House of Representatives Vocabulary Analysis

By Eoin Morgan and Andy Quach

The Problem

The problem we are examining is what subjects the United States House of Representatives references most frequently. We do this by using `openNLP` libraries to extract all of the proper nouns from the House's floor proceedings XML log. We can then compare the relative frequencies of each proper noun to learn more about what kind of subjects the House of Representatives is talking about. Keep in mind this data only covers the January 6 - December 1, 2015 period.

The Data

Most of the floor proceeding description data is encoded deep within the XML tree heirarchy. We use the `XML` package to parse the XML and obtain the string value descriptions. We pass these string values through to a series of openNLP annotator objects, which utilize a pre-compiled classification model for the English language to associate parts of speach (POS) with each element (word) of the string value. Once we have the POS, we filter over the annotation results and keep only words tagged as proper nouns.

In order to facilitate a fast string-based lookup table to store and update the frequencies of each word, we chose to repurpose the R "env" class. "Env" variables already use a hash-based lookup system for variable names, so it was fairly simple to repurpose the class to act as a more familiar hash table with the `increment()` method in `EnvFunctions.R`. This allowed us to efficiently store and modify the noun frequencies in memory without a linear lookup time for each new incoming word - which initally made the code too slow to use practically. You can see this process in action in `./code/DataPrep.R`. To recreate this process, set the following block to `eval = TRUE`. Be warned; it requires some openNLP packages to be installed, and could take up to 3 hours depending on your machine. This is a big data set!

```
source("./code/DataPrep.R")
```

We knew that we wanted to leverage the expressive power of R in our analysis, which would not have been easy with the noun frequencies stored in a "env" class object. Instead, we wrote the `exportEnvCSV()` method in `EnvFunctions.R` to write the frequencies to file in a standard CSV format. This serves two purposes; it separates the "prep" and "analysis" portions of our scripts, and it allows the data to be easily imported into a data frame object. Once we have a data frame object, we can create exploratory plots using the "ggplot2" package to visualize the data.

We can import these frequencies as a data frame and take a look at the most frequent terms:

```
source("./code/StaticVars.R")

noun_df = read.csv(clean_data_file_name, header = TRUE, sep = "\t",
                   stringsAsFactors = FALSE)
noun_df = noun_df[order(noun_df$count, decreasing = TRUE),]
head(noun_df, n = 10)
```

```
##           name count
## 1123     House  3873
## 1579       Mr.  2107
## 64      Agreed  1839
```

```
## 1978       Res  1563
## 1040        H.  1533
## 465  Committee  1532
## 373      Chair  1382
## 1042      H.r.  1365
## 1888  Pursuant  1197
## 2209   Speaker   924
```

We can see that openNLP did not do a perfect job tagging POS. This is an unfortunate limitation of the technology combined with the format of our data. openNLP is typically used for a more literate corpus (or "text body") of data. The frequent abbreviations, combined with a capitalization scheme that doesn't match well with the precompiled models for the English language, introduce some innaccuracies into our data set. It seems like openNLP tagged some abbreviations as proper nouns. This does give us an opportunity to examine some data points we weren't expecting. An interesting trend to notice is that "Mr." is among the top 10 records. "Mrs." and "Ms." combined don't do so well...

