# 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:

where:

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

## 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](https://data.census.gov/). The percentage fields were derived from the raw counts provided by the ACS. All fields have a two year lag to match what information was 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

# Recognize package conflicts

library(conflicted)

# Multi-purpose package for data import, tidying, manipulation, visualisation, and programming

# Most common packages include: ggplot2, purrr, tibble, dplyr, tidyr, stringr, readr, forcats

library(tidyverse)

# Deal with missing data

library(mice)

# Related to plots

library(maps)

#library(ggmap)

library(gridExtra)

library(mapdata)

library(ggcorrplot)

library(corrplot)

# Training

library(caret)

library(gbm)

# Color palette

library(viridis)

# Future Selection

#library(KernSmooth)

library(glmnet)

library(randomForest)

# libraries required for calculating SMAPE

library(forecast)

library(Metrics)

# 3. Exploratory 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\_2019-11-01" ...

## $ 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 : 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" ...

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

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

# 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 ...

#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

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

# Change first\_day\_of\_month data type to Date

train\_df$first\_day\_of\_month <- as.Date(train\_df$first\_day\_of\_month)

test\_df$first\_day\_of\_month <- as.Date(test\_df$first\_day\_of\_month)

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" ...

## 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") +

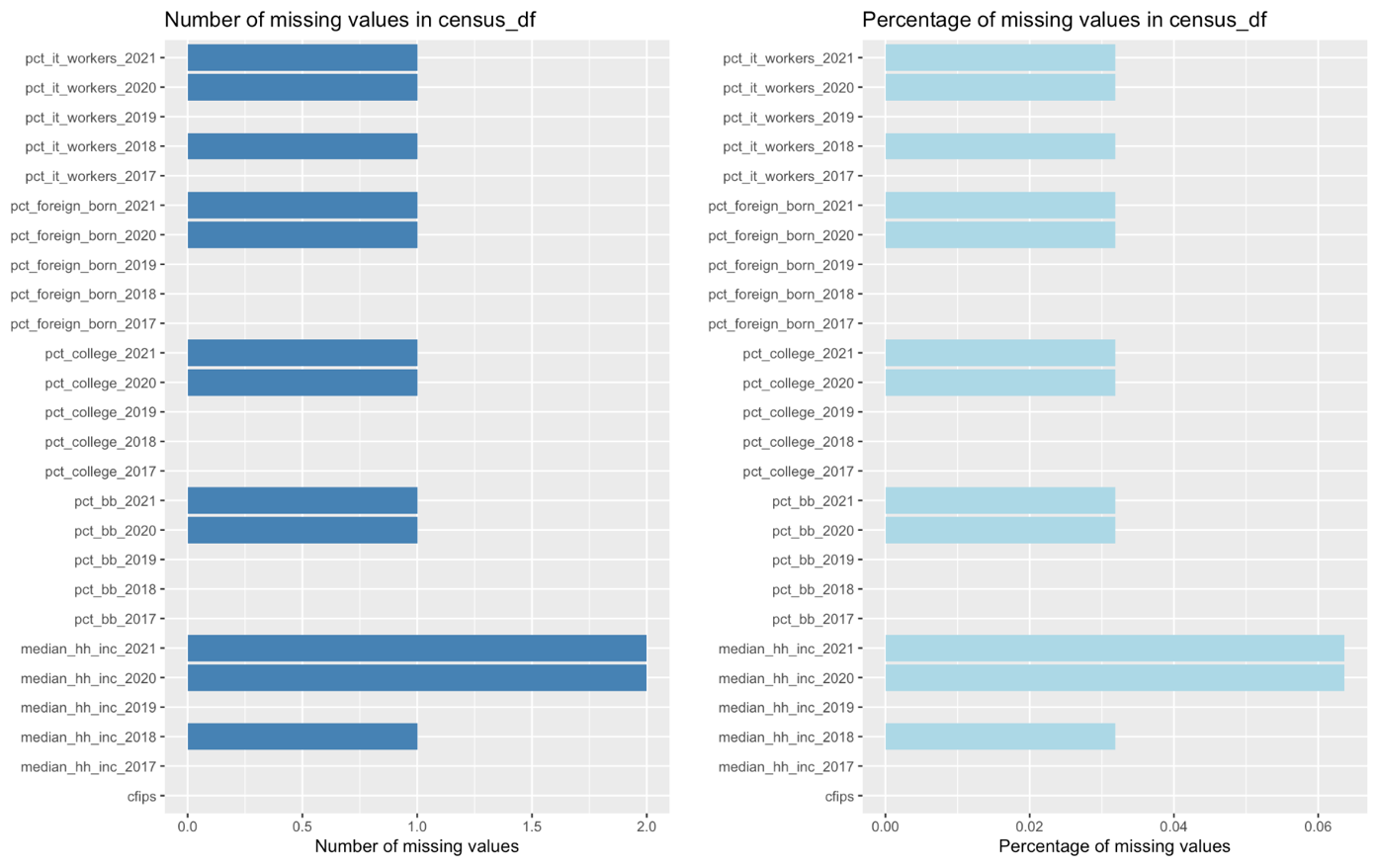
labs(x = "Percentage of missing values", y = "") +

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, 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

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 81.0 83.5 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 15.1

## 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 15.1 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 6.0 7.6

## 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 0.5 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 5.6 1.0 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 78889

## 1817 36148 39952 42264

## 2645 53194 53088 42285

## 2674 81875 83750 44076

## median\_hh\_inc\_2021

## 93 81817

## 1817 46994

## 2645 38659

## 2674 52842

## 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)

str(train\_df)

## '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" ...

# 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, year, month, pct\_bb, pct\_college, pct\_foreign\_born, pct\_it\_workers, median\_hh\_inc)

merged\_test <- test\_df

for (var in vars) {

# Select columns and pivot longer

merged\_test <- imputed\_data %>%

select(cfips, paste0(var, "\_2020"):paste0(var, "\_2021")) %>%

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\_test, by = c("cfips", "year", "month"), all.x = TRUE) %>%

arrange(cfips, row\_id)

}

merged\_test <- merged\_test %>%

select(row\_id, cfips, first\_day\_of\_month, year\_month, year, month, pct\_bb, pct\_college, pct\_foreign\_born, pct\_it\_workers, median\_hh\_inc)

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

colSums(is.na(merged\_test))

## 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)

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

colSums(is.na(merged\_test))

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

## 0 0 0 0

## year month pct\_bb pct\_college

## 0 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

## Min. :2019-08-01 Min. : 0.000 Min. : 0

## 1st Qu.:2020-05-01 1st Qu.: 1.639 1st Qu.: 145

## Median :2021-03-01 Median : 2.587 Median : 488

## Mean :2021-03-01 Mean : 3.818 Mean : 6443

## 3rd Qu.:2022-01-01 3rd Qu.: 4.519 3rd Qu.: 2124

## Max. :2022-10-01 Max. :284.340 Max. :1167744

## year\_month year month pct\_bb

## Length:122265 Min. :2019 Min. : 1.000 Min. :24.50

## Class :character 1st Qu.:2020 1st Qu.: 4.000 1st Qu.:69.30

## Mode :character Median :2021 Median : 7.000 Median :75.80

## Mean :2021 Mean : 6.692 Mean :74.69

## 3rd Qu.:2022 3rd Qu.:10.000 3rd Qu.:81.20

## Max. :2022 Max. :12.000 Max. :97.10

## pct\_college pct\_foreign\_born pct\_it\_workers median\_hh\_inc

## Min. : 0.00 Min. : 0.000 Min. : 0.000 Min. : 19264

## 1st Qu.:10.10 1st Qu.: 1.400 1st Qu.: 0.700 1st Qu.: 43505

## Median :13.20 Median : 2.700 Median : 1.200 Median : 51094

## Mean :14.22 Mean : 4.745 Mean : 1.356 Mean : 52830

## 3rd Qu.:17.30 3rd Qu.: 5.700 3rd Qu.: 1.800 3rd Qu.: 59230

## Max. :48.00 Max. :54.000 Max. :17.400 Max. :147111

## 3.7. 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.

### 3.7.1. Overall Microbusiness Density and Count

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

# Create plots

p1 <- merged\_df %>%

# Group merged\_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 <- merged\_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",

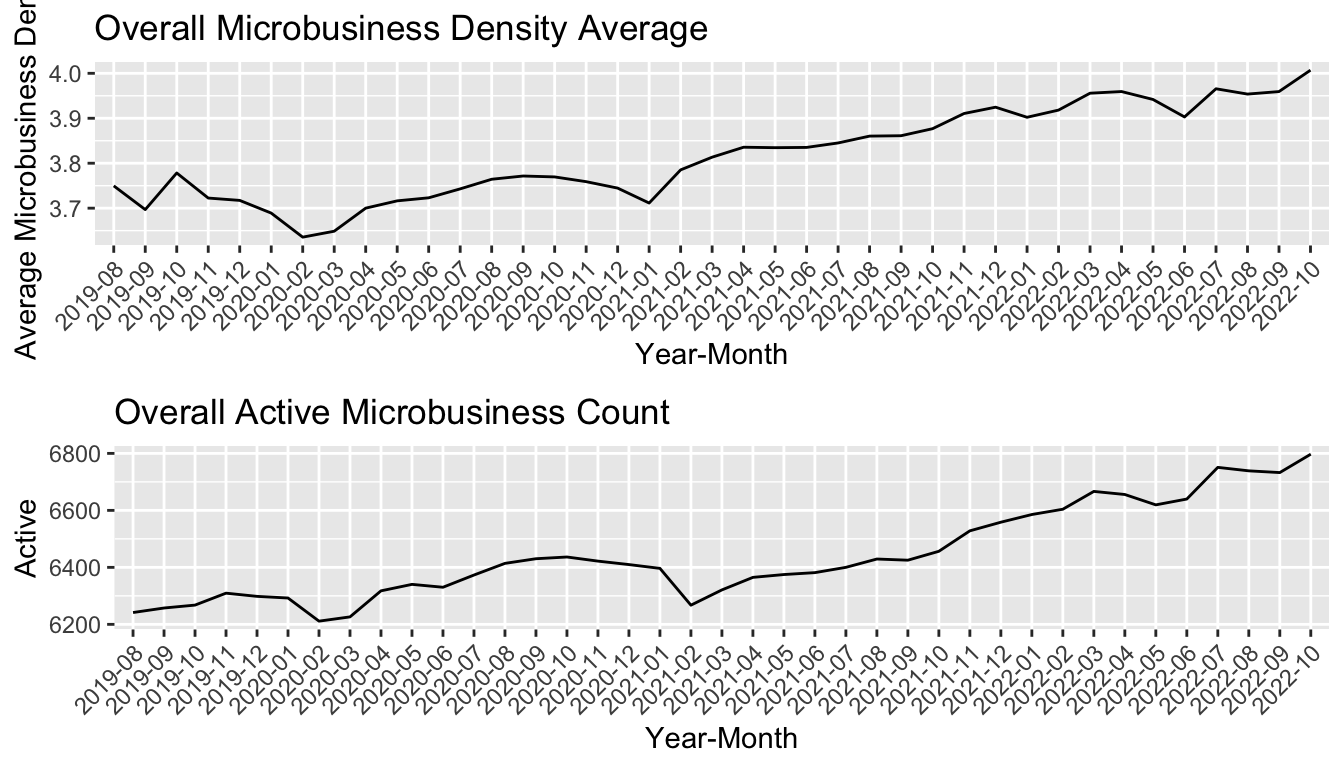
x = "Year-Month",

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.

# Create plots

p1 <- merged\_df %>%

# Group merged\_df by year\_month

group\_by(year\_month) %>%

# calculate the median value of microbusiness\_density for each group

summarise(median\_microbusiness\_density = median(microbusiness\_density)) %>%

ggplot(aes(x = year\_month, y = median\_microbusiness\_density, group = 1)) +

geom\_line() +

labs(title = "Overall Microbusiness Density Median",

x = "Year-Month",

y = "Microbusiness Density Median") +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

p2 <- merged\_df %>%

group\_by(year\_month) %>%

summarize(median\_active = median(active)) %>%

ggplot(aes(x = year\_month, y = median\_active)) +

geom\_line(group = 1) +

labs(title = "Overall Active Microbusiness Count Median",

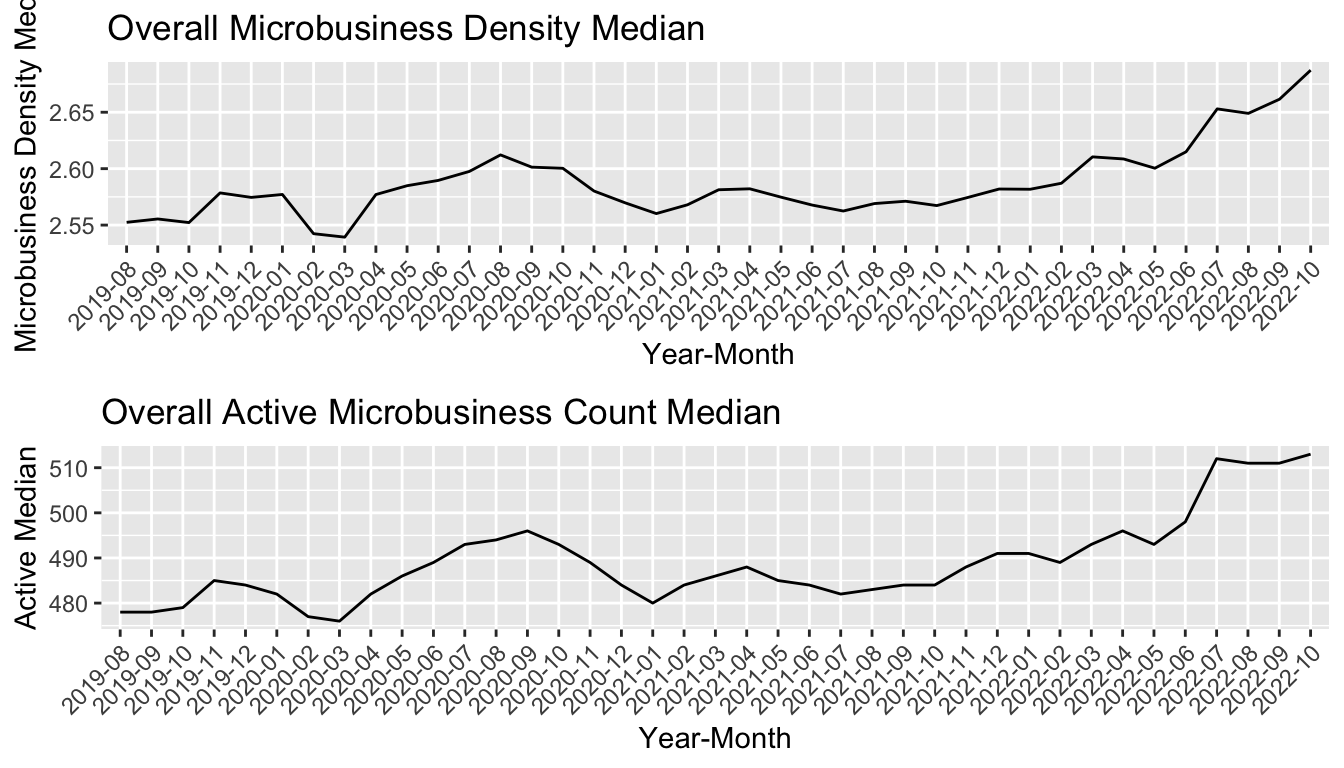
x = "Year-Month",

y = "Active Median") +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

# Display the plots

grid.arrange(p1, p2, nrow = 2)



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

# Group merged\_df by year and calculate the mean value of microbusiness\_density and active for each group

merged\_df\_mean\_year <- merged\_df %>%

group\_by(year) %>%

summarize(avg\_microbusiness\_density = mean(microbusiness\_density),

avg\_active = mean(active))

# Group merged\_df by month and calculate the mean value of the microbusiness\_density for each group

merged\_df\_mean\_month <- merged\_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(merged\_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(merged\_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(merged\_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(merged\_df\_mean\_year, aes(x = year, y = avg\_active)) +

geom\_line() +

ggtitle("Average Yearly Active Microbusiness Count") +

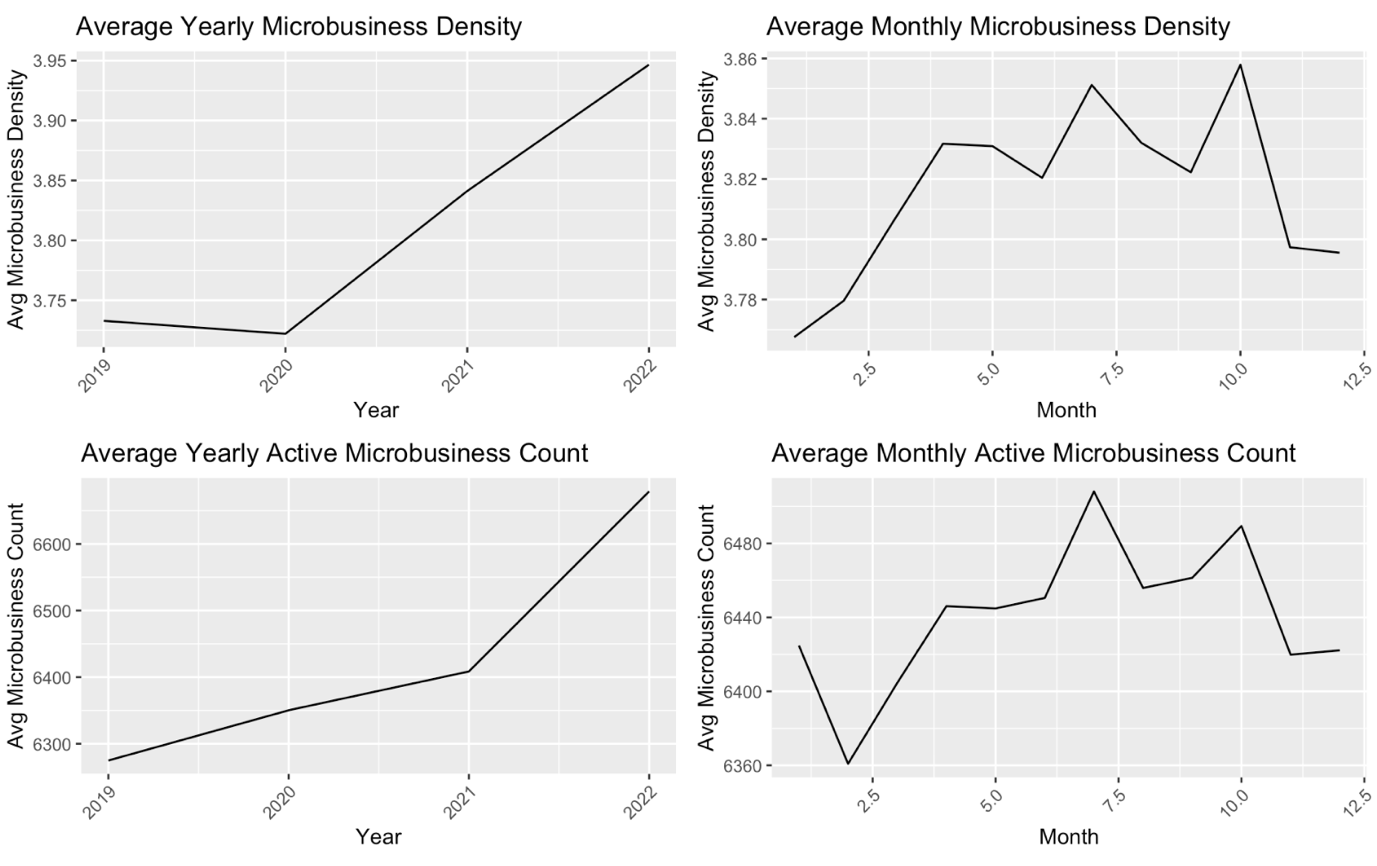
xlab("Year") +

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.

