GoDaddy - Microbusiness Density Forecasting

Mohammad Solki

2023-02-28

1 Introduction

1.1 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:

$$\text{SMAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|F_t| + |A_t|)/2}$$

where:

- n is the number of observations in the time series
- F_t is the forecasted value at time t
- A_t is the actual value at time t
- $|x|$ denotes the absolute value of x.

1.2 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. 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).

1.3 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)
```

2 Exploratory Data Analysis (EDA)

2.1 Data Preprocessing & Cleaning

2.1.1 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)
```

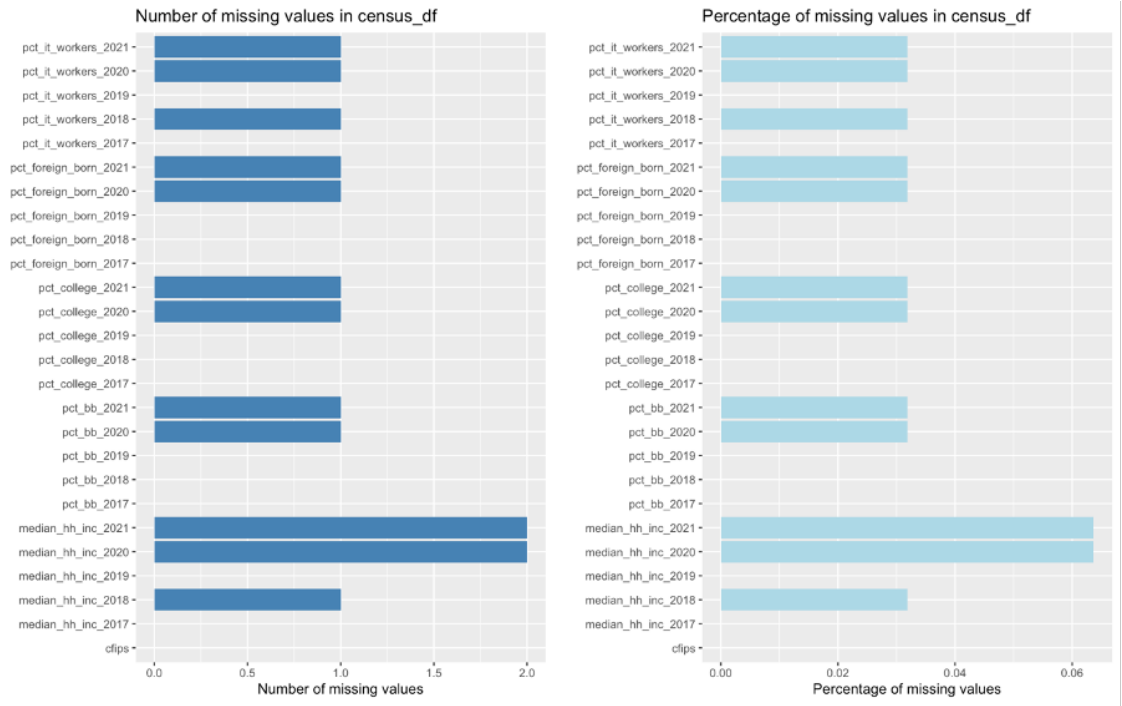
```
##  Date[1:122265], format: "2019-08-01" "2019-09-01" "2019-10-01" "2019-11-
01" "2019-12-01" ...
```

2.1.2 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
```

2.1.3 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
```

2.1.4 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" ...
```

2.1.5 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.

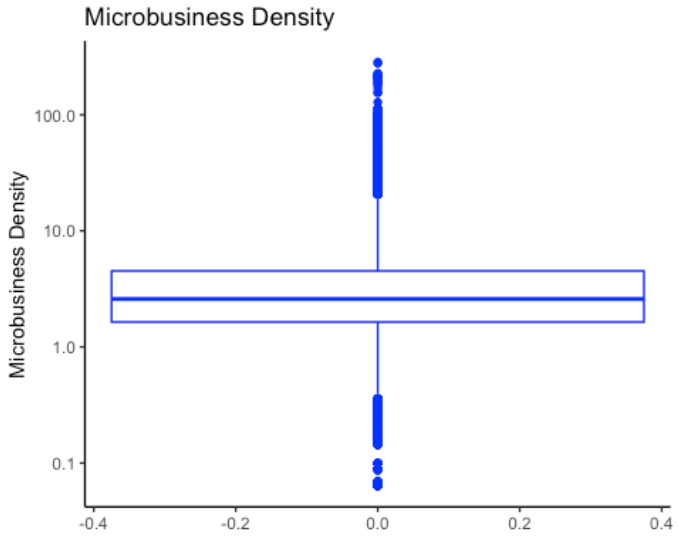
```
# remove NA values created in merged_df
merged_df <- merged_df %>%
  na.omit(merged_df)

