Data Wrangling

Will Lawson

5/11/2020

Packages

I imported the collection of `tidyverse` packages which contains multiple packages used for data wrangling and cleaning.

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(knitr) # tables
```

Raw data

counties.csv - contains features at the county level **deaths__timeseries.scv** - timeseries data of deaths at the county level (1/22 to 4/27)

```
# local paths to two files
counties <- read.csv(file = "../COVID-19_US_County-level_Summaries-master/data/counties.csv") %>%
  as.data.frame()

deaths_timeseries <- read.csv(file =
                                "../COVID-19_US_County-level_Summaries-master/data/deaths_timeseries.cs
  as.data.frame()
```

Aggregation

We want to aggregate the deaths into one variable for our prediction. Note that the numerical data starts on column 3, and ends on the last column (99), representing daily deaths from 1/22/2020 to 4/27/2020.

```
# Showing structure of dataframe
head(colnames(deaths_timeseries))
```

```
## [1] "FIPS"         "Combined_Key" "X1.22.20"     "X1.23.20"     "X1.24.20"
## [6] "X1.25.20"
```

```r
tail(colnames(deaths_timeseries))
```

```
## [1] "X4.22.20" "X4.23.20" "X4.24.20" "X4.25.20" "X4.26.20" "X4.27.20"
```

```r
# Aggergate rows to get total deaths for each row in dataset (column 3 to 99)
total_deaths <- apply(X = deaths_timeseries[,3:ncol(deaths_timeseries)],
                      MARGIN = 1, # row aggregations
                      sum)
```

Combining data

Now we are adding the `FIPS` (county identifier code) variable to this new dataframe, which will allow us to join the two tables together (`total_deaths` and `counties`)

```r
# Turn into df with county code (FIPS) for table joining
total_deaths <- data.frame(FIPS = deaths_timeseries$FIPS,
                           "deaths" = total_deaths)
```

Next, we will join the # of deaths with the `counties` dataframe, which consists of all of the features. Additionally, We are filtering out US and State level aggregations by the `filter()` function.

```r
# Join total_deaths with counties to make full dataset.
# Filter out US/State codes (0, 1000, 2000, etc.).
dat <- full_join(x = counties, y = total_deaths, by = "FIPS") %>%
  filter(FIPS %% 1000 != 0)
```

Removing NA values

This dataset has lots of NA values. We can see specifically how many:

```r
total_NA = 0
for (i in 1:ncol(dat))
{
  # Sum all NA values for a given column
  total_NA = total_NA + sum(is.na(dat[,i]))
}
```

Of the `dim(dat)[1] * dim(dat)[2]` $= 1120560$ datapoints, 1.63586×10^5 of them are NA, which is about 14.6% of the data.

Next, we made a list of the features and how many times a NA value shows up.

```r
# Count of NAs per feature
which_NAs <- vector("list", ncol(dat)) # Pre allocate
which_NAs <- apply(X = dat, MARGIN = 2, is.na) %>%
  apply(MARGIN = 2, sum)

