DS-Capstone-Final

A Machine Learning Approach to Predicting County Level COVID-19 Death Rates

Becky Johnson

4/12/2021

Contents

---

1 Introduction

In this project I set out to build a model to predict the county level COVID-19 death rate as defined by the county level deaths divided by county level cases from January 2020 to early April 2021. To do this, I built a data set using COVID-19 data from the New York Times, demographic data from the 5 Year American Community Survey (Census.gov), and health data from "The County Health Rankings & Roadmaps, a program of the University of Wisconsin Population Health Institute". For the model, I used linear regression, cross-validated bootstrap aggregated regression tree and random forest approaches. This project is the final submission for the edX course, "HarvardX PH125.9x Data Science: Capstone".

1.1 The data

For this project, I've created a data set from three sources which are described here. The R code downloads the pre-processed files from my Github repository, but I've provided the links to the original source data.

1.1.1 The US Census Bureau

For demographic data, I've pulled data from the 5-Year American Community Survey. I started this project with little knowledge of the massive data available from census.gov. I started at https://census.gov and navigated to the "Surveys/Programs" section to get an overview of the different programs run by the US Census. Two important data sets to know about are the decennial census where all US households are inventoried and the ongoing American Community Survey.

Decennial Census The Decennial Census, also known as the Population and Housing Census, is mandated by the U.S. constitution (Article 2, Section 2, Clause 3) and takes place every 10 years. It is intended to be a complete count of every person living in the US and is used to determine the number of seats allotted to each state in the U.S. House of Representatives.

ACS The American Community Survey (ACS) is conducted monthly and aggregated annually. The Census Bureau contacts 3.5 million households over the course of each year to create this rich data set of demographic information. The data is broadly used to set policies, allocate resources, study emerging societal trends, assess the effectiveness of programs, and to plan for emergencies. The data is used by government entities at all levels, researchers, businesses, journalists, and educators.

Why the 5-Year ACS Table For purposes of this analysis, I've chosen to used demographic data from the 5-Year American Community Survey. The 5-year estimates are the least current, but because survey results are aggregated over 5 years the 5-year estimates are the most accurate and cover all geographies. Source: https://www.census.gov/content/dam/Census/library/publications/2018/acs/acs_general_handbook_2018_ch03.pdf

Within the 5 year ACS there are choices on which tables. I've chosen to work with Data Profiles which contain broad social, economic, housing, and demographic information. The data are presented as population counts and percentages. There are over 1,000 variables.

The variable names for the ACS are cryptic, but they follow a pattern. These are reasonably well explained here - https://www.census.gov/programs-surveys/acs/guidance/which-data-tool/table-ids-explained.html In general, for each variable name in the ACS the first letter indicates the table type. This is a "B" for base table, "C" for a collapsed table, or "DP" for a Data Profile table. The next two digits are generally a subject identifier. For example, "01" represents Age and Sex. The next three represent a table number within a subject. Some tables end in an alpha numeric character indicating

the metrics represent a particular population group, such as American Indian or non-Hispanic White. This is then followed by an underscore and three digits indicating the line number within a table. The final component is another alpha numeric character, E, M, PE, or PM. These are defined as follows:

- E - Estimate
- M - Margin of Error
- PE - Estimate representing a percent of the total
- PM - Margin of error for a percentage

A note on FIPS codes To match data across the different data sources used for my analysis, I've used FIPS codes. These are numbers which uniquely identify geographic areas and importantly are included in the Census data. The number of digits in FIPS codes vary depending on the level of geography. State-level FIPS codes have two digits, county-level FIPS codes have five digits of which the first two are the FIPS code of the state to which the county belongs. Source: https://transition.fcc.gov/oet/info/maps/census/fips/fips.txt

Downloading Census Data Pulling data from the census site is remarkably easy using the Public API. However, to learn how to do this required some effort and watching a few videos. To help choose which variables I consulted the documentation here - https://api.census.gov/data/2019/acs/acs5/profile/variables.html. Based on news reports I've read over the past year the following variables felt like a good place to start the analysis:

- Percent of families in poverty - DP03_0119PE
- Percent of population over 85 years - DP05_0017PE
- Percent of population between 75 and 84 years - DP05_0016PE
- Health insurance, all - DP03_0096PE
- Health insurance coverage, private - DP03_0097PE
- Household size - DP02_0016E
- Percent of owner occupied housing - DP04_0046PE
- Percent of population that is African American - DP05_0038PE
- Percent of population that is Native American - DP05_0039PE
- Percent of population that is Hispanic or Latino (all races) - DP05_0071PE

1.1.2 New York Times

The New York Times data set on COVID-19 cases and deaths begins with the first reported coronavirus case in Washington State on Jan. 21, 2020. I pulled the county level file in early April and used data from April 1, 2021. To assemble this data the New York Times has compiled information from various state and county level sources across the US. To learn more I recommend the "README.md" file on the data's Github site. In addition to COVID-19 cases and deaths, the New York Times site includes a file on county level mask wearing which I've used as well.

Note Missing FIPS codes: The New York Times data set includes 29 rows for the April 1, 2021 observations for which the FIPS codes are NA. This includes a row for "New York City" which represents the five underlying New York metropolitan counties. For this analysis, the NA rows are discarded when joined using FIPS codes to the Census data. This means, importantly, that the data for New York City is excluded from the analysis.

1.1.3 County Health Rankings & Roadmaps

For county level health related metrics, I used the County Health Ranking & Roadmaps site. The file used is the "2020 CHR CSV Analytic Data" which can be downloaded here:

https://www.countyhealthrankings.org/explore-health-rankings/rankings-data-documentation Documentation can be found here on the data is available here: https://www.countyhealthrankings.org/sites/default/files/media/document/2020%20Analytic%20Documentation_0.pdf Based on my understanding of COVID-19, these were variables chosen for the analysis that seemed likely to be important:

- poor_health = Poor.physical.health.days.raw.value
- smoking = Adult.smoking.raw.value
- obesity = Adult.obesity.raw.value
- sedentary = Physical.inactivity.raw.value
- flu_vac = Flu.vaccinations.raw.value
- air_quality = Air.pollution. . . particulate.matter.raw.value
- overcrowding = Percentage.of.households.with.overcrowding
- diabetes = Diabetes.prevalence.raw.value
- sleep = Insufficient.sleep.raw.value
- doctors = Ratio.of.population.to.primary.care.providers.other.than.physicians.

1.2 Methods Overview

For this analysis, I use data from the New York Times for county level COVID-19 deaths and cases then create a death rate field defined simply as deaths divided by cases for each county. I join the COVID-19 data to census and health metrics to create a data frame with 3,133 counties and 23 descriptive variables. To build a predictive model, I split the data into three subsets - train_set, test_set, and validation_set. The modelling includes a simple "Just the Average" approach, a linear regression, a bootstrap aggregated regression tree, and a random forest approach. The models are measured on a Root Mean Squared Error (RMSE) standard. The final model improves the overall RMSE significantly over "Just the Average".

