week12

Mathi Manavalan

4/14/2020

Libraries

Importing the libraries necessary for the data importing, cleaning, and analyzing.

```
library(twitteR)
library(tidyverse)
library(tm)
library(textstem)
library(qdap)
library(RWeka)
library(wordcloud)
library(stringi)
library(ldatuning)
library(topicmodels)
library(tidytext)
library(caret)
library(parallel)
library(doParallel)
library(psych)
```

Data Import and Cleaning

The keys inputted into the function below are the keys found for my particular app through my Twitter developer account. This allows me to then pull tweets. I am pulling original tweets with the hashtag #psychology and saving it as a tibble. I am also saving the tibble as a csv file. (After this process was done, I changed it so that I am pulling the twitter data from the output file I created.)

I am then creating a corpus of the imported tweets. Here, I am also preprocessing the tweets such that unnecessary fillers, such as punctuation, whitespace, and stopwords, are removed appropriately so that we are left with relevant content. (I have added additional comments to lines that I have newly added or modified in some way that is not immediately intuitive.)

I am then creating a DocumentTermMatrix containing unigrams and bigrams of the content in my corpus. Extremely sparse tokens are removed, as well as those that no longer appear (post preprocessing). Last but not least, I am transforming this into a tibble.

```
twitter_cp <- VCorpus(VectorSource(imported_tbl$text))

twitter_cp<-tm_map(twitter_cp, PlainTextDocument)

# below line removes all hashtags
```

```
twitter_cp<-tm_map(twitter_cp, content_transformer( (function(x) {
  str_remove_all( x, pattern = "#+[a-zA-Z0-9(_)]{0,}")
})))

# below line removes all hashtags that begin with or are web URLS
twitter_cp<-tm_map(twitter_cp, content_transformer( (function(x) {
  str_remove_all( x, pattern = "^(http)[a-zA-Z0-9(_)]{0,}")
})))

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)
# below line removes all stopwords as well as the word 'psychology', as that
  #is the hashtag term that was searched for, so that doesn't add meaning to
  #our analysis
twitter_cp<-tm_map(twitter_cp, removeWords, c(stopwords("en"), "psychology"))
twitter_cp<-tm_map(twitter_cp, stripWhitespace)
twitter_cp<-tm_map(twitter_cp, content_transformer(lemmatize_words))


myTokenizer <- function(x) {
  NGramTokenizer(x, Weka_control(min = 1, max = 2))
}

twitter_dtm <- DocumentTermMatrix(twitter_cp,
                                  control = list(tokenize = myTokenizer))
#Here I am using a sparsity factor of 0.99 because otherwise, I am getting very very few terms.
twitter_dtm <- removeSparseTerms(twitter_dtm, 0.99)


tokenCounts <- apply(twitter_dtm, 1, sum)
twitter_dtm <- twitter_dtm[tokenCounts > 0, ]

DTM.matrix <- as.matrix(twitter_dtm)

dropped_tbl <- imported_tbl[tokenCounts > 0, ]
```

