License plate EDA

Joseph Marlo

2/8/2020

```
library(tidyverse)
library(tidytext)
library(igraph)
library(ggraph)
library(stringdist)
library(knitr)
source("Analyses/Helper_functions.R")
set.seed(44)
```

Goal

The goal is to perform exploratory data analysis to understand if there are any natural groupings or trends that can be exploited for a later classification algorithm.

```
# read in the data then filter to plates that were either accepted
#  or rejected and add spaces between plate characters so we can split it
#  it into ngrams later
app.plates <- read_csv("Inputs/applications.csv") %>%
  filter(status %in% c("Y", "N")) %>%
  rowid_to_column() %>%
  mutate(plate.sep = gsub(" ", "", plate, fixed = TRUE),
         plate.sep = gsub("*", "\\1 \\2", plate.sep)) %>%
  rename(id = rowid)

head(app.plates) %>% kable(format = 'markdown')
```

id	plate	review_reason_code	customer_meaning	reviewer_comments	status	plate.sep
1	AZIZ714	2	LAST NAME	714 AREA CODE	N	A Z I Z 7 1 4
2	BATBOX1	1	BATMOBILE (BATMAN) PLUS SHAPE OF VEHICLE (SCION XB)	BOX	N	B A T B O X 1
3	BBOMBS	2	NO MICRO AVAILABLE	BOMBS	N	B B O M B S
4	BEACHY1	4	LOVE THE BEACH	BEACHY LOOKS LIKE BITCHY 1	N	B E A C H Y 1
5	BLK PWR5	2	STRENGTH OF FAMILY	BLACK POWER	N	B L K P W R 5
6	BOT TAK	NA	THIS IS IT	CAN NOT TRANSLATE	N	B O T T A K

Create ngrams

First, we need to create ngrams of the license plate text. The license plates are typically around 6.9 characters and appear to be rejected based on only 3, or 4 characters. The idea is to identify which characters are being flagged by the DMV reviewers. We first need to split the plates into ngrams. Typically, this can be easily done using the package `tidytext` but since we are using the characters, not words, it will be easier to write our own function. `parse_plate()` will handle this and allow us to specify multiple lengths of the ngram (e.g. 3 and 4 characters).

```
# create tokenzied data: custom method
plate.ngrams <- app.plates %>%
  select(id, plate) %>%
  rowwise() %>%
  mutate(word = list(parse_plate(plate, ngram.nchar = 3:4))) %>%
  ungroup() %>%
  unnest(word)

head(plate.ngrams) %>% kable()
```

