week13

Dana Tomeh

4/17/2020

# RStudio API Code

# Libraries

library(twitteR)  
library(tidyverse)

## ── Attaching packages ───────────────────────────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ ggplot2 3.2.1 ✓ purrr 0.3.3  
## ✓ tibble 2.1.3 ✓ dplyr 0.8.4  
## ✓ tidyr 1.0.2 ✓ stringr 1.4.0  
## ✓ readr 1.3.1 ✓ forcats 0.4.0

## ── Conflicts ──────────────────────────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::filter() masks stats::filter()  
## x dplyr::id() masks twitteR::id()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::location() masks twitteR::location()

library(readr)  
library(tm)

## Loading required package: NLP

##   
## Attaching package: 'NLP'

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

library(qdap)

## Loading required package: qdapDictionaries

## Loading required package: qdapRegex

##   
## Attaching package: 'qdapRegex'

## The following object is masked from 'package:dplyr':  
##   
## explain

## The following object is masked from 'package:ggplot2':  
##   
## %+%

## Loading required package: qdapTools

##   
## Attaching package: 'qdapTools'

## The following object is masked from 'package:dplyr':  
##   
## id

## The following object is masked from 'package:twitteR':  
##   
## id

## Loading required package: RColorBrewer

## Registered S3 methods overwritten by 'qdap':  
## method from  
## t.DocumentTermMatrix tm   
## t.TermDocumentMatrix tm

##   
## Attaching package: 'qdap'

## The following objects are masked from 'package:tm':  
##   
## as.DocumentTermMatrix, as.TermDocumentMatrix

## The following object is masked from 'package:NLP':  
##   
## ngrams

## The following object is masked from 'package:forcats':  
##   
## %>%

## The following object is masked from 'package:stringr':  
##   
## %>%

## The following object is masked from 'package:dplyr':  
##   
## %>%

## The following object is masked from 'package:purrr':  
##   
## %>%

## The following object is masked from 'package:tidyr':  
##   
## %>%

## The following object is masked from 'package:base':  
##   
## Filter

library(textstem)

## Loading required package: koRpus.lang.en

## Loading required package: koRpus

## Loading required package: sylly

## For information on available language packages for 'koRpus', run  
##   
## available.koRpus.lang()  
##   
## and see ?install.koRpus.lang()

##   
## Attaching package: 'koRpus'

## The following object is masked from 'package:qdap':  
##   
## SMOG

## The following object is masked from 'package:readr':  
##   
## tokenize

##   
## Attaching package: 'textstem'

## The following object is masked from 'package:qdap':  
##   
## stem\_words

library(RWeka)  
library(wordcloud)  
library(ggplot2)  
library(tidytext)  
library(ldatuning)  
library(parallel)  
library(doParallel)

## Loading required package: foreach

##   
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':  
##   
## accumulate, when

## Loading required package: iterators

library(topicmodels)  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

# Data Import and Cleaning

* Setting up the twitter API so that tweets can be stripped
* The second chunk of cooded then pulls the first 5000 tweets that reference the hashtag #Weekend, and then strips retweets. Finally it converts the data into a data frame.
* Selected the necessary variables - the tweet itself, the favorite count, and the retweet count
* Finally, the imported\_tbl is exported as a csv and saved in the output folder as “tweets\_original.csv”

# api <- "i8qKfmFT2QK0yAXI4q8zon1zt"  
# apiSecret <- "uzXTpCJqbMInxXvOmIEuMgkYPoU2KZzAWYdSExQKykh3HHOVNS"  
# access <- "48436651-qNVHDPMkTOmmxTKdViX3AxuLET1COqil6yS0Hk9U3"  
# accessSecret <- "qqFeYeSliCQuhQ38Cl0LwiA8JwBcffU6lbxlBa1Uqi8Ab"  
# setup\_twitter\_oauth(api, apiSecret, access, accessSecret)  
#   
# tweets <- searchTwitter("#Weekend", 5000)  
# tweets\_clean <- strip\_retweets(tweets)  
# tweets\_tbl <- twListToDF(tweets\_clean)   
  
imported\_tbl <- read\_csv("../output/tweets\_original.csv")

