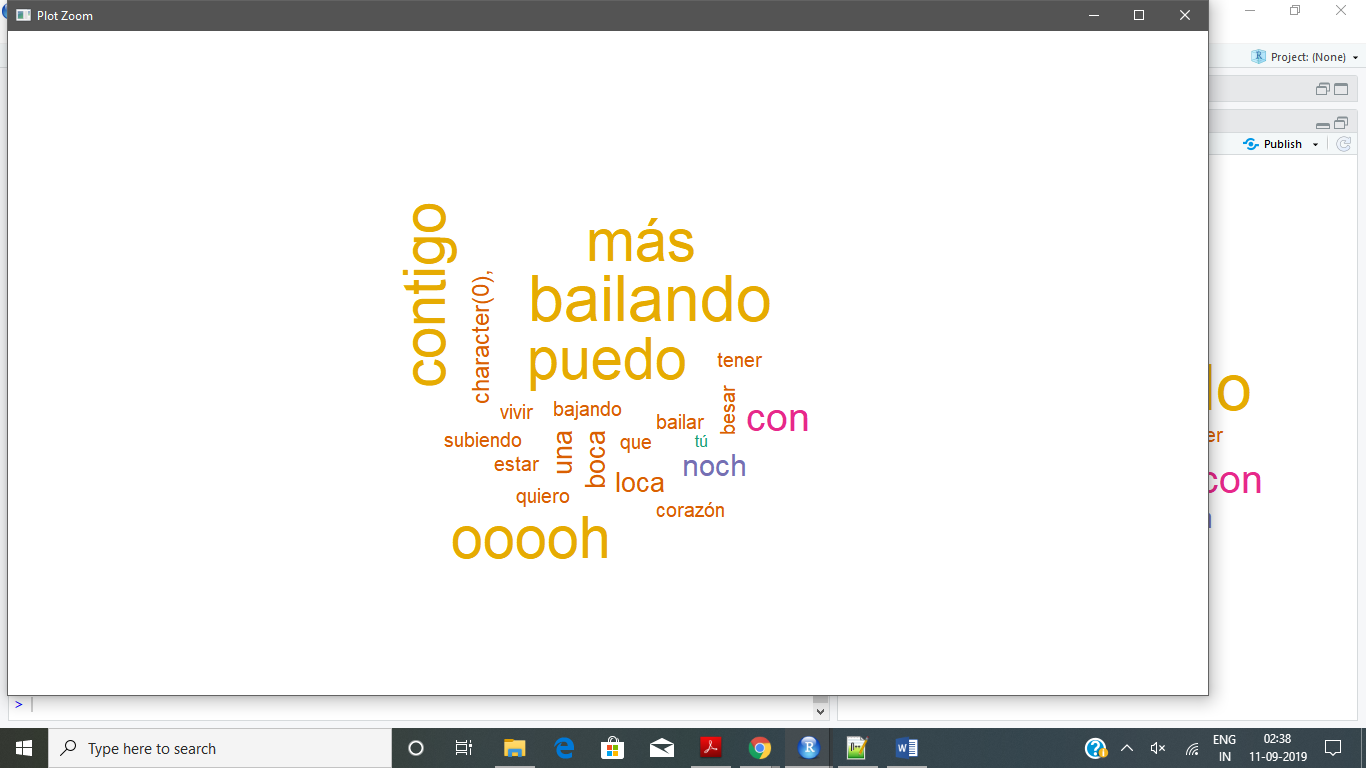


> ##Plot the 100 most frequently occurring words.

> set.seed(142)

> dark2 <- brewer.pal(6, "Dark2")

> wordcloud(names(freq), freq, max.words=100, rot.per=0.2, colors=dark2)



> ## To do this well, you should always first remove a lot of the uninteresting or infrequent words.

> ## If you have not done so already, you can remove these with the following code.

> dtmss <- removeSparseTerms(dtm, 0.90) # This makes a matrix that is only 15% empty space, maximum.

