For this assignment, do the following:

1. Connect the Facebook API with R (either with the "permanent" 2-day token or the temporary token as outlined in the weekly readings)

R Code:   
  
install.packages("devtools")

install.packages("Rfacebook")

install.packages("Rook")

library(devtools)

install\_github("pablobarbera/Rfacebook/Rfacebook")

require("Rfacebook")

fb\_appid <- '1675043216133450'

fb\_appsec <- '5407d86646a2525720beec8f42affb5c'

fbOAuth(fb\_appid,fb\_appsec, extended\_permissions = TRUE, legacy\_permissions = FALSE)

fb\_oauth <- fbOAuth(fb\_appid,fb\_appsec, extended\_permissions = TRUE, legacy\_permissions = FALSE)

save(fb\_oauth, file="~/R/fb\_oauth")

load("~/R/fb\_oauth")

1. Import into R 100 records from EITHER a public Facebook account (such as a newspaper site) OR your own profile

## Saving my own public information in the variable "me".

library(V8)

library(RCurl)

library(RJSONIO)

library(rjson)

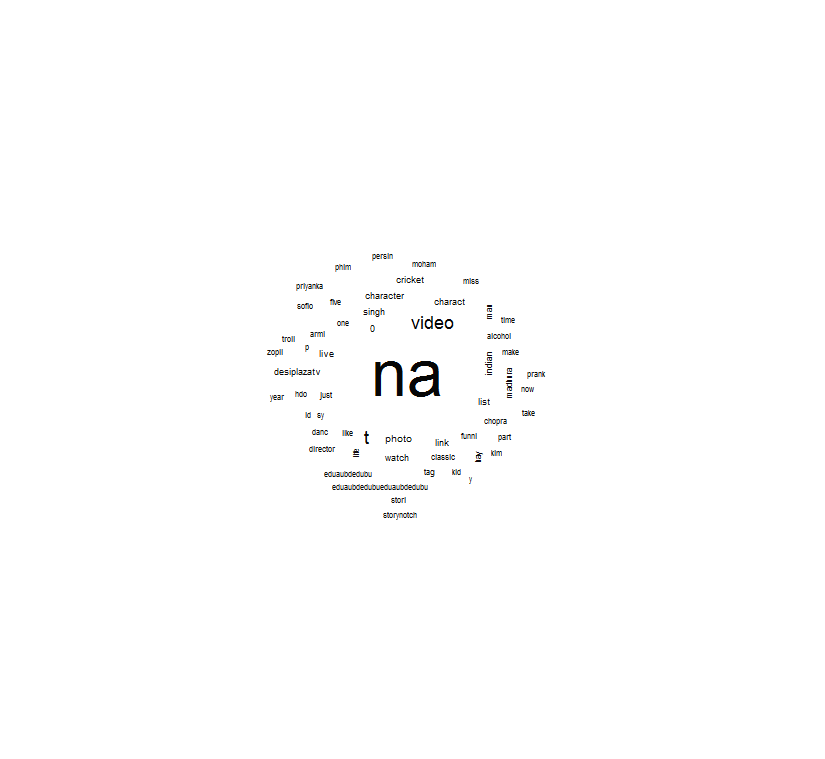
SriharshaGoteti <- getUsers("Sriharsha Goteti",token=fb\_oauth)

SriharshaGoteti

my\_likes <- getLikes(user="Sriharsha Goteti", token=fb\_oauth)

my\_likes

my\_friends <- getFriends(token, simplify = FALSE)

my\_friends  
  
  


