# 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 of 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 update 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 avaiable 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).

## 2. Setup the Environment

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
library(tidyverse)
# ggplot2, purrr, tibble, dplyr, tidyr, stringr, readr, forcats
library(mice)
library(maps)
library(gridExtra)
library(caret)
library(gbm)
#library(png)
#library(ggmap)
library(viridis)
library(mapdata)
library(corrplot)
library(reshape2)

# libraries required for calculating SMAPE
#library(forecast)
library(Metrics)
#actual <- c(10, 20, 30, 40, 50)
#forecasted <- c(20, 20, 10, 40, 60)
#smape(actual, forecasted)
```

## 3. Explanatory Data Analysis

### 3.1. Exploring the datasets

Explore the datasets to get a better understanding of the data.
Load the **train**, **test**, and **census_starter** datasets into R dataframes.

```
# Load train.csv into a dataframe
train_df <- read.csv("./datasets/train.csv")

# Load test.csv into a dataframe
test_df <- read.csv("./datasets/test.csv")

# Load census_starter.csv into a dataframe
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 `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 6 rows of the dataframes
head(train_df)
```

```
##            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
## 4 1001_2019-11-01  1001 Autauga County Alabama         2019-11-01
## 5 1001_2019-12-01  1001 Autauga County Alabama         2019-12-01
## 6 1001_2020-01-01  1001 Autauga County Alabama         2020-01-01
##   microbusiness_density active
## 1              3.007682   1249
## 2              2.884870   1198
## 3              3.055843   1269
## 4              2.993233   1243
## 5              2.993233   1243
## 6              2.969090   1242
```

```
head(test_df)
```

```
##            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
## 4 1007_2022-11-01  1007         2022-11-01
## 5 1009_2022-11-01  1009         2022-11-01
## 6 1011_2022-11-01  1011         2022-11-01
```

```
head(census_df)
```

```
##   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
## 4        62.0        66.1        69.2        76.1        74.6  1007
## 5        65.8        68.5        73.0        79.6        81.0  1009
## 6        49.4        58.9        60.1        60.6        59.4  1011
##   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
## 4              8.1              7.6              6.5              7.4
## 5              8.7              8.1              8.6              8.9
## 6              6.6              7.4              7.4              6.1
##   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
## 4              7.9                    1.0                    1.4
## 5              9.3                    4.5                    4.4
## 6              8.1                    1.8                    0.9
##   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
## 4                  1.5                  1.6                  1.1
## 5                  4.5                  4.4                  4.5
## 6                  0.7                  1.5                  1.2
##   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
## 4                 1.2                 1.4                 1.6
## 5                 1.3                 1.4                 0.9
## 6                 0.4                 0.3                 0.5
##   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
## 4                 1.7                 2.1              43404              45340
## 5                 1.1                 0.9              47412              48695
## 6                 0.3                 0.2              29655              32152
##   median_hh_inc_2019 median_hh_inc_2020 median_hh_inc_2021
## 1              58731              57982              62660
## 2              58320              61756              64346
## 3              32525              34990              36422
## 4              47542              51721              54277
## 5              49358              48922              52830
## 6              37785              33866              29063
```

```
# Display the last 6 rows of the dataframes
tail(train_df)
```

```
##                  row_id cfips        county   state first_day_of_month
## 122260 56045_2022-05-01 56045 Weston County Wyoming         2022-05-01
## 122261 56045_2022-06-01 56045 Weston County Wyoming         2022-06-01
## 122262 56045_2022-07-01 56045 Weston County Wyoming         2022-07-01
## 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
## 122260              1.803249    101
## 122261              1.803249    101
## 122262              1.803249    101
## 122263              1.785395    100
## 122264              1.785395    100
## 122265              1.785395    100
```

```
tail(test_df)
```

```
##                row_id cfips first_day_of_month
## 25075 56035_2023-06-01 56035         2023-06-01
## 25076 56037_2023-06-01 56037         2023-06-01
## 25077 56039_2023-06-01 56039         2023-06-01
## 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)
```

```
##      pct_bb_2017 pct_bb_2018 pct_bb_2019 pct_bb_2020 pct_bb_2021 cfips
## 3137        82.9        81.7        85.6        88.1        89.8 56035
## 3138        82.2        82.4        84.0        86.7        88.4 56037
## 3139        83.5        85.9        87.1        89.1        90.5 56039
## 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
## 3137             19.2             19.0             16.7             21.7
## 3138             15.3             15.2             14.8             13.7
## 3139             37.7             37.8             38.9             37.2
## 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
## 3137             20.9                   3.9                   3.1
## 3138             12.4                   5.0                   5.3
## 3139             38.3                  10.8                  11.2
## 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
## 3137                   4.4                   5.1                   5.1
## 3138                   4.7                   5.2                   5.5
## 3139                  11.8                  11.4                  11.1
## 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
## 3137                 0.1                 0.0                 0.0
## 3138                 0.6                 0.6                 1.0
## 3139                 0.7                 1.2                 1.4
## 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
## 3137                 0.0                 0.0              84911
## 3138                 0.9                 1.0              71083
## 3139                 1.5                 2.0              80049
## 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
## 3137              78680              77403              78655
## 3138              73008              74843              73384
## 3139              83831              84678              87053
## 3140              58235              63403              72458
## 3141              53426              54158              57306
## 3142              52867              57031              53333
```

```
##      median_hh_inc_2021
## 3137              82342
## 3138              76668
## 3139              94498
## 3140              75106
## 3141              62271
## 3142              65566
```

```
# Display information about the train dataframe
str(train_df)
```

```
## 'data.frame':    122265 obs. of  7 variables:
##  $ row_id              : chr  "1001_2019-08-01" "1001_2019-09-01" "1001_2019-10-01" "1001_2019-11-0
##  $ 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  : chr  "2019-08-01" "2019-09-01" "2019-10-01" "2019-11-01" ...
##  $ 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 ...
```

```
cat(rep("=", 40), "\n") # Print a line of 40 equal signs
```

```
## = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

```
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
```

```
# Display information about the test dataframe
str(test_df)
```

```
## 'data.frame':    25080 obs. of  3 variables:
##  $ row_id            : chr  "1001_2022-11-01" "1003_2022-11-01" "1005_2022-11-01" "1007_2022-11-01"
##  $ cfips             : int  1001 1003 1005 1007 1009 1011 1013 1015 1017 1019 ...
##  $ first_day_of_month: chr  "2022-11-01" "2022-11-01" "2022-11-01" "2022-11-01" ...
```

```r
cat(rep("=", 40), "\n") # Print a line of 40 equal signs
```

```
## = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

```r
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
```

```r
# Display information about the census dataframe
str(census_df)
```

```
## 'data.frame':    3142 obs. of  26 variables:
##  $ pct_bb_2017          : num  76.6 74.5 57.2 62 65.8 49.4 58.2 71 62.8 67.5 ...
##  $ pct_bb_2018          : num  78.9 78.1 60.4 66.1 68.5 58.9 62.1 73 66.5 68.6 ...
##  $ pct_bb_2019          : num  80.6 81.8 60.5 69.2 73 60.1 64.6 75.1 69.4 70.7 ...
##  $ pct_bb_2020          : num  82.7 85.1 64.6 76.1 79.6 60.6 73.6 79.8 74.5 75 ...
##  $ pct_bb_2021          : num  85.5 87.9 64.6 74.6 81 59.4 76.3 81.6 77.1 76.7 ...
##  $ cfips                : int  1001 1003 1005 1007 1009 1011 1013 1015 1017 1019 ...
##  $ pct_college_2017     : num  14.5 20.4 7.6 8.1 8.7 6.6 9.6 10.2 9 6.6 ...
##  $ pct_college_2018     : num  15.9 20.7 7.8 7.6 8.1 7.4 9.7 10.2 9.3 6.8 ...
##  $ pct_college_2019     : num  16.1 21 7.6 6.5 8.6 7.4 9.7 10.5 9.5 6.6 ...
##  $ pct_college_2020     : num  16.7 20.2 7.3 7.4 8.9 6.1 10.1 10.5 10.5 6.3 ...
##  $ pct_college_2021     : num  16.4 20.6 6.7 7.9 9.3 8.1 8.1 11.4 9.6 6.2 ...
##  $ pct_foreign_born_2017: num  2.1 3.2 2.7 1 4.5 1.8 1 2.6 1.3 0.7 ...
##  $ pct_foreign_born_2018: num  2 3.4 2.5 1.4 4.4 0.9 1.4 2.7 1.4 0.8 ...
##  $ pct_foreign_born_2019: num  2.3 3.7 2.7 1.5 4.5 0.7 0.8 2.7 1.8 0.9 ...
##  $ pct_foreign_born_2020: num  2.3 3.4 2.6 1.6 4.4 1.5 1.9 2.5 1.9 1.9 ...
##  $ pct_foreign_born_2021: num  2.1 3.5 2.6 1.1 4.5 1.2 1.7 2.5 2 2 ...
##  $ pct_it_workers_2017  : num  1.3 1.4 0.5 1.2 1.3 0.4 1.1 1.4 2.4 1.4 ...
##  $ pct_it_workers_2018  : num  1.1 1.3 0.3 1.4 1.4 0.3 1.4 1.4 2.1 1.3 ...
##  $ pct_it_workers_2019  : num  0.7 1.4 0.8 1.6 0.9 0.5 1.7 1.2 2.1 1.2 ...
##  $ pct_it_workers_2020  : num  0.6 1 1.1 1.7 1.1 0.3 1.3 1 2.3 0.9 ...
##  $ pct_it_workers_2021  : num  1.1 1.3 0.8 2.1 0.9 0.2 1.4 1 1.8 0.4 ...
##  $ median_hh_inc_2017   : int  55317 52562 33368 43404 47412 29655 36326 43686 37342 40041 ...
##  $ median_hh_inc_2018   : num  58786 55962 34186 45340 48695 ...
##  $ median_hh_inc_2019   : int  58731 58320 32525 47542 49358 37785 40688 47255 42289 41919 ...
##  $ median_hh_inc_2020   : num  57982 61756 34990 51721 48922 ...
##  $ median_hh_inc_2021   : num  62660 64346 36422 54277 52830 ...
```

```r
cat(rep("=", 40), "\n") # Print a line of 40 equal signs
```

```
## = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

```
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
```

### 3.2. Checking the Dataframes for Missing Values

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.

```
# Check for missing values in the train data frame
colSums(is.na(train_df))
```

```
##                row_id                 cfips                county
##                     0                     0                     0
##                 state    first_day_of_month microbusiness_density
##                     0                     0                     0
##                active
##                     0
```

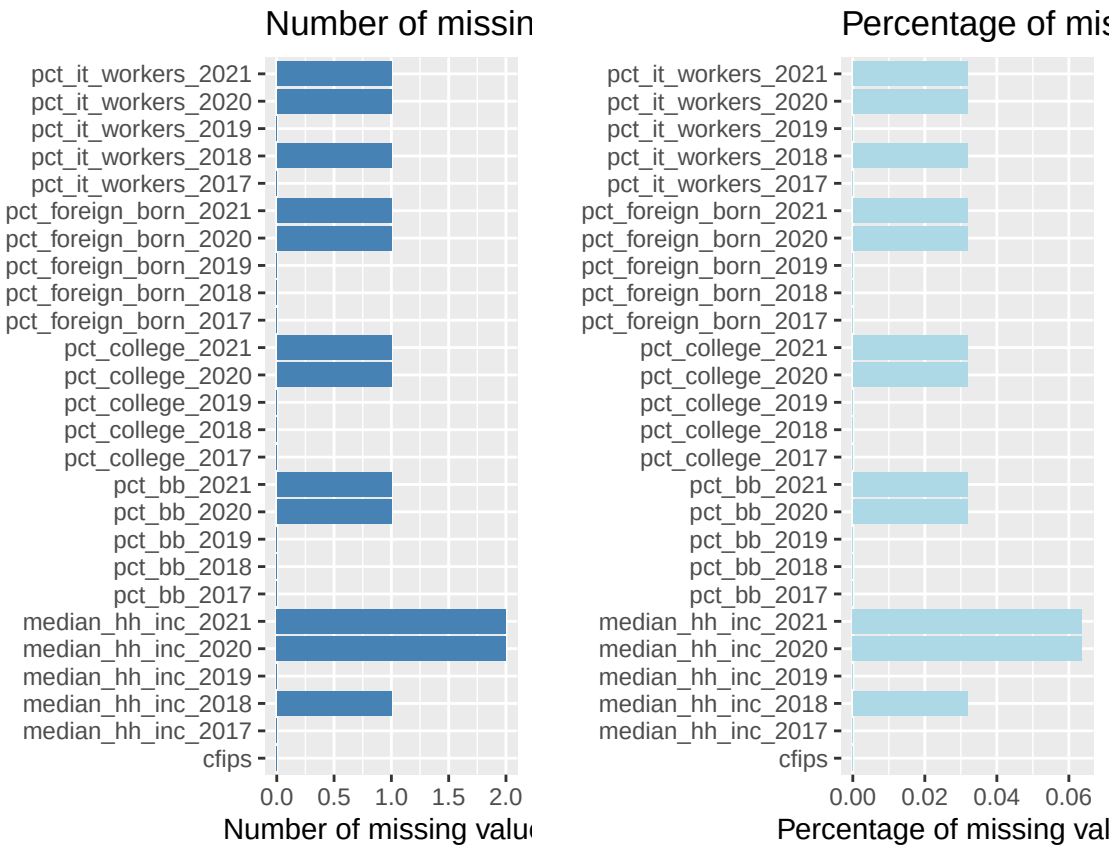
```
# Check for missing values in the test data frame
colSums(is.na(test_df))
```

```
##             row_id              cfips first_day_of_month
##                  0                  0                  0
```

```
# Check for missing values in the census data frame
colSums(is.na(census_df))
```

```
##           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
```

```
#{r fig.width=7, fig.align='center', fig.height=4, out.width='100%'}
# Calculate the number and percentage of missing values for each column
missing_data <- census_df %>%
  summarise_all(~ sum(is.na(.))) %>%
  gather(variable, missing_count) %>%
  mutate(missing_percent = missing_count/nrow(census_df)*100)

# Create two plots side by side
plot1 <- ggplot(missing_data, aes(x = missing_count, y = variable)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(x = "Number of missing values", y = "") +
  ggtitle("Number of missing values in census_df") +
  theme_gray()

plot2 <- ggplot(missing_data, aes(x = missing_percent, y = variable)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  labs(x = "Percentage of missing values", y = "") +
  ggtitle("Percentage of missing values in census_df") +
  theme_gray()

# Arrange the two plots side by side
grid.arrange(plot1, plot2, ncol = 2)
```