# Create a grid of box plots

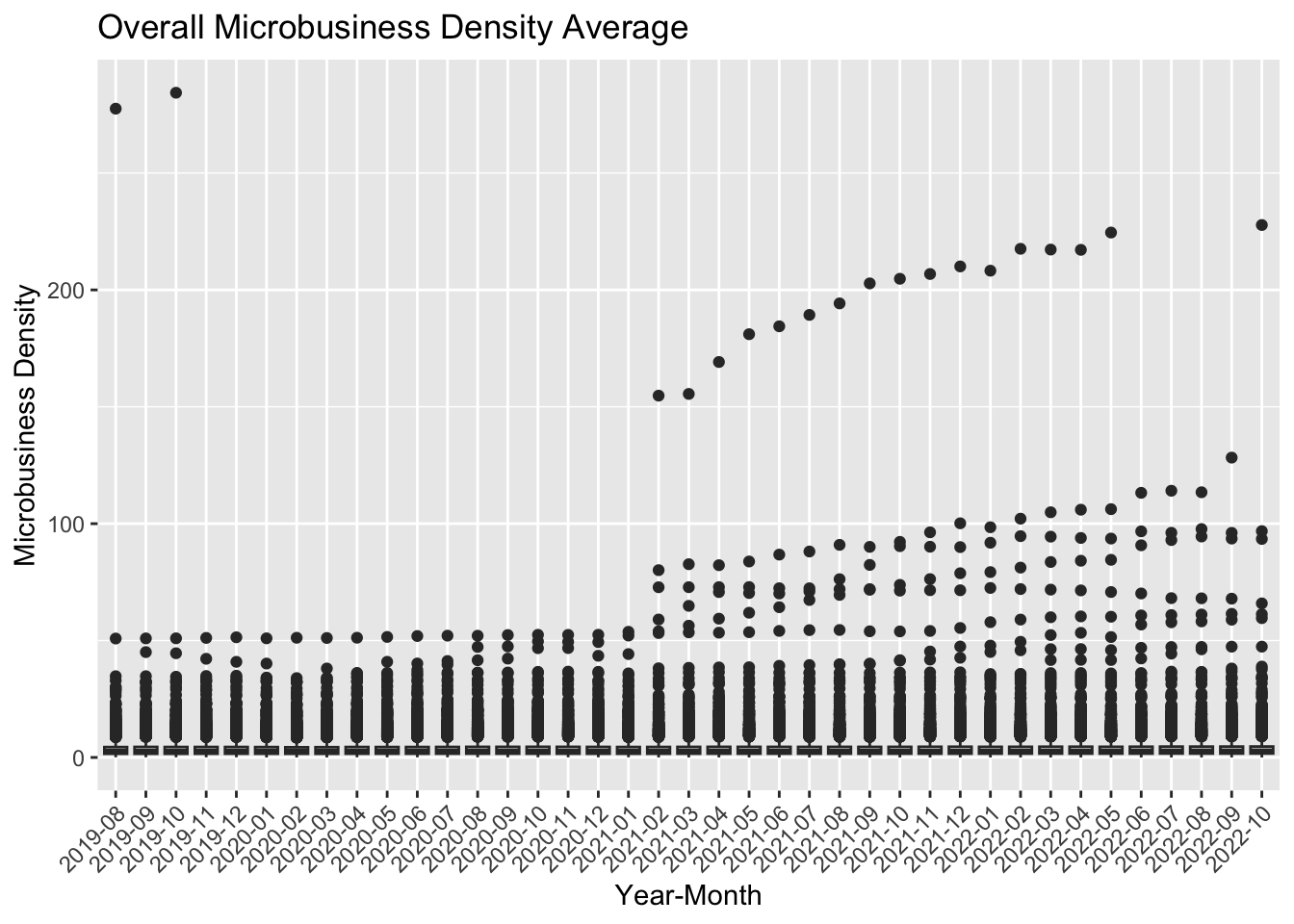
ggplot(merged\_df, aes(x=year\_month, y=microbusiness\_density)) +

geom\_boxplot() +

labs(x="Year-Month", y="Microbusiness Density") +

ggtitle("Overall Microbusiness Density Average") +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))



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.

# Create a grid of box plots

ggplot(merged\_df, aes(x=year\_month, y=microbusiness\_density)) +

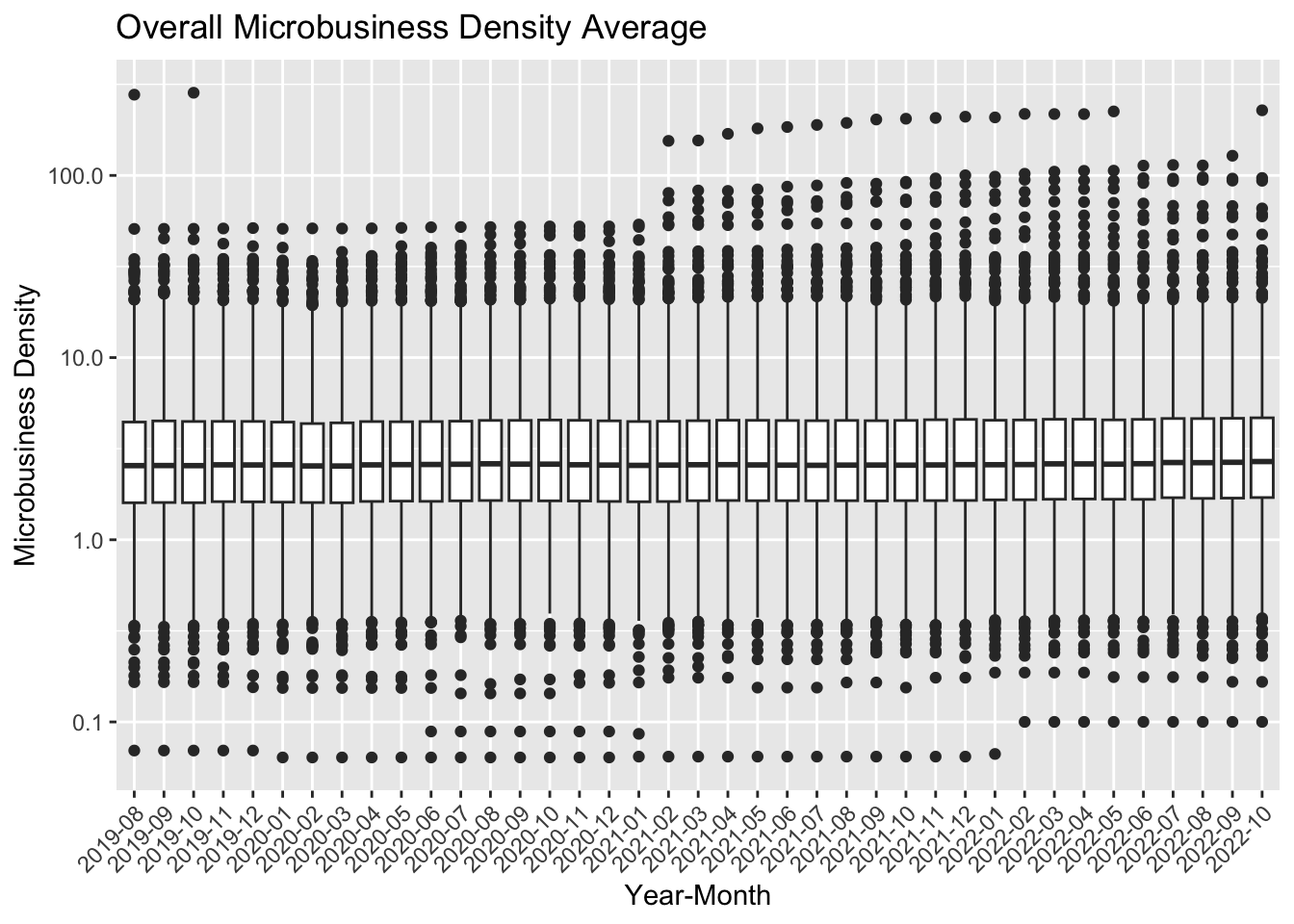
geom\_boxplot() +

scale\_y\_log10() +

labs(x="Year-Month", y="Microbusiness Density") +

ggtitle("Overall Microbusiness Density Average") +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))



### 3.7.2. Economic Regional Divisions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

# Convert state and county columns in merged\_df to lowercase

merged\_df <- merged\_df %>%

mutate(state = tolower(state)) %>%

mutate(county = tolower(county))

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

# Create a new column named region and initialize all values as NA

merged\_df$region <- NA

# Assign region values based on state column

for (i in 1:nrow(merged\_df)) {

if (merged\_df$state[i] %in% c("connecticut", "maine", "massachusetts", "new hampshire", "rhode island", "vermont")) {

merged\_df$region[i] <- "new england"

} else if (merged\_df$state[i] %in% c("delaware", "maryland", "new jersey", "new york", "pennsylvania", "district of columbia")) {

merged\_df$region[i] <- "mideast"

} else if (merged\_df$state[i] %in% c("illinois", "indiana", "michigan", "ohio", "wisconsin")) {

merged\_df$region[i] <- "great lakes"

} else if (merged\_df$state[i] %in% c("iowa", "kansas", "minnesota", "missouri", "nebraska", "north dakota", "south dakota")) {

merged\_df$region[i] <- "plains"

} else if (merged\_df$state[i] %in% c("alabama", "arkansas", "florida", "georgia", "kentucky", "louisiana", "mississippi", "north carolina", "south carolina", "tennessee", "virginia", "west virginia")) {

merged\_df$region[i] <- "southeast"

} else if (merged\_df$state[i] %in% c("arizona", "new mexico", "oklahoma", "texas")) {

merged\_df$region[i] <- "southwest"

} else if (merged\_df$state[i] %in% c("colorado", "idaho", "montana", "utah", "wyoming")) {

merged\_df$region[i] <- "rocky mountain"

} else if (merged\_df$state[i] %in% c("alaska", "california", "hawaii", "nevada", "oregon", "washington")) {

merged\_df$region[i] <- "far west"

} else {

merged\_df$region[i] <- "other"

}

}

# Print all the unique values in the region column

unique(merged\_df$region)

## [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:

# 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", "vermont"), "New England",

ifelse(region %in% c("delaware", "maryland", "new jersey", "new york", "pennsylvania", "district of columbia"), "Mideast",

ifelse(region %in% c("illinois", "indiana", "michigan", "ohio", "wisconsin"), "Great Lakes",

ifelse(region %in% c("iowa", "kansas", "minnesota", "missouri", "nebraska", "north dakota", "south dakota"), "Plains",

ifelse(region %in% c("alabama", "arkansas", "florida", "georgia", "kentucky", "louisiana", "mississippi", "north carolina", "south carolina", "tennessee", "virginia", "west virginia"), "Southeast",

ifelse(region %in% c("arizona", "new mexico", "oklahoma", "texas"), "Southwest",

ifelse(region %in% c("colorado", "idaho", "montana", "utah", "wyoming"), "Rocky Mountain",

ifelse(region %in% c("alaska", "california", "hawaii", "nevada", "oregon", "washington"), "Far West", NA

)))))))))

# 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 = "white", fontface = "bold") +

# scale\_fill\_gradient(low = "white", high = "darkred") +

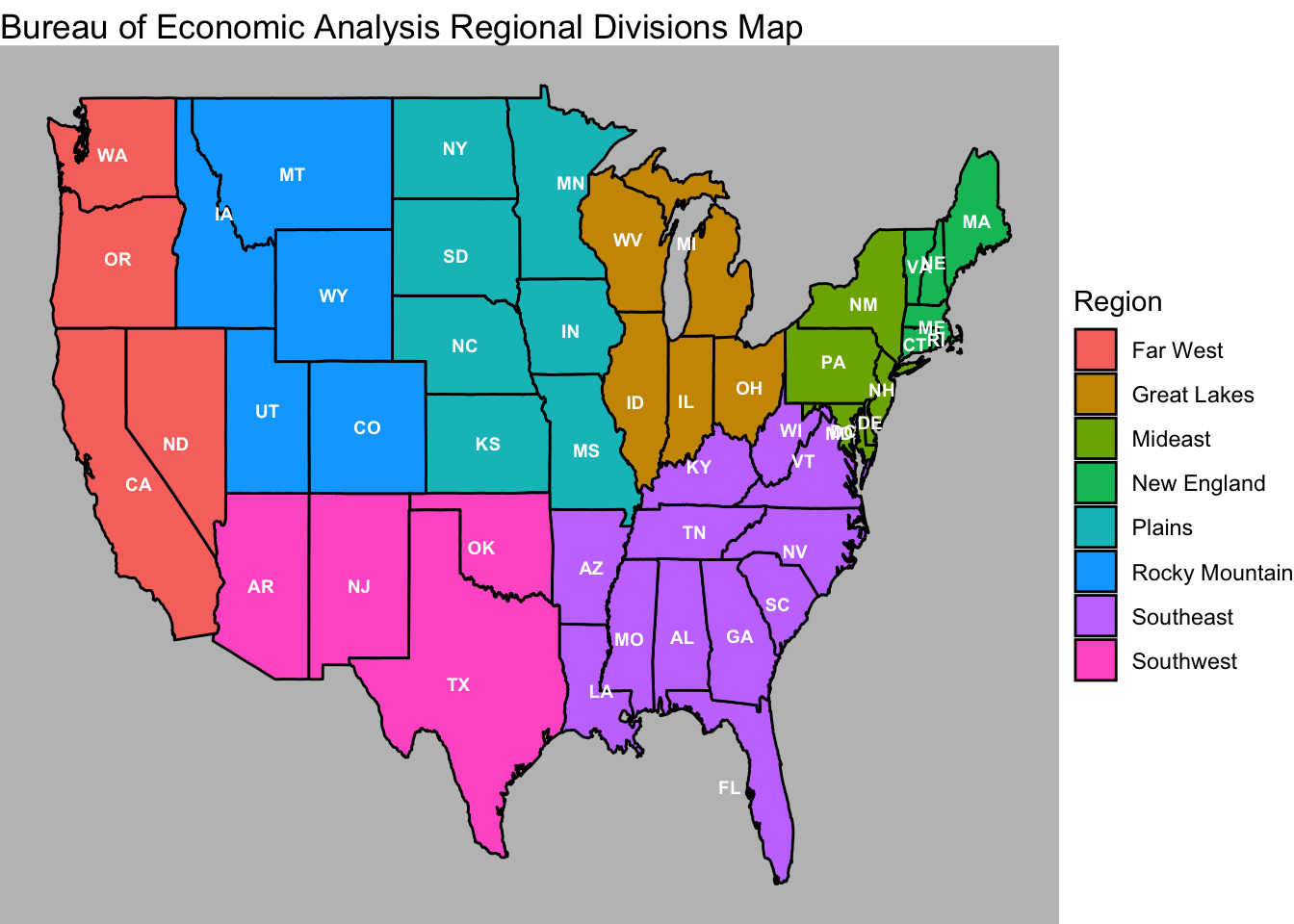
# scale\_fill\_manual(values = viridis(n = 60), na.value = "gray") +

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))



# Group merged\_df by region and calculate average microbusiness density

merged\_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") +

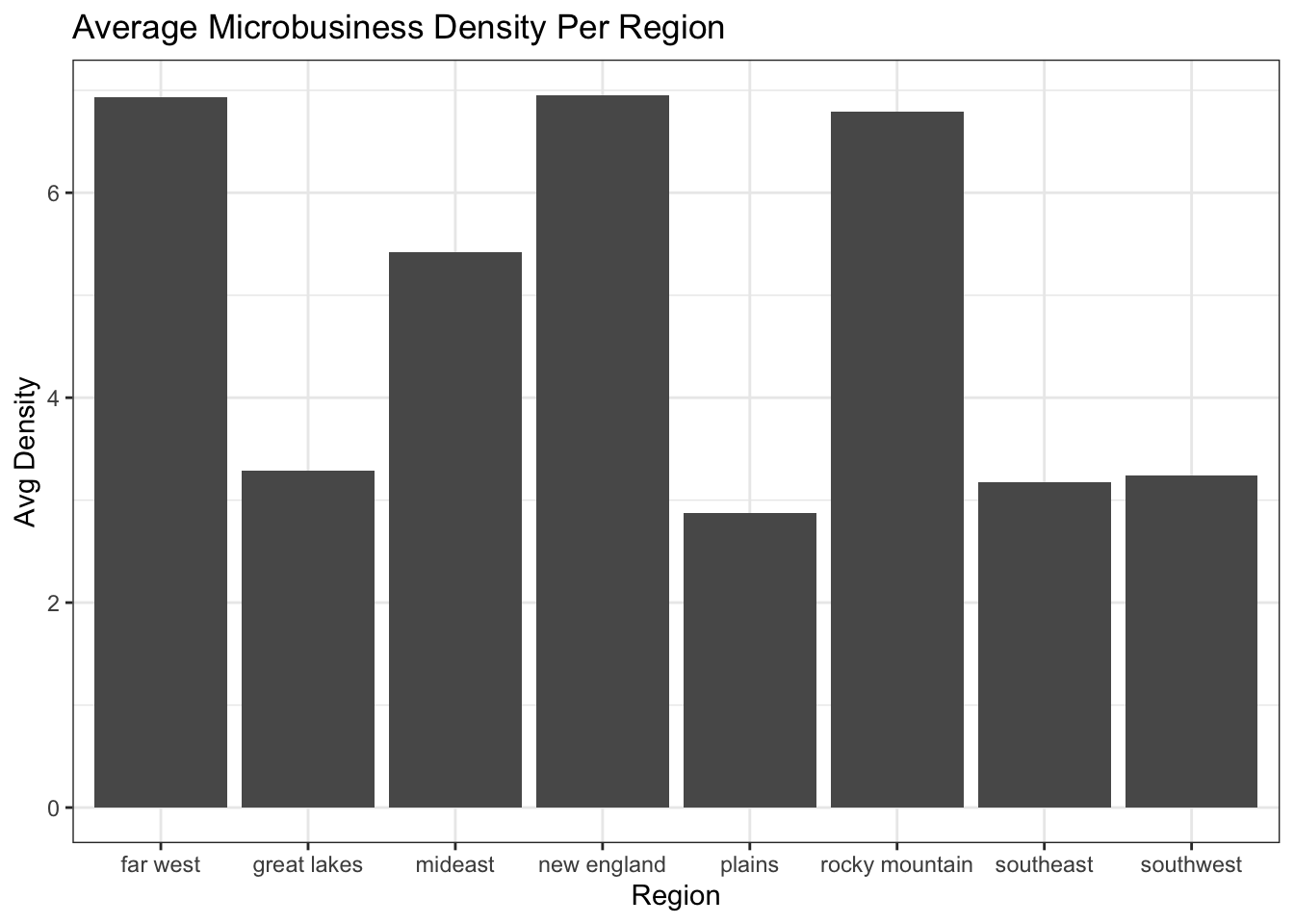
# Add plot title and axis labels

labs(title = "Average Microbusiness Density Per Region",

x = "Region", y = "Avg Density") +

# Apply a black and white theme to the plot

theme\_bw()



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.