1. Produce a corpus and apply the necessary text transformations (including stopwords, URL removal, etc.)

library(Rfacebook)

token <- "EAACEdEose0cBAIAcnz0MePuBnGBFQBcOGgDHltSQoG4wAbAzasprUIkw9ORoRvSJkbskwMMMz3tZBFKOSTvtrW0qYBZC1sjkEubjPH0U3ZCLMlIPopZA8uI2ZC0UGLyUZCK6d5HZASb8RJAfEeGUILlhKLAyCLoUU1qeKd0HyQvmu8RCK5FRUxw"

me <- getUsers("me", token, private\_info = TRUE)

me$name # my name

## my\_newsfeed <- getNewsfeed(token=fb\_oauth, n=100) ## need to debug; throws API permission error

my\_newsfeed <- getNewsfeed(token, n=100) ##

post <- getPost(10205469233992080,token=token, n = 1000, likes = TRUE, comments = TRUE)

str(me)

str(my\_newsfeed)

library(tm)

myCorpus <- Corpus(VectorSource(my\_newsfeed))

myCorpus1 <- Corpus(VectorSource(my\_newsfeed))

library(wordcloud)

m <- as.matrix(me)

## convert to lower case

myCorpus <- tm\_map(myCorpus, tolower)

## remove punctuation

myCorpus <- tm\_map(myCorpus, removePunctuation)

## remove numbers

myCorpus <- tm\_map(myCorpus, removeNumbers)

## remove URLs

removeURL <- function(x) gsub("http[[:alnum:]]\*", "", x)

myCorpus <- tm\_map(myCorpus, removeURL)

## add two extra stop words: "available" and "via"

myStopwords <- c(stopwords('english'), "available", "via")

## remove "r" and "big" from stopwords

myStopwords <- setdiff(myStopwords, c("r", "big"))

## remove stopwords from corpus

myCorpus <- tm\_map(myCorpus, removeWords, myStopwords)

## keep a copy of corpus to use later as a dictionary for stem completion

myCorpusCopy <- myCorpus

myCorpus <- tm\_map(myCorpus, stemDocument)

## inspect documents (tweets) numbered 11 to 15

inspect(myCorpus[11:15])

myCorpus <- tm\_map(myCorpus, PlainTextDocument)

myCorpus <- Corpus(VectorSource(myCorpus))

myTdm <- TermDocumentMatrix(myCorpus, control=list(wordLengths=c(1,Inf)))

## myTdm <- TermDocumentMatrix(myCorpus)

myTdm

idx <- which(dimnames(myTdm)$Terms == "r")

inspect(myTdm[idx+(0:5),101:110])

layout(matrix(c(1,2),2,1)) # set to two graphs per page

plot(pamResult, color=F, labels=4, lines=0, cex=.8, col.clus=1, col.p=pamResult$clustering)

layout(matrix(1)) # change back to one graph per page

library(wordcloud)

m <- as.matrix(myTdm)

wordFreq <- sort(rowSums(m), decreasing=TRUE)

set.seed(375) # to make it reproducible

grayLevels <- gray( (wordFreq+10) / (max(wordFreq)+10) )

wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=3, random.order=F, colors=grayLevels)

1. Produce an appropriate number of clusters to show semantic fields
2. Produce AT A MINIMUM a basic network graph.

## load termDocMatrix

termDocMatrix <- wordFreq

## inspect part of the matrix

termDocMatrix[1:10,1:20]

## change it to a Boolean matrix

termDocMatrix[termDocMatrix>=1] <- 1

## transform into a term-term adjacency matrix

termMatrix <- termDocMatrix %\*% t(termDocMatrix)

## inspect terms numbered 5 to 10

termMatrix[5:10,5:10]

## Now we have built a term-term adjacency matrix, where the rows and columns represent

## terms, and every entry is the number of concurrences of two terms. Next we can build a graph

## with graph.adjacency() from package igraph

library(igraph)

library(fpc)

## build a graph from the above matrix

g <- graph.adjacency(termMatrix, weighted=T, mode="undirected")

\*\*\*Graph Modulation\*\*\*\*

V(g)$label.cex <- 2.2 \* V(g)$degree / max(V(g)$degree)+ .2

V(g)$label.color <- rgb(0, 0, .2, .8)

V(g)$frame.color <- NA

egam <- (log(E(g)$weight)+.4) / max(log(E(g)$weight)+.4)

E(g)$color <- rgb(.5, .5, 0, egam)

E(g)$width <- egam

## plot the graph in layout1

plot(g, layout=layout1)

plot(g)

##### Network of Tweets #####

## We can also build a graph of tweets base on the number of terms that they have in common.

## Because most tweets contain one or more words from \r", \data" and \mining", most tweets

## are connected with others and the graph of tweets is very crowded. To simplify the graph

## and find relationships between tweets beyond the above three keywords, we remove the three words before building a graph.