> inspect(dtmss)

<<DocumentTermMatrix (documents: 2, terms: 105)>>

Non-/sparse entries: 105/105

Sparsity : 50%

Maximal term length: 17

Weighting : term frequency (tf)

Sample :

Terms

Docs bailando boca con contigo loca más noch ooooh puedo una

content 18 6 10 16 6 16 7 16 16 6

meta 0 0 0 0 0 0 0 0 0 0

> ## First calculate distance between words beginning with "a..." & then cluster them according to similarity.

> library(cluster)

> d <- dist(t(dtmss[,1:5]), method="euclidian")

> fit <- hclust(d=d, method="ward")

The "ward" method has been renamed to "ward.D"; note new "ward.D2"

> fit

Call:

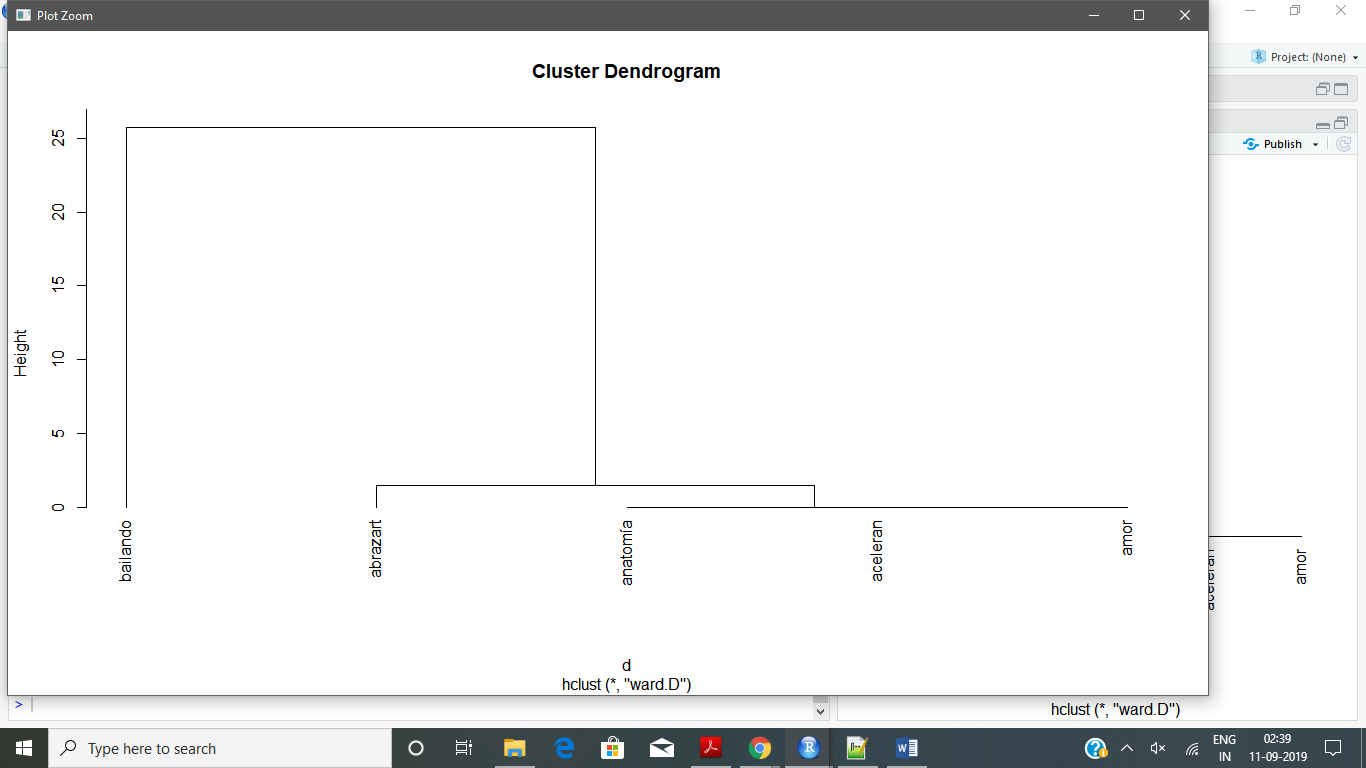
hclust(d = d, method = "ward")