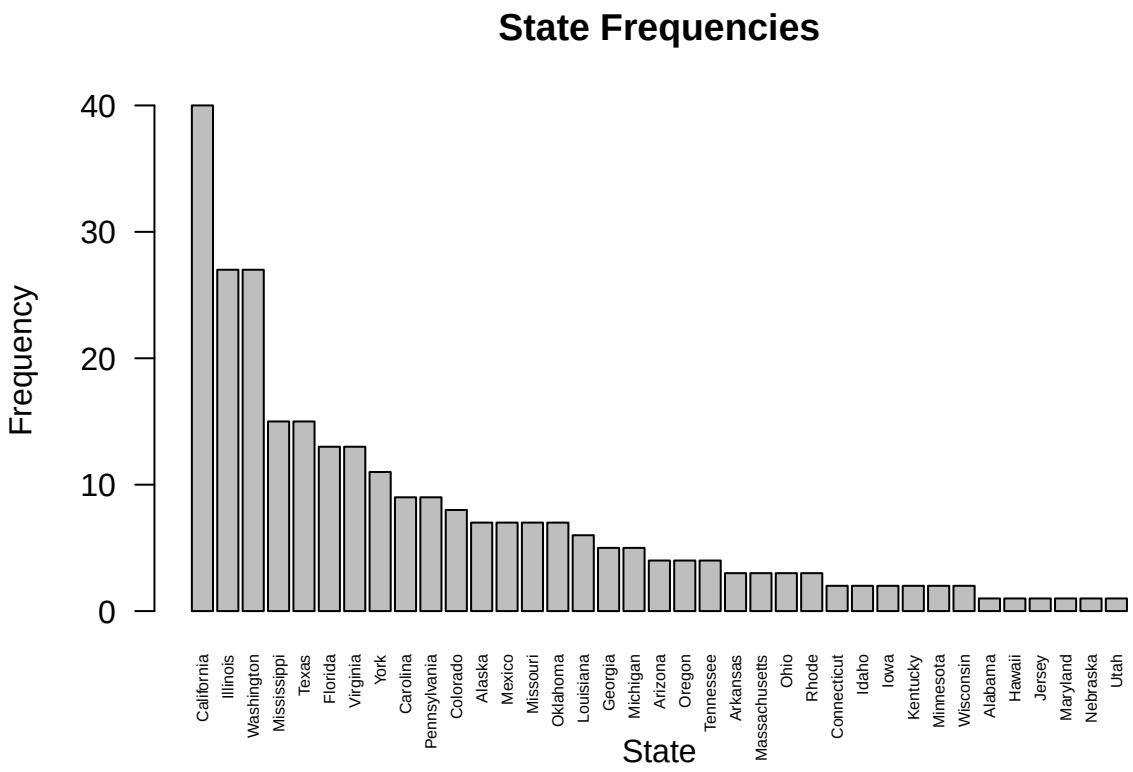
```
noun_df[noun_df$name == "Mr." | noun_df$name == "Mrs." | noun_df$name == "Ms.",]
```

```
##      name count
## 1579  Mr.  2107
## 1583  Ms.   356
## 1581 Mrs.   120
```

It seems like the House spends a lot more time talking about men than women...

Lets see the relative frequencies of how states are mentioned. Notice that due to how openNLP tags words, we can only use single-word references to the states ("Hampshire" vs "New Hampsire"). This means we can't differentiate between North/South states (the Virginias), or Washington vs Washington, D.C. It also introduces some false positives surround Mexico vs New Mexico.

```
states_df = noun_df[noun_df$name %in% states,]
barplot(states_df$count, main = "State Frequencies", horiz = FALSE,
        names.arg = states_df$name, cex.names=.5, las = 2,
        xlab = "State", ylab = "Frequency")
```



We observe that California is mentioned significantly more frequently than any other state. The fact that Washtington is #3 can be attributed to the fact that the Floor proceedings frequently mention Washington, D.C. Due to the limitations of our openNLP processing mode, both Washington and Washington, D.C. map to the same frequency value. The same issue applies to North/South Carolina and Virginia vs West Virginia.