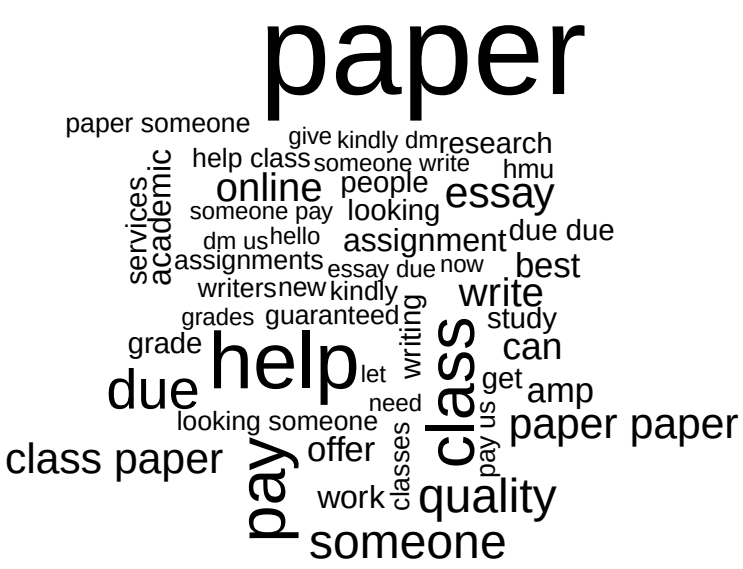
Visualization

Here, I am creating a wordcloud visualization of up to the 50 most frequent words, as well as a horizontal bar chart of the top 20 bigram lemmas.

```
twitter_tbl <- as_tibble(DTM.matrix)

#Word cloud

wordCounts <- colSums(twitter_tbl)
wordNames <- names(twitter_tbl)
wordcloud(wordNames, wordCounts, max.words=50)
```

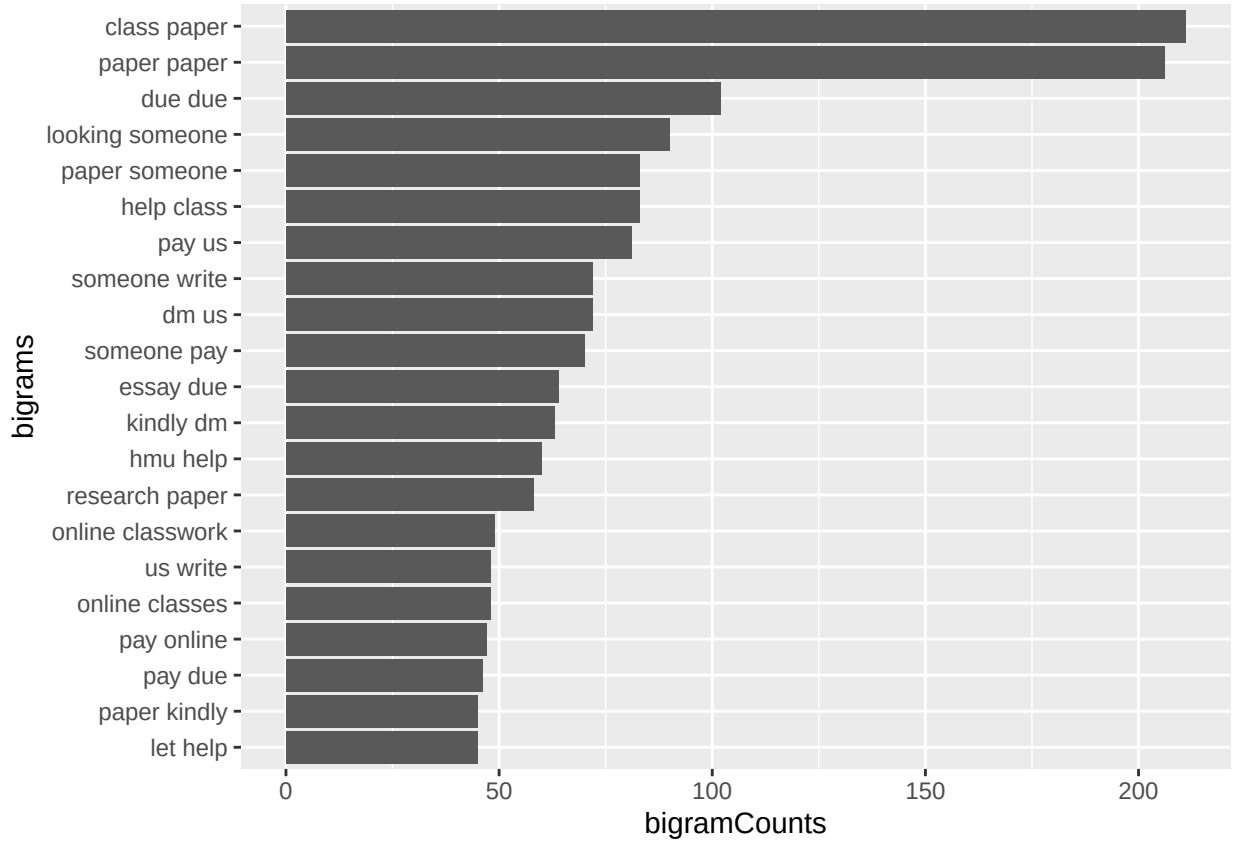



```
#Horizontal bar chart

bigramCounts <- wordCounts[stri_count_words(wordNames) == 2]
bigrams <- wordNames[stri_count_words(wordNames) == 2]

tibble(bigrams, bigramCounts) %>%
  arrange(desc(bigramCounts)) %>%
  top_n(20) %>%
  mutate(bigrams = reorder(bigrams, bigramCounts)) %>%
  ggplot(aes(x=bigrams,y=bigramCounts)) + geom_col() + coord_flip()
```

```
## Selecting by bigramCounts
```