---

2 Data, Methods, Model Building

This section covers building the data set, exploring the data, splitting the data into train, test, and validation sets, and fitting the models.

2.1 Create the Data set (Step 1 in DS-Capstone-Final.R)

To create the data set we will aggregate data from the New York Times, Census, and County Health Rankings & Roadmaps sites.

2.1.1 Census Data (Step 1a in DS-Capstone-Final.R)

The data has been captured from census.gov and previously stored in the project Github repository. To download from the Census Bureau's site, I followed these steps:

1. Paste the link below into a Chrome browser - https://api.census.gov/data/2019/acs/acs5/profile?get=NAME,DP03_0119PE,DP05_0017PE,DP05_0016PE,DP03_0096PE,DP03_

0097PE,DP02_0016E,DP04_0046PE,DP05_0038PE,DP05_0039PE,DP05_0071PE&for=county:*

2. Save Page As a .csv changing the file type to "All Files"

```
# Download prepared file from my Github repository
dl <- tempfile()
download.file("https://raw.githubusercontent.com/Yowza63/DS-Capstone-Final/main/census.csv", dl)
dat <- read.csv(dl)

# Drop the "X" column
dat <- select(dat, -"X")

# Change the column names to be meaningful
names(dat) <-
  c("name", "poverty", "over85", "age74to85", "insured", "private_insured", "household_size",
    "owner_occ", "african_am", "native_am", "hispanic", "state", "county")

# Remove any "[" or "]" from the name and county fields
dat$name <- gsub("\\[|\\]", "", dat$name)
dat$county <- gsub("\\[|\\]", "", dat$county)

# Create a fips code field made up of the state and county identifiers
dat <- dat %>% tidyr::unite("fips", state:county, sep = "", remove = FALSE)
dat$fips <- strtoi(dat$fips)

# Reorder the columns
dat <- dat[, c("name", "fips", "state", "county", "poverty", "over85", "age74to85", "insured",
  "private_insured", "household_size", "owner_occ", "african_am", "native_am", "hispanic")]
```

2.1.2 New York Times COVID-19 Data (Step 1b in DS-Capstone-Final.R)

Next, incorporate the New York Times COVID-19 cases and deaths to compute the death rate by county. This file was downloaded from the New York Times site and then stored in my Github repository. To download the file I used the r code: download.file("https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-counties-recent.csv", "us-counties-recent.csv"). Because the New York Times files are updated daily, it was important to save one to create repeatability of the analysis.

This file has 29 rows of data for which the FIPS code is "NA" including New York City. These will be ignored when the dat_nyc is joined to dat which is the census data previously downloaded.

```
# Download New York Times file from  Github repository
dl_nyt <- tempfile()
download.file("https://raw.githubusercontent.com/Yowza63/DS-Capstone-Final/main/us-counties-recent.cs
dat_nyt <- as.data.frame(read.csv(dl_nyt)) # load the data
dat_nyt <- dat_nyt %>% dplyr::filter(date == "2021-04-01") # keep only the most recent data

# Examine the rows with NA values in the fips field. This highlights the issue with the New York City
# These rows will be discarded when we join to the census data
dat_nyt[which(is.na(dat_nyt$fips)),]
```

```
nrow(dat_nyt[which(is.na(dat_nyt$fips)),])

# Add a death_rate column in dat_nyt
dat_nyt$death_rate <- dat_nyt$deaths/dat_nyt$cases

# Keep only columns we need
dat_nyt <- select(dat_nyt, fips, cases, deaths, death_rate)

# Add the COVID-19 data to dat using inner_join() to keep only the fips codes that are in both tables
dat <- inner_join(dat, dat_nyt, by = "fips")
```

2.1.3 Incorporate the New York Times mask use information (Step 1c in DS-Capstone-Final.R)

The data on mask wearing is from the site https://github.com/nytimes/covid-19-data and I used the R code - download.file("https://raw.githubusercontent.com/nytimes/covid-19-data/master/mask-use/mask-use-by-county.csv", "mask-use.csv"). Here we are just downloading a previously stored file from my Github repository.

```
# Download stored file from  Github repository
dl_mask <- tempfile()
download.file("https://raw.githubusercontent.com/Yowza63/DS-Capstone-Final/main/mask-use.csv", dl_mas
mask <- as.data.frame(read.csv(dl_mask)) # load the data
mask <- rename(mask, fips = COUNTYFP)
# add two summary columns for "good" and "bad" mask compliance
mask$good = mask$FREQUENTLY + mask$ALWAYS
mask$bad = mask$NEVER + mask$RARELY
# keep just these new columns
mask <- mask %>% select(fips, good, bad)

# Add the mask data to dat
dat <- inner_join(dat, mask, by = "fips")
```

2.1.4 Add county level health statistics (Step 1d in DS-Capstone-Final.R)

This data is from the University of Wisconsin Population Health Institute. To download the original file I selected the file "2020 CHR CSV Analytic Data", from the following site - https://www.countyhealthrankings.org/explore-health-rankings/rankings-data-documentation Documentation for this data can be found here - https://www.countyhealthrankings.org/sites/default/files/media/document/2021%20Analytic%20Documentation.pdf

```
# Download the previously downloaded file stored on my Github repository
dl_chr <- tempfile()
download.file("https://raw.githubusercontent.com/Yowza63/DS-Capstone-Final/main/analytic_data2020_0.c
chr <- read.csv(dl_chr)

# keep only columns we need
chr <- chr %>% select(fips = X5.digit.FIPS.Code,
  poor_health = Poor.physical.health.days.raw.value,
```

```
  smoking = Adult.smoking.raw.value,
  obesity = Adult.obesity.raw.value,
  sedentary = Physical.inactivity.raw.value,
  flu_vac = Flu.vaccinations.raw.value,
  air_quality = Air.pollution...particulate.matter.raw.value,
  overcrowding = Percentage.of.households.with.overcrowding,
  diabetes = Diabetes.prevalence.raw.value,
  sleep = Insufficient.sleep.raw.value,
  doctors = Ratio.of.population.to.primary.care.providers.other.than.physicians.)

# Convert fips to integer
chr$fips = as.integer(chr$fips)

# For these variables we want to join for all available fips codes
dat <- left_join(dat, chr, by = "fips")

# Convert fields to numeric or integer
dat$poor_health = as.numeric(dat$poor_health)
dat$household_size = as.numeric(dat$household_size)
dat$smoking = as.numeric(dat$smoking)
dat$obesity = as.numeric(dat$obesity)
dat$sedentary = as.numeric(dat$sedentary)
dat$flu_vac = as.numeric(dat$flu_vac)
dat$air_quality = as.numeric(dat$air_quality)
dat$overcrowding = as.numeric(dat$overcrowding)
dat$diabetes = as.numeric(dat$diabetes)
dat$sleep = as.numeric(dat$sleep)
dat$doctors = as.numeric(dat$doctors)

# For "doctors", reset negative values and NA's to the mean
dat$doctors[which(dat$doctors < 0)] <- 0
dat$doctors[which(is.na(dat$doctors) == TRUE)] <- 0
dat$doctors[which(dat$doctors == 0)] <- mean(dat$doctors[which(dat$doctors > 0)])

# Replace "flu_vac" NA's with the mean
dat$flu_vac[which(is.na(dat$flu_vac) == TRUE)] <- mean(dat$flu_vac[which(is.na(dat$flu_vac) == FALSE)

# Replace "air_quality" NA's with the mean
dat$air_quality[which(is.na(dat$air_quality) == TRUE)] <- mean(dat$air_quality[which(is.na(dat$air_qu
```