# choropleth map

# Group merged\_df by region and calculate average microbusiness density

avg\_density <- merged\_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

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,

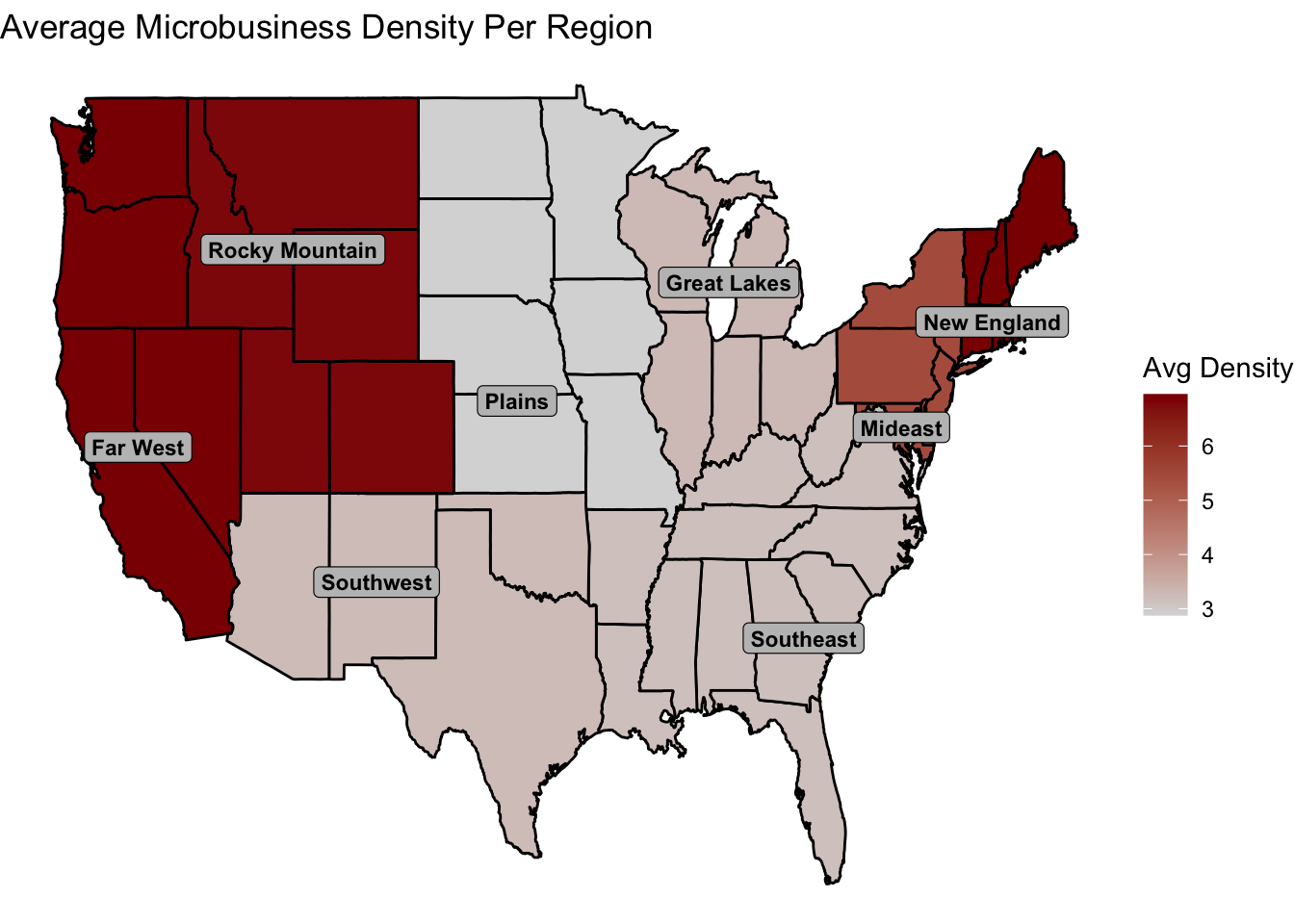
fill = "gray75", color = "black") +

scale\_fill\_gradient(low = "gray85", high = "darkred") +

# scale\_fill\_viridis(name = "Avg Density", na.value = "gray") +

labs(title = "Average Microbusiness Density Per Region", fill = "Avg Density") +

theme\_void()



# theme(panel.background = element\_rect(fill = "gray90", color = NA))

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.

# Aggregate data by region

df\_by\_region <- aggregate(microbusiness\_density ~ region, merged\_df, median)

# Create boxplot

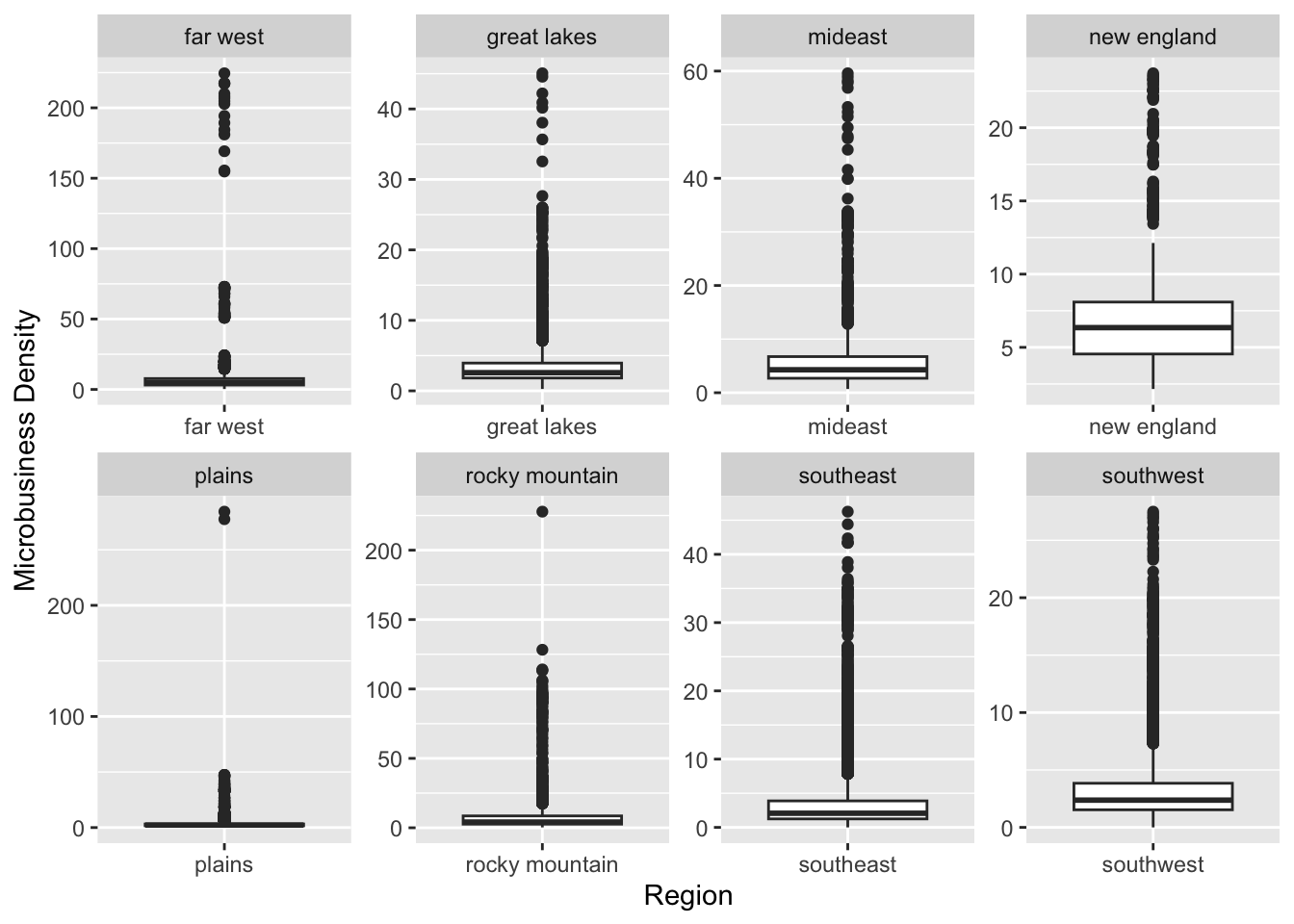
ggplot(merged\_df, aes(x = region, y = microbusiness\_density)) +

geom\_boxplot() +

labs(x = "Region", y = "Microbusiness Density") +

# Arrange plots in grid

facet\_wrap(~ region, scales = "free", nrow = 2)



Above plots are 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.

# Create a grid of box plots

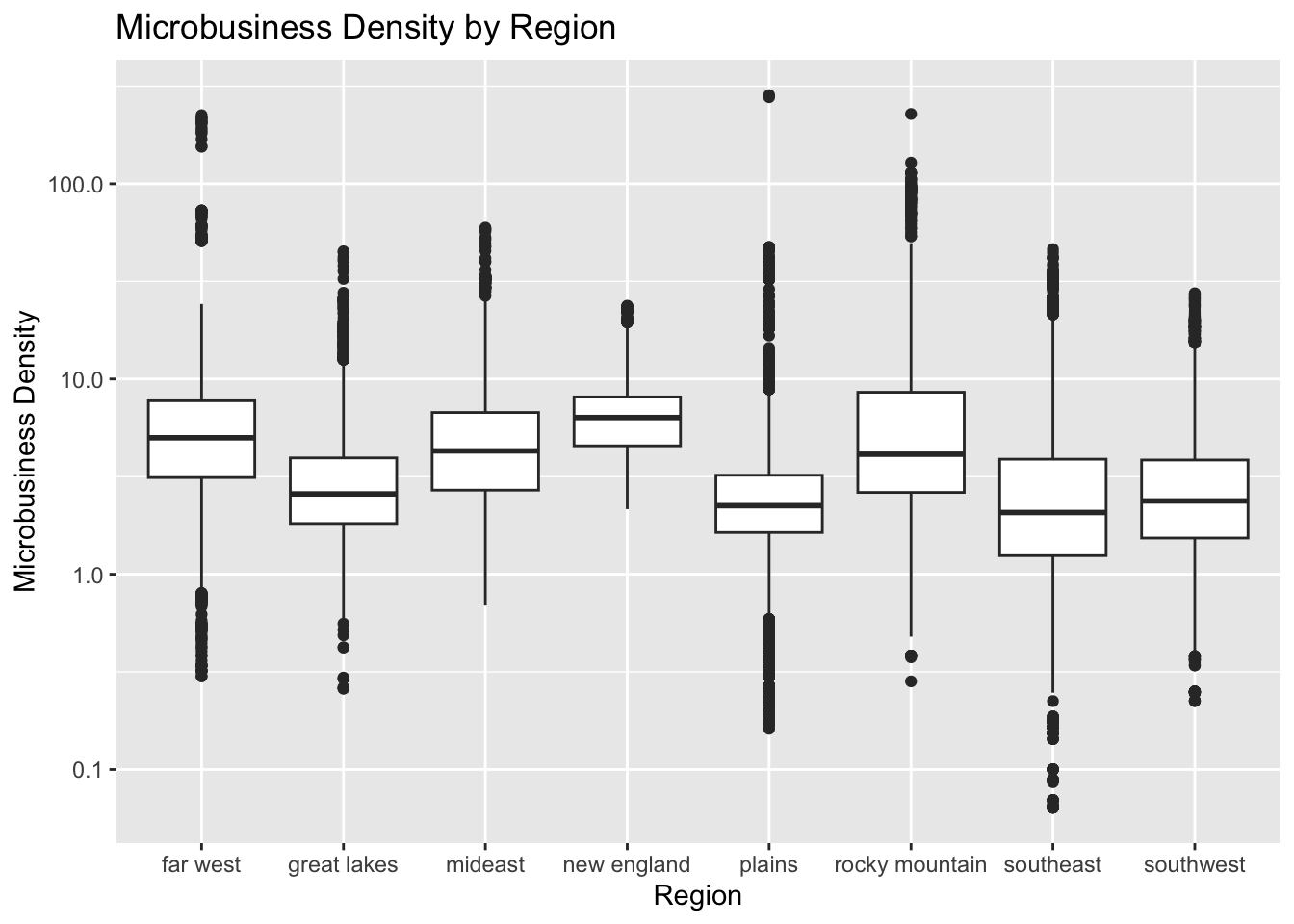
ggplot(merged\_df, aes(x=region, y=microbusiness\_density)) +

geom\_boxplot() +

scale\_y\_log10() +

labs(x="Region", y="Microbusiness Density") +

ggtitle("Microbusiness Density by Region")



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:

* All regions have at least some microbusiness activity. Because minimum microbusiness density is greater than 0 in all regions.
* The median microbusiness density is highest in the New England region, followed by the Far West and the Mideast regions.
* The 3rd quartile microbusiness density is highest in the Rocky Mountain region, followed by the New England and the Far West regions.
* The maximum microbusiness density is highest in the Plains region, followed by the Rocky Mountain and the Far West regions.
* The mean microbusiness density is highest in the New England region, followed by the Far West and the Rocky Mountain regions.
* The interquartile range (IQR = the difference between the 1st and 3rd quartiles) of microbusiness density is widest in the Rocky Mountain region, indicating that there is a greater range of microbusiness density in that region. In contrast, the IQR is narrowest in the Plains region.

# by(merged\_df$microbusiness\_density, merged\_df$region, summary)

# Aggregate microbusiness density by state

state\_avg <- aggregate(microbusiness\_density ~ state, data = merged\_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() +

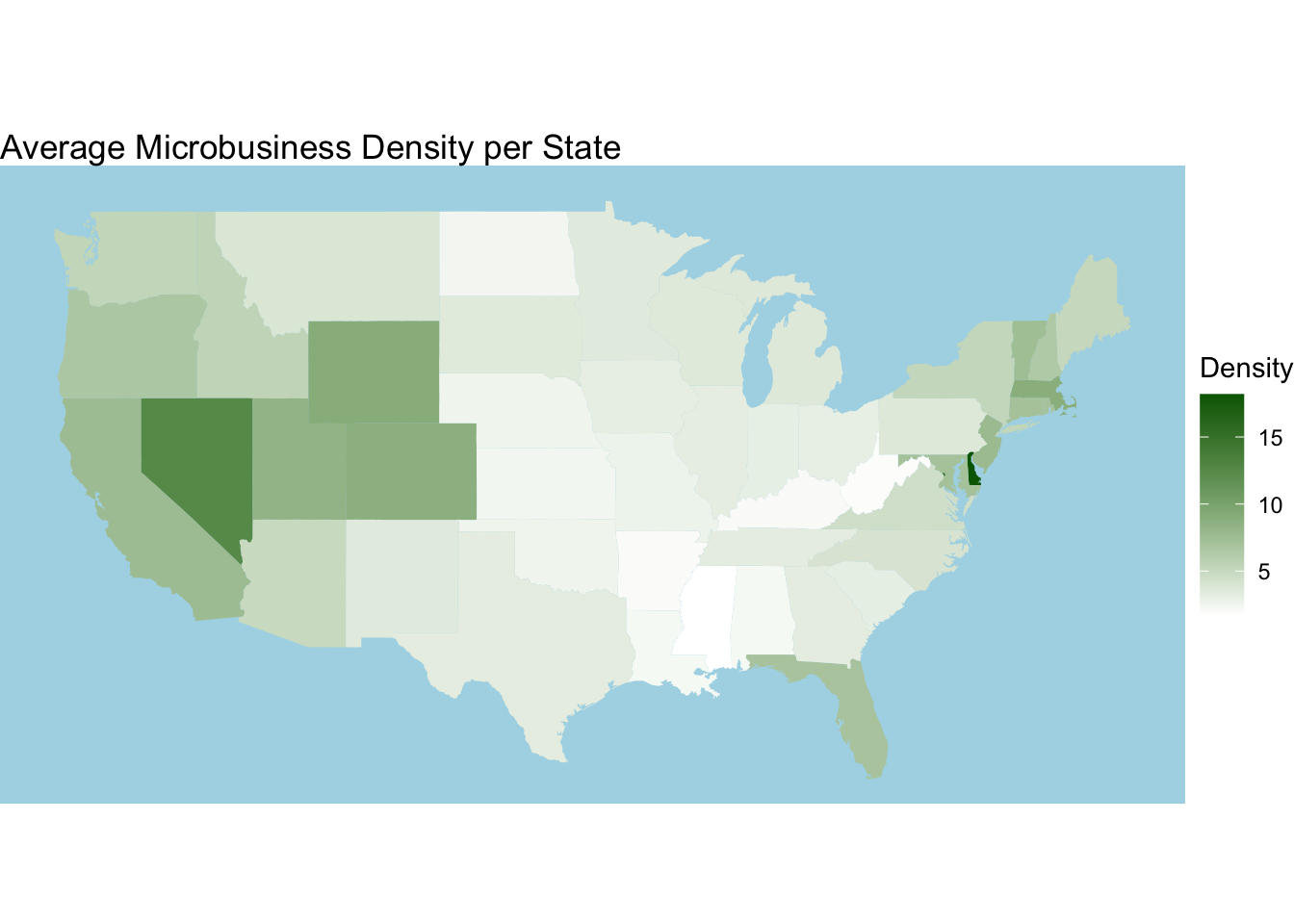
scale\_fill\_gradient(low = "white", high = "darkgreen") +

coord\_map() +

labs(title = "Average Microbusiness Density per State", fill = "Density") +

theme\_void() +

theme(panel.background = element\_rect(fill = "lightblue", color = NA))



# Create a grid of box plots

ggplot(merged\_df, aes(x=state, y=microbusiness\_density)) +

geom\_boxplot(colour = "darkblue", outlier.colour = "pink") +

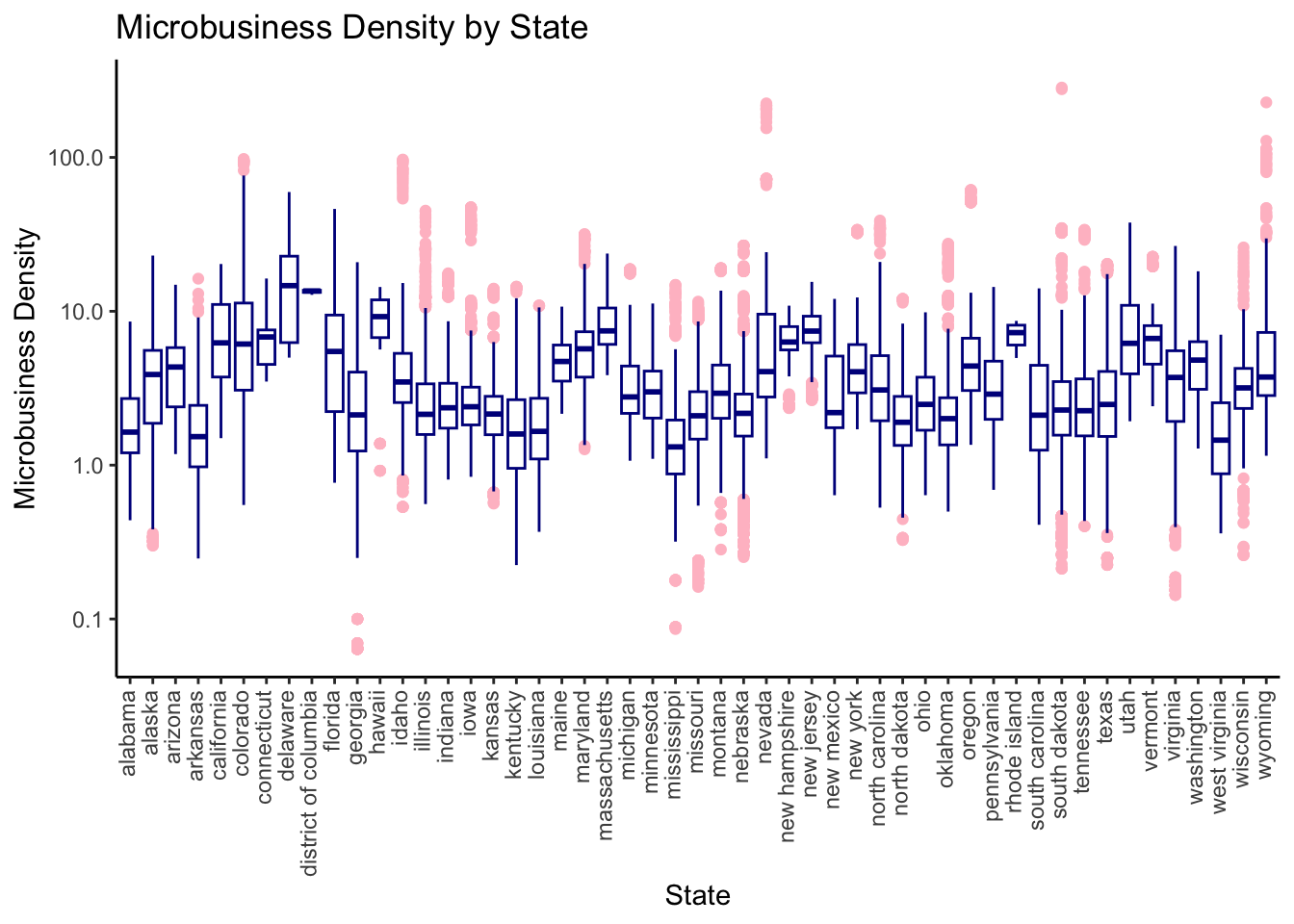
scale\_y\_log10() +

labs(x="State", y="Microbusiness Density") +

ggtitle("Microbusiness Density by State") +

theme\_classic() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1, vjust = 0.3))