Analysis

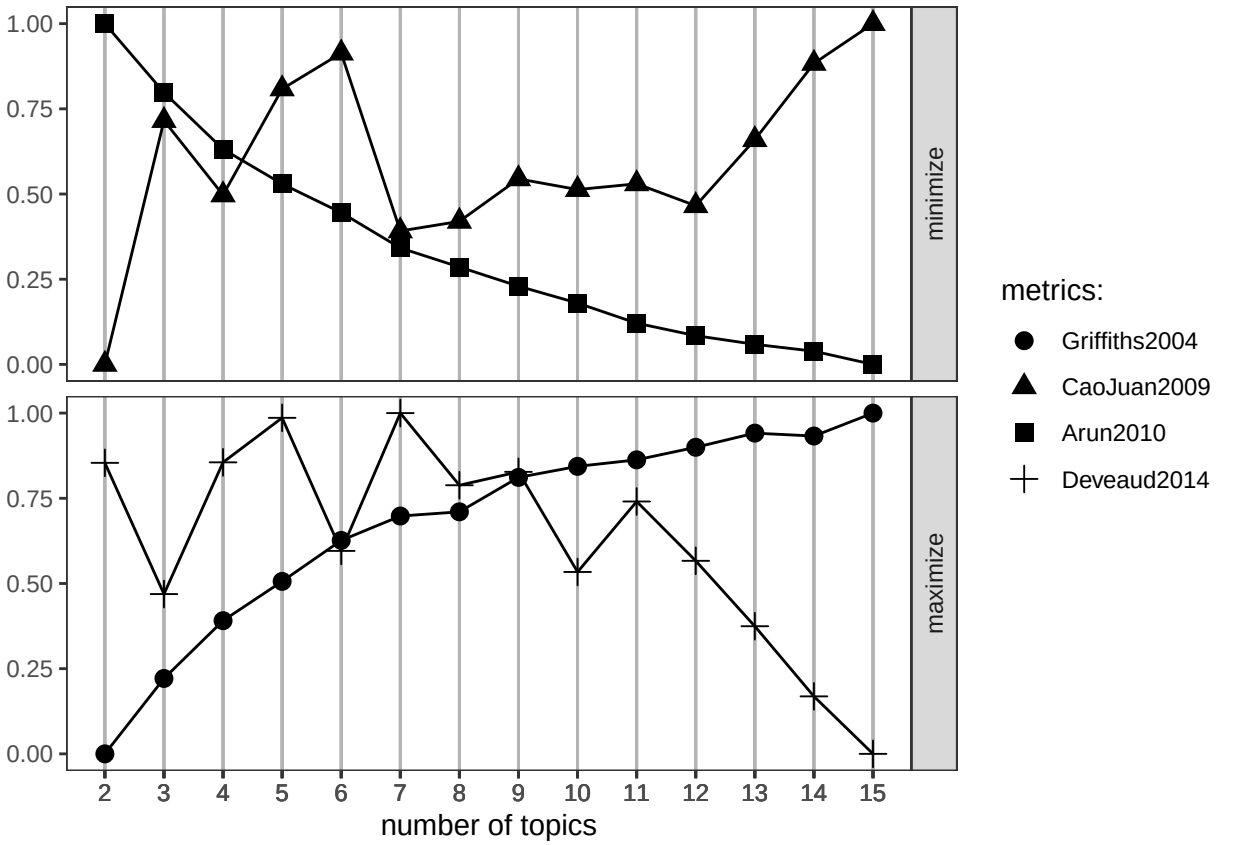
Topic Modeling

Here, I am creating 2 graphical summaries to aid in understanding the ideal number of topics.

```
tuning <- FindTopicsNumber(twitter_tbl, topics = seq(2,15,1), metrics = c("Griffiths2004", "CaoJuan2009
```

