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.csv
  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"
```

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

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

```
# 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)

```
# 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.

```
# 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:

```
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] $= 1120560$ data points, 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.

```
# 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

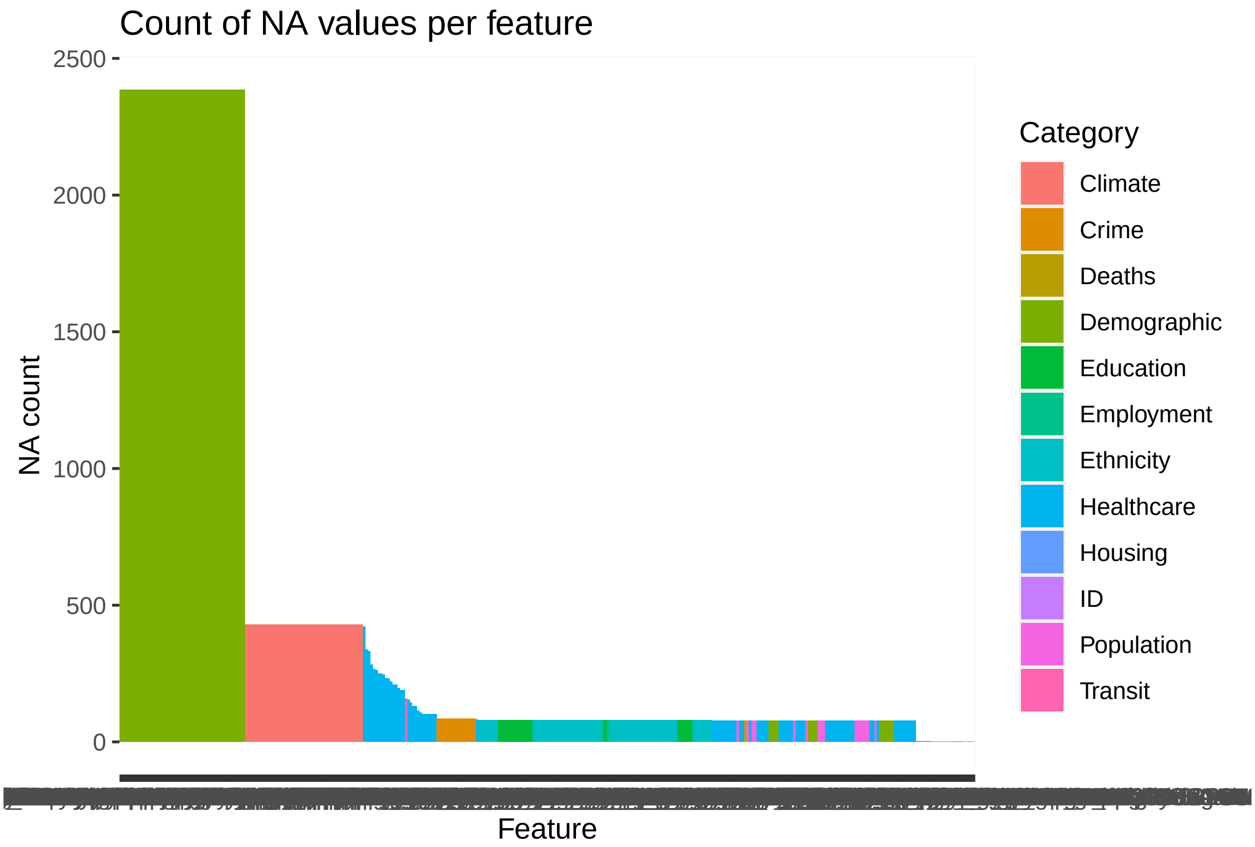
We also wanted to visualize which "category" these features are. This is according to the original databook (link here)

```
##### 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)
###############################################################
```