merged_test <- merged_test %>%
  na.omit(merged_test)
```

2.2 Descriptive Statistics & Multidimensional Data Analysis

2.2.1 Data Visualization

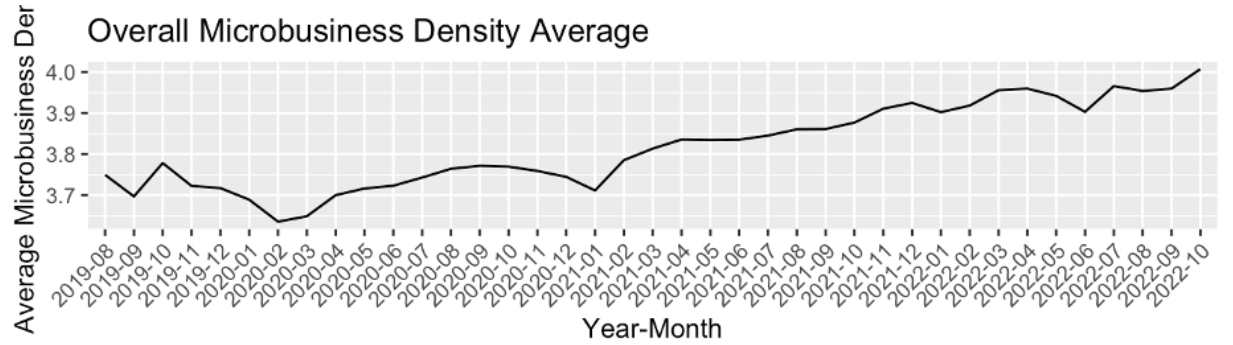
The main feature in this project is `microbusiness_density` provided in the **merged_df.** Also, the number of active microbusinesses is provided in the `active` column.

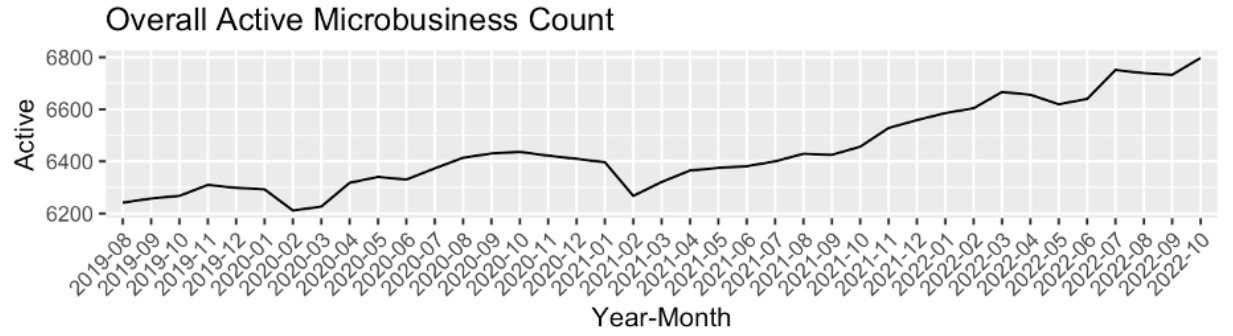
Boxplots are a visualization tool that provide insights into the central tendency and spread of a dataset, as well as identify outliers and skewness. They are useful for detecting anomalies and comparing variable distributions in a dataset, providing valuable insights into data distribution for exploratory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

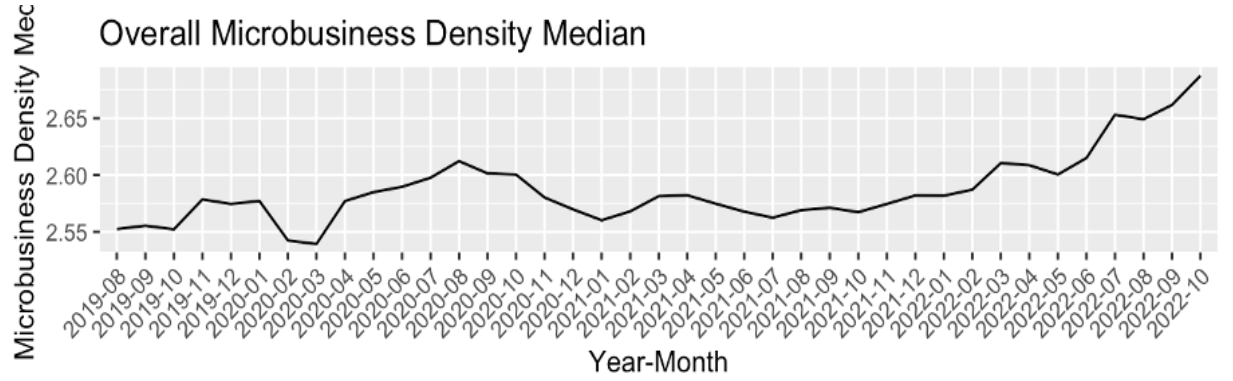


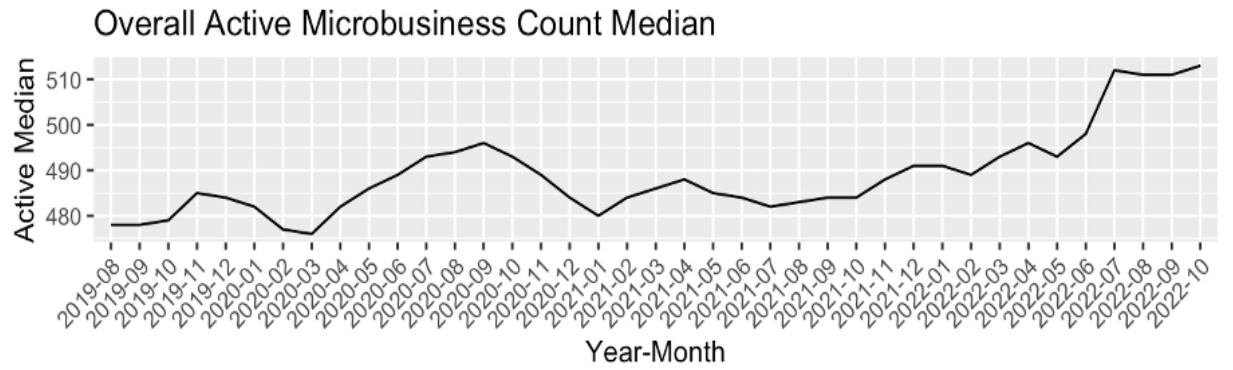
2.2.1.1 Microbusiness Density and Count

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



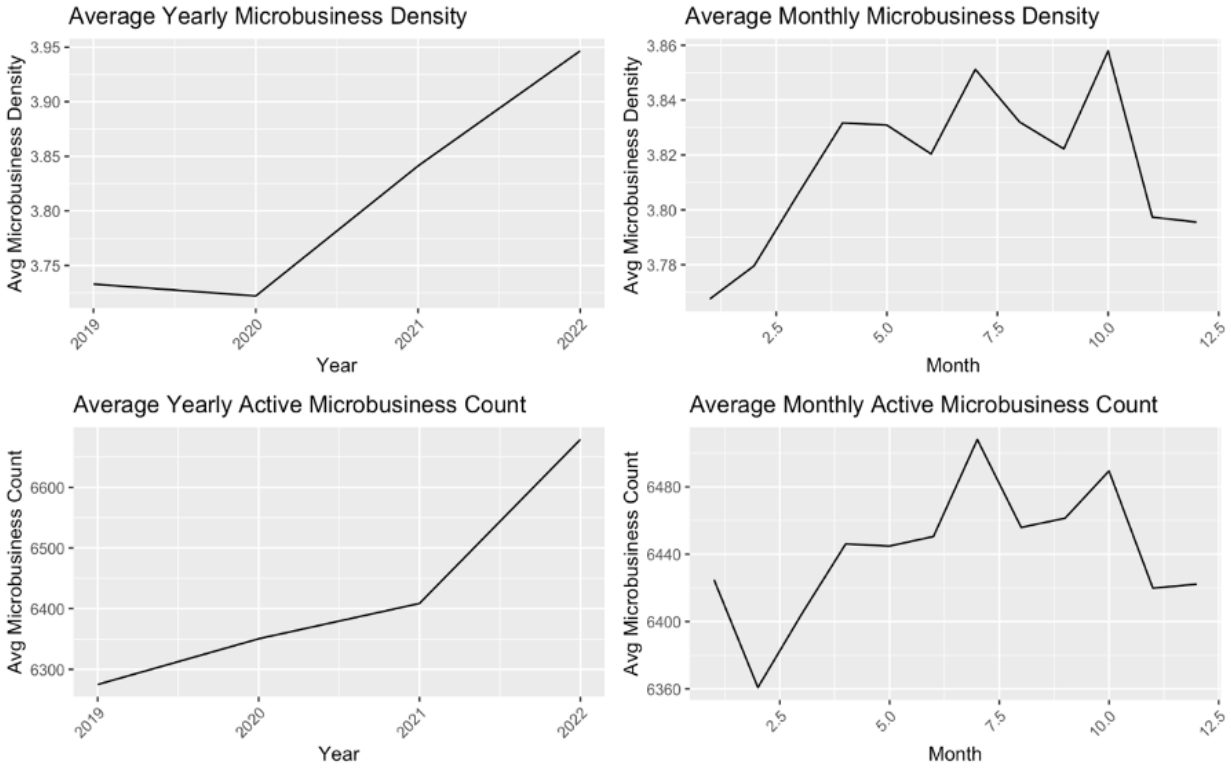






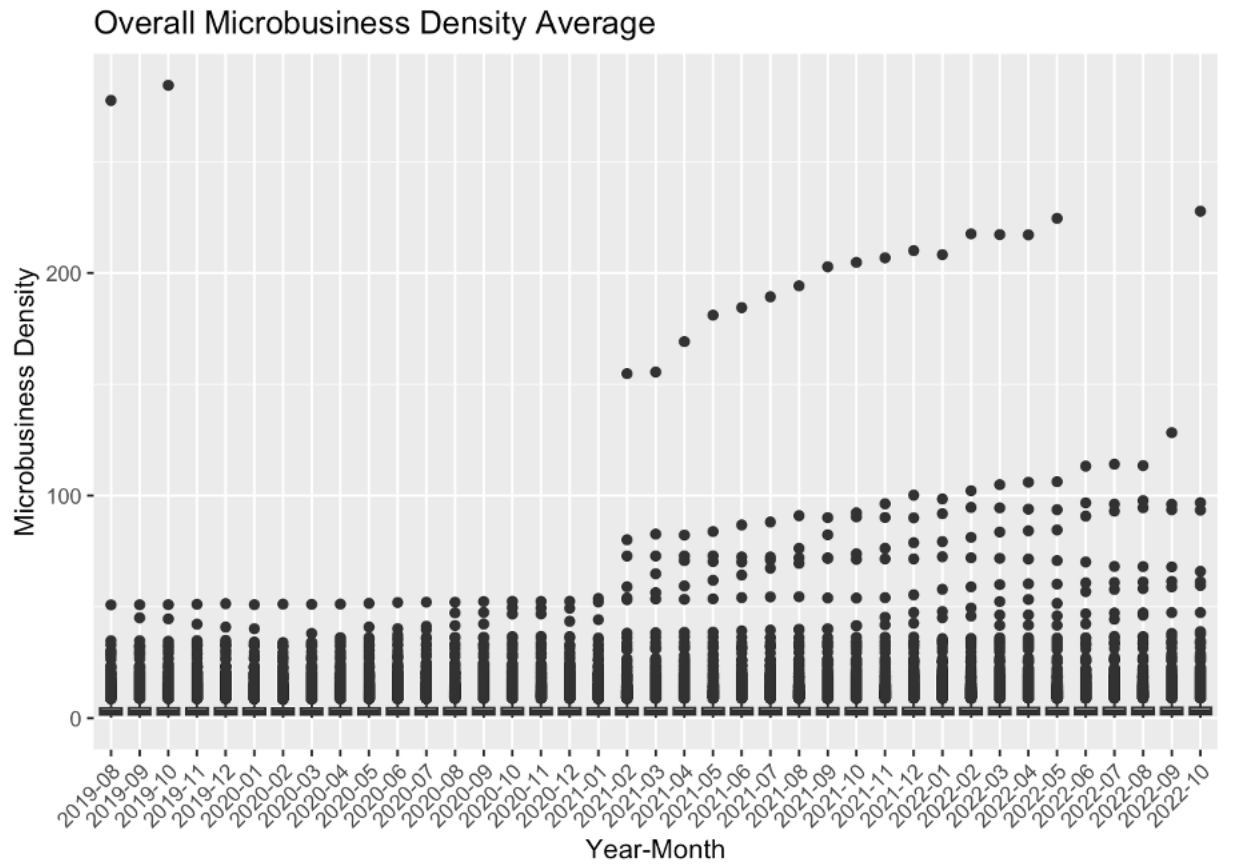
As expected, these two graphs show almost similar behavior. If we ignore the slight fluctuations of the two graphs, the general microbusiness density and count are growing over the whole time frame.

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

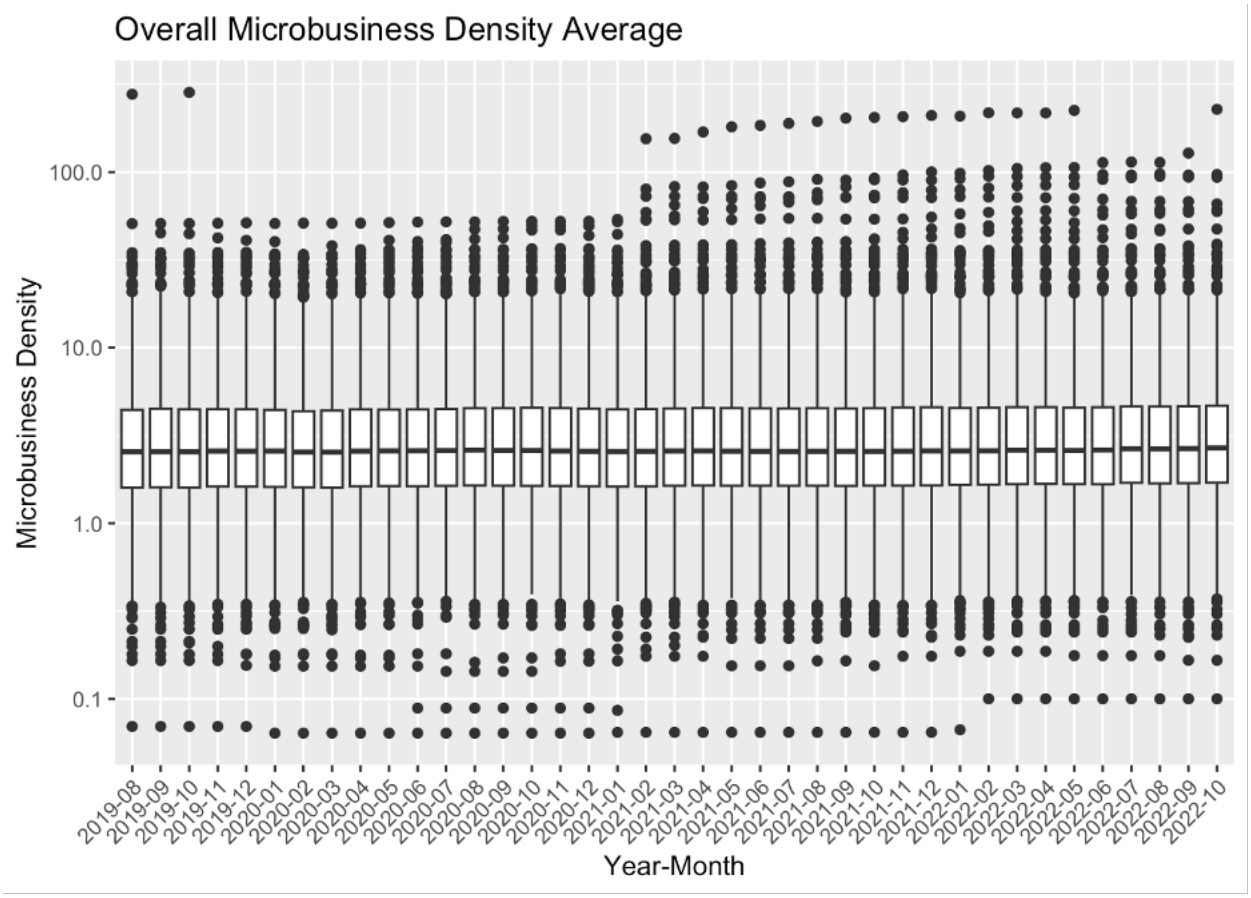


The left plots show that the average microbusiness density has increased slightly over the years, starting at approximately 3.73 in 2019 and reaching 3.94 in 2022. On the other hand, the average active count has also increased, starting at approximately 6274 in 2019 and reaching 6679 in 2022. In comparison, 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.

2.2.1.2 Economic Regional Divisions

The Bureau of Economic Analysis (BEA) divides the United States into eight distinct economic regions[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.

[1] https://doi.org/10.1371/journal.pone.0256407.g001

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

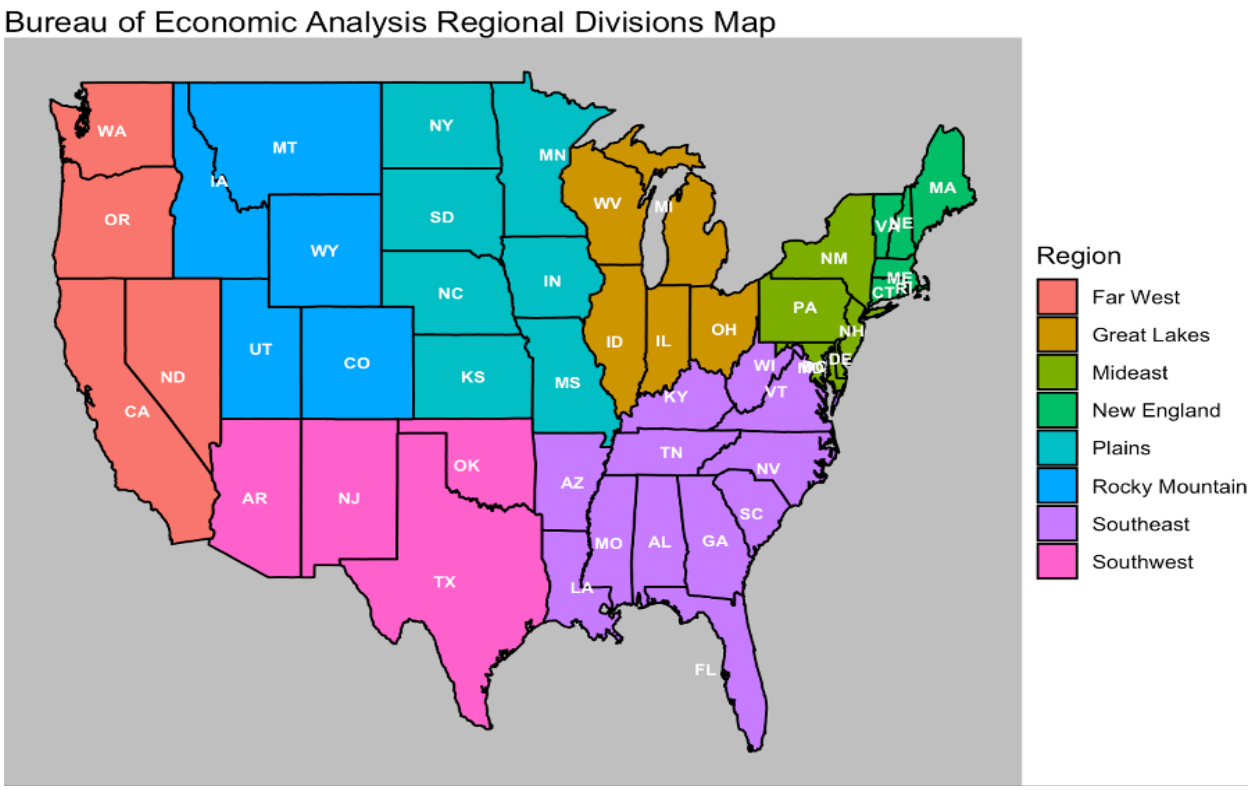
2. **Mideast**: Delaware, Maryland, New Jersey, New York, Pennsylvania, and the District of Columbia.
   The region has a diverse economy, with a mix of manufacturing, finance, healthcare, and professional services.
3. **Great Lakes**: Illinois, Indiana, Michigan, Ohio, and Wisconsin.
   The region has a strong manufacturing base, particularly in the automotive industry, and also has a significant healthcare sector.
4. **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.
5. **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.
6. **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.
7. **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.
8. **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.

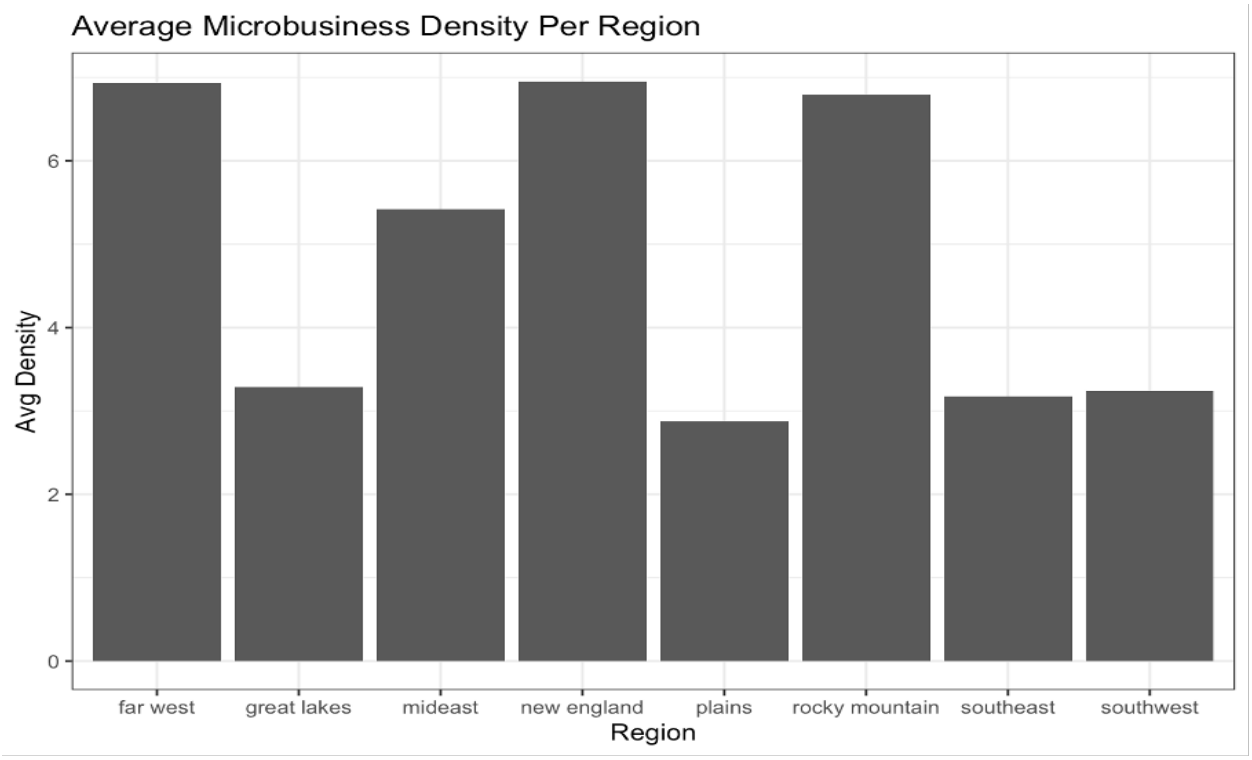
To draw the map for the BEA regions, first, we need to convert state and county columns in **merged_df** to lowercase letters. Merging two dataframes will cause problems because the data from **`map_data()`** will be in lowercase letters.

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"
```