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:

```
# Identify rows with missing values in census_df
missing_rows <- which(!complete.cases(census_df))

# Identify columns with missing values for each missing row
for (i in missing_rows) {
  cat("Row", i, "has missing values in columns:",
      paste(names(census_df)[is.na(census_df[i,])], collapse = ", "), "\n")
}
```

```
## Row 93 has missing values in columns: pct_bb_2020, pct_bb_2021, pct_college_2020, pct_college_2021, p
## 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
```

```
print(census_df[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          NA          NA  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               NA
## 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                NA                   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                    NA                    NA
## 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                  NA                 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                    NA                  NA              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                 NA
## 1817                 NA              39952              42264
## 2645              53194              53088                 NA
```

```
## 2674              81875              83750              44076
##      median_hh_inc_2021
## 93                   NA
## 1817              46994
## 2645              38659
## 2674                 NA
```

### 3.3. Dealing with Missing Values

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

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.

```
# Impute missing data using mice
imputed_data <- mice(census_df, m = 5, maxit = 50, method = "pmm", print = FALSE)
# Extract imputed data
imputed_data <- complete(imputed_data)
```

```
# Check for missing values in imputed data
colSums(is.na(imputed_data))
```

```
##           pct_bb_2017           pct_bb_2018           pct_bb_2019
##                     0                     0                     0
##           pct_bb_2020           pct_bb_2021                 cfips
##                     0                     0                     0
##      pct_college_2017      pct_college_2018      pct_college_2019
##                     0                     0                     0
##      pct_college_2020      pct_college_2021 pct_foreign_born_2017
##                     0                     0                     0
## pct_foreign_born_2018 pct_foreign_born_2019 pct_foreign_born_2020
##                     0                     0                     0
## pct_foreign_born_2021   pct_it_workers_2017   pct_it_workers_2018
##                     0                     0                     0
##   pct_it_workers_2019   pct_it_workers_2020   pct_it_workers_2021
##                     0                     0                     0
##    median_hh_inc_2017    median_hh_inc_2018    median_hh_inc_2019
##                     0                     0                     0
##    median_hh_inc_2020    median_hh_inc_2021
##                     0                     0
```

```
# 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        84.6        82.9  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             17.8
## 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               17.0                   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                   6.7                   6.5
## 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                 0.9                 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                   3.4                 3.1              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              81169
## 1817              37610              39952              42264
## 2645              53194              53088              43950
## 2674              81875              83750              44076
##      median_hh_inc_2021
## 93                81817
## 1817              46994
## 2645              38659
## 2674              53138
```

### 3.4. Checking the Time Frame of *train* and *test* Dataframes

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

```
index <- unique(train_df$first_day_of_month)
print(index)
```

```
##  [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

```
index <- unique(test_df$first_day_of_month)
print(index)
```

```
## [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

### 3.5. Adding New Columns to *train* and *test*

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 is a string that 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.

```
# Add year, month and year_month columns to train_df
train_df$year <- as.integer(substr(train_df$first_day_of_month, 1, 4))
train_df$month <- as.integer(substr(train_df$first_day_of_month, 6, 7))
train_df$year_month <- substr(train_df$first_day_of_month, 1, 7)

# Add year, month and year_month columns to test_df
test_df$year <- as.integer(substr(test_df$first_day_of_month, 1, 4))
test_df$month <- as.integer(substr(test_df$first_day_of_month, 6, 7))
test_df$year_month <- substr(test_df$first_day_of_month, 1, 7)
```

### 3.6. Merging the *train* and *imputed_data* datasets

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** dataframe. 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. 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**.

```
# Set variables of interest
vars <- c("pct_bb", "pct_college", "pct_foreign_born", "pct_it_workers", "median_hh_inc")

# Loop through variables and merge with train_df
merged_df <- train_df
```

```
for (var in vars) {
  # Select columns and pivot longer
  merged_df <- imputed_data %>%
    select(cfips, paste0(var, "_2017"):paste0(var, "_2020")) %>%
    pivot_longer(cols = starts_with(var),
                 names_to = "year",
                 values_to = var) %>%
    # Modify year and month columns
    mutate(year = as.integer(str_sub(year, -4)) + 2) %>%
    uncount(12, .id = "month") %>%
    mutate(month = month) %>%
    # Merge with merged_df
    merge(merged_df, by = c("cfips", "year", "month"), all.x = TRUE) %>%
    arrange(cfips, row_id)
}
merged_df <- merged_df %>%
  select(row_id, cfips, county, state, first_day_of_month, microbusiness_density, active, year_month, y
```

```
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
```

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)
```

```
colSums(is.na(merged_df))
```

```
##                row_id                 cfips                county
##                     0                     0                     0
##                 state    first_day_of_month microbusiness_density
##                     0                     0                     0
##                active            year_month                  year
##                     0                     0                     0
##                 month                pct_bb           pct_college
##                     0                     0                     0
##      pct_foreign_born        pct_it_workers         median_hh_inc
##                     0                     0                     0
```

```
summary(merged_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         year_month
##  Length:122265      Min.   :  0.000       Min.   :      0   Length:122265
##  Class :character   1st Qu.:  1.639       1st Qu.:    145   Class :character
##  Mode  :character   Median :  2.587       Median :    488   Mode  :character
##                     Mean   :  3.818       Mean   :   6443
##                     3rd Qu.:  4.519       3rd Qu.:   2124
##                     Max.   :284.340       Max.   :1167744
##       year          month            pct_bb       pct_college
##  Min.   :2019   Min.   : 1.000   Min.   :24.50   Min.   : 0.00
##  1st Qu.:2020   1st Qu.: 4.000   1st Qu.:69.30   1st Qu.:10.10
##  Median :2021   Median : 7.000   Median :75.80   Median :13.20
##  Mean   :2021   Mean   : 6.692   Mean   :74.69   Mean   :14.22
##  3rd Qu.:2022   3rd Qu.:10.000   3rd Qu.:81.20   3rd Qu.:17.30
##  Max.   :2022   Max.   :12.000   Max.   :97.10   Max.   :48.00
##  pct_foreign_born pct_it_workers   median_hh_inc
##  Min.   : 0.000   Min.   : 0.000   Min.   : 19264
##  1st Qu.: 1.400   1st Qu.: 0.700   1st Qu.: 43507
##  Median : 2.700   Median : 1.200   Median : 51094
##  Mean   : 4.745   Mean   : 1.356   Mean   : 52830
##  3rd Qu.: 5.700   3rd Qu.: 1.800   3rd Qu.: 59230
##  Max.   :54.000   Max.   :17.400   Max.   :147111
```

## 3.7. Data Visualization

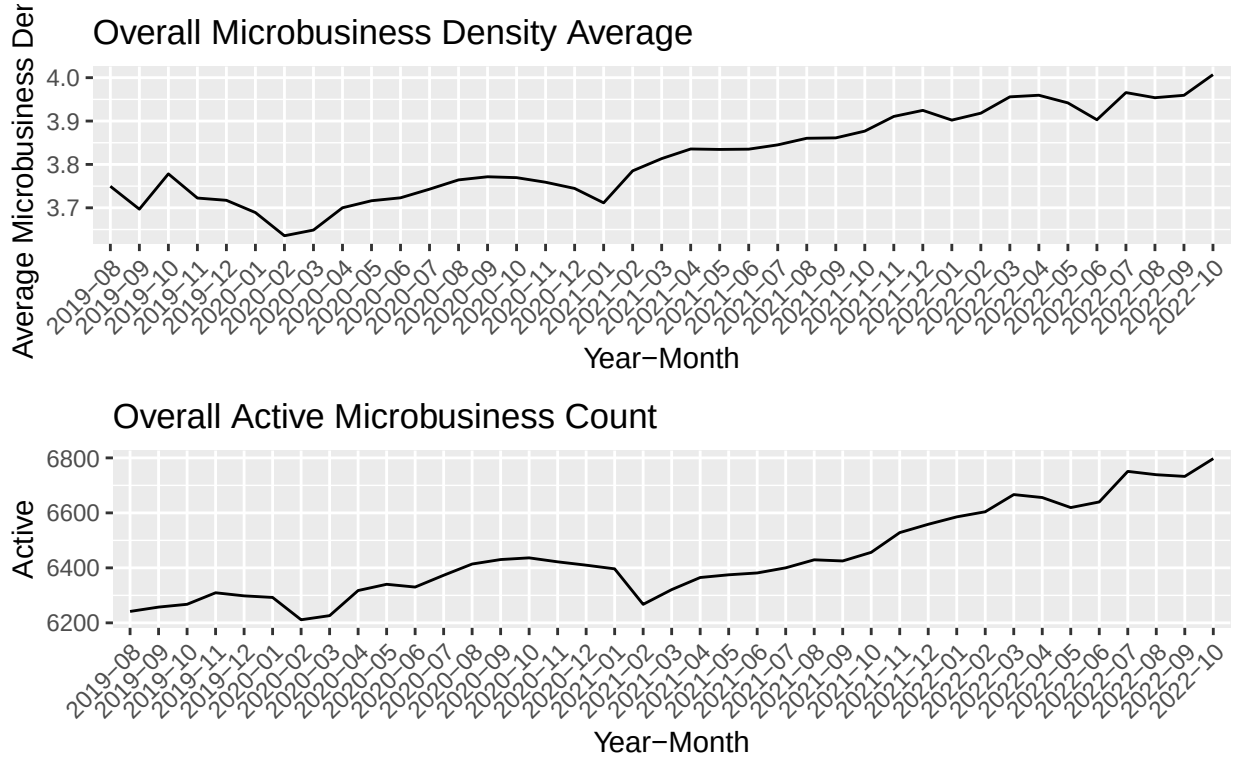
The main feature in this project is `microbusiness_density` provided in the **train_df.** Also, the number of active microbusinesses is provided in the `active` column.

### 3.7.1. Using Time Variables

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

```
# Create plots
p1 <- train_df %>%
  # Group train_df by year_month
  group_by(year_month) %>%
  # calculate the mean value of microbusiness_density for each group
  summarise(mean_microbusiness_density = mean(microbusiness_density)) %>%
  ggplot(aes(x = year_month, y = mean_microbusiness_density, group = 1)) +
  geom_line() +
  labs(title = "Overall Microbusiness Density Average",
       x = "Year-Month",
       y = "Average Microbusiness Density") +
```

```
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

p2 <- train_df %>%
  group_by(year_month) %>%
  summarize(avg_active = mean(active)) %>%
  ggplot(aes(x = year_month, y = avg_active)) +
  geom_line(group = 1) +
  labs(title = "Overall Active Microbusiness Count",
       x = "Year-Month",
       y = "Active") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


# Display the plots
grid.arrange(p1, p2, nrow = 2)
```

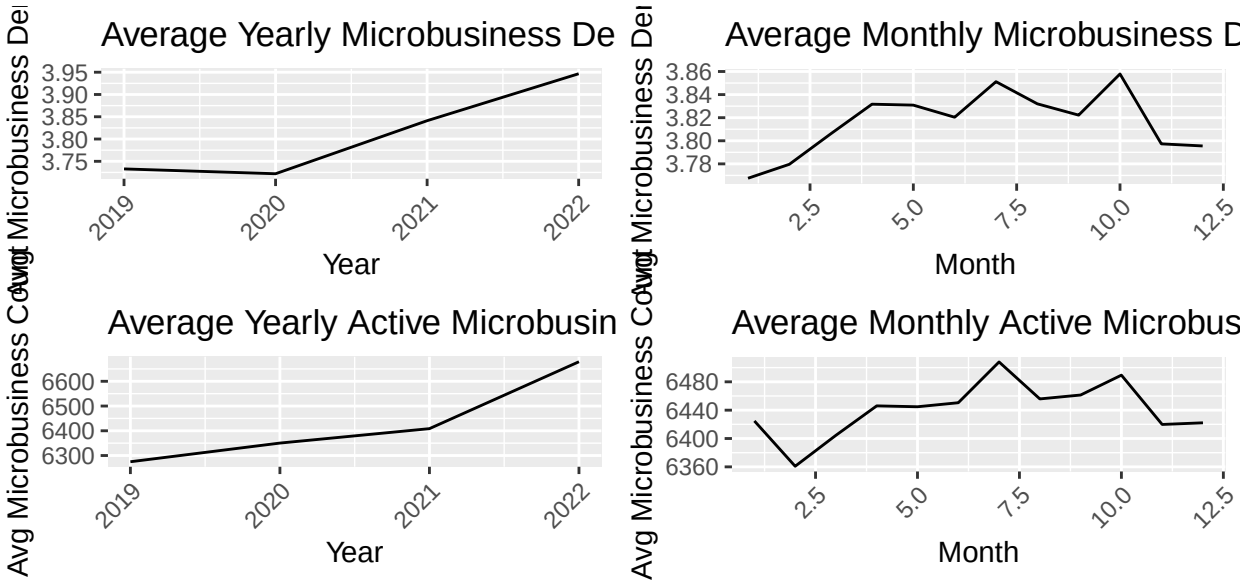



As expected, these two graphs show almost similar behavior. If we ignore the slight fluctuations of the two graphs, the general microbusiness density and count are growing over the whole time frame.

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

```
# Group train_df by year and calculate the mean value of microbusiness_density and active for each grou
train_df_mean_year <- train_df %>%
  group_by(year) %>%
  summarize(avg_microbusiness_density = mean(microbusiness_density),
            avg_active = mean(active))

# Group train_df by month and calculate the mean value of the microbusiness_density for each group
```

```r
train_df_mean_month <- train_df %>%
  group_by(month) %>%
  summarize(avg_microbusiness_density = mean(microbusiness_density),
            avg_active = mean(active))

# Plot the monthly mean values for microbusiness density
p1 <-
  ggplot(train_df_mean_month, aes(x = month, y = avg_microbusiness_density)) +
  geom_line() +
  ggtitle("Average Monthly Microbusiness Density") +
  xlab("Month") +
  ylab("Avg Microbusiness Density") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Plot the yearly mean values for microbusiness density
p2 <-
  ggplot(train_df_mean_year, aes(x = year, y = avg_microbusiness_density)) +
  geom_line() +
  ggtitle("Average Yearly Microbusiness Density") +
  xlab("Year") +
  ylab("Avg Microbusiness Density") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Plot the monthly mean values for active
p3 <-
  ggplot(train_df_mean_month, aes(x = month, y = avg_active)) +
  geom_line() +
  ggtitle("Average Monthly Active Microbusiness Count") +
  xlab("Month") +
  ylab("Avg Microbusiness Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Plot the yearly mean values for active
p4 <-
  ggplot(train_df_mean_year, aes(x = year, y = avg_active)) +
  geom_line() +
  ggtitle("Average Yearly Active Microbusiness Count") +
  xlab("Year") +
  ylab("Avg Microbusiness Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Display the plots side by side
grid.arrange(p2, p1, p4, p3, nrow = 2, ncol = 2)
```




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.

### 3.7.2. Using Geographical Measures

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.
   *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.*

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

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 **train_df** to lowercase letters. Merging two dataframes will cause problems because the data from `map_data()` will be in lowercase letters.