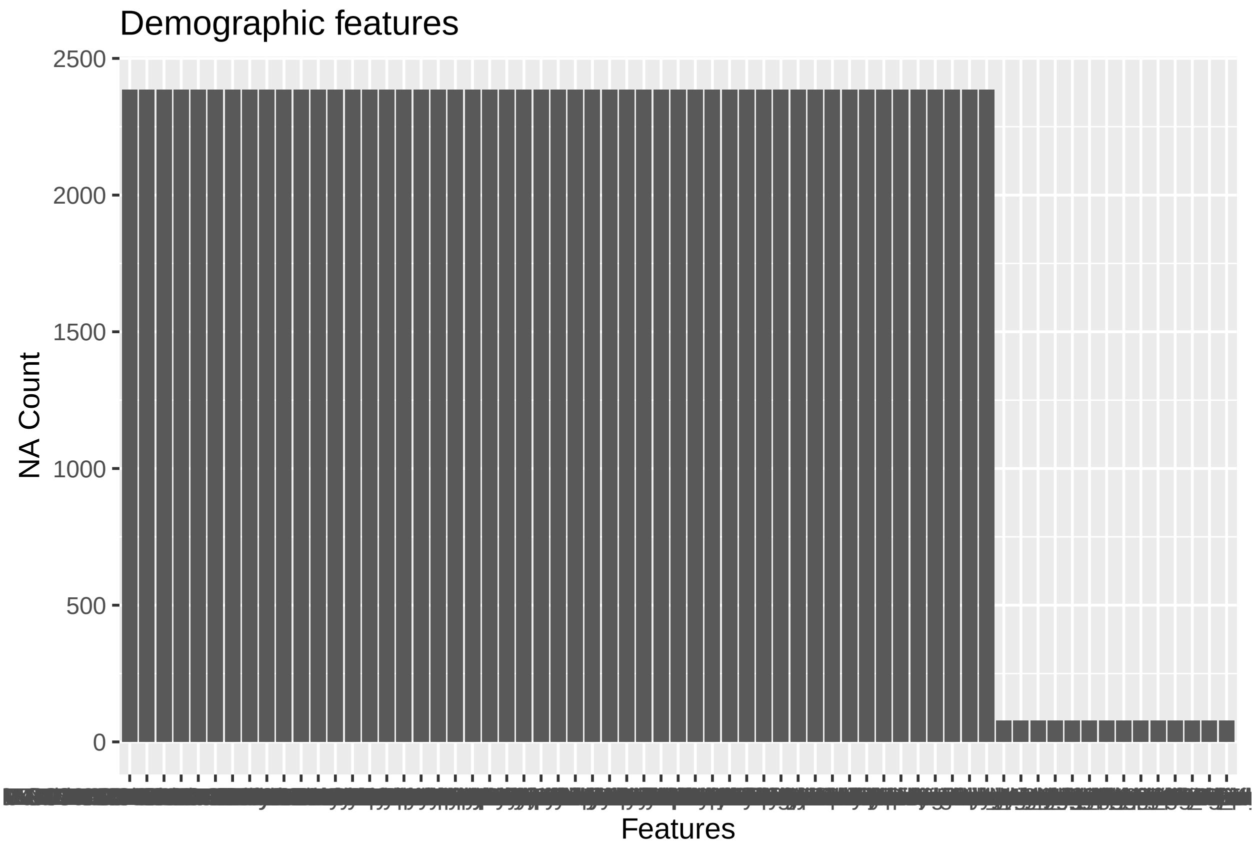
```
# Features sorted by # of NA values
NAs <- unsorted_NAs %>%
  arrange(desc(NA.val))

# 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). These two categories, as we see, are “Demographic” and “Climate” features. We can look somewhat further:

```
# lot all demographic features
ggplot(data = NAs[NAs$Category=="Demographic",], aes(x = reorder(Feature, -NA.val), y = NA.val)) +
  geom_col() + labs(title = "Demographic features", x = "Features", y = "NA Count")
```



It looks like some features are left, so some demographic data is left. This happens to be age data, as shown below:

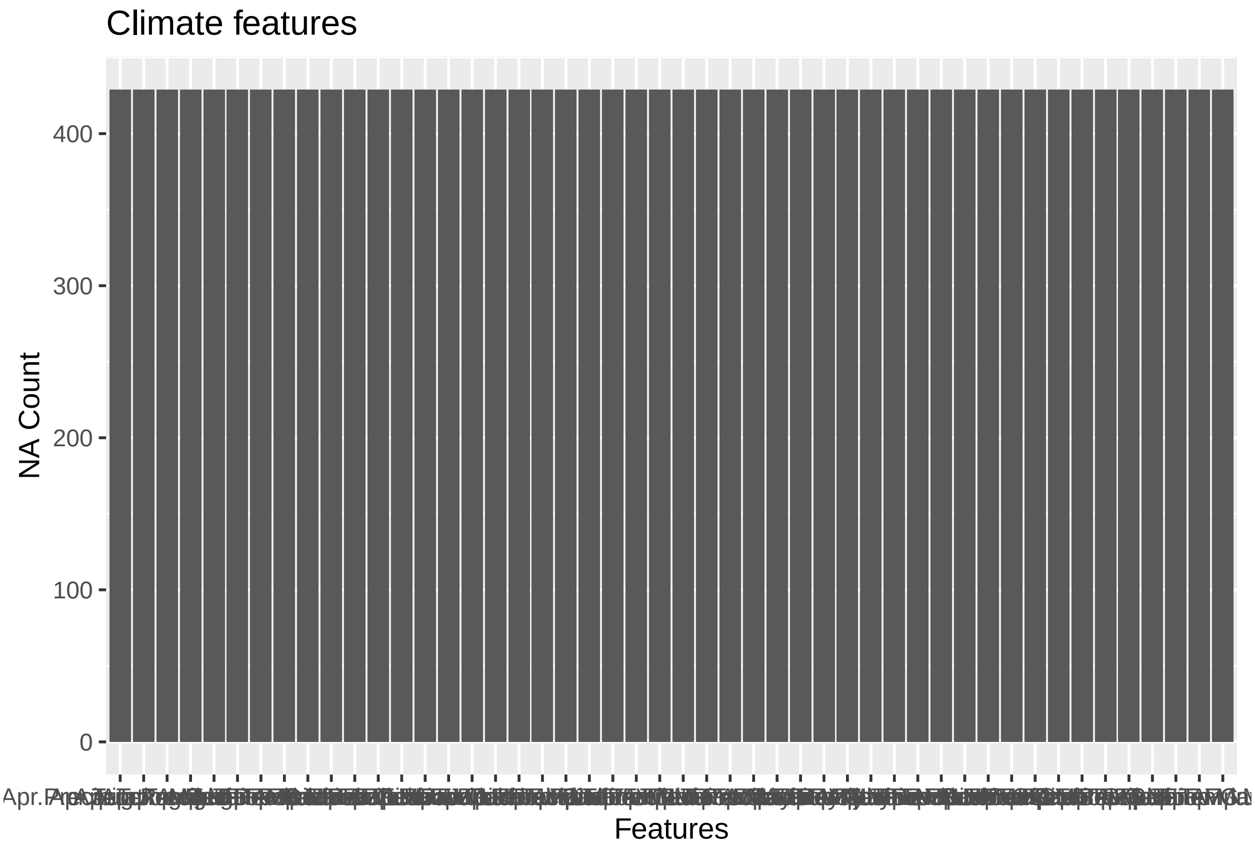
```
NAs[NAs$NA.val < 2000 & NAs$Category=="Demographic",]$Feature
```

```
##  [1] Total_Male        Total_Female      Total_age0to17    Male_age0to17
##  [5] Female_age0to17   Total_age18to64   Male_age18to64    Female_age18to64
##  [9] Total_age65plus   Male_age65plus    Female_age65plus  Total_age85plusr
## [13] Male_age85plusr   Female_age85plusr
## 348 Levels: AA_FEMALE AA_MALE AAC_FEMALE ... WAC_MALE
```