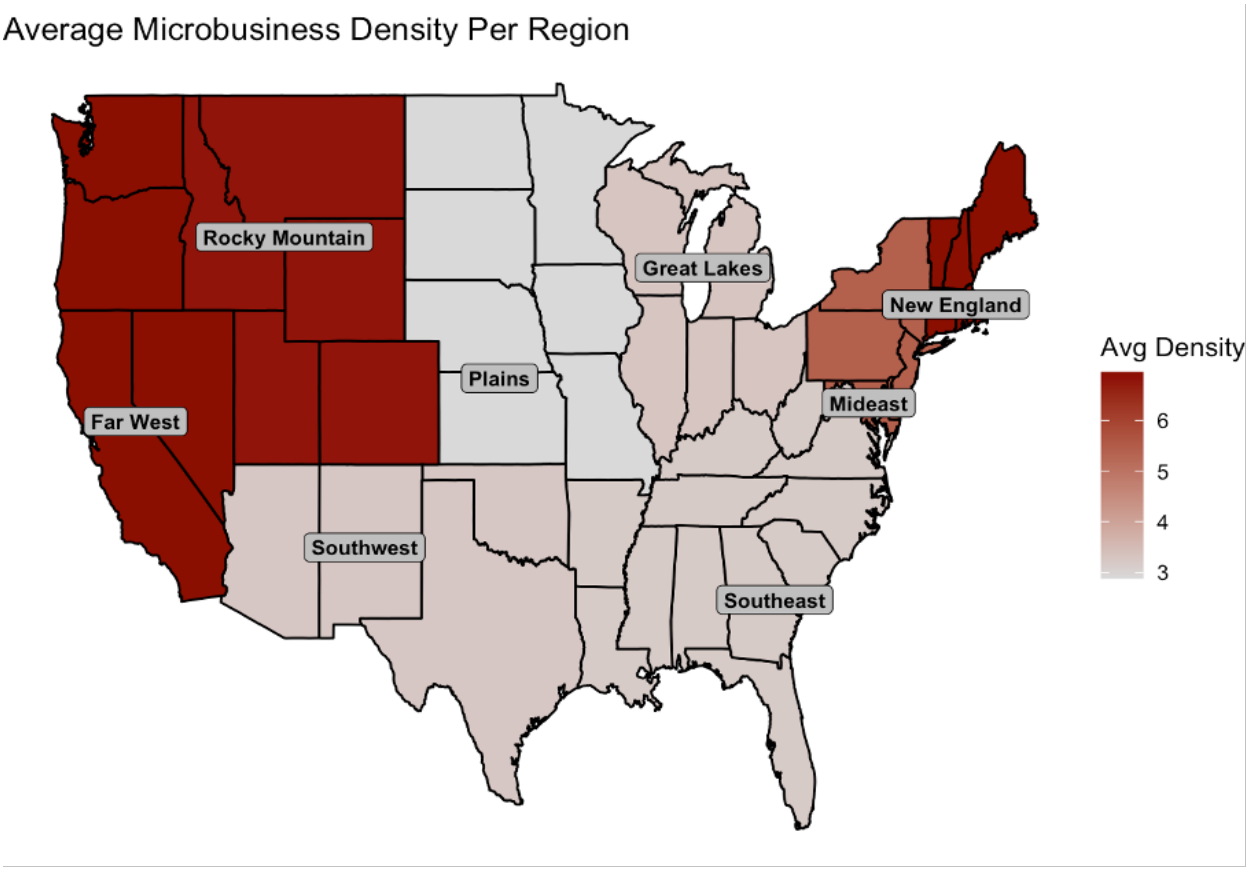
Now that the data in the dataframe matches the **`map_data()`** output, we appoint each state to the region it belongs to and then use **`ggplot()`** to draw the map:



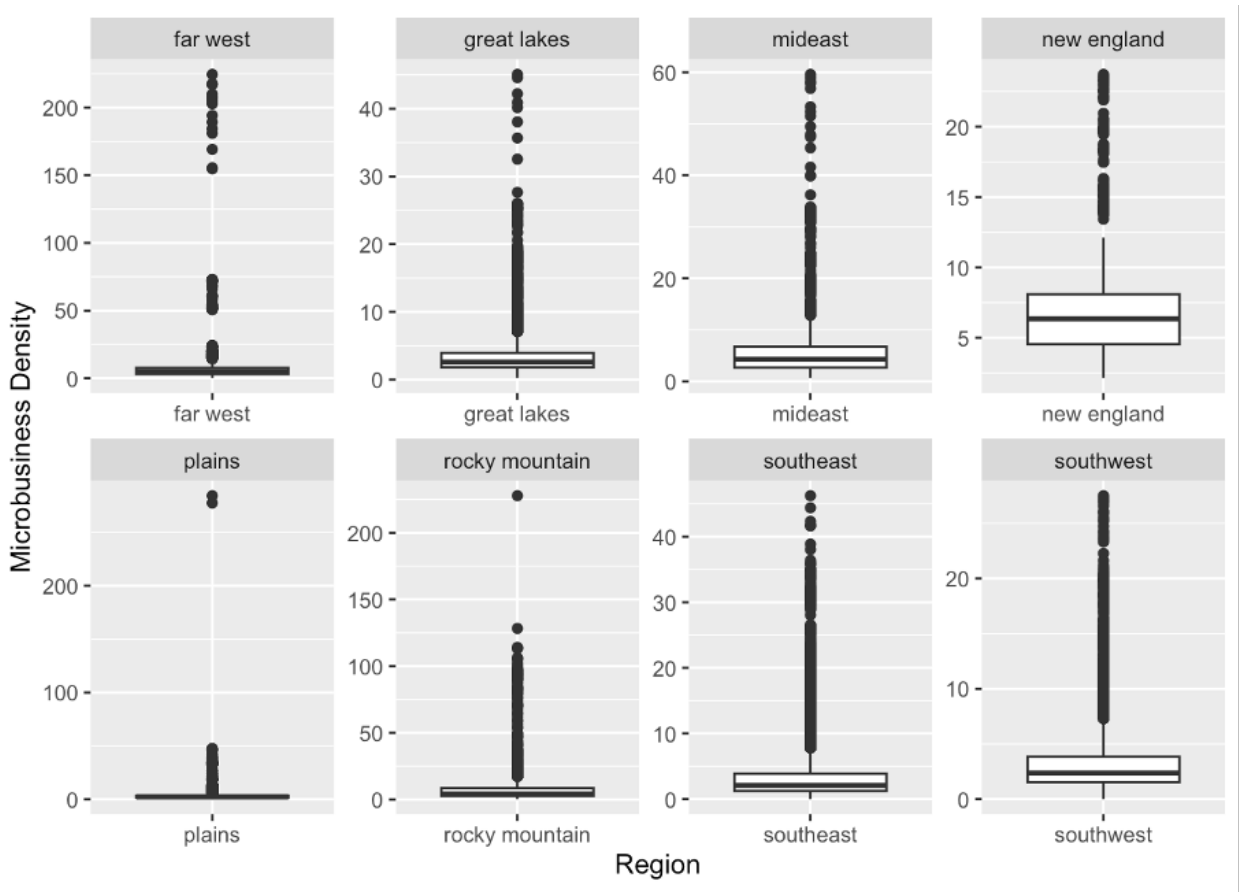


According to the above plot *New England* has the highest average microbusiness density, followed by *Farwest* and *Rocky Mountain* respectively, with a tiny difference, valuing more than 6.75. In contrast, *plains* has the lowest average microbusiness density, followed by

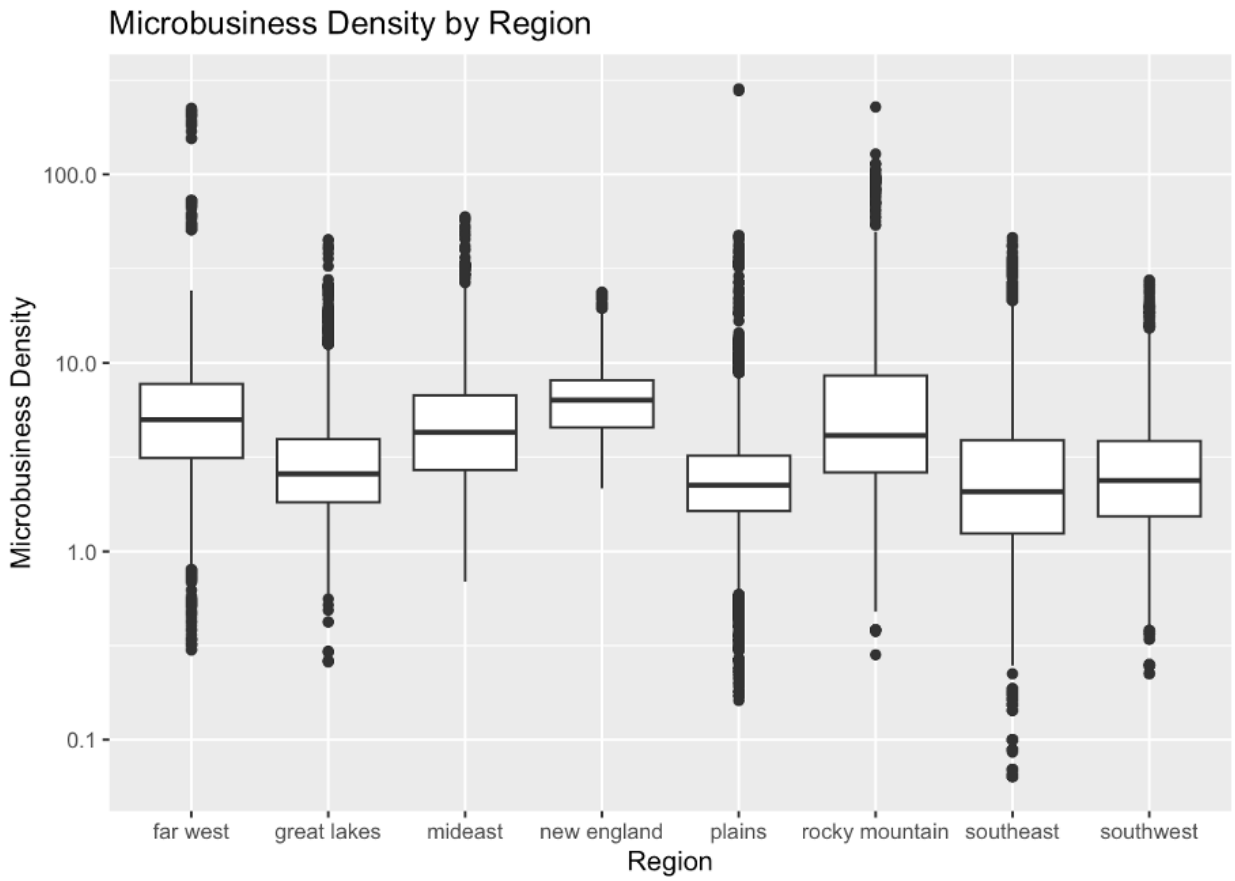
Southeast and *Southwest*, all valued under 3.25. We can use a choropleth map to get a better view on the above information. A choropleth map provides an easy way to visualize how a variable varies across a geographic area or show the level of variability within one region or multiple regions.



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.

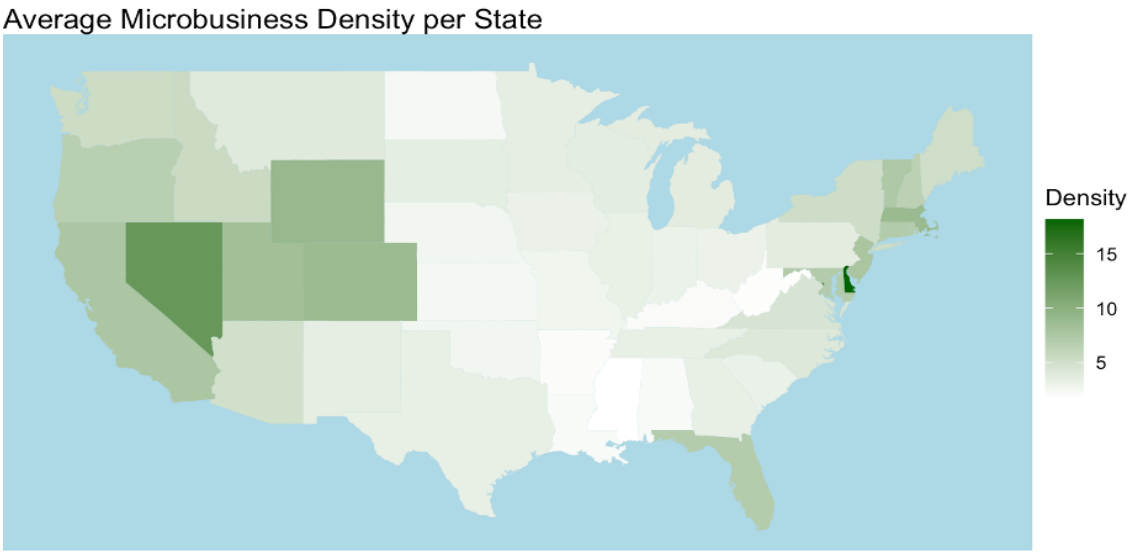


These boxplots are more informative, because using a logarithmic scale on the Y-axis helps to better reveal the differences and similarities between regions that helps to highlight any potential patterns or trends in the data. Some of the points that can be inferred from this boxplot include:

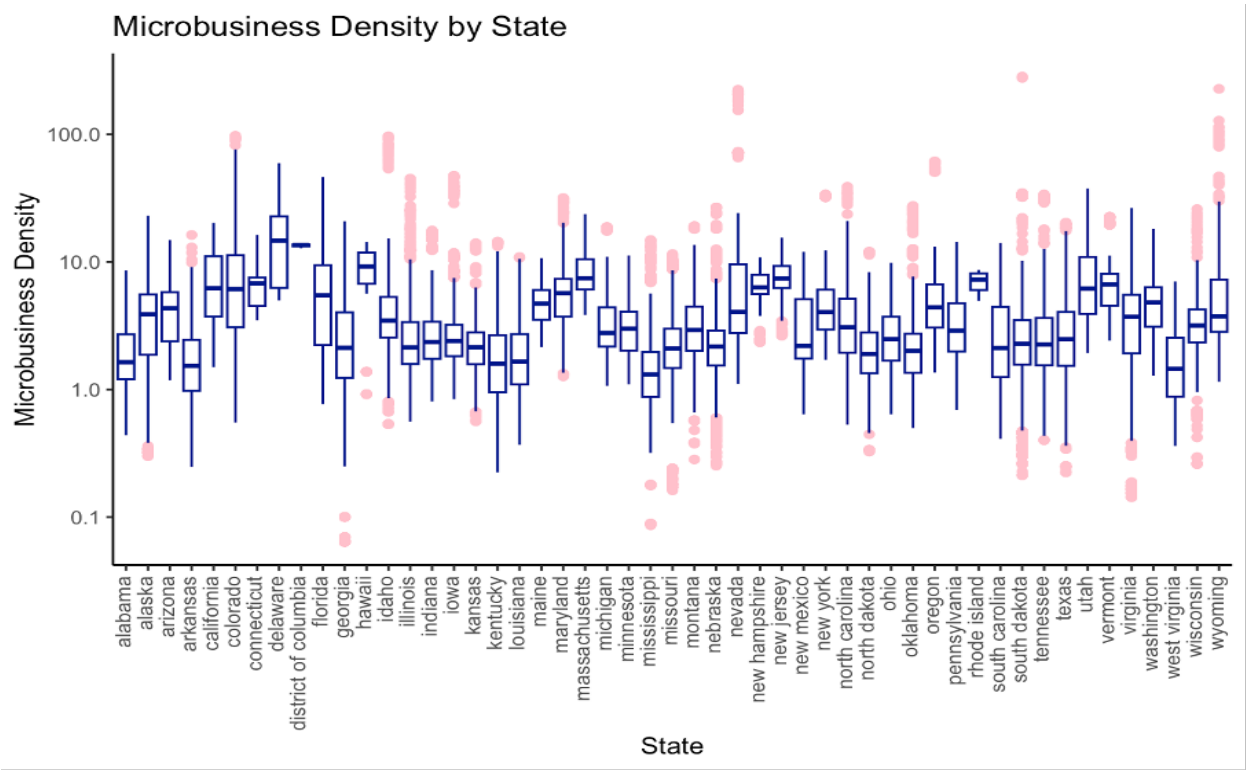
- The median microbusiness density is highest in the *New England* region, followed by the *Far West* and the *Mideast* regions.
- The 3rd quartile microbusiness density is highest in the *Rocky Mountain* region, followed by the *New England* and the *Far West* regions indicating that a significant proportion of the microbusiness density in the Rocky Mountain region is higher compared to other 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.

2.2.1.3 U.S. State Divisions

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



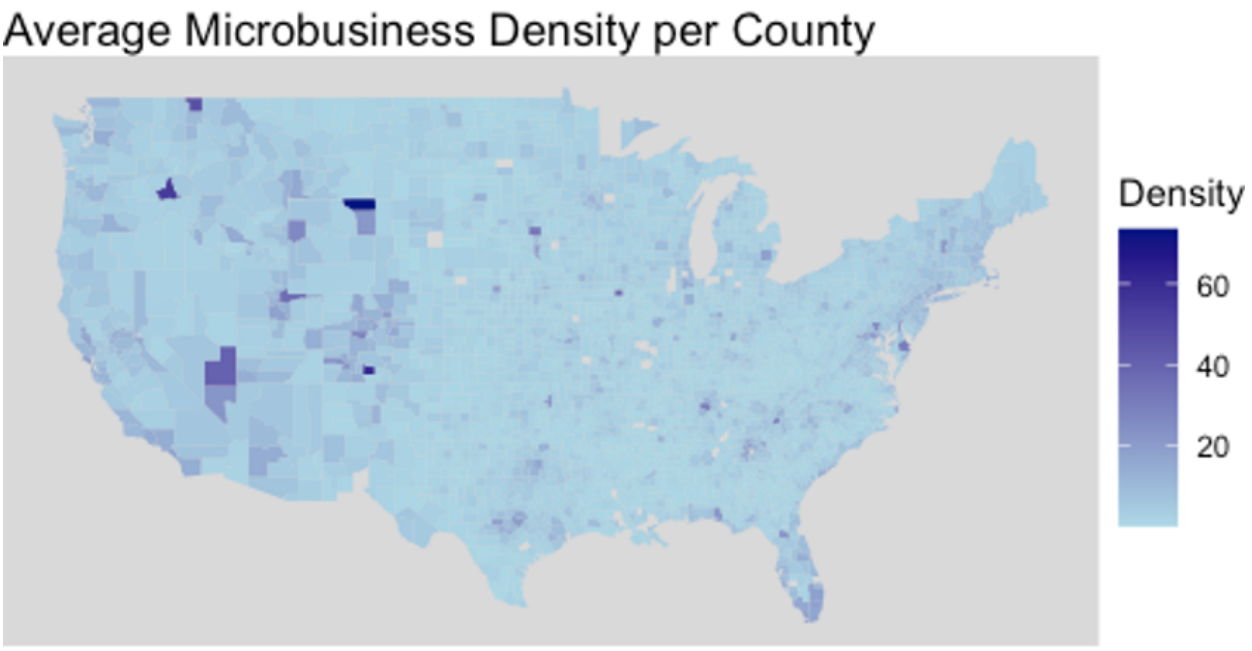
Although, this map provides an easy way to visualize how average microbusiness density varies across U.S. states but, as mentioned earlier it only demonstrates one parameter (average microbusiness density). Therefore, we can visualize average microbusiness density per each U.S. state using boxplots to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

2.2.1.4 U.S. County Divisions

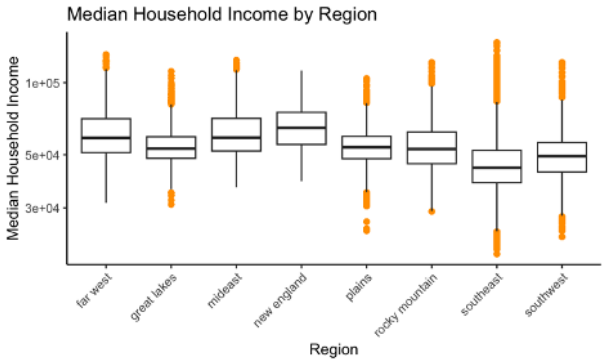
We can continue our analysis on the county level. There are around 3000 counties in the U.S., therefore, a choropleth map will prove to be useful since we cannot fit 3000 line or boxplots in a single figure.

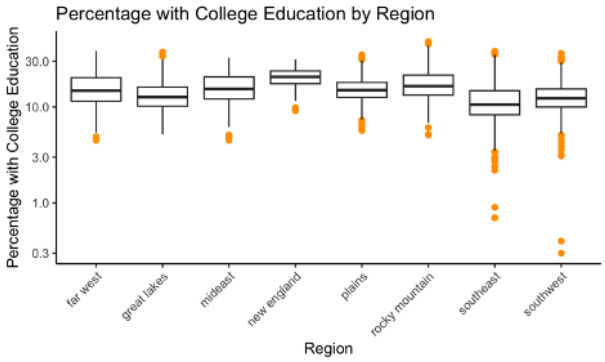


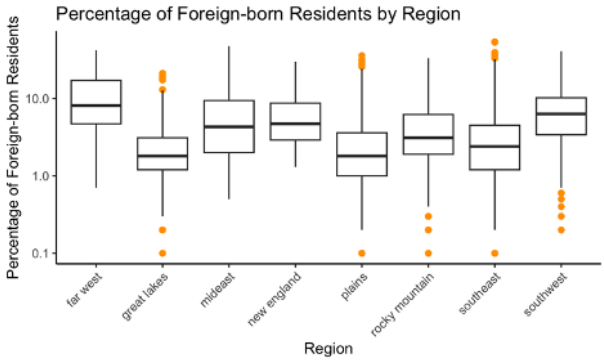
The distribution of the microbusiness density throughout the U.S. on county level is almost monolith with a few exceptions. It is necessary to analyze the data on county level to acquire cleaner data since there might be outliers on the county level data.

2.2.1.5 Census Data Visualizations

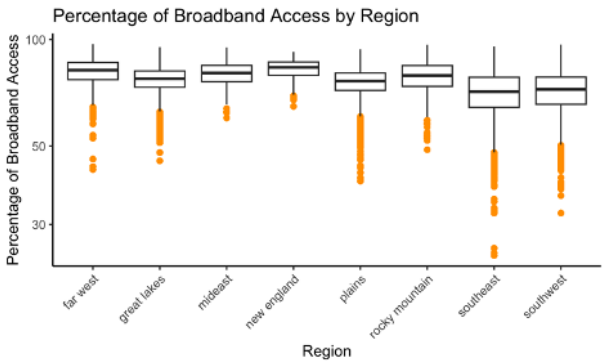
Since the data from the census dataset and train dataset are merged, we can analyze our census data on the region, state, and county levels. In the upcoming pages you can see the boxplots and choropleth maps of different census variables including median household income, percentage of people with college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

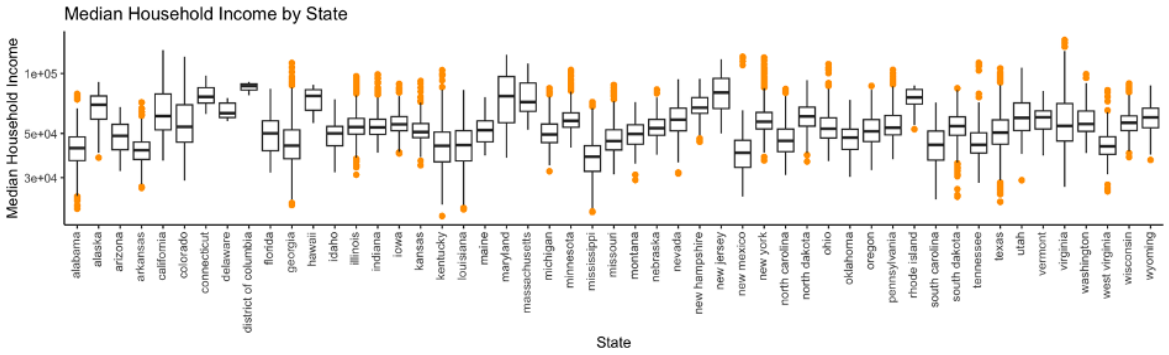

Median Household Income by Region
Median Household Income
1e+05
5e+04
3e+04
far west
great lakes
mideast
new england
plains
rocky mountain
southeast
southwest
Region

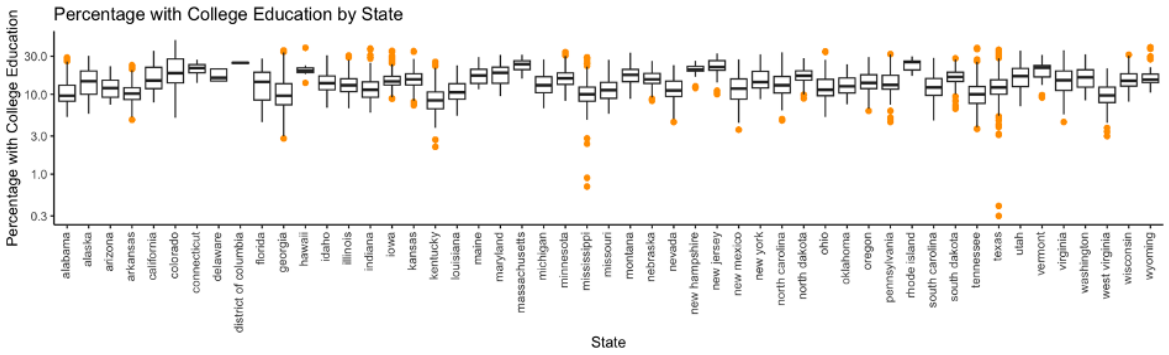

Percentage with College Education by Region
Percentage with College Education
30.0
10.0
3.0
1.0
0.3
far west
great lakes
mideast
new england
plains
rocky mountain
southeast
southwest
Region

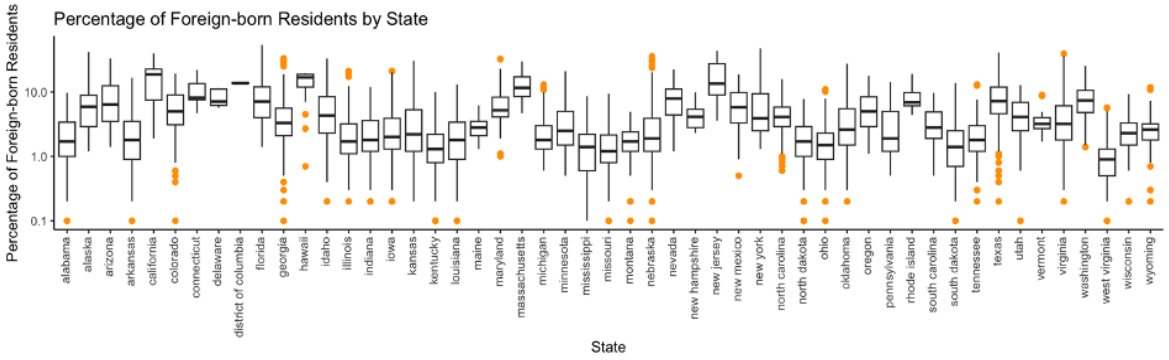

Percentage of Foreign-born Residents by Region
Percentage of Foreign-born Residents
10.0
1.0
0.1
far west
great lakes
mideast
new england
plains
rocky mountain
southeast
southwest
Region