## Parsed with column specification:  
## cols(  
## text = col\_character(),  
## favoriteCount = col\_double(),  
## retweetCount = col\_double()  
## )

imported\_tbl <- imported\_tbl %>%  
 select(text, favoriteCount, retweetCount)  
  
imported\_tbl$text <- imported\_tbl$text %>% iconv("UTF-8", "ASCII", sub="")  
write\_csv(imported\_tbl, "../output/tweets\_original.csv")

* In this second block of data import and cleaning, the tweets from above will be converted into preprocessed lemmas
* Created the twitter\_cp corpus
* Mapped the corpus as a plain text document
* Replaced abbreviations with full words
* Replaced contractions with the words that make them up
* Converted all letters to lowercawse
* Removed numbers from corpus
* Removed punctuation (including hashtags)
* Removed Stop words
* Stripped extra white space from tweets
* Lemmatized the words to their english stems

twitter\_cp <- VCorpus(VectorSource(imported\_tbl$text))  
twitter\_cp <- tm\_map(twitter\_cp, PlainTextDocument)  
twitter\_cp <- tm\_map(twitter\_cp, content\_transformer(replace\_abbreviation))  
twitter\_cp <- tm\_map(twitter\_cp, content\_transformer(replace\_contraction))  
twitter\_cp <- tm\_map(twitter\_cp, content\_transformer(str\_to\_lower))  
twitter\_cp <- tm\_map(twitter\_cp, removeNumbers)  
twitter\_cp <- tm\_map(twitter\_cp, removePunctuation)  
twitter\_cp <- tm\_map(twitter\_cp, removeWords, stopwords("en"))  
twitter\_cp <- tm\_map(twitter\_cp, stripWhitespace)  
twitter\_cp <- lemmatize\_words(twitter\_cp)

We continue DAta cleaning and import in the following block \* myTokenizer contains a function to tokenize the preprocessed corpus into a unigram and bigram \* twitter\_dtm converts the corpus into a matrix of unigram and bigram tokens \* we then remove all sparse terms from twitter\_dtm, and end up with 129 tokens \* we then count the rows that have no endorsed tokens, and remove them \* Finally, the rows to be dropped from the DTM are applied to the imported\_tbl to create a finalized tibble with tokens as columns, and all rows of remaining tweets including at least one endorsed token.

myTokenizer <- function(x) { NGramTokenizer(x, Weka\_control(min=1, max=2)) }  
twitter\_dtm <- DocumentTermMatrix(twitter\_cp, control = list(tokenize = myTokenizer ))  
twitter\_dtm <-removeSparseTerms(twitter\_dtm, .99)  
tokenCounts <- apply(twitter\_dtm, 1, sum)  
twitter\_dtm <- twitter\_dtm[tokenCounts > 0, ]  
dropped\_tbl <- imported\_tbl[tokenCounts >0, ]

# Visualization

* Here we set our slimmed and cleaned matrix as a tibble and then drop the term “weekend” which was our original hashtag search term on twitter
* wordCounts calculates how many times each token was endorsed
* wordNames pulls the actual tokens
* in wordcloud, word names are set as the words for the clouds, the counts from wordCounts tell us how many times they have appeared, and we decide to show the top 50 words.

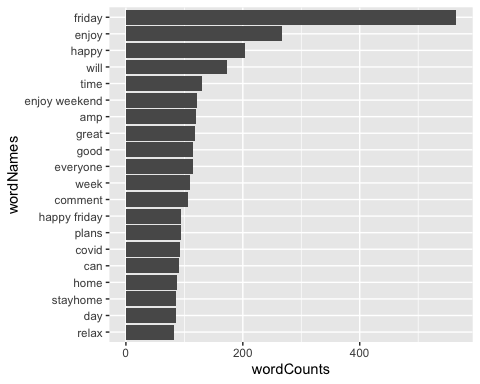
twitter\_tbl <- as\_tibble(as.matrix(twitter\_dtm)) %>%  
 select(!weekend)   
wordCounts <- colSums(twitter\_tbl)  
wordNames <- names(twitter\_tbl)  
wordcloud(wordNames, wordCounts, max.words=50)