2.2 Exploration (Step 2 in DS-Capstone-Final.R)

The dataframe "dat", has 3133 counties across 29 columns. Each row represents a county. There are four columns that identify the county - fips, name, county, and state. The remaining columns are death-rate defined as total COVID-19 related deaths divided by total cases and 24 features that will be used to build a predictive model.

Lets look at counties with the top 10 highest death rates. The highest death rate is 10.5% for people who've contracted COVID-19, but the number of cases is only 38 so likely an anomaly. In fact, all of

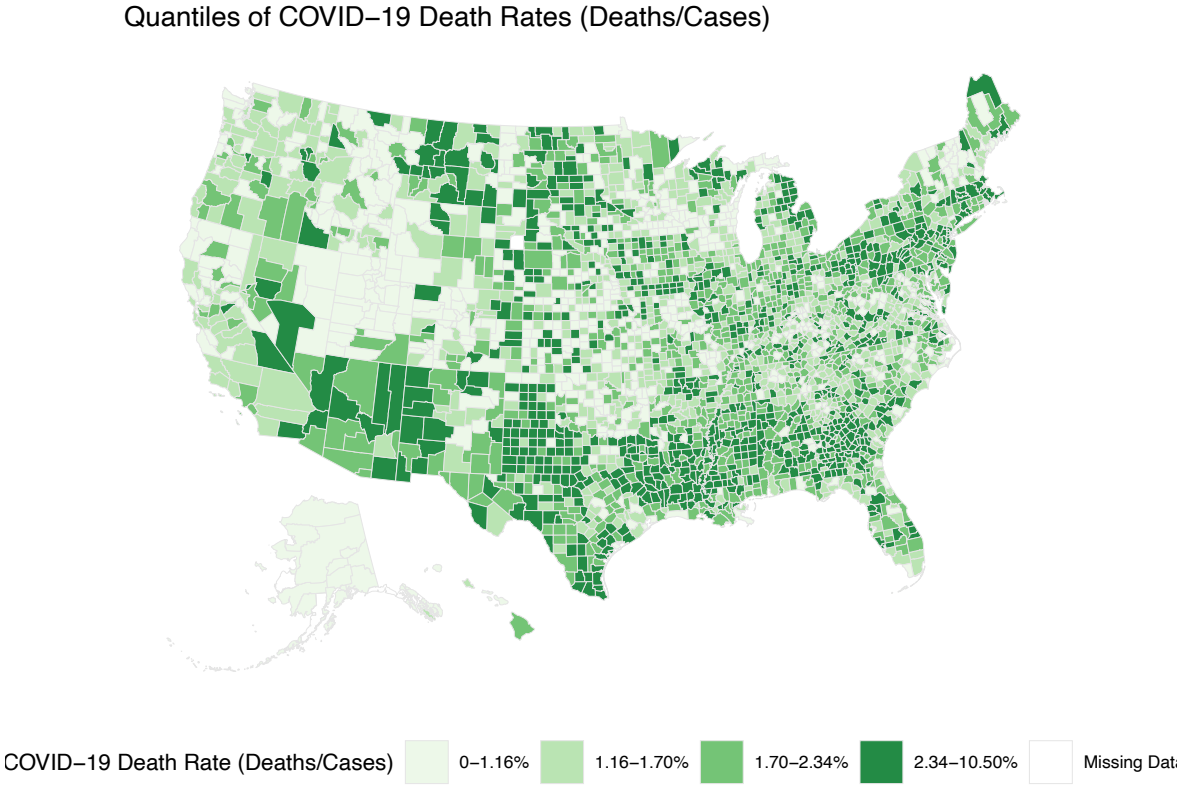
the top 10 counties in terms of death rate have had fewer than 1,000 cases. However, relatively high percentages of their population are in the age range of 74 to 85 and some have higher than average concentrations of Hispanic and/or African Americans.

	name	death_rate	cases	african_am	native_am	hispanic	age74to85
8	Grant County, Nebraska	0.1052632	38	0.8	0.0	1.2	5.8
1	Motley County, Texas	0.0888889	90	1.5	0.0	19.6	9.3
6	Sabine County, Texas	0.0844794	509	5.6	0.7	4.6	9.7
7	Petroleum County, Montana	0.0833333	12	0.0	0.0	0.0	13.2
9	Foard County, Texas	0.0833333	120	2.3	0.5	22.3	11.6
5	Robertson County, Kentucky	0.0696517	201	1.8	0.0	4.3	6.7
4	Candler County, Georgia	0.0685225	934	25.4	0.0	11.8	4.8
10	Throckmorton County, Texas	0.0684932	73	1.0	0.0	13.2	10.3
3	Glascock County, Georgia	0.0681818	264	10.2	0.3	1.2	5.4
2	Hancock County, Georgia	0.0664452	903	73.2	0.0	1.7	6.9