## remove "r", "data" and "mining"

idx <- which(dimnames(termDocMatrix)$Terms %in% c("r", "data", "mining"))

M <- termDocMatrix[-idx,]

## build a tweet-tweet adjacency matrix

tweetMatrix <- t(M) %\*% M

library(igraph)

g <- graph.adjacency(tweetMatrix, weighted=T, mode = "undirected")

V(g)$degree <- degree(g)

g <- simplify(g)

## set labels of vertices to tweet IDs

V(g)$label <- V(g)$name

V(g)$label.cex <- 1

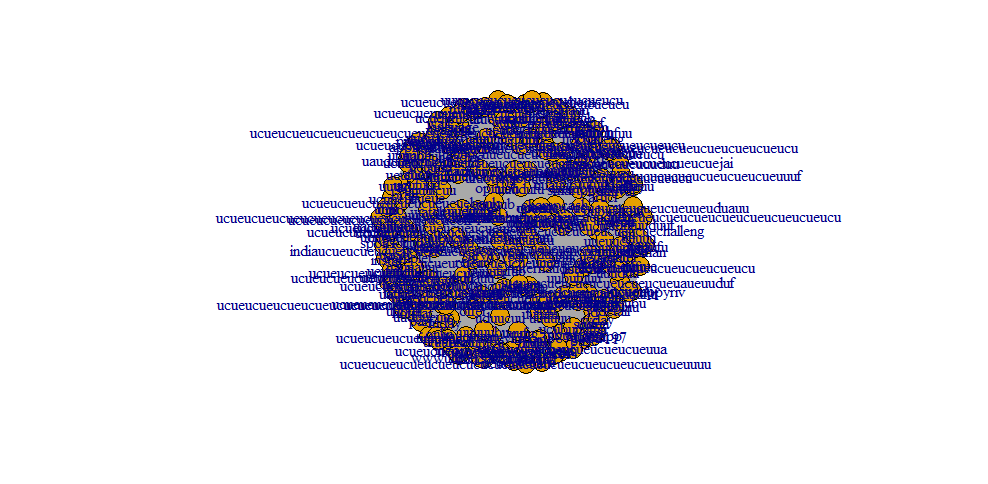
V(g)$label.color <- rgb(.4, 0, 0, .7)

V(g)$size <- 2

V(g)$frame.color <- NA

pdf("tweet-network.pdf")

barplot(table(V(g)$degree))

dev.off()  
  
  
  
  


**Total Code:**

install.packages("devtools")

install.packages("Rfacebook")

install.packages("Rook")

library(devtools)

install\_github("pablobarbera/Rfacebook/Rfacebook")

require("Rfacebook")

fb\_appid <- '1675043216133450'

fb\_appsec <- '5407d86646a2525720beec8f42affb5c'

fbOAuth(fb\_appid,fb\_appsec, extended\_permissions = TRUE, legacy\_permissions = FALSE)

fb\_oauth <- fbOAuth(fb\_appid,fb\_appsec, extended\_permissions = TRUE, legacy\_permissions = FALSE)

save(fb\_oauth, file="~/R/fb\_oauth")

load("~/R/fb\_oauth")

################## Let's look around Facebook! ########################

## Saving my own public information in the variable "me".

library(V8)

library(RCurl)

library(RJSONIO)

library(rjson)

SriharshaGoteti <- getUsers("Sriharsha Goteti",token=fb\_oauth)

