Final Project EDA

```r
library(mltools)
library(data.table)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(stringr)
library(klaR)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(gapminder)
library(ggplot2)
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.15.2
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------

##
```

```
## Attaching package: 'dendextend'

## The following object is masked from 'package:data.table':
##
##     set

## The following object is masked from 'package:stats':
##
##     cutree
```

```r
data <- read.csv('data/immigration_policies/policy_list.csv')
# summary(data)
```

```r
colSums(is.na(data))[colSums(is.na(data)) != 0]
```

```
##               ISO2           AIR_TYPE        TARGETS_AIR          LAND_TYPE
##                  7               1073               1169               1511
##       TARGETS_LAND           SEA_TYPE        TARGETS_SEA       CITIZEN_LIST
##               1571               1534               1554               1568
##   HISTORY_BAN_LIST       REFUGEE_LIST      VISA_BAN_TYPE      VISA_BAN_LIST
##               1492               1760               1699               1741
## CITIZEN_EXCEP_LIST COUNTRY_EXCEP_LIST
##               1390               1625
```

```r
mod_df <- data.frame(data)

# dropping columns that will not affect our data analysis in any way
mod_df <- mod_df[, -c(32:44)]
colSums(is.na(mod_df))[colSums(is.na(mod_df)) != 0]
```

```
##               ISO2           AIR_TYPE        TARGETS_AIR          LAND_TYPE
##                  7               1073               1169               1511
##       TARGETS_LAND           SEA_TYPE        TARGETS_SEA       CITIZEN_LIST
##               1571               1534               1554               1568
##   HISTORY_BAN_LIST       REFUGEE_LIST      VISA_BAN_TYPE      VISA_BAN_LIST
##               1492               1760               1699               1741
## CITIZEN_EXCEP_LIST COUNTRY_EXCEP_LIST
##               1390               1625
```

```r
colSums(is.na(mod_df))[colSums(is.na(mod_df)) == 0]
```

```
##             ID   COUNTRY_NAME           ISO3    POLICY_TYPE POLICY_SUBTYPE
##              0              0              0              0              0
##     START_DATE       END_DATE            AIR           LAND            SEA
##              0              0              0              0              0
##        CITIZEN    HISTORY_BAN        REFUGEE       VISA_BAN  CITIZEN_EXCEP
##              0              0              0              0              0
##  COUNTRY_EXCEP     WORK_EXCEP
##              0              0
```

```r
# tables to summarize data
# find twelve variables that most interested in, and do correlatin matrix
# if certain variables are very highly correlated, then only use one of the two

# geom jitter -- points won't be laying on top of each other

for (i in 1:length(colnames(mod_df))) {
  column = colnames(mod_df)[i]
  if (sum(is.na(mod_df[, column])) == 0) {
    if (!(column %in% c("ID", "COUNTRY_NAME", "ISO2", "ID", "START_DATE",
                        "END_DATE", "ISO3"))) {
      print(column)
      print(table(mod_df[, column]))
    }
  }
}
```

```
## [1] "POLICY_TYPE"
## 
##            COMPLETE NOPOLICYIMPLEMENTED             PARTIAL 
##                 422                   7                1333 
## [1] "POLICY_SUBTYPE"
## 
##   BORDER_CLOSURE    CITIZEN_EXCEP  CITIZENSHIP_BAN   ESSENTIAL_ONLY 
##              828              177              194               36 
##      HISTORY_BAN             NONE      REFUGEE_BAN SPECIFIC_COUNTRY 
##              245                7                3               79 
##         VISA_BAN       WORK_EXCEP 
##               63              130 
## [1] "AIR"
## 
##    0    1 
## 1073  689 
## [1] "LAND"
## 
##    0    1 
## 1511  251 
## [1] "SEA"
## 
##    0    1 
## 1534  228 
## [1] "CITIZEN"
## 
##    0    1 
## 1568  194 
## [1] "HISTORY_BAN"
## 
##    0    1 
## 1492  270 
## [1] "REFUGEE"
## 
##    0    1 
## 1759    3 
## [1] "VISA_BAN"
## 
##    0    1 
## 1699   63 
## [1] "CITIZEN_EXCEP"
## 
##    0    1 
## 1390  372 
## [1] "COUNTRY_EXCEP"
```

```
## 
##    0    1 
## 1625  137 
## [1] "WORK_EXCEP"
## 
##    0    1 
## 1632  130
```