# {r fig.width = 10 ,fig.height = 12, out.width='100%', fig.align='center'}

# Aggregate microbusiness density by county

#county\_avg <- merged\_df %>%

# group\_by(cfips, county) %>%

# summarise(microbusiness\_density = mean(microbusiness\_density))

county\_avg <- aggregate(microbusiness\_density ~ county + state, data = merged\_df, FUN = mean)

# Get rid of county, city, and parish in the end of county names

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 16 variables:

## $ row\_id : chr "1001\_2019-08-01" "1001\_2019-09-01" "1001\_2019-10-01" "1001\_2019-11-01" ...

## $ cfips : int 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 ...

## $ county : chr "autauga county" "autauga county" "autauga county" "autauga county" ...

## $ state : chr "alabama" "alabama" "alabama" "alabama" ...

## $ first\_day\_of\_month : Date, format: "2019-08-01" "2019-09-01" ...

## $ microbusiness\_density: num 3.01 2.88 3.06 2.99 2.99 ...

## $ active : int 1249 1198 1269 1243 1243 1242 1217 1227 1255 1257 ...

## $ year\_month : chr "2019-08" "2019-09" "2019-10" "2019-11" ...

## $ year : num 2019 2019 2019 2019 2019 ...

## $ month : int 8 9 10 11 12 1 2 3 4 5 ...

## $ pct\_bb : num 76.6 76.6 76.6 76.6 76.6 78.9 78.9 78.9 78.9 78.9 ...

## $ pct\_college : num 14.5 14.5 14.5 14.5 14.5 15.9 15.9 15.9 15.9 15.9 ...

## $ pct\_foreign\_born : num 2.1 2.1 2.1 2.1 2.1 2 2 2 2 2 ...

## $ pct\_it\_workers : num 1.3 1.3 1.3 1.3 1.3 1.1 1.1 1.1 1.1 1.1 ...

## $ median\_hh\_inc : num 55317 55317 55317 55317 55317 ...

## $ region : chr "southeast" "southeast" "southeast" "southeast" ...

## - 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.

ggplot(merged\_df, aes(y = microbusiness\_density)) +

# ggplot(merged\_df, aes(x=region, y=microbusiness\_density)) +

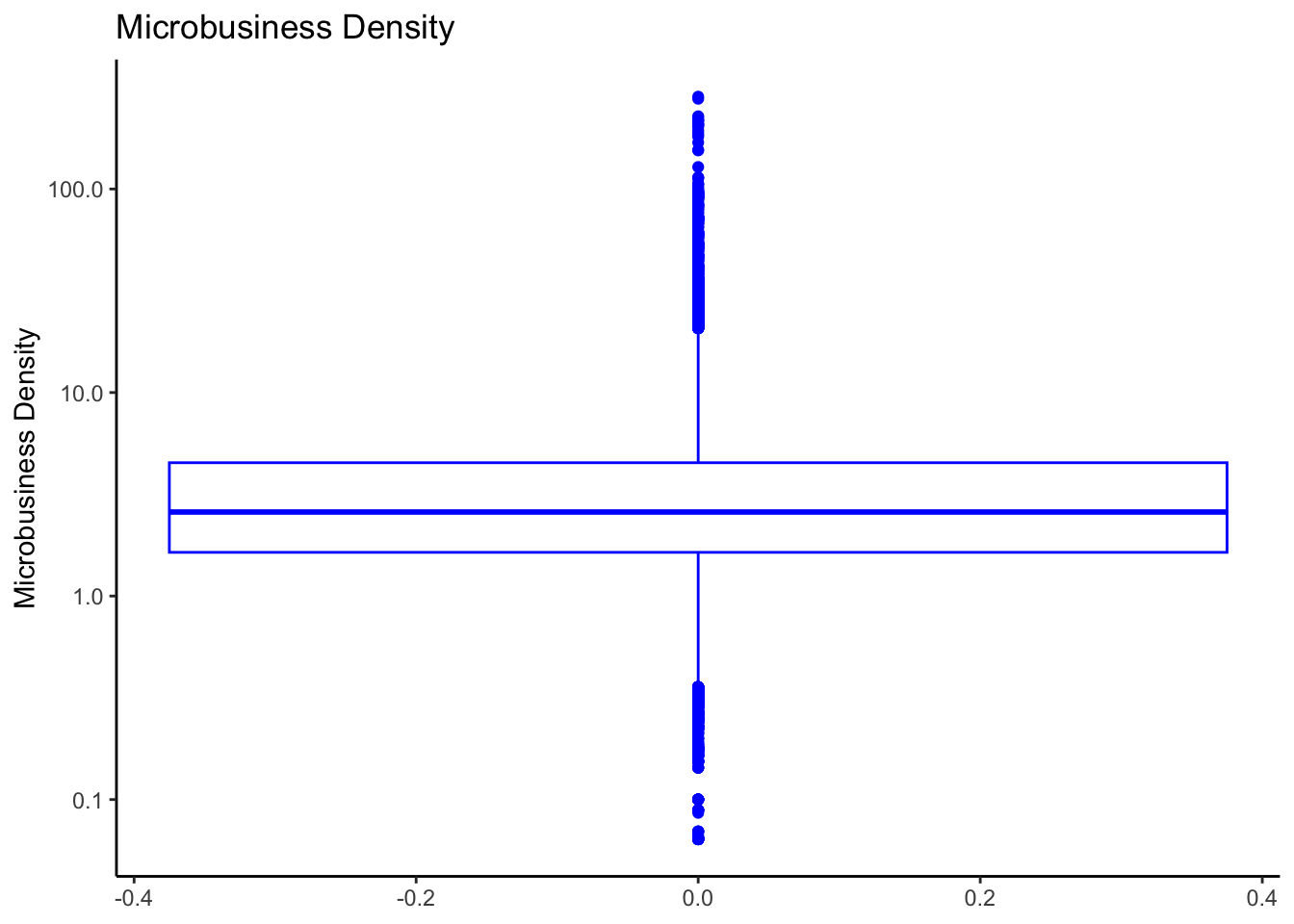
geom\_boxplot(colour = "blue") +

scale\_y\_log10() +

labs(y="Microbusiness Density") +

ggtitle("Microbusiness Density") +

theme\_classic()



# Create a grid of box plots

par(mfrow=c(3,2))

p1 <- ggplot(merged\_df, aes(x=region, y=median\_hh\_inc)) +

geom\_boxplot(outlier.colour = "orange") +

scale\_y\_log10() +

labs(x="Region", y="Median Household Income") +

ggtitle("Median Household Income by Region") +

theme\_classic() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

p2 <- ggplot(merged\_df, aes(x=region, y=pct\_college)) +

geom\_boxplot(outlier.colour = "orange") +

scale\_y\_log10() +

labs(x="Region", y="Percentage with College Education") +

ggtitle("Percentage with College Education by Region") +

theme\_classic() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

p3 <- ggplot(merged\_df, aes(x=region, y=pct\_foreign\_born)) +

geom\_boxplot(outlier.colour = "orange") +

scale\_y\_log10() +

labs(x="Region", y="Percentage of Foreign-born Residents") +

ggtitle("Percentage of Foreign-born Residents by Region") +

theme\_classic() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

p4 <- ggplot(merged\_df, aes(x=region, y=pct\_it\_workers)) +

geom\_boxplot(outlier.colour = "orange") +

scale\_y\_log10() +

labs(x="Region", y="Percentage of IT Workers") +

ggtitle("Percentage of IT Workers by Region") +

theme\_classic() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

p5 <- ggplot(merged\_df, aes(x=region, y=pct\_bb)) +

geom\_boxplot(outlier.colour = "orange") +

scale\_y\_log10() +

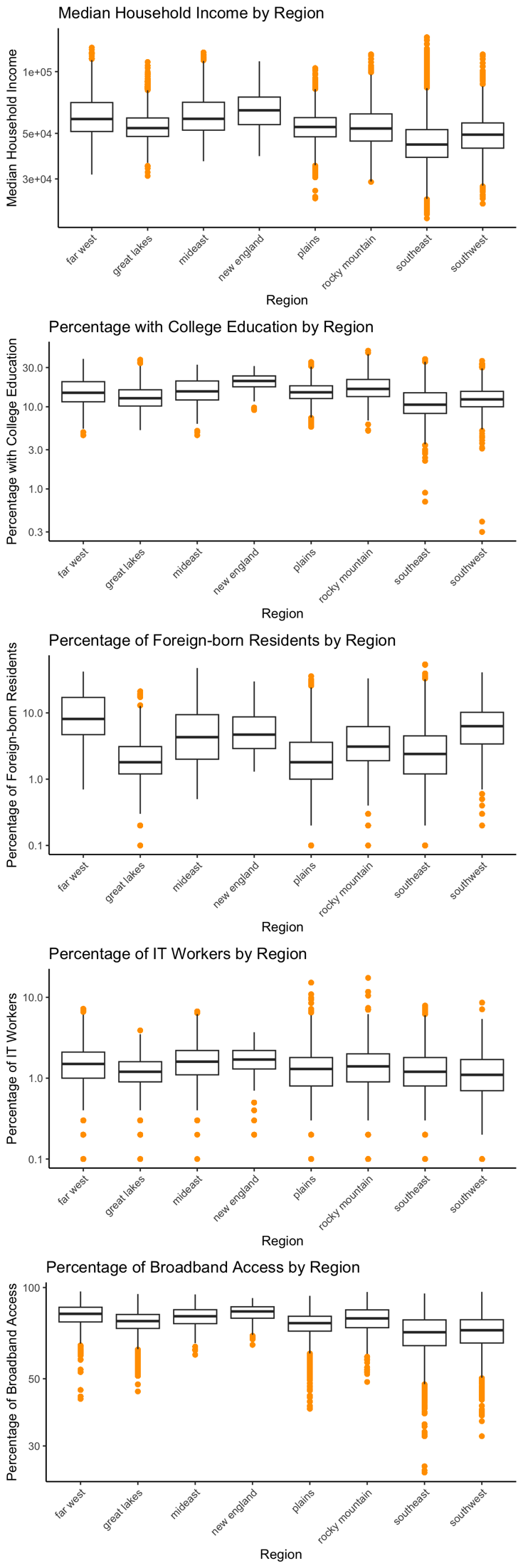
labs(x="Region", y="Percentage of Broadband Access") +

ggtitle("Percentage of Broadband Access by Region") +

theme\_classic() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

grid.arrange(p1, p2, p3, p4, p5, nrow = 5)



# Create a grid of box plots

p1 <- ggplot(merged\_df, aes(x=state, y=median\_hh\_inc)) +

geom\_boxplot(outlier.colour = "orange") +

scale\_y\_log10() +

labs(x="State", y="Median Household Income") +

ggtitle("Median Household Income by State") +

theme\_classic() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1, vjust = 0.3))

p2 <- ggplot(merged\_df, aes(x=state, y=pct\_college)) +

geom\_boxplot(outlier.colour = "orange") +

scale\_y\_log10() +

labs(x="State", y="Percentage with College Education") +

ggtitle("Percentage with College Education by State") +

theme\_classic() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1, vjust = 0.3))

p3 <- ggplot(merged\_df, aes(x=state, y=pct\_foreign\_born)) +

geom\_boxplot(outlier.colour = "orange") +

scale\_y\_log10() +

labs(x="State", y="Percentage of Foreign-born Residents") +

ggtitle("Percentage of Foreign-born Residents by State") +

theme\_classic() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1, vjust = 0.3))

p4 <- ggplot(merged\_df, aes(x=state, y=pct\_it\_workers)) +

geom\_boxplot(outlier.colour = "orange") +

scale\_y\_log10() +

labs(x="State", y="Percentage of IT Workers") +

ggtitle("Percentage of IT Workers by State") +

theme\_classic() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1, vjust = 0.3))

p5 <- ggplot(merged\_df, aes(x=state, y=pct\_bb)) +

geom\_boxplot(outlier.colour = "orange") +

scale\_y\_log10() +

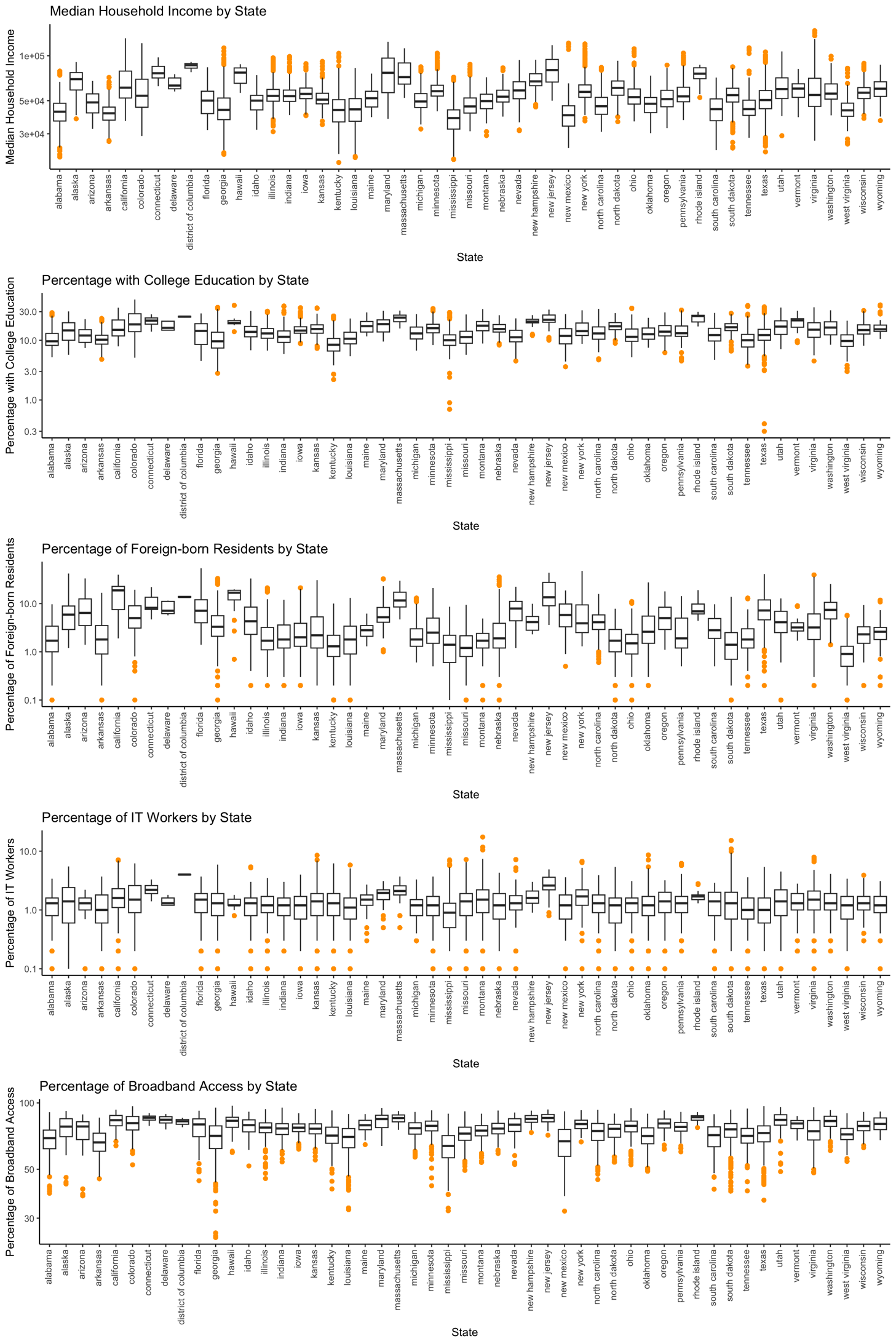
labs(x="State", y="Percentage of Broadband Access") +

ggtitle("Percentage of Broadband Access by State") +

theme\_classic() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1, vjust = 0.3))

grid.arrange(p1, p2, p3, p4, p5, nrow = 5)



# choropleth map with 5 variables

# Group merged\_df by region and calculate average variables

avg\_variables <- merged\_df %>%

group\_by(region) %>%

summarize(avg\_pct\_bb = mean(pct\_bb),

avg\_pct\_college = mean(pct\_college),

avg\_pct\_foreign\_born = mean(pct\_foreign\_born),

avg\_pct\_it\_workers = mean(pct\_it\_workers),

avg\_median\_hh\_inc = mean(median\_hh\_inc))

# 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

region\_map <- region\_map %>%

mutate(region = tolower(region))

# Merge the average density data with the region\_map data

plot\_data <- merge(region\_map, avg\_variables, 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 with a grid of 5 rows and 1 column

p1 <- ggplot(plot\_data, aes(x = long, y = lat, group = group, fill = avg\_pct\_bb)) +

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 = "gray85", high = "darkred") +

labs(title = "Average Percent of Broadband Access Per Region", fill = "Avg Density") +

theme\_void()

p2 <- ggplot(plot\_data, aes(x = long, y = lat, group = group, fill = avg\_pct\_college)) +

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 = "gray85", high = "darkred") +

labs(title = "Average Percent of College Graduates Per Region", fill = "Avg Percent") +

theme\_void()

p3 <- ggplot(plot\_data, aes(x = long, y = lat, group = group, fill = avg\_pct\_foreign\_born)) +

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 = "gray85", high = "darkred") +

labs(title = "Average Percent of Foreign-Born Population Per Region", fill = "Avg Percent") +

theme\_void()

p4 <- ggplot(plot\_data, aes(x = long, y = lat, group = group, fill = avg\_pct\_it\_workers)) +

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 = "gray85", high = "darkred") +

labs(title = "Average Percent of IT Workers Per Region", fill = "Avg Percent") +

theme\_void()

p5 <- ggplot(plot\_data, aes(x = long, y = lat, group = group, fill = avg\_median\_hh\_inc)) +

