GoDaddy - Microbusiness Density Forecasting

Mohammad Solki

2023-02-28

# Introduction

## Goal of the Competition

The challenge in this competition is to forecast microbusiness activity across the United States, as measured by the density of microbusinesses in US counties. Microbusinesses are often too small or too new to show up in traditional economic data sources, but microbusiness activity may be correlated with other economic indicators of general interest.

This work will help policymakers gain visibility into microbusinesses, a growing trend of very small entities. Additional information will enable new policies and programs to improve the success and impact of these smallest businesses.

GoDaddy’s Venture Forward team has gathered data on over 20 million microbusinesses in the United States, defined as businesses with an online presence and ten or fewer employees, to help policymakers understand the factors associated with these small businesses. While traditional economic data sources often miss these businesses, GoDaddy’s survey data can provide insights into this sector of the economy. The data can be used to improve predictions and inform decision-making to create more inclusive and resilient economies. The competition hosted by GoDaddy aims to empower entrepreneurs by giving them the tools they need to grow online and make a substantial impact on communities across the country.

Model accuracy will be evaluated on SMAPE (Symmetric mean absolute percentage error) between forecasts and actual values. We define SMAPE = 0 when the actual and predicted values are both 0.

SMAPE formula is usually defined as follows:

where:

* is the number of observations in the time series
* is the forecasted value at time
* is the actual value at time
* denotes the absolute value of .

## Datasets

A great deal of data is publicly available about counties and we have not attempted to gather it all here. You are strongly encouraged to use external data sources for features.

**train.csv**

* row\_id An ID code for the row.
* cfips A unique identifier for each county using the Federal Information Processing System. The first two digits correspond to the state FIPS code, while the following 3 represent the county.
* county\_name The written name of the county.
* state\_name The name of the state.
* first\_day\_of\_month The date of the first day of the month.
* microbusiness\_density Microbusinesses per 100 people over the age of 18 in the given county. This is the target variable. The population figures used to calculate the density are on a two-year lag due to the pace of updates provided by the U.S. Census Bureau, which provides the underlying population data annually. 2021 density figures are calculated using 2019 population figures, etc.
* active The raw count of microbusinesses in the county. Not provided for the test set.

**test.csv** Metadata for the submission rows. This file will remain unchanged throughout the competition.

* row\_id An ID code for the row.
* cfips A unique identifier for each county using the Federal Information Processing System. The first two digits correspond to the state FIPS code, while the following 3 represent the county.
* first\_day\_of\_month The date of the first day of the month.

**census\_starter.csv** Examples of useful columns from the Census Bureau’s American Community Survey (ACS) at [data.census.gov](https://data.census.gov/). The percentage fields were derived from the raw counts provided by the ACS. All fields have a two-year lag to match what information was available at the time a given microbusiness data update was published.

* pct\_bb\_[year] The percentage of households in the county with access to broadband of any type. Derived from ACS table B28002: PRESENCE AND TYPES OF INTERNET SUBSCRIPTIONS IN HOUSEHOLD.
* cfips The CFIPS code.
* pct\_college\_[year] The percent of the population in the county over age 25 with a 4-year college degree. Derived from ACS table S1501: EDUCATIONAL ATTAINMENT.
* pct\_foreign\_born\_[year] The percent of the population in the county born outside of the United States. Derived from ACS table DP02: SELECTED SOCIAL CHARACTERISTICS IN THE UNITED STATES.
* pct\_it\_workers\_[year] The percent of the workforce in the county employed in information-related industries. Derived from ACS table S2405: INDUSTRY BY OCCUPATION FOR THE CIVILIAN EMPLOYED POPULATION 16 YEARS AND OVER.
* median\_hh\_inc\_[year] The median household income in the county. Derived from ACS table S1901: INCOME IN THE PAST 12 MONTHS (IN 2021 INFLATION-ADJUSTED DOLLARS).

## Environment Setup

First, we’ll set the working directory using **setwd()**, and then import the required libraries. As we proceed through the report the list of libraries might change.

# Set the working directory  
setwd("/Users/dreamer/Downloads/Godaddy/godaddy\_microbusiness\_forecasting")

## Importing the libraries  
# Recognize package conflicts  
library(conflicted)  
  
# Multi-purpose package for data import, tidying, manipulation, visualization, and programming  
library(tidyverse)  
  
# Deal with missing data  
library(mice)  
  
# Related to plots  
library(maps)  
library(gridExtra)  
library(mapdata)  
library(ggcorrplot)  
library(corrplot)  
  
# Training  
library(forecast)  
library(Metrics)  
library(caret)  
library(gbm)  
  
# Color palette  
library(viridis)  
  
# Future Selection   
library(glmnet)  
library(randomForest)  
library(rpart)

# Exploratory Data Analysis (EDA)

## Data Preprocessing & Cleaning

### Exploring the datasets

To explore the datasets, we load the **train**, **test**, and **census\_starter** datasets into R dataframes to get a better understanding of the data.

train\_df <- read.csv("./datasets/train.csv")  
  
test\_df <- read.csv("./datasets/test.csv")  
  
census\_df <- read.csv("./datasets/census\_starter.csv")

After reading the CSV files into dataframes, we should check whether the data is loaded correctly or not. We can use the head() function of R to display the first few rows of the dataframes and the tail() function to display the last rows. This will display the first and last six rows of the **train**, **test,** and **census** dataframes. We can also use other R functions such as str() and summary() to get more information about the dataframes, such as column names, data types, and summary statistics.

# Display the first 3 rows of the dataframes  
head(train\_df, n = 3)

## row\_id cfips county state first\_day\_of\_month  
## 1 1001\_2019-08-01 1001 Autauga County Alabama 2019-08-01  
## 2 1001\_2019-09-01 1001 Autauga County Alabama 2019-09-01  
## 3 1001\_2019-10-01 1001 Autauga County Alabama 2019-10-01  
## microbusiness\_density active  
## 1 3.007682 1249  
## 2 2.884870 1198  
## 3 3.055843 1269

head(test\_df, n = 3)

## row\_id cfips first\_day\_of\_month  
## 1 1001\_2022-11-01 1001 2022-11-01  
## 2 1003\_2022-11-01 1003 2022-11-01  
## 3 1005\_2022-11-01 1005 2022-11-01

head(census\_df, n = 3)

## pct\_bb\_2017 pct\_bb\_2018 pct\_bb\_2019 pct\_bb\_2020 pct\_bb\_2021 cfips  
## 1 76.6 78.9 80.6 82.7 85.5 1001  
## 2 74.5 78.1 81.8 85.1 87.9 1003  
## 3 57.2 60.4 60.5 64.6 64.6 1005  
## pct\_college\_2017 pct\_college\_2018 pct\_college\_2019 pct\_college\_2020  
## 1 14.5 15.9 16.1 16.7  
## 2 20.4 20.7 21.0 20.2  
## 3 7.6 7.8 7.6 7.3  
## pct\_college\_2021 pct\_foreign\_born\_2017 pct\_foreign\_born\_2018  
## 1 16.4 2.1 2.0  
## 2 20.6 3.2 3.4  
## 3 6.7 2.7 2.5  
## pct\_foreign\_born\_2019 pct\_foreign\_born\_2020 pct\_foreign\_born\_2021  
## 1 2.3 2.3 2.1  
## 2 3.7 3.4 3.5  
## 3 2.7 2.6 2.6  
## pct\_it\_workers\_2017 pct\_it\_workers\_2018 pct\_it\_workers\_2019  
## 1 1.3 1.1 0.7  
## 2 1.4 1.3 1.4  
## 3 0.5 0.3 0.8  
## pct\_it\_workers\_2020 pct\_it\_workers\_2021 median\_hh\_inc\_2017 median\_hh\_inc\_2018  
## 1 0.6 1.1 55317 58786  
## 2 1.0 1.3 52562 55962  
## 3 1.1 0.8 33368 34186  
## median\_hh\_inc\_2019 median\_hh\_inc\_2020 median\_hh\_inc\_2021  
## 1 58731 57982 62660  
## 2 58320 61756 64346  
## 3 32525 34990 36422

# Display the last 3 rows of the dataframes  
tail(train\_df, n = 3)

## row\_id cfips county state first\_day\_of\_month  
## 122263 56045\_2022-08-01 56045 Weston County Wyoming 2022-08-01  
## 122264 56045\_2022-09-01 56045 Weston County Wyoming 2022-09-01  
## 122265 56045\_2022-10-01 56045 Weston County Wyoming 2022-10-01  
## microbusiness\_density active  
## 122263 1.785395 100  
## 122264 1.785395 100  
## 122265 1.785395 100

tail(test\_df, n = 3)

## row\_id cfips first\_day\_of\_month  
## 25078 56041\_2023-06-01 56041 2023-06-01  
## 25079 56043\_2023-06-01 56043 2023-06-01  
## 25080 56045\_2023-06-01 56045 2023-06-01

tail(census\_df, n = 3)

## pct\_bb\_2017 pct\_bb\_2018 pct\_bb\_2019 pct\_bb\_2020 pct\_bb\_2021 cfips  
## 3140 83.8 88.2 89.5 91.4 90.6 56041  
## 3141 76.4 78.3 78.2 82.8 85.4 56043  
## 3142 71.1 73.3 76.8 79.7 81.3 56045  
## pct\_college\_2017 pct\_college\_2018 pct\_college\_2019 pct\_college\_2020  
## 3140 11.9 10.5 11.1 12.6  
## 3141 15.4 15.0 15.4 15.0  
## 3142 14.1 13.5 13.4 12.7  
## pct\_college\_2021 pct\_foreign\_born\_2017 pct\_foreign\_born\_2018  
## 3140 12.3 2.9 3.1  
## 3141 17.2 2.3 1.4  
## 3142 13.9 3.8 4.1  
## pct\_foreign\_born\_2019 pct\_foreign\_born\_2020 pct\_foreign\_born\_2021  
## 3140 2.9 2.9 2.9  
## 3141 1.6 2.2 1.0  
## 3142 1.7 2.3 1.6  
## pct\_it\_workers\_2017 pct\_it\_workers\_2018 pct\_it\_workers\_2019  
## 3140 1.2 1.2 1.4  
## 3141 1.3 1.0 0.9  
## 3142 0.6 0.6 0.0  
## pct\_it\_workers\_2020 pct\_it\_workers\_2021 median\_hh\_inc\_2017  
## 3140 1.7 0.9 54672  
## 3141 0.9 1.1 51362  
## 3142 0.0 0.0 59605  
## median\_hh\_inc\_2018 median\_hh\_inc\_2019 median\_hh\_inc\_2020  
## 3140 58235 63403 72458  
## 3141 53426 54158 57306  
## 3142 52867 57031 53333  
## median\_hh\_inc\_2021  
## 3140 75106  
## 3141 62271  
## 3142 65566

# Display information about the dataframes  
summary(train\_df)

## row\_id cfips county state   
## Length:122265 Min. : 1001 Length:122265 Length:122265   
## Class :character 1st Qu.:18177 Class :character Class :character   
## Mode :character Median :29173 Mode :character Mode :character   
## Mean :30376   
## 3rd Qu.:45077   
## Max. :56045   
## first\_day\_of\_month microbusiness\_density active   
## Length:122265 Min. : 0.000 Min. : 0   
## Class :character 1st Qu.: 1.639 1st Qu.: 145   
## Mode :character Median : 2.587 Median : 488   
## Mean : 3.818 Mean : 6443   
## 3rd Qu.: 4.519 3rd Qu.: 2124   
## Max. :284.340 Max. :1167744

summary(test\_df)

## row\_id cfips first\_day\_of\_month  
## Length:25080 Min. : 1001 Length:25080   
## Class :character 1st Qu.:18177 Class :character   
## Mode :character Median :29173 Mode :character   
## Mean :30376   
## 3rd Qu.:45077   
## Max. :56045

summary(census\_df)

## pct\_bb\_2017 pct\_bb\_2018 pct\_bb\_2019 pct\_bb\_2020   
## Min. :24.50 Min. :25.70 Min. :34.80 Min. :33.30   
## 1st Qu.:64.20 1st Qu.:67.42 1st Qu.:70.50 1st Qu.:74.10   
## Median :70.70 Median :73.60 Median :76.45 Median :79.60   
## Mean :69.92 Mean :72.69 Mean :75.40 Mean :78.54   
## 3rd Qu.:76.40 3rd Qu.:78.80 3rd Qu.:81.40 3rd Qu.:84.10   
## Max. :94.60 Max. :95.50 Max. :96.00 Max. :97.10   
## NA's :1   
## pct\_bb\_2021 cfips pct\_college\_2017 pct\_college\_2018  
## Min. :37.00 Min. : 1001 Min. : 2.40 Min. : 0.00   
## 1st Qu.:76.40 1st Qu.:18178 1st Qu.: 9.70 1st Qu.: 9.90   
## Median :81.70 Median :29176 Median :12.80 Median :13.00   
## Mean :80.54 Mean :30384 Mean :13.81 Mean :14.01   
## 3rd Qu.:85.90 3rd Qu.:45080 3rd Qu.:16.80 3rd Qu.:17.10   
## Max. :97.60 Max. :56045 Max. :43.70 Max. :48.00   
## NA's :1   
## pct\_college\_2019 pct\_college\_2020 pct\_college\_2021 pct\_foreign\_born\_2017  
## Min. : 0.00 Min. : 0.00 Min. : 0.00 Min. : 0.000   
## 1st Qu.:10.10 1st Qu.:10.50 1st Qu.:10.60 1st Qu.: 1.400   
## Median :13.25 Median :13.60 Median :13.80 Median : 2.700   
## Mean :14.24 Mean :14.63 Mean :14.85 Mean : 4.702   
## 3rd Qu.:17.30 3rd Qu.:17.90 3rd Qu.:18.00 3rd Qu.: 5.700   
## Max. :45.40 Max. :43.00 Max. :43.70 Max. :52.900   
## NA's :1 NA's :1   
## pct\_foreign\_born\_2018 pct\_foreign\_born\_2019 pct\_foreign\_born\_2020  
## Min. : 0.000 Min. : 0.000 Min. : 0.000   
## 1st Qu.: 1.400 1st Qu.: 1.400 1st Qu.: 1.400   
## Median : 2.700 Median : 2.700 Median : 2.800   
## Mean : 4.725 Mean : 4.769 Mean : 4.749   
## 3rd Qu.: 5.700 3rd Qu.: 5.700 3rd Qu.: 5.700   
## Max. :53.300 Max. :53.700 Max. :54.000   
## NA's :1   
## pct\_foreign\_born\_2021 pct\_it\_workers\_2017 pct\_it\_workers\_2018  
## Min. : 0.000 Min. : 0.000 Min. : 0.000   
## 1st Qu.: 1.400 1st Qu.: 0.800 1st Qu.: 0.800   
## Median : 2.700 Median : 1.300 Median : 1.300   
## Mean : 4.744 Mean : 1.427 Mean : 1.382   
## 3rd Qu.: 5.700 3rd Qu.: 1.900 3rd Qu.: 1.800   
## Max. :54.000 Max. :17.400 Max. :11.700   
## NA's :1 NA's :1   
## pct\_it\_workers\_2019 pct\_it\_workers\_2020 pct\_it\_workers\_2021 median\_hh\_inc\_2017  
## Min. : 0.000 Min. : 0.000 Min. : 0.000 Min. : 19264   
## 1st Qu.: 0.700 1st Qu.: 0.700 1st Qu.: 0.600 1st Qu.: 41123   
## Median : 1.200 Median : 1.200 Median : 1.100 Median : 48066   
## Mean : 1.339 Mean : 1.309 Mean : 1.273 Mean : 49754   
## 3rd Qu.: 1.800 3rd Qu.: 1.800 3rd Qu.: 1.700 3rd Qu.: 55764   
## Max. :10.500 Max. :15.200 Max. :15.200 Max. :129588   
## NA's :1 NA's :1   
## median\_hh\_inc\_2018 median\_hh\_inc\_2019 median\_hh\_inc\_2020 median\_hh\_inc\_2021  
## Min. : 20188 Min. : 21504 Min. : 22292 Min. : 17109   
## 1st Qu.: 42480 1st Qu.: 44155 1st Qu.: 45653 1st Qu.: 48180   
## Median : 49888 Median : 51758 Median : 52842 Median : 55907   
## Mean : 51583 Mean : 53476 Mean : 55012 Mean : 58223   
## 3rd Qu.: 57611 3rd Qu.: 59867 3rd Qu.: 61501 3rd Qu.: 64930   
## Max. :136268 Max. :142299 Max. :147111 Max. :156821   
## NA's :1 NA's :2 NA's :2

The data type of first\_day\_of\_month column in **train\_df** and **test\_df** is *character*. We will use the **as.Date()** function to convert the character to *Date* format.

str(train\_df$first\_day\_of\_month)

## Date[1:122265], format: "2019-08-01" "2019-09-01" "2019-10-01" "2019-11-01" "2019-12-01" ...

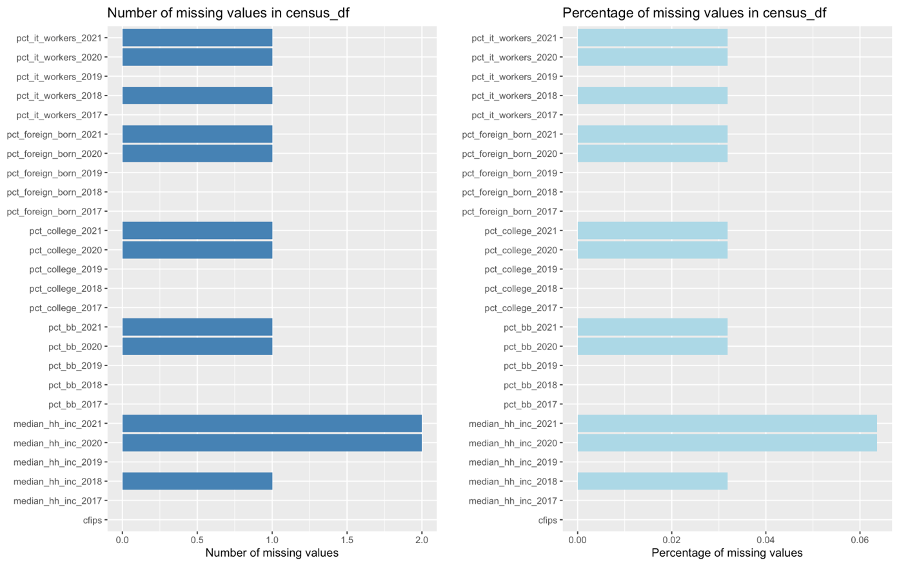
### Missing Values Identification

The **is.na()** function is used to create a logical matrix where *TRUE* represents a missing value and *FALSE* represents a non-missing value. The **colSums()** function is then used to count the number of missing values in each column of the data frame. If the sum of a column is greater than 0, it means that there is at least one missing value in that column.