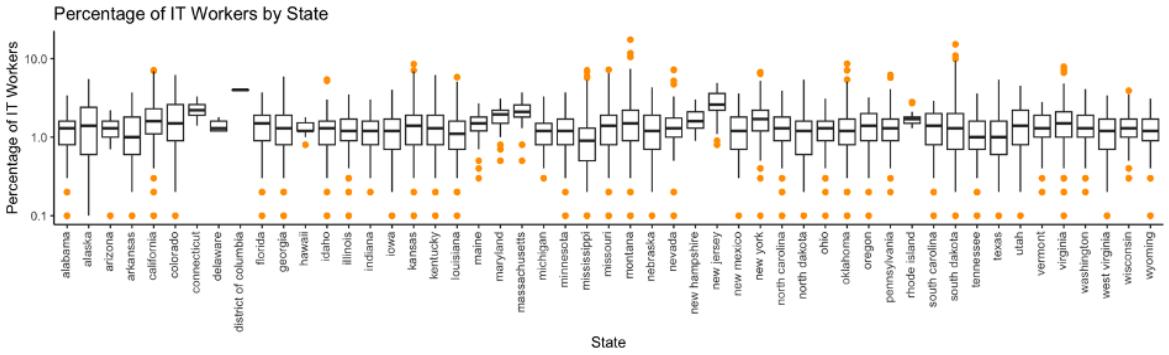

Percentage of IT Workers by Region
Percentage of IT Workers
10.0
1.0
0.1
far west
great lakes
mideast
new england
plains
rocky mountain
southeast
southwest
Region

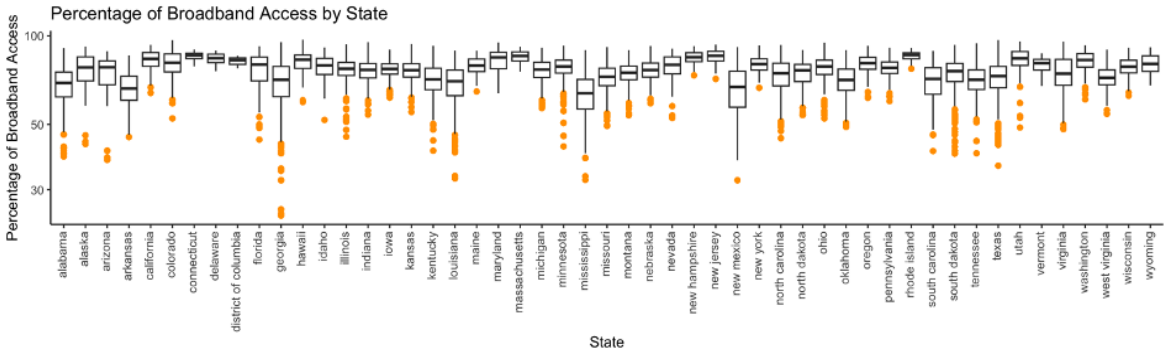

Percentage of Broadband Access by Region
Percentage of Broadband Access
100
50
30
far west
great lakes
mideast
new england
plains
rocky mountain
southeast
southwest
Region


Median Household Income by State
Median Household Income
1e+05
5e+04
3e+04
alabama
alaska
arizona
arkansas
california
colorado
connecticut
delaware
district of columbia
florida
georgia
hawaii
idaho
illinois
indiana
iowa
kansas
kentucky
louisiana
maine
maryland
massachusetts
michigan
minnesota
mississippi
missouri
montana
nebraska
nevada
new hampshire
new jersey
new mexico
new york
north carolina
north dakota
ohio
oklahoma
oregon
pennsylvania
rhode island
south carolina
south dakota
tennessee
texas
utah
vermont
virginia
washington
west virginia
wisconsin
wyoming
State

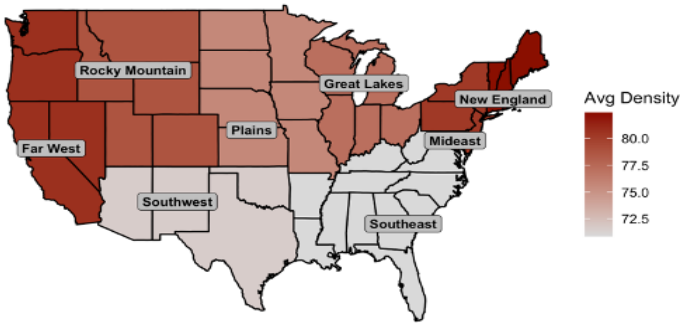

Percentage with College Education by State
Percentage with College Education
30.0
10.0
3.0
1.0
0.3
State


Percentage of Foreign-born Residents by State
Percentage of Foreign-born Residents
10.0
1.0
0.1
State

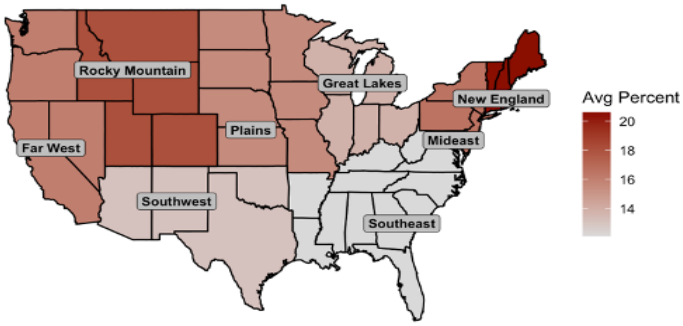

Percentage of IT Workers by State
Percentage of IT Workers
10.0
1.0
0.1
State


Percentage of Broadband Access by State
Percentage of Broadband Access
100
50
30
State

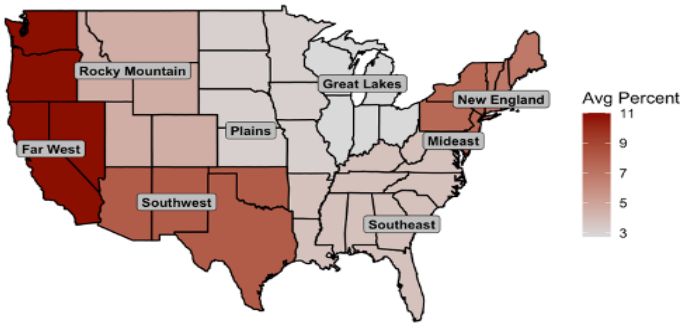
Average Percent of Broadband Access Per Region



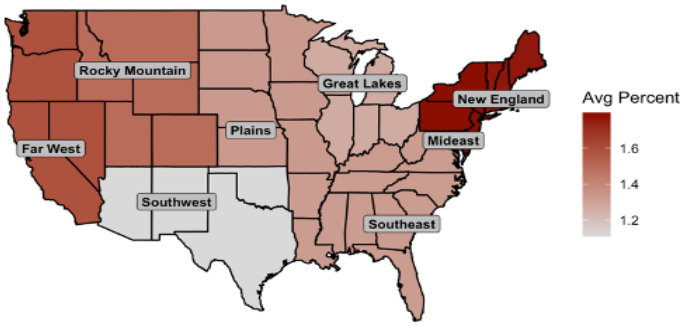
Average Percent of College Graduates Per Region



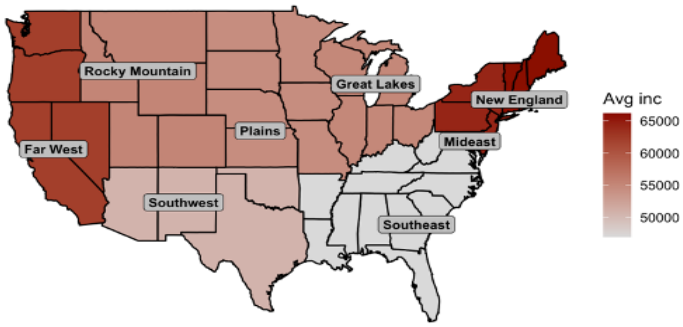
Average Percent of Foreign-Born Population Per Region



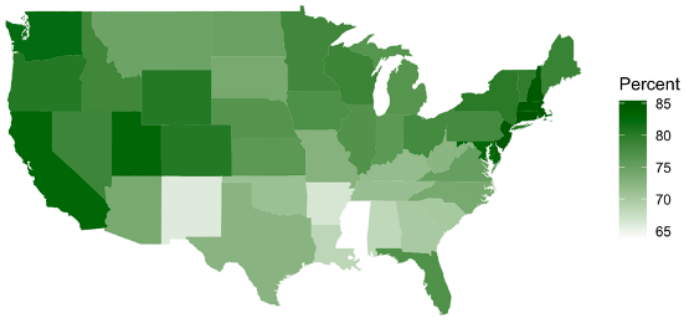
Average Percent of IT Workers Per Region



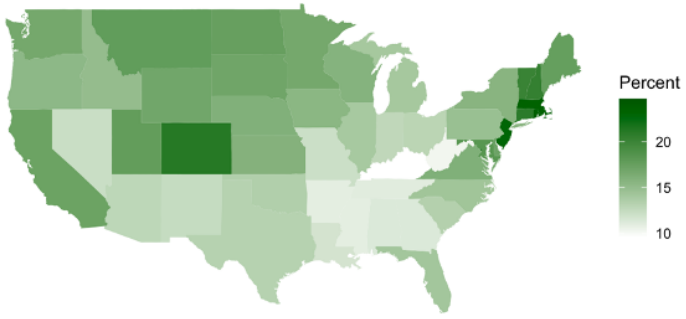
Average Median Household inc Per Region



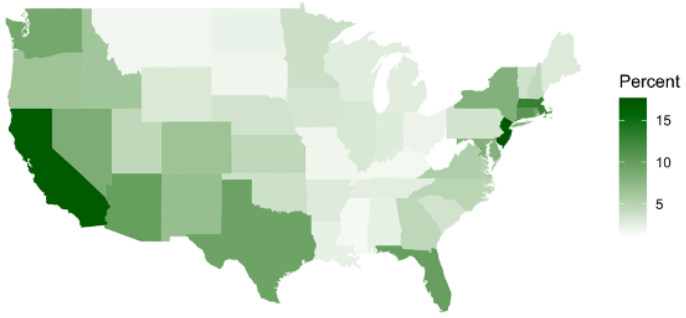
Percent of Broadband Access per State



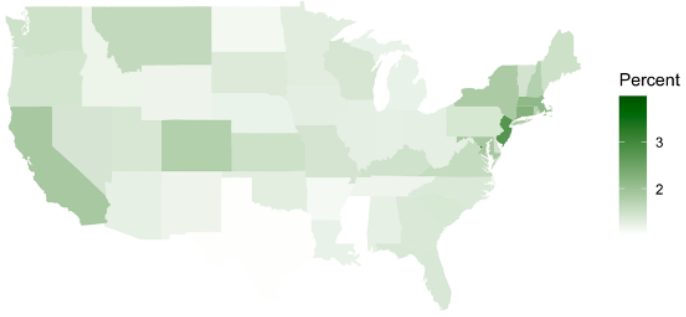
Percent of Population with College Education per State



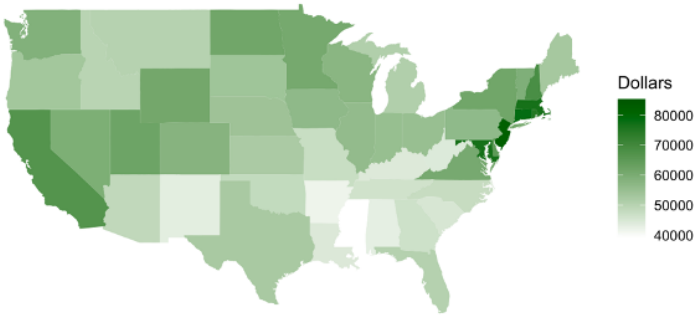
Percent of Foreign-born Population per State



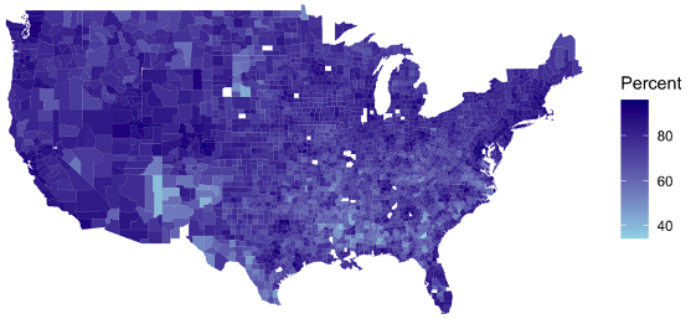
Percent of IT Workers per State



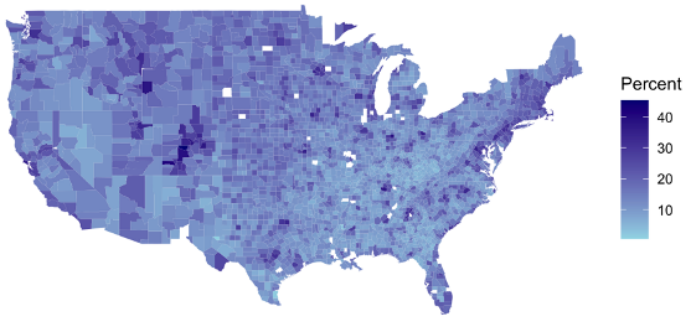
Median Household Income per State



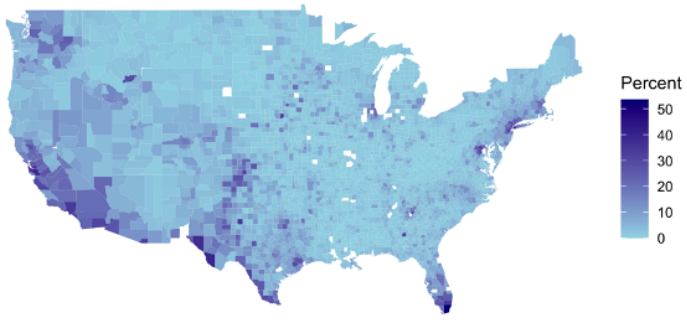
Percent of Broadband Access per County



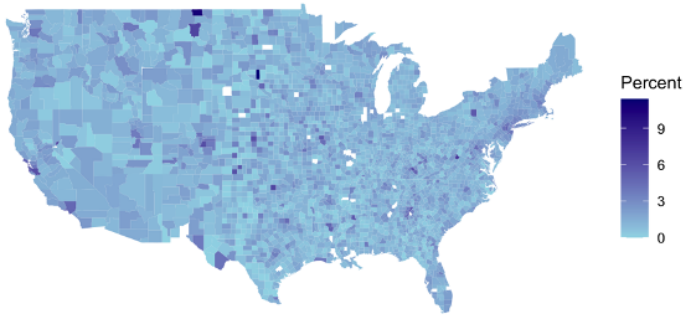
Percent of Population with College Education per County