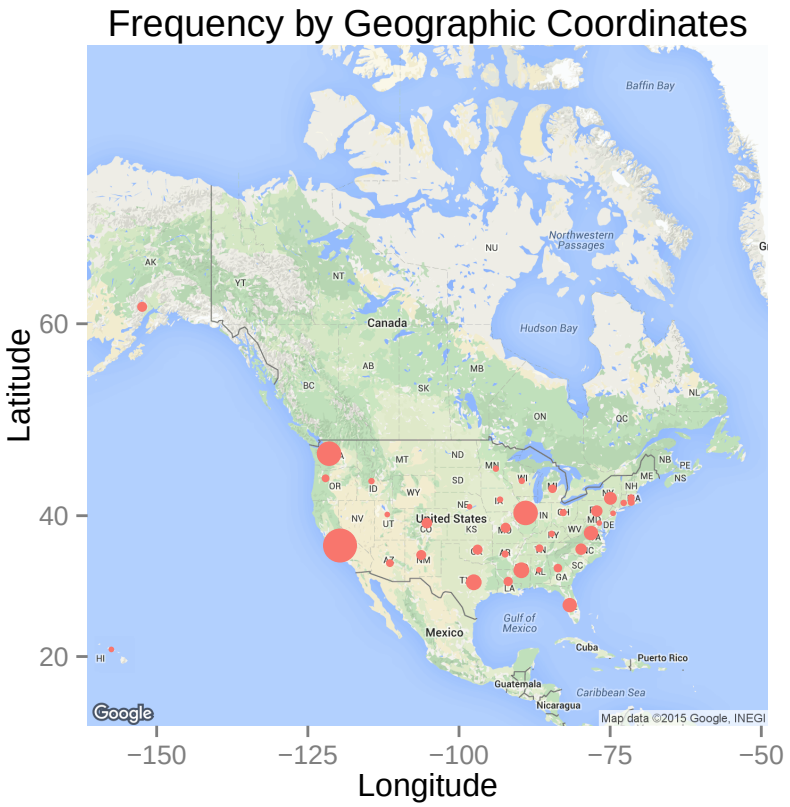
Let's see how those states line up geographically. We can add average latitude and longitude columns to the data frame and construct a weighted scatterplot, using a Google Maps image as the background and a geographic coordinate system. The latitudes and longitudes were found at https://inkplant.com/code/state-latitudes-longitudes.

```
# assign correct lat/lon values
states_lat = numeric(length(states_df$name))
states_lon = numeric(length(states_df$name))
for (i in 1:length(states_df$name)) {
  state_index = match(states_df$name[i], states)
  states_lat[i] = lat[state_index]
  states_lon[i] = lon[state_index]
}
states_df = cbind(states_df, latitude = states_lat, longitude = states_lon)

library(ggplot2)
library(ggmap)
# download background map image
map<-get_map(location='north america', zoom=3, maptype = "roadmap",
             source='google',color='color')
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=north+america&zoom=3&size=640x64
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=north%20america&sens
```

```
fill_color = rgb(1, 0, 0, 0.5)
outline_color = rgb(1, 0, 0, 1)

p = ggmap(map) + geom_point(aes(x = longitude, y = latitude,
              size = count, fill = fill_color, color = outline_color),
              data=states_df, scale = 1, guide = FALSE)
p = p + labs(x = "Longitude", y = "Latitude")
p = p + ggtitle("Frequency by Geographic Coordinates")
p = p + theme(legend.position = 'none')
print(p)
```