## row\_id cfips county   
## 0 0 0   
## state first\_day\_of\_month microbusiness\_density   
## 0 0 0   
## active   
## 0

## row\_id cfips first\_day\_of\_month   
## 0 0 0

## pct\_bb\_2017 pct\_bb\_2018 pct\_bb\_2019   
## 0 0 0   
## pct\_bb\_2020 pct\_bb\_2021 cfips   
## 1 1 0   
## pct\_college\_2017 pct\_college\_2018 pct\_college\_2019   
## 0 0 0   
## pct\_college\_2020 pct\_college\_2021 pct\_foreign\_born\_2017   
## 1 1 0   
## pct\_foreign\_born\_2018 pct\_foreign\_born\_2019 pct\_foreign\_born\_2020   
## 0 0 1   
## pct\_foreign\_born\_2021 pct\_it\_workers\_2017 pct\_it\_workers\_2018   
## 1 0 1   
## pct\_it\_workers\_2019 pct\_it\_workers\_2020 pct\_it\_workers\_2021   
## 0 1 1   
## median\_hh\_inc\_2017 median\_hh\_inc\_2018 median\_hh\_inc\_2019   
## 0 1 0   
## median\_hh\_inc\_2020 median\_hh\_inc\_2021   
## 2 2



There are no missing values in **train\_df** and **test\_df** dataframes but, there are missing values in **census\_df**. We use the **complete.cases()** function to determine which rows have complete data and which rows have missing values. This function returns a logical vector indicating which rows have no missing values. Therefore, to identify the rows with missing values, we use the **!** operator to negate the logical vector returned by **complete.cases()**. Then, we use the **is.na()** function to identify which columns have missing values for each missing row:

## Row 93 has missing values in columns: pct\_bb\_2020, pct\_bb\_2021, pct\_college\_2020, pct\_college\_2021, pct\_foreign\_born\_2020, pct\_foreign\_born\_2021, pct\_it\_workers\_2020, pct\_it\_workers\_2021, median\_hh\_inc\_2020, median\_hh\_inc\_2021   
## Row 1817 has missing values in columns: pct\_it\_workers\_2018, median\_hh\_inc\_2018   
## Row 2645 has missing values in columns: median\_hh\_inc\_2020   
## Row 2674 has missing values in columns: median\_hh\_inc\_2021

### Missing Values Imputation

The **mice** package implements a method to deal with missing data. The package creates multiple imputations (replacement values) for multivariate missing data. (“mice function - RDocumentation”)

We’ll use the **mice** package to impute missing values in the **census\_df** dataframe with below arguments:

* *m*: The number of imputations to generate was set to 5, because, generally, *m* should be set to at least 5 for good imputation performance. Creating too many datasets will increase the computational load and may not necessarily lead to better results.
* *maxit*: The *maxit* value was set to 50 to allow for a larger number of iterations to ensure that the imputation algorithm converges and fills in missing values as accurately as possible.
* *method*: In this case, we are using *“pmm”* which stands for *Predictive Mean Matching*, because it is a flexible and widely used imputation method that works well with continuous variables. The method estimates the missing values by drawing from a set of observed values that have similar characteristics to the missing values.
* *print*: The print value is set to *FALSE* because this function prints a huge log output to console.

# Check the filled missing values   
print(imputed\_data[missing\_rows,])

## pct\_bb\_2017 pct\_bb\_2018 pct\_bb\_2019 pct\_bb\_2020 pct\_bb\_2021 cfips  
## 93 80.5 79.1 80.4 84.5 85.9 2261  
## 1817 49.1 52.1 57.6 60.7 63.5 35039  
## 2645 66.3 66.6 61.2 63.2 70.1 48243  
## 2674 64.5 72.7 73.3 96.8 97.0 48301  
## pct\_college\_2017 pct\_college\_2018 pct\_college\_2019 pct\_college\_2020  
## 93 23.1 19.0 16.5 17.4  
## 1817 12.0 12.5 12.6 10.6  
## 2645 18.4 16.0 10.8 14.3  
## 2674 4.7 0.0 0.0 0.0  
## pct\_college\_2021 pct\_foreign\_born\_2017 pct\_foreign\_born\_2018  
## 93 16.3 4.9 6.3  
## 1817 10.1 4.5 3.7  
## 2645 10.9 22.4 14.9  
## 2674 0.0 10.8 15.7  
## pct\_foreign\_born\_2019 pct\_foreign\_born\_2020 pct\_foreign\_born\_2021  
## 93 6.6 8.5 7.3  
## 1817 4.2 4.5 4.8  
## 2645 20.9 10.1 12.7  
## 2674 12.2 0.0 1.2  
## pct\_it\_workers\_2017 pct\_it\_workers\_2018 pct\_it\_workers\_2019  
## 93 3.3 3.9 5.3  
## 1817 0.8 0.7 0.8  
## 2645 0.0 0.0 0.0  
## 2674 0.0 0.0 0.0  
## pct\_it\_workers\_2020 pct\_it\_workers\_2021 median\_hh\_inc\_2017  
## 93 2.6 0.0 86019  
## 1817 0.4 0.7 33422  
## 2645 0.0 0.0 46534  
## 2674 0.0 0.0 80938  
## median\_hh\_inc\_2018 median\_hh\_inc\_2019 median\_hh\_inc\_2020  
## 93 82306 79867 82426  
## 1817 36049 39952 42264  
## 2645 53194 53088 45063  
## 2674 81875 83750 44076  
## median\_hh\_inc\_2021  
## 93 83765  
## 1817 46994  
## 2645 38659  
## 2674 58750

### Time Frame Determination

After dealing with the missing values, we will have to check the time frames provided in the **train** and **test** datasets.

## [1] "2019-08-01" "2019-09-01" "2019-10-01" "2019-11-01" "2019-12-01"  
## [6] "2020-01-01" "2020-02-01" "2020-03-01" "2020-04-01" "2020-05-01"  
## [11] "2020-06-01" "2020-07-01" "2020-08-01" "2020-09-01" "2020-10-01"  
## [16] "2020-11-01" "2020-12-01" "2021-01-01" "2021-02-01" "2021-03-01"  
## [21] "2021-04-01" "2021-05-01" "2021-06-01" "2021-07-01" "2021-08-01"  
## [26] "2021-09-01" "2021-10-01" "2021-11-01" "2021-12-01" "2022-01-01"  
## [31] "2022-02-01" "2022-03-01" "2022-04-01" "2022-05-01" "2022-06-01"  
## [36] "2022-07-01" "2022-08-01" "2022-09-01" "2022-10-01"

The training data time frame includes **08/2019** to **10/2022**.

## [1] "2022-11-01" "2022-12-01" "2023-01-01" "2023-02-01" "2023-03-01"  
## [6] "2023-04-01" "2023-05-01" "2023-06-01"

The prediction dates provided include **11/2022** to **06/2023**.

To make analysis easier and be able to group the data by year and month, we will use **substr()** function to extract the relevant characters of the first\_day\_of\_month column, which contains the date in the format “YYYY-MM-DD”. Then, **as.integer()** function is used to convert the extracted year and month values from character strings to integers.

## 'data.frame': 122265 obs. of 10 variables:  
## $ row\_id : chr "1001\_2019-08-01" "1001\_2019-09-01" "1001\_2019-10-01" "1001\_2019-11-01" ...  
## $ cfips : int 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 ...  
## $ county : chr "Autauga County" "Autauga County" "Autauga County" "Autauga County" ...  
## $ state : chr "Alabama" "Alabama" "Alabama" "Alabama" ...  
## $ first\_day\_of\_month : Date, format: "2019-08-01" "2019-09-01" ...  
## $ microbusiness\_density: num 3.01 2.88 3.06 2.99 2.99 ...  
## $ active : int 1249 1198 1269 1243 1243 1242 1217 1227 1255 1257 ...  
## $ year : int 2019 2019 2019 2019 2019 2020 2020 2020 2020 2020 ...  
## $ month : int 8 9 10 11 12 1 2 3 4 5 ...  
## $ year\_month : chr "2019-08" "2019-09" "2019-10" "2019-11" ...

### Merging the Dataframes

The merging process is challenging because all data fields provided in the **imputed\_data (**formerly **census\_df)** dataframe have a two-year lag to match the data in the **train\_df** and **test\_df** dataframes. Also, the data provided in the **imputed\_data** is on a yearly basis, but the data in the **train\_df** dataframe is on a monthly basis. The result of merging train\_df and imputed\_data will be called merged\_df and the result of To merge these two dataframes, it is assumed that the yearly data provided is valid for all the months of the corresponding year. For example, data provided in the pct\_bb\_2017 is valid for all the months of *2019* in the **train\_df**.

colSums(is.na(merged\_df))

## row\_id cfips county   
## 28551 0 28551   
## state first\_day\_of\_month microbusiness\_density   
## 28551 28551 28551   
## active year\_month year   
## 28551 28551 0   
## month pct\_bb pct\_college   
## 0 0 0   
## pct\_foreign\_born pct\_it\_workers median\_hh\_inc   
## 0 0 0

## row\_id cfips first\_day\_of\_month year\_month   
## 50328 0 50328 50328   
## year month pct\_bb pct\_college   
## 0 0 0 0   
## pct\_foreign\_born pct\_it\_workers median\_hh\_inc   
## 0 0 0

Since the data from 1/2019 to 7/2019 and 11/2022 to 12/2022 is not available in **train\_df** merging the data has created NA values in **merged\_df** for those months. Now we have to remove the rows with missing values.

# remove NA values created in merged\_df  
merged\_df <- merged\_df %>%  
 na.omit(merged\_df)  
  
merged\_test <- merged\_test %>%  
 na.omit(merged\_test)

## Descriptive Statistics & Multidimensional Data Analysis

### Data Visualization

The main feature in this project is microbusiness\_density provided in the **merged\_df.** Also, the number of active microbusinesses is provided in the active column.

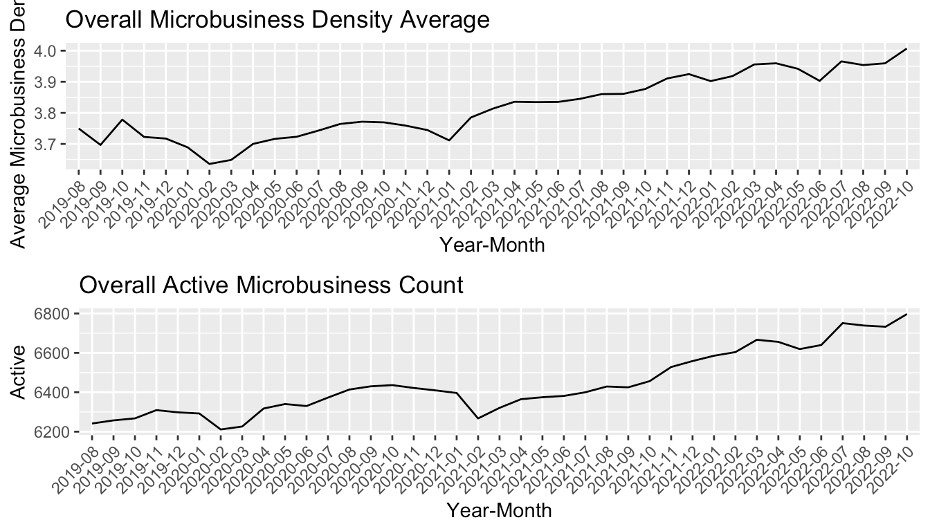
Boxplots are a visualization tool that provide insights into the central tendency and spread of a dataset, as well as identify outliers and skewness. They are useful for detecting anomalies and comparing variable distributions in a dataset, providing valuable insights into data distribution for exploratory data analysis.

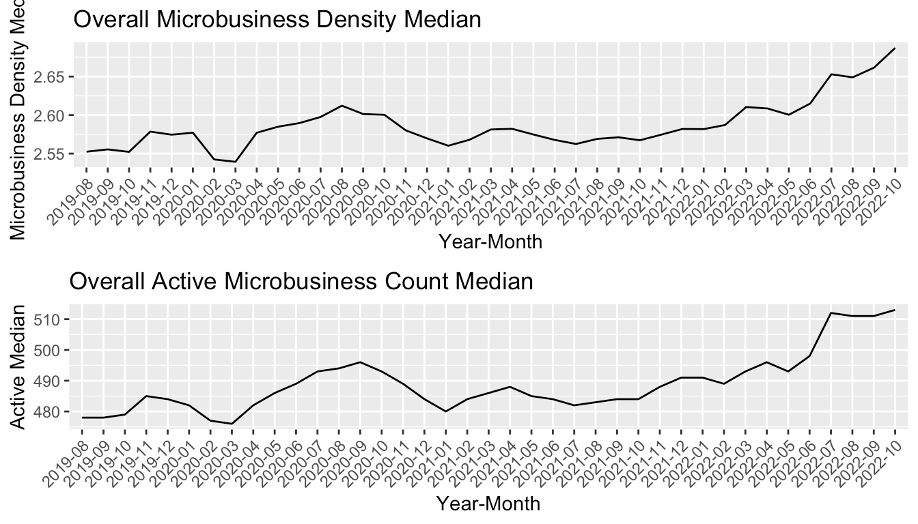
Chart, box and whisker chart

Description automatically generated

#### Microbusiness Density and Count

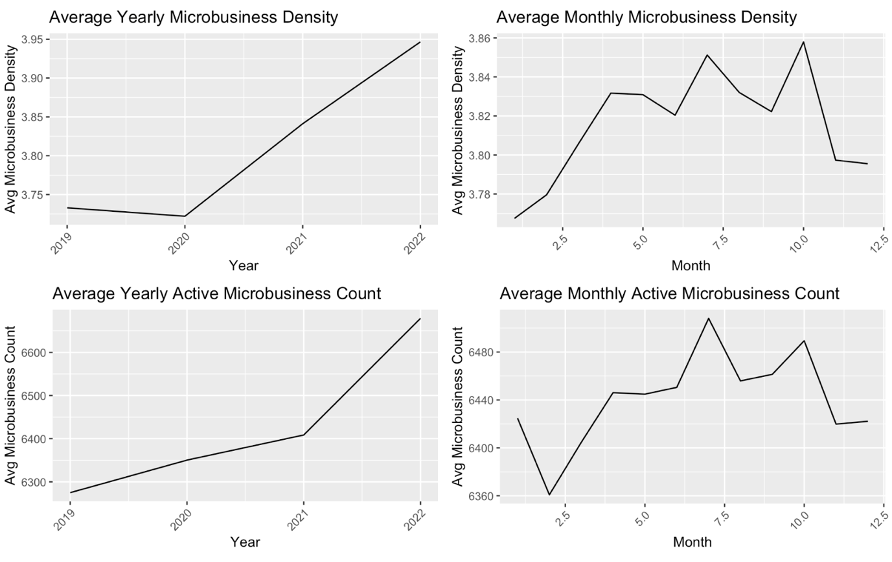
First, we will plot overall microbusiness density and count of active microbusiness in the United States in the complete timeframe:





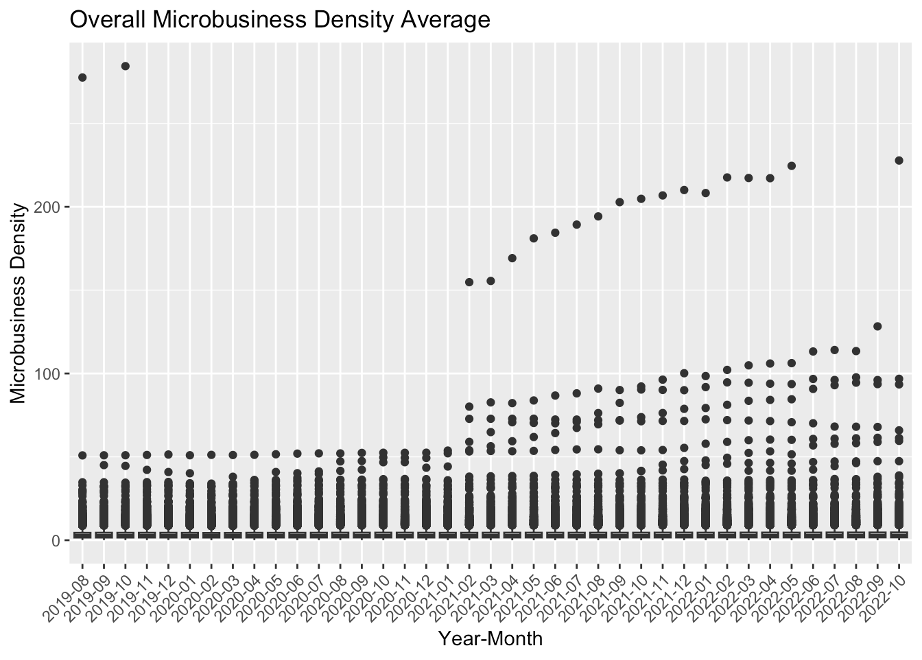
As expected, these two graphs show almost similar behavior. If we ignore the slight fluctuations of the two graphs, the general microbusiness density and count are growing over the whole time frame.