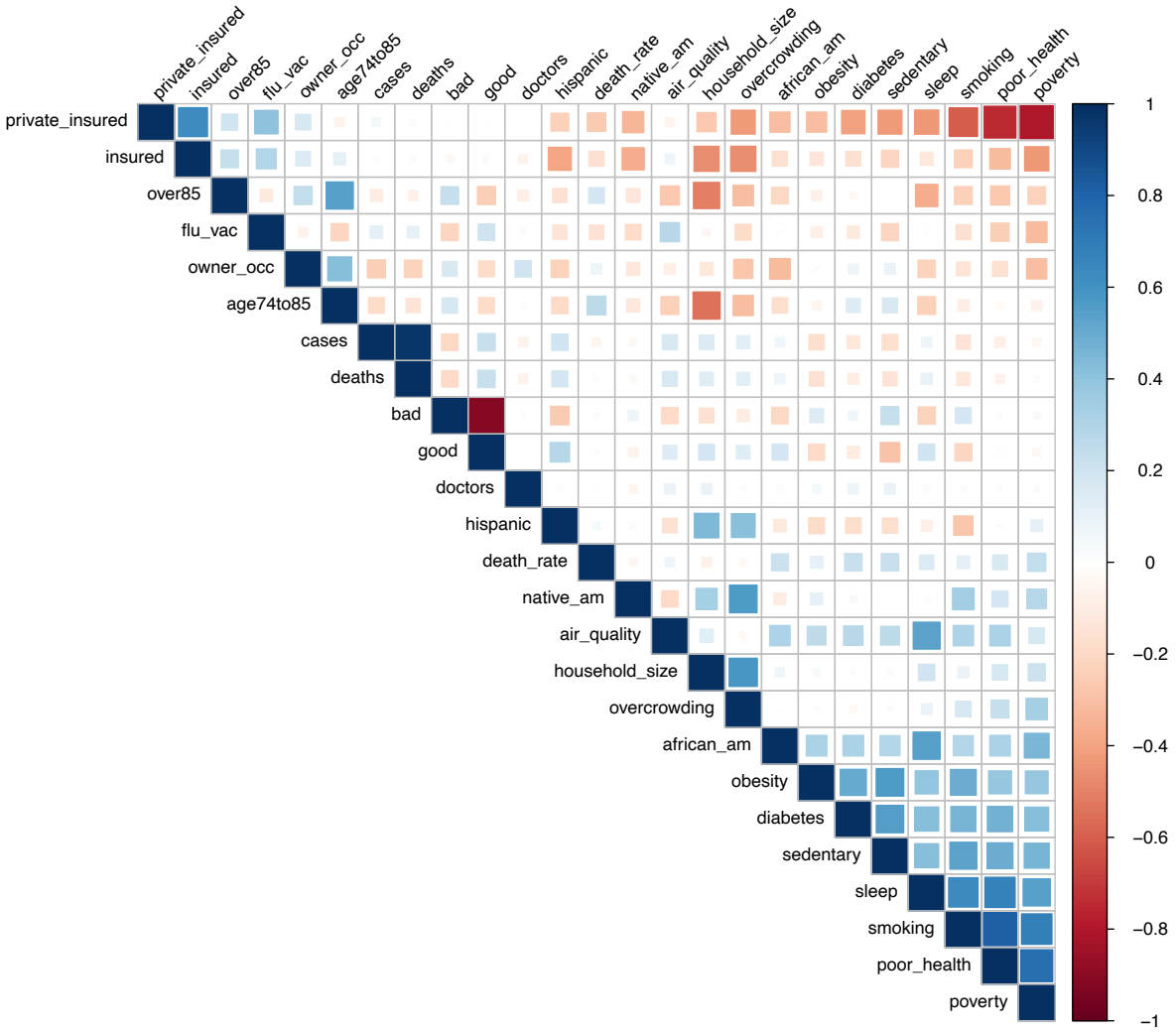
Next, we examine counties with the highest number of cases. These are not surprisingly, counties with large overall populations. The percentage of people in these counties who identify as “Hispanic” or “African American” are notably high.

	name	death_rate	cases	african_am	native_am	hispanic	age74to85
8	Los Angeles County, California	0.0189934	1220900	8.1	0.7	48.5	3.9
7	Maricopa County, Arizona	0.0184483	524547	5.6	2.0	31.0	4.5
5	Cook County, Illinois	0.0204774	499429	23.4	0.3	25.3	4.3
6	Miami-Dade County, Florida	0.0131487	447041	17.4	0.2	68.5	5.2
9	Harris County, Texas	0.0154241	377720	19.0	0.4	42.9	2.8
2	Riverside County, California	0.0147276	294617	6.5	0.8	48.9	4.4
10	Dallas County, Texas	0.0132687	291589	22.6	0.4	40.2	2.9
3	San Bernardino County, California	0.0149679	291022	8.3	0.8	53.3	3.2
1	San Diego County, California	0.0131374	270602	5.0	0.7	33.7	4.0
4	Orange County, California	0.0178522	266353	1.8	0.5	34.1	4.3

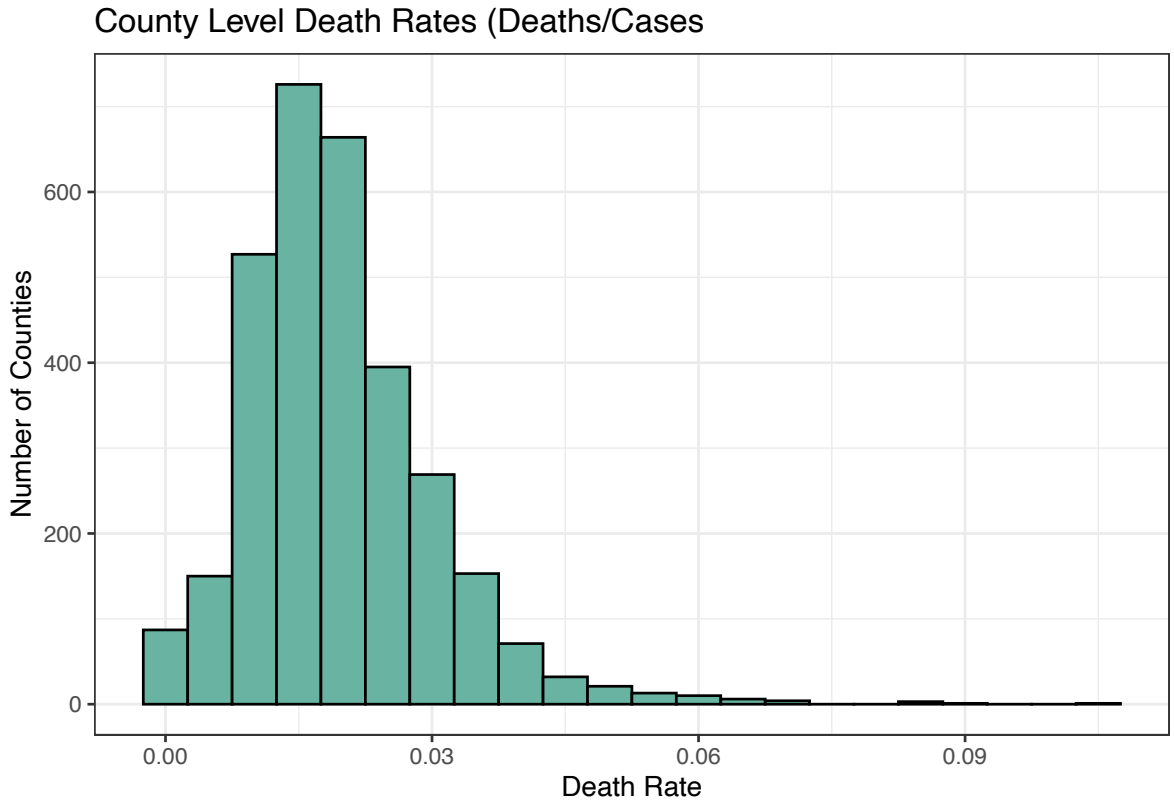
A county level map of the US, shows interesting patterns of high COVID-19 death rates. The map below is color-coded by quartiles of rates of death, the highest rates being the darkest green. It shows significant concentrations in the Southwest, South, and Northeast.