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

```
FindTopicsNumber_plot(tuning)
```



From the above line plots, we can see that 5 (as the number of topics) seems to be the most ideal according to CaoJuan2009 (since it is the lowest point on that line) and 7 seems to be the most ideal number of topics according to Deveaud2014. * The above plot changes on every run, so the topics most ideal (seen by choosing the highest point on Deveaud2014 line and the lowest point on the CaoJuan2009 line) changes. My above explanation was for one particular run.

```
lda_results <- LDA(twitter_tbl, 10)

lda_betas<-tidy(lda_results, matrix="beta")

betas <- lda_betas %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  arrange(topic, -beta)

View(betas)
```

From the **betas** data frame, we can see that the tokens connected to topic 5 seems to mostly be about class and paper and writing, so I would say that an overal interpretation you could make for this topic is that it is about writing papers for a psychology course. For topic 7, most of the words seem to pertain to performance in the class (as tokens such as grades, assignments, and help are included).

```
lda_gammas <- tidy(lda_results, matrix="gamma")

gammas <- lda_gammas %>%
  group_by(document) %>%
```

```
  top_n(1, gamma) %>%
  slice(1) %>%
  ungroup %>%
  mutate(document = as.numeric(document)) %>%
  arrange(document)

View(gammas)
```

Here, we can see that gammas shows us which topic is contained in each document, with associated probabilities.

Then, I am creating a tabular summary of most likely topic per tweet (aka, the number of documents categorized in each topic).

```
gammas_long <- gammas %>%
  select(-gamma) %>%
  table() %>%
  as_tibble() %>%
  pivot_wider(names_from = topic, values_from = n)

tabSum <- colSums(gammas_long[2:11])
tabSum
```

```
##   1   2   3   4   5   6   7   8   9  10
## 221 132 459 274 453 151 145  82  85 315
```

Here, I am adding a column called **topic** containing the topic identifiers for the tokens in **twitter_tbl**.

```
twitter_tbl$topic <- gammas$topic
```