we know that there are 1762 observations total. we substitute out visa_ban (0 or 1 values) with visa_ban_type, which encapsulates all, specific, or none – we will need to one-hot encode this! other ones to explore: history_ban_list and citizen_list. If I use these, then eliminate history_ban and citizen from consideration (these are values that don't have N/As)

```r
# data cleaning for NA values

## VISA_BAN_LIST

colSums(is.na(mod_df))[colSums(is.na(mod_df)) != 0]
```

```
##               ISO2           AIR_TYPE        TARGETS_AIR          LAND_TYPE 
##                  7               1073               1169               1511 
##       TARGETS_LAND           SEA_TYPE        TARGETS_SEA       CITIZEN_LIST 
##               1571               1534               1554               1568 
##   HISTORY_BAN_LIST       REFUGEE_LIST      VISA_BAN_TYPE      VISA_BAN_LIST 
##               1492               1760               1699               1741 
## CITIZEN_EXCEP_LIST COUNTRY_EXCEP_LIST 
##               1390               1625
```

```r
mod_df$VISA_BAN_NONE <- rep(0, nrow(mod_df))
mod_df[is.na(mod_df$VISA_BAN_TYPE), ]$VISA_BAN_NONE <- 1

mod_df$VISA_BAN_ALL <- rep(0, nrow(mod_df))
mod_df[mod_df$VISA_BAN_TYPE == "All"
       & !is.na(mod_df$VISA_BAN_TYPE), ]$VISA_BAN_ALL <- 1

mod_df$VISA_BAN_SPECIFIC <- rep(0, nrow(mod_df))
mod_df[mod_df$VISA_BAN_TYPE == "specific"
       & !is.na(mod_df$VISA_BAN_TYPE), ]$VISA_BAN_SPECIFIC <- 1
```

```r
## HISTORY_BAN_LIST

# for now, will count the number of commas
# it would be interesting to explore whether certain countries are banned more often than others, but I

# helper function to determine the number of countries
# i.e., number of commas plus one

country_counter <- function(obj) {
  if (is.na(obj)) {
    return(0)
  }
  return ((str_count(obj, ','))[1] + 1)
}

mod_df$HISTORY_BAN_CLEANED <- unlist(lapply(mod_df$HISTORY_BAN_LIST, country_counter))
mod_df$CITIZEN_LIST_CLEANED <- unlist(lapply(mod_df$CITIZEN_LIST, country_counter))
```

for clustering, will use - policy_type, (maybe policy_subtype?) – need to one-hot-encode - length of policy (end_date - start_date) - air, land, sea, refugee, country_excep, work_excep - visa_ban, citizen_list, and history_ban are already covered by the “list" values we are including

```
# data cleaning for non-NA values
colSums(is.na(mod_df))[colSums(is.na(mod_df)) == 0]
```

```
##                   ID         COUNTRY_NAME                 ISO3
##                    0                    0                    0
##          POLICY_TYPE       POLICY_SUBTYPE           START_DATE
##                    0                    0                    0
##             END_DATE                  AIR                 LAND
##                    0                    0                    0
##                  SEA              CITIZEN          HISTORY_BAN
##                    0                    0                    0
##              REFUGEE             VISA_BAN        CITIZEN_EXCEP
##                    0                    0                    0
##        COUNTRY_EXCEP           WORK_EXCEP        VISA_BAN_NONE
##                    0                    0                    0
##         VISA_BAN_ALL    VISA_BAN_SPECIFIC  HISTORY_BAN_CLEANED
##                    0                    0                    0
## CITIZEN_LIST_CLEANED
##                    0
```

```
## DATES
mod_df$START_DATE_CLEANED <- as.Date(mod_df$START_DATE, tryFormats = "%m_%d_%y")
mod_df$END_DATE_CLEANED <- as.Date(mod_df$END_DATE, tryFormats = "%m_%d_%y")
# making assumption that "NA" end date means the policy is still in place
# na values --> setting them equal to today's date
mod_df[is.na(mod_df$END_DATE_CLEANED), ]$END_DATE_CLEANED <- Sys.Date()

# making (possibly faulty assumption) that the ``negative" policy lengths were never in place
# set these values equal to zero
mod_df$POLICY_LENGTH <- difftime(mod_df$END_DATE_CLEANED, mod_df$START_DATE_CLEANED, units = c("days"))
mod_df[mod_df$POLICY_LENGTH < 0 & !is.na(mod_df$POLICY_LENGTH), ]$POLICY_LENGTH <- 0
# no policy implemented will have start date of none --> need to set this to zero as well
mod_df[mod_df$POLICY_TYPE == "NOPOLICYIMPLEMENTED", ]$POLICY_LENGTH <- 0
mod_df$POLICY_LENGTH <- as.numeric(mod_df$POLICY_LENGTH)
```

```
## one-hot encoding the policy type

# 0 --> not implemented, 1 --> partially implemented, 2 --> complete
mod_df$POLICY_TYPE_CLEANED <- rep(0, nrow(mod_df))
mod_df[mod_df$POLICY_TYPE == "PARTIAL", ]$POLICY_TYPE_CLEANED <- 1
mod_df[mod_df$POLICY_TYPE == "COMPLETE", ]$POLICY_TYPE_CLEANED <- 2
```