The corrplot function is used to create a table showing the correlation across variables. There are not strong correlations apparent between the death_rate column and other variables, but there are strong correlations across variables such as poverty and poor_health.



The histogram of death rates shows that the data is strongly right skewed.



TODO: Start here!! ## Model Fitting (Methods) To build a predictive model for the percent of people who die of those who have contracted COVID-19 (death_rate), we'll start with a basic model of predicting at the overall average and a linear regression model using obesity, age, air quality, and poverty. Then we'll apply the more advanced approaches of a cross-validated bootstrap aggregated regression tree and random forest. First, however, we'll split the data in to three subsets - 1) train_set (70%) to fit the models, 2) test_set (15%) to test the models, and 3) validation_set (15%) to independently validate the model at the very end.

2.2.1 Split the Data (Step 3 in DS-Capstone-Final.R)

Here we'll start by removing a few columns we don't need for modelling - county name, state number, and county number. We set the seed so the analysis can be repeated and use the function createDataPartition to generate an index of random row numbers to define the train_set and then to split the remaining 30% into the test_set and validation_set.

```
# remove columns from dat not needed for regression
tmp <- dat%>% select(-"name", -"state", -"county")

# set the seed for replicability
set.seed(755)

# create an index to split the data into a train (70%) and tmp (30%)
# then split tmp into test (15%) and validation (15%)
```

```r
# create the first index for the train_set / tmp_set split
test_index <- createDataPartition(y = tmp$fips, times = 1, p = 0.3, list = FALSE)
# split the data into a training set and testing set
train_set <- tmp[-test_index,]
tmp_set <- tmp[test_index,]

# create the second index for the validation_set / test_set split
test_index <- createDataPartition(y = tmp_set$fips, times = 1, p = 0.5, list = FALSE)
# split the data into a training set and testing set
validation_set <- tmp_set[-test_index,]
test_set <- tmp_set[test_index,]
```

2.2.2 Model Fitting (Step 4 in DS-Capstone-Final.R)

To measure the effectiveness of the model, we'll use a loss function that measures the differences between predicted values and actual values in the test_set or validation_set. This measure is called the residual (or root) mean squared error or RMSE. Here, an RMSE of .0100 implies that the predictions are 1.0% points away on average from the actual values. We define y_c as the COVID-19 death rate for a county c. The prediction is $\hat{y}_c$. The RMSE is then defined as shown in the formula below with N being the number of counties:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{c}(\hat{y}_c - y_c)^2}$$

```r
# Step 4a: Define the loss function to measure model effectivenesss
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

We'll start with the simplest of models - predicting all values at the overall train_set average. Then we create a table to store the RMSE results of each model for easy comparison.

```r
# Step 4b: The simplest model
just_the_average <- RMSE(test_set$death_rate, mean(train_set$death_rate))

# Create a table to store the results and add this simple model to it
rmse_results <-
  tibble("Method" = "Just the Average", RMSE = just_the_average)
kbl(rmse_results, booktabs = T) %>% kable_styling(latex_options = "striped")
```

Method	RMSE
Just the Average	0.0110098

The second approach is a linear regression model incorporating the effects of obesity, poverty, age, and air quality. This model is an improvement over the basic model, but we are still off by over 1.1% points and the overall average death rate is just 1.956%.

```r
# Step 4c: A linear regression model
lm_model <- lm(death_rate ~ obesity + poverty + age74to85 + air_quality, data = train_set)
pred <- predict(lm_model, test_set)
rmse_results <- bind_rows(rmse_results,
  tibble("Method" = "Linear Regression - Obesity, Poverty, Age, Air Quality",
         RMSE = RMSE(pred, test_set$death_rate)))
kbl(rmse_results, booktabs = T) %>% kable_styling(latex_options = "striped")
```

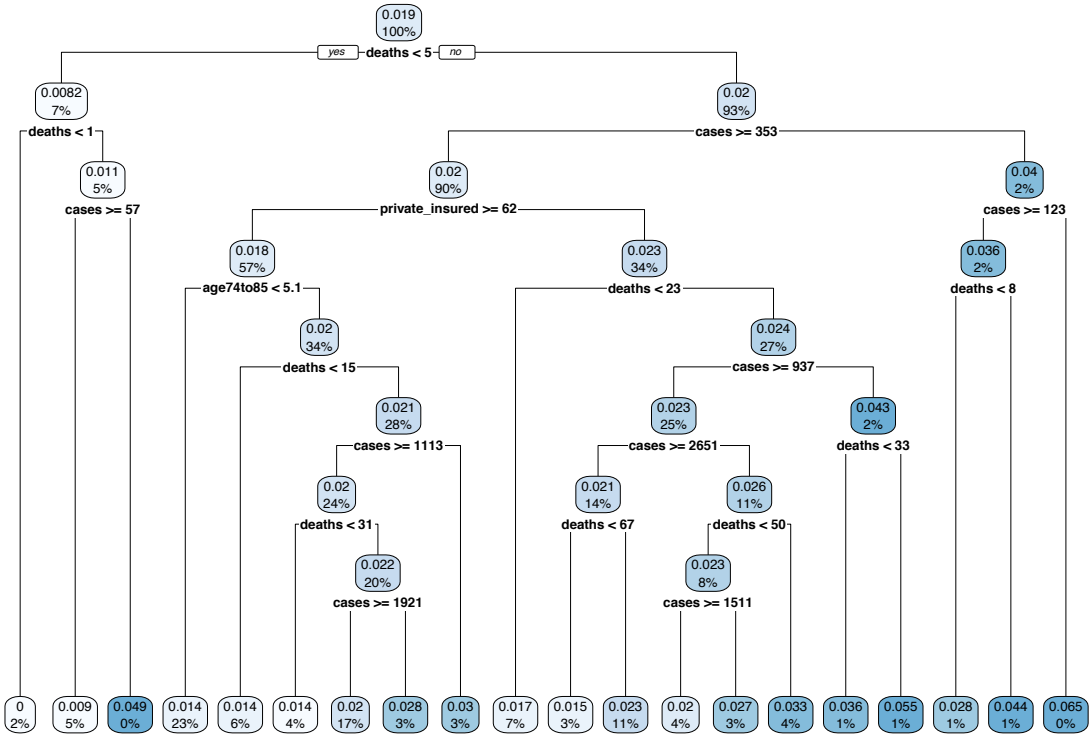
Method	RMSE
Just the Average	0.0110098
Linear Regression - Obesity, Poverty, Age, Air Quality	0.0099805

We turn now to a more advanced approach the Bootstrapped Aggregated or "Bagged" Regression Tree. The basic idea is to "partition the data in to smaller subgroups and then fit a constant for each observation in the subgroup. The partitioning is achieved by successive binary partitions (aka recursive partitioning) based on the different predictors. The constant to predict is based on the average response values for all observations that fall in that subgroup." Source: http://uc-r.github.io/regression_trees First, we create a simple regression tree and then implement cross-validation to improve the performance of our model.