```
# Features that don't have ~ 2000 NA values (ones we will keep)
```

We can do the same process for looking at the missing data for Climate-related features:

```
# # of climate features missing
ggplot(data = NAs[NAs$Category=="Climate",], aes(x = reorder(Feature, -NA.val), y = NA.val)) +
  geom_col()  + labs(title = "Climate features", x = "Features", y = "NA Count")
```



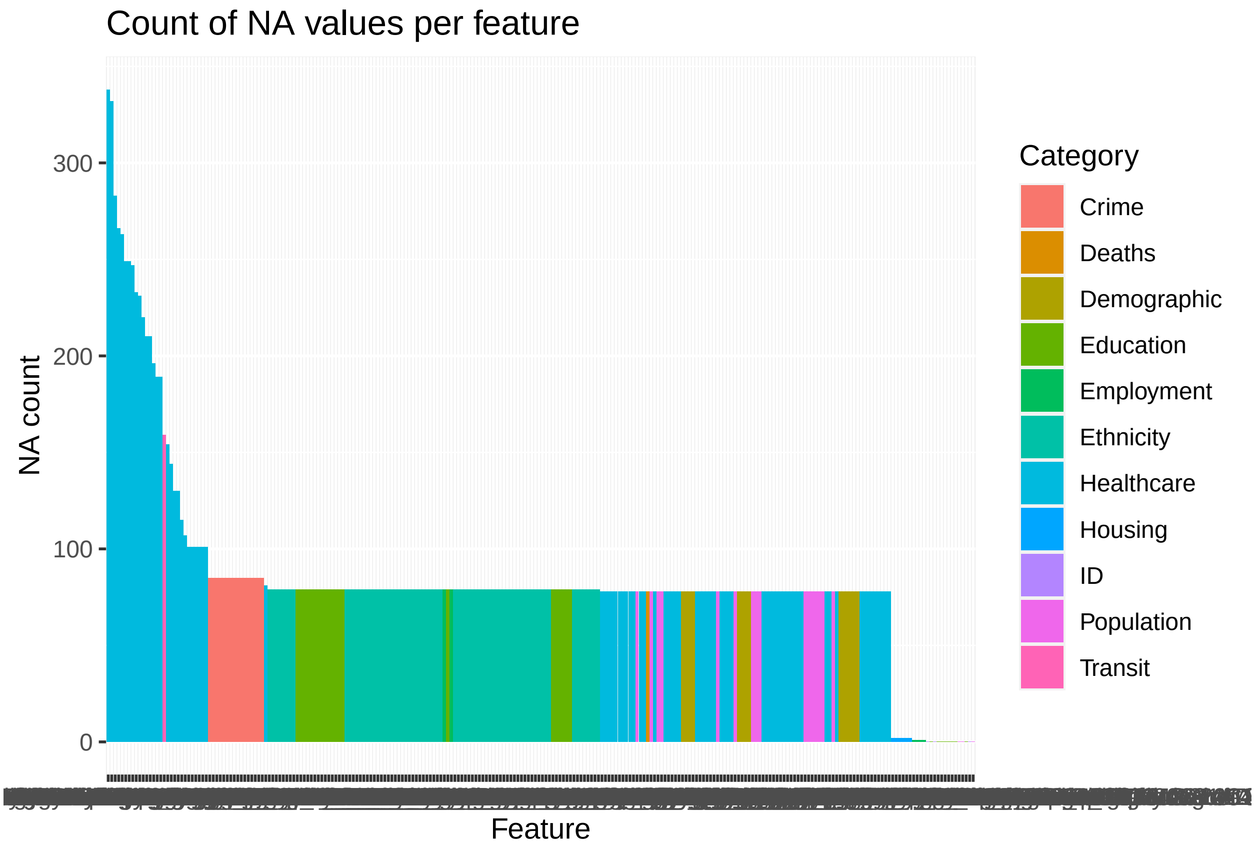
All climate features has lots of missing data (429). We will not keep any cliamate features because of this.

We can use a cutoff of 400 for features to be thrown away initially.