SriharshaGoteti

my\_likes <- getLikes(user="Sriharsha Goteti", token=fb\_oauth)

my\_likes

my\_friends <- getFriends(token, simplify = FALSE)

my\_friends

################## The remainder didn't work for me with fb\_oauth

################## Generated temporary token at https://developers.facebook.com/tools/explorer

library(Rfacebook)

token <- "EAACEdEose0cBAIAcnz0MePuBnGBFQBcOGgDHltSQoG4wAbAzasprUIkw9ORoRvSJkbskwMMMz3tZBFKOSTvtrW0qYBZC1sjkEubjPH0U3ZCLMlIPopZA8uI2ZC0UGLyUZCK6d5HZASb8RJAfEeGUILlhKLAyCLoUU1qeKd0HyQvmu8RCK5FRUxw"

me <- getUsers("me", token, private\_info = TRUE)

me$name # my name

## my\_newsfeed <- getNewsfeed(token=fb\_oauth, n=100) ## need to debug; throws API permission error

my\_newsfeed <- getNewsfeed(token, n=100) ##

post <- getPost(10205469233992080,token=token, n = 1000, likes = TRUE, comments = TRUE)

str(me)

str(my\_newsfeed)

library(tm)

myCorpus <- Corpus(VectorSource(my\_newsfeed))

myCorpus1 <- Corpus(VectorSource(my\_newsfeed))

library(wordcloud)

m <- as.matrix(me)

## convert to lower case

myCorpus <- tm\_map(myCorpus, tolower)

## remove punctuation

myCorpus <- tm\_map(myCorpus, removePunctuation)

## remove numbers

myCorpus <- tm\_map(myCorpus, removeNumbers)

## remove URLs

removeURL <- function(x) gsub("http[[:alnum:]]\*", "", x)

myCorpus <- tm\_map(myCorpus, removeURL)

## add two extra stop words: "available" and "via"

myStopwords <- c(stopwords('english'), "available", "via")

## remove "r" and "big" from stopwords

myStopwords <- setdiff(myStopwords, c("r", "big"))

## remove stopwords from corpus

myCorpus <- tm\_map(myCorpus, removeWords, myStopwords)

## keep a copy of corpus to use later as a dictionary for stem completion

myCorpusCopy <- myCorpus

myCorpus <- tm\_map(myCorpus, stemDocument)

## inspect documents (tweets) numbered 11 to 15

inspect(myCorpus[11:15])

myCorpus <- tm\_map(myCorpus, PlainTextDocument)

myCorpus <- Corpus(VectorSource(myCorpus))

myTdm <- TermDocumentMatrix(myCorpus, control=list(wordLengths=c(1,Inf)))

## myTdm <- TermDocumentMatrix(myCorpus)

myTdm

idx <- which(dimnames(myTdm)$Terms == "r")

inspect(myTdm[idx+(0:5),101:110])

layout(matrix(c(1,2),2,1)) # set to two graphs per page

plot(pamResult, color=F, labels=4, lines=0, cex=.8, col.clus=1, col.p=pamResult$clustering)

layout(matrix(1)) # change back to one graph per page

library(wordcloud)

m <- as.matrix(myTdm)

wordFreq <- sort(rowSums(m), decreasing=TRUE)

set.seed(375) # to make it reproducible

grayLevels <- gray( (wordFreq+10) / (max(wordFreq)+10) )

wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=3, random.order=F, colors=grayLevels)

##Network Graph####

## load termDocMatrix

termDocMatrix <- wordFreq

## inspect part of the matrix

termDocMatrix[1:10,1:20]

## change it to a Boolean matrix

termDocMatrix[termDocMatrix>=1] <- 1

## transform into a term-term adjacency matrix

termMatrix <- termDocMatrix %\*% t(termDocMatrix)

## inspect terms numbered 5 to 10

termMatrix[5:10,5:10]

## Now we have built a term-term adjacency matrix, where the rows and columns represent

## terms, and every entry is the number of concurrences of two terms. Next we can build a graph

## with graph.adjacency() from package igraph

library(igraph)

library(fpc)

## build a graph from the above matrix

g <- graph.adjacency(termMatrix, weighted=T, mode="undirected")