Cluster method : ward.D

Distance : euclidean

Number of objects: 5

> plot(fit, hang=-1)



> ## Helping to Read a Dendrogram

> ## If you find dendrograms difficult to read, there is still hope:

> ## To get a better idea of where the groups are in the dendrogram,

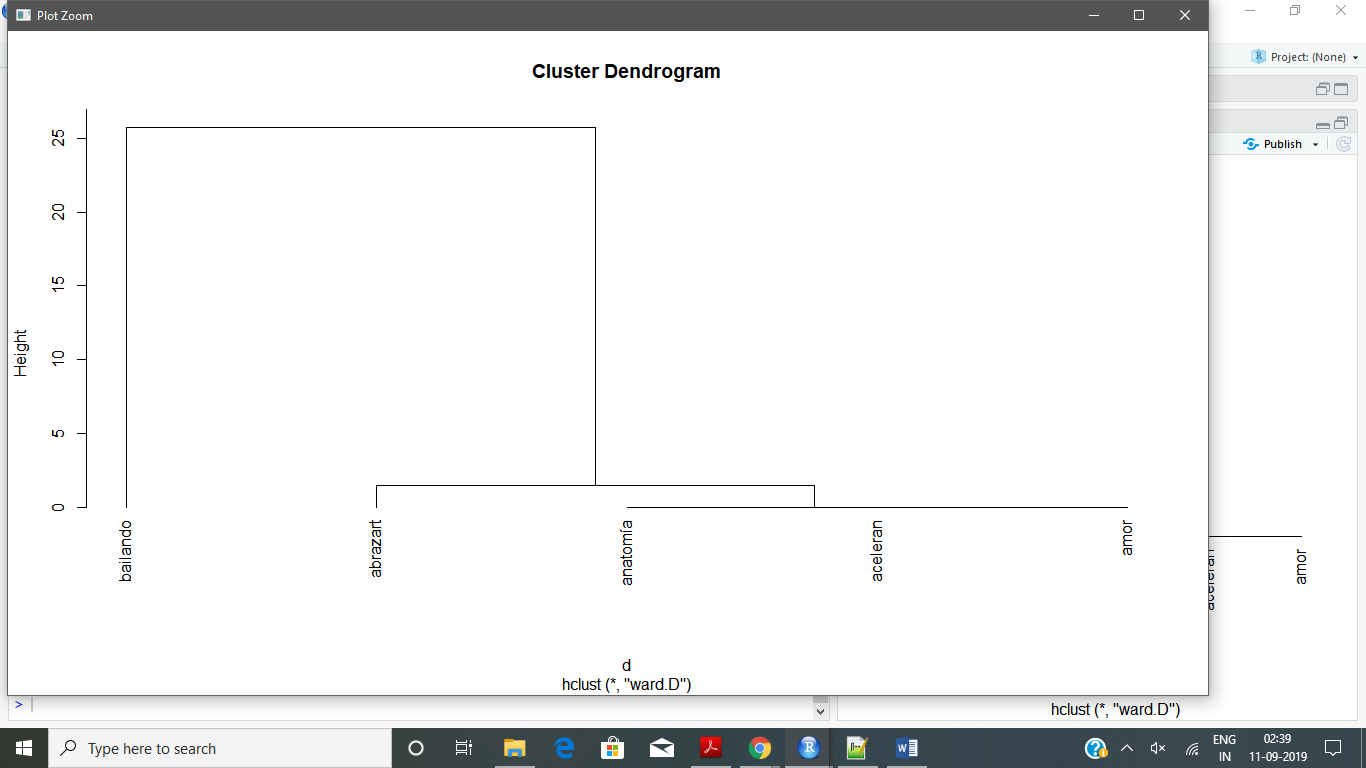
> ## you can also ask R to help identify the clusters.

> ## Here, we have arbitrarily chosen to look at five clusters, as indicated by the red boxes.