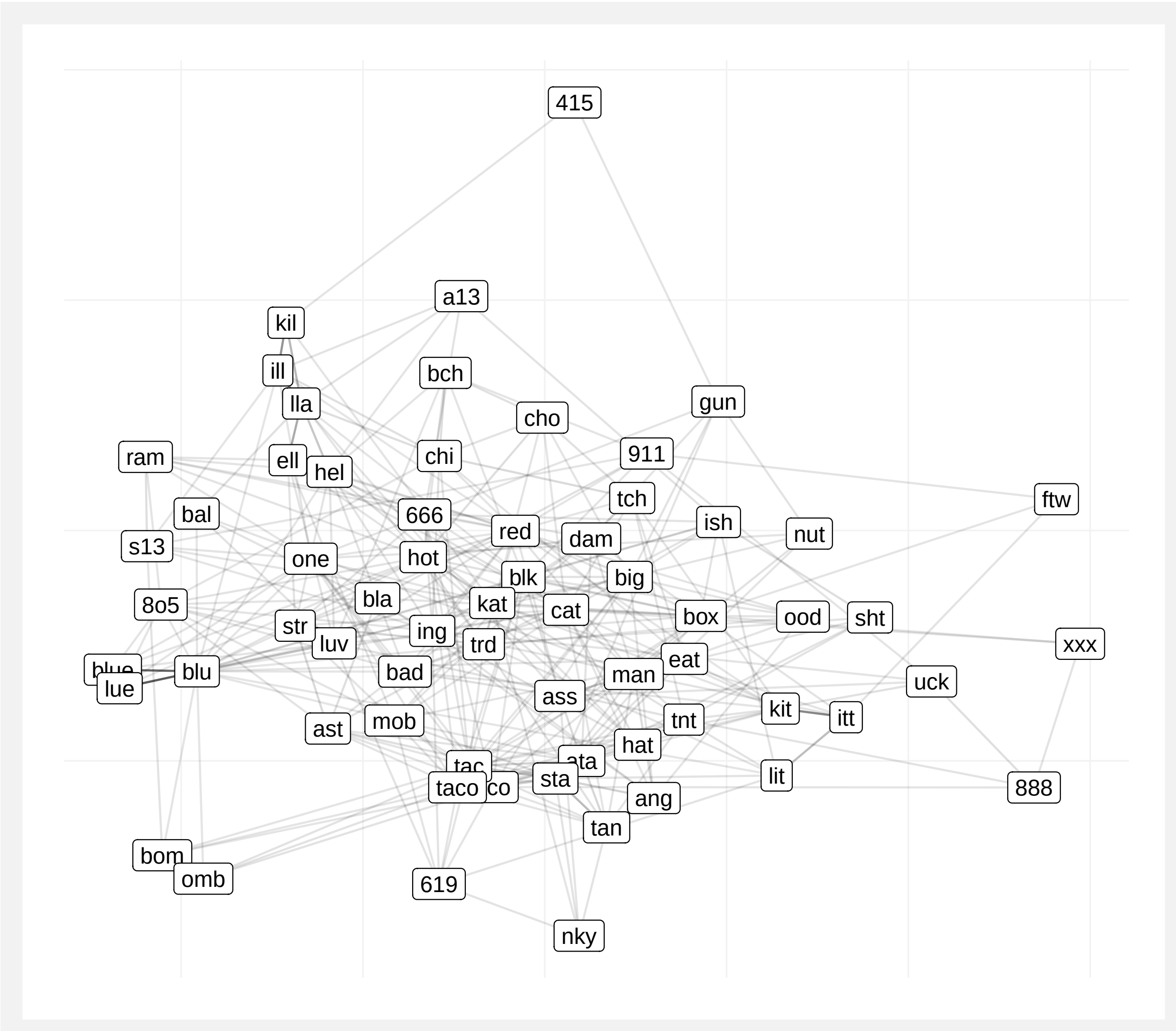
id	plate	word
1	AZIZ714	azi
1	AZIZ714	aziz
1	AZIZ714	ziz
1	AZIZ714	ziz7
1	AZIZ714	iz7
1	AZIZ714	iz71

Cosine similarity and clustering

To see if there are any natural groupings we can visualize the relationships between the ngrams based on cosine simlilarity. This method examines how often ngrams are co-present within a given license plate. We visualize the relationships with a graph and then cluster to see if there are any clear groupings.

See this post on markhw.com to learn more about the method and the source of the code.

```
set.seed(44)
# calculate the cosine matrix of our ngrams
cos.mat <- cosine_matrix(plate.ngrams, lower = .003, upper = .80, filt = .8)

# plot the matrix
ngram.graph <- graph_from_adjacency_matrix(cos.mat,
                                           mode = "undirected",
                                           weighted = TRUE)
ngram.graph %>%
  ggraph(layout = "nicely") +
  geom_edge_link(aes(alpha = weight), show.legend = FALSE) +
  geom_node_label(aes(label = name))
```