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,

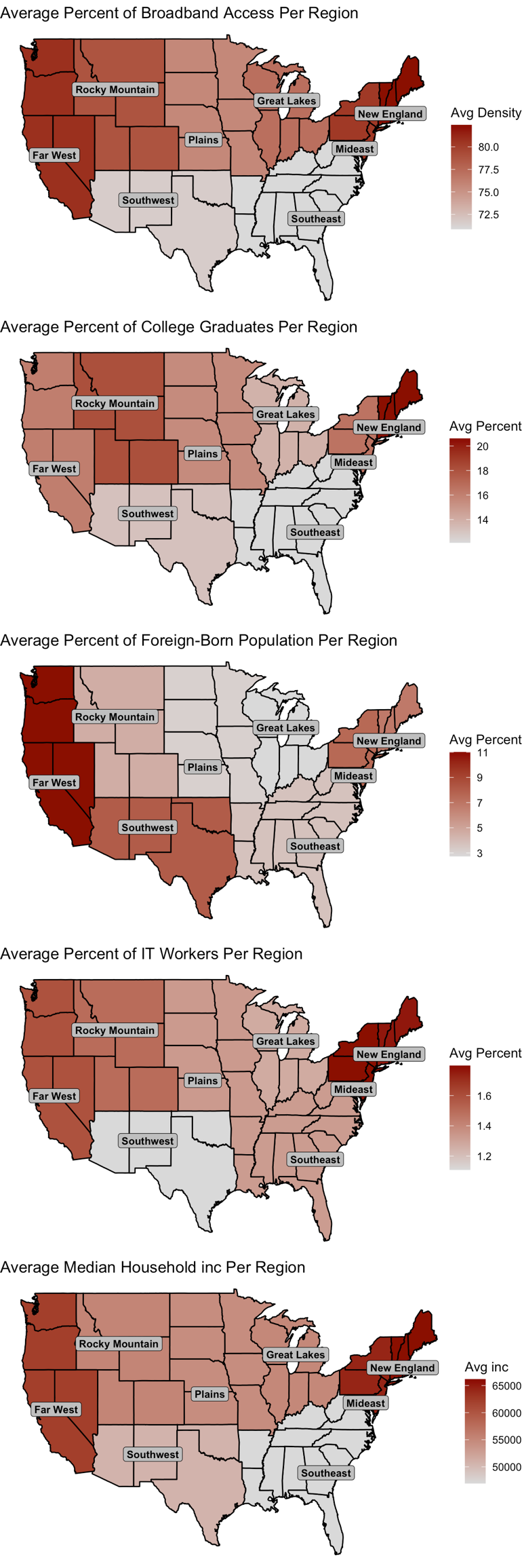
fill = "gray75", color = "black") +

scale\_fill\_gradient(low = "gray85", high = "darkred") +

labs(title = "Average Median Household inc Per Region", fill = "Avg inc") +

theme\_void()

grid.arrange(p1, p2, p3, p4, p5, nrow = 5)



state\_avg <- aggregate(cbind(pct\_bb, pct\_college, pct\_foreign\_born, pct\_it\_workers, median\_hh\_inc) ~ state, data = merged\_df, FUN = mean)

#Load US state 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 = c("region"), by.y = c("state")) %>%

arrange(order)

#Create a grid of heatmaps for each variable

grid\_arrange\_shared\_legend <- function(...) {

plots <- list(...)

g <- ggplotGrob(plots[[1]] + theme(legend.position="bottom"))$grobs

legend <- g[[which(sapply(g, function(x) x$name) == "guide-box")]]

lheight <- sum(legend$height)

grid.arrange(

do.call(arrangeGrob, lapply(plots, function(x)

x + theme(legend.position="none") + theme(panel.background = element\_rect(fill = "gray85", color = NA)))

),

bottom = legend,

ncol = 5,

heights = rep((unit(1, "npc") - lheight) / length(plots), length(plots))

)

}

heatmap\_bb <- ggplot(map\_data, aes(x = long, y = lat, group = group, fill = pct\_bb)) +

geom\_polygon() +

scale\_fill\_gradient(low = "white", high = "darkgreen") +

coord\_map() +

labs(title = "Percent of Broadband Access per State", fill = "Percent") +

theme\_void()

heatmap\_college <-

ggplot(map\_data, aes(x = long, y = lat, group = group, fill = pct\_college)) +

geom\_polygon() +

scale\_fill\_gradient(low = "white", high = "darkgreen") +

coord\_map() +

labs(title = "Percent of Population with College Education per State", fill = "Percent") +

theme\_void()

heatmap\_foreign\_born <-

ggplot(map\_data, aes(x = long, y = lat, group = group, fill = pct\_foreign\_born)) +

geom\_polygon() +

scale\_fill\_gradient(low = "white", high = "darkgreen") +

coord\_map() +

labs(title = "Percent of Foreign-born Population per State", fill = "Percent") +

theme\_void()

heatmap\_it\_workers <-

ggplot(map\_data, aes(x = long, y = lat, group = group, fill = pct\_it\_workers)) +

geom\_polygon() +

scale\_fill\_gradient(low = "white", high = "darkgreen") +

coord\_map() +

labs(title = "Percent of IT Workers per State", fill = "Percent") +

theme\_void()

heatmap\_inc <-

ggplot(map\_data, aes(x = long, y = lat, group = group, fill = median\_hh\_inc)) +

geom\_polygon() +

scale\_fill\_gradient(low = "white", high = "darkgreen") +

coord\_map() +

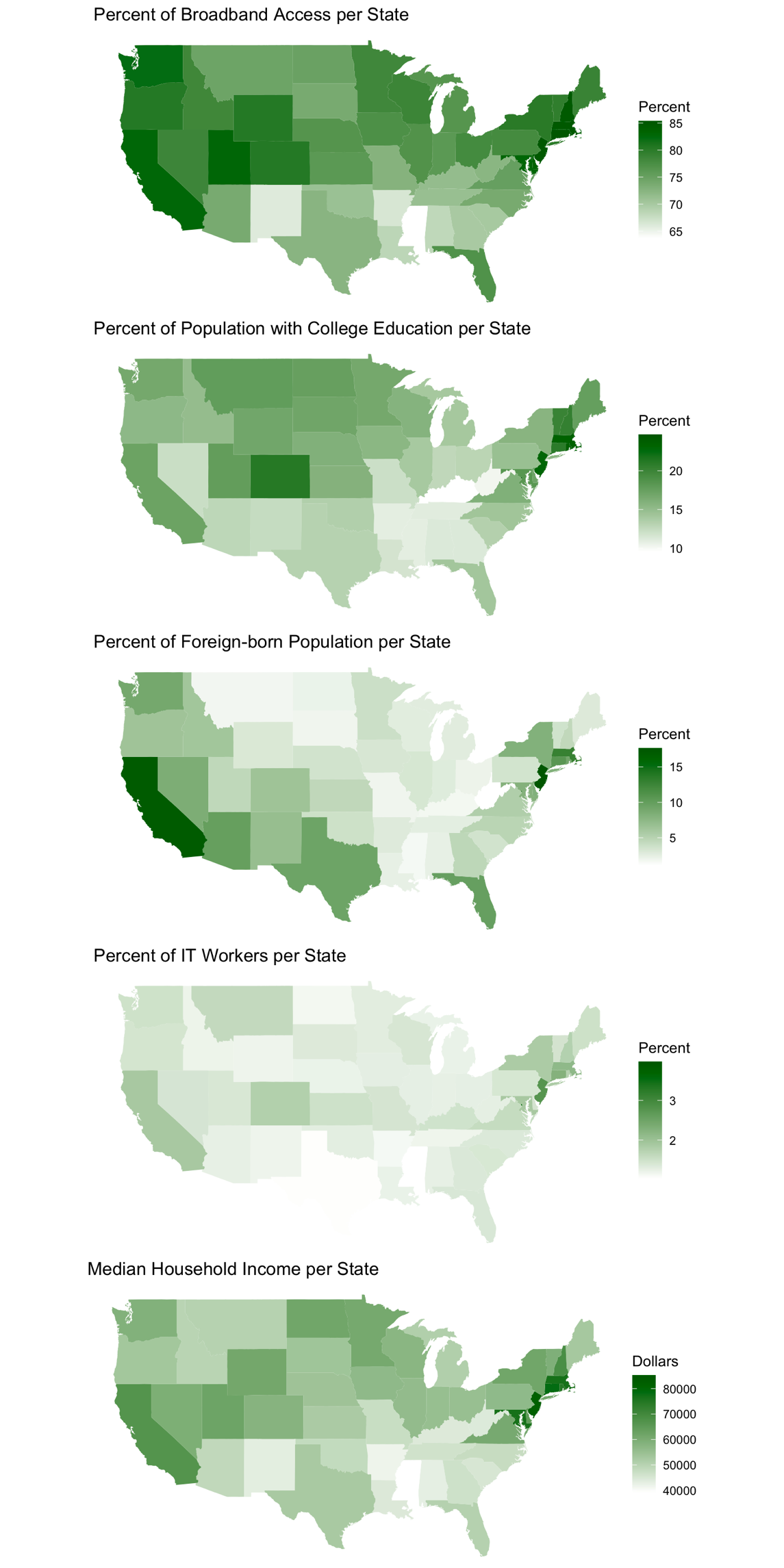
labs(title = "Median Household Income per State", fill = "Dollars") +

theme\_void()

#Plot the grid of heatmaps

#grid\_arrange\_shared\_legend(heatmap\_bb, heatmap\_college, heatmap\_foreign\_born, heatmap\_it\_workers, heatmap\_inc)

grid.arrange(heatmap\_bb, heatmap\_college, heatmap\_foreign\_born, heatmap\_it\_workers, heatmap\_inc, nrow = 5)



county\_avg <- aggregate(cbind(pct\_bb, pct\_college, pct\_foreign\_born, pct\_it\_workers, median\_hh\_inc) ~ county + state, data = merged\_df, FUN = mean)

#Get rid of county, city, and parish in the end of county names

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 grid of heatmaps for each variable

grid\_arrange\_shared\_legend <- function(...) {

plots <- list(...)

g <- ggplotGrob(plots[[1]] + theme(legend.position="bottom"))$grobs

legend <- g[[which(sapply(g, function(x) x$name) == "guide-box")]]

lheight <- sum(legend$height)

grid.arrange(

do.call(arrangeGrob, lapply(plots, function(x)

x + theme(legend.position="none") + theme(panel.background = element\_rect(fill = "gray85", color = NA)))

),

bottom = legend,

ncol = 5,

heights = rep((unit(1, "npc") - lheight) / length(plots), length(plots))

)

}

heatmap\_bb <- ggplot(map\_data, aes(x = long, y = lat, group = group, fill = pct\_bb)) +

geom\_polygon() +

scale\_fill\_gradient(low = "lightblue", high = "navyblue") +

coord\_map() +

labs(title = "Percent of Broadband Access per County", fill = "Percent") +

theme\_void()

heatmap\_college <-

ggplot(map\_data, aes(x = long, y = lat, group = group, fill = pct\_college)) +

geom\_polygon() +

scale\_fill\_gradient(low = "lightblue", high = "navyblue") +

coord\_map() +

labs(title = "Percent of Population with College Education per County", fill = "Percent") +

theme\_void()

heatmap\_foreign\_born <-

ggplot(map\_data, aes(x = long, y = lat, group = group, fill = pct\_foreign\_born)) +

geom\_polygon() +

scale\_fill\_gradient(low = "lightblue", high = "navyblue") +

coord\_map() +

labs(title = "Percent of Foreign-born Population per County", fill = "Percent") +

theme\_void()

heatmap\_it\_workers <-

ggplot(map\_data, aes(x = long, y = lat, group = group, fill = pct\_it\_workers)) +

geom\_polygon() +

scale\_fill\_gradient(low = "lightblue", high = "navyblue") +

coord\_map() +

labs(title = "Percent of IT Workers per County", fill = "Percent") +

theme\_void()

heatmap\_inc <-

ggplot(map\_data, aes(x = long, y = lat, group = group, fill = median\_hh\_inc)) +

geom\_polygon() +

scale\_fill\_gradient(low = "lightblue", high = "navyblue") +

coord\_map() +

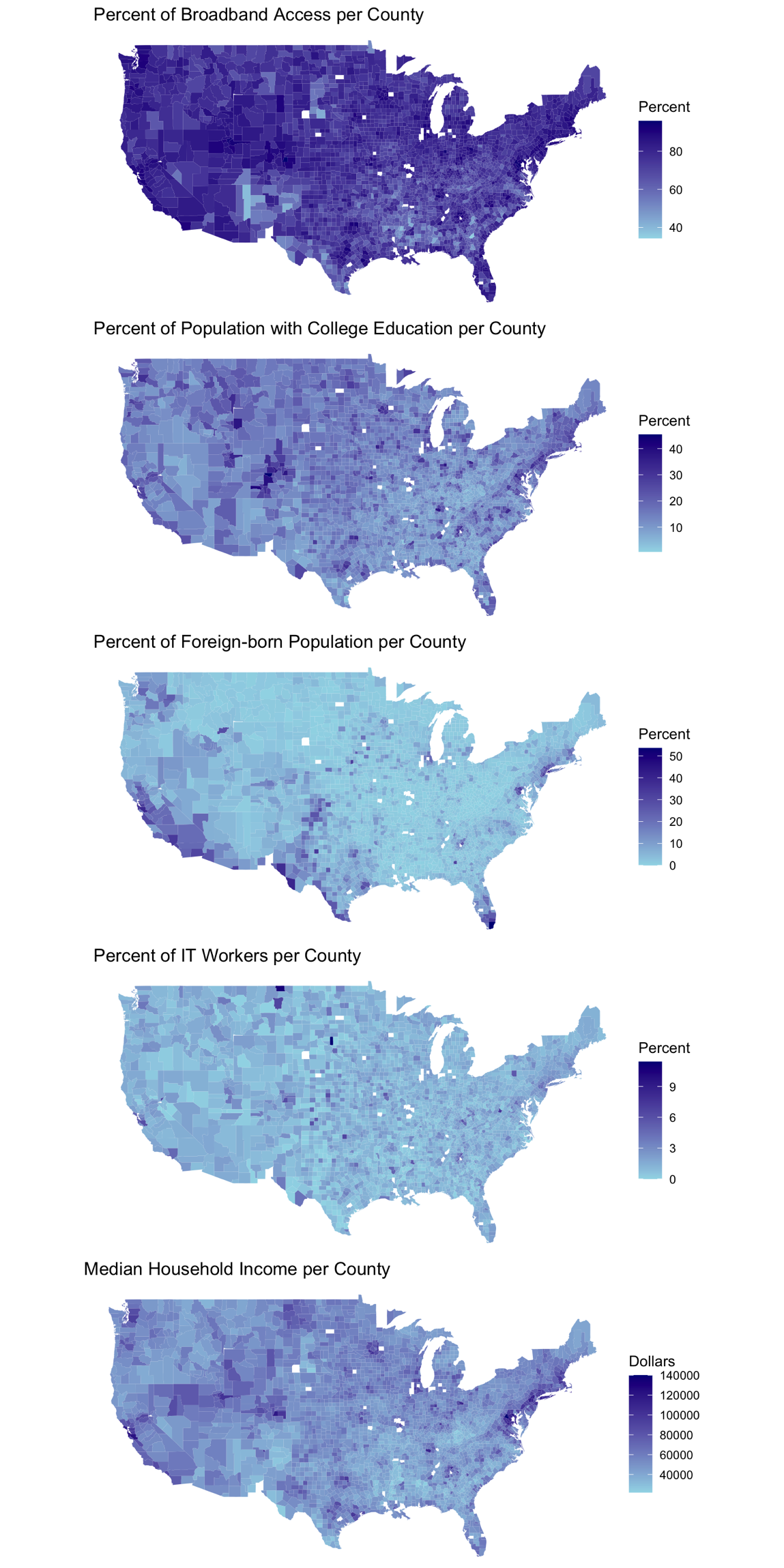
labs(title = "Median Household Income per County", fill = "Dollars") +

theme\_void()

#Plot the grid of heatmaps

#grid\_arrange\_shared\_legend(heatmap\_bb, heatmap\_college, heatmap\_foreign\_born, heatmap\_it\_workers, heatmap\_inc)

grid.arrange(heatmap\_bb, heatmap\_college, heatmap\_foreign\_born, heatmap\_it\_workers, heatmap\_inc, nrow = 5)



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")

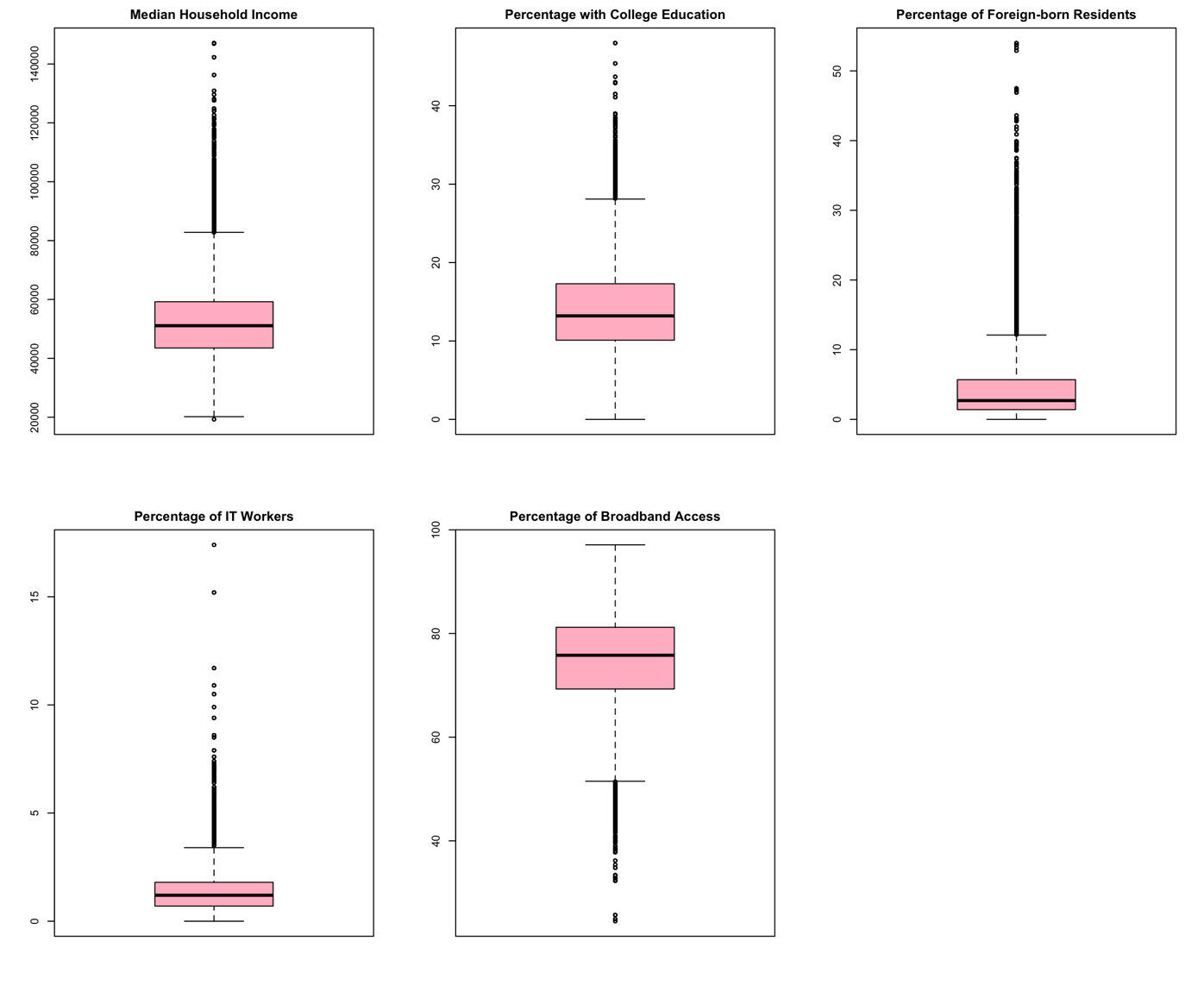
boxplot(merged\_df$pct\_college, col = "pink", main = "Percentage with College Education")

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, col = "pink", main = "Percentage of Broadband Access")

#boxplot(merged\_df$active, col = "pink", main = "Active Microbusiness Count")



## 3.8. Outlier Detection

Outlier detection is an important step in data analysis, as outliers can significantly affect the results of statistical analyses. 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).