```r
# Step 4d: A regression tree model
# Source: "Regression Trees" tutorial found here: http://uc-r.github.io/regression_trees

# Start with a simple regression tree
tree1 <- rpart(
  formula = death_rate ~ .,
  data    = train_set,
  method  = "anova"
)

# visualize the model
rpart.plot(tree1)
```




```
# Predict the results and add to the table
pred <- predict(tree1, test_set)
rmse_results <- bind_rows(rmse_results,
      tibble("Method" = "Simple Regression Tree", RMSE = RMSE(pred, test_set$death_rate)))
kbl(rmse_results, booktabs = T) %>% kable_styling(latex_options = "striped")
```

Method	RMSE
Just the Average	0.0110098
Linear Regression - Obesity, Poverty, Age, Air Quality	0.0099805
Simple Regression Tree	0.0076355

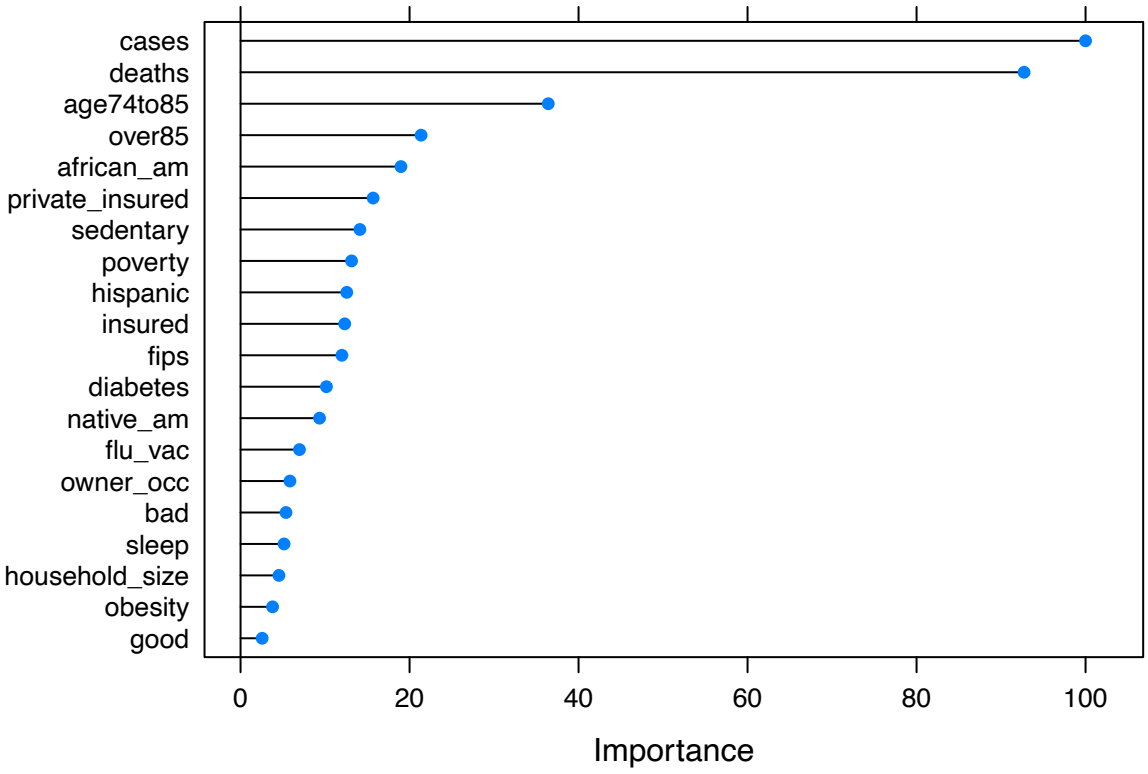
```
# Create a regression tree model with bootstrap aggregating or "bagging" with the Caret package
# Specify 10-fold cross validation
ctrl <- trainControl(method = "cv",  number = 10)

# Cross validation bagged model
bagged_cv <- train(
  death_rate ~ .,
  data = train_set,
  method = "treebag",
  trControl = ctrl,
  importance = TRUE
)

# assess results
```

```
# bagged_cv

# plot most important variables
plot(varImp(bagged_cv), 20)
```




```
# Evaluate on the test_set, add to the results table and print the table
pred <- predict(bagged_cv, test_set)
rmse_results <- bind_rows(rmse_results,
        tibble("Method" = "Cross Validated Bagged Tree", RMSE = RMSE(pred, test_set$death_rate)))
kbl(rmse_results, booktabs = T) %>% kable_styling(latex_options = "striped")
```

Method	RMSE
Just the Average	0.0110098
Linear Regression - Obesity, Poverty, Age, Air Quality	0.0099805
Simple Regression Tree	0.0076355
Cross Validated Bagged Tree	0.0058532

Random Forest is the last approach applied and offers a significant improvement over the earlier models. "Bagging (bootstrap aggregating) regression trees is a technique that can turn a single tree model with high variance and poor predictive power into a fairly accurate prediction function. Unfortunately, bagging regression trees typically suffers from tree correlation, which reduces the overall performance of the model. Random forests are a modification of bagging that builds a large collection of de-correlated trees and have become a very popular "out-of-the-box" learning algorithm that enjoys

good predictive performance. This tutorial will cover the fundamentals of random forests." Source: http://uc-r.github.io/random_forests

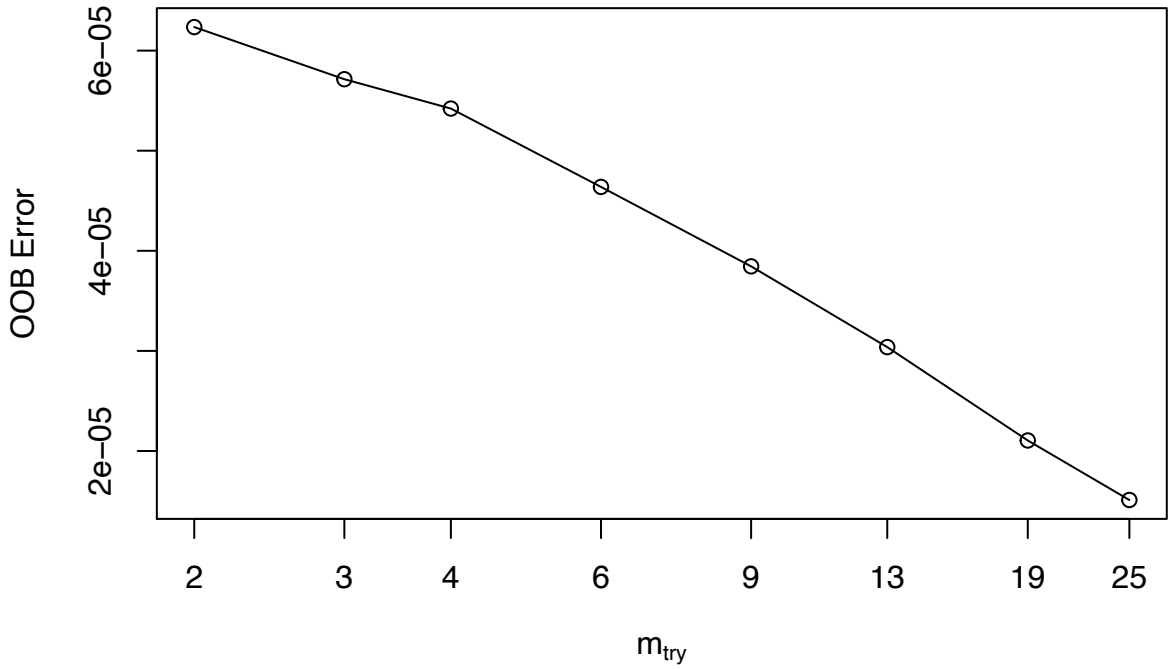
The Random Forest model here outperforms the other approaches significantly with an RMSE below .005. Looking at the variable importance implies that the percent of a county's population that is age 74 to 85 and those with private insurance are drivers of predicting death rates for COVID-19 patients.