Then, we will examine the behavior of these two variables (*microbusiness density* and *active*) while grouping the data by month and year:

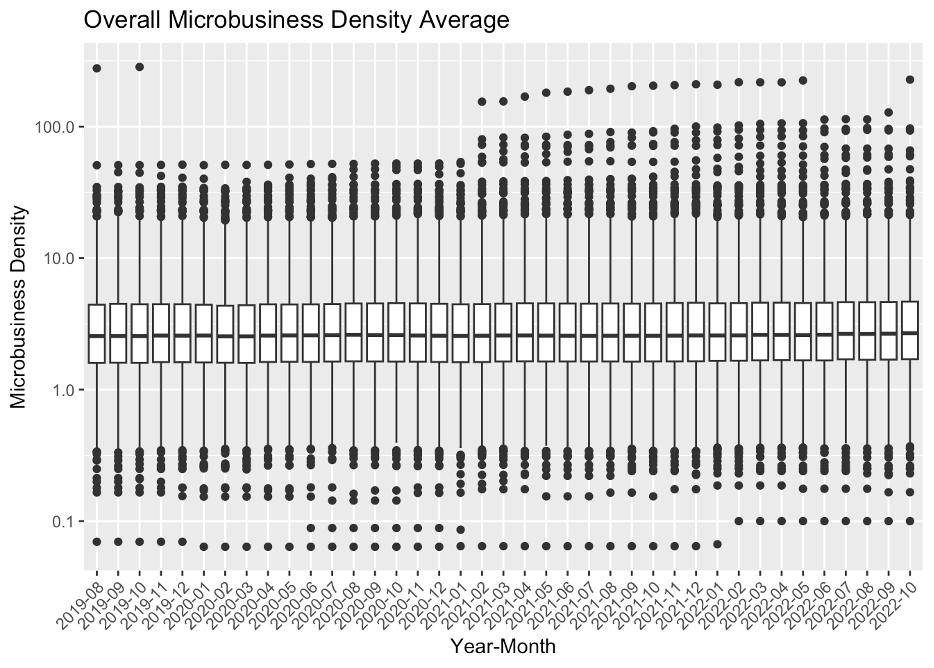


The left plots show that the average microbusiness density has increased slightly over the years, starting at approximately 3.73 in 2019 and reaching 3.94 in 2022. On the other hand, the average active count has also increased, starting at approximately 6274 in 2019 and reaching 6679 in 2022. In comparison, the right plots show fluctuations in the monthly averages for both variables. Generally, it follows a slightly upward trend over the year, with some peak values observed in July and October for the microbusiness density and active count, respectively. These peak values may represent seasonal variations, indicating that microbusinesses are more active during certain months. Overall, the plot shows some correlation between the monthly average values of microbusiness\_density and active count, indicating that common factors may influence both variables.

We can visualize the overall average microbusiness density using boxplots to gain a better understanding and additional information like central tendency and spread of this parameter from the plot.



Above plot is not very informative because the small values are obscured by the larger ones. Therefore, using a logarithmic scale on the Y-axis can help to reduce this distortion and provide a more informative visualization of the data.



This plot is more informative than a line plot, because the approximate value of the minimum, maximum, median, and first and third quartiles are also available.

According to the above plot, it is obvious that most of the average microbusiness density values are in the (1, 10) interval. Also, the median of the average microbusiness density is almost equal in the whole timeframe.

We can explore our merged\_df dataframe further according to the other metrics available.

#### Economic Regional Divisions

The Bureau of Economic Analysis (BEA) divides the United States into eight distinct economic regions[[1]](#footnote-1).

These regions are based on similarities in economic characteristics such as industry composition, income levels, and employment patterns. The eight regions are:

1. **New England**: Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island, and Vermont.

* *The economy in this region is largely based on manufacturing, healthcare, education, and finance.*

1. **Mideast**: Delaware, Maryland, New Jersey, New York, Pennsylvania, and the District of Columbia.

* *The region has a diverse economy, with a mix of manufacturing, finance, healthcare, and professional services.*

1. **Great Lakes**: Illinois, Indiana, Michigan, Ohio, and Wisconsin.

* *The region has a strong manufacturing base, particularly in the automotive industry, and also has a significant healthcare sector.*

1. **Plains**: Iowa, Kansas, Minnesota, Missouri, Nebraska, North Dakota, and South Dakota.

* *Agriculture and energy production are major industries in this region, along with manufacturing and healthcare.*

1. **Southeast**: Alabama, Arkansas, Florida, Georgia, Kentucky, Louisiana, Mississippi, North Carolina, South Carolina, Tennessee, Virginia, and West Virginia.

* *The Southeast has a diverse economy, with significant industries in healthcare, finance, and manufacturing, as well as tourism and agriculture.*

1. **Southwest**: Arizona, New Mexico, Oklahoma, and Texas.

* *The region has a strong energy sector, particularly in oil and gas production, and also has significant industries in manufacturing, healthcare, and finance.*

1. **Rocky Mountain**: Colorado, Idaho, Montana, Utah, and Wyoming.

* *The region is known for its natural resources, particularly in mining and energy production, as well as tourism, healthcare, and manufacturing.*

1. **Far West**: Alaska, California, Hawaii, Nevada, Oregon, and Washington.

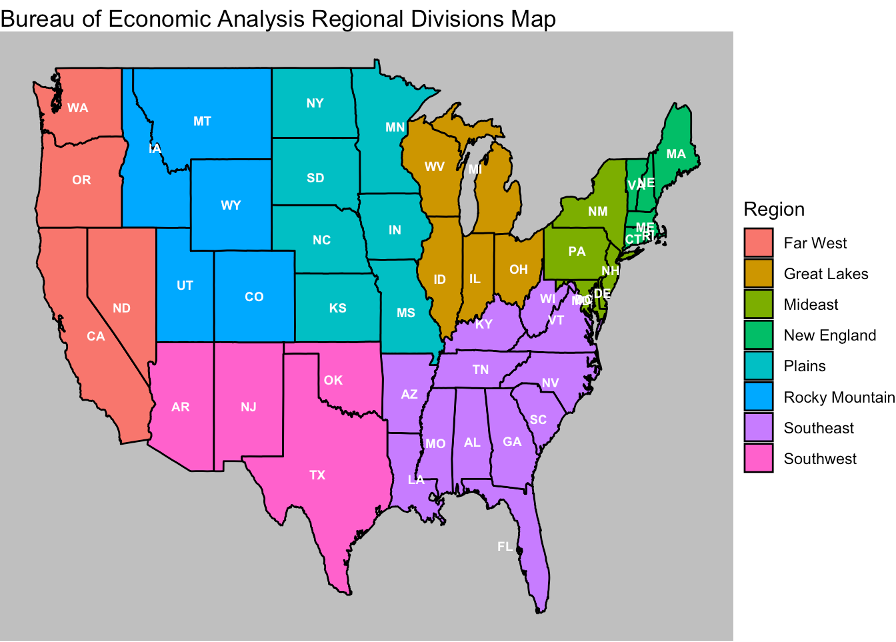
* *This region has a diverse economy, with significant industries in technology, finance, healthcare, and manufacturing, as well as tourism and agriculture.*

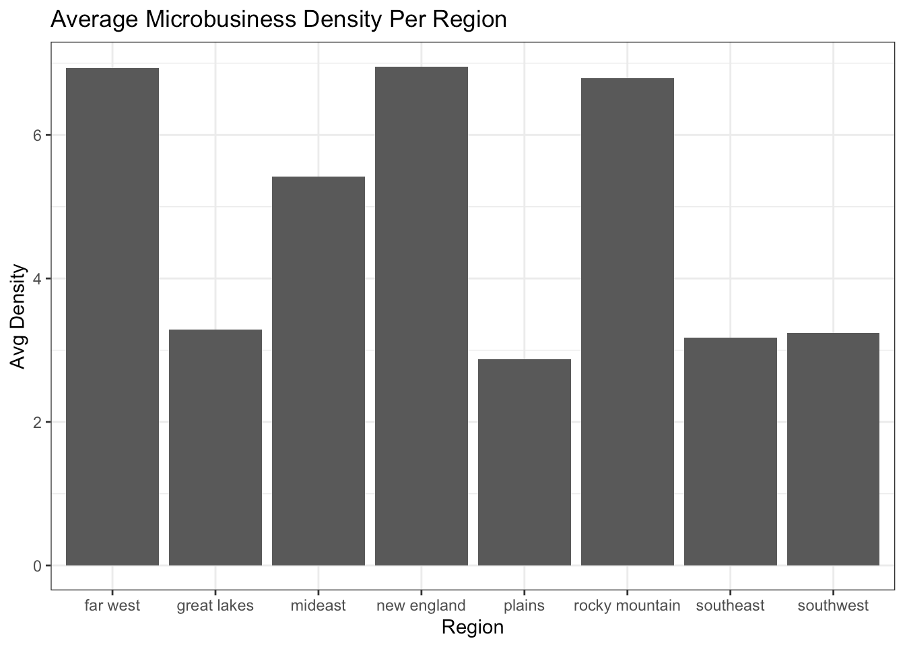
To draw the map for the BEA regions, first, we need to convert state and county columns in **merged\_df** to lowercase letters. Merging two dataframes will cause problems because the data from **map\_data()** will be in lowercase letters.

Then, we’ll create a new column in **merged\_df** named region and assign region values based on state column:

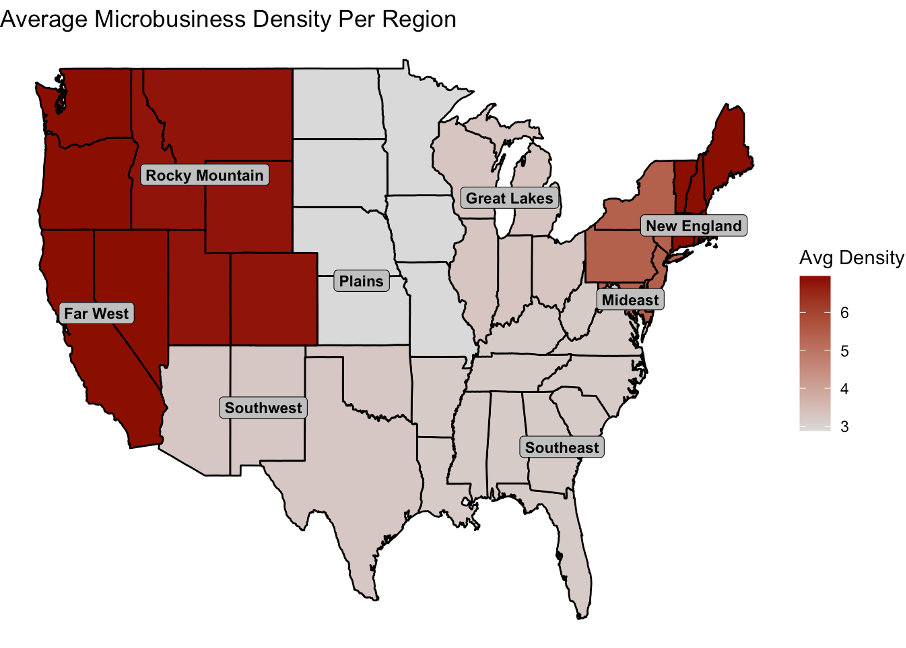
## [1] "southeast" "far west" "southwest" "rocky mountain"  
## [5] "new england" "mideast" "great lakes" "plains"

Now that the data in the dataframe matches the **map\_data()** output, we appoint each state to the region it belongs to and then use **ggplot()** to draw the map:

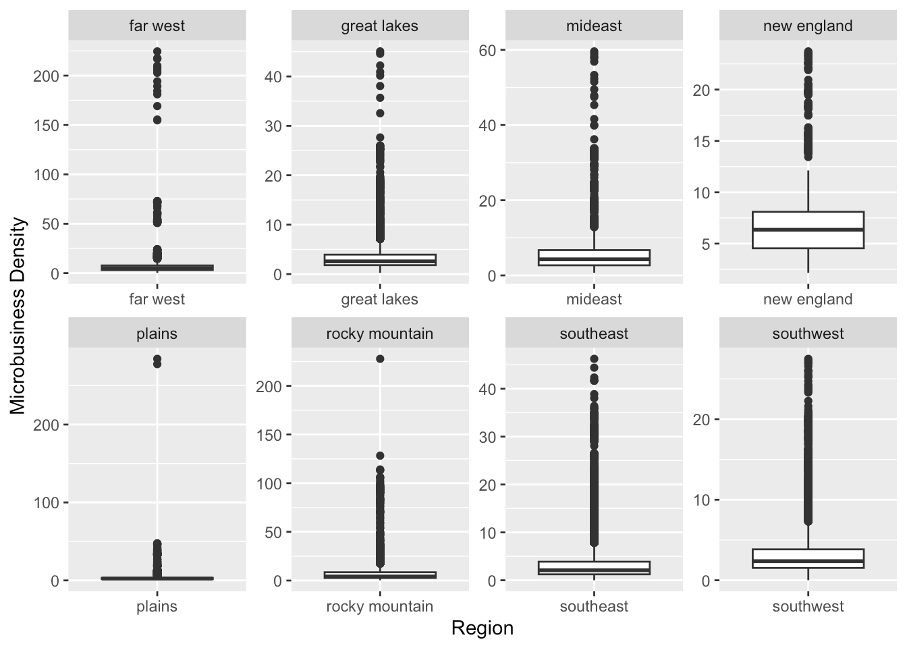




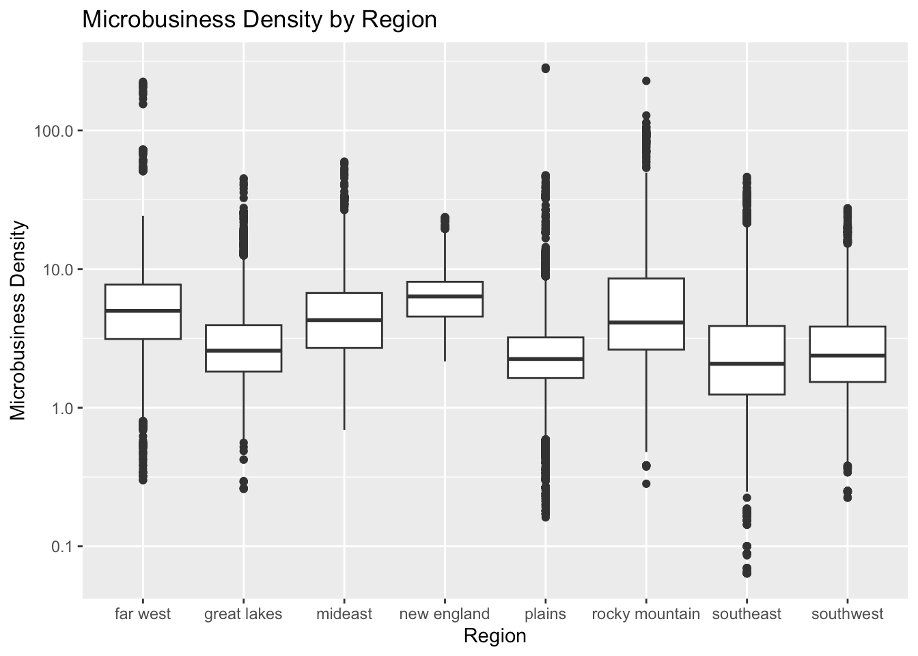
According to the above plot *New England* has the highest average microbusiness density, followed by *Farwest* and *Rocky Mountain* respectively, with a tiny difference, valuing more than 6.75. In contrast, *plains* has the lowest average microbusiness density, followed by *Southeast* and *Southwest*, all valued under 3.25. We can use a choropleth map to get a better view on the above information. A choropleth map provides an easy way to visualize how a variable varies across a geographic area or show the level of variability within one region or multiple regions.