Machine Learning

Here, I am creating two 10-fold cross-validated support vector regression models to predict tweet popularity. One model uses just the tokens as predictors, and the other uses both tokens as well as topics as predictors. I am taking advtange of parallelizing in order to run the models more quickly.

```
twitter_tbl$tweetPop <- dropped_tbl$favoriteCount


dummies <- dummy.code(gammas$topic)
dummies_tbl <- as_tibble(dummies)
names(dummies_tbl) <- paste0(rep("topicNum", 10), names(dummies_tbl)) # or better names
twitter_tbl <- twitter_tbl %>% bind_cols(dummies_tbl)


holdout <- sample(nrow(twitter_tbl), 200)
train <- (1:nrow(twitter_tbl))[-holdout]

train_tbl <- twitter_tbl[train, ]
test_tbl <- twitter_tbl[holdout,]

cores <- detectCores()
```

```
local_cluster <- makeCluster(cores - 1)
registerDoParallel(local_cluster)

svr_model <- train(
  tweetPop ~ .,
  select(train_tbl, -starts_with("topicNum")), #-topic), #train_tbl,
  method = "svmLinear3",
  preProcess = c("center", "scale", "nzv", "knnImpute"),
  trControl = trainControl(method = "cv", number = 10, verboseIter = T),
  na.action = na.pass
)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting cost = 0.25, Loss = L2 on full training set
```

```
svr_topic_model <- train(
  tweetPop ~ .,
  train_tbl,
  method = "svmLinear3",
  preProcess = c("center", "scale", "nzv", "knnImpute"),
  trControl = trainControl(method = "cv", number = 10, verboseIter = T),
  na.action = na.pass
)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting cost = 0.25, Loss = L2 on full training set
```

```
stopCluster(local_cluster)
registerDoSEQ()

cor <- cor(predict(svr_model, select(test_tbl, -starts_with("topicNum")), na.action = na.pass), test_tbl

svr_model
```

```
## L2 Regularized Support Vector Machine (dual) with Linear Kernel
##
## 2117 samples
##  139 predictor
##
## Pre-processing: centered (18), scaled (18), nearest neighbor imputation
##  (18), remove (121)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1906, 1905, 1905, 1905, 1906, 1906, ...
## Resampling results across tuning parameters:
##
##   cost  Loss  RMSE      Rsquared     MAE
##   0.25  L1    3.762198  0.003794210  1.022728
##   0.25  L2    3.635542  0.027593743  1.343670
##   0.50  L1    3.762566  0.002532749  1.027081
##   0.50  L2    3.636937  0.027146421  1.346036
```

```
##   1.00  L1    3.762161  0.001059275  1.020907
##   1.00  L2    3.638540  0.026231369  1.351120
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were cost = 0.25 and Loss = L2.
```

```
cor
```

```
## [1] 0.02038745
```

```
cor_topic <- cor(predict(svr_topic_model, test_tbl, na.action = na.pass), test_tbl$tweetPop)^2

svr_topic_model
```

```
## L2 Regularized Support Vector Machine (dual) with Linear Kernel
##
## 2117 samples
##  149 predictor
##
## Pre-processing: centered (26), scaled (26), nearest neighbor imputation
##  (26), remove (123)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1905, 1905, 1905, 1906, 1905, 1906, ...
## Resampling results across tuning parameters:
##
##   cost  Loss  RMSE      Rsquared     MAE
##   0.25  L1    3.810383  0.002894543  1.020283
##   0.25  L2    3.689350  0.025789660  1.353303
##   0.50  L1    3.811922  0.003446681  1.020710
##   0.50  L2    3.690844  0.025441695  1.352070
##   1.00  L1    3.809403  0.003530754  1.021279
##   1.00  L2    3.692513  0.024728175  1.350836
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were cost = 0.25 and Loss = L2.
```

```
cor_topic
```

```
## [1] 0.01773989
```

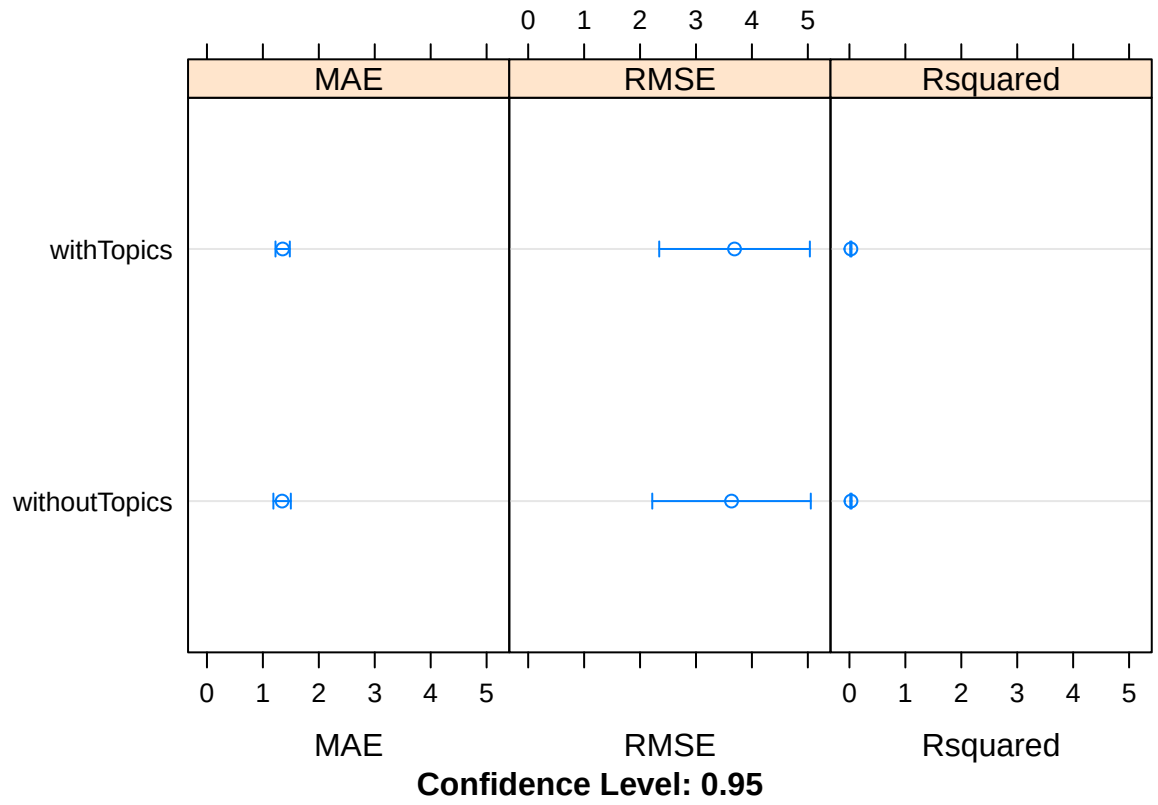
Here, I am visually comparing the two models shown above.

```
resamples(list("withoutTopics" = svr_model,
               "withTopics" = svr_topic_model)) %>%
  summary
```