* Below we create a horizontal bar graph in descending order of frequency of endorsement
* First we use wordNames and wordCounts from above and create a tibble
* Then we arrange the tibble in descending order based on the count of the word
* We choose to select the top 20 words
* Reorder them
* Plot from most to least frequent (of the top 20) in a horizontal bar graph

tibble(wordNames, wordCounts) %>%  
 arrange(desc(wordCounts)) %>%  
 top\_n(20) %>%  
 mutate(wordNames = reorder(wordNames, wordCounts)) %>%   
 ggplot(aes(x=wordNames,y=wordCounts))+ geom\_col() + coord\_flip()

## Selecting by wordCounts



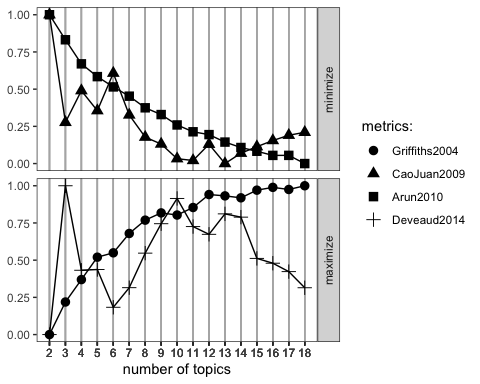
# Analysis

## Topic Modeling

# Preemptively reset the document names in the twitter\_dtm document Term matrix to be the text from the tweets. This allows us to group by document value later on when looking at gamma from the LDA. If we don't do this we end up with a bunch of charactger(0) and are unable to group.  
  
twitter\_dtm$dimnames$Docs <- dropped\_tbl$text  
  
#Setting up parallelization to make the following code faster   
  
local\_cluster <- makeCluster(detectCores()-1)  
registerDoParallel(local\_cluster)  
  
#Tuning four different metrics to decide what number of topics we have across our dataset of tokens  
  
tuning <- FindTopicsNumber( twitter\_dtm, topics = seq(2,18,1), metrics = c("Griffiths2004","CaoJuan2009", "Arun2010", "Deveaud2014"), verbose = T)

## fit models... done.  
## calculate metrics:  
## Griffiths2004... done.  
## CaoJuan2009... done.  
## Arun2010... done.  
## Deveaud2014... done.

FindTopicsNumber\_plot(tuning)



#returning to normal processing  
stopCluster(local\_cluster)   
registerDoSEQ()

Griffiths (2004) line is essentially a perfectly increasing curve so its meaningless. There are some dips and climps but it flattens out between 11 and 15. Since it’s a max curve it indicates a number of topics between 11 and 15 Deveaud (2014) has three peaks, at 5, 10 and 13. The peaks at 5 and 13 are about the same, and the one at 10 is slightly higher. Minima of Both Arun (2010) and Cao Juan (2009) indicate that we might be looking at 13 topics. The minimum is technically 15 but there isn’t a lot of differentiation there so i’m going with 13.

As a result of these curves, i’m going with 13 topics.

Below we will use the LDA betas to determine the probability that a word belongs to a topic

lda\_results <- LDA(twitter\_dtm, 13)  
lda\_betas <- tidy(lda\_results, matrix="beta")  
  
betas <-lda\_betas %>%   
 group\_by(topic) %>%   
 top\_n(10, beta) %>%   
 arrange(topic, -beta)   
betas

## # A tibble: 134 x 3  
## # Groups: topic [13]  
## topic term beta  
## <int> <chr> <dbl>  
## 1 1 enjoy 0.198   
## 2 1 plans 0.115   
## 3 1 friyay 0.0967  
## 4 1 going 0.0967  
## 5 1 enjoy weekend 0.0849  
## 6 1 comment 0.0623  
## 7 1 friday enjoy 0.0600  
## 8 1 comment friday 0.0575  
## 9 1 weekend comment 0.0575  
## 10 1 friyay plans 0.0563  
## # … with 124 more rows