```
# Delete features w/ large # of NA values (> 400)
features_to_be_gone <- filter(NAs, NA.val > 400)$Feature %>% # Feature names with 400 > missing values
  as.vector() # turn to string

# Revise NAs to exclude columns w/ large # of NA valeus
NAs <- NAs[-which(NAs$Feature %in% features_to_be_gone),]

# Plot agian
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")
```

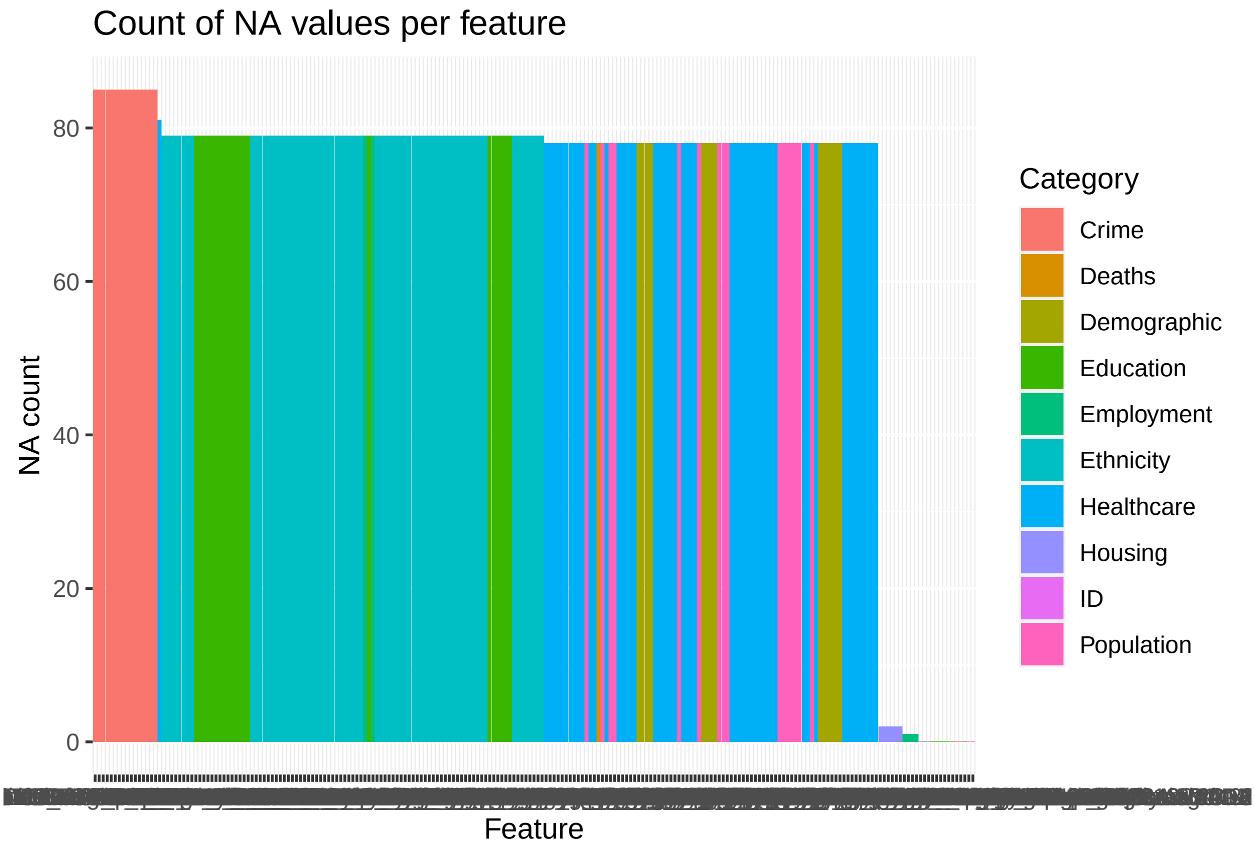


The next cutoff is around 100. We see visually that all of these are healthcare features with the exception of one, transit. We will get rid of all of these, since there are plenty more healthcare features with not as many misisng values. For Transit, the missing values vary significantly from county to county, so imputation would be difficult. For this reason we will not use this feature.