Although, we can only see only one parameter on above map. To have a better look on the distribution, central tendency, spread, and variability of the microbusiness\_density variable, we can use boxplots.



Above plots are also not very informative because the small values are obscured by the larger ones. Therefore, using a logarithmic scale on the Y-axis can help to reduce this distortion and provide a more informative visualization of the data.

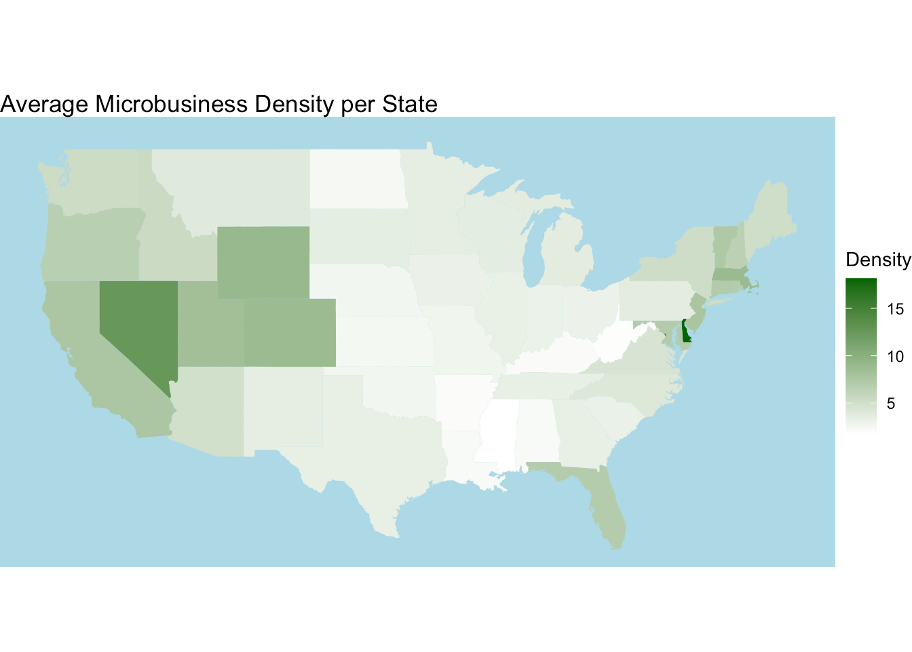


These boxplots are more informative, because using a logarithmic scale on the Y-axis helps to better reveal the differences and similarities between regions that helps to highlight any potential patterns or trends in the data. Some of the points that can be inferred from this boxplot include:

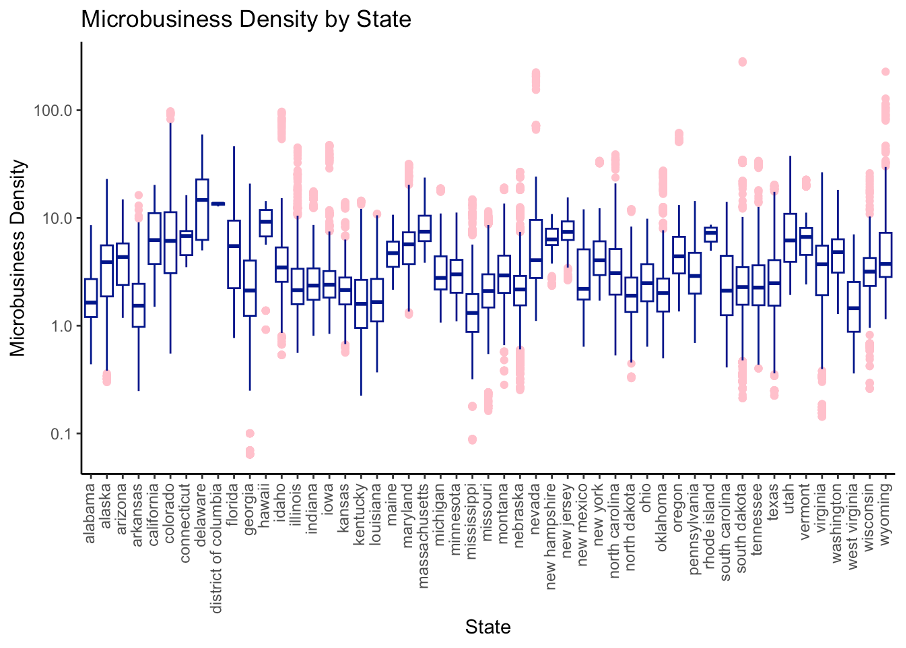
* The median microbusiness density is highest in the *New England* region, followed by the *Far West* and the *Mideast* regions.
* The 3rd quartile microbusiness density is highest in the *Rocky Mountain* region, followed by the *New England* and the *Far West* regions.
* The maximum microbusiness density is highest in the *Plains* region, followed by the *Rocky Mountain* and the *Far West* regions.
* The mean microbusiness density is highest in the *New England* region, followed by the *Far West* and the *Rocky Mountain* regions.
* The interquartile range (IQR = the difference between the 1st and 3rd quartiles) of microbusiness density is widest in the *Rocky Mountain* region, indicating that there is a greater range of microbusiness density in that region. In contrast, the IQR is narrowest in the *Plains* region.

#### U.S. State Divisions

We can use a choropleth map to get a view on the average microbusiness density for each U.S. state.



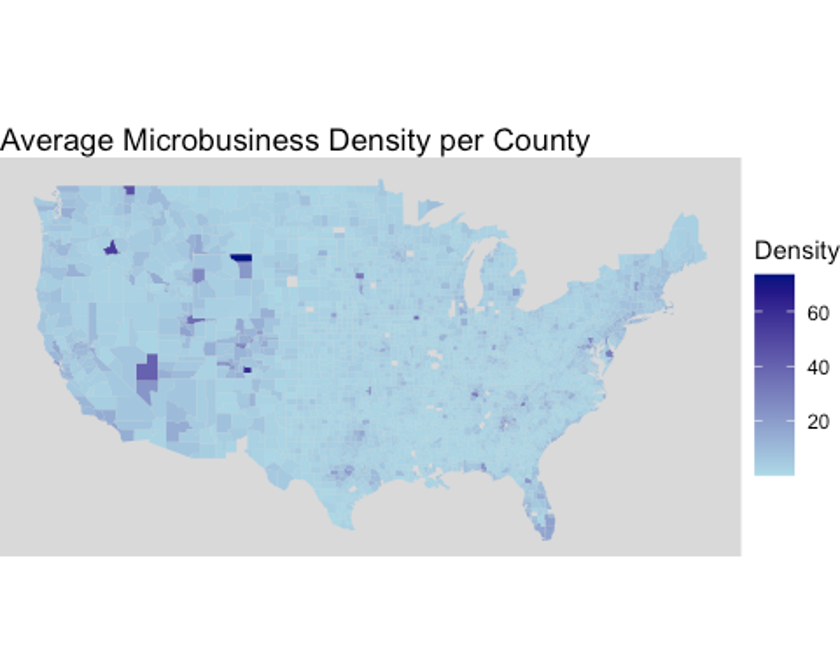
Although, this map provides an easy way to visualize how average microbusiness density varies across U.S. states but, as mentioned earlier it only demonstrates one parameter (average microbusiness density). Therefore, we can visualize average microbusiness density per each U.S. state using boxplots to gain deeper understanding of this parameter.



This boxplot provides a lot of information, also the logarithmic scale on y-axis prevents obscuring small values by larger values, especially the outliers. Also, we know each state belongs to which region so, this plot can be compared with the microbusiness density in regions to provide new useful insights. In addition, this plot is providing good information on the outliers in each state microbusiness density values. We will use the information interpreted from this plot in the upcoming data analysis and outlier detection.

#### U.S. County Divisions

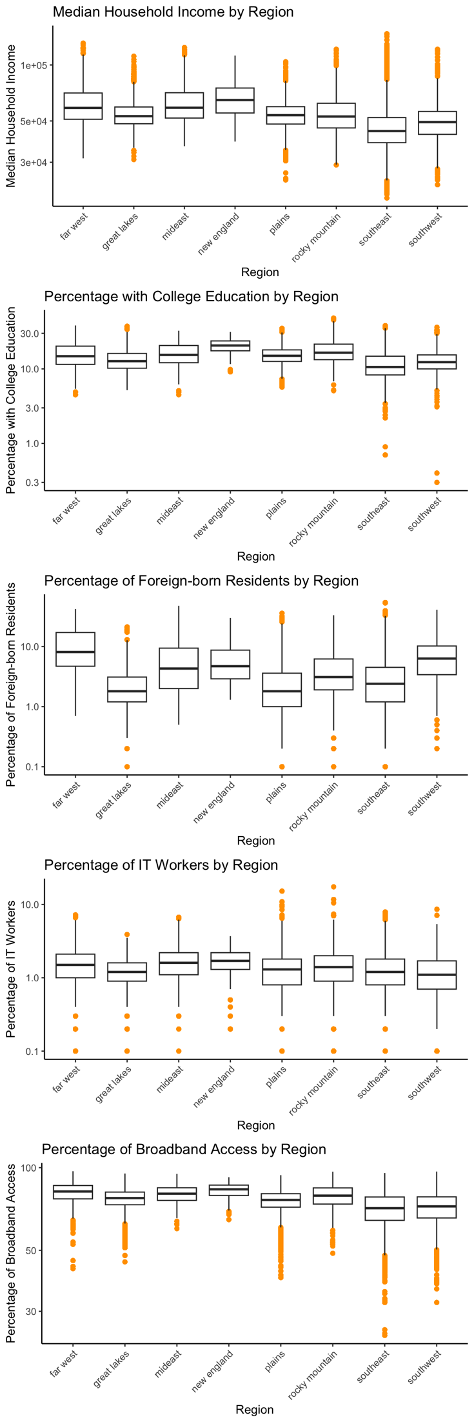
We can continue our analysis on the county level. There are around 3000 counties in the U.S., therefore, a choropleth map will prove to be useful since we cannot fit 3000 line or boxplots in a single figure.

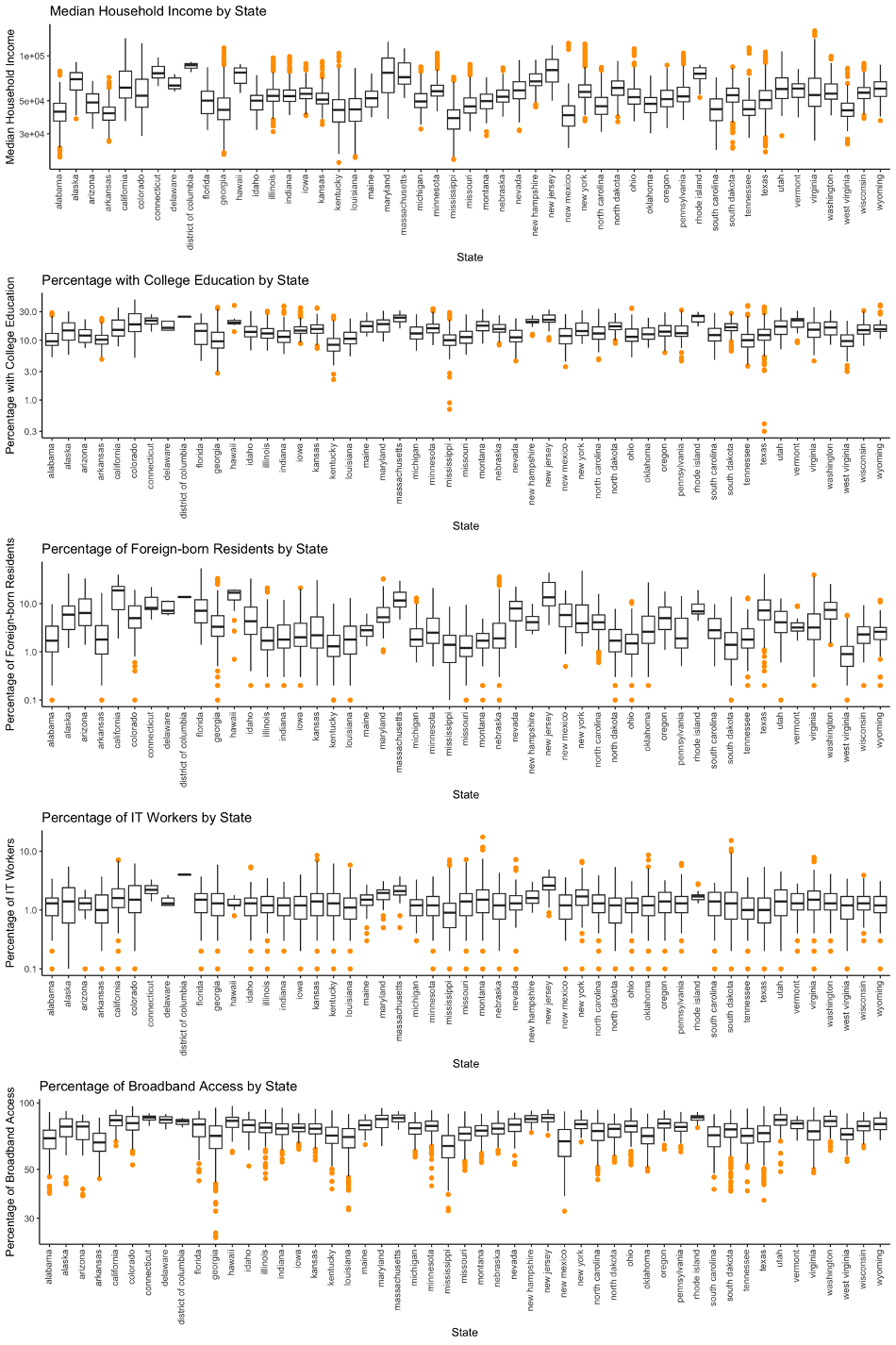


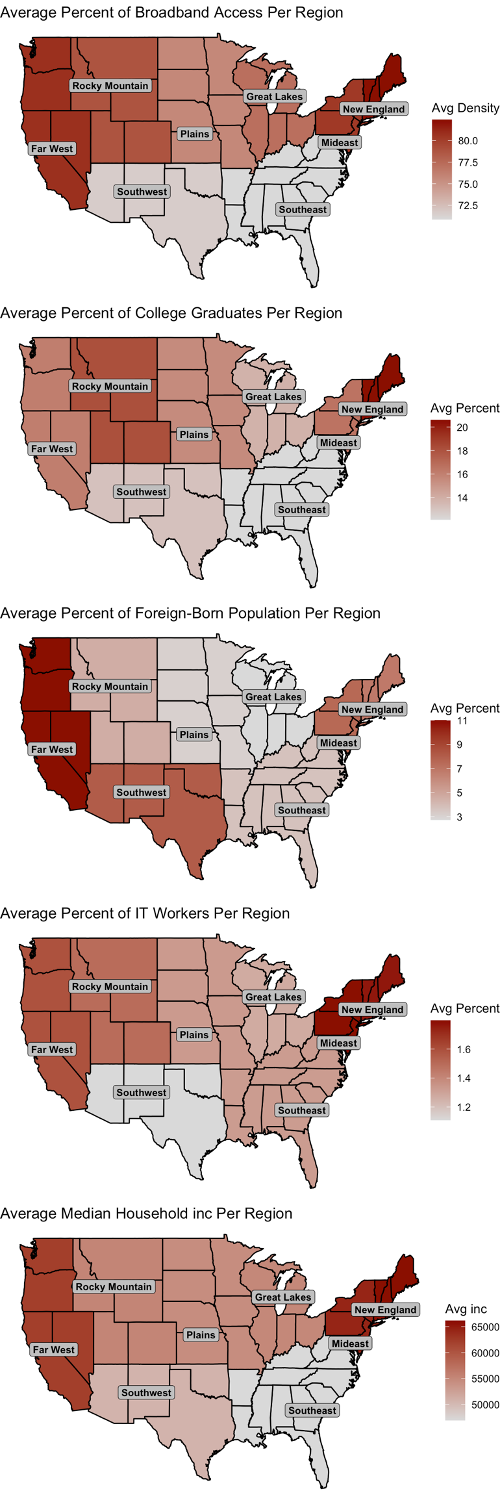
The distribution of the microbusiness density throughout the U.S. on county level is almost monolith with a few exceptions. It is necessary to analyze the data on county level to acquire cleaner data since there might be outliers on the county level data.