Below we will use the LDA Gammas to determine the probability that the document contains the topic

* the topics variable also shows us the most likely topic per tweet in this dataset (by using top\_n(1, gamma) we select the most likely topic)

lda\_gammas <- tidy(lda\_results, matrix="gamma")  
  
topics <- lda\_gammas%>%  
 group\_by(document) %>%  
 top\_n(1, gamma) %>%  
 ungroup %>%  
 arrange(document) #this is the tweet value. we changed this above to make sure that it says the actual tweet and not character(0)  
  
topics

## # A tibble: 2,411 x 3  
## document topic gamma  
## <chr> <int> <dbl>  
## 1 "\n\n\n\n\n\n\n\n#\n#WEEKEND https://t.co/kwNeuxMnyo" 13 0.100  
## 2 "\n\n\n\nUPLift\nweekend\n\n\nPA\n\n#UPLift\n#weekend" 13 0.130  
## 3 "\n\n\n#mood #goodnight #weekend #sailormoon #vaporwave https://… 13 0.100  
## 4 "\n\n \n &amp; 15% https://t.co/qb0ZHHudrz" 12 0.150  
## 5 "\n\n #weekend \n https://t.co/BlMnxmg7Yx" 13 0.100  
## 6 "\n\n#Weekend Farmer \n # https://t.co/tzzDiqrXpM" 13 0.100  
## 7 "\n\nTake Free Mock Tests here: https://t.co/ViLAiOlxZM\n\n#Exam… 12 0.131  
## 8 "\n.\n#latenight #nightlife #breathe #wander #mood #wanderlust #… 13 0.100  
## 9 "\n...\n\*\n# #cutie #cute #beautiful #girl #love #dayoff #weeken… 4 0.152  
## 10 "\n@amacc\_sa\n.\n.\n#amacc\n#earrings\n#Friday\n#FridayReads\n#F… 11 0.130  
## # … with 2,401 more rows

* Below is the code to add the topic number to the twitter tibble we created above that contains all the tokens.
* First I wanted to make sure that we were adding the right topic to the right tweet based on the gamma values from the LDA. So to do that I added the tweet to the twitter tibble then arranged by tweet (resulting in 129 columns), which puts the rows in the same order as the code that created the topics dataframe above (which is arranged by tweet in a column called document)
* Used cbind to add in the topic column to the twitter\_tbl, resulting in 130 columns

twitter\_tbl <- twitter\_tbl %>%  
 mutate(tweet = dropped\_tbl$text) %>%  
 arrange(tweet)  
  
twitter\_tbl <- cbind(twitter\_tbl, topics$topic)

Based on the analysis above here are speculations regarding what the topics may mean: Topic 1: References time and relaxation, so maybe taking time to refresh over the weekend Topic 2: Doesn’t seem super cohesive. It references taking care and getting stuff done Topic 3: Seems to reference doing stuff, like music, family, and getting ready Topic 4: Just mentions friday and the weekend again, so just comments that its friday maybe? Topic 5: Also references friday thoughts, vibes and feelings. Topic 6: References tokens like beautiful, nature, and love. So separates beauty and love out Topic 7: Has a lot of two word tokens referencing enjoyment and plans. So people enjoy the weekend Topic 8: Mentions almost entirely safety related tokens. So styaing safe Topic 9: Referencing staying home and staying safe again, so likely related to staying home because of the virus Topic 10: Lots of combinations of happy weekend and friday so even in lockdown people are excited for the weekend Topic 11: A little incohesive. Mentioning great, day, today, and another. Maybe people are having good fridays? Topic 12: Better, fantastic, good, and enjoy. Seems like wishes for a good weekend Topic 13: This last topic mentions covid, quarantine, and work. So references to be quarantined for the weekend cuz of covid?