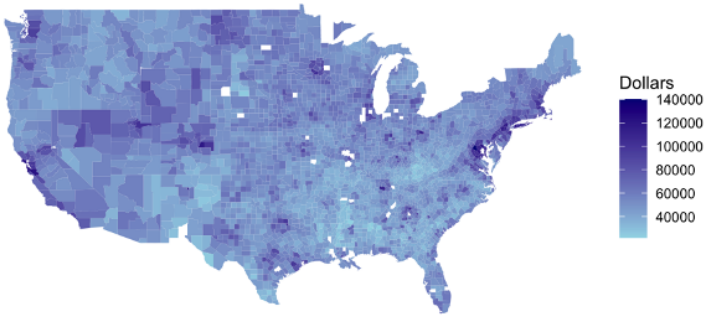
Percent of Foreign-born Population per County

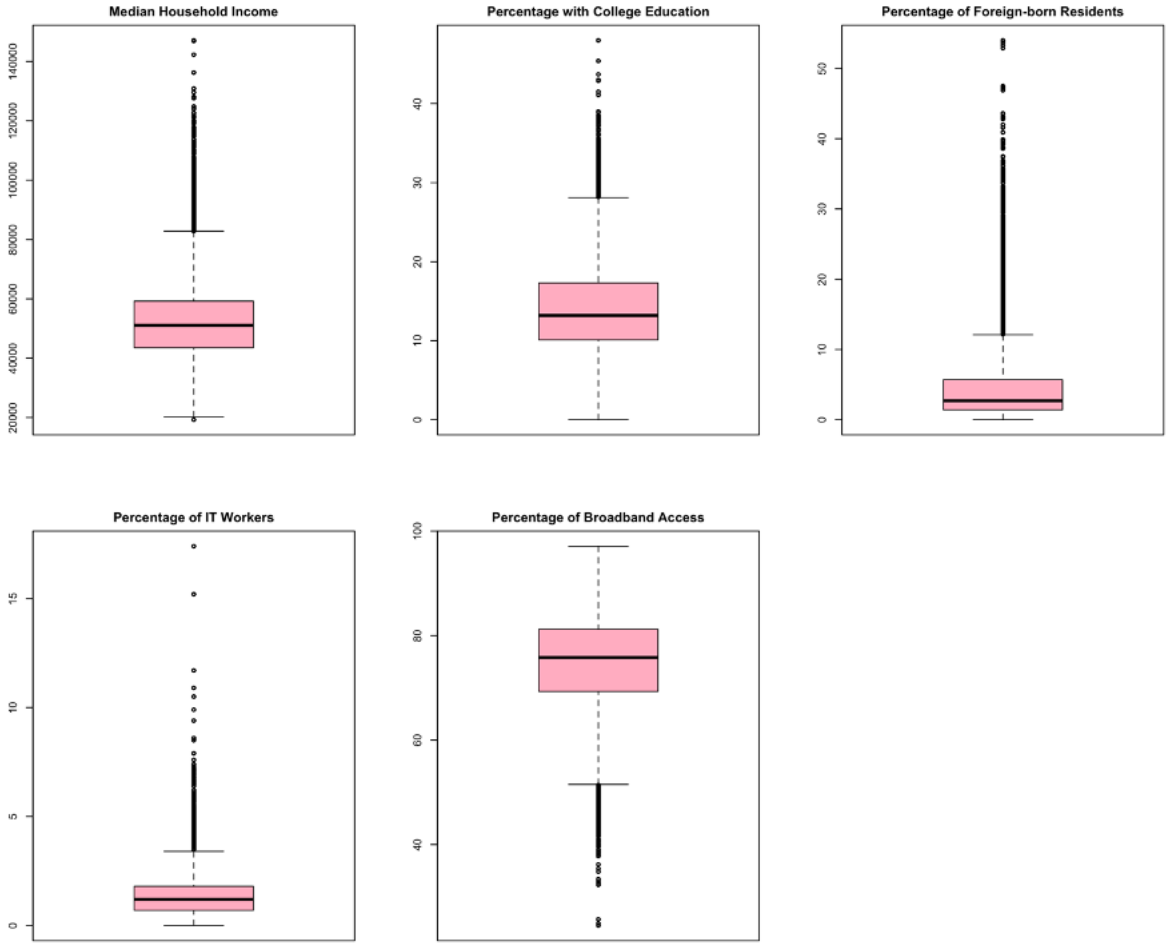


Percent of IT Workers per County



Median Household Income per County





2.2.2 Outlier Detection

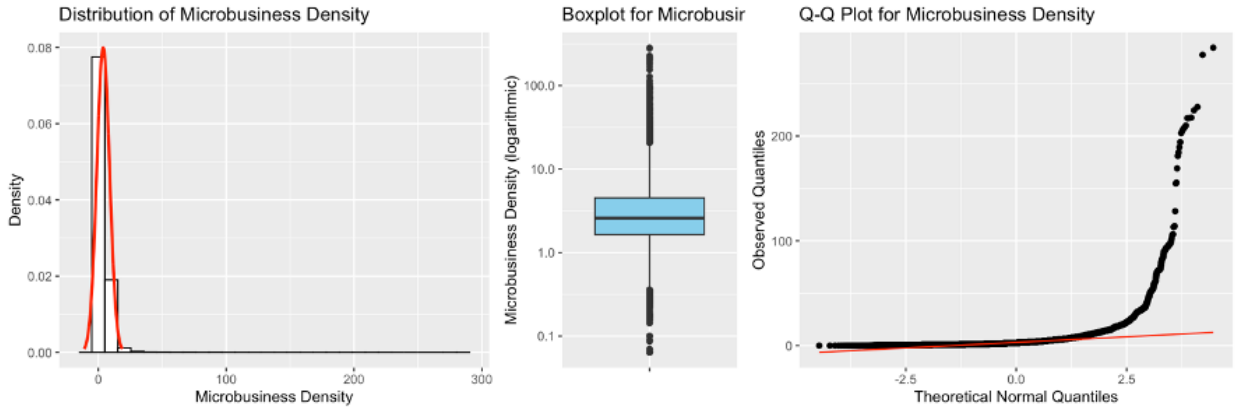
Outlier detection is an important step in data analysis, Outliers can potentially skew statistical measures, such as means and standard deviations, leading to biased results. They can also affect the assumptions of certain statistical models or algorithms. There are several methods to detect outliers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

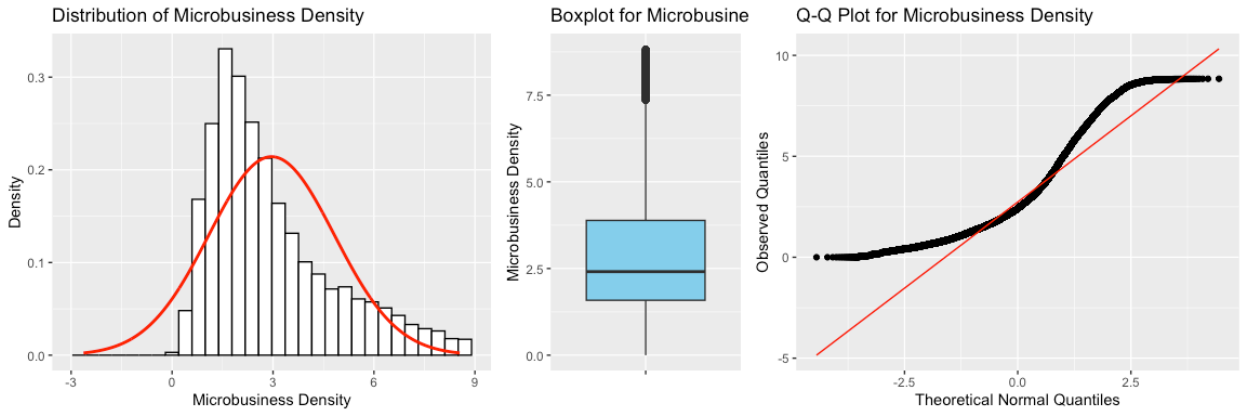
```
##   microbusiness_density
##         0%        25%        50%        75%       100%
##   0.000000   1.639344   2.586543   4.519231 284.340030
```

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 $Q1 - 1.5 * IQR$ to $Q3 + 1.5 * IQR$. 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
```

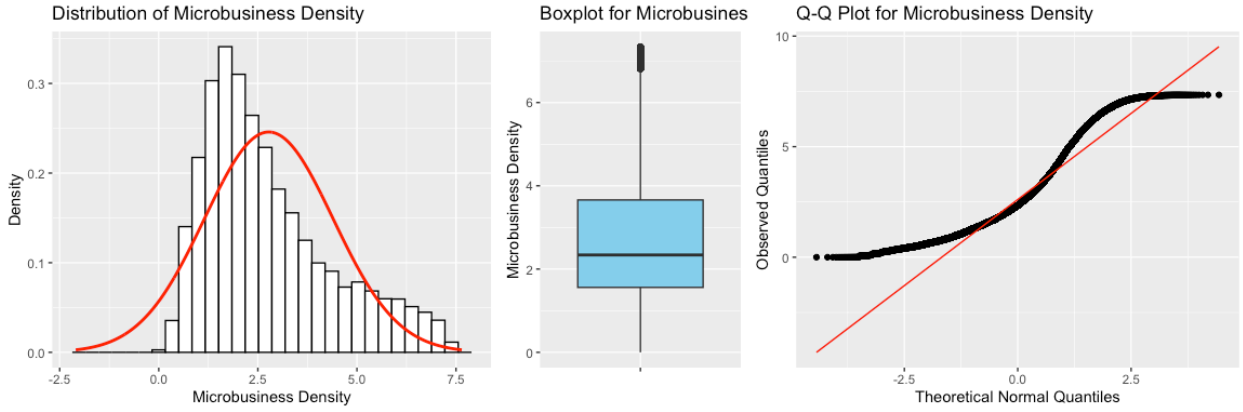
```
## Percent of outliers: 7.153315 %

## 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**.

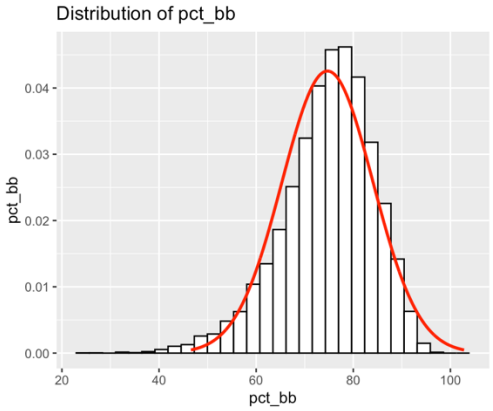


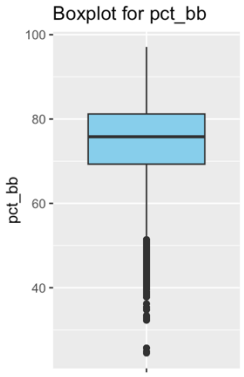
The distribution of microbusiness density has improved dramatically after removing the outliers using the decision range approach. We can repeat the mentioned method to further remove the values that we are now considering outliers.

```
## Number of outliers: 3946

## Percent of outliers: 3.47607 %

## Number of rows removed: 3946

## Total rows removed from merged_df: 12692
```

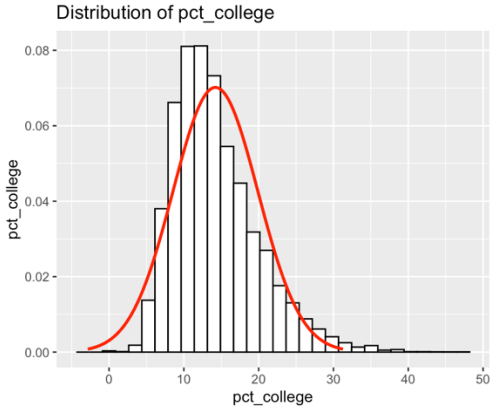


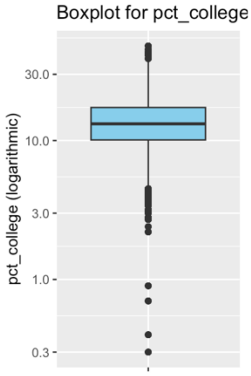
We can repeat the above steps for the census features (`pct_bb`, `pct_college`, `pct_foreign_born`, `pct_it_workers`, and `median_hh_inc`) as well.