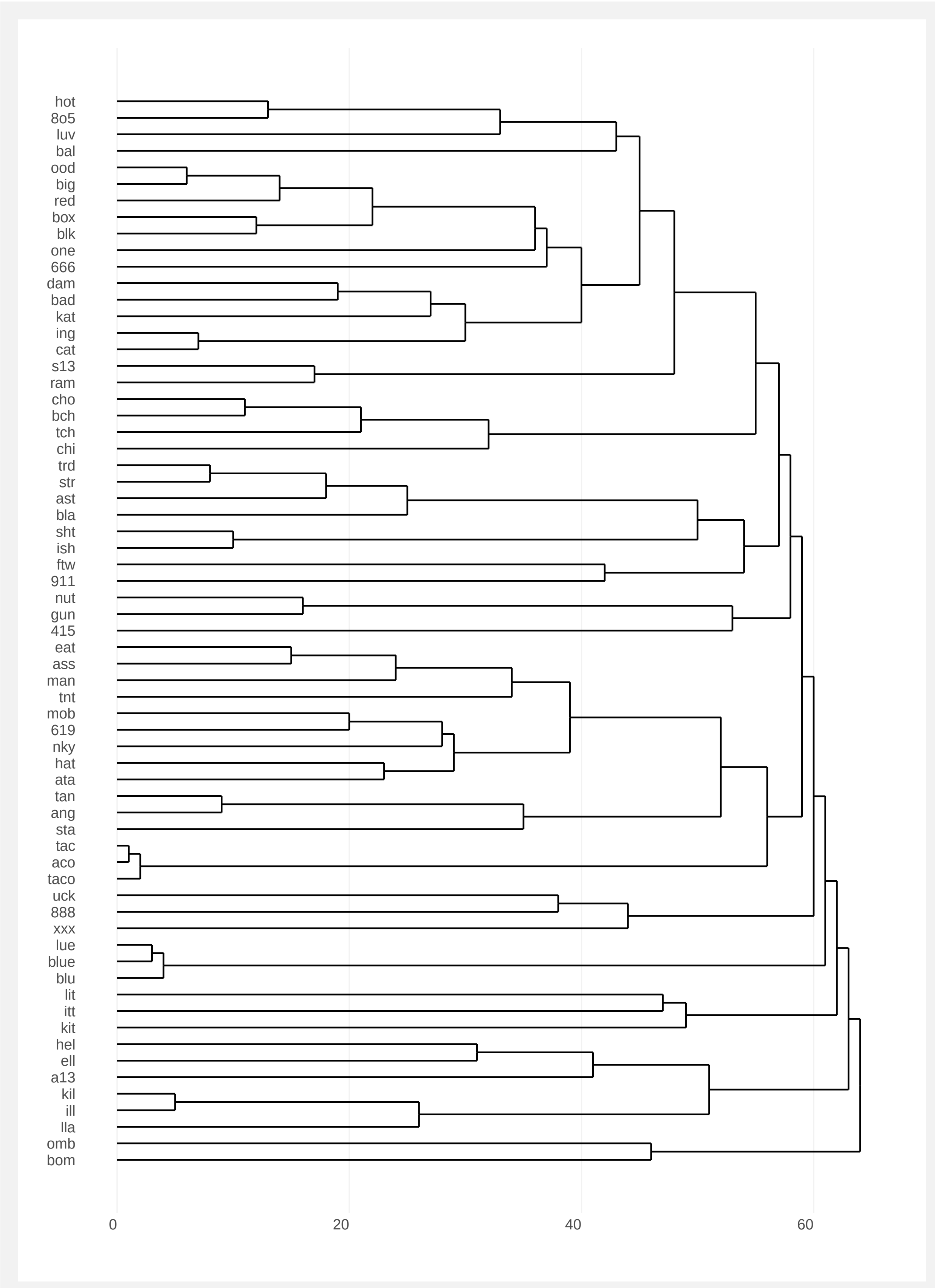


There doesn't appear to be any clear groupings It's important to view this graph a few different times with various random seeds. There's many correct ways to visualize the same graph and sometimes you can draw the wrong conclusions from a single visualization.