AT THIS POINT, WE ARE DONE WITH CLEANING. THESE ARE THE VARIABLE NAMES WE WANT TO USE:

ones we’ve cleaned:

VISA_BAN_NONE, VISA_BAN_SPECIFIC, VISA_BAN_ALL, HISTORY_BAN_CLEANED, CITIZEN_LIST_CLEANED, POLICY_LENGTH, POLICY_TYPE_CLEANED

ones we’ve left alone:

AIR, LAND, SEA, REFUGEE, COUNTRY_EXCEP, WORK_EXCEP

```r
# post data cleaning -- need to aggregate by country
vars <- c("COUNTRY_NAME", "ISO3", "VISA_BAN_NONE", "VISA_BAN_SPECIFIC", "VISA_BAN_ALL",
          "HISTORY_BAN_CLEANED", "CITIZEN_LIST_CLEANED", "POLICY_LENGTH",
          "POLICY_TYPE_CLEANED", "AIR", "LAND", "SEA", "REFUGEE",
          "COUNTRY_EXCEP", "WORK_EXCEP")

cleaned_df <- subset(mod_df, select=vars)
```

```r
# hopefully ISO3 can be easily matched with other data sets
by_country <- aggregate(cbind(VISA_BAN_NONE, VISA_BAN_SPECIFIC, VISA_BAN_ALL,
                              HISTORY_BAN_CLEANED,
                              CITIZEN_LIST_CLEANED, POLICY_LENGTH, POLICY_TYPE_CLEANED,
                              AIR, LAND,
                              SEA, REFUGEE, COUNTRY_EXCEP, WORK_EXCEP)~ISO3, cleaned_df, mean)
```