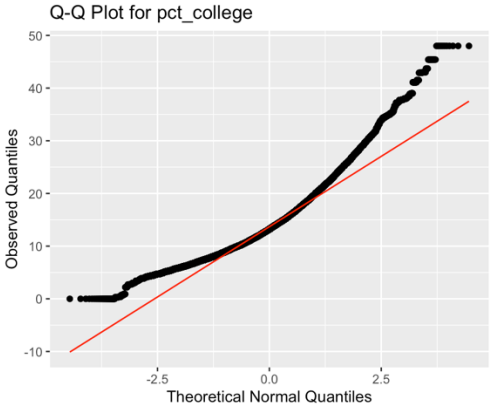

Distribution of pct_bb
pct_bb
pct_bb
0.00
0.01
0.02
0.03
0.04
20
40
60
80
100

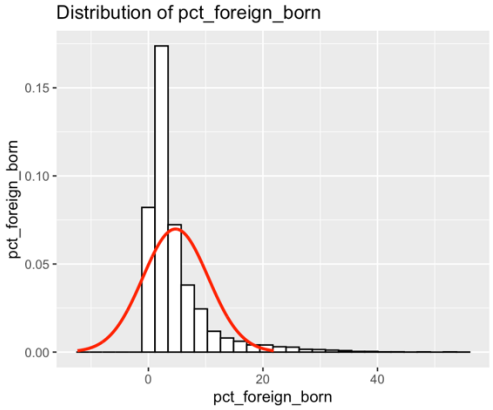

Boxplot for pct_bb
pct_bb
40
60
80
100

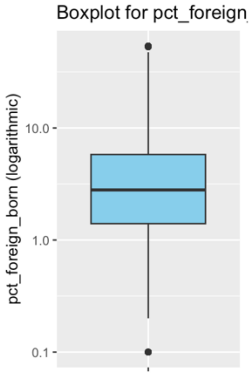

Q-Q Plot for pct_bb
Observed Quantiles
Theoretical Normal Quantiles
20
40
60
80
100
-2.5
0.0
2.5

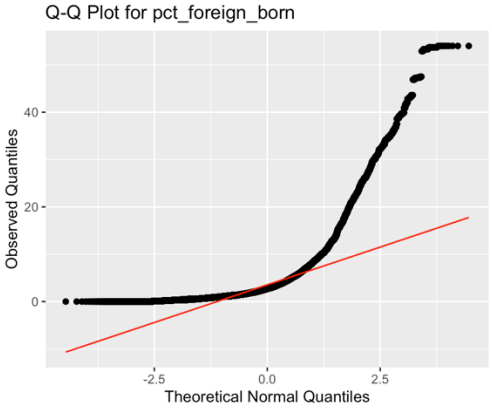

Distribution of pct_college
pct_college
pct_college
0.00
0.02
0.04
0.06
0.08
0
10
20
30
40
50

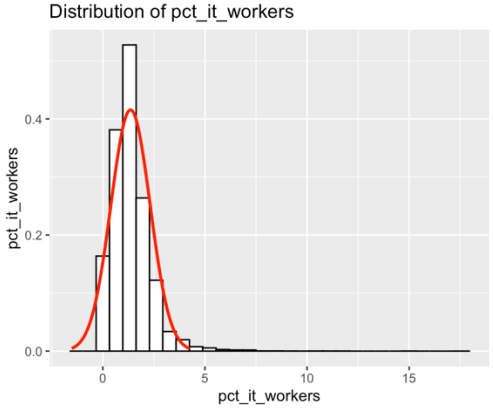

Boxplot for pct_college
pct_college (logarithmic)
0.3
1.0
3.0
10.0
30.0

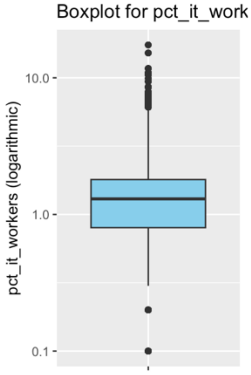

Q-Q Plot for pct_college
Observed Quantiles
Theoretical Normal Quantiles
-10
0
10
20
30
40
50
-2.5
0.0
2.5

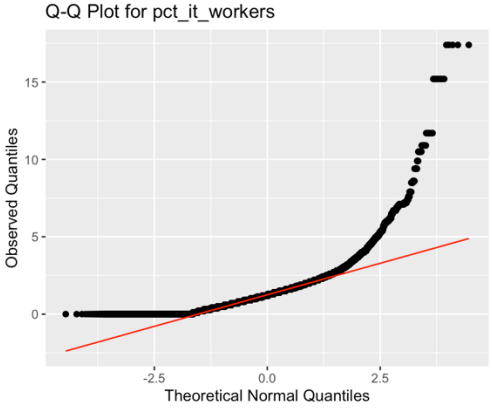

Distribution of pct_foreign_born
pct_foreign_born
pct_foreign_born
0.00
0.05
0.10
0.15
0
20
40

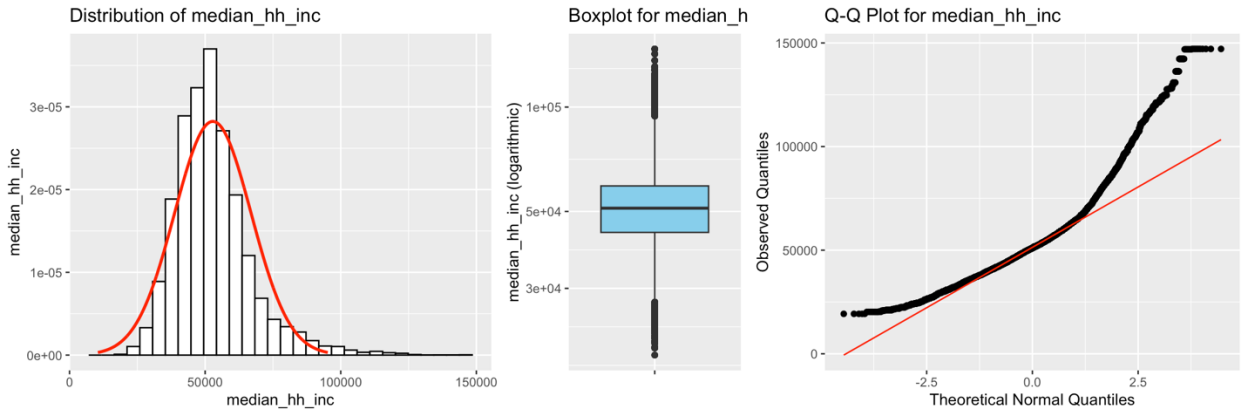

Boxplot for pct_foreign
pct_foreign_born (logarithmic)
0.1
1.0
10.0


Q-Q Plot for pct_foreign_born
Observed Quantiles
Theoretical Normal Quantiles
0
20
40
-2.5
0.0
2.5


Distribution of pct_it_workers
pct_it_workers
pct_it_workers
0.0
0.2
0.4
0
5
10
15


Boxplot for pct_it_work
pct_it_workers (logarithmic)
0.1
1.0
10.0


Q-Q Plot for pct_it_workers
Observed Quantiles
Theoretical Normal Quantiles
0
5
10
15
-2.5
0.0
2.5



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.

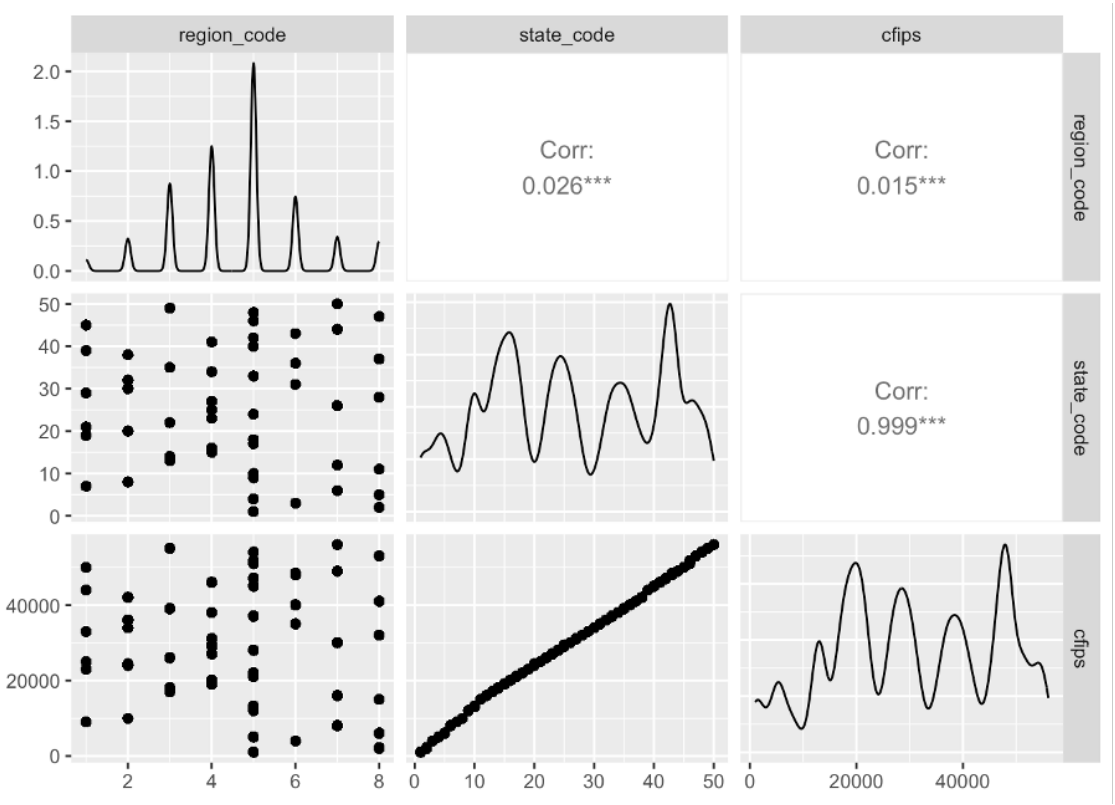
2.2.3 Correlation Plot

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.

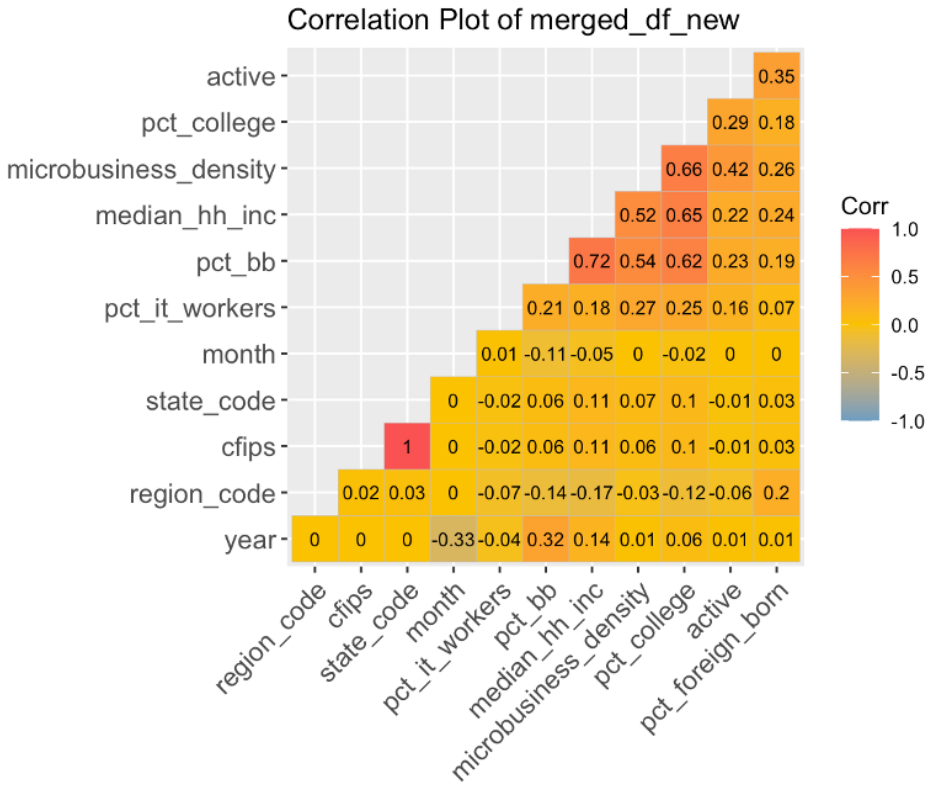
The correlation coefficient, which ranges from -1 to 1, measures the strength and direction of the relationship between two variables. A correlation coefficient of 1 indicates a perfect positive correlation, while a correlation coefficient of -1 indicates a perfect negative correlation.



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.

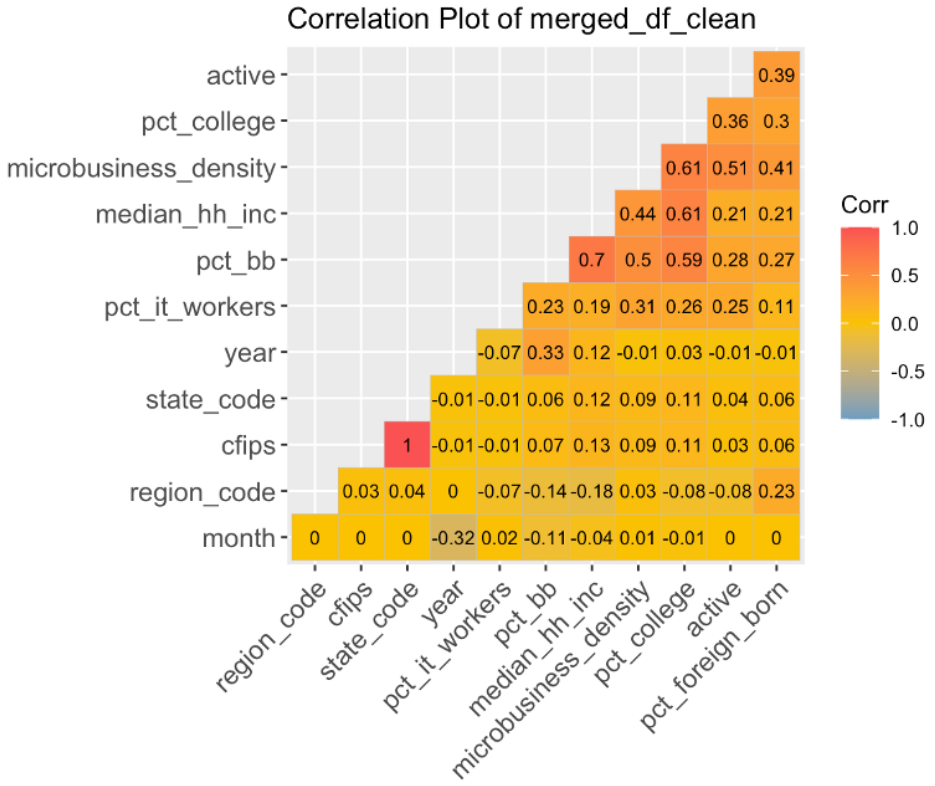
Now, we first plot the correlation between all the numeric features in our dataframe after removing the outliers from `microbusiness_density` column. Then, we plot the same plot on the dataframe we created after removing outliers from `microbusiness_density` and census features as well.