> ## If you would like to highlight a different number of groups, then feel free to change the code accordingly

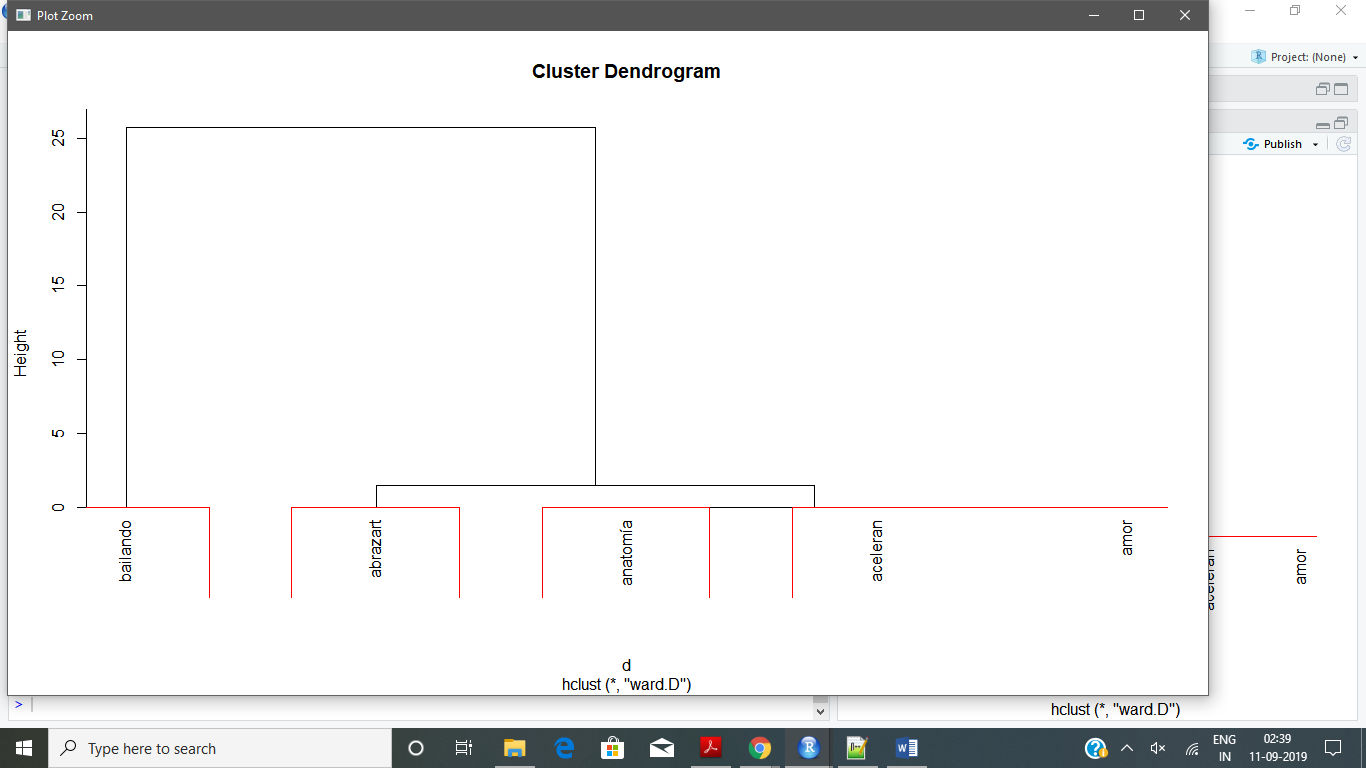
> plot.new()

> plot(fit, hang=-1)



> groups <- cutree(fit, k=4) # "k=" defines the number of clusters you are using

> rect.hclust(fit, k=4, border="red") # draw dendogram with red borders around the 5 clusters



> ## The k-means clustering method will attempt to cluster words

> ## into a specified number of groups (in this case 2), such that

> ## the sum of squared distances between individual words and one

> ## of the group centers. You can change the number of groups you

> ## seek by changing the number specified within the kmeans() command.

> ## Can you create meaningful clusters?

> library(fpc)

> d <- dist(t(dtmss), method="euclidian")

> kfit <- kmeans(d, 2)

> clusplot(as.matrix(d), kfit$cluster, color=T, shade=T, labels=2, lines=0)

>