```
# Convert state and county columns in train_df to lowercase
train_df <- train_df %>%
  mutate(state = tolower(state)) %>%
  mutate(county = tolower(county))
```

Then, we'll create a new column in **train_df** named `region` and assign region values based on `state` column:

```
# Create a new column named region and initialize all values as NA
train_df$region <- NA

# Assign region values based on state column
for (i in 1:nrow(train_df)) {
  if (train_df$state[i] %in% c("connecticut", "maine", "massachusetts", "new hampshire", "rhode island"
    train_df$region[i] <- "new england"
  } else if (train_df$state[i] %in% c("delaware", "maryland", "new jersey", "new york", "pennsylvania",
    train_df$region[i] <- "mideast"
  } else if (train_df$state[i] %in% c("illinois", "indiana", "michigan", "ohio", "wisconsin")) {
    train_df$region[i] <- "great lakes"
  } else if (train_df$state[i] %in% c("iowa", "kansas", "minnesota", "missouri", "nebraska", "north dak
    train_df$region[i] <- "plains"
  } else if (train_df$state[i] %in% c("alabama", "arkansas", "florida", "georgia", "kentucky", "louisia
    train_df$region[i] <- "southeast"
  } else if (train_df$state[i] %in% c("arizona", "new mexico", "oklahoma", "texas")) {
    train_df$region[i] <- "southwest"
  } else if (train_df$state[i] %in% c("colorado", "idaho", "montana", "utah", "wyoming")) {
    train_df$region[i] <- "rocky mountain"
  } else if (train_df$state[i] %in% c("alaska", "california", "hawaii", "nevada", "oregon", "washington
    train_df$region[i] <- "far west"
  } else {
    train_df$region[i] <- "other"
  }
}
```

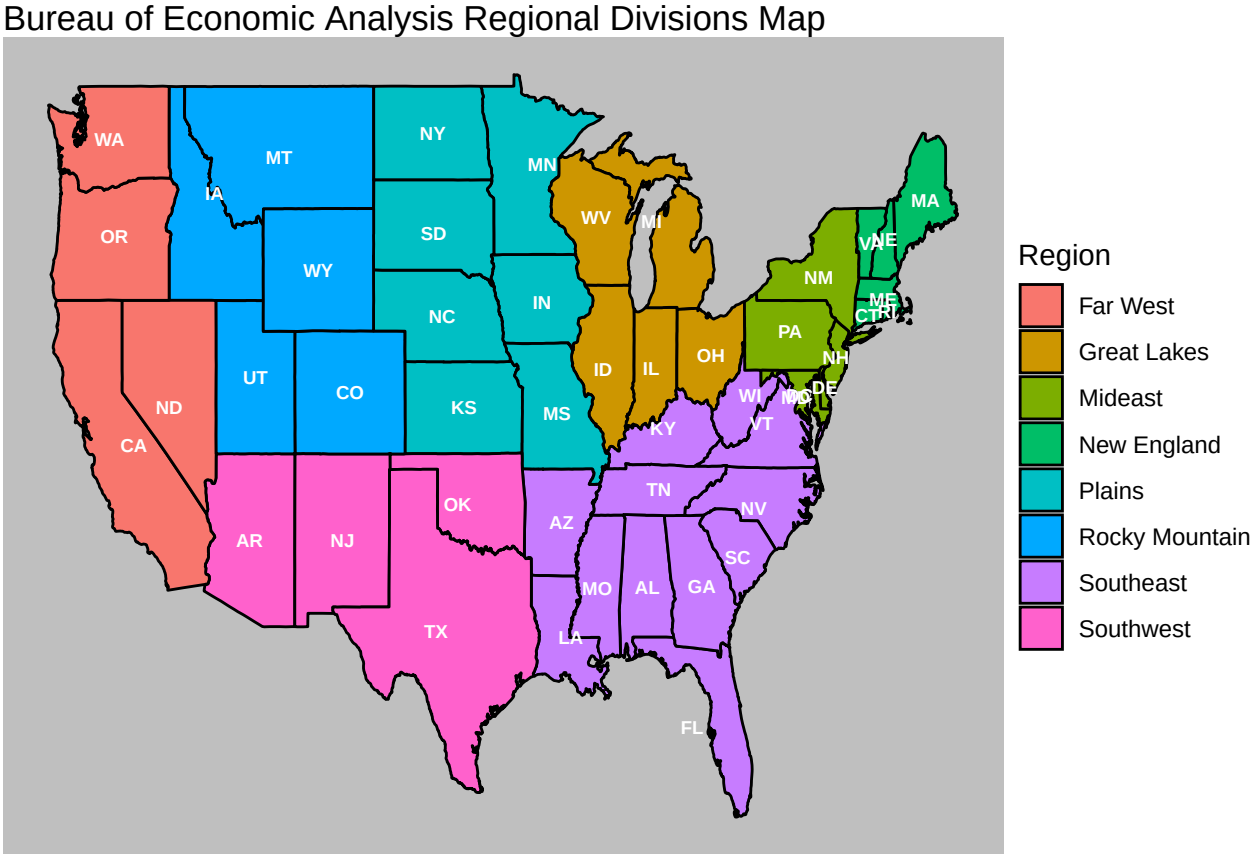
```
str(train_df$states)
```

```
##  NULL
```

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:

```
# Get the map of the United States
us_map <- map_data("state")

# Create a lookup table for state abbreviations and their corresponding full names
state_names <- data.frame(state = state.abb, name = tolower(state.name))

# Map the regions to the states
region_map <- us_map %>%
#left_join(state_names, by = c("region" = "state")) %>%
  left_join(state_names, by = c("region" = "name")) %>%
# merge(us_map, state_names, by.x=c("region"), by.y=c("name")) %>%
  mutate(region =
           ifelse(region %in% c("connecticut", "maine", "massachusetts", "new hampshire", "rhode island
                        ifelse(region %in% c("delaware", "maryland", "new jersey", "new york", "pennsy
                               ifelse(region %in% c("illinois", "indiana", "michigan", "ohio", "wiscon
                                      ifelse(region %in% c("iowa", "kansas", "minnesota", "missouri",
                                             ifelse(region %in% c("alabama", "arkansas", "florida", "g
                                                    ifelse(region %in% c("arizona", "new mexico", "okl
                                                           ifelse(region %in% c("colorado", "idaho", "
                                                                  ifelse(region %in% c("alaska", "cali
                                                                  )))))))))


# Summarize the data to get the center coordinates of each state
#state_centers <- region_map %>%
#  group_by(state) %>%
#  summarise(long = mean(long), lat = mean(lat))
# add labels
states <- aggregate(cbind(long, lat) ~ region, data=us_map,
                    FUN=function(x)mean(range(x)))
states$group <- c("AL", "AR", "AZ", "CA", "CO", "CT", "DE", "DC", "FL", "GA", "IA",
                  "ID", "IL", "IN", "KS", "KY", "LA", "MA", "MD", "ME", "MI", "MN",
                  "MO", "MS", "MT", "NC", "ND", "NE", "NH", "NJ", "NM", "NV", "NY",
                  "OH", "OK", "OR", "PA", "RI", "SC", "SD", "TN", "TX", "UT", "VA",
                  "VT", "WA", "WI", "WV", "WY")

# names(states)[names(states) == "region"] <- "group"


#Plot the map
ggplot(region_map, aes(x = long, y = lat, group = group, fill = region)) +
  geom_polygon(color = "black", show.legend = TRUE) +
#  geom_text(aes(label = state), data = region_map, size = 3, vjust = 2, hjust = 2) +
#  geom_text(aes(label = state), data = state_centers, size = 2, vjust = 2, hjust = 2) +
  geom_text(data = states, aes(long, lat, label = group), size = 2.5, inherit.aes = FALSE, color = "whi
#  scale_fill_gradient(low = "white", high = "darkred") +
```

```
#  scale_fill_manual(values = viridis(n = 60), na.value = "gray") +
  labs(title = "Bureau of Economic Analysis Regional Divisions Map", fill = "Region") +
#  geom_text(aes(x = long, y = lat, label = state), data = state_centers, size = 3, color = "white") +
  theme_void() +
  theme(panel.background = element_rect(fill = "gray75", color = NA))
```



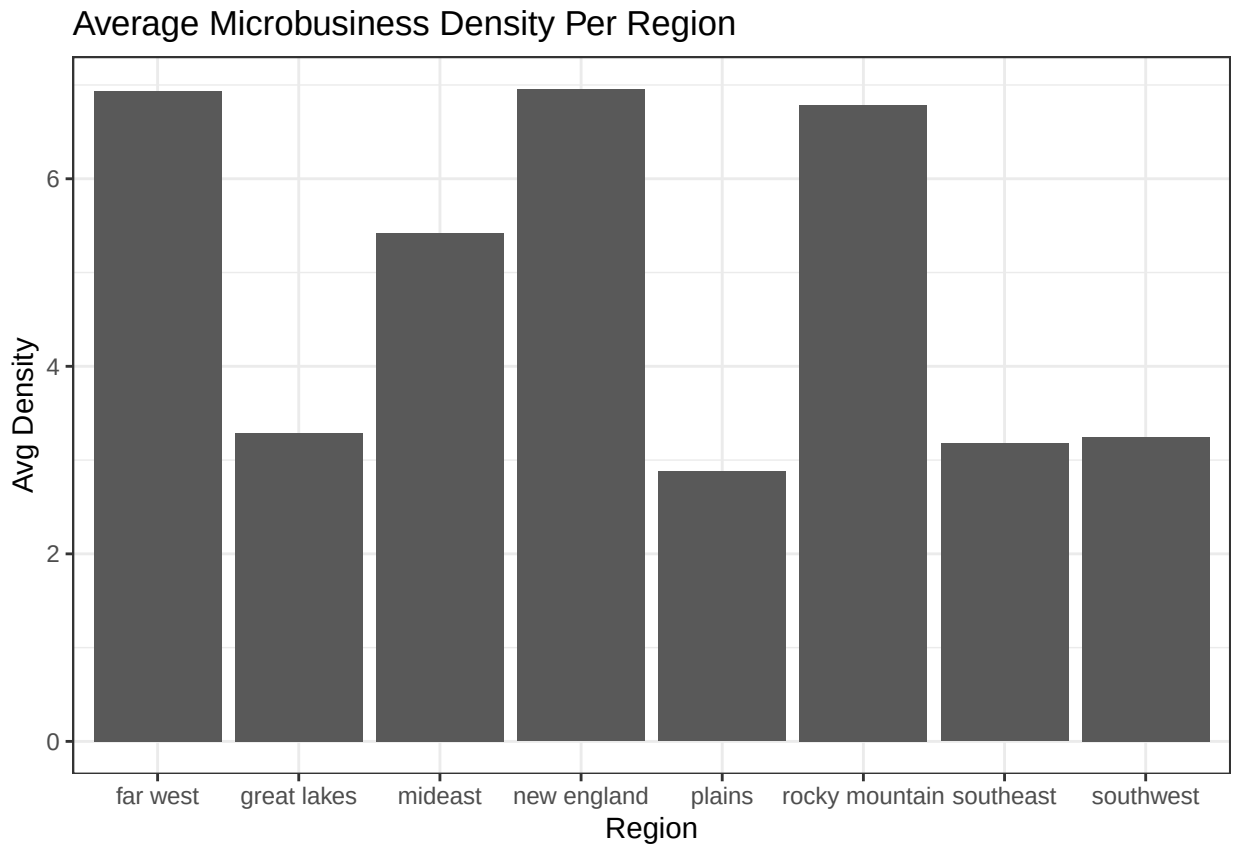

First, we need to convert state and county columns in `train_df` to lowercase letters. Because, the data from map_data() will be in lowercase and when merging two dataframes it might cause problems.