# Sample 5000 observations from microbusiness\_density column

set.seed(92) # Set seed for reproducibility

sample\_data <- sample(merged\_df$microbusiness\_density, 5000)

# Perform Shapiro-Wilk test on sample\_data

shapiro.test(sample\_data)

##

## 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.

# 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 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

density\_df <- data.frame(x = x\_values, y = y\_values)

# Create a boxplot

boxplot <- ggplot(data = merged\_df, aes(x = "", y = merged\_df$microbusiness\_density)) +

geom\_boxplot(fill = "skyblue") +

scale\_y\_log10() +

labs(x = "", y = "Microbusiness Density (logarithmic)") +

ggtitle("Boxplot for Microbusiness Density")

# Create a Q-Q plot and add a diagonal line

qqplot <- ggplot(data = merged\_df, aes(sample = microbusiness\_density)) +

stat\_qq() +

stat\_qq\_line(colour = "red") +

labs(x = "Theoretical Normal Quantiles", y = "Observed Quantiles") +

ggtitle("Q-Q Plot for Microbusiness Density")

# Create a histogram and add the bell curve

densityplot <- ggplot(data = merged\_df, aes(x = microbusiness\_density)) +

geom\_histogram(aes(y = after\_stat(density)), bins = 30, colour = "black", fill = "white") +

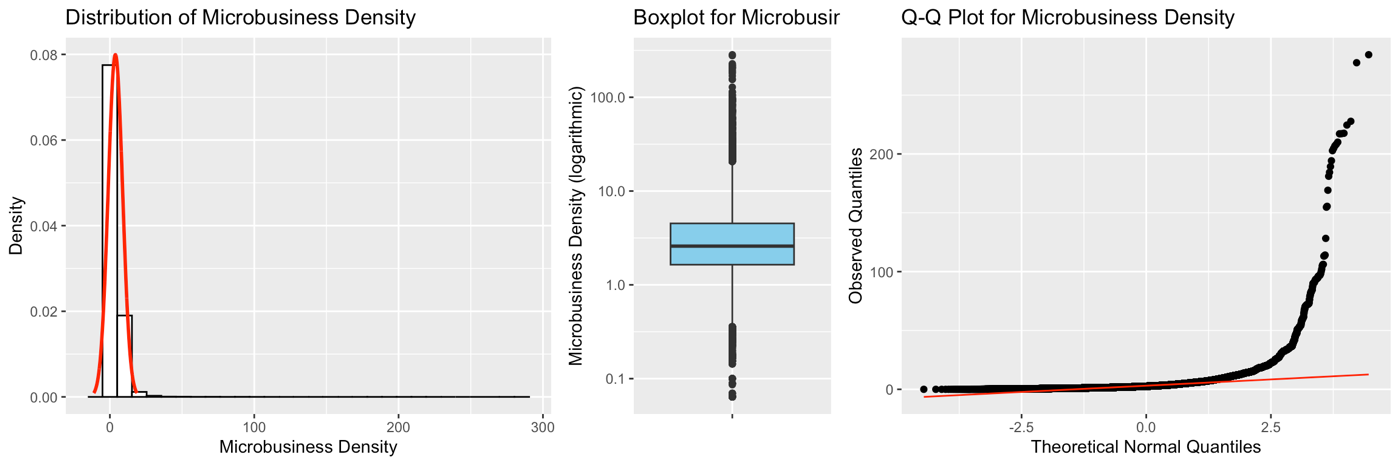
geom\_line(data = density\_df, aes(x = x, y = y), colour = "red", linewidth = 1) +

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))



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.

Since the data is right-skewed and not normally distributed, a common approach to detecting outliers is to use the interquartile range (IQR) method. Now, we have to find the boundary of minimum and maximum values, out of which data would be considered an outlier.

The decision range approach involves setting a range of values outside of which any observations are considered outliers. 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 IQR of microbusiness\_density column

q <- quantile(merged\_df$microbusiness\_density, c(0.25, 0.75))

iqr <- q[2] - q[1]

# Calculate lower and upper bounds for outliers

lower\_bound <- q[1] - 1.5\*iqr

upper\_bound <- q[2] + 1.5\*iqr

# Count number of outliers

num\_outliers <- sum(merged\_df$microbusiness\_density < lower\_bound | merged\_df$microbusiness\_density > upper\_bound)

# Calculate percent of outliers

percent\_outliers <- num\_outliers / length(merged\_df$microbusiness\_density) \* 100

# Print results

cat("Number of outliers:", num\_outliers, "\n")

## Number of outliers: 8746

cat("Percent of outliers:", percent\_outliers, "%\n")

## Percent of outliers: 7.153315 %

# Create new dataframe without outliers

merged\_df\_new <- merged\_df[merged\_df$microbusiness\_density >= lower\_bound & merged\_df$microbusiness\_density <= upper\_bound,]

# Print number of rows removed

cat("Number of rows removed:", nrow(merged\_df) - nrow(merged\_df\_new), "\n")

## Number of rows removed: 8746

# Calculate the mean and standard deviation of 'microbusiness\_density'

mean\_density <- mean(merged\_df\_new$microbusiness\_density)

sd\_density <- sd(merged\_df\_new$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 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

density\_df <- data.frame(x = x\_values, y = y\_values)

# Create a boxplot

boxplot <- ggplot(data = merged\_df\_new, aes(x = "", y = merged\_df\_new$microbusiness\_density)) +

geom\_boxplot(fill = "skyblue") +

# scale\_y\_log10() +

labs(x = "", y = "Microbusiness Density") +

ggtitle("Boxplot for Microbusiness Density")

# Create a Q-Q plot and add a diagonal line

qqplot <- ggplot(data = merged\_df\_new, aes(sample = microbusiness\_density)) +

stat\_qq() +

stat\_qq\_line(colour = "red") +

labs(x = "Theoretical Normal Quantiles", y = "Observed Quantiles") +

ggtitle("Q-Q Plot for Microbusiness Density")

# Create a bell curve and add the bell curve

densityplot <- ggplot(data = merged\_df\_new, aes(x = microbusiness\_density)) +

geom\_histogram(aes(y = after\_stat(density)), bins = 30, colour = "black", fill = "white") +

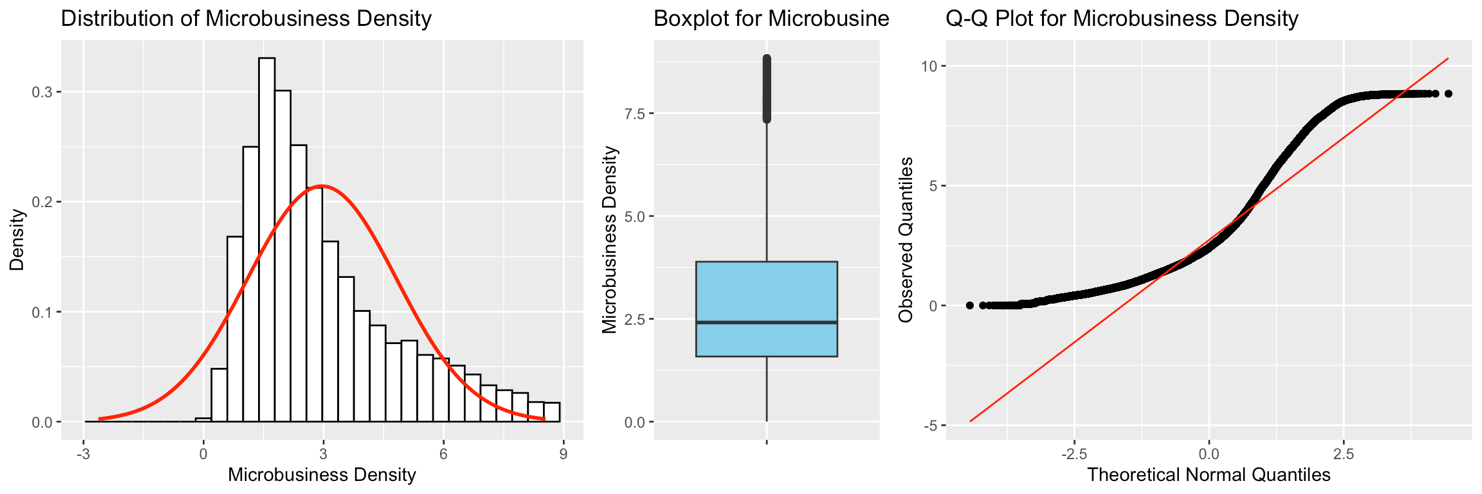
geom\_line(data = density\_df, aes(x = x, y = y), colour = "red", linewidth = 1) +

labs(x = "Microbusiness Density", y = "Density") +

ggtitle("Distribution of Microbusiness Density")

# Arrange the plots in one row

grid.arrange(densityplot, boxplot, qqplot, ncol = 3, widths = c(2, 1, 2))



# Calculate IQR of microbusiness\_density column

q <- quantile(merged\_df\_new$microbusiness\_density, c(0.25, 0.75))

iqr <- q[2] - q[1]

# Calculate lower and upper bounds for outliers

lower\_bound <- q[1] - 1.5\*iqr

upper\_bound <- q[2] + 1.5\*iqr

# Count number of outliers

num\_outliers <- sum(merged\_df\_new$microbusiness\_density < lower\_bound | merged\_df\_new$microbusiness\_density > upper\_bound)

# Calculate percent of outliers

percent\_outliers <- num\_outliers / length(merged\_df\_new$microbusiness\_density) \* 100

# Print results

cat("Number of outliers:", num\_outliers, "\n")

## Number of outliers: 3946

cat("Percent of outliers:", percent\_outliers, "%\n")

## Percent of outliers: 3.47607 %

# Create new dataframe without outliers

merged\_df\_clean <- merged\_df\_new[merged\_df\_new$microbusiness\_density >= lower\_bound & merged\_df\_new$microbusiness\_density <= upper\_bound,]

# Print number of rows removed

cat("Number of rows removed:", nrow(merged\_df\_new) - nrow(merged\_df\_clean), "\n")

## Number of rows removed: 3946

cat("Total rows removed:", nrow(merged\_df) - nrow(merged\_df\_clean))

## Total rows removed: 12692

# Calculate the mean and standard deviation of 'microbusiness\_density'

mean\_density <- mean(merged\_df\_clean$microbusiness\_density)

sd\_density <- sd(merged\_df\_clean$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 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

density\_df <- data.frame(x = x\_values, y = y\_values)

# Create a boxplot

boxplot <- ggplot(data = merged\_df\_clean, aes(x = "", y = merged\_df\_clean$microbusiness\_density)) +

geom\_boxplot(fill = "skyblue") +

# scale\_y\_log10() +

labs(x = "", y = "Microbusiness Density") +

ggtitle("Boxplot for Microbusiness Density")

# Create a Q-Q plot and add a diagonal line

qqplot <- ggplot(data = merged\_df\_clean, aes(sample = microbusiness\_density)) +

stat\_qq() +

stat\_qq\_line(colour = "red") +

labs(x = "Theoretical Normal Quantiles", y = "Observed Quantiles") +

ggtitle("Q-Q Plot for Microbusiness Density")

# Create a histogram and add the bell curve

densityplot <- ggplot(data = merged\_df\_clean, aes(x = microbusiness\_density)) +

geom\_histogram(aes(y = after\_stat(density)), bins = 30, colour = "black", fill = "white") +

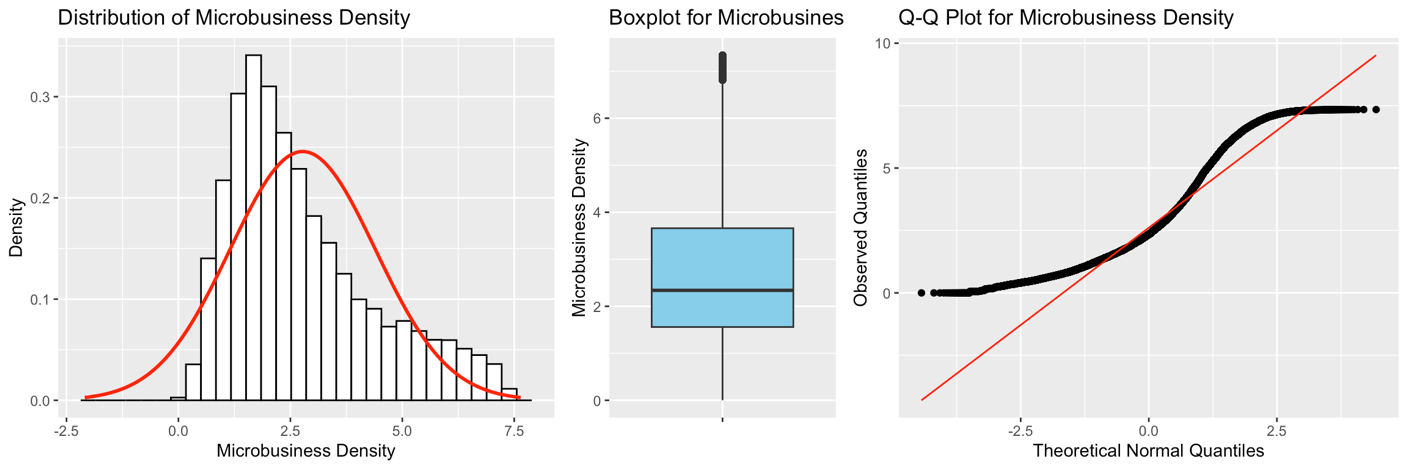
geom\_line(data = density\_df, aes(x = x, y = y), colour = "red", linewidth = 1) +

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))



# Calculate the mean and standard deviation of 'pct\_bb'

mean\_pct\_bb <- mean(merged\_df$pct\_bb)

sd\_pct\_bb <- sd(merged\_df$pct\_bb)

# Create a range of values for the x-axis

x\_values <- seq(mean\_pct\_bb - 3\*sd\_pct\_bb, mean\_pct\_bb + 3\*sd\_pct\_bb, length.out = 1000)

# Create a bell curve with mean and standard deviation calculated above

y\_values <- dnorm(x\_values, mean = mean\_pct\_bb, sd = sd\_pct\_bb)

# Combine the 'x\_values' and 'y\_values' into a data frame

pct\_bb\_df <- data.frame(x = x\_values, y = y\_values)

# Create a boxplot

boxplot <- ggplot(data = merged\_df, aes(x = "", y = merged\_df$pct\_bb)) +

geom\_boxplot(fill = "skyblue") +

scale\_y\_log10() +

labs(x = "", y = "Microbusiness pct\_bb (logarithmic)") +

ggtitle("Boxplot for Microbusiness pct\_bb")

# Create a Q-Q plot and add a diagonal line

qqplot <- ggplot(data = merged\_df, aes(sample = pct\_bb)) +

stat\_qq() +

stat\_qq\_line(colour = "red") +

labs(x = "Theoretical Normal Quantiles", y = "Observed Quantiles") +

ggtitle("Q-Q Plot for Microbusiness pct\_bb")

# Create a histogram and add the bell curve

densityplot <- ggplot(data = merged\_df, aes(x = pct\_bb)) +

geom\_histogram(aes(y = after\_stat(density)), bins = 30, colour = "black", fill = "white") +

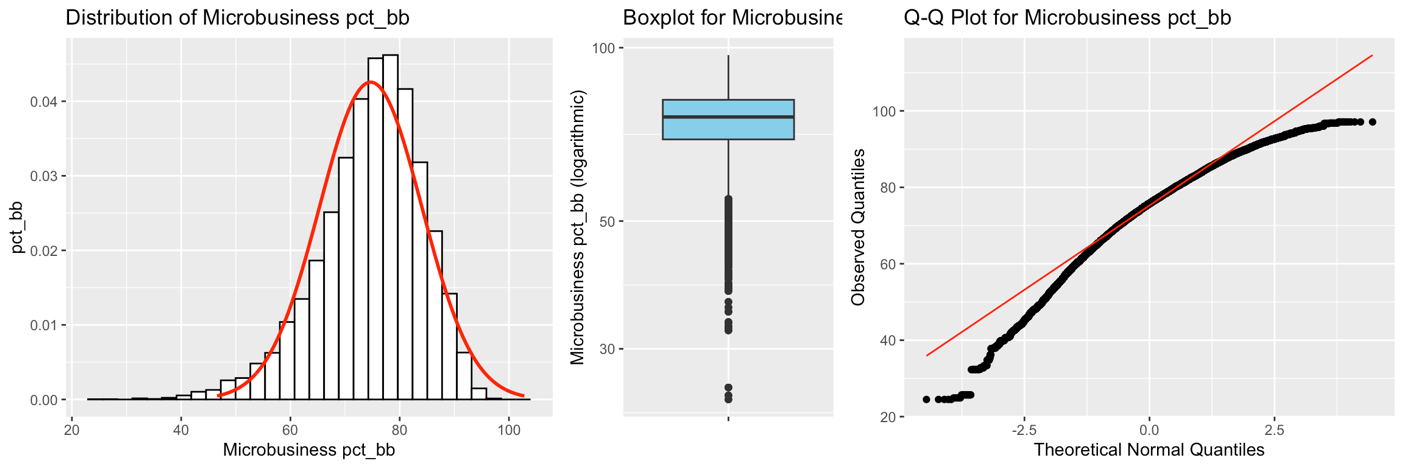
geom\_line(data = pct\_bb\_df, aes(x = x, y = y), colour = "red", linewidth = 1) +

labs(x = "Microbusiness pct\_bb", y = "pct\_bb") +

ggtitle("Distribution of Microbusiness pct\_bb")

# 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))



# Specify the columns to analyze

cols\_to\_analyze <- c("pct\_bb", "pct\_college", "pct\_it\_workers", "pct\_foreign\_born", "median\_hh\_inc")

# Loop through each column and detect outliers using the IQR method

for (col in cols\_to\_analyze) {

# Calculate the interquartile range (IQR) of the column

q1 <- quantile(merged\_df\_new[[col]], 0.25)

q3 <- quantile(merged\_df\_new[[col]], 0.75)

iqr <- q3 - q1

# Calculate the upper and lower bounds for outliers

upper\_bound <- q3 + (1.5 \* iqr)

lower\_bound <- q1 - (1.5 \* iqr)

# Identify the outliers in the column

outliers <- merged\_df\_new[[col]][merged\_df\_new[[col]] > upper\_bound | merged\_df\_new[[col]] < lower\_bound]

# Print the results

# cat(paste("Outliers in", col, ":", toString(outliers), "\n"))

# Count number of outliers

num\_outliers <- sum(merged\_df\_new[[col]] < lower\_bound | merged\_df\_new[[col]] > upper\_bound)

# Calculate percent of outliers

percent\_outliers <- num\_outliers / length(merged\_df\_new[[col]]) \* 100

# Print results

cat("Number of outliers in", col,":", num\_outliers, "\n")

cat("Percent of outliers in", col,":", percent\_outliers, "%\n")

}

## 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 %

# Create a copy of the original dataframe

merged\_df\_clean <- merged\_df\_new

# Loop through each column and detect outliers using the IQR method

for (col in cols\_to\_analyze) {

# Calculate the interquartile range (IQR) of the column

q1 <- quantile(merged\_df\_clean[[col]], 0.25)

q3 <- quantile(merged\_df\_clean[[col]], 0.75)

iqr <- q3 - q1

# Calculate the upper and lower bounds for outliers

upper\_bound <- q3 + (1.5 \* iqr)

lower\_bound <- q1 - (1.5 \* iqr)

# Identify the outliers in the column

outliers <- merged\_df\_clean[[col]][merged\_df\_clean[[col]] > upper\_bound | merged\_df\_clean[[col]] < lower\_bound]

# Remove the outliers from the dataframe

merged\_df\_clean <- merged\_df\_clean[!(merged\_df\_clean[[col]] %in% outliers), ]

}

# Print the dimensions of the original and cleaned dataframes

cat("Original dataframe dimensions:", nrow(merged\_df\_new), "\n")

## Original dataframe dimensions: 113519

cat("Cleaned dataframe dimensions:", nrow(merged\_df\_clean), "\n")

## Cleaned dataframe dimensions: 97098

cat("Total rows removed:", nrow(merged\_df) - nrow(merged\_df\_clean), "\n")

## Total rows removed: 25167

cat("Total percent removed:", 100 \* (nrow(merged\_df) - nrow(merged\_df\_clean)) / nrow(merged\_df),"%", "\n")

## Total percent removed: 20.58398 %

## 3.9. Non-Parametric Test

# Specify a list of column pairs to compare

col\_pairs <- list(c("microbusiness\_density", "pct\_bb"), c("microbusiness\_density", "pct\_college"), c("microbusiness\_density", "pct\_foreign\_born"), c("microbusiness\_density", "pct\_it\_workers"), c("microbusiness\_density", "median\_hh\_inc"))