## Machine Learning

* Here I arranged dropped\_tbl by tweet (which is called text here) to make sure that it is in the same order as twitter\_tbl.
* Then I added the favoriteCount from the dropped\_tbl to twitter\_tbl
* I used colnames to rename the two added columns for clarity
* svr\_mod1 does not include the topic as a predictor of tweet popularity (favoriteCount)
* svr\_mod2 does include the topic as a predictor of tweet popularity (favoriteCount)

dropped\_tbl <- dropped\_tbl %>%  
 arrange(text)  
  
twitter\_tbl <- cbind(twitter\_tbl, dropped\_tbl$favoriteCount)  
colnames(twitter\_tbl)[130] <- "topic"  
colnames(twitter\_tbl)[131] <-"favoriteCount"  
  
# setting up Parallelization  
local\_cluster <- makeCluster(detectCores()-1)  
registerDoParallel(local\_cluster)  
  
svr\_mod1 <- train(  
 favoriteCount ~ . -tweet -topic, #models all variables as predictors of favorite count except for the tweet itself and topic number  
 twitter\_tbl,   
 method="svmLinear",  
 preProcess=c("center", "scale", "zv", "medianImpute"),   
 trControl= trainControl(method="cv", number = 10, verboseIter = T),   
 na.action = na.pass  
)

## Aggregating results  
## Fitting final model on full training set

svr\_mod2 <- train(  
 favoriteCount ~ . -tweet, #models all variables as predictors of favorite count except for the tweet itself   
 twitter\_tbl,   
 method="svmLinear",  
 preProcess=c("center", "scale", "zv", "medianImpute"),   
 trControl= trainControl(method="cv", number = 10, verboseIter = T),   
 na.action = na.pass  
)

## Aggregating results  
## Fitting final model on full training set

#stopping parallelization  
  
stopCluster(local\_cluster)   
registerDoSEQ()  
  
#testing the model on the holdout data   
  
#testing the correlation between the svr\_mod1 predictions and the actual outcome variables   
mod1\_cor <- cor.test(predict(svr\_mod1, twitter\_tbl, na.action=na.pass)  
, twitter\_tbl$favoriteCount)  
mod2\_cor <- cor.test(predict(svr\_mod2, twitter\_tbl, na.action=na.pass)  
, twitter\_tbl$favoriteCount)

The correlation between the actual favoriteCounts and the predicted ones in model one (does not include topic number) is 0.0181999 and for model two (includes topic number) is 0.0208867. They are essentially identical.

summary(resamples(list("SVR without Topic" = svr\_mod1, "SVR with Topic" = svr\_mod2)))

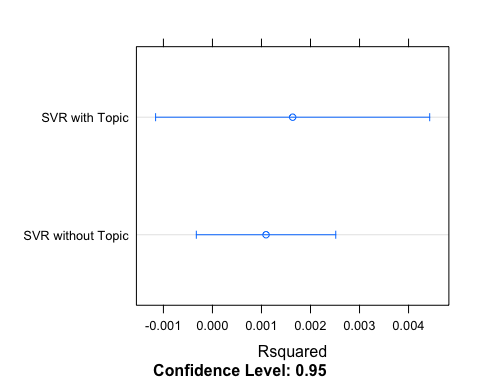
##   
## Call:  
## summary.resamples(object = resamples(list(`SVR without Topic` = svr\_mod1,  
## `SVR with Topic` = svr\_mod2)))  
##   
## Models: SVR without Topic, SVR with Topic   
## Number of resamples: 10   
##   
## MAE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## SVR without Topic 3.835684 4.176261 4.877684 5.827633 6.496246 11.822388 0  
## SVR with Topic 3.524716 4.105092 6.006652 5.823811 7.480593 7.826966 0  
##   
## RMSE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## SVR without Topic 6.895983 9.747182 21.79840 26.31338 29.56456 79.40911 0  
## SVR with Topic 5.257058 7.534707 26.29782 27.66337 41.81192 63.94132 0  
##   
## Rsquared   
## Min. 1st Qu. Median Mean  
## SVR without Topic 1.139541e-06 4.291359e-05 9.576907e-05 0.001093184  
## SVR with Topic 3.422345e-06 1.055294e-04 1.200628e-04 0.001635783  
## 3rd Qu. Max. NA's  
## SVR without Topic 0.0008385366 0.005982262 0  
## SVR with Topic 0.0008780362 0.012664612 0