```
# Step 4e: A Random Forest Model
# Source: "Random Forests" tutorial found here: https://uc-r.github.io/random_forests
# Create a Random Forest model using the tuneRF from the randomForest package

# Create a list of features which is the column names without "death_rate"
features <- setdiff(names(train_set), "death_rate")

# Run the tuning process, this takes a few seconds
set.seed(123)
rf_tune <- tuneRF(
  x          = train_set[features],
  y          = train_set$death_rate,
  ntreeTry   = 500,
  mtryStart  = 2,
  stepFactor = 1.5,
  improve    = 0.001,
  trace      = FALSE      # to not show real-time progress
)
```

```
## 0.08350873 0.001
## 0.05132146 0.001
## 0.1443847 0.001
## 0.1711439 0.001
## 0.2098019 0.001
## 0.3069633 0.001
## 0.2824293 0.001
```

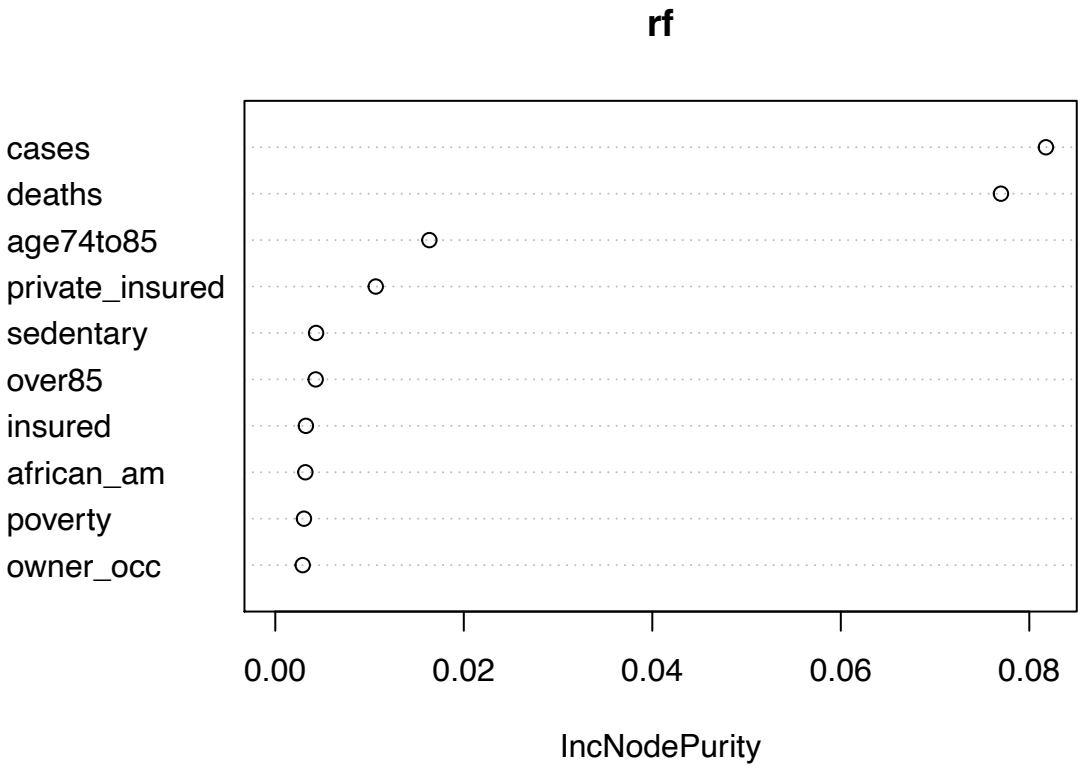



```r
# Find the best mtry for our model
rf_tune <- as.data.frame(rf_tune)
best_mtry <- rf_tune$mtry[which(rf_tune$OOBError == min(rf_tune$OOBError))]

# Update our model with the results of tuning
rf <- randomForest::randomForest(
  formula = death_rate ~ .,
  data    = train_set,
  mtry = best_mtry
)

# Look at the model and plot the errors associated with the different number of trees
# rf
# plot(rf)

# Show the most important variables
varImpPlot(rf, n.var = 10)
```




```
# Find the number of trees with lowest MSE
# which.min(rf$mse)

# RMSE of the best random forest
# sqrt(rf$mse[which.min(rf$mse)])

# Predict the results and add to our table
pred <- predict(rf, test_set)
rmse_results <- bind_rows(rmse_results,
  tibble("Method" = "Random Forest with Tuning", RMSE = RMSE(pred, test_set$death_rate)))
kbl(rmse_results, booktabs = T) %>% kable_styling(latex_options = "striped")
```

Method	RMSE
Just the Average	0.0110098
Linear Regression - Obesity, Poverty, Age, Air Quality	0.0099805
Simple Regression Tree	0.0076355
Cross Validated Bagged Tree	0.0058532
Random Forest with Tuning	0.0042580

3 Results (Step 5 in DS-Capstone-Final.R)

We use the validation set which is 15% of the total data to evaluate the performance of the models. This shows results consistent with those generated using the train_set and test_set.