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

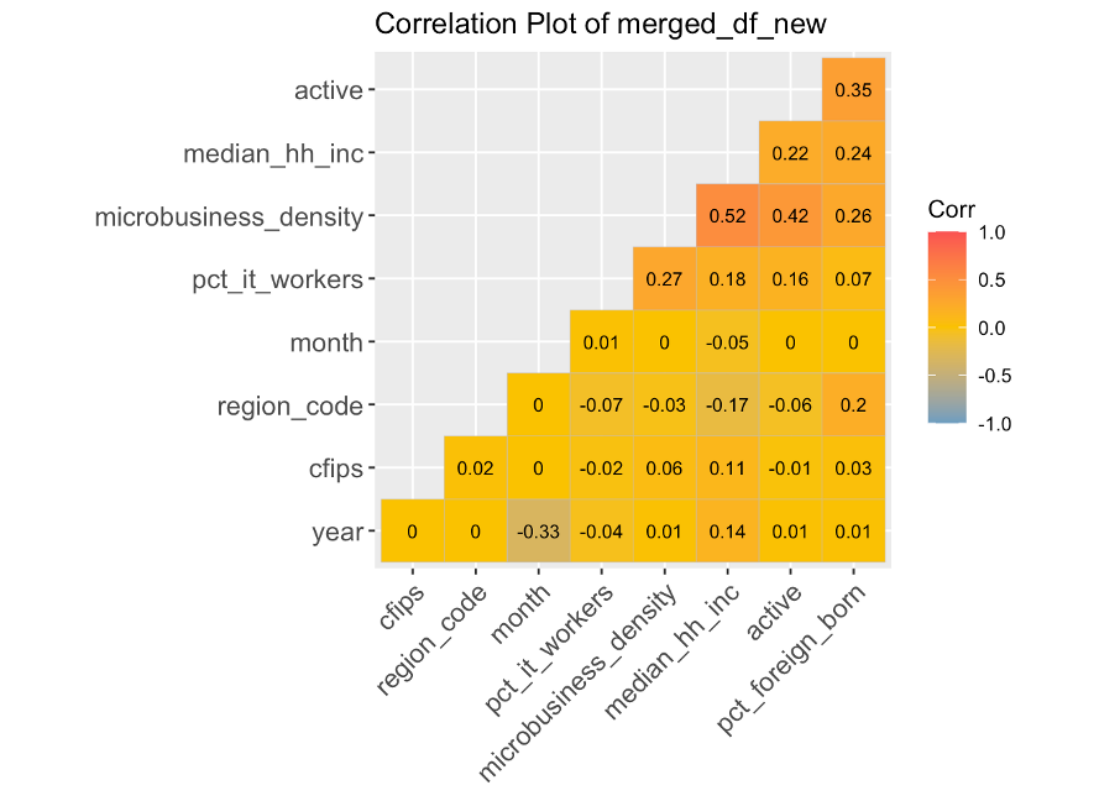
1. **0.01** to **0.19**: Very weak correlation
2. **0.20** to **0.39**: Weak correlation
3. **0.40** to **0.59**: Moderate correlation
4. **0.60** to **0.79**: Strong correlation
5. **0.80** to **0.99**: Very strong correlation

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.

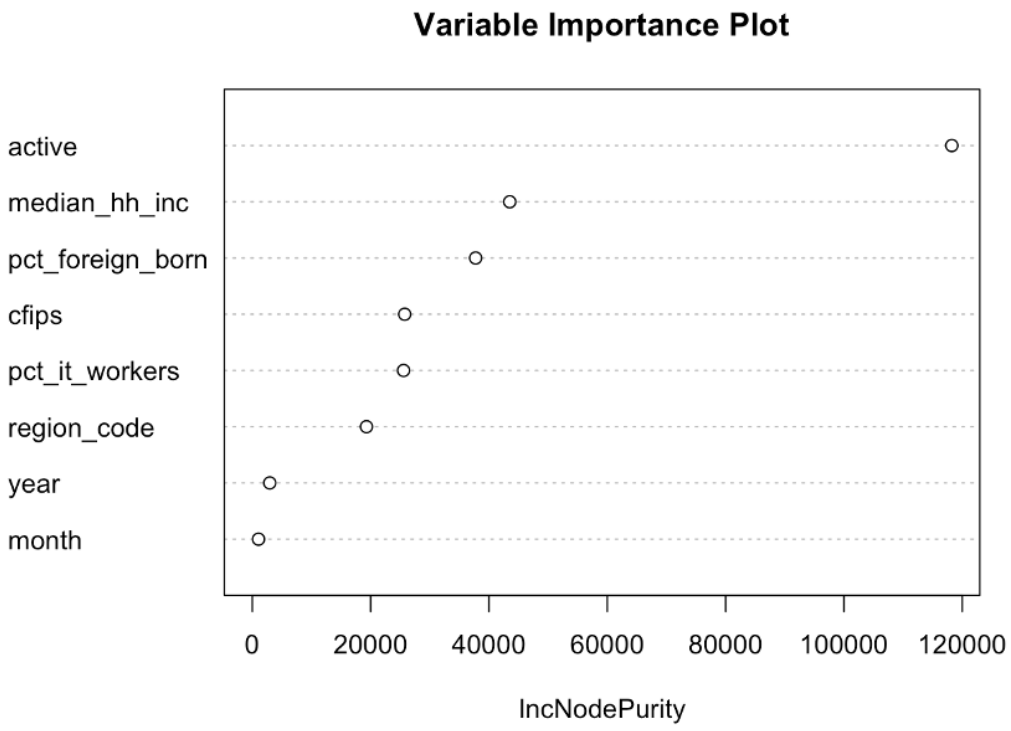


2.2.4 Feature Importance Using Random Forest

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.

2.3 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.

3 Confirmatory Data Analysis

3.1 Time Series Forecasting[2]

It is possible to predict future values of a time-dependent variable using time series forecasting based on observations from the past. The variable that is being measured in a time series changes over time, and the aim of time series forecasting is to spot patterns and trends in the data that can be used to make precise predictions about future values.

From predicting stock prices and weather patterns to predicting demand for goods and services, time series forecasting is utilized in a wide range of applications. The procedure usually involves analyzing historical data to recognize trends, seasonality, and other patterns, and then using statistical models and machine learning algorithms to predict future values.

Moving averages, exponential smoothing, and autoregressive integrated moving average (ARIMA) models are frequently used methods in time series forecasting. Accurate predictions can also be made using time series data analysis techniques that are more sophisticated, such as neural networks and deep learning algorithms that we will not cover in this review.

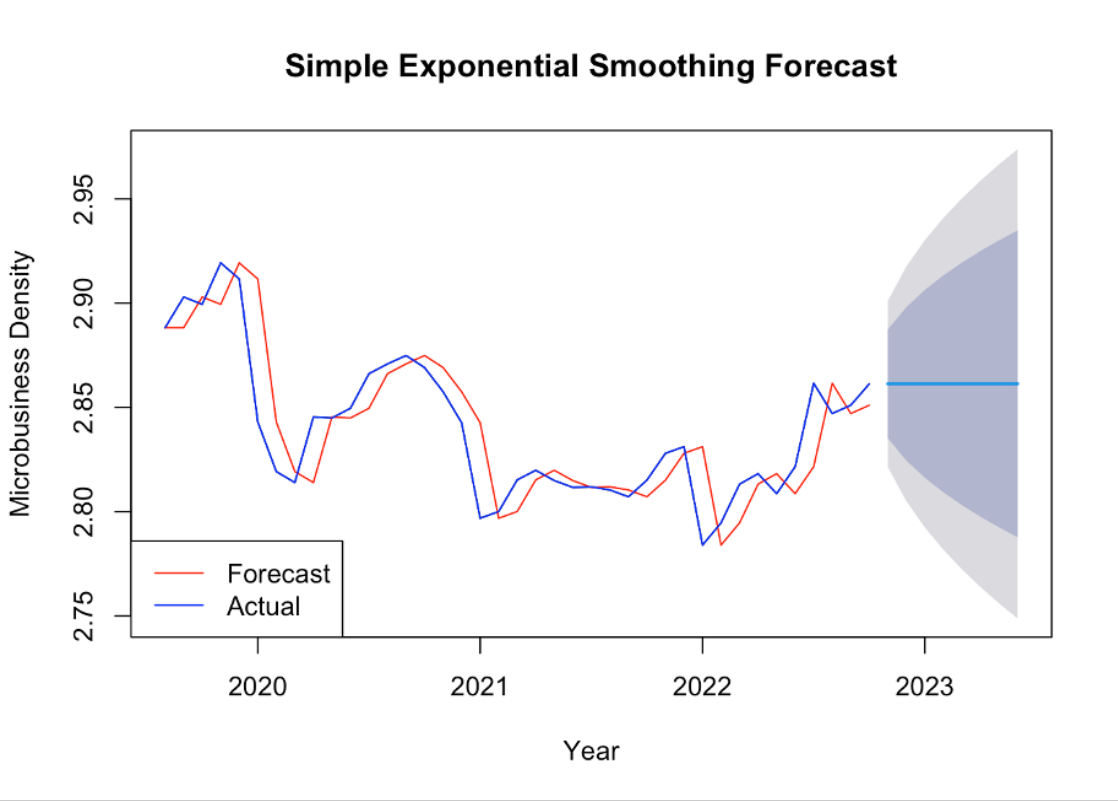
3.1.1 Simple Exponential Smoothing Forecasting

Exponential smoothing was proposed in the late 1950s and has motivated some of the most successful forecasting methods. 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.

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. 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.

---

[2] Forecasting: Principles and Practice (3rd ed) by Rob. J. Hyndman and George Athanasopoulos



The provided image displays forecasts for the period from November 2022 to June 2023. Additionally, it illustrates the one-step-ahead fitted values and the corresponding data spanning from August 2019 to October 2022. However, it is worth noting that the predicted values generated by this method exhibit a wide range and lack accuracy, making them unsuitable for effectively predicting microbusiness density. Moreover, it seems that the other predictors at our disposal have not been utilized in this analysis. As a result, it is recommended to proceed with statistical models, such as linear regression and other appropriate techniques, to enhance the accuracy and usefulness of the predictions for microbusiness density. By leveraging these statistical models, we can harness their capabilities to improve the precision and reliability of our forecasts.

3.2 Linear Models

Linear models are a class of statistical models that assume a linear relationship between the input features and the target variable. The fundamental idea behind linear models is to fit a linear equation to the data that minimizes the sum of squared errors. Linear models, such as Linear Regression, Ridge, Lasso, and ElasticNet, are widely used in various domains due to their simplicity, interpretability, and fast computation. They provide a straightforward approach to understanding the impact of individual features on the target variable and can handle large datasets efficiently. Linear models can be extended with regularization techniques like Ridge, Lasso, and ElasticNet to prevent overfitting, handle multicollinearity, and perform feature selection. While linear models work well when the relationship between features and the target variable is linear, they may not capture complex nonlinear patterns in the data.

3.2.1 Linear Regression

Linear Regression is a simple linear model that is used for regression tasks. It works by fitting a linear function to the data that minimizes the sum of the squared errors. Linear Regression is a basic model that is often used as a benchmark for more complex models. It is suitable for situations where the relationship between the features and target variable is linear.

```
## 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]]
```

3.2.2 Ridge

Ridge is a linear regression model that is used for regularization. It works by adding a penalty term to the loss function that is proportional to the square of the coefficients. This penalty encourages the model to produce solutions where the magnitude of the coefficients is small. Ridge is often used in situations where there are many features and the data is highly correlated.

```
## 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]]
```

3.2.3 Lasso

Lasso is another linear regression model that is used for feature selection and regularization. It works by adding a penalty term to the loss function that is proportional to the absolute value of the coefficients. This penalty encourages the model to produce sparse solutions where many of the coefficients are set to zero. Lasso is commonly used in situations where there are many features and a limited number of samples.

```
## 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]]
```

3.2.4 Elastic Net

Elastic Net is a linear regression model that is used for feature selection and regularization. It combines the penalties of L1 and L2 regularization to balance the strengths of each approach. Elastic Net works by minimizing the sum of the squared errors while also adding a penalty term to the loss function. It is often used in situations where there are many features and a limited number of samples.

```
## 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]]
```

3.3 Ensemble Models

Ensemble models are machine learning models that combine the predictions of multiple individual models to make more accurate and robust predictions. These models, including Random Forest, XGBoost, and LightGBM, leverage the concept of "wisdom of the crowd" to improve prediction performance. Ensemble models work by training a collection of base models, such as decision trees, and aggregating their predictions through methods like voting or averaging. By combining the strengths of multiple models, ensemble models can capture complex relationships, handle noisy data, and reduce the risk of overfitting. They excel in scenarios where individual models may struggle, and their ensemble nature allows them to make more accurate predictions compared to a single model. Ensemble models require more computational resources and tuning compared to linear models, but they often deliver higher predictive power and are popular choices for a wide range of regression problems.

3.3.1 Random Forest

Random Forest is a machine learning model that belongs to the family of ensemble methods. It works by building multiple decision trees on random subsets of the data and then combining their predictions to make a final prediction. Random Forest is known for its ability to handle noisy data and high-dimensional feature spaces. It is a popular choice for regression problems in many domains.

```
## 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]]
```

3.3.2 Extreme Gradient Boosting (XGB)

XGB Regressor is gradient boosting algorithm. It uses an optimized gradient boosting framework to improve the accuracy of predictions. XGB Regressor works by iteratively adding new decision trees to the model and then combining their results to make a final prediction. It is known for its high predictive power and ability to handle complex datasets.

```
## 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]]
```

3.3.3 Light Gradient-Boosting Machine (LGBM)

LGBM is a machine learning model that belongs to the family of gradient boosting algorithms. It uses a light gradient boosting framework, which enables it to train models faster and with better accuracy than traditional gradient boosting algorithms. LGBM Regressor works by adding new decision trees to the model in a way that minimizes the loss function. It is suitable for regression problems with large datasets and high-dimensional feature spaces.

```
## 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]]
```

3.4 Conclusion

Based on the results of the various models trained and tested, it can be concluded that the Random Forest model performs the best in terms of predicting the density of microbusinesses in the United States. The model achieved an accuracy score of 98.03%, indicating that it is highly effective in predicting the density of microbusinesses. The XGB model also performs well with an accuracy score of 87.31%, followed by the LGBM model with an accuracy score of 82.16%.

The SMAPE metric, which measures the percentage difference between the predicted and actual density values, shows that the Random Forest model is the most accurate, with a score of only 3.19%. The XGB model has the second lowest SMAPE score of 17.17%, followed by the LGBM model with a score of 24.46%.

The TPR (True Positive Rate) and FPR (False Positive Rate) metrics indicate the ability of the models to correctly identify areas with high density of microbusinesses and avoid false positives, respectively. The Random Forest model has the highest TPR at 98.73% and the lowest FPR at 3.60%, indicating its superior performance in this regard.

Overall, the results suggest that ensemble models such as Random Forest, XGB, and LGBM are well-suited for predicting the density of microbusinesses in the United States. These models can capture the complex relationships between the various features and are less prone to overfitting compared to linear models. However, further analysis and experimentation may be necessary to determine the most appropriate model for this specific task, as well as to identify other relevant features that could improve the accuracy of the prediction.