```
# Convert state and county columns in train_df to lowercase
train_df <- train_df %>%
  mutate(state = tolower(state)) %>%
  mutate(county = tolower(county))
```

```
# Print all the unique values in the region column
unique(train_df$region)
```

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

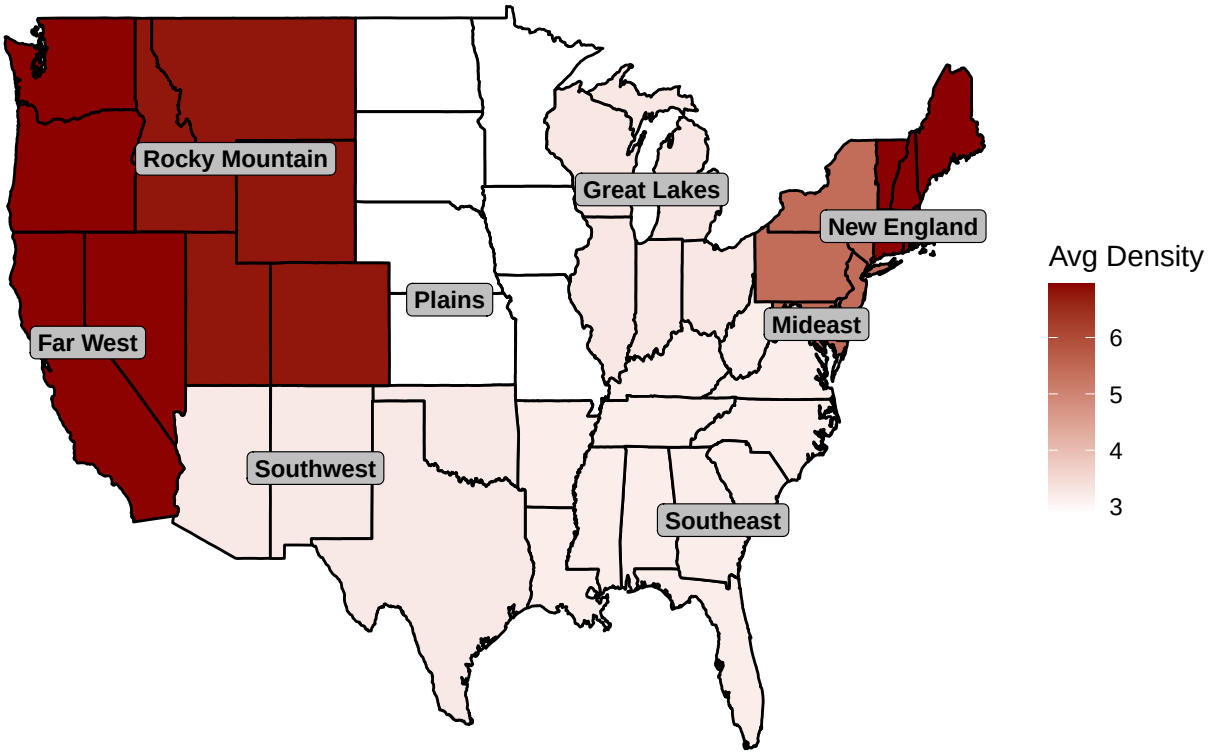
```r
# Group train_df by region and calculate average microbusiness density
train_df %>%
  group_by(region) %>%
  summarize(avg_density = mean(microbusiness_density)) %>%

  # Create bar plot of average density by region
  ggplot(aes(x = region, y = avg_density)) +
  geom_bar(stat = "identity") +

  # Add plot title and axis labels
  labs(title = "Average Microbusiness Density Per Region",
       x = "Region", y = "Avg Density") +

  # Apply a black and white theme to the plot
  theme_bw()
```



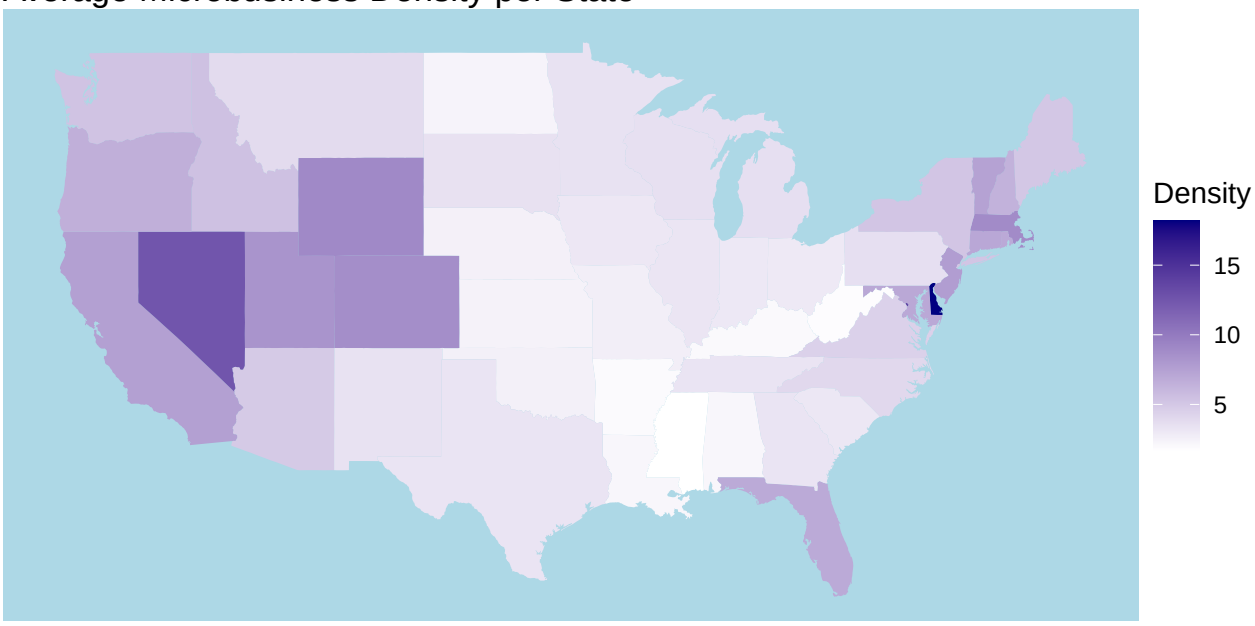

```r
# Group train_df by region and calculate average microbusiness density
avg_density <- train_df %>%
  group_by(region) %>%
  summarize(avg_density = mean(microbusiness_density))

# Create a lookup table for state abbreviations and their corresponding full names
state_names <- data.frame(state = state.abb, name = tolower(state.name))

# Lowercase region column of region_map
```

```r
region_map <- region_map %>%
  mutate(region = tolower(region))

# Merge the average density data with the region_map data
plot_data <- merge(region_map, avg_density, by = "region") %>%
  arrange(order)

# Coordinates of the center of regions
bea_regions <- data.frame(
  group = c("New England", "Mideast", "Great Lakes", "Plains",
            "Southeast", "Southwest", "Rocky Mountain", "Far West"),
  x = c(-71.8, -76.9, -86.6, -98.5, -82.4, -106.4, -111.1, -119.8),
  y = c(42.2, 39, 43.4, 39.8, 32.6, 34.3, 44.4, 38.4)
)

# Create the plot
ggplot(plot_data, aes(x = long, y = lat, group = group, fill = avg_density)) +
  geom_polygon(color = "black") +
  geom_label(data = bea_regions,
             aes(x = x, y = y, label = group),
             size = 3, fontface = "bold",
             label.padding = unit(0.2, "lines"),
             label.size = 0.2,
             fill = "gray75", color = "black") +
  scale_fill_gradient(low = "white", high = "darkred") +
  # scale_fill_viridis(name = "Avg Density", na.value = "gray") +
  labs(title = "Average Microbusiness Density Per Region", fill = "Avg Density") +
  theme_void()
```

Average Microbusiness Density Per Region

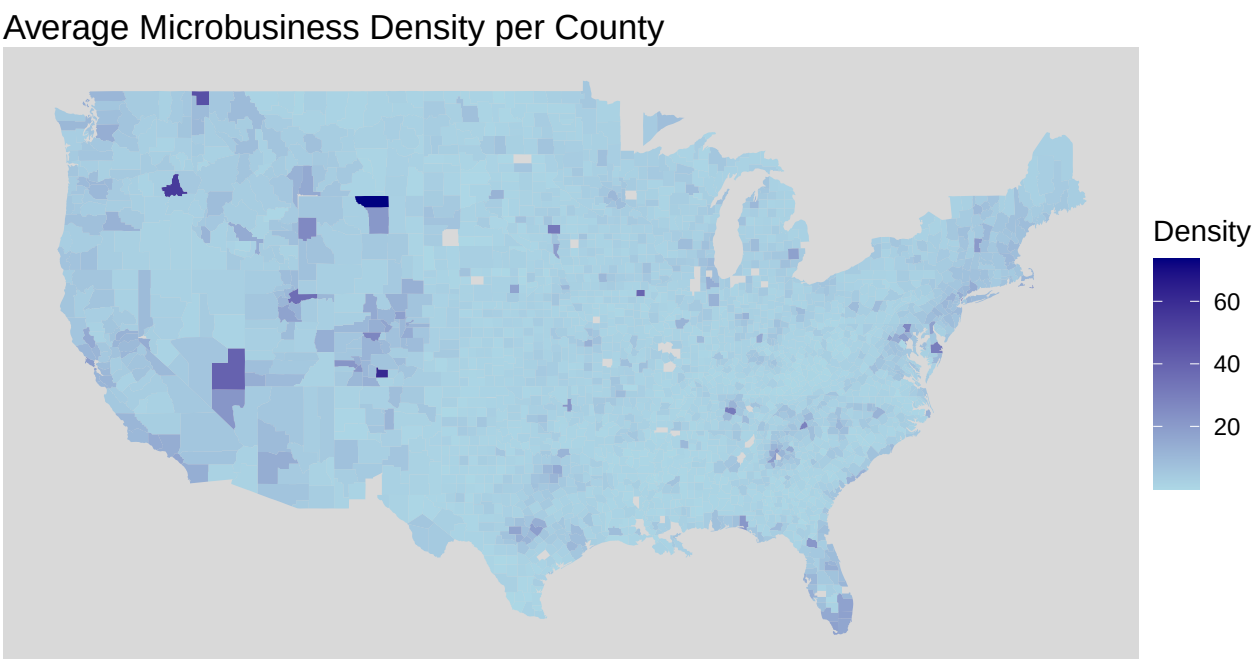



```
# Aggregate microbusiness density by state
state_avg <- aggregate(microbusiness_density ~ state, data = train_df, FUN = mean)

# Load US map data
us_map <- map_data("state")

# Merge state_avg with us_map based on region and state
map_data <- merge(us_map, state_avg, by.x = "region", by.y = "state")

# Create a heatmap of microbusiness density by state
ggplot(map_data, aes(x = long, y = lat, group = group, fill = microbusiness_density)) +
  geom_polygon() +
  scale_fill_gradient(low = "white", high = "navyblue") +
  coord_map() +
  labs(title = "Average Microbusiness Density per State", fill = "Density") +
  theme_void() +
  theme(panel.background = element_rect(fill = "lightblue", color = NA))
```




```
#{r fig.width = 10 ,fig.height = 12, out.width='100%', fig.align='center'}
# Aggregate microbusiness density by county

#county_avg <- train_df %>%
#  group_by(cfips, county) %>%
#  summarise(microbusiness_density = mean(microbusiness_density))

county_avg <- aggregate(microbusiness_density ~ county + state, data = train_df, FUN = mean)
county_avg$county <- gsub(" county", "", county_avg$county)
county_avg$county <- gsub(" city", "", county_avg$county)
county_avg$county <- gsub(" parish", "", county_avg$county)


# Load US county map data
us_map <- map_data("county")

# Merge county_avg with us_map based on region and county
map_data <- merge(us_map, county_avg, by.x = c("subregion", "region"), by.y = c("county", "state")) %>%
  arrange(order)

# Create a heatmap of microbusiness density by county using ggplot2
ggplot(map_data, aes(x = long, y = lat, group = group, fill = microbusiness_density)) +
  geom_polygon() +
  scale_fill_gradient(low = "lightblue", high = "navyblue") +
  coord_map() +
  labs(title = "Average Microbusiness Density per County", fill = "Density") +
```

```
  theme_void() +
  theme(panel.background = element_rect(fill = "gray85", color = NA))
```



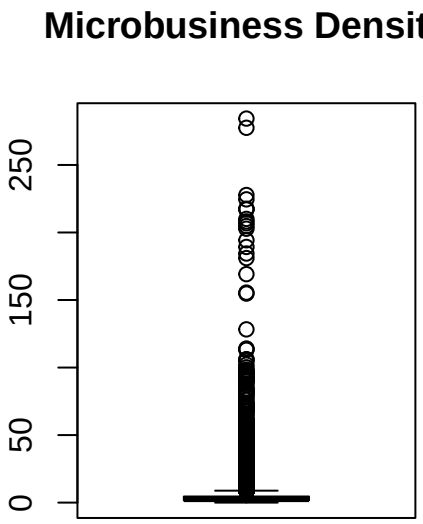

```
str(merged_df)
```

```
## 'data.frame':    122265 obs. of  15 variables:
##  $ row_id               : chr  "1001_2019-08-01" "1001_2019-09-01" "1001_2019-10-01" "1001_2019-11-0
##  $ 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   : chr  "2019-08-01" "2019-09-01" "2019-10-01" "2019-11-01" ...
##  $ 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 ...
##  - 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" ...
```

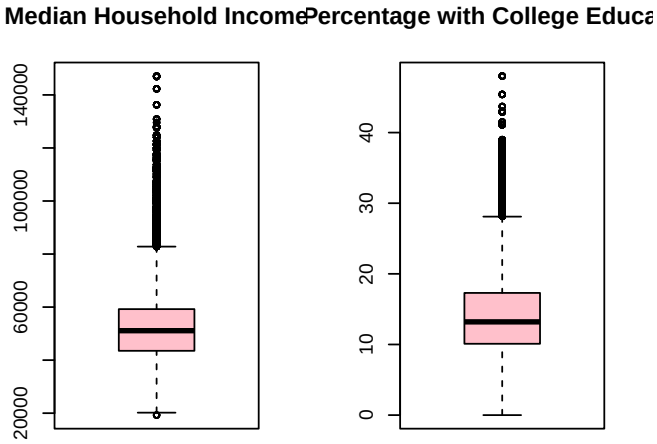
Boxplots are a visualization tool that provide insights into the central tendency and spread of a dataset,

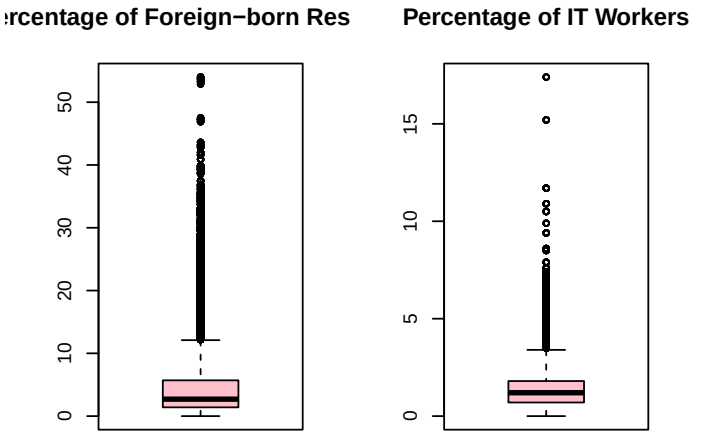
as well as identify outliers and skewness. They are useful for detecting anomalies and comparing variable distributions in a dataset, providing valuable insights into data distribution for exploratory data analysis.

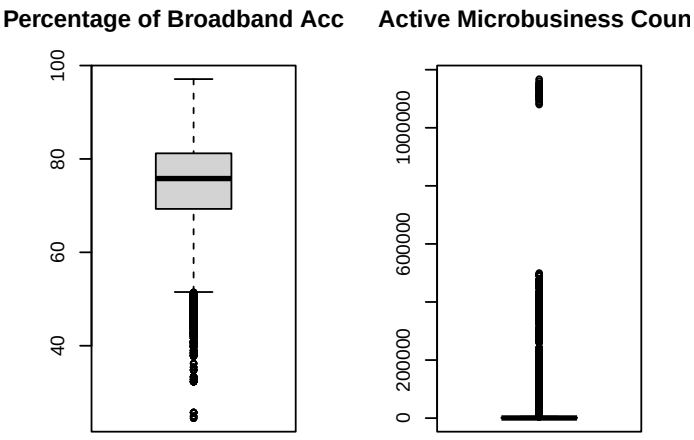
```
boxplot(merged_df$microbusiness_density, col = "blue", main = "Microbusiness Density")
```