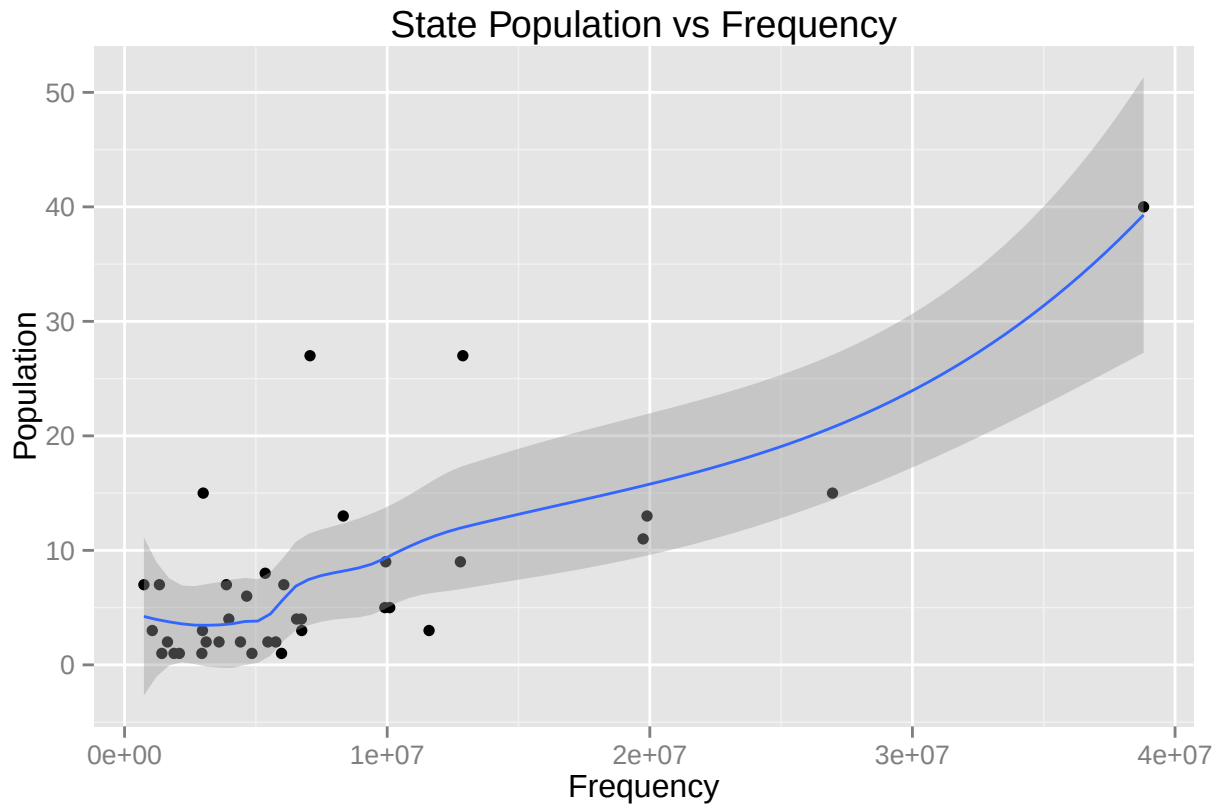


We can see that many Midwestern states are completely ignored. The East Coast has fairly even coverage, but the largest single points are located on the West Coast (keep in mind our potential "Washington" error).

Finaly, let's examine how the frequencies of state references compare to state populations according to the 2014 US Census data. Similarly to our geographic analysis, we can add a population column to our states data frame and use that and the frequency as axes in a scatterplot.

```
states_pop = numeric(length(states_df$name))
for (i in 1:length(states_df$name)) {
  state_index = match(states_df$name[i], states)
  states_pop[i] = state_populations[state_index]
}
```

```
states_df = cbind(states_df, pop = states_pop)
fit_line = lm(states_df$count ~ states_df$pop)
p = ggplot(data = states_df)
p = p + geom_point(aes(y = states_df$count,
                       x = states_df$pop))
p = p + geom_smooth(method = "loess", aes(y = states_df$count,  x = states_df$pop))
p = p + labs(x = "Frequency", y = "Population")
p = p + ggtitle("State Population vs Frequency")
print(p)
```



We can see that there is a signifcant amount of noise in the smoothing model at the lower frequency end of the graph; despite this, there is a general trend linking population and frequency.

Another way to visualize this in a more popular format is to create a "wordcloud" with sizes and colors based on the frequencies of each state. This format would be more appropriate for an artistic display of the data, or presenting the findings to an audience without a scientific or mathematical background that was not comfortable interpreting plots. Younger readers might also relate more strongly with this kind of data presentation because it draws attention quickly, and easily highlights the more prominent trends.

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
colors = c("#ff0000", "#ff9900", "#ffff00", "#66ff33", "#0099ff")
colors = rev(colors)
```

```
wordcloud(states_df$name, states_df$count, min.freq = 5, colors = colors)
```