```r
# Delete more features w/ # of NA values > 100
features_to_be_gone <- filter(NAs, NA.val > 100)$Feature %>% # Feature names with 100 > missing values
  as.vector() # turn to string

# Again revise NAs to exclude columns w/ large # of NA valeus (>100)
NAs <- NAs[-which(NAs$Feature %in% features_to_be_gone),]

# Plot agian
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 have now reduced the number of features from 347 to 218 purely through eliminating features with large amounts of NA values.

Updating data

Now we want to update the features in our main dataframe (dat).

```r
# vector of feature names to keep
features_to_keep <- NAs$Feature %>%
  as.vector()
# take only columns that are in features_to_keep
dat <- dat[,which(colnames(dat) %in% features_to_keep)]
```

Removing observations with NA values

Now that we have removed features, we can look to remove certain rows from the dataset that have NA values, since the features left don't contain a ton of rows with NA values.

First, we want to see how many total rows contain at least 1 NA value. We also make a table of states and how many of the rows (counties) belong to a certain state.

```r
# Want to know and visualize how many rows (counties) have NA values
dat$has_NA <- apply(dat, 1, function(x) anyNA(x))

# Total # of rows w/ NA value
sum(dat$has_NA)
```

```
## [1] 91
```

```r
# Distribution of counties/states that have missing rows
table(dat[dat$has_NA==TRUE, "State"])
```

```
##
## AK AL AR AZ CA CO CT DC DE FL GA HI IA ID IL IN KS KY LA MA MD ME MI MN MO MS
##  7  0  0  0  0  0  0  0  3  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## MT NC ND NE NH NJ NM NV NY OH OK OR PA PR RI SC SD TN TX US UT VA VT WA WI WV
##  0  0  0  0  0  0  1  0  0  0  0  0  0 78  0  0  1  0  0  0  0  0  0  0  0  0
## WY
##  0
```

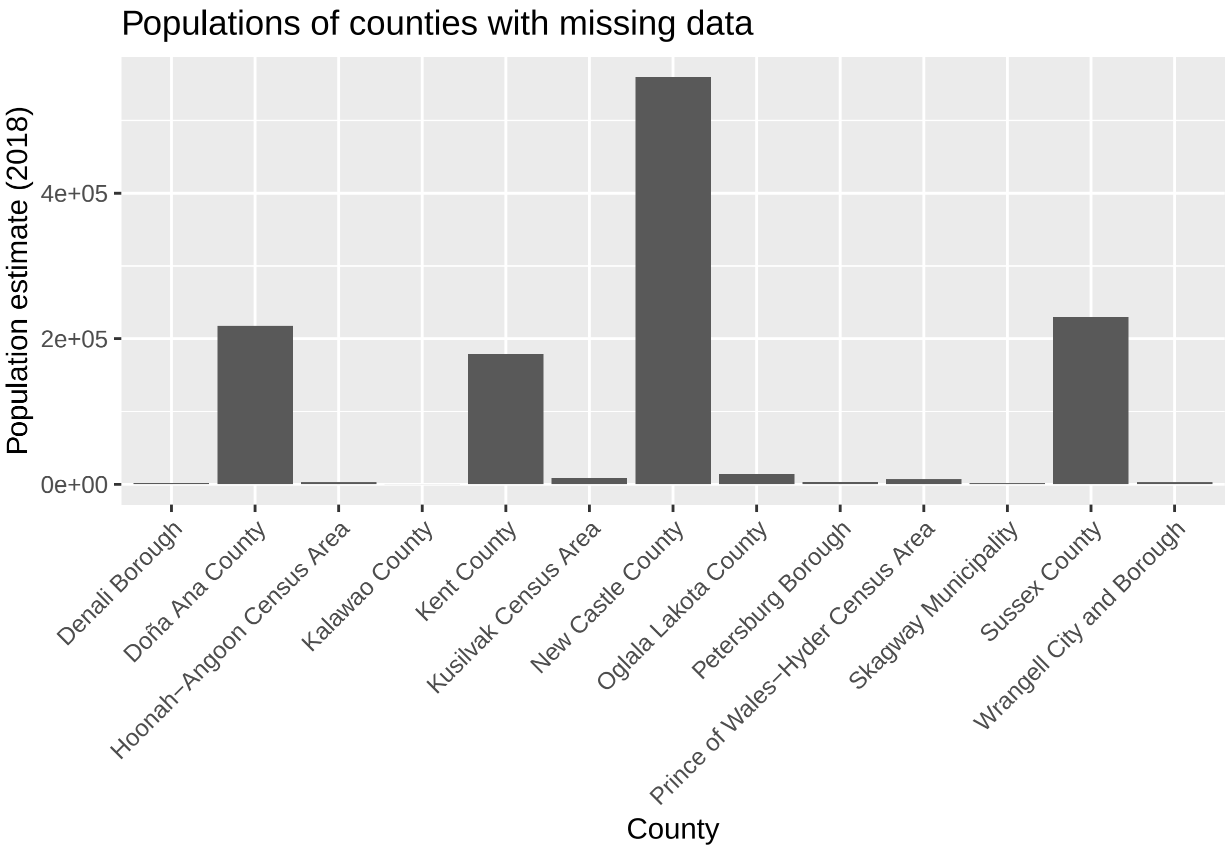
91 rows have missing features, which is not many compared the # of rows/counties we have (3220). A ton of them come from Puerto Rico, which we can take out of the dataset since there are lots of columns that are missing for these observations. This leaves 13 values which we can explore a little bit.