```
par(mfrow=c(3,2)) # set plot layout to 3 rows and 2 columns
boxplot(merged_df$median_hh_inc, col = "pink", main = "Median Household Income")
boxplot(merged_df$pct_college, col = "pink", main = "Percentage with College Education")
boxplot(merged_df$pct_foreign_born, col = "pink", main = "Percentage of Foreign-born Residents")
boxplot(merged_df$pct_it_workers, col = "pink", main = "Percentage of IT Workers")
boxplot(merged_df$pct_bb, color = "pink", main = "Percentage of Broadband Access")
boxplot(merged_df$active, color = "pink", main = "Active Microbusiness Count")
```


Median Household Income
20000
60000
100000
140000
Percentage with College Educa
0
10
20
30
40


rcentage of Foreign−born Res
0
10
20
30
40
50
Percentage of IT Workers
0
5
10
15


Percentage of Broadband Acc
40
60
80
100
Active Microbusiness Coun
0
200000
600000
1000000

## 3.8. Outlier Detection

Outlier detection is an important step in data analysis, as outliers can significantly affect the results of statistical analyses. One method for outlier detection is the decision range approach.

The decision range approach involves setting a range of values outside of which any observations are considered outliers. The decision range is determined based on the data distribution and the researcher's judgment. One common approach 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.

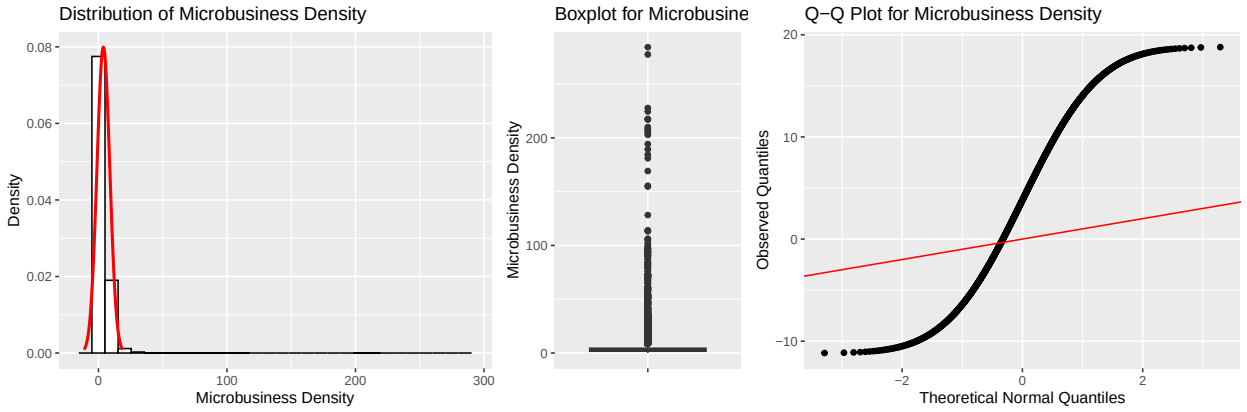
The decision range is then defined as the range from Q1 - 1.5$IQR$ *to* $Q3 + 1.5$IQR. Any observations that fall outside of this range are considered outliers. This method is useful for identifying potential outliers in a dataset and can help to ensure that statistical analyses are robust and accurate.

```
quartiles <- quantile(merged_df$microbusiness_density, probs = seq(0, 1, 0.25), na.rm = FALSE,
         names = TRUE, type = 7, digits = 6)
quartiles
```

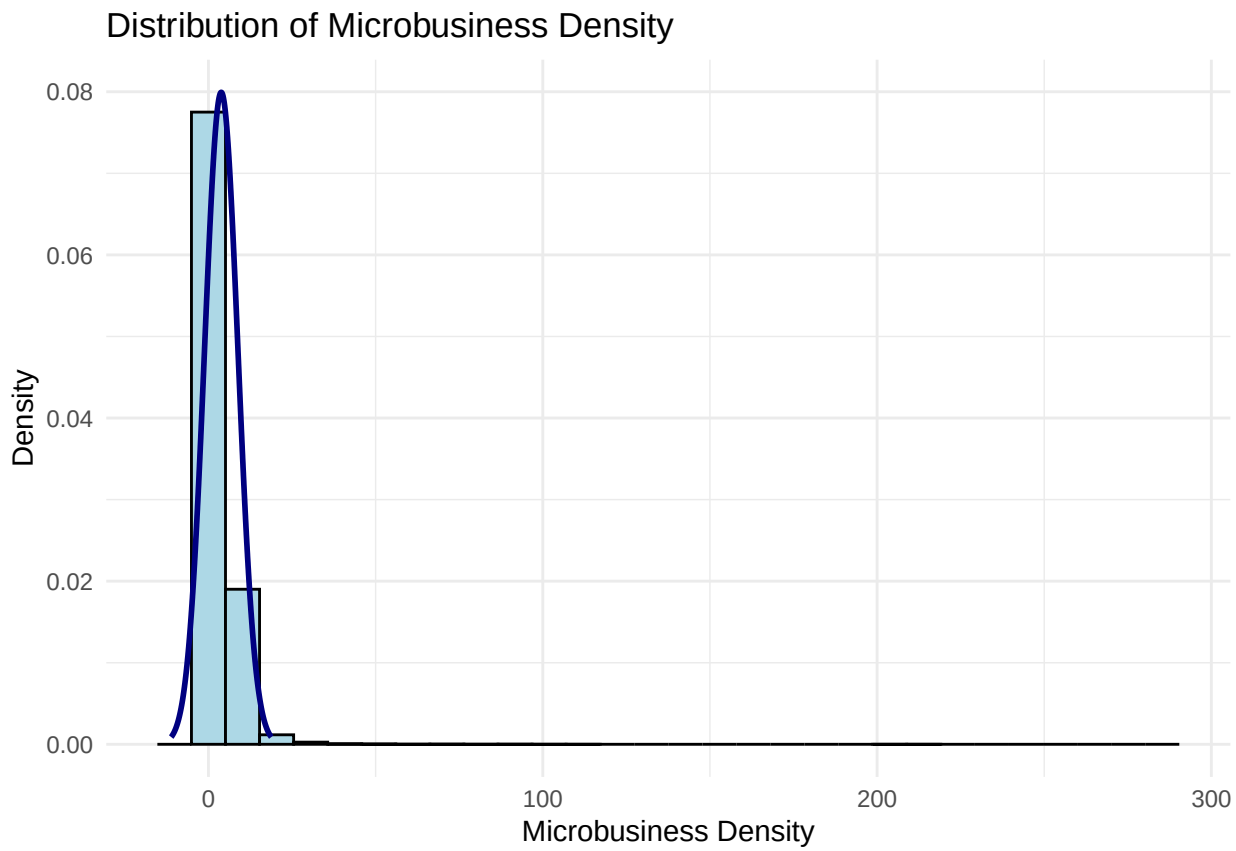
```
##         0%        25%        50%        75%       100%
##   0.000000   1.639344   2.586543   4.519231 284.340030
```

```
# Calculate the mean and standard deviation of 'microbusiness_density'
mean_density <- mean(merged_df$microbusiness_density)
sd_density <- sd(merged_df$microbusiness_density)

# Create a range of values for the x-axis
x_values <- seq(mean_density - 3*sd_density, mean_density + 3*sd_density, length.out = 1000)

# Create a bell curve using the 'dnorm' function with mean and standard deviation calculated above
y_values <- dnorm(x_values, mean = mean_density, sd = sd_density)

# Combine the 'x_values' and 'y_values' into a data frame using the 'data.frame' function
density_df <- data.frame(x = x_values, y = y_values)

# Create a boxplot using the 'ggplot' function from 'ggplot2' package
boxplot <- ggplot(data = merged_df, aes(x = "", y = merged_df$microbusiness_density)) +
  geom_boxplot(fill = "skyblue") +
  labs(x = "", y = "Microbusiness Density") +
  ggtitle("Boxplot for Microbusiness Density")

# Create a Q-Q plot using the 'ggplot' function from 'ggplot2' package and add a diagonal line using th
qqplot <- ggplot(data = density_df, aes(sample = x)) +
  geom_qq() +
  geom_abline(intercept = 0, slope = 1, colour = "red") +
  labs(x = "Theoretical Normal Quantiles", y = "Observed Quantiles") +
  ggtitle("Q-Q Plot for Microbusiness Density")

# Create a bell curve using the 'ggplot' function from 'ggplot2' package and add the bell curve using t
densityplot <- ggplot(data = merged_df, aes(x = microbusiness_density)) +
  geom_histogram(aes(y = after_stat(density)), bins = 30, colour = "black", fill = "white") +
  geom_line(data = density_df, aes(x = x, y = y), colour = "red", linewidth = 1) +
  labs(x = "Microbusiness Density", y = "Density") +
```

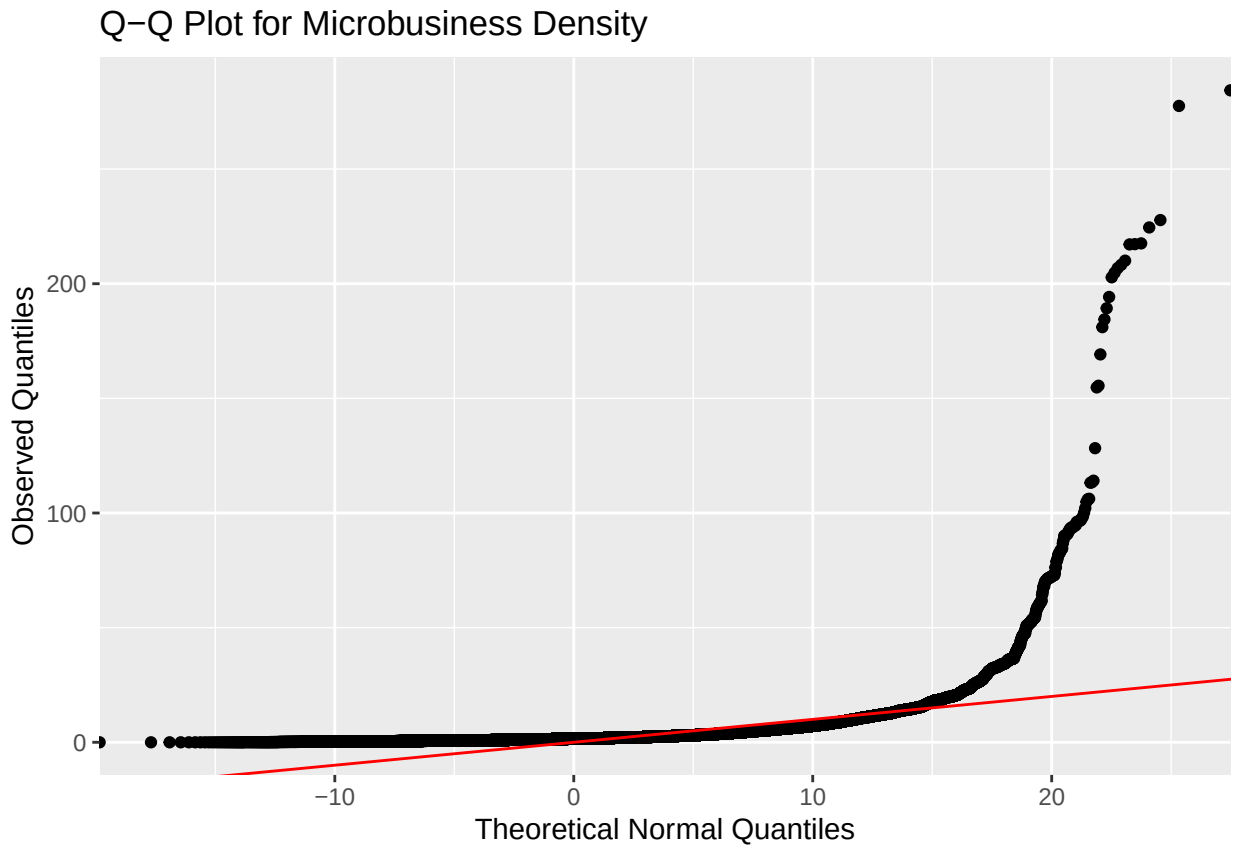
```
  ggtitle("Distribution of Microbusiness Density")