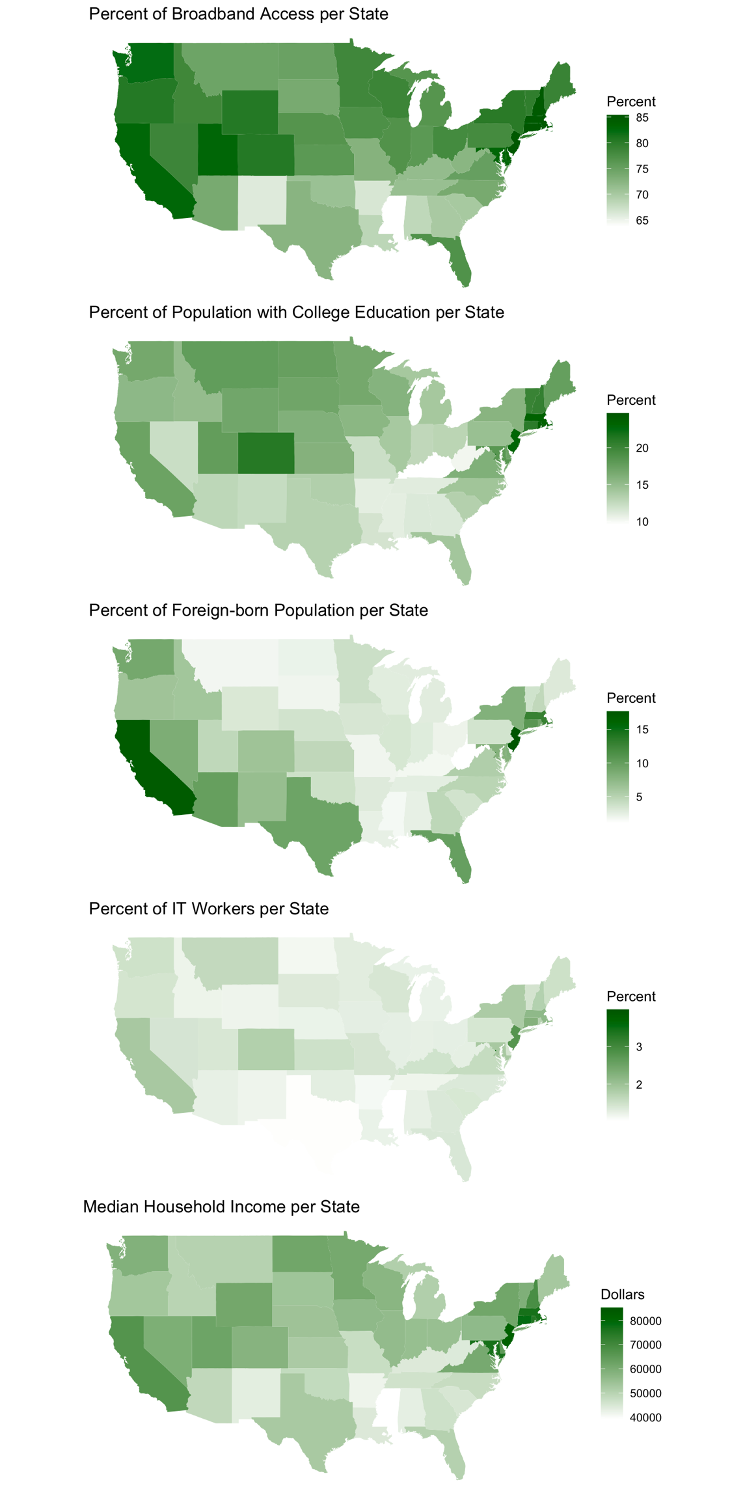
#### Census Data Visualizations

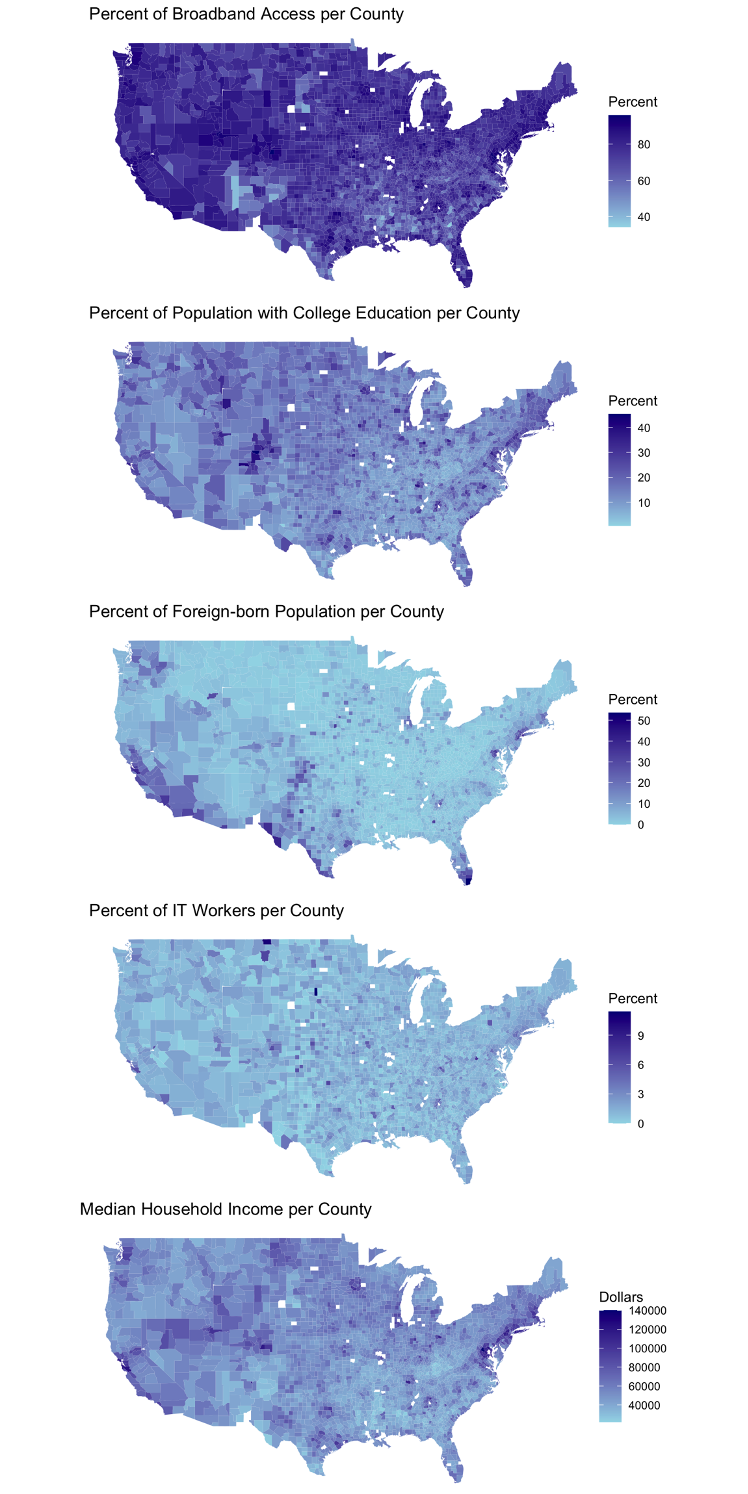
Since the data from the census dataset and train dataset are merged, we can analyze our census data on the region, state, and county levels. In the upcoming pages you can see the boxplots and choropleth maps of different census variables including median household income, percentage of people with college education, percentage of foreign-born residents, percentage of IT industry workers, and percentage of broadband access among residents for further assessment and evaluation.

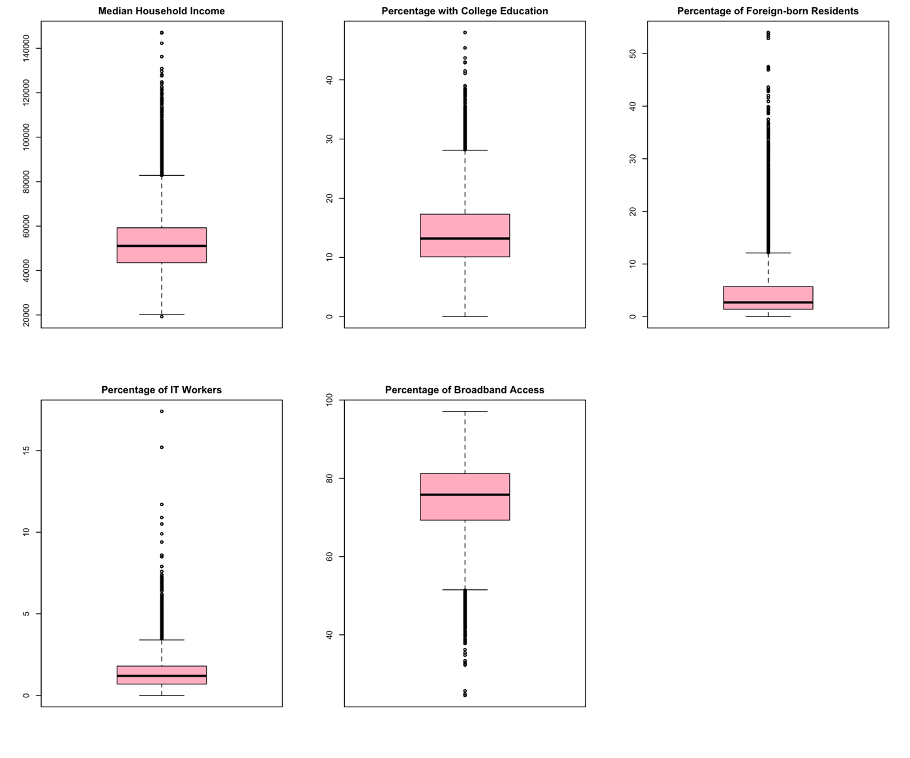












### Outlier Detection

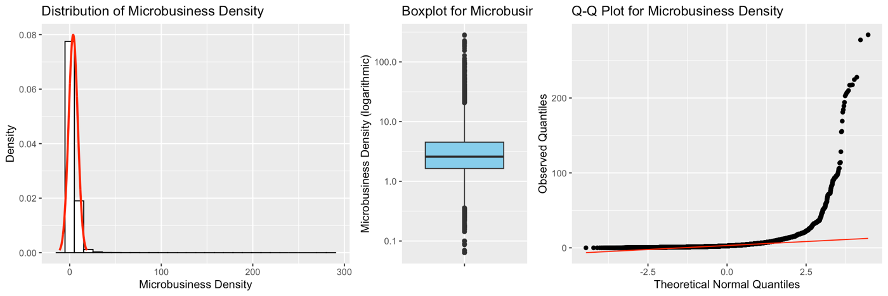
Outlier detection is an important step in data analysis, Outliers can potentially skew statistical measures, such as means and standard deviations, leading to biased results. They can also affect the assumptions of certain statistical models or algorithms. There are several methods to detect outliers depending on the distribution of the data.

The Shapiro-Wilk normality test is a statistical test used to determine if a given dataset follows a normal distribution. We will perform a Shapiro-Wilk normality test on a random sample of 5000 observations from the microbusiness\_density column of **merged\_df** dataframe. To run this test we first set the seed value using **set.seed()** function to a specific random seed to ensure that the results are reproducible if the code is run again. Then, we weill use the **shapiro.test()** function to perform the Shapiro-Wilk normality test on the **sample\_data** object. The function returns the test statistic (W) and the p-value. A W value closer to 1 indicates that the data is more normally distributed, while a W value closer to 0 indicates greater deviation from normality. If the p-value is less than the significance level (typically 0.05), then the null hypothesis (that the sample data is normally distributed) is rejected in favor of the alternative hypothesis (that the sample data is not normally distributed).

##   
## Shapiro-Wilk normality test  
##   
## data: sample\_data  
## W = 0.55179, p-value < 2.2e-16

The test resulted in a W statistic of 0.55179 and a p-value of less than 2.2e-16. Based on the results of the Shapiro-Wilk normality test, it can be concluded that the **sample\_data** is not normally distributed.

To have a better visual on distribution of the data we will use a bell curve, a boxplot, and a Q-Q plot on microbusiness\_density.



The above distribution plot explains that the dataset is right-skewed. The boxplot shows some data points away from the upper whisker; hence outliers are present in microbusiness\_density. Q-Q plot’s alignment is away from the 45-degree angle depicting outliers in the dataset.

## microbusiness\_density   
## 0% 25% 50% 75% 100%   
## 0.000000 1.639344 2.586543 4.519231 284.340030

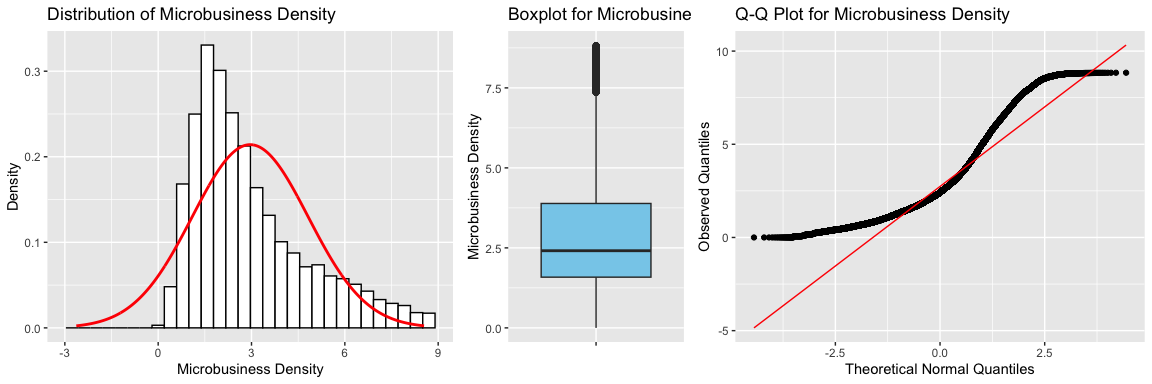
Since the data is right-skewed and not normally distributed, a common approach to detecting outliers is the decision range approach which involves setting a range of values outside of which any observations are considered outliers. A common method is to use the interquartile range (IQR) to define the decision range. The IQR is calculated as the difference between the third quartile (Q3) and the first quartile (Q1) of the data. The decision range is then defined as the range from to . Any observations that fall outside of this range are considered outliers. This method is useful for identifying potential outliers in a dataset and can help to ensure that statistical analyses are robust and accurate. To do this, we will calculate IQR of microbusiness\_density column, then we’ll calculate lower and upper bounds for outliers and count number of outliers. Finally, we calculate the percentage of outliers and create a new dataframe named **merged\_df\_new** without the outliers. We can also verify the outlier removing procedure by calculating the difference between rows of **merged\_df** and **merged\_df\_new**.

## Number of outliers: 8746

## Percent of outliers: 7.153315 %

## Number of rows removed: 8746

Now, we can again plot the bell curve, boxplot, and Q-Q plot on microbusiness\_density of **merged\_df\_new**.



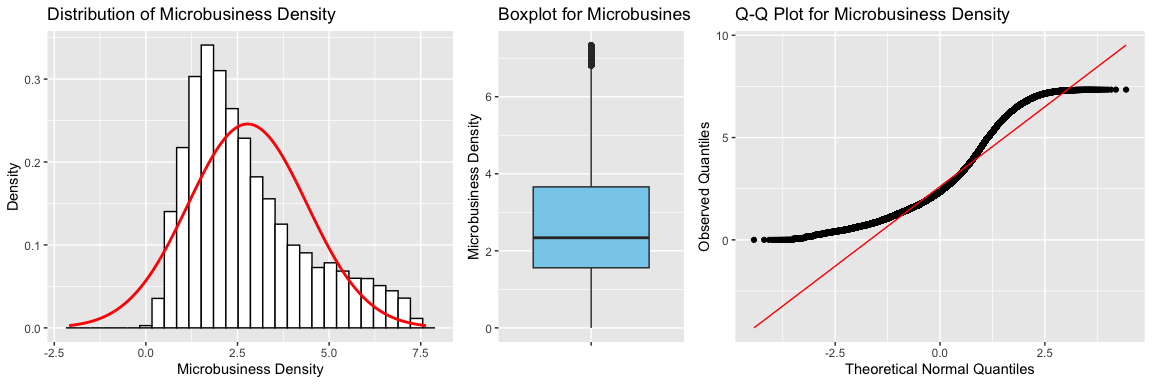
The distribution of microbusiness density has improved dramatically after removing the outliers using the decision range approach. We can repeat the mentioned method to further remove the values that we are now considering outliers.