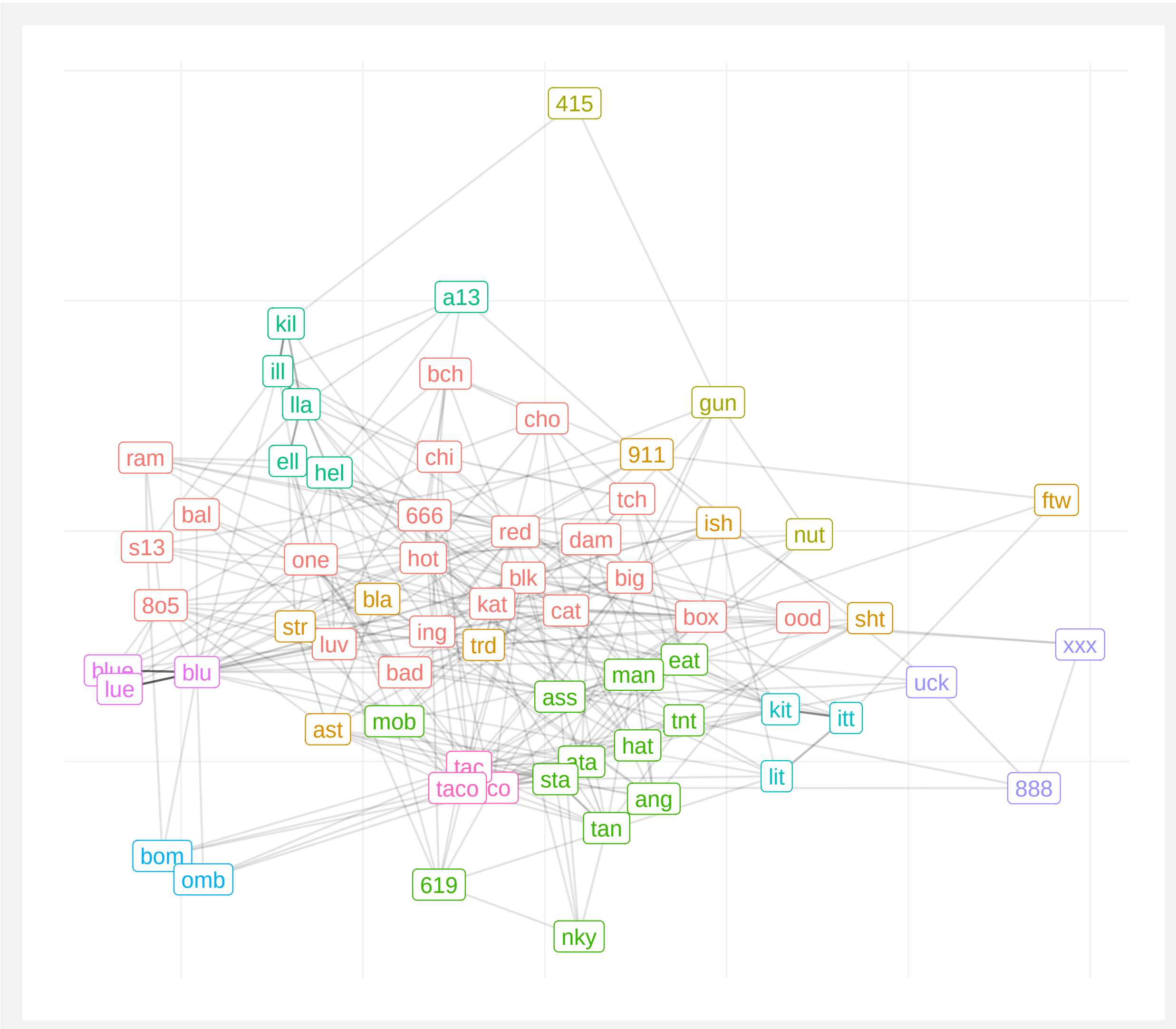
Clustering

Since we already built a 'network' using the cosine similarity we can cluster this network using the walktrap algorithm and see if there are any large groupings or communities. Hopefully, there will be materially differences in the communities and some will have materially higher license plate rejection rates.

```
topics <- walktrap_topics(ngram.graph)

ggdendro::ggdendrogram(topics$dendrogram,
                        rotate = TRUE) +
  theme(panel.grid.major.y = element_line(color = NA))
```


hot
8o5
luv
bal
ood
big
red
box
blk
one
666
dam
bad
kat
ing
cat
s13
ram
cho
bch
tch
chi
trd
str
ast
bla
sht
ish
ftw
911
nut
gun
415
eat
ass
man
tnt
mob
619
nky
hat
ata
tan
ang
sta
tac
aco
taco
uck
888
xxx
lue
blue
blu
lit
itt
kit
hel
ell
a13
kil
ill
lla
omb
bom
0
20
40
60