# Arrange the plots in one row using the 'grid.arrange' function from the 'gridExtra' package
grid.arrange(densityplot, boxplot, qqplot, ncol = 3, widths = c(2, 1, 2))
```



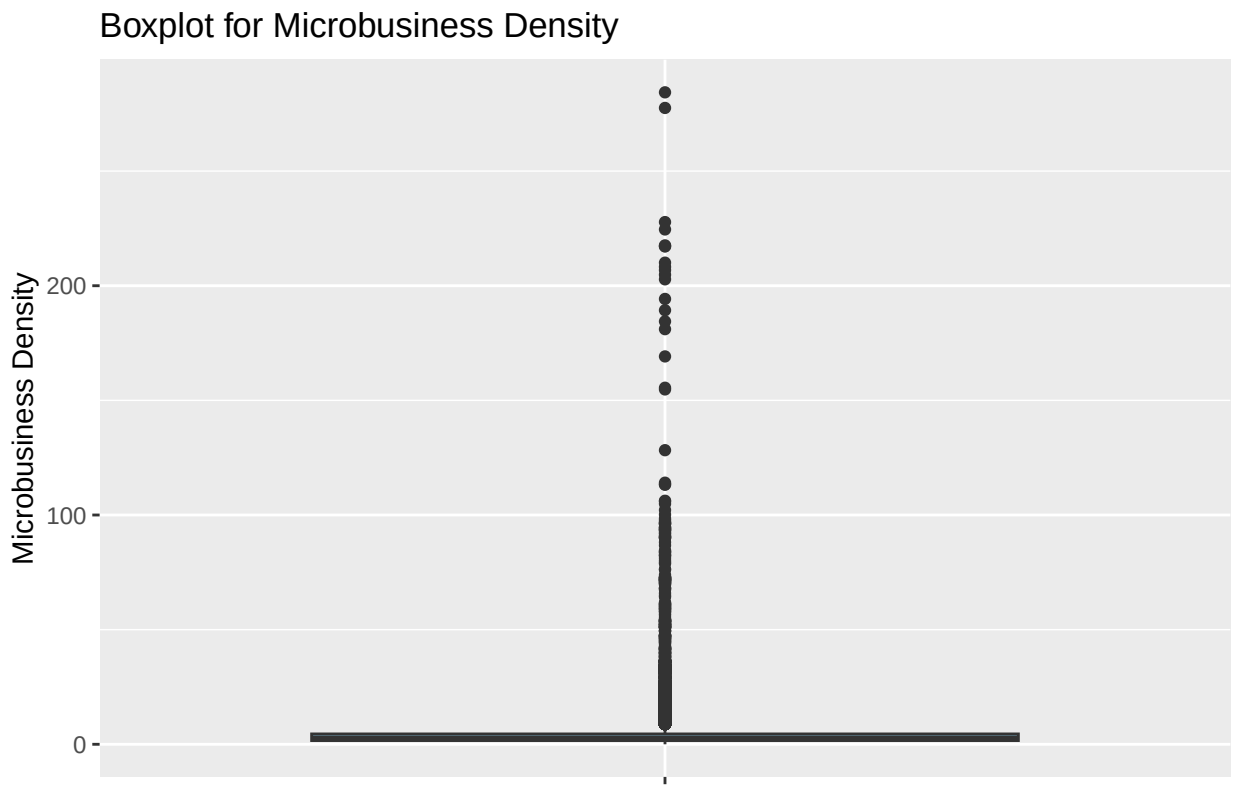

```
par(mfrow=c(1,3)) # set plot layout to 3 rows and 2 columns
mean_density <- mean(merged_df$microbusiness_density)
sd_density <- sd(merged_df$microbusiness_density)
x_values <- seq(mean_density - 3*sd_density, mean_density + 3*sd_density, length.out = 1000)
y_values <- dnorm(x_values, mean = mean_density, sd = sd_density)
density_df <- data.frame(x = x_values, y = y_values)
ggplot(merged_df, aes(x = microbusiness_density)) +
  geom_histogram(aes(y = ..density..), bins = 30, colour = "black", fill = "lightblue") +
  geom_line(data = density_df, aes(x = x, y = y), colour = "navy", linewidth = 1) +
  labs(x = "Microbusiness Density", y = "Density") +
  ggtitle("Distribution of Microbusiness Density") +
  theme_minimal()
```



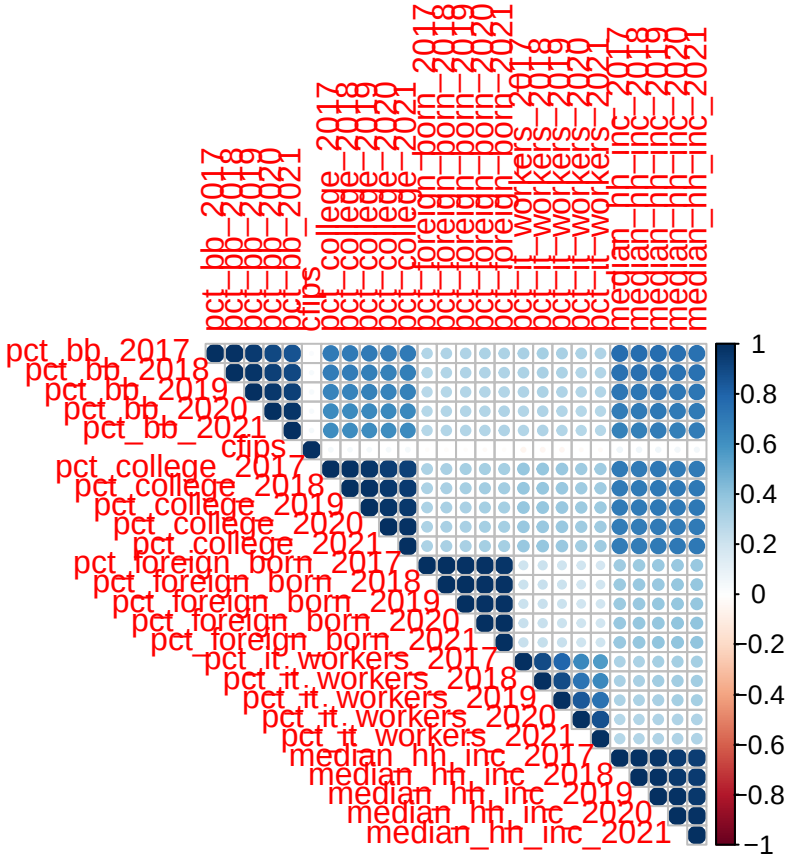

```
theoretical_norm <- qnorm(p = seq(0, 1, length.out = length(merged_df$microbusiness_density)), mean = me
density_df <- data.frame(observed = sort(merged_df$microbusiness_density), theoretical = theoretical_no
ggplot(density_df, aes(x = theoretical, y = observed)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, colour = "red") +
  labs(x = "Theoretical Normal Quantiles", y = "Observed Quantiles") +
  ggtitle("Q-Q Plot for Microbusiness Density")
```



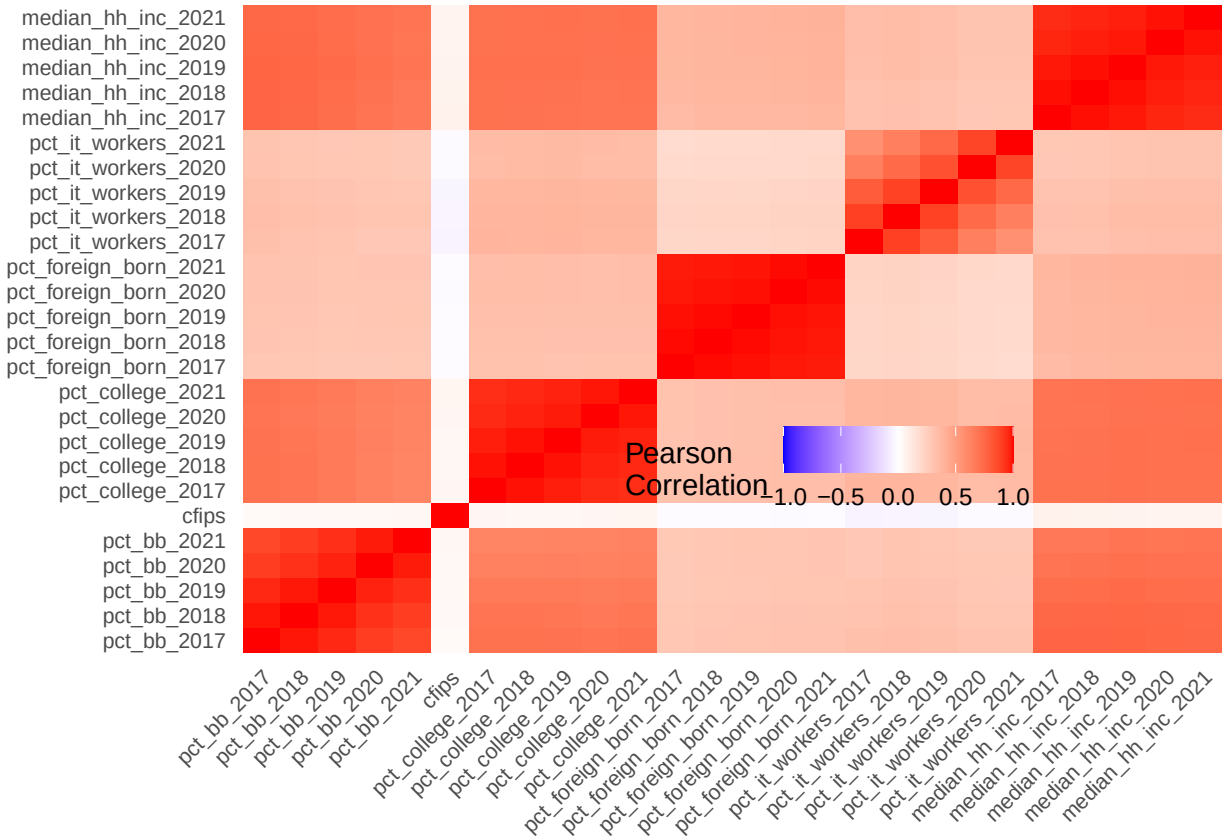

```
ggplot(data = merged_df, aes(x = "", y = microbusiness_density)) +
  geom_boxplot(fill = "skyblue") +
  labs(x = "", y = "Microbusiness Density") +
  ggtitle("Boxplot for Microbusiness Density")
```




```r
# draft (this is a test plot)

# compute the correlation matrix
corr_matrix <- cor(imputed_data)

# visualize the correlation matrix using corrplot
corrplot(corr_matrix, type = "upper", method = "circle")
```




```
# draft (this is a test plot)

# Calculate the correlation matrix
cor_mat <- cor(imputed_data)

# Melt the correlation matrix to a long format
melted_cor <- melt(cor_mat)

# Generate the heatmap
ggplot(melted_cor, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 8, hjust = 1),
        axis.text.y = element_text(size = 8),
        axis.title = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.justification = c(1, 0),
        legend.position = c(0.8, 0.2),
        legend.direction = "horizontal")
```

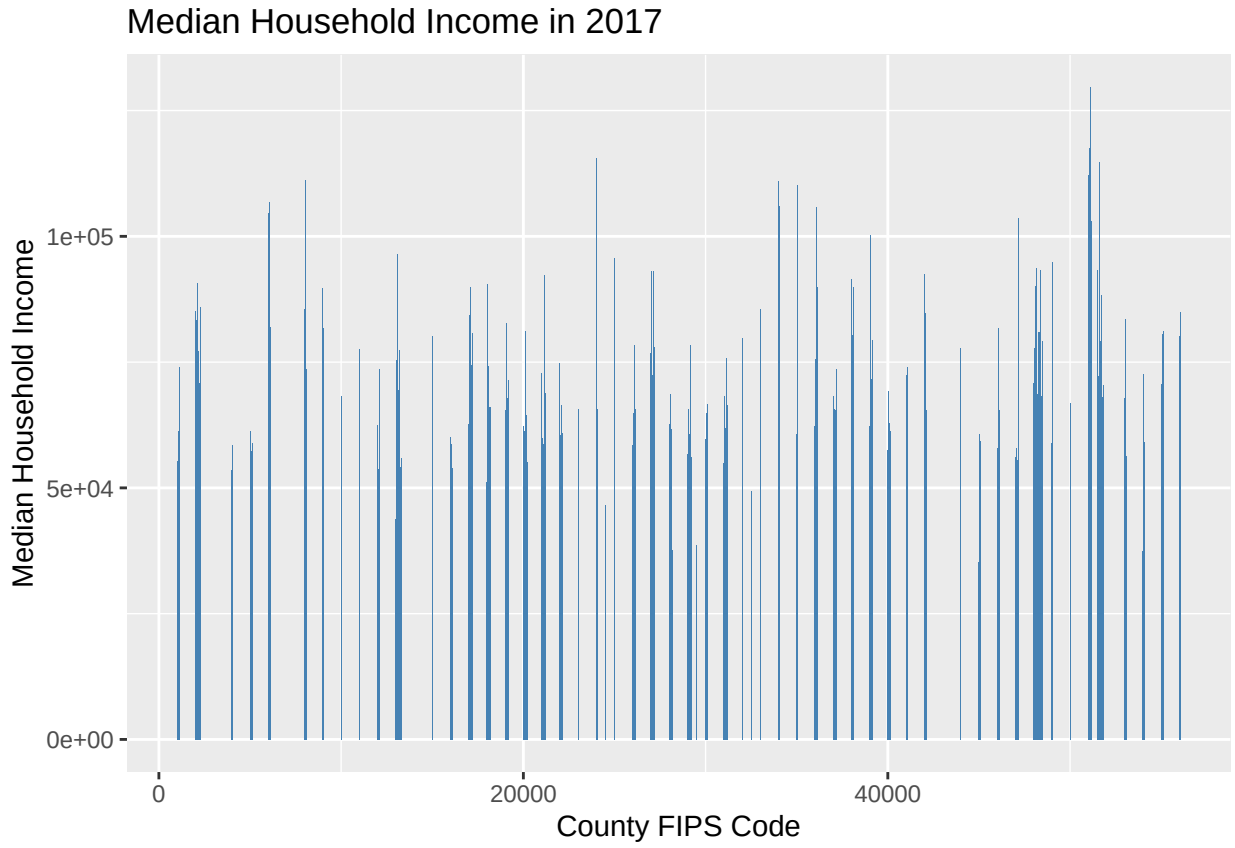