## Number of outliers: 3946

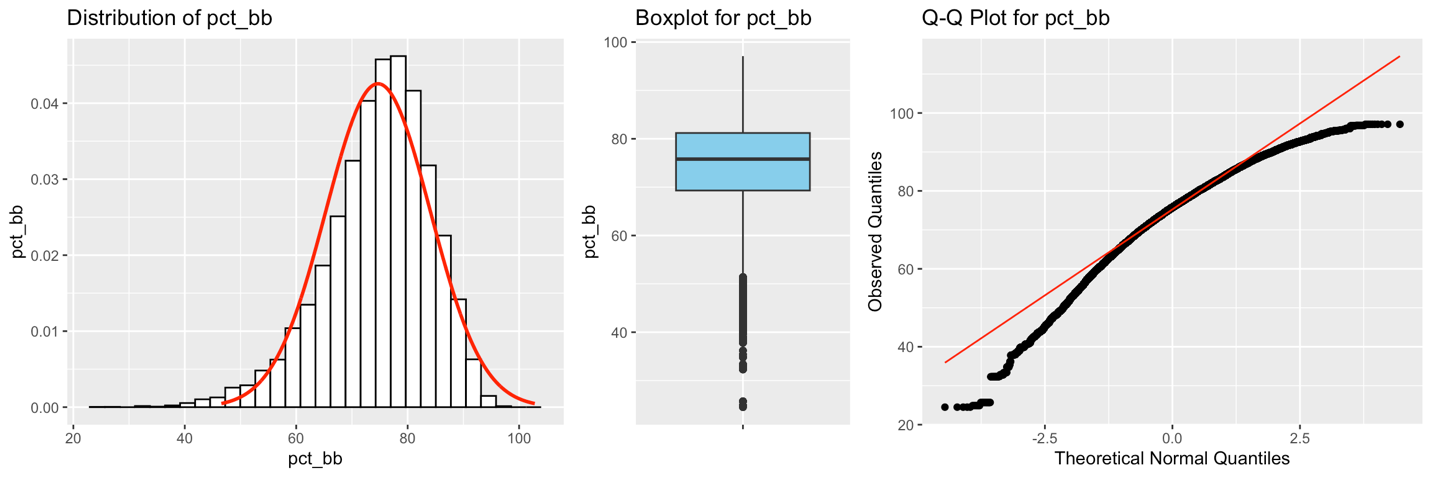
## Percent of outliers: 3.47607 %

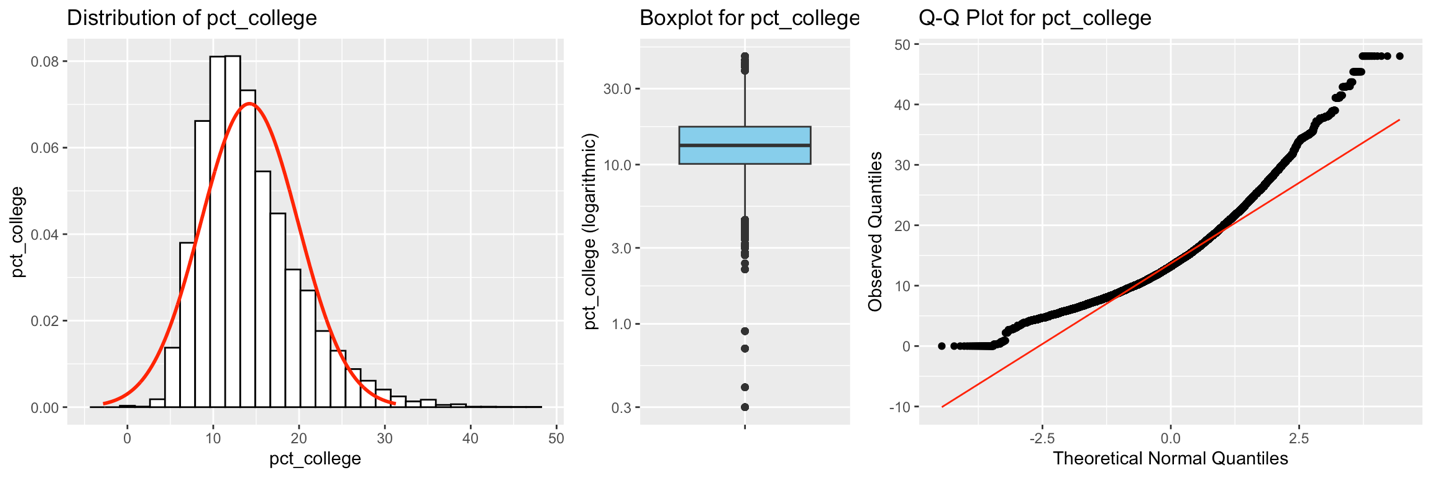
## Number of rows removed: 3946

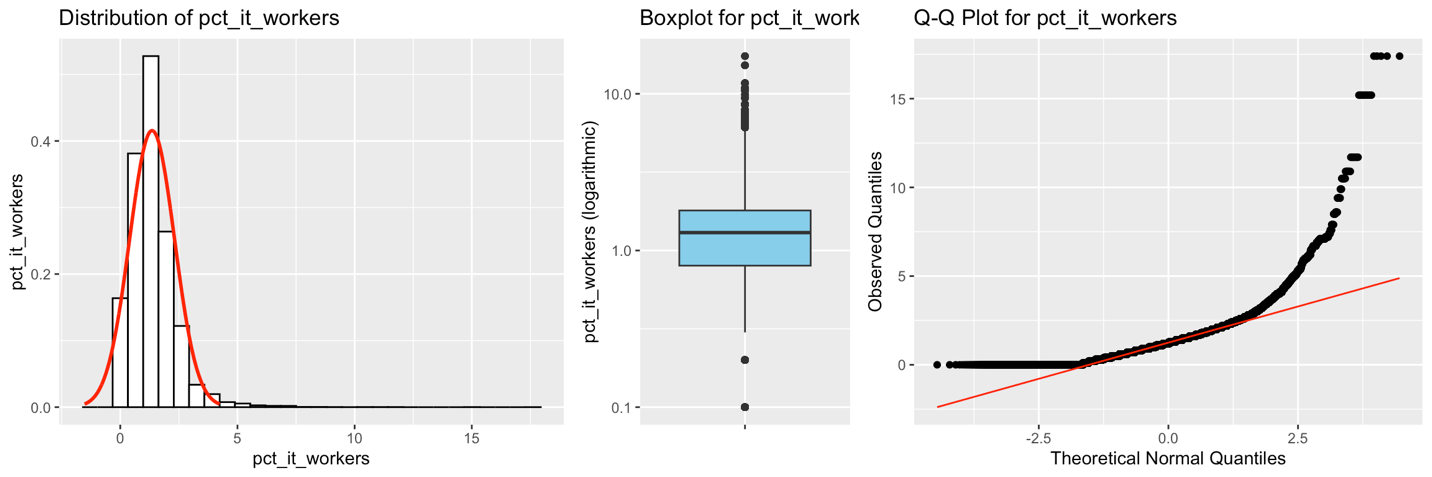
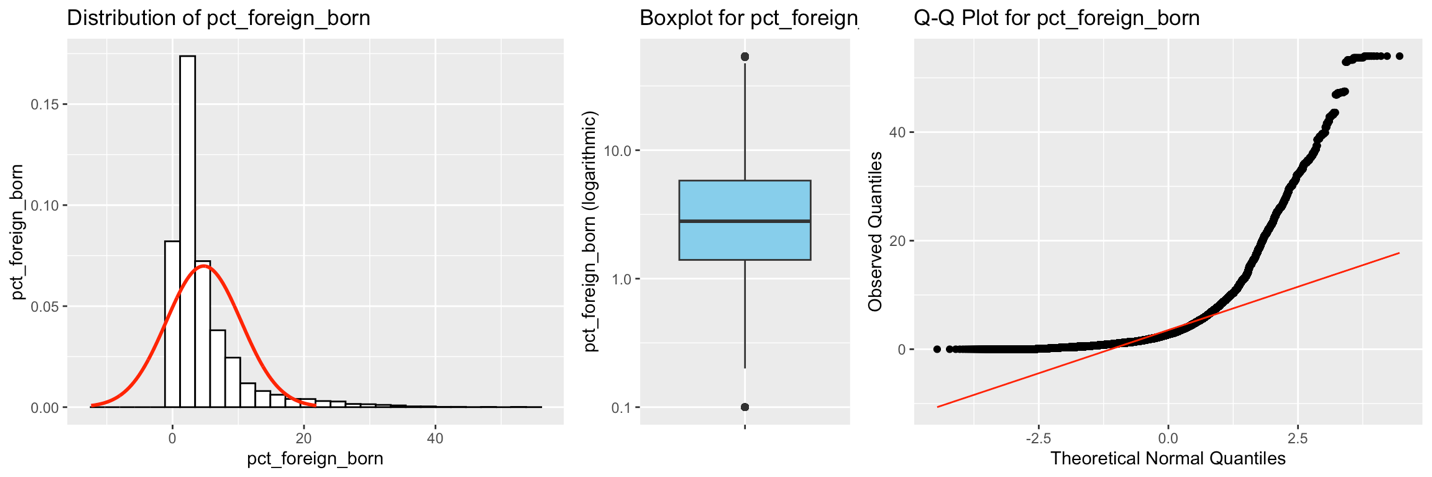
## Total rows removed from merged\_df: 12692



We can repeat the above steps for the census features (pct\_bb, pct\_college, pct\_foreign\_born, pct\_it\_workers, and median\_hh\_inc) as well.









Above plots suggest that there are outliers available in pct\_bb, pct\_college, pct\_foreign\_born, pct\_it\_workers, and median\_hh\_inc vailable too. We repeat the same decision range approach with IQR method on these features to remove the outliers.

## Number of outliers in pct\_bb : 2507   
## Percent of outliers in pct\_bb : 2.208441 %

## Number of outliers in pct\_college : 1428   
## Percent of outliers in pct\_college : 1.257939 %

## Number of outliers in pct\_it\_workers : 2556   
## Percent of outliers in pct\_it\_workers : 2.251605 %

## Number of outliers in pct\_foreign\_born : 8329   
## Percent of outliers in pct\_foreign\_born : 7.337098 %

## Number of outliers in median\_hh\_inc : 3410   
## Percent of outliers in median\_hh\_inc : 3.003902 %

## Original dataframe length: 113519

## Cleaned dataframe length: 97098

## Total rows removed: 25167

## Total percent removed: 20.58398 %

The highest count of outliers belongs to the percent of foreign born with more than 7% of the data. We removed 25167 rows from our merged dataframe in total. Now that we’ve cleared the outliers from our features and microbusiness\_density as well, we can proceed with patterns, similarities, and correlations.

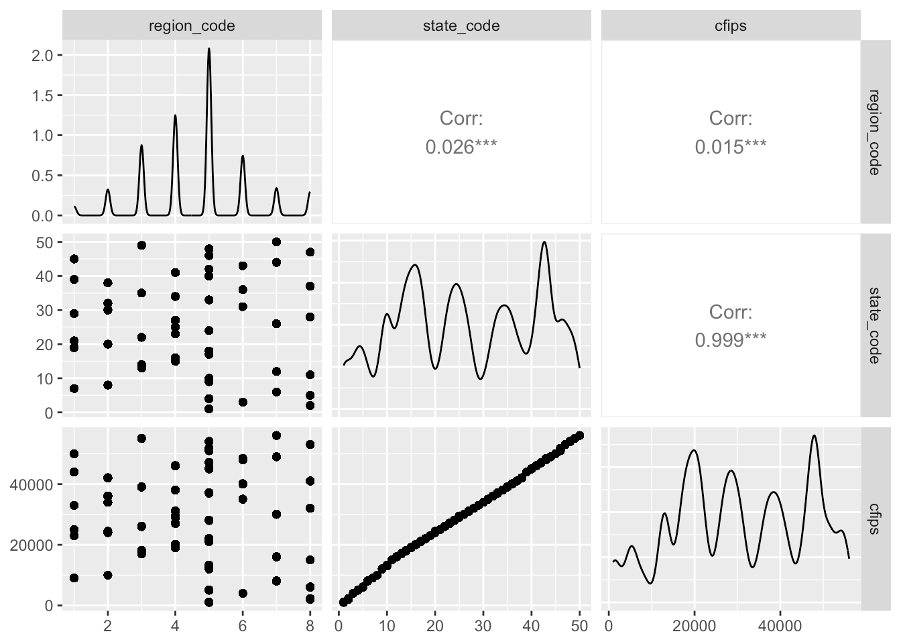
### Correlation Plot

A correlation plot, also called a correlation matrix or a heatmap, is a graphical representation of the pairwise correlations between variables in a dataset or dataframe. It shows the strength and direction of the linear relationship between each pair of variables, usually using a color scale to indicate the magnitude of the correlation coefficient.

Correlation plots are useful for exploring the relationships between multiple variables in a data set and identifying patterns or trends. They can reveal which variables are strongly positively or negatively correlated with each other, which variables are independent, and which variables may be redundant or highly related. They are often used in data analysis and EDA to gain insights into the relationships between variables. They can also be used as a tool for feature selection, where highly correlated variables can be identified and removed to improve model performance.

Correlation plots can be particularly useful when dealing with high-dimensional data, where it may be difficult to visualize or analyze the relationships between all the variables. By summarizing the pairwise correlations in a single plot, these plots can provide a quick and intuitive overview of the data and help guide further analysis.

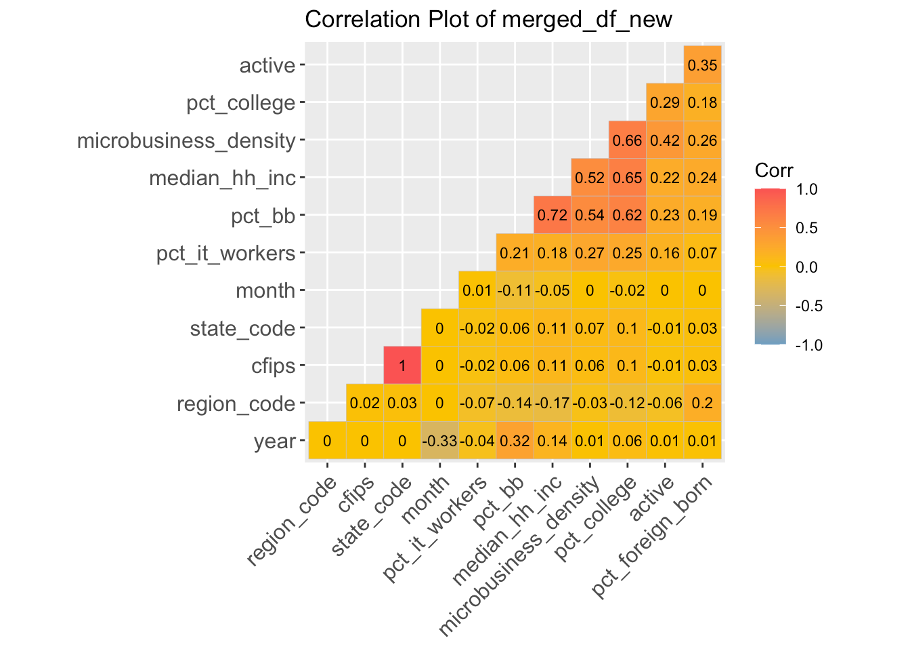
The correlation coefficient, which ranges from -1 to 1, measures the strength and direction of the relationship between two variables. A correlation coefficient of 1 indicates a perfect positive correlation, while a correlation coefficient of -1 indicates a perfect negative correlation.



For example, the above correlation plot on cfips, state\_code, and region\_code demonstrates an almost perfect positive correlation between the cfips and state\_code, which means that as one variable increases, the other variable also increases. The reason behind this high correlation value is because state\_code variable is defined alphabetically, at the same time cfips value of each county is also produced alphabetically and increases based on the state code.

It is important to note that high correlation does not necessarily imply causation. In other words, while two features may be highly correlated, this does not necessarily mean that one variable causes the other. Careful analysis and consideration of other factors are necessary to determine any causal relationship between the two variables.

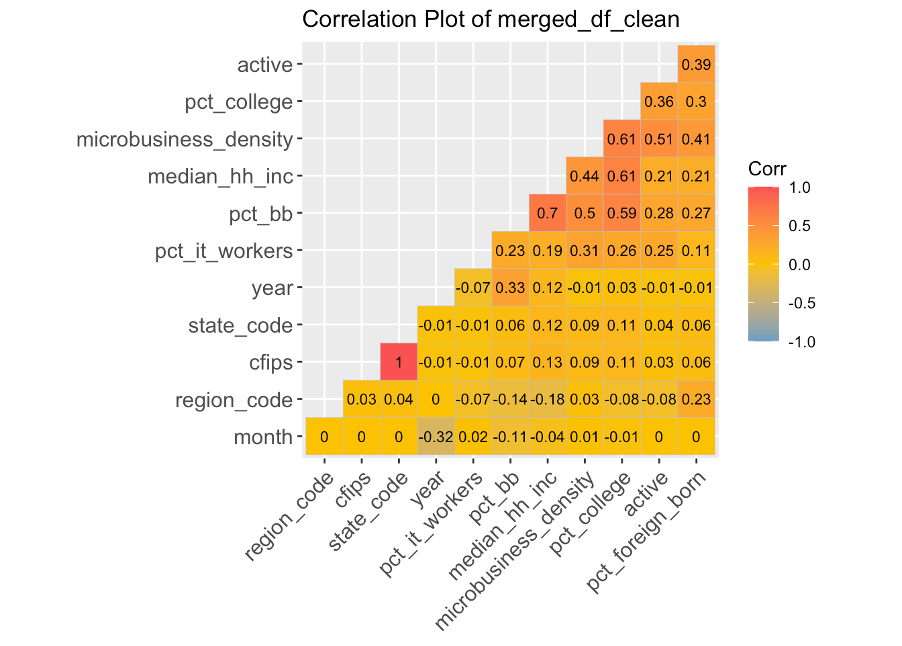
Now, we first plot the correlation between all the numeric features in our dataframe after removing the outliers from microbusiness\_density column. Then, we plot the same plot on the dataframe we created after removing outliers from microbusiness\_density and census features as well.