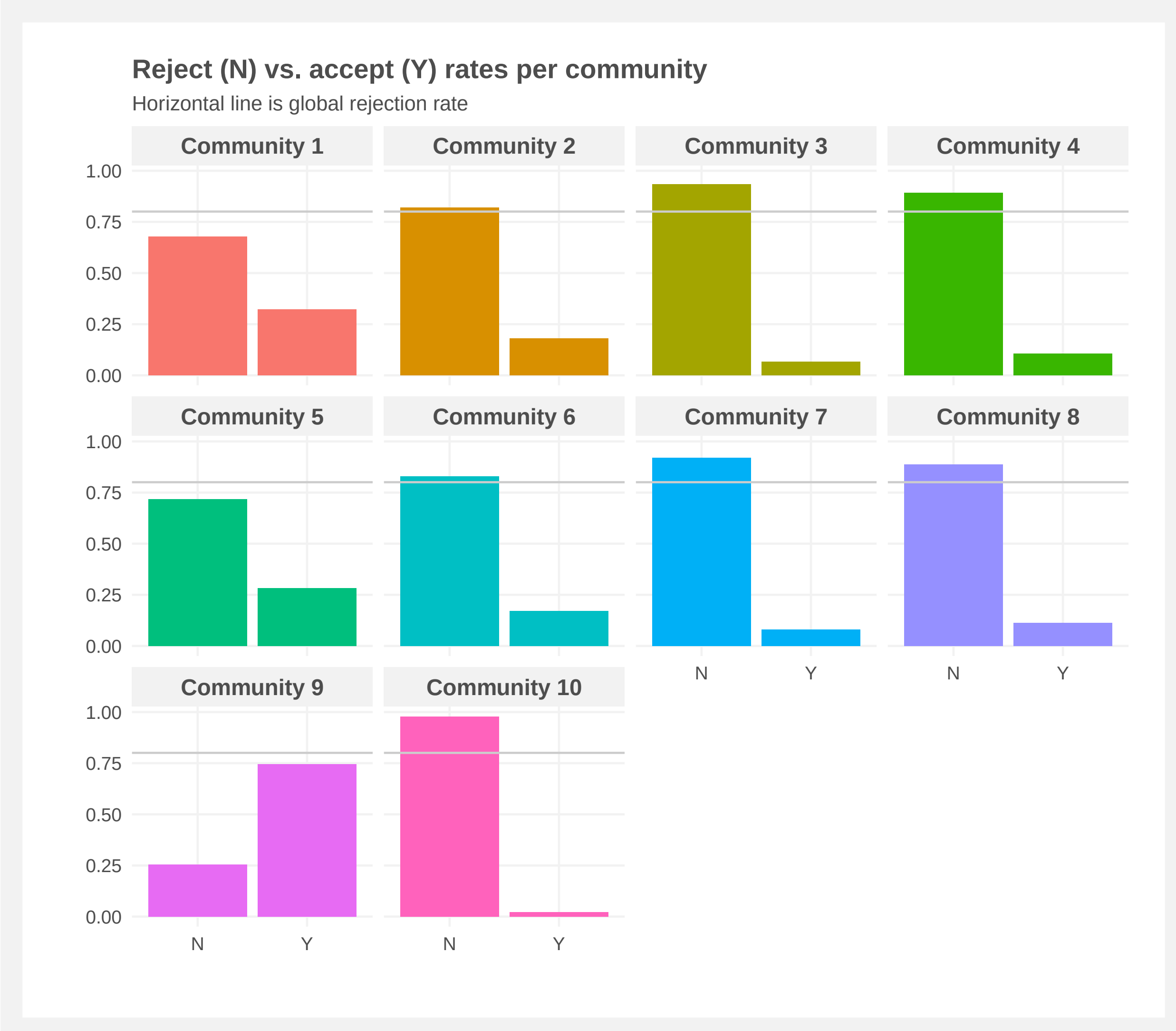
```r
set.seed(44)
# grab the identifiers for the clusters
V(ngram.graph)$cluster <- arrange(topics$membership, word)$group

# plot the matrix but color by cluster
ngram.graph %>%
  ggraph(layout = "nicely") +
  geom_edge_link(aes(alpha = weight), show.legend = FALSE) +
  geom_node_label(aes(label = name,
                      color = factor(cluster)),
                  show.legend = FALSE)
```