```
str(imputed_data)
```

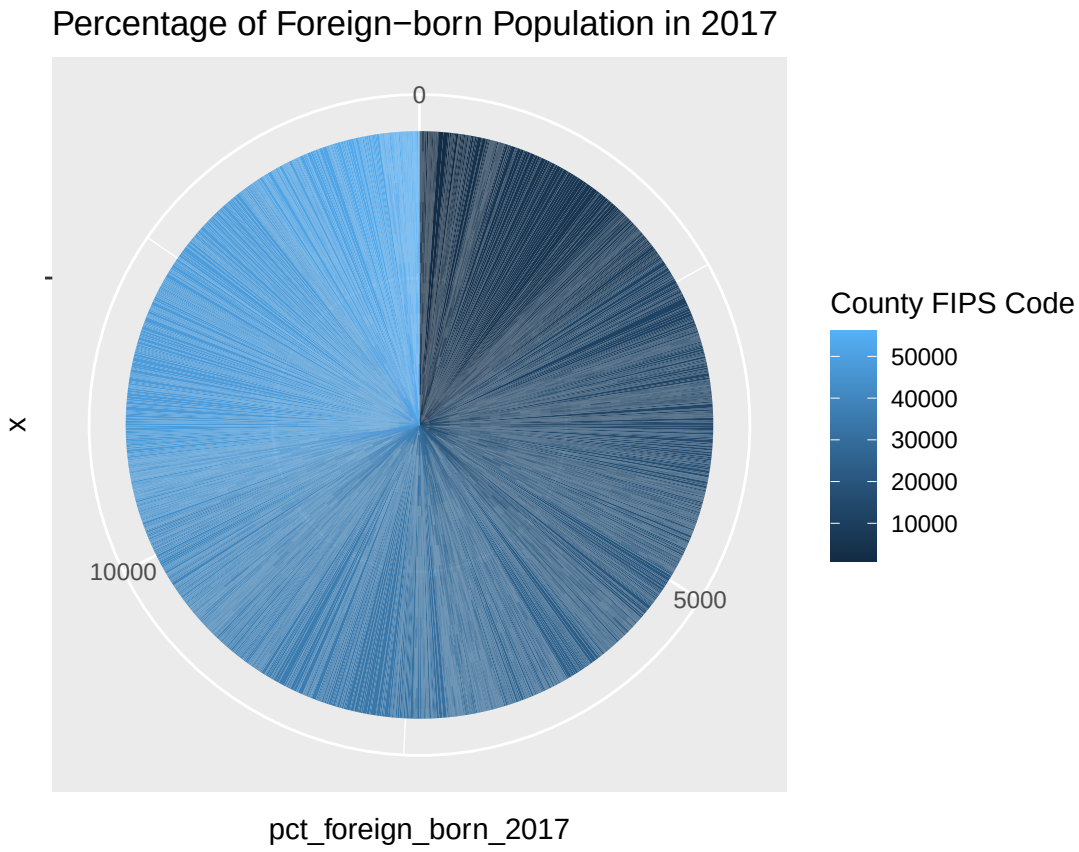
```
## 'data.frame':    3142 obs. of  26 variables:
##  $ pct_bb_2017          : num  76.6 74.5 57.2 62 65.8 49.4 58.2 71 62.8 67.5 ...
##  $ pct_bb_2018          : num  78.9 78.1 60.4 66.1 68.5 58.9 62.1 73 66.5 68.6 ...
##  $ pct_bb_2019          : num  80.6 81.8 60.5 69.2 73 60.1 64.6 75.1 69.4 70.7 ...
##  $ pct_bb_2020          : num  82.7 85.1 64.6 76.1 79.6 60.6 73.6 79.8 74.5 75 ...
##  $ pct_bb_2021          : num  85.5 87.9 64.6 74.6 81 59.4 76.3 81.6 77.1 76.7 ...
##  $ cfips                : int  1001 1003 1005 1007 1009 1011 1013 1015 1017 1019 ...
##  $ pct_college_2017     : num  14.5 20.4 7.6 8.1 8.7 6.6 9.6 10.2 9 6.6 ...
##  $ pct_college_2018     : num  15.9 20.7 7.8 7.6 8.1 7.4 9.7 10.2 9.3 6.8 ...
##  $ pct_college_2019     : num  16.1 21 7.6 6.5 8.6 7.4 9.7 10.5 9.5 6.6 ...
##  $ pct_college_2020     : num  16.7 20.2 7.3 7.4 8.9 6.1 10.1 10.5 10.5 6.3 ...
##  $ pct_college_2021     : num  16.4 20.6 6.7 7.9 9.3 8.1 8.1 11.4 9.6 6.2 ...
##  $ pct_foreign_born_2017: num  2.1 3.2 2.7 1 4.5 1.8 1 2.6 1.3 0.7 ...
##  $ pct_foreign_born_2018: num  2 3.4 2.5 1.4 4.4 0.9 1.4 2.7 1.4 0.8 ...
##  $ pct_foreign_born_2019: num  2.3 3.7 2.7 1.5 4.5 0.7 0.8 2.7 1.8 0.9 ...
##  $ pct_foreign_born_2020: num  2.3 3.4 2.6 1.6 4.4 1.5 1.9 2.5 1.9 1.9 ...
##  $ pct_foreign_born_2021: num  2.1 3.5 2.6 1.1 4.5 1.2 1.7 2.5 2 2 ...
##  $ pct_it_workers_2017  : num  1.3 1.4 0.5 1.2 1.3 0.4 1.1 1.4 2.4 1.4 ...
##  $ pct_it_workers_2018  : num  1.1 1.3 0.3 1.4 1.4 0.3 1.4 1.4 2.1 1.3 ...
##  $ pct_it_workers_2019  : num  0.7 1.4 0.8 1.6 0.9 0.5 1.7 1.2 2.1 1.2 ...
##  $ pct_it_workers_2020  : num  0.6 1 1.1 1.7 1.1 0.3 1.3 1 2.3 0.9 ...
##  $ pct_it_workers_2021  : num  1.1 1.3 0.8 2.1 0.9 0.2 1.4 1 1.8 0.4 ...
##  $ median_hh_inc_2017   : int  55317 52562 33368 43404 47412 29655 36326 43686 37342 40041 ...
```

```
##  $ median_hh_inc_2018   : num  58786 55962 34186 45340 48695 ...
##  $ median_hh_inc_2019   : int  58731 58320 32525 47542 49358 37785 40688 47255 42289 41919 ...
##  $ median_hh_inc_2020   : num  57982 61756 34990 51721 48922 ...
##  $ median_hh_inc_2021   : num  62660 64346 36422 54277 52830 ...
```

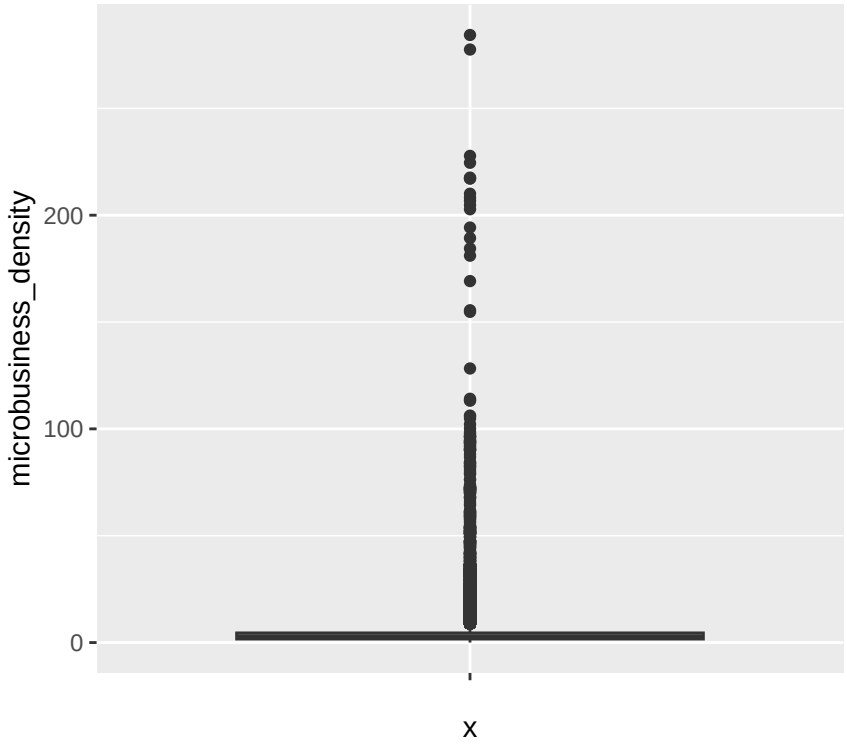
```
# draft (this is a test plot)

# Bar Plot
ggplot(imputed_data, aes(x=cfips, y=median_hh_inc_2017)) +
  geom_bar(stat="identity", fill="steelblue") +
  labs(title = "Median Household Income in 2017",
       x = "County FIPS Code", y = "Median Household Income")
```



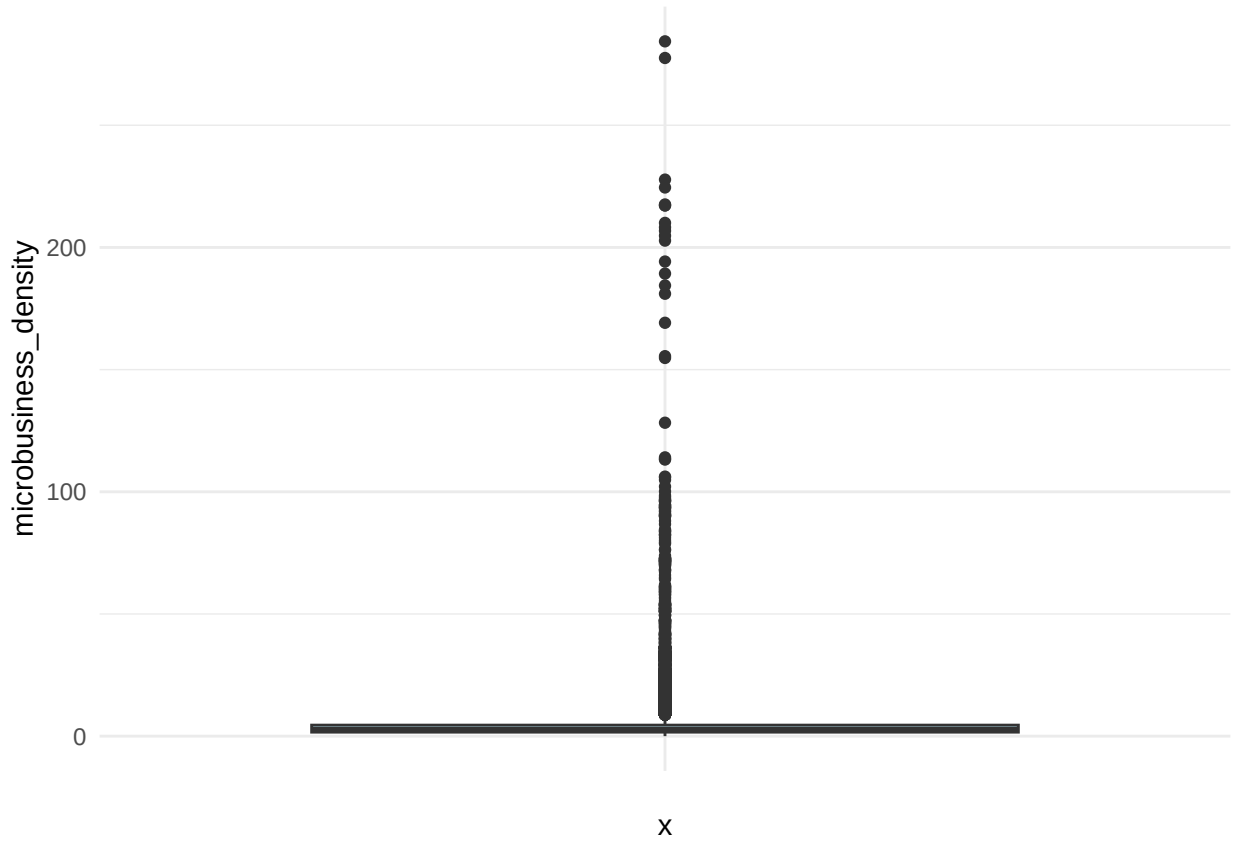

```
# draft (this is a test plot)
# Pie Chart
ggplot(imputed_data, aes(x="", y=pct_foreign_born_2017, fill=cfips)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start=0) +
  labs(title = "Percentage of Foreign-born Population in 2017",
       fill = "County FIPS Code")
```



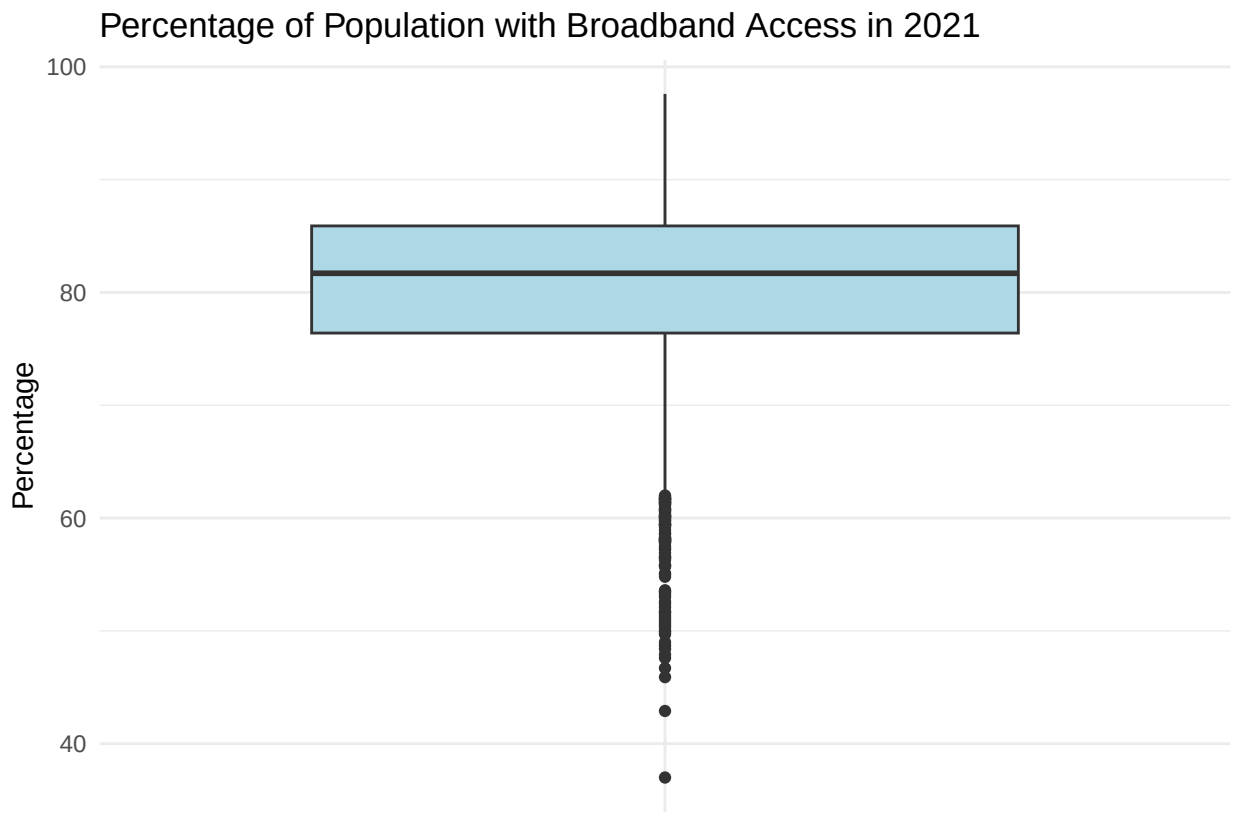