Let's visualize the populations to see if removing them would impact the results:

```r
# Removing 78 observations based on missing data
dat <- filter(dat, State != "PR")

# 13 more counties with missing data
NA_rows_remaining <- filter(dat, has_NA==TRUE)

# Plot the populations to see how influential it will be if deleted
ggplot(NA_rows_remaining, aes(x=Area_Name, y=POP_ESTIMATE_2018)) +
  geom_col() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Populations of counties with missing data",
       x = "County",
       y = "Population estimate (2018)")
```



The only very populous counties are all of Delaware's counties (New Castle County, Sussex County, and Kent County), and Doña Ana County (NM).

Doña Ana County has lots of missing features so we delete it. As for the other three counties, we can see how many features they are missing.

```
NA_rows_remaining <- filter(NA_rows_remaining, Area_Name %in% c("New Castle County",
                                                                "Sussex County",
                                                                "Kent County")==TRUE)


# drop all rows (13) with NA value
dat <- dat[-which(dat$has_NA==TRUE),]
# Join the 3 DE counties
dat <- rbind(dat, NA_rows_remaining)

# Check how many NA values there are
which_NAs <- vector("list", ncol(dat)) # Pre allocate
which_NAs <- apply(X = dat, MARGIN = 2, is.na) %>%
  apply(MARGIN = 2, sum)

sum(which_NAs) # 3 NA values, so there is only 1 column with missing data for the 3 observations of del
```

```
## [1] 3
```

There is only 1 column with missing data for the 3 observations of Delaware. This variable happens to be `Neurological.Surgery..AAMC.`, which we will drop because it varies significantly from area to area.

Additionally, we can drop `has_NA` since these are the last NA values in the dataset.

```r
dat <- dat[-which(colnames(dat) %in% c("Neurological.Surgery..AAMC.", "has_NA"))]
```

We can do a final check to make sure that there are still no NA values in the dataset.

```r
# Check to see if we have any NA values left
anyNA(dat)
```

```
## [1] FALSE
```

We are done cleaning out NA values, so we export it as a new CSV.

```r
write.csv(dat, "./COVID_data.csv", row.names = FALSE)
```