There still doesn't seem to be any clear separation among the communities. There still may be differences between the communities based on plate rejection, though. I.e. do some communities consist of mostly 'bad words' and therefore are being rejected at higher rates?

```r
# frequency of denials vs. approved by cluster
plate.ngrams %>%
  filter(word %in% colnames(cos.mat)) %>%
```

```r
  inner_join(app.plates, by = "id") %>%
  select(id, word, status) %>%
  inner_join(topics$membership, by = "word") %>%
  rename(cluster = group) %>%
  # group_by(cluster) %>%
  count(cluster, status) %>%
  group_by(cluster) %>%
  mutate(n = n/sum(n)) %>%
  ungroup() %>%
  mutate(cluster = factor(paste0("Community ", cluster),
                          levels = paste0("Community ",
                                          1:max(topics$membership["group"])))) %>%
  ggplot(aes(x = status, y = n, fill = cluster)) +
  geom_col() +
  geom_hline(yintercept = mean(app.plates$status == "N"),
             color = "grey80") +
  facet_wrap(~cluster) +
  labs(title = "Reject (N) vs. accept (Y) rates per community",
       subtitle = "Horizontal line is global rejection rate",
       x = "",
       y = "") +
  theme(legend.position = 'none')
```



Almost all the communities match the global rejection rate except for community 9, however, the magnitude is not great enough to say it clearly differentiates. The conclusion is that the cosine similarity measure and resulting clustering are not separating the ngrams (and therefore license plates) on rejection rate, nor do we see any other trends that are worth exploiting.

The list of bad words

The second method to explore is comparing the ngrams to a list of pre-determined 'bad words.' I've taken these lists from this kaggle post and this github repo, combined, and then removed the duplicates.

Comparisons can be made in a few different ways. First, we can directly compare by seeing if the ngram is in the other list. Second, we can use various string distance algorithms to see if there is a match in the list. Hypothetically, this should capture words like 'sht' because it will be a close distance to 'shit' which is contained in the `bad.words` list. The soundex phoentic algorithm is especially good at this.

```
bad.words <- read_delim("Data/bad_words.txt",
                        delim = "\n",
                        col_names = FALSE) %>%
  pull()
```

```r
# test if there is a perfect match, and calculate two string distance measures
#   between the ngram and it's best match in the bad.words list
plate.ngrams <- plate.ngrams %>%
  select(word) %>%
  distinct() %>%
  mutate(perfect.match = word %in% bad.words) %>%
  rowwise() %>%
  mutate(soundex = stringsim(word, bad.words, method = 'soundex') %>% max(),
         osa = stringsim(word, bad.words, method = 'osa') %>% max()) %>%
  ungroup() %>%
  right_join(plate.ngrams, by = 'word') %>%
  select(id, plate, word, perfect.match, soundex, osa)

head(plate.ngrams) %>% kable()
```

id	plate	word	perfect.match	soundex	osa
1	AZIZ714	azi	FALSE	1	0.7500000
1	AZIZ714	aziz	FALSE	1	0.5000000
1	AZIZ714	ziz	FALSE	0	0.6666667
1	AZIZ714	ziz7	FALSE	0	0.5000000
1	AZIZ714	iz7	FALSE	1	0.5000000
1	AZIZ714	iz71	FALSE	1	0.4000000