```r
# Predict the results for the simple average and add to our table
just_the_average <- RMSE(validation_set$death_rate, mean(validation_set$death_rate))
rmse_results <- bind_rows(rmse_results,
  tibble("Method" = "Validation of Just the Average", RMSE = just_the_average))

# Predict the results for the simple average and add to our table
pred <- predict(lm_model, validation_set)
rmse_results <- bind_rows(rmse_results,
  tibble("Method" = "Validation of Linear Regression", RMSE = just_the_average))

# Predict the results for the cross validation bagged tree on the validation set and add to our table
pred <- predict(bagged_cv, validation_set)
rmse_results <- bind_rows(rmse_results,
  tibble("Method" = "Validation of Cross Validated Bagged Tree", RMSE = RMSE(pred, validation_set$dea

# Predict the results for the random forest model on the validation set and add to our table
pred <- predict(rf, validation_set)
rmse_results <- bind_rows(rmse_results,
  tibble("Method" = "Validation of Random Forest with Tuning", RMSE = RMSE(pred, validation_set$death

# Show the table of all results including validation
kbl(rmse_results, booktabs = T) %>% kable_styling(latex_options = "striped")
```

Method	RMSE
Just the Average	0.0110098
Linear Regression - Obesity, Poverty, Age, Air Quality	0.0099805
Simple Regression Tree	0.0076355
Cross Validated Bagged Tree	0.0058532
Random Forest with Tuning	0.0042580
Validation of Just the Average	0.0109505
Validation of Linear Regression	0.0109505
Validation of Cross Validated Bagged Tree	0.0067254
Validation of Random Forest with Tuning	0.0047573

4 Conclusion

This study set out to create a county level data set of demographic, health, and COVID-19 observations to then build a predictive model to guess the rate of death from COVID-19 for those who've contracted the disease (deaths/cases). The modelling approaches included a simple "Just the Average" approach, a linear regression model, a regression tree model with bootstrap aggregating, and finally a random forest approach. The best performing model was the random forest which had an RMSE of less than .005%.

While the random forest model preforms relatively well, there is clearly room for improvement. The model results are also a bit of a "black box" and would be difficult to translate into policy. That could be addressed by pairing the model results with descriptive analytics. Re-running the analysis with

more granular data such as Census tract level information if available could build a model that was both more predictive and provide analytics that were more specific to needs of different geographies.

Finally, The New York City data, ideally, would be included in the analysis. To do so, we'd define a way to aggregate the five counties into the comparable New York City data available at the New York Times. This may or may not improve the predictability of the overall model, but would be important for any demographic analysis.

Resources

During the process of putting together this report and building the underlying R script I consulted many online resources. The most important are listed here.

1. Irizarry, Rafael, A., "Introduction to Data Science", Published: October 24, 2019, Accessed: February 2021, https://rafalab.github.io/dsbook/ (A comprehensive guide to Data Science and R)

2. US Census Bureau, "American Community Survey 5-Year Data (2009-2019): Data Profile", Accessed: April 9, 2021, https://www.census.gov/data/developers/data-sets/acs-5year.html (API portal for 5-year ACS data)

3. US Census Bureau, "Using the Census Data API With the American Community Survey, What Data Users Need to Know", Published: February 2020, Accessed March-April 2021, https://www.census.gov/content/dam/Census/library/publications/2020/acs/acs_api_handbook_2020.pdf (A guide to pulling data using the public API)

4. US Census Bureau, "Demystifying the Census API", Published: July 22, 2020, Accessed: March 2021, https://www.youtube.com/watch?v=dlOUZcrqiUA (Step-by-step instructions on how to pull data using the public API)

5. US Census Bureau, "UNDERSTANDING AND USING ACS SINGLE- YEAR AND MULTI-YEAR ESTIMATES", Accessed: March 2021, https://www.census.gov/content/dam/Census/library/publications/2018/acs/acs_general_handbook_2018_ch03.pdf (A description of the 1-year, 1-year supplemental 3-year, and 5-year ACS tables)

6. US Census Bureau, "Table IDs Explained", Accessed: March 2021, https://www.census.gov/programs-surveys/acs/guidance/which-data-tool/table-ids-explained.html (An explanation of American Community Survey variable names)

7. US Census Bureau, "American Community Survey Information Guide", Accessed: March 2021, https://www.census.gov/content/dam/Census/programs-surveys/acs/about/ACS_Information_Guide.pdf (General overview of the ACS)

8. US Census Bureau, "Census Data API: Variables in /data/2019/acs/acs5/profile/variables", Accessed: March 2021, https://api.census.gov/data/2019/acs/acs5/profile/variables.html (A comprehensive list of variables available for the ACS 5-year table)

9. The New York Times. (2021). Coronavirus (COVID-19) Data in the United States. Retrieved April 9, 2021, from https://github.com/nytimes/covid-19-data (Data from The New York Times, based on reports from state and local health agencies)

10. Mask Usage Estimates from The New York Times, based on roughly 250,000 interviews conducted by Dynata from July 2 to July 14, Published: July 2021, Accessed: April 2, 2021, https://github.com/nytimes/covid-19-data/tree/master/mask-use (Data showing level of consistent mask use by county)

11. County Health Rankings & Roadmaps, University of Wisconsin Population Health Institute, 2020, Accessed: March 28, 2021, https://www.countyhealthrankings.org/explore-health-rankings/rankings-data-documentation (A comprehensive data set of county level population health metrics. File layout, data explanations and sources are available in the documentation found here - https://www.countyhealthrankings.org/sites/default/files/media/document/2020%20Analytic%20Documentation_0.pdf)

12. University of Cincinnati, "Regression Trees", Published: April 28, 2018, Accessed: March: 2021, http://uc-r.github.io/regression_trees (A helpful tutorial on building a Regression Tree model in R)

13. University of Cincinnati, "Random Forests", Published: May 9, 2018, Accessed: March: 2021, http://uc-r.github.io/random_forests (A helpful tutorial on building a Random Forest model in R)

14. Kieran Healy, "Data Visualization - A practical introduction, Chapter 7: Draw Maps", Published: 2018-04-25, Accessed: April 2021. https://socviz.co/maps.html (An easy to follow approach to creating a map of county level data)

15. Nevo Itzhak, Tomer Shahar, Robert Moskovich, "The Impact of U.S. County-Level Factors on COVID-19 Morbidity and Mortality", Accessed: March 2021, https://doi.org/10.1101/2021.01.19.21250092, (A research paper describing a machine learning model to predict Morbidity and Mortality)

16. "Federal Information Processing System (FIPS) Codes for States and Counties", Accessed: March 2021, Source: https://transition.fcc.gov/oet/info/maps/census/fips/fips.txt (An explanation of FIPS codes)