\*\*\*Graph Modulation\*\*\*\*

V(g)$label.cex <- 2.2 \* V(g)$degree / max(V(g)$degree)+ .2

V(g)$label.color <- rgb(0, 0, .2, .8)

V(g)$frame.color <- NA

egam <- (log(E(g)$weight)+.4) / max(log(E(g)$weight)+.4)

E(g)$color <- rgb(.5, .5, 0, egam)

E(g)$width <- egam

## plot the graph in layout1

plot(g, layout=layout1)

plot(g)

##### Network of Tweets #####

## We can also build a graph of tweets base on the number of terms that they have in common.

## Because most tweets contain one or more words from \r", \data" and \mining", most tweets

## are connected with others and the graph of tweets is very crowded. To simplify the graph

## and find relationships between tweets beyond the above three keywords, we remove the three words before building a graph.

## remove "r", "data" and "mining"

idx <- which(dimnames(termDocMatrix)$Terms %in% c("r", "data", "mining"))

M <- termDocMatrix[-idx,]

## build a tweet-tweet adjacency matrix

tweetMatrix <- t(M) %\*% M

library(igraph)

g <- graph.adjacency(tweetMatrix, weighted=T, mode = "undirected")

V(g)$degree <- degree(g)

g <- simplify(g)

## set labels of vertices to tweet IDs

V(g)$label <- V(g)$name

V(g)$label.cex <- 1

V(g)$label.color <- rgb(.4, 0, 0, .7)

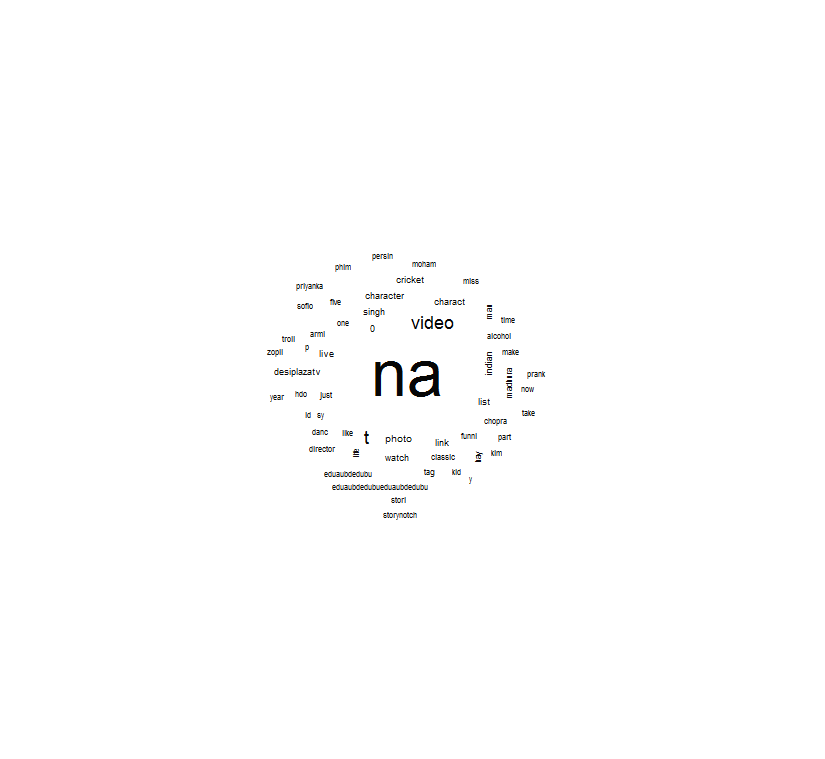
V(g)$size <- 2

V(g)$frame.color <- NA

pdf("tweet-network.pdf")

barplot(table(V(g)$degree))

dev.off()

  
  
  
  