Based on the absolute value of the correlation coefficient, we define intervals to determine the strength of the correlation as follows:

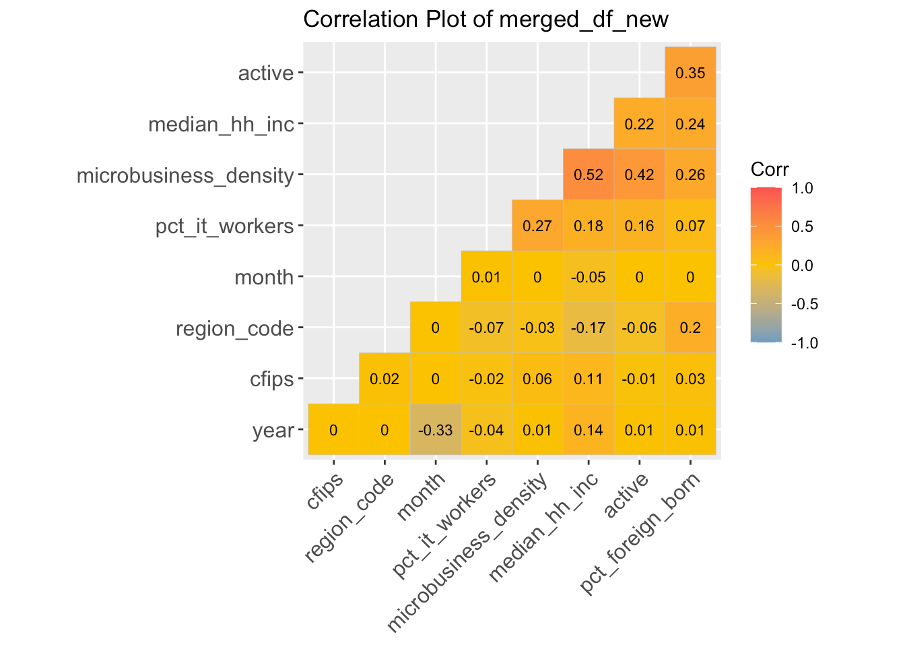
1. **0.01** to **0.19**: Very weak correlation
2. **0.20** to **0.39**: Weak correlation
3. **0.40** to **0.59**: Moderate correlation
4. **0.60** to **0.79**: Strong correlation
5. **0.80** to **0.99**: Very strong correlation

We can see the absolute positive correlation between cfips and state\_code. Also, there is a strong correlation between (median\_hh\_inc, pct\_bb), (pct\_college, microbusiness\_density), (pct\_college, pct\_bb) and (pct\_college, median\_hh\_inc) pairs as well.



The above correlation plot belongs to our merged dataframe after removing the outliers from the census columns as well as the microbusiness\_density from the dataframe. Based on the intervals we defined earlier, the correlation coefficient between pct\_bb and pct\_college after removing the outliers decreased from strong to moderate.

The overall correlation percentage decreased after removing the outliers from our dataframe. We will remove the redundant feature state\_code because it has absolute positive correlation with cfips. Also, median\_hh\_inc strongly correlates with pct\_bb. Also, pct\_college has a moderately strong correlation with microbusiness\_density and median\_hh\_inc. We will also remove median\_hh\_inc before further analysis. Still, we will not remove pct\_college because there is a limited number of features, and this feature might be useful in forecasting the microbusiness\_density value. Now, we will draw the new correlation plot after removing the redundant features.

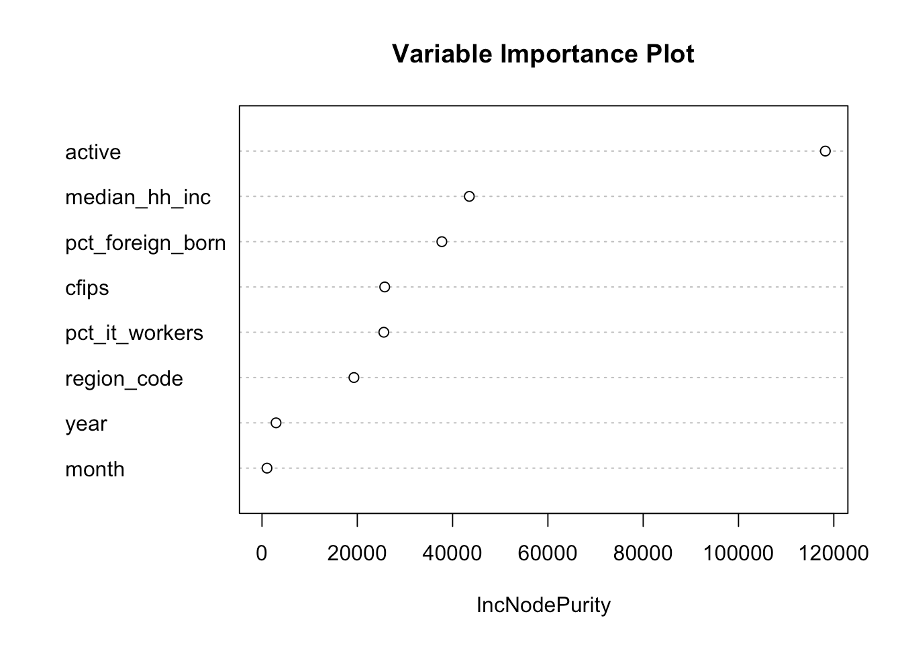


### Feature Importance Using Random Forest

Random Forest is a statistical learning and machine learning algorithm that is often used for classification and regression tasks. It is a type of ensemble learning method that combines the results of multiple decision trees to improve the accuracy of predictions. Each decision tree is trained on a random subset of the features, and the final prediction is made by averaging the predictions of all the trees. The feature importance of a random forest model is determined by calculating the decrease in the impurity of the nodes when splitting on a particular feature.

In other words, the feature importance score measures how much the accuracy of the model decreases when the values of a particular feature are randomly shuffled. The higher the decrease in accuracy, the more important the feature is considered to be in the model. This technique is used to identify the most significant features or variables that contribute to the prediction accuracy of a random forest model. This information can be used to simplify the model, improve its performance, or gain insights into the underlying data.

We will build a Random Forest model with active, region\_code, year, month, pct\_bb, pct\_college, pct\_foreign\_born, pct\_it\_workers, and median\_hh\_inc as predictors to predict the microbusiness\_density value. Then we will plot the variable importances and print the variable importance scores.



## Overall  
## cfips 20781.2157  
## region\_code 15544.1647  
## active 106976.1732  
## year 2312.1258  
## month 322.2596  
## pct\_bb 23669.2182  
## pct\_college 55792.0725  
## pct\_foreign\_born 24527.4115  
## pct\_it\_workers 13941.9729  
## median\_hh\_inc 17990.3220

The variable importance scores indicate how much each predictor contributes to the accuracy of the model in predicting the outcome. The higher the importance score, the more important the predictor is in the model. This output indicates that the most important predictors for predicting microbusiness\_density are active, pct\_college, pct\_foreign\_born, pct\_bb, and cfips, while year and month have the least impact on the model’s performance. We will use these results to decide which predictors to include in the model and which ones to exclude. Of course, we cannot use the active variable as a predictor in our model because there is a linear relation between microbusiness\_density and active values.

## Non-Parametric Tests

A non-parametric test is a statistical test that does not make any assumptions about the underlying distribution of the population from which the data is sampled. Instead, it uses alternative methods to test hypotheses, such as comparing the medians or ranks of the data. Non-parametric tests are often used when the data is not normally distributed. Since our data is not normally distributed, non-parametric tests may be more appropriate.

**Wilcoxon signed-rank test** and **Wilcoxon rank-sum test** are non-parametric statistical tests used to determine whether two samples come from populations with the same median. They are used when the data cannot be assumed to be normally distributed or when the sample sizes are small. Both tests do not assume any particular distribution of the data. Instead, they rely on the ranks of the observations, making them robust to outliers and other deviations from normality.

The Wilcoxon signed-rank test is used when there are paired samples, such as when the same group of individuals is measured before and after an intervention. The test compares the difference between the paired observations and tests whether the median of the differences is equal to zero.

We perform Wilcoxon signed-rank tests on pairs of variables in **merged\_df\_new** dataframe. We specify the variable pairs in a list and then loop through each pair in the list and perform the Wilcoxon signed-rank test using the **wilcox.test()** function. The "paired" argument is set to **TRUE**, to indicate that the test is being performed on paired samples. For each pair of variables, the code prints the variable pair being tested and the results of the test, including the test statistic, the p-value, and the alternative hypothesis.

## Wilcoxon signed-rank test results for microbusiness\_density and pct\_bb :  
##   
## Wilcoxon signed rank test with continuity correction  
##   
## data: merged\_df\_new[[pair[1]]] and merged\_df\_new[[pair[2]]]  
## V = 0, p-value < 2.2e-16  
## alternative hypothesis: true location shift is not equal to 0  
##   
##   
## Wilcoxon signed-rank test results for microbusiness\_density and pct\_college :  
##   
## Wilcoxon signed rank test with continuity correction  
##   
## data: merged\_df\_new[[pair[1]]] and merged\_df\_new[[pair[2]]]  
## V = 6780, p-value < 2.2e-16  
## alternative hypothesis: true location shift is not equal to 0  
##   
##   
## Wilcoxon signed-rank test results for microbusiness\_density and pct\_foreign\_born :  
##   
## Wilcoxon signed rank test with continuity correction  
##   
## data: merged\_df\_new[[pair[1]]] and merged\_df\_new[[pair[2]]]  
## V = 2843324130, p-value < 2.2e-16  
## alternative hypothesis: true location shift is not equal to 0  
##   
##   
## Wilcoxon signed-rank test results for microbusiness\_density and pct\_it\_workers :  
##   
## Wilcoxon signed rank test with continuity correction  
##   
## data: merged\_df\_new[[pair[1]]] and merged\_df\_new[[pair[2]]]  
## V = 5996100625, p-value < 2.2e-16  
## alternative hypothesis: true location shift is not equal to 0  
##   
##   
## Wilcoxon signed-rank test results for microbusiness\_density and median\_hh\_inc :  
##   
## Wilcoxon signed rank test with continuity correction  
##   
## data: merged\_df\_new[[pair[1]]] and merged\_df\_new[[pair[2]]]  
## V = 0, p-value < 2.2e-16  
## alternative hypothesis: true location shift is not equal to 0

The Wilcoxon rank-sum test, also known as the Mann-Whitney U test, is used when there are two independent samples. The test compares the ranks of the observations in the two samples and tests whether the medians of the two samples are equal.

Now, we perform Wilcoxon signed-rank tests on pairs of variables in **merged\_df\_new** dataframe. We specify the variable pairs in a list and then loop through each pair in the list and perform the Wilcoxon signed-rank test using the **wilcox.test()** function. The "paired" argument is set to **FALSE**, indicating that the test is being performed on two independent samples. For each pair of variables, the code prints the variable pair being tested and the results of the test, including the test statistic, the p-value, and the alternative hypothesis.

## Wilcoxon rank-sum test results for microbusiness\_density and pct\_bb :  
##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: merged\_df\_new[[pair[1]]] and merged\_df\_new[[pair[2]]]  
## W = 0, p-value < 2.2e-16  
## alternative hypothesis: true location shift is not equal to 0  
##   
##   
## Wilcoxon rank-sum test results for microbusiness\_density and pct\_college :  
##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: merged\_df\_new[[pair[1]]] and merged\_df\_new[[pair[2]]]  
## W = 112644499, p-value < 2.2e-16  
## alternative hypothesis: true location shift is not equal to 0  
##   
##   
## Wilcoxon rank-sum test results for microbusiness\_density and pct\_foreign\_born :  
##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: merged\_df\_new[[pair[1]]] and merged\_df\_new[[pair[2]]]  
## W = 6244818672, p-value < 2.2e-16  
## alternative hypothesis: true location shift is not equal to 0  
##   
##   
## Wilcoxon rank-sum test results for microbusiness\_density and pct\_it\_workers :  
##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: merged\_df\_new[[pair[1]]] and merged\_df\_new[[pair[2]]]  
## W = 1.0468e+10, p-value < 2.2e-16  
## alternative hypothesis: true location shift is not equal to 0  
##   
##   
## Wilcoxon rank-sum test results for microbusiness\_density and median\_hh\_inc :  
##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: merged\_df\_new[[pair[1]]] and merged\_df\_new[[pair[2]]]  
## W = 0, p-value < 2.2e-16  
## alternative hypothesis: true location shift is not equal to 0

The results show that for each pair of columns, the p-value is less than the significance level of 0.05, which means that we reject the null hypothesis that the median difference between the two columns is zero. Instead, we conclude that there is a statistically significant difference between the two columns. These results suggest that there is evidence to support the hypothesis that the population median of each column is different from the population median of microbusiness\_density.

# Confirmatory Data Analysis

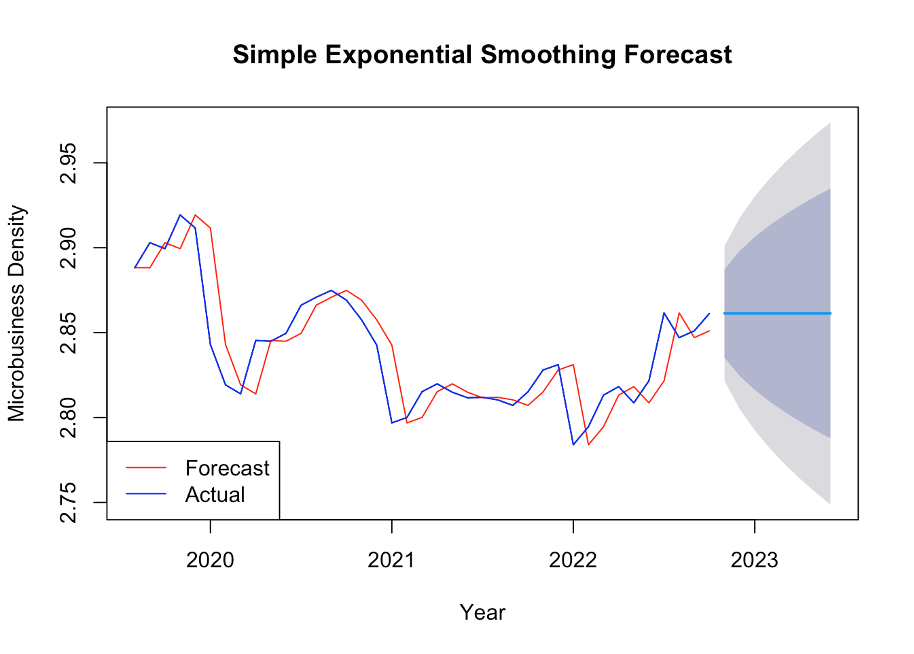
## Time Series Forecasting

Based on the feature importance scores provided by the **Random Forest** model, we can identify which variables are most strongly associated with microbusiness\_density. The variables that have the highest importance scores are likely to be the most important predictors of microbusiness\_density. In this case, the variables with the highest importance scores are: - active (116924.3551) - pct\_college (57310.5642) - pct\_foreign\_born (27024.1637) - pct\_bb (24301.8238) - cfips (23658.8595) - region\_code (17026.2925) - median\_hh\_inc (22442.3452) - pct\_it\_workers (14892.8354) Therefore, we can frame our forecasting problem as a cross-sectional data problem. To frame microbusiness\_density forecasting as a cross-sectional data problem, we will use these variables as predictors in a statistical model. We will use the **merged\_df\_clean** dataset that includes these variables as well as microbusiness\_density for a particular month and county. The goal would be to build a model that predicts the microbusiness\_density of the county for that specific month, based on the other variables in the dataset.

### Simple Exponential Smoothing Forecasting

Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight. This framework generates reliable forecasts quickly and for a wide range of time series, which is a great advantage and of major importance to applications in industry.

Exponential smoothing is a general technique for smoothing time series data by giving more weight to recent observations. The simplest of the exponentially smoothing methods is naturally called simple exponential smoothing (SES). This method is suitable for forecasting data with no clear trend or seasonal pattern. (There is a grow in the last few years, which might suggest a trend. We will consider whether a trended method would be better for this series later.)