# Loop through each column pair and perform the Wilcoxon signed-rank test

for (pair in col\_pairs) {

test\_result <- wilcox.test(merged\_df\_new[[pair[1]]], merged\_df\_new[[pair[2]]], paired = TRUE)

cat(paste("Wilcoxon signed-rank test results for", pair[1], "and", pair[2], ":\n"))

print(test\_result)

cat("\n")

}

## 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 = 5996103713, 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

# Specify a list of column pairs to compare

col\_pairs <- list(c("microbusiness\_density", "pct\_bb"), c("microbusiness\_density", "pct\_college"), c("microbusiness\_density", "pct\_foreign\_born"), c("microbusiness\_density", "pct\_it\_workers"), c("microbusiness\_density", "median\_hh\_inc"))

# Loop through each column pair and perform the Wilcoxon rank sum test

for (pair in col\_pairs) {

test\_result <- wilcox.test(merged\_df\_new[[pair[1]]], merged\_df\_new[[pair[2]]], paired = FALSE)

cat(paste("Wilcoxon rank-sum test results for", pair[1], "and", pair[2], ":\n"))

print(test\_result)

cat("\n")

}

## 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.

The 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.

kernel regression

# create a vector of region names and their corresponding codes

region\_codes <- c("new england" = 1,

"mideast" = 2,

"great lakes" = 3,

"plains" = 4,

"southeast" = 5,

"southwest" = 6,

"rocky mountain" = 7,

"far west" = 8)

# use the `match()` function to find the region code for each row in `merged\_df\_new$region`

merged\_df\_new$region\_code <- match(merged\_df\_new$region, names(region\_codes))

merged\_df\_clean$region\_code <- match(merged\_df\_clean$region, names(region\_codes))

# create a vector of state names and their corresponding codes

state\_codes <- sort(unique(merged\_df\_new$state))

state\_codes <- setNames(1:length(state\_codes), state\_codes)

# use the `match()` function to find the state code for each row in `merged\_df\_new$state`

merged\_df\_new$state\_code <- match(merged\_df\_new$state, names(state\_codes))

state\_codes <- sort(unique(merged\_df\_clean$state))

state\_codes <- setNames(1:length(state\_codes), state\_codes)

merged\_df\_clean$state\_code <- match(merged\_df\_clean$state, names(state\_codes))

# print head of the updated dataframe

head(merged\_df\_new)

## 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 year\_month year month pct\_bb pct\_college

## 1 3.007682 1249 2019-08 2019 8 76.6 14.5

## 2 2.884870 1198 2019-09 2019 9 76.6 14.5

## 3 3.055843 1269 2019-10 2019 10 76.6 14.5

## 4 2.993233 1243 2019-11 2019 11 76.6 14.5

## 5 2.993233 1243 2019-12 2019 12 76.6 14.5

## 6 2.969090 1242 2020-01 2020 1 78.9 15.9

## pct\_foreign\_born pct\_it\_workers median\_hh\_inc region region\_code

## 1 2.1 1.3 55317 southeast 5

## 2 2.1 1.3 55317 southeast 5

## 3 2.1 1.3 55317 southeast 5

## 4 2.1 1.3 55317 southeast 5

## 5 2.1 1.3 55317 southeast 5

## 6 2.0 1.1 58786 southeast 5

## state\_code

## 1 1

## 2 1

## 3 1

## 4 1

## 5 1

## 6 1

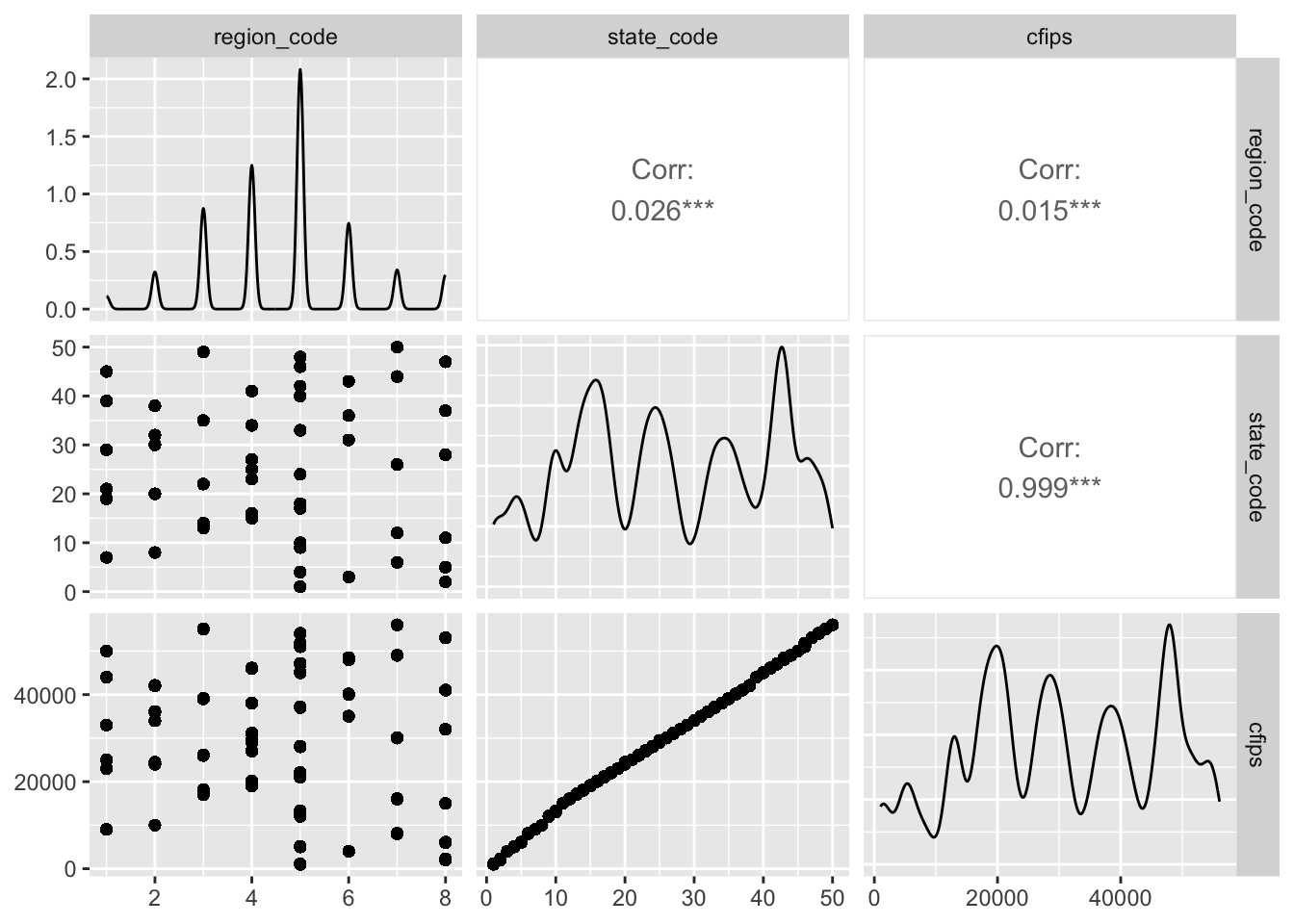
merged\_df\_new |>

GGally::ggpairs(columns = c(17,18,2))

## Registered S3 method overwritten by 'GGally':

## method from

## +.gg ggplot2



# Select the columns with numeric data

numeric\_cols <- c("cfips", "state\_code", "region\_code", "microbusiness\_density", "active", "year", "month", "pct\_bb", "pct\_college", "pct\_foreign\_born", "pct\_it\_workers", "median\_hh\_inc")

# Subset the dataframe with the selected columns

merged\_df\_numeric <- merged\_df\_new[, numeric\_cols]

# Calculate the correlation matrix

cor\_matrix <- cor(merged\_df\_numeric, use="pairwise.complete.obs")

# Plot the correlation matrix using ggcorrplot

ggcorrplot(cor\_matrix,

hc.order = TRUE,

type = "lower",

method = "square",

lab = TRUE,

lab\_size = 3,

title = "Correlation Plot of merged\_df\_new",

colors = c("#6D9EC1", "#FAC200", "#FA5252"),

ggtheme = ggplot2::theme\_gray,

show.legend = TRUE)



# Select the columns with numeric data

numeric\_cols <- c("cfips", "state\_code", "region\_code", "microbusiness\_density", "active", "year", "month", "pct\_bb", "pct\_college", "pct\_foreign\_born", "pct\_it\_workers", "median\_hh\_inc")

# Subset the dataframe with the selected columns

merged\_df\_numeric <- merged\_df\_clean[, numeric\_cols]

# Calculate the correlation matrix

cor\_matrix <- cor(merged\_df\_numeric, use="pairwise.complete.obs")

# Plot the correlation matrix using ggcorrplot

ggcorrplot(cor\_matrix,

hc.order = TRUE,

type = "lower",

method = "square",

lab = TRUE,

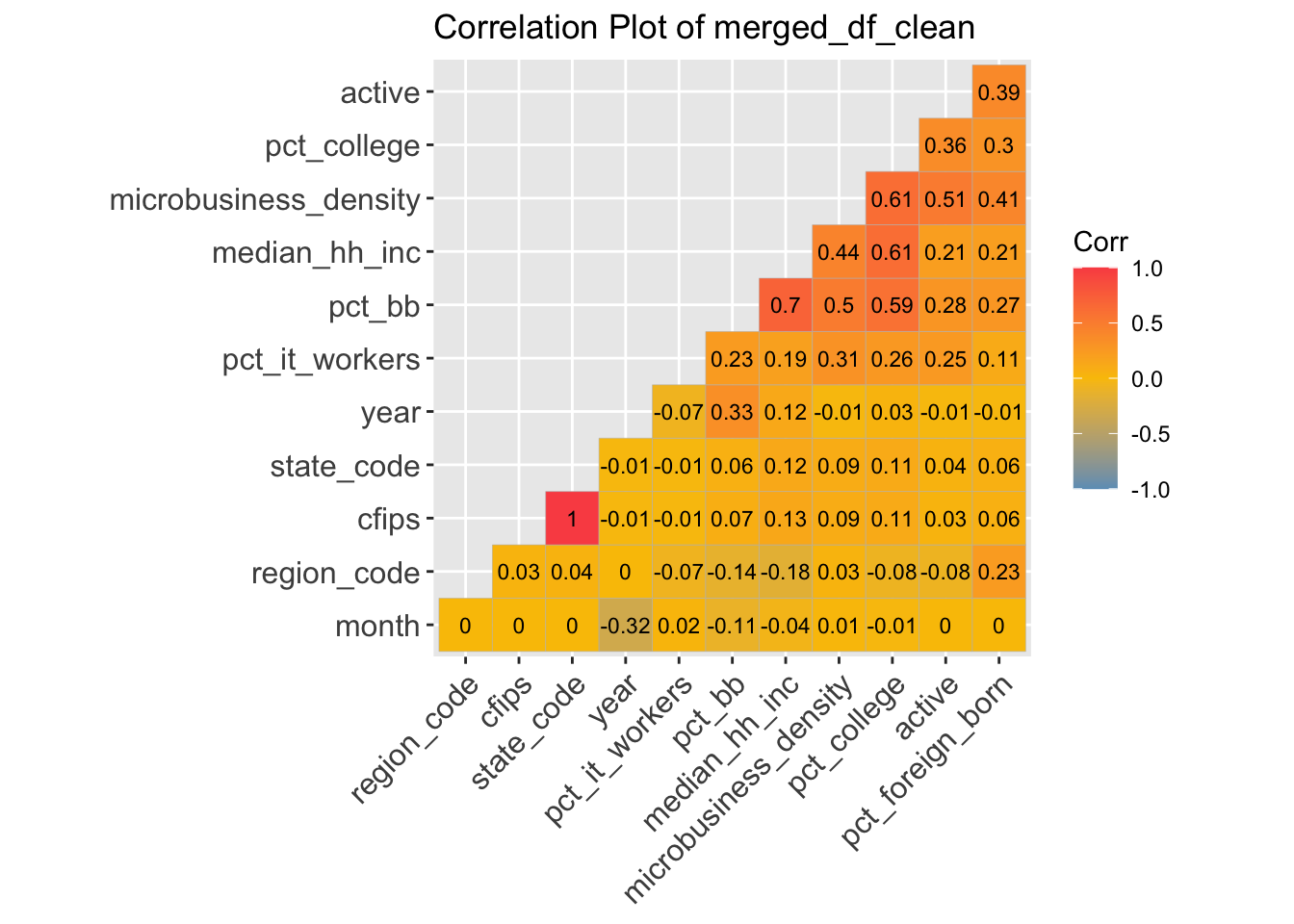
lab\_size = 3,

title = "Correlation Plot of merged\_df\_clean",

colors = c("#6D9EC1", "#FAC200", "#FA5252"),

ggtheme = ggplot2::theme\_gray,

show.legend = TRUE)



The cfips column and state\_code have a completely positive correlation. So, we will remove the redundant feature state\_code. Also, pct\_bb strongly correlates with median\_hh\_inc and pct\_college. Also, pct\_college strongly correlates with microbusiness\_density, median\_hh\_inc and pct\_bb. As features with moderate strong and strong correlation are redundant, we will remove these features (pct\_bb and pct\_college) before further analysis. Now, we draw the new correlation plot.

# Select the columns with numeric data (not including "state\_code", "pct\_bb", "pct\_college")

numeric\_cols <- c("cfips", "region\_code", "microbusiness\_density", "active", "year", "month", "pct\_foreign\_born", "pct\_it\_workers", "median\_hh\_inc")

# Subset the dataframe with the selected columns

merged\_df\_numeric <- merged\_df\_new[, numeric\_cols]

# Calculate the correlation matrix

cor\_matrix <- cor(merged\_df\_numeric, use="pairwise.complete.obs")

# Plot the correlation matrix using ggcorrplot

ggcorrplot(cor\_matrix,

hc.order = TRUE,

type = "lower",

method = "square",

lab = TRUE,

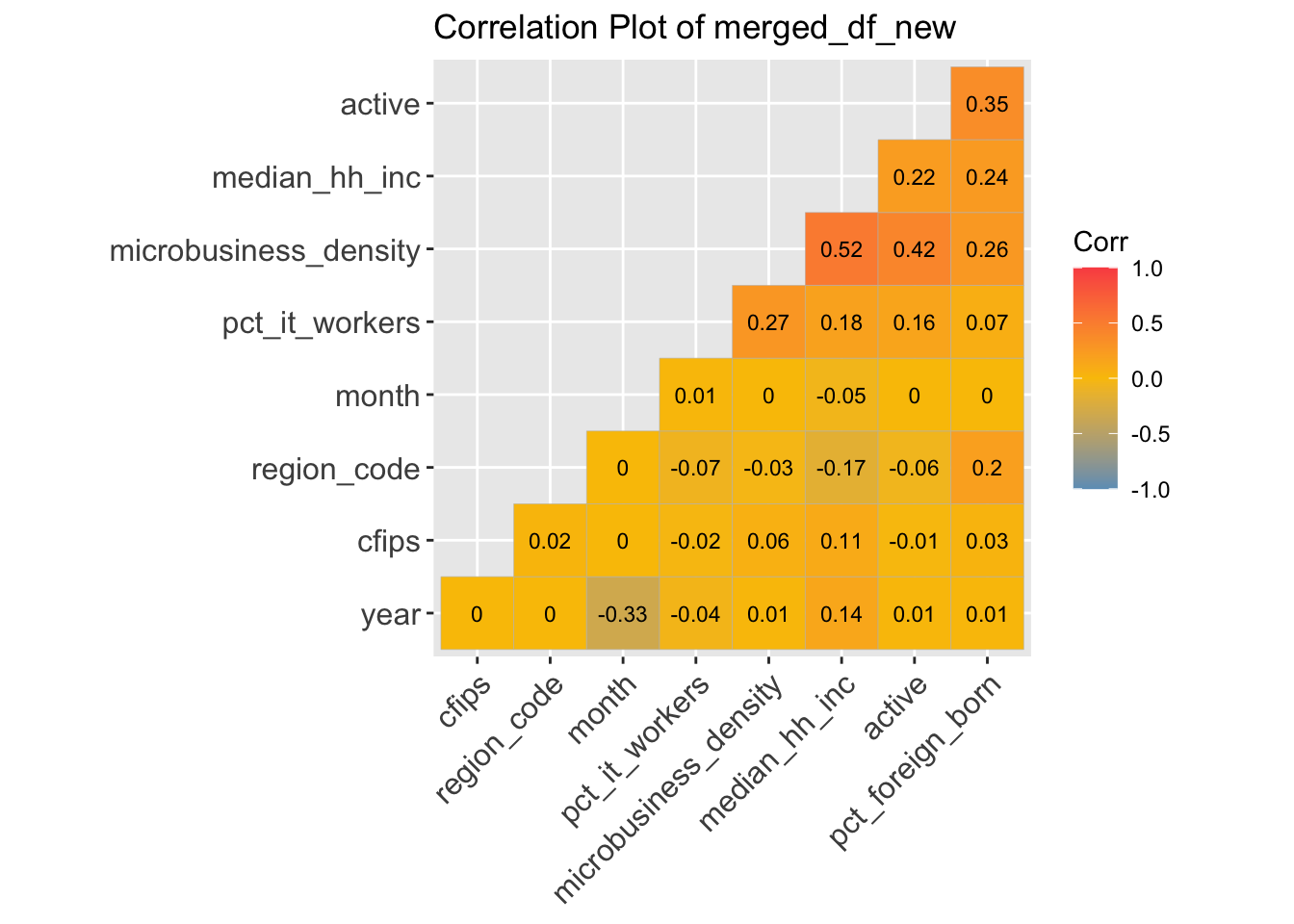
lab\_size = 3,

title = "Correlation Plot of merged\_df\_new",

colors = c("#6D9EC1", "#FAC200", "#FA5252"),

ggtheme = ggplot2::theme\_gray,

show.legend = TRUE)



## 3.10. Feature Importance Using Random Forest

# Split the dataset into training and testing sets

set.seed(92) # for reproducibility

train\_index <- sample(nrow(merged\_df\_new), 0.7 \* nrow(merged\_df\_new))

train\_data <- merged\_df\_new[train\_index, ]

test\_data <- merged\_df\_new[-train\_index, ]

# Build the random forest model

model <- randomForest(microbusiness\_density ~ cfips + region\_code + active + year + month + pct\_foreign\_born + pct\_it\_workers + median\_hh\_inc,

data = train\_data)

# Make predictions on the test set

predictions <- predict(model, test\_data)