NOW, we can work with the by_country data frame!!!

```r
# kmeans clustering
cluster.results.3 <- kmeans(by_country[,2:ncol(by_country)], 3,
                          iter.max = 10, nstart = 1)

cluster.results.6 <- kmeans(by_country[,2:ncol(by_country)], 6,
                          iter.max = 10, nstart = 1)

kcluster_by_country = data.frame(by_country)
kcluster_by_country$cluster3 <- as.factor(cluster.results.3$cluster)
kcluster_by_country$cluster6 <- as.factor(cluster.results.6$cluster)
```

```r
# hierarchical clustering

dist_mat <- dist(by_country[,2:ncol(by_country)], method = 'euclidean')
hclust_avg <- hclust(dist_mat, method = 'average')

jpeg(file="cluster_den.jpg")
plot(hclust_avg)
dev.off()
```

```
## pdf
##   2
```

```r
cut_avg3 <- cutree(hclust_avg, k = 3)
cut_avg6 <- cutree(hclust_avg, k = 6)

avg_dend_obj <- as.dendrogram(hclust_avg)
avg_col_dend3 <- color_branches(avg_dend_obj, k = 3)
avg_col_dend6 <- color_branches(avg_dend_obj, k = 6)

jpeg(file="cluster_den3.jpg")
plot(avg_col_dend3)
dev.off()
```

```
## pdf
##   2
```

```r
jpeg(file="cluster_den6.jpg")
plot(avg_col_dend6)
dev.off()
```

```
## pdf
##   2
```

```r
jpeg(file="cluster_den_3.jpg")
hcluster_by_country3 <- mutate(by_country, cluster = cut_avg3)
hcluster_by_country6 <- mutate(by_country, cluster = cut_avg6)

hcluster_by_country <- data.frame(by_country)

hcluster_by_country$cluster3 <- as.factor(hcluster_by_country3$cluster)
hcluster_by_country$cluster6 <- as.factor(hcluster_by_country6$cluster)
```

```r
# bringing in demographic data
gdp <- read.csv('data/demographic/gdp.csv')
population <- read.csv('data/demographic/population.csv')
```

```r
master_df_k <- merge(kcluster_by_country, gdp, by.x = "ISO3", by.y = "Code")
master_df_k <- merge(master_df_k, population, by.x = "ISO3", by.y = "Code")
master_df_k <- subset(master_df_k, select = -c(X.x, X.y, Name.x, Name.y))

master_df_h <- merge(hcluster_by_country, gdp, by.x = "ISO3", by.y = "Code")
master_df_h <- merge(master_df_h, population, by.x = "ISO3", by.y = "Code")
master_df_h <- subset(master_df_h, select = -c(X.x, X.y, Name.x, Name.y))
```

```r
# plot(master_df_k$GDP, master_df_k$Pop, col = master_df_k$cluster3)
# plot(jitter(master_df_k$Pop), master_df_k$GDP, pch = 16, col = master_df_k$cluster3)

jpeg(file="kmeans_3.jpg")
p1 <- ggplot(master_df_k, aes(x = Pop, y = GDP, color = cluster3)) + geom_point(size=2)
p1 + ggtitle("K-Means: 3 Clusters") + scale_fill_brewer(palette="Set3")
```

```
## Warning: Removed 22 rows containing missing values (geom_point).
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
jpeg(file="kmeans_6.jpg")
p2 <- ggplot(master_df_k, aes(x = Pop, y = GDP, color = cluster6)) + geom_point(size=2)
p2 + ggtitle("K-Means: 6 Clusters") + scale_fill_brewer(palette="Set3")
```

```
## Warning: Removed 22 rows containing missing values (geom_point).
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
jpeg(file="hac_3.jpg")
p3 <- ggplot(master_df_h, aes(x = Pop, y = GDP, color = cluster3)) + geom_point(size=2)
p3 + ggtitle("HAC: 3 Clusters") + scale_fill_brewer(palette="Set3")
```

```
## Warning: Removed 22 rows containing missing values (geom_point).
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
jpeg(file="hac_6.jpg")
p4 <- ggplot(master_df_h, aes(x = Pop, y = GDP, color = cluster6)) + geom_point(size=2)
p4 + ggtitle("HAC: 6 Clusters") + scale_fill_brewer(palette="Set3")
```

```
## Warning: Removed 22 rows containing missing values (geom_point).
```

```r
dev.off()
```

```
## pdf
##   2
```

goals by next Wednesday: - kMeans cluster on selected variables - hierarchical cluster - (not needed by next Wednesday, but we can vary the number of clusters and where you stop on the dendrogram) – can talk about this as next steps - plot two variables from demographics – then plot the clusters we previously generated (for immigration policies) – this can be a wednesday goal! - can also run the cluster algorithm on the demographics data – does not need to be a wednesday goal - WorldBank, Gap Minder (may have an R package!) – other potential data sets for the demographic - try different distance metrics to see how much the answer changes (how robust is it to that choice?) - k-modes clustering – better suited for categorical data

- see how clusters change with inclusion of different variables

FEEDBACK FROM PRESENTATION:

- log of GDP, population to adjust the scale
- formal tests: 2-sample means on a metric between clusters
- PCA on demographic factors for ease of visualization
- formal test to determine how many clusters there are
- some way to score the different policies, and then see if there is a correlation between that and certain demographic covariates
- find some indicator of "natural" clustering – do we see patterns among certain continents, developed vs developing, etc – then adjust number of clusters based on the number of natural clusters, and see whether the contents of those clusters are the same
- try running PCA on immigration policies (???)