  
Output:

|  |
| --- |
| > install.packages("devtools")  Installing package into ‘C:/Users/gotet/OneDrive/Documents/R/win-library/3.3’  (as ‘lib’ is unspecified)  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.3/devtools\_1.12.0.zip'  Content type 'application/zip' length 432279 bytes (422 KB)  downloaded 422 KB  package ‘devtools’ successfully unpacked and MD5 sums checked  The downloaded binary packages are in  C:\Users\gotet\AppData\Local\Temp\RtmpoVq43g\downloaded\_packages  > install.packages("Rfacebook")  Installing package into ‘C:/Users/gotet/OneDrive/Documents/R/win-library/3.3’  (as ‘lib’ is unspecified)  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.3/Rfacebook\_0.6.11.zip'  Content type 'application/zip' length 86102 bytes (84 KB)  downloaded 84 KB  package ‘Rfacebook’ successfully unpacked and MD5 sums checked  The downloaded binary packages are in  C:\Users\gotet\AppData\Local\Temp\RtmpoVq43g\downloaded\_packages  > install.packages("Rook")  Installing package into ‘C:/Users/gotet/OneDrive/Documents/R/win-library/3.3’  (as ‘lib’ is unspecified)  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.3/Rook\_1.1-1.zip'  Content type 'application/zip' length 280127 bytes (273 KB)  downloaded 273 KB  package ‘Rook’ successfully unpacked and MD5 sums checked  The downloaded binary packages are in  C:\Users\gotet\AppData\Local\Temp\RtmpoVq43g\downloaded\_packages  > library(devtools)  Warning message:  package ‘devtools’ was built under R version 3.3.3  > install\_github("pablobarbera/Rfacebook/Rfacebook")  Downloading GitHub repo pablobarbera/Rfacebook@master  from URL https://api.github.com/repos/pablobarbera/Rfacebook/zipball/master  Installing Rfacebook  "C:/PROGRA~1/R/R-33~1.2/bin/x64/R" --no-site-file --no-environ --no-save --no-restore --quiet CMD INSTALL \  "C:/Users/gotet/AppData/Local/Temp/RtmpoVq43g/devtools38603dd4789e/pablobarbera-Rfacebook-800cf34/Rfacebook" \  --library="C:/Users/gotet/OneDrive/Documents/R/win-library/3.3" --install-tests  \* installing \*source\* package 'Rfacebook' ...  \*\* R  \*\* preparing package for lazy loading  Warning: package 'httpuv' was built under R version 3.3.3  \*\* help  \*\*\* installing help indices  \*\* building package indices  \*\* testing if installed package can be loaded  \*\*\* arch - i386  Warning: package 'httpuv' was built under R version 3.3.3  \*\*\* arch - x64  Warning: package 'httpuv' was built under R version 3.3.3  \* DONE (Rfacebook)  > require("Rfacebook")  Loading required package: Rfacebook  Loading required package: httr  Loading required package: rjson  Loading required package: httpuv  Attaching package: ‘Rfacebook’  The following object is masked from ‘package:methods’:  getGroup  Warning message:  package ‘httpuv’ was built under R version 3.3.3  > fb\_appid <- '1675043216133450'  > fb\_appsec <- '5407d86646a2525720beec8f42affb5c'  > fbOAuth(fb\_appid,fb\_appsec, extended\_permissions = TRUE, legacy\_permissions = FALSE)  Copy and paste into Site URL on Facebook App Settings: http://localhost:1410/  When done, press any key to continue...  Waiting for authentication in browser...  Press Esc/Ctrl + C to abort  Authentication complete.  <Token>  <oauth\_endpoint>  authorize: https://www.facebook.com/dialog/oauth  access: https://graph.facebook.com/oauth/access\_token  <oauth\_app> facebook  key: 1675043216133450  secret: <hidden>  <credentials> {"access\_token":"EAAXzcWPhjUoBAHkIewDJZCM3cafo8ljokBDm7gyr0uDqNbbTORCeWz2D90DDlYZB7lxrUExn7uLDbwLqS3Jmu6ZBkKEj3BNtXJ9mUO4Ws1cZAyyJ6ZB1JAYZCtdnHzHDZCH2h46WpetTKBUQfWuZCVyolGjYcMTBRPUZD","token\_type":"bearer","expires\_in":5175666}  --- |
|  |
| |  | | --- | | > | |