```
# draft (this is a test plot)
#creating a boxplot
ggplot(merged_df, aes("", microbusiness_density)) +
  geom_boxplot()+scale_fill_manual(values = c("0" = "#009999","1" = "#CC3300"))
```



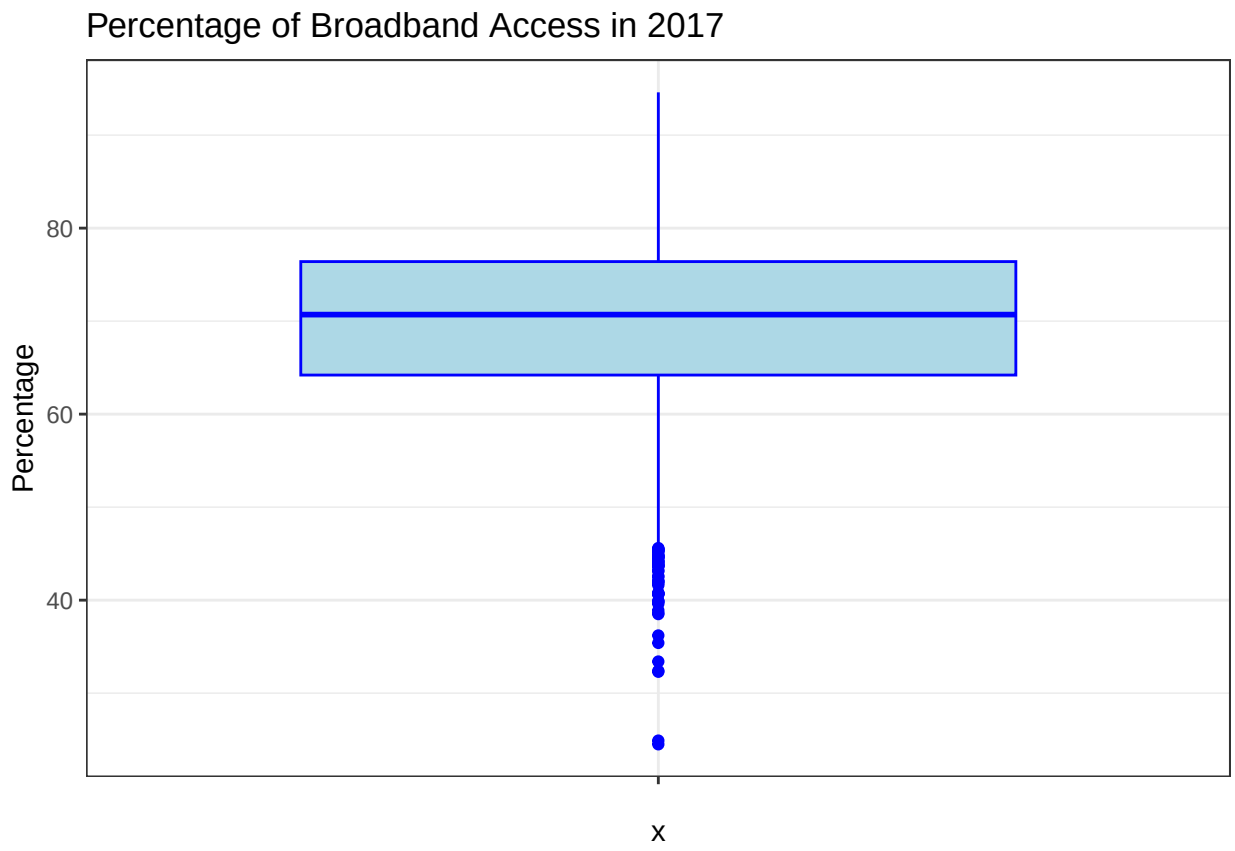

```
# draft (this is a test plot)

# Create the plot
ggplot(merged_df, aes(x = "", y = microbusiness_density)) +
  geom_boxplot(fill = "lightblue") +
  #labs(title = "doesn;t matter",
  #     x = NULL, y = NULL) +
  theme_minimal()
```



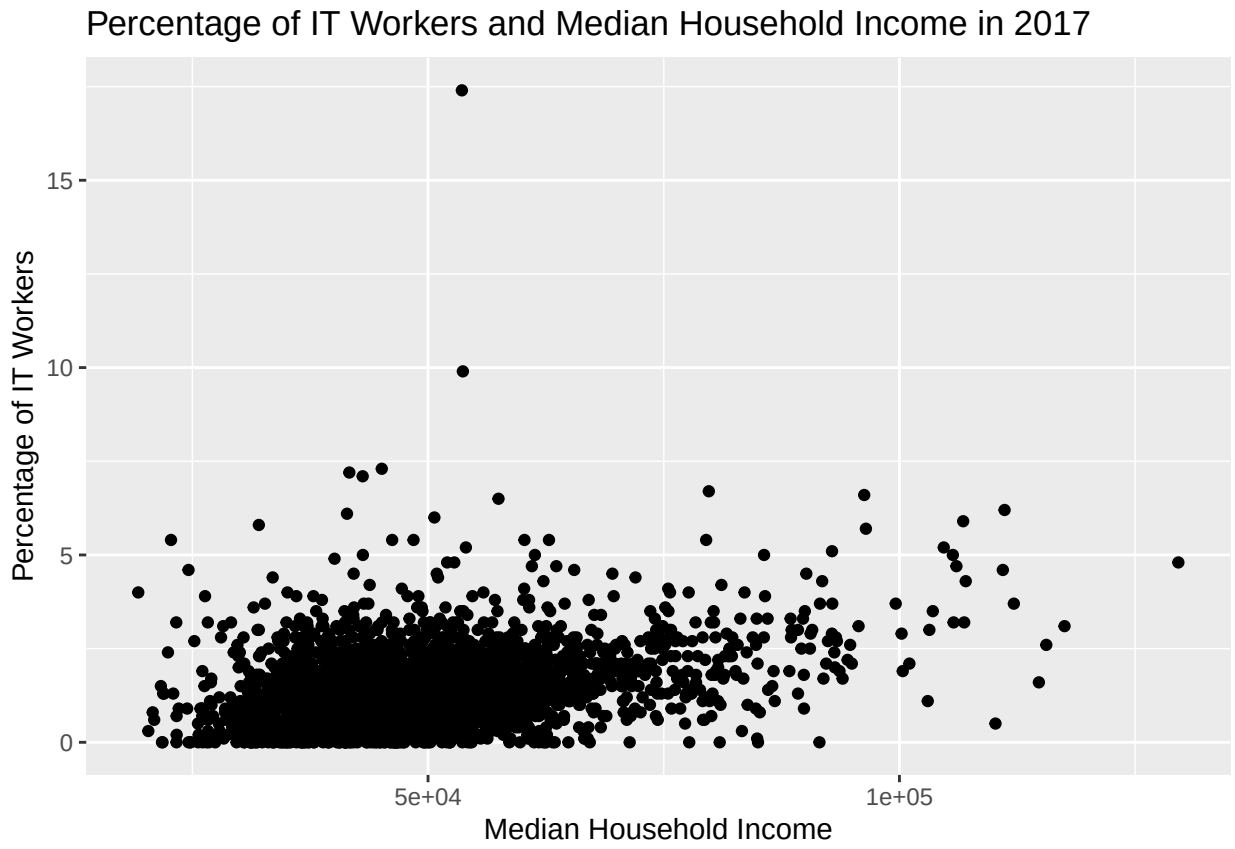

```
# draft (this is a test plot)

# Create the plot
ggplot(imputed_data, aes(x = "", y = pct_bb_2021)) +
  geom_boxplot(fill = "lightblue") +
  labs(title = "Percentage of Population with Broadband Access in 2021",
       x = NULL, y = "Percentage") +
  theme_minimal()
```



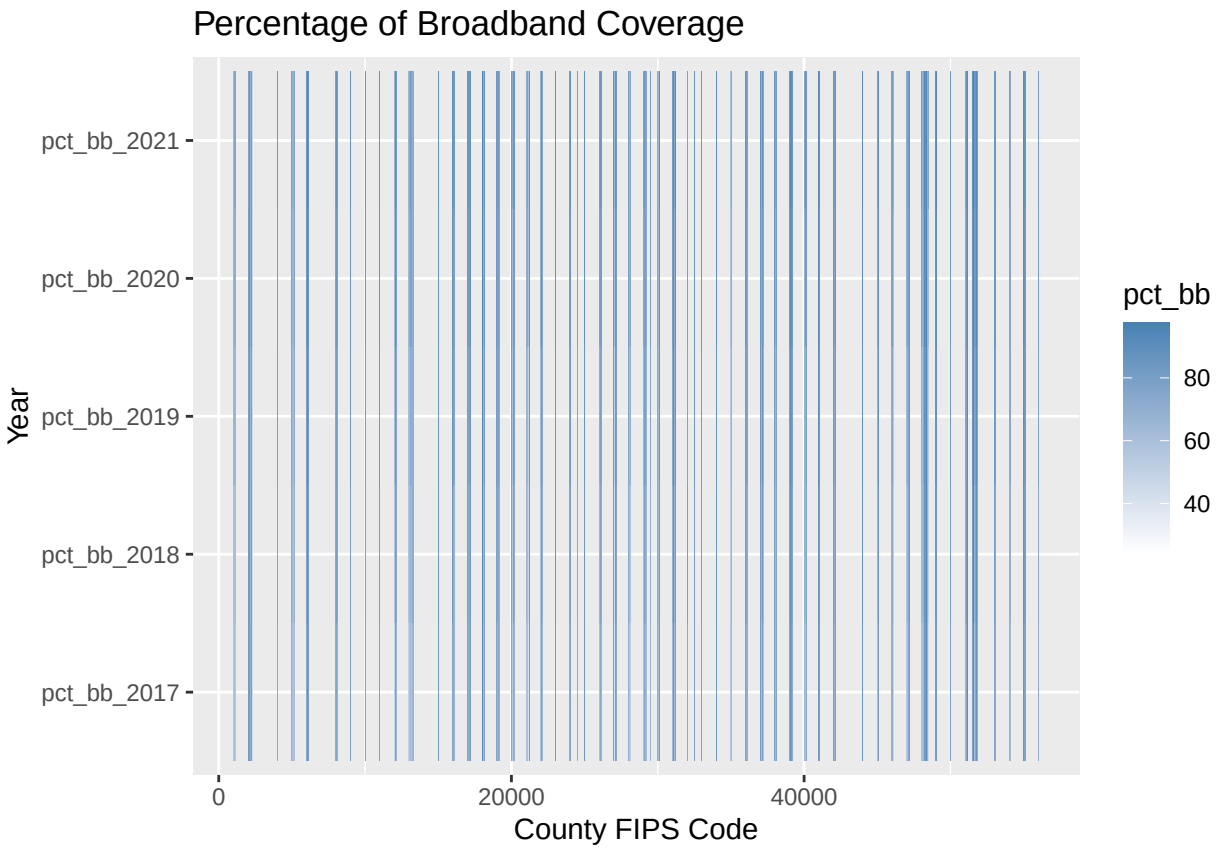

```
# draft (this is a test plot)

# Create the box plot
ggplot(imputed_data, aes(x = "", y = pct_bb_2017)) +
  geom_boxplot(fill = "lightblue", color = "blue") +
  labs(title = "Percentage of Broadband Access in 2017", y = "Percentage") +
  theme_bw()
```




```
# draft (this is a test plot)

# Scatter Plot
ggplot(imputed_data, aes(x=median_hh_inc_2017, y=pct_it_workers_2017)) +
  geom_point() +
  labs(title = "Percentage of IT Workers and Median Household Income in 2017",
       x = "Median Household Income", y = "Percentage of IT Workers")
```




```
# Heatmap
imputed_data_long <- melt(imputed_data, id.vars="cfips",
                          measure.vars=c("pct_bb_2017", "pct_bb_2018", "pct_bb_2019",
                                         "pct_bb_2020", "pct_bb_2021"),
                          variable.name="year", value.name="pct_bb")

ggplot(imputed_data_long, aes(x=cfips, y=year, fill=pct_bb)) +
  geom_tile() +
  scale_fill_gradient(low="white", high="steelblue") +
  labs(title = "Percentage of Broadband Coverage",
       x = "County FIPS Code", y = "Year")
```


Percentage of Broadband Coverage
pct_bb_2021
pct_bb_2020
pct_bb_2019
pct_bb_2018
pct_bb_2017
Year
0
20000
40000
County FIPS Code
pct_bb
80
60
40