# Df representing how many NA values each feature has
unsorted_NAs <- data.frame("Feature" = colnames(dat),
                           NA.val = which_NAs)
# used for checking category of predictors
```

Feature categories

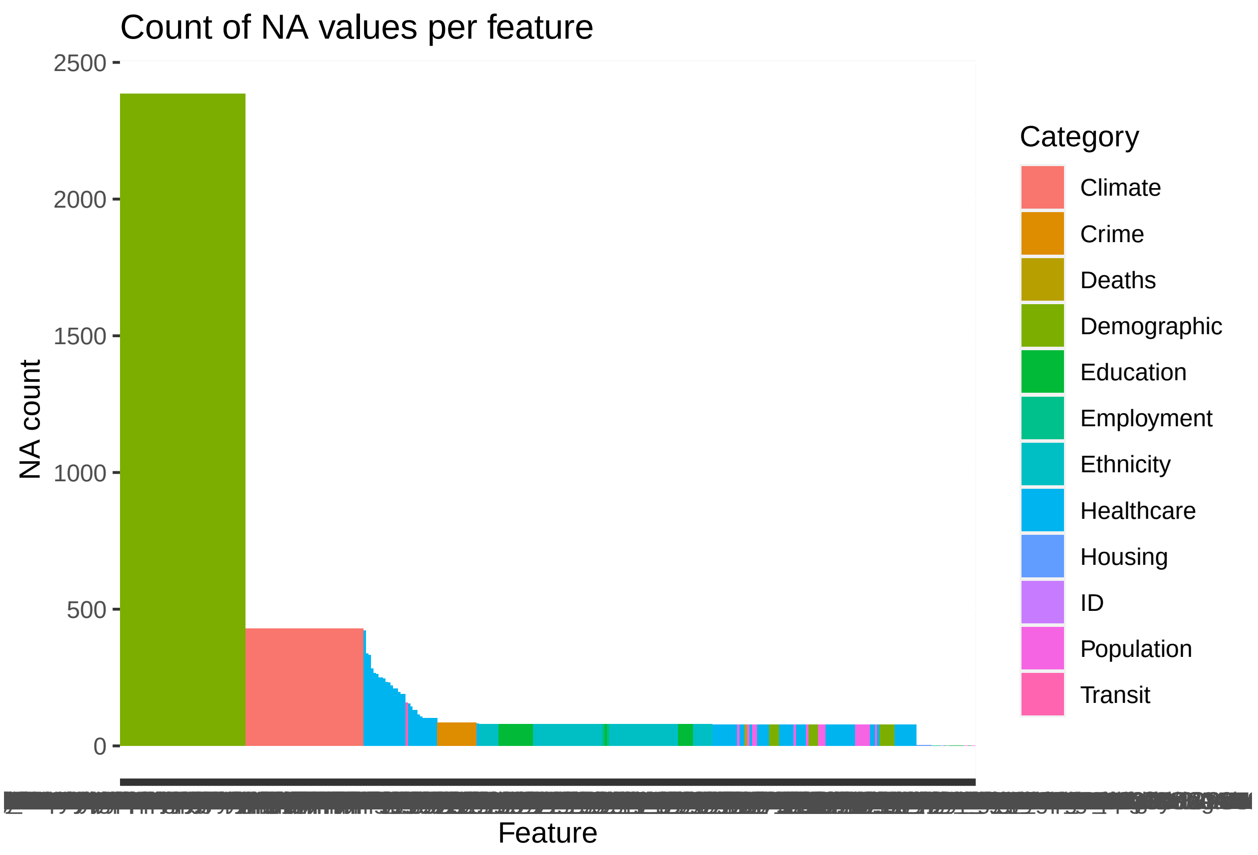
We also wanted to visualize which "category" these features are. This is according to the original databook (link here)

```
##### FEATURE CATEGORIES ACCORDING TO list_of_columns.md #####
# Identifying features
unsorted_NAs[1:3, "Category"] <- "ID"
# Population features
unsorted_NAs[4:22,"Category"] <- "Population"
# Education
unsorted_NAs[23:51, "Category"] <- "Education"
# Employment and median household income
unsorted_NAs[52:57, "Category"] <- "Employment"
# Climate
unsorted_NAs[58:105, "Category"] <- "Climate"
# Housing
unsorted_NAs[106:111, "Category"] <- "Housing"
# Demographic
unsorted_NAs[112:176,"Category"] <- "Demographic"
# Ethnicity
unsorted_NAs[177:248, "Category"] <- "Ethnicity"
# Healthcare
unsorted_NAs[249:330, "Category"] <- "Healthcare"
# Transit
unsorted_NAs[331, "Category"] <- "Transit"
# Crime
unsorted_NAs[332:347, "Category"] <- "Crime"
# deaths/response
unsorted_NAs[348, "Category"] <- "Deaths"

# Convert to factor
unsorted_NAs$Category <- as.factor(unsorted_NAs$Category)
###############################################################
```

```
# Features sorted by # of NA values
NAs <- unsorted_NAs %>%
  arrange(desc(NA.val))

# Plot to visualize
ggplot(data = NAs, aes(x = reorder(Feature, -NA.val), y = NA.val, fill = Category)) +
  geom_col() + labs(title = "Count of NA values per feature",
                    x = "Feature",
                    y = "NA count")
```



We see very clearly that there are a decent amount of features that contain missing values at a constant level (2386 and 429). There are demographic features ms