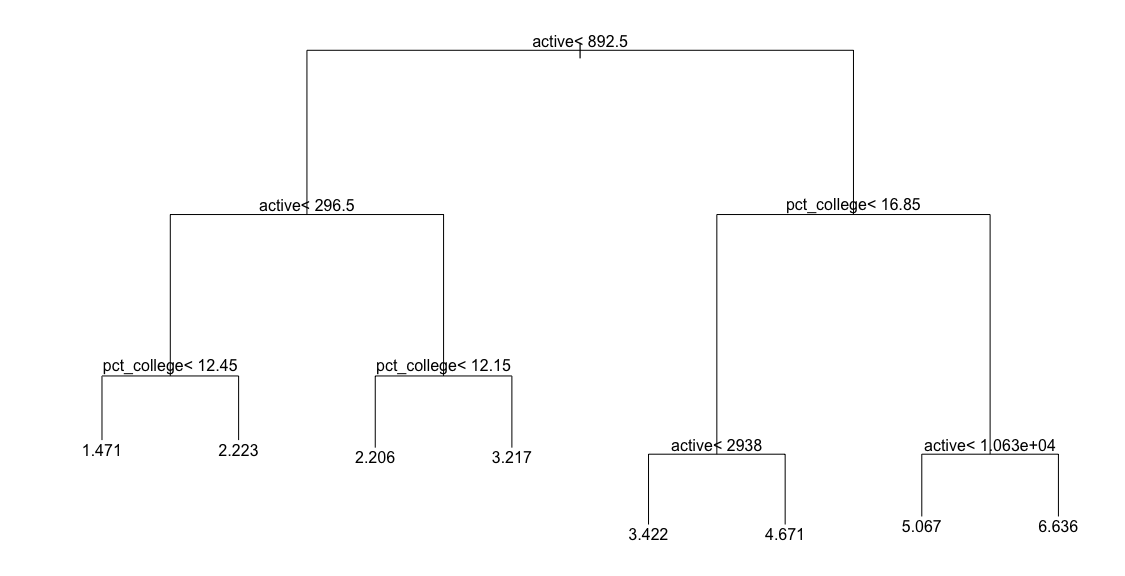


The forecasts for the period 11/2022 to 06/2023 are plotted in above. Also, plotted are one-step-ahead fitted values alongside the data over the period 08/2019 to 10/2022.

This confusion matrix shows the performance of the random forest model in predicting the microbusiness density for each category in the test set. The rows represent the predicted categories, and the columns represent the actual categories.

For example, the value in the first row and first column (24) represents the number of test data points that were predicted to be in category 0 (i.e., a microbusiness density of 0) and were actually in category 0. The value in the second row and first column (272) represents the number of test data points that were predicted to be in category 1 (i.e., a microbusiness density of 1) and were actually in category 0.

From the matrix, we can see that the model performed well for some classes (e.g., classes 0 and 9), but had more difficulty with others (e.g., classes 3, 4, and 5). We can also see that there were very few cases in some classes (e.g., class 9).



## [1] 1.192643

## n= 67969   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 67969 197474.000 2.838718   
## 2) active< 892.5 46138 59082.070 2.096465   
## 4) active< 296.5 28696 27606.270 1.769856   
## 8) pct\_college< 12.45 17287 8913.186 1.470701 \*  
## 9) pct\_college>=12.45 11409 14801.870 2.223137 \*  
## 5) active>=296.5 17442 23378.470 2.633811   
## 10) pct\_college< 12.15 10056 5855.861 2.205779 \*  
## 11) pct\_college>=12.15 7386 13171.840 3.216575 \*  
## 3) active>=892.5 21831 59251.300 4.407407   
## 6) pct\_college< 16.85 13581 27422.430 3.771278   
## 12) active< 2938.5 9786 15687.310 3.422249 \*  
## 13) active>=2938.5 3795 7468.837 4.671306 \*  
## 7) pct\_college>=16.85 8250 17286.300 5.454590   
## 14) active< 10632 6211 11538.410 5.066827 \*  
## 15) active>=10632 2039 1969.283 6.635755 \*

## 'data.frame': 97098 obs. of 18 variables:  
## $ row\_id : chr "1001\_2019-08-01" "1001\_2019-09-01" "1001\_2019-10-01" "1001\_2019-11-01" ...  
## $ cfips : int 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 ...  
## $ county : chr "autauga county" "autauga county" "autauga county" "autauga county" ...  
## $ state : chr "alabama" "alabama" "alabama" "alabama" ...  
## $ first\_day\_of\_month : Date, format: "2019-08-01" "2019-09-01" ...  
## $ microbusiness\_density: num 3.01 2.88 3.06 2.99 2.99 ...  
## $ active : int 1249 1198 1269 1243 1243 1242 1217 1227 1255 1257 ...  
## $ year\_month : chr "2019-08" "2019-09" "2019-10" "2019-11" ...  
## $ year : num 2019 2019 2019 2019 2019 ...  
## $ month : int 8 9 10 11 12 1 2 3 4 5 ...  
## $ pct\_bb : num 76.6 76.6 76.6 76.6 76.6 78.9 78.9 78.9 78.9 78.9 ...  
## $ pct\_college : num 14.5 14.5 14.5 14.5 14.5 15.9 15.9 15.9 15.9 15.9 ...  
## $ pct\_foreign\_born : num 2.1 2.1 2.1 2.1 2.1 2 2 2 2 2 ...  
## $ pct\_it\_workers : num 1.3 1.3 1.3 1.3 1.3 1.1 1.1 1.1 1.1 1.1 ...  
## $ median\_hh\_inc : num 55317 55317 55317 55317 55317 ...  
## $ region : chr "southeast" "southeast" "southeast" "southeast" ...  
## $ region\_code : int 5 5 5 5 5 5 5 5 5 5 ...  
## $ state\_code : int 1 1 1 1 1 1 1 1 1 1 ...  
## - attr(\*, "na.action")= 'omit' Named int [1:28551] 40 41 42 43 44 45 46 47 48 88 ...  
## ..- attr(\*, "names")= chr [1:28551] "40" "41" "42" "43" ...

Model: LGBM

Accuracy: 0.8215756951596292

SMAPE: 24.461656782882617

TPR: 0.9474959612277868, FPR: 0.47397449155463633

Type 1 error: 0.47397449155463633

Type 2 error: 0.05250403877221325

Confusion matrix:

[[ 3052 2750]

[ 715 12903]]

Model: XGB

Accuracy: 0.8730690010298661

SMAPE: 17.17386845964759

TPR: 0.9554266412101631, FPR: 0.32023440193036884

Type 1 error: 0.32023440193036884

Type 2 error: 0.04457335878983698

Confusion matrix:

[[ 3944 1858]

[ 607 13011]]

Model: RandomForest

Accuracy: 0.9803295571575695

SMAPE: 3.190121728977069

TPR: 0.9872962255837862, FPR: 0.03602206135815236

Type 1 error: 0.03602206135815236

Type 2 error: 0.012703774416213835

Confusion matrix:

[[ 5593 209]

[ 173 13445]]

Model: ElasticNet

Accuracy: 0.7735839340885685

SMAPE: 34.92263770110042

TPR: 0.9478631223380819, FPR: 0.6354705274043433

Type 1 error: 0.6354705274043433

Type 2 error: 0.052136877661918046

Confusion matrix:

[[ 2115 3687]

[ 710 12908]]

Model: Lasso

Accuracy: 0.7646240988671472

SMAPE: 36.560206270932646

TPR: 0.9586576589807607, FPR: 0.6907962771458118

Type 1 error: 0.6907962771458118

Type 2 error: 0.04134234101923924

Confusion matrix:

[[ 1794 4008]

[ 563 13055]]

Model: Ridge

Accuracy: 0.7804840370751802

SMAPE: 34.26222364111129

TPR: 0.9230430312821266, FPR: 0.5541192692175112

Type 1 error: 0.5541192692175112

Type 2 error: 0.0769569687178734

Confusion matrix:

[[ 2587 3215]

[ 1048 12570]]

Model: LinearRegression

Accuracy: 0.7804840370751802

SMAPE: 34.2622449744924

TPR: 0.9230430312821266, FPR: 0.5541192692175112

Type 1 error: 0.5541192692175112

Type 2 error: 0.0769569687178734

Confusion matrix:

[[ 2587 3215]

[ 1048 12570]]

Partition 0

Model: LGBM

Accuracy: 0.8438102583879676

SMAPE: 20.306180301144938

TPR: 0.8440860215053764, FPR: 0.15639810426540285

Type 1 error: 0.15639810426540285

Type 2 error: 0.15591397849462366

Confusion matrix:

[[3738 693]

[ 522 2826]]

Model: XGB

Accuracy: 0.9096284869520503

SMAPE: 11.75940928233866

TPR: 0.9124850657108722, FPR: 0.09252990295644324

Type 1 error: 0.09252990295644324

Type 2 error: 0.08751493428912784

Confusion matrix:

[[4021 410]

[ 293 3055]]

Model: RandomForest

Accuracy: 0.9727471397351845

SMAPE: 3.6139251529715

TPR: 0.9716248506571087, FPR: 0.026404874746106973

Type 1 error: 0.026404874746106973

Type 2 error: 0.028375149342891277

Confusion matrix:

[[4314 117]

[ 95 3253]]

Model: ElasticNet

Accuracy: 0.6400565625401723

SMAPE: 34.618915319382225

TPR: 0.6833930704898447, FPR: 0.3926878808395396

Type 1 error: 0.3926878808395396

Type 2 error: 0.31660692951015534

Confusion matrix:

[[2691 1740]

[1060 2288]]

Model: Lasso

Accuracy: 0.6342717572952822

SMAPE: 34.93806684270428

TPR: 0.6666666666666666, FPR: 0.3902053712480253

Type 1 error: 0.3902053712480253

Type 2 error: 0.3333333333333333

Confusion matrix:

[[2702 1729]

[1116 2232]]

Model: Ridge

Accuracy: 0.6856922483609719

SMAPE: 32.32389894154523

TPR: 0.7461170848267622, FPR: 0.35996389076957797

Type 1 error: 0.35996389076957797

Type 2 error: 0.25388291517323774

Confusion matrix:

[[2836 1595]

[ 850 2498]]

Model: LinearRegression

Accuracy: 0.6856922483609719

SMAPE: 32.32390011004467

TPR: 0.7461170848267622, FPR: 0.35996389076957797

Type 1 error: 0.35996389076957797

Type 2 error: 0.25388291517323774

Confusion matrix:

[[2836 1595]

[ 850 2498]]

Partition 1

Model: LGBM

Accuracy: 0.8471440064360418

SMAPE: 17.410390770115402

TPR: 0.9808102345415778, FPR: 0.5639344262295082

Type 1 error: 0.5639344262295082

Type 2 error: 0.019189765458422176

Confusion matrix:

[[ 532 688]

[ 72 3680]]

Model: XGB

Accuracy: 0.9432823813354787

SMAPE: 8.00433181855996

TPR: 0.9848081023454158, FPR: 0.18442622950819673

Type 1 error: 0.18442622950819673

Type 2 error: 0.015191897654584221

Confusion matrix:

[[ 995 225]

[ 57 3695]]

Model: RandomForest

Accuracy: 0.9802896218825422

SMAPE: 3.2539371063431486

TPR: 0.9898720682302772, FPR: 0.04918032786885246

Type 1 error: 0.04918032786885246

Type 2 error: 0.010127931769722815

Confusion matrix:

[[1160 60]

[ 38 3714]]

Model: ElasticNet

Accuracy: 0.754827031375704

SMAPE: 34.43970276681718

TPR: 1.0, FPR: 0.9991803278688525

Type 1 error: 0.9991803278688525

Type 2 error: 0.0

Confusion matrix:

[[ 1 1219]

[ 0 3752]]

Model: Lasso

Accuracy: 0.7546259050683829

SMAPE: 35.941535985964144

TPR: 1.0, FPR: 1.0

Type 1 error: 1.0

Type 2 error: 0.0

Confusion matrix:

[[ 0 1220]

[ 0 3752]]

Model: Ridge

Accuracy: 0.7670957361222848

SMAPE: 32.46163635656927

TPR: 0.990405117270789, FPR: 0.919672131147541

Type 1 error: 0.919672131147541

Type 2 error: 0.009594882729211088

Confusion matrix:

[[ 98 1122]

[ 36 3716]]

Model: LinearRegression

Accuracy: 0.7670957361222848

SMAPE: 32.46160399677549

TPR: 0.990405117270789, FPR: 0.919672131147541

Type 1 error: 0.919672131147541

Type 2 error: 0.009594882729211088

Confusion matrix:

[[ 98 1122]

[ 36 3716]]

Partition 2

Model: LGBM

Accuracy: 0.8883732291157792

SMAPE: 16.34367987243351

TPR: 0.9927388905024688, FPR: 0.663594470046083

Type 1 error: 0.663594470046083

Type 2 error: 0.007261109497531223

Confusion matrix:

[[ 219 432]

[ 25 3418]]

Model: XGB

Accuracy: 0.9474841231069858

SMAPE: 7.232079063779462

TPR: 0.9875108916642463, FPR: 0.2642089093701997

Type 1 error: 0.2642089093701997

Type 2 error: 0.012489108335753703

Confusion matrix:

[[ 479 172]

[ 43 3400]]

Model: RandomForest

Accuracy: 0.9814362481680508

SMAPE: 3.170988806667623

TPR: 0.9889631135637525, FPR: 0.05837173579109063

Type 1 error: 0.05837173579109063

Type 2 error: 0.01103688643624746

Confusion matrix:

[[ 613 38]

[ 38 3405]]

Model: ElasticNet

Accuracy: 0.8409868099658037

SMAPE: 37.766631558105786

TPR: 1.0, FPR: 1.0

Type 1 error: 1.0

Type 2 error: 0.0

Confusion matrix:

[[ 0 651]

[ 0 3443]]

Model: Lasso

Accuracy: 0.8409868099658037

SMAPE: 38.7988942819885

TPR: 1.0, FPR: 1.0

Type 1 error: 1.0

Type 2 error: 0.0

Confusion matrix:

[[ 0 651]

[ 0 3443]]

Model: Ridge

Accuracy: 0.8431851489985345

SMAPE: 34.60586265332106

TPR: 0.9968051118210862, FPR: 0.9692780337941628

Type 1 error: 0.9692780337941628

Type 2 error: 0.003194888178913738

Confusion matrix:

[[ 20 631]

[ 11 3432]]

Model: LinearRegression

Accuracy: 0.8431851489985345

SMAPE: 34.60579777243087

TPR: 0.9968051118210862, FPR: 0.9692780337941628

Type 1 error: 0.9692780337941628

Type 2 error: 0.003194888178913738

Confusion matrix:

[[ 20 631]

[ 11 3432]]

Partition 3

Model: LGBM

Accuracy: 0.9755529685681025

SMAPE: 8.18848746483556

TPR: 0.996328029375765, FPR: 0.42857142857142855

Type 1 error: 0.42857142857142855

Type 2 error: 0.0036719706242350062

Confusion matrix:

[[ 72 54]

[ 9 2442]]

Model: XGB

Accuracy: 0.9926270857586341

SMAPE: 3.0235519383373974

TPR: 0.9983680130558955, FPR: 0.11904761904761904

Type 1 error: 0.11904761904761904

Type 2 error: 0.0016319869441044472

Confusion matrix:

[[ 111 15]

[ 4 2447]]

Model: RandomForest

Accuracy: 0.996895615056267

SMAPE: 2.0111335728330193

TPR: 0.9983680130558955, FPR: 0.031746031746031744

Type 1 error: 0.031746031746031744

Type 2 error: 0.0016319869441044472

Confusion matrix:

[[ 122 4]

[ 4 2447]]

Model: ElasticNet

Accuracy: 0.9511059371362048

SMAPE: 30.271709216136426

TPR: 1.0, FPR: 1.0

Type 1 error: 1.0

Type 2 error: 0.0

Confusion matrix:

[[ 0 126]

[ 0 2451]]

Model: Lasso

Accuracy: 0.9511059371362048

SMAPE: 30.732884684514328

TPR: 1.0, FPR: 1.0

Type 1 error: 1.0

Type 2 error: 0.0

Confusion matrix:

[[ 0 126]

[ 0 2451]]

Model: Ridge

Accuracy: 0.9491656965463717

SMAPE: 27.36115729283564

TPR: 0.9979600163198694, FPR: 1.0

Type 1 error: 1.0

Type 2 error: 0.002039983680130559

Confusion matrix:

[[ 0 126]

[ 5 2446]]

Model: LinearRegression

Accuracy: 0.9491656965463717

SMAPE: 27.360957058888026

TPR: 0.9979600163198694, FPR: 1.0

Type 1 error: 1.0

Type 2 error: 0.002039983680130559

Confusion matrix:

[[ 0 126]

[ 5 2446]]

1. <https://doi.org/10.1371/journal.pone.0256407.g001> [↑](#footnote-ref-1)