We now have string distance scores for each ngram within each plate. Some ngrams for a given plate are meaningless, though. E.g. the plate 'hello' would contain ngrams 'hell' which is bad and 'lo' which is meaningless. How would we score this plate? We have to score the plates on some aggregate measure: the average, median, maximum, or some percentile. We can calculate these scores and then see if any correlate well with the rejection rate.

```r
# join the data set back to original so we get the `status` variable
#  then summarize the string dist results by plate
summ.ngrams <- plate.ngrams %>%
  left_join(app.plates[, c("id", "status")], by = 'id') %>%
  mutate(status = recode(status, Y = "Plate approved", N = "Plate rejected")) %>%
  pivot_longer(cols = c("soundex", "osa", "perfect.match"), names_to = "algo") %>%
  group_by(id, status, algo) %>%
  summarize(Maximum = max(value),
            Nintieth.percentile = quantile(value, .90),
            Fiftieth.percentile = quantile(value, .50),
            Twentieth.percentile = quantile(value, .10)) %>%
  ungroup()

# plot the summarizde results
summ.ngrams %>%
  pivot_longer(cols = 4:7) %>%
  mutate(name = factor(name,
                       levels = c("Twentieth.percentile",
                                  "Fiftieth.percentile",
                                  "Nintieth.percentile",
                                  "Maximum"))) %>%
  ggplot(aes(x = value)) +
  geom_density(aes(fill = status), alpha = 0.01) +
```

```r
  geom_density(aes(color = status)) +
  scale_color_manual(values = c("forestgreen", "firebrick2")) +
  scale_fill_manual(values = c("forestgreen", "firebrick2")) +
  facet_grid(algo~name) +
  labs(title = "String distance scores from each plate to a list of bad words",
       subtitle = "Scores are the X percentile of all ngrams grouped by plate",
       x = NULL,
       y = 'Density') +
  theme(legend.title = element_blank(),
        legend.position = "bottom")
```


```r
# cross tab of the thresholds and the rejection rate
xtabs(~ status + (Maximum > 0.5) + algo, data = summ.ngrams)
```

```
## , , algo = osa
##
##                   Maximum > 0.5
## status             FALSE  TRUE
##   Plate approved    755   3918
```

```
##   Plate rejected  3026 15724
##
## , , algo = perfect.match
##
##                  Maximum > 0.5
## status            FALSE  TRUE
##   Plate approved  4383   290
##   Plate rejected 16916  1834
##
## , , algo = soundex
##
##                  Maximum > 0.5
## status            FALSE  TRUE
##   Plate approved   408  4265
##   Plate rejected  1504 17246
```

```
xtabs(~ status + (Nintieth.percentile > 0.5) + algo, data = summ.ngrams)
```

```
## , , algo = osa
##
##                  Nintieth.percentile > 0.5
## status            FALSE  TRUE
##   Plate approved   766  3907
##   Plate rejected  3044 15706
##
## , , algo = perfect.match
##
##                  Nintieth.percentile > 0.5
## status            FALSE  TRUE
##   Plate approved  4639    34
##   Plate rejected 18548   202
##
## , , algo = soundex
##
##                  Nintieth.percentile > 0.5
## status            FALSE  TRUE
##   Plate approved   680  3993
##   Plate rejected  2462 16288
```

```
xtabs(~ status + (Fiftieth.percentile > 0.5) + algo, data = summ.ngrams)
```

```
## , , algo = osa
##
##                  Fiftieth.percentile > 0.5
## status            FALSE  TRUE
##   Plate approved  3314  1359
##   Plate rejected 12885  5865
##
## , , algo = perfect.match
##
##                  Fiftieth.percentile > 0.5
## status            FALSE  TRUE
##   Plate approved  4672     1
##   Plate rejected 18750     0
##
```

```
## , , algo = soundex
##
##                  Fiftieth.percentile > 0.5
## status            FALSE  TRUE
##   Plate approved  2768  1905
##   Plate rejected 10158  8592
```

Based on the cross tabs, the algorithms are performing better than the simple 'check if its in the list' method. Taking the maximum score per plate of the soundex algo appears to be the best method. However, it's still "falsely approving" 4,200 plates and "falsely rejecting" 1,500.

```
...
```

. . . continuing