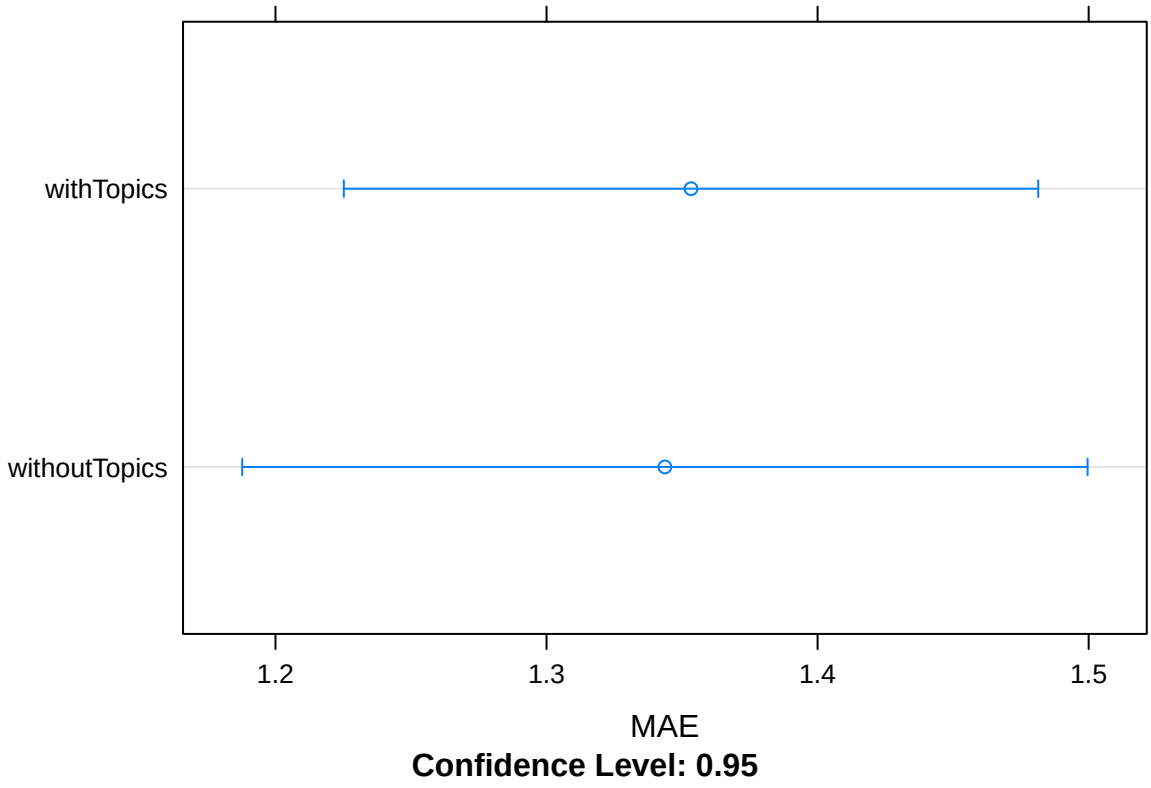
```
##
## Call:
## summary.resamples(object = .)
##
## Models: withoutTopics, withTopics
```

```
## Number of resamples: 10
##
## MAE
##                    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## withoutTopics 1.030061 1.246134 1.335814 1.343670 1.458036 1.667813    0
## withTopics    1.138845 1.246779 1.301420 1.353303 1.467907 1.662373    0
##
## RMSE
##                    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## withoutTopics 1.294144 2.455633 3.218131 3.635542 4.535643 7.719791    0
## withTopics    1.778245 2.354394 3.015450 3.689350 4.274678 7.597848    0
##
## Rsquared
##                        Min.    1st Qu.     Median       Mean    3rd Qu.
## withoutTopics 0.0011902723 0.01846353 0.02573740 0.02759374 0.04029997
## withTopics    0.0005990861 0.01331224 0.02721929 0.02578966 0.03870273
##                       Max. NA's
## withoutTopics 0.05007563    0
## withTopics    0.04566387    0
```

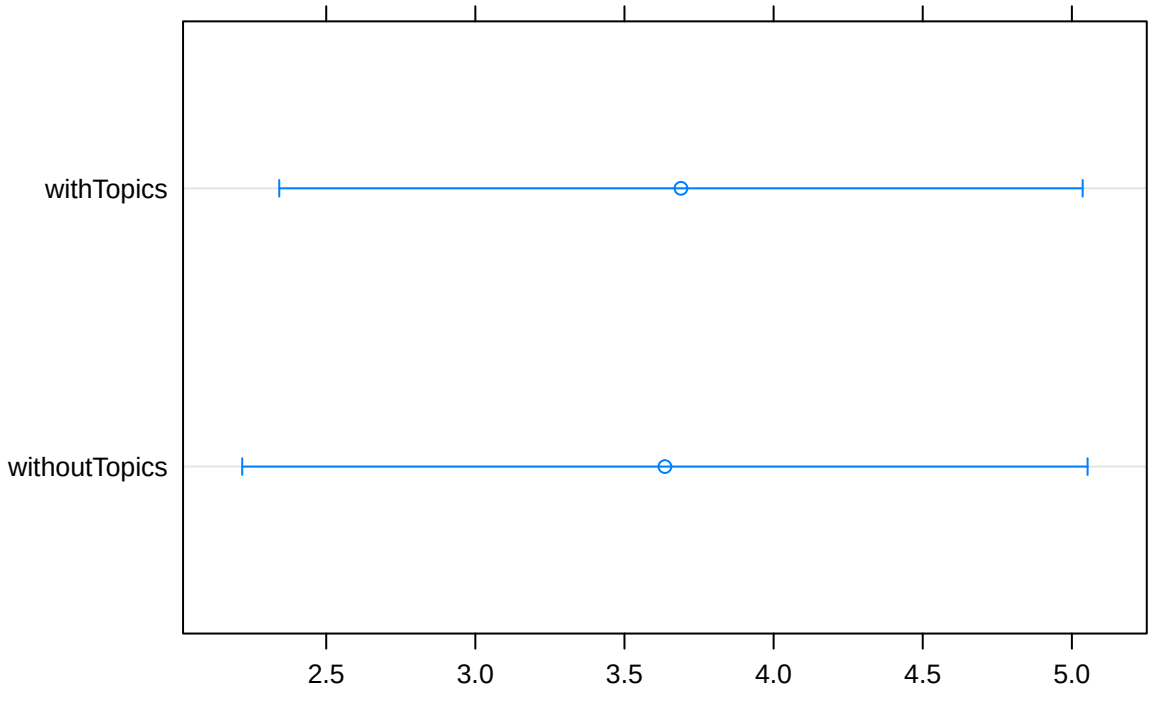
```
resamples(list("withoutTopics" = svr_model,
               "withTopics" = svr_topic_model)) %>%
  dotplot
```



```
resamples(list("withoutTopics" = svr_model,
               "withTopics" = svr_topic_model)) %>%
  dotplot(metric = "MAE")
```

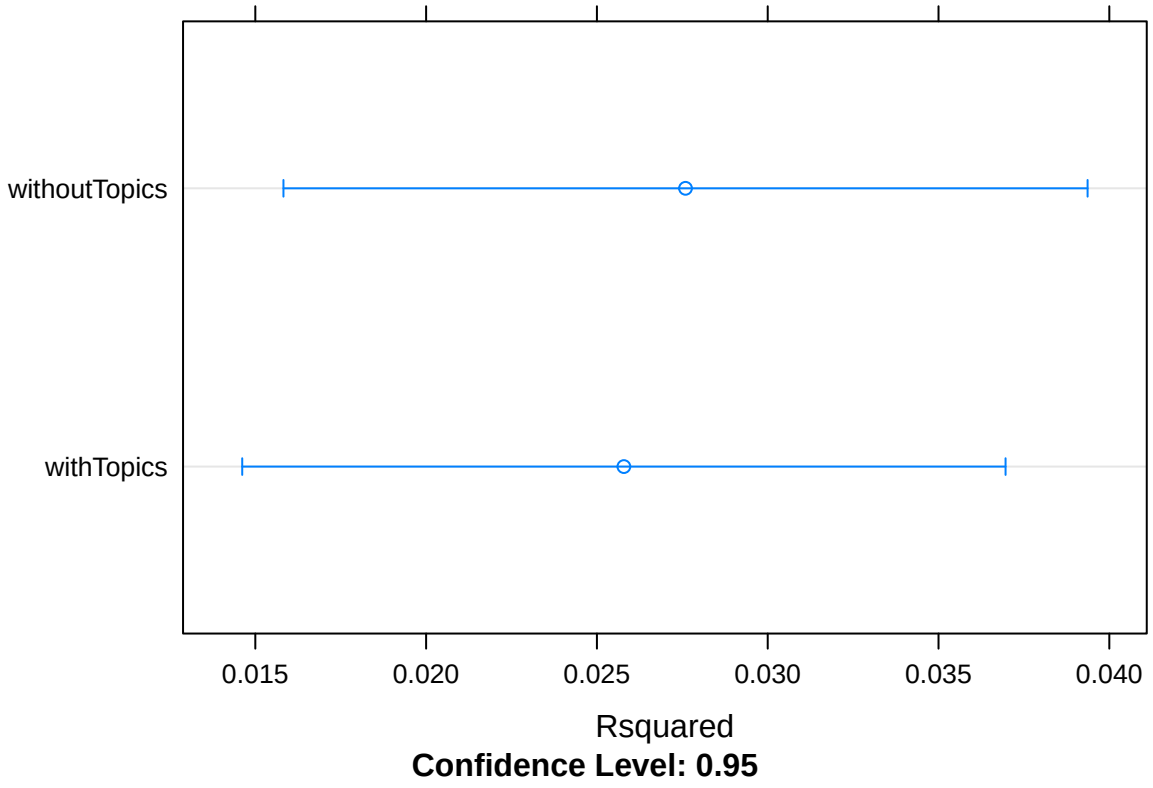



```
resamples(list("withoutTopics" = svr_model,
               "withTopics" = svr_topic_model)) %>%
  dotplot(metric = "RMSE")
```



Confidence Level: 0.95

```
resamples(list("withoutTopics" = svr_model,
               "withTopics" = svr_topic_model)) %>%
  dotplot(metric = "Rsquared")
```



Final Interpretation

From the model output of the first chunk under *Machine Learning*, we can see that the correlation value for the model that predicted tweet popularity based on just the tokens was 0.0149. For the model that predicted tweet popularity based on both the tokens as well as the topics, the correlation value was 0.004. In other words, the model without topics as additional predictors performed better (relatively, as both models are still quite poor) at predicting tweet popularity than the model that included the topics as predictors. Because the numbers are so close, it is not quite as evident on the overall dotplot above as to which model is better. With the additional input of metric, we can see that the errors are actually very similar, with very similar intervals as well. With the r-squared metric, it becomes a little more clear that the model with topics has a greater R-squared value, but at the same time, it has such a large 95% confidence interval. In addition, this confidence interval covers the R-squared value for the model without topics as well, so it becomes difficult to truly differentiate the two models.

If we were to point-blank look at R-squared values, I would conclude that including topics are important in predicting the popularity of tweets.