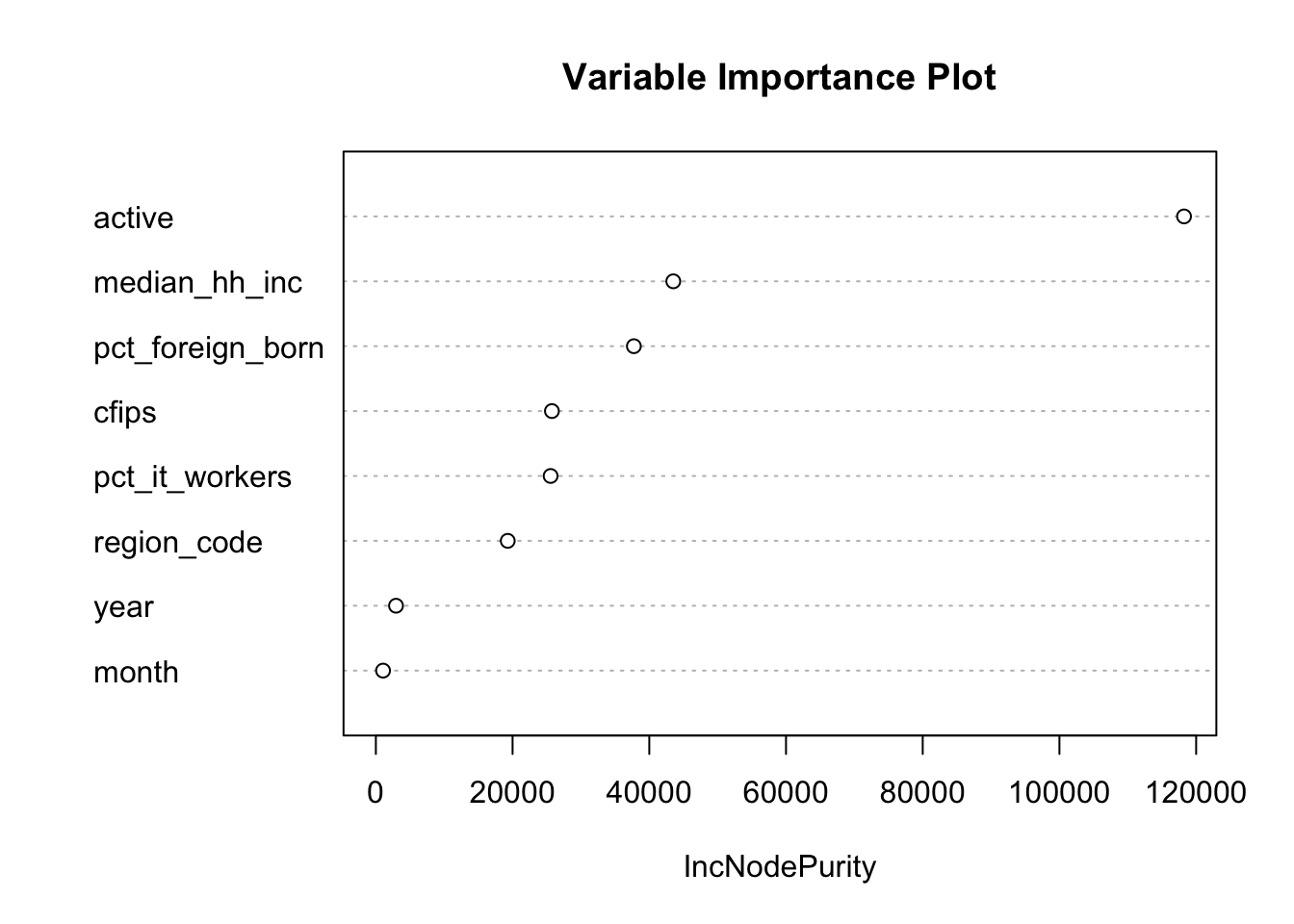
# Check the accuracy of the model

# accuracy <- sum(predictions == test\_data$microbusiness\_density) / nrow(test\_data)

# cat("Accuracy:", accuracy, "\n")

# Plot the variable importance

varImpPlot(model, main = "Variable Importance Plot")



# Load revealed\_test.csv into a dataframe

revealed\_test\_df <- read.csv("./datasets/revealed\_test.csv")

# Change first\_day\_of\_month format in "revealed\_test\_df" to Date

revealed\_test\_df$first\_day\_of\_month <- as.Date(revealed\_test\_df$first\_day\_of\_month)

# 4. Time Series Forecasting

## 4.1 Exponential Forecasting

Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight. This framework generates reliable forecasts quickly and for a wide range of time series, which is a great advantage and of major importance to applications in industry. ### 4.1.1. Simple Exponential Smoothing Forecast Exponential smoothing is a general technique for smoothing time series data by giving more weight to recent observations. The simplest of the exponentially smoothing methods is naturally called simple exponential smoothing (SES). This method is suitable for forecasting data with no clear trend or seasonal pattern. (There is a grow in the last few years, which might suggest a trend. We will consider whether a trended method would be better for this series later.)

# Select relevant predictor columns

# predictors <- merged\_df\_new[, c("microbusiness\_density", "cfips")]

merged\_df\_ts <- merged\_df\_clean %>%

group\_by(first\_day\_of\_month) %>%

summarise(microbusiness\_density = mean(microbusiness\_density))

# Convert data to time series format

ts\_data <- ts(merged\_df\_ts$microbusiness\_density, start = c(2019, 8), frequency = 12)

# fit a simple exponential smoothing model to the time series data

fit <- ets(ts\_data, model = "ANN")

# generate forecasts for the time series data using the fitted model

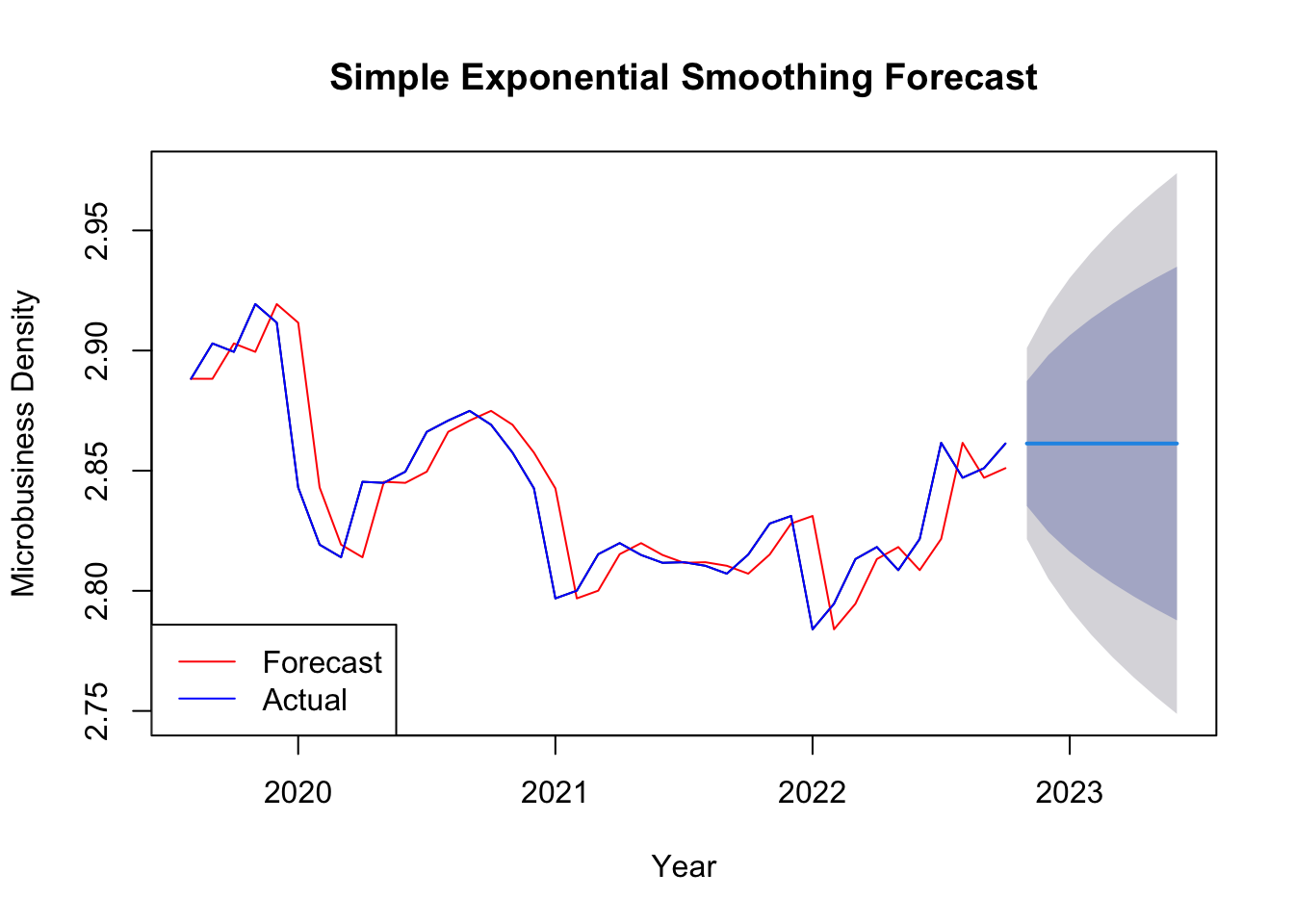
forecast\_values <- forecast::forecast(fit, h = 8)

plot(forecast\_values, main = "Simple Exponential Smoothing Forecast", xlab = "Year", ylab = "Microbusiness Density")

lines(forecast\_values$fitted, col = "red")

lines(ts\_data, col = "blue")

legend("bottomleft", legend = c("Forecast", "Actual"), col = c("red", "blue"), lty = 1)

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

# Split data into training and testing sets

train\_data <- window(ts\_data, end = c(2022, 2))

test\_data <- window(ts\_data, start = c(2022, 3))

# Train neural network autoregression model

nnetar\_model <- nnetar(ts\_data)

# Make forecasts using the trained model

forecast\_data <- forecast::forecast(nnetar\_model, h = length(test\_data))

# Convert data to time series format

ts\_data <- ts(merged\_df\_ts$microbusiness\_density, start = c(2019, 8), frequency = 12)

# Split data into training and testing sets

train\_data <- window(ts\_data, end = c(2022, 2))

test\_data <- window(ts\_data, start = c(2022, 3))

# Fit a neural network autoregression model with error("A"), trend("N"), season("N") preferences

nnetar\_model <- nnetar(train\_data, lambda = 0, P = 12, size = 10, repeats = 10,

decay = 0.5, maxit = 1000, trace = TRUE,

entropy = FALSE, parallel = FALSE, xreg = NULL)

## # weights: 41

## initial value 21.283645

## iter 10 value 11.004232

## iter 20 value 10.950528

## iter 30 value 10.940432

## iter 40 value 10.939499

## final value 10.939493

## converged

## # weights: 41

## initial value 59.692453

## iter 10 value 11.964849

## iter 20 value 10.974613

## iter 30 value 10.949670

## iter 40 value 10.945596

## iter 50 value 10.941198

## iter 60 value 10.939534

## final value 10.939493

## converged

## # weights: 41

## initial value 70.392818

## iter 10 value 11.771520

## iter 20 value 10.980551

## iter 30 value 10.943868

## iter 40 value 10.939699

## iter 50 value 10.939518

## final value 10.939493

## converged

## # weights: 41

## initial value 26.849765

## iter 10 value 10.989322

## iter 20 value 10.955714

## iter 30 value 10.951777

## iter 40 value 10.951562

## final value 10.951541

## converged

## # weights: 41

## initial value 32.025911

## iter 10 value 11.050677

## iter 20 value 10.979900

## iter 30 value 10.942657

## iter 40 value 10.939626

## iter 50 value 10.939493

## final value 10.939493

## converged

## # weights: 41

## initial value 42.111868

## iter 10 value 11.083042

## iter 20 value 10.953153

## iter 30 value 10.951628

## iter 40 value 10.951561

## iter 50 value 10.951540

## iter 50 value 10.951540

## iter 50 value 10.951540

## final value 10.951540

## converged

## # weights: 41

## initial value 51.452718

## iter 10 value 13.681090

## iter 20 value 10.979947

## iter 30 value 10.958008

## iter 40 value 10.954194

## iter 50 value 10.952654

## iter 60 value 10.951666

## iter 70 value 10.951540

## iter 70 value 10.951540

## iter 70 value 10.951540

## final value 10.951540

## converged

## # weights: 41

## initial value 38.123377

## iter 10 value 11.066611

## iter 20 value 10.957299

## iter 30 value 10.951926

## iter 40 value 10.951545

## final value 10.951540

## converged

## # weights: 41

## initial value 33.945638

## iter 10 value 11.005029

## iter 20 value 10.957049

## iter 30 value 10.951637

## iter 40 value 10.951559

## iter 50 value 10.951540

## final value 10.951540

## converged

## # weights: 41

## initial value 53.766483

## iter 10 value 12.197989

## iter 20 value 10.958533

## iter 30 value 10.943320

## iter 40 value 10.941039

## iter 50 value 10.939557

## iter 60 value 10.939496

## final value 10.939493

## converged

# Make predictions using the fitted model

nnetar\_fc <- forecast::forecast(nnetar\_model, h = length(test\_data))

# Print the forecasted values

print(nnetar\_fc$mean)

## Mar Apr May Jun Jul Aug Sep Oct

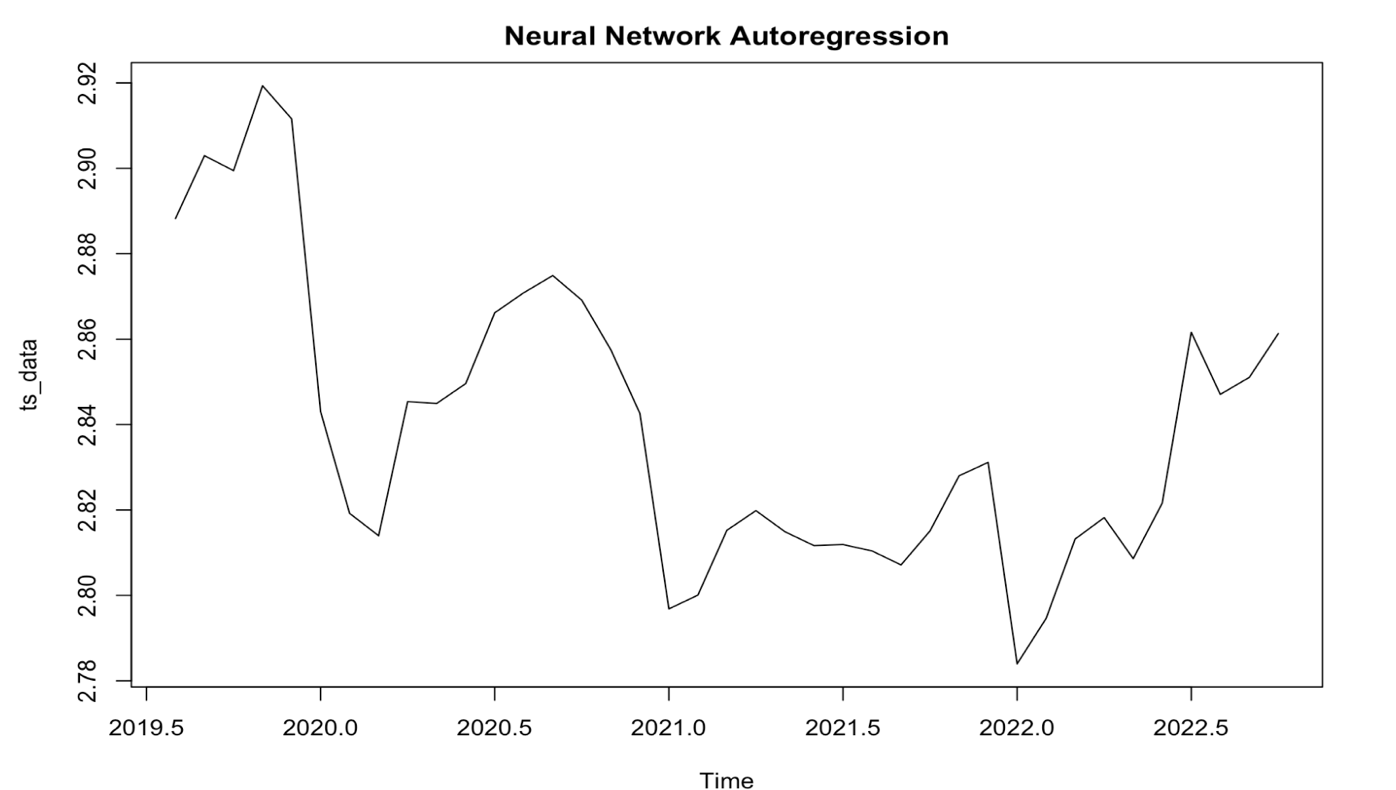
## 2022 2.804899 2.811903 2.816620 2.819850 2.822088 2.823652 2.824751 2.825525

# Plot actual and predicted values for comparison

plot(ts\_data, type = "l", main = "Neural Network Autoregression")

#lines(nnetar\_fc$mean, col = "blue")

legend("topleft", legend = c("Actual", "Predicted"), col = c("black", "blue"), lty = 1)



# Calculate accuracy measures

forecast::accuracy(nnetar\_fc$mean, test\_data)

## ME RMSE MAE MPE MAPE ACF1

## Test set 0.01666848 0.02306722 0.0186722 0.5839484 0.6552904 0.3798398

## Theil's U

## Test set 1.367118

# Calculate forecast errors

nnetar\_errors <- test\_data - nnetar\_fc$mean

mean(nnetar\_errors)

## [1] 0.01666848

sd(nnetar\_errors)

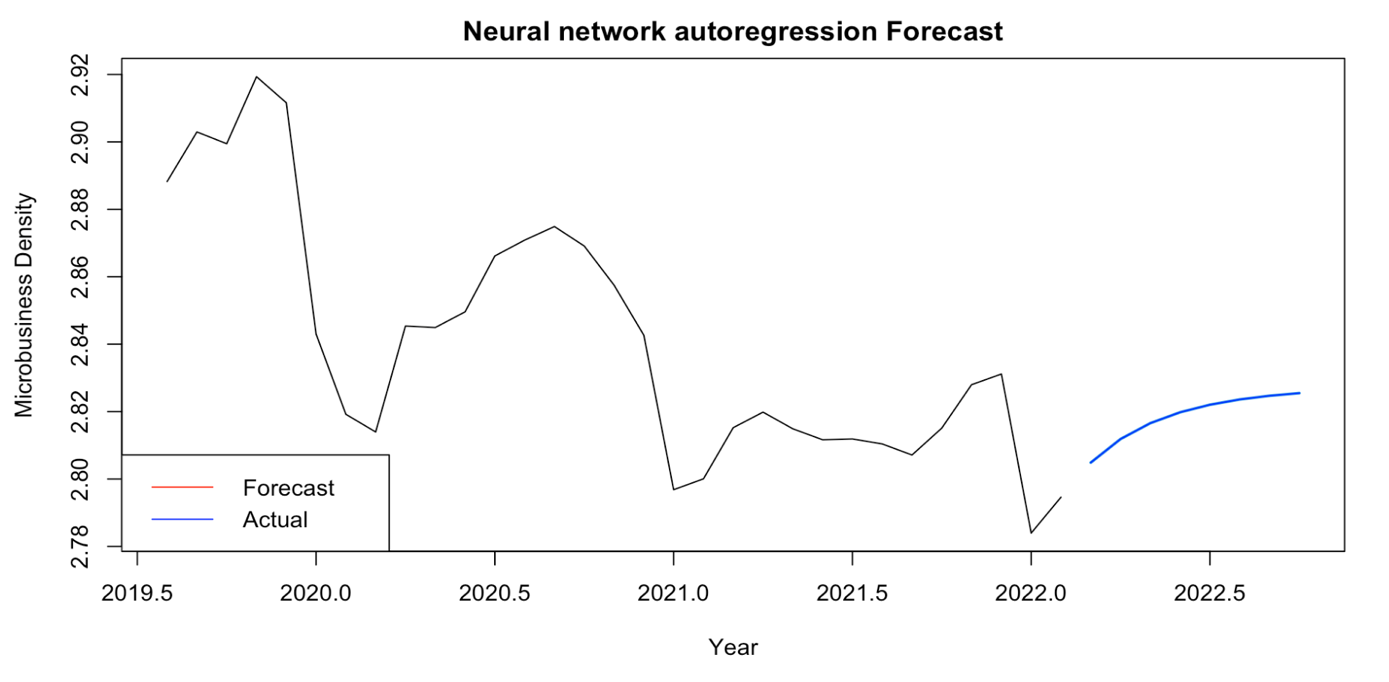
## [1] 0.01704644

plot(nnetar\_fc, main = "Neural network autoregression Forecast", xlab = "Year", ylab = "Microbusiness Density")

lines(nnetar\_fc$mean, col = "red")

#lines(ts\_data, col = "blue")