Without including the Topic in the model, the mean MAE is smaller (5.817) than when the topic is included in the model (5.822). A smaller MAE is preferred, which indicates excluding the topic from the model. However, the MAEs are quite similar so it may not be actually indicative of one model being better than the other. Without inlcuding the Topic in the model, the mean RMSE is smaller (25.584) than when the topic is included in the model (26.584). A smaller RMSE is preferred, which indicates excluding the topic from the model. However, the RMSEs are quite similar so it may not be actually indicative of one model being better than the other. Without including the Topic in the model, the mean Rsquared is smaller (.0007) than when the topic is included in the model (.0015). A larger Rsquared is preferred, which indicates including the topic score in the model. The difference here is a little more drastic than the MAE or RMSE (the Rsquared of the model that includes the topic is double that of the one that does not), but the values are miniscule, so again there is a lack of clear indication as to model preference.

### Rsquared plot

This plot shows a larger Rsquared for the model that includes the topic than the model that excludes it. However, the CI of the model that excludes the topic is completely included within the CI of the model that includes it.

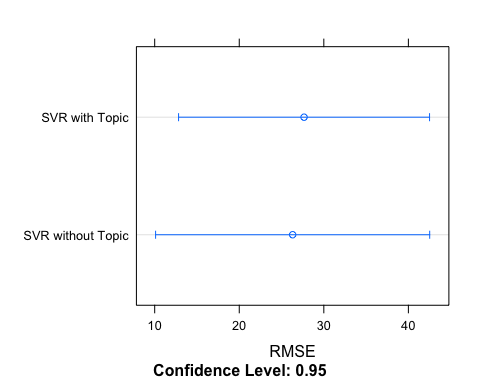
dotplot(resamples(list("SVR without Topic" = svr\_mod1, "SVR with Topic" = svr\_mod2)), metric = "Rsquared")



### RMSE Plot

This plot shows a smaller RMSE for the model that excludes the topic value, than the one that includes it. However the CI of the model that includes the topic is completely nested within the model that excludes the topic.

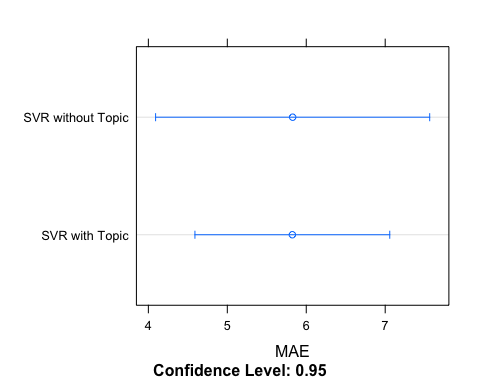
dotplot(resamples(list("SVR without Topic" = svr\_mod1, "SVR with Topic" = svr\_mod2)), metric = "RMSE")



### MAE

This plot shows very similar MAE values for the two models. However the CI of the model that includes the topic is completely nested within the model that excludes the topic.

dotplot(resamples(list("SVR without Topic" = svr\_mod1, "SVR with Topic" = svr\_mod2)), metric = "MAE")



## Final Interpretation

In this case its hard to say that topic modeling had a lot to do with predicting the popularity of tweets. There are many reasons why this might be. The first is that I used the hashtag #weekend which I’d beleive generally most people are positive about, meaning emotion didn’t really come into play as much. Also, My topics as determined by the gamma analysis are all super similar, it was hard to distinguish between many of them. The two models had similar RMSE, MAE, and Rsquared values, and their correlations between the predicted and actual values were essentually the same (and pretty small). That being said, I think it would be impossible to pick a model that better predicted popularity, so the inclusion of the topic didn’t really help the modeling. I actually did expect to see a realationship between the topic and the prediction, but I think I limited myself by pulling tweets that had very little variance in terms of emotion or content . Even just reading through the tokens that I ended up with, you can kind of tell the sentiment there. Many reference positive emotion, fridays, happy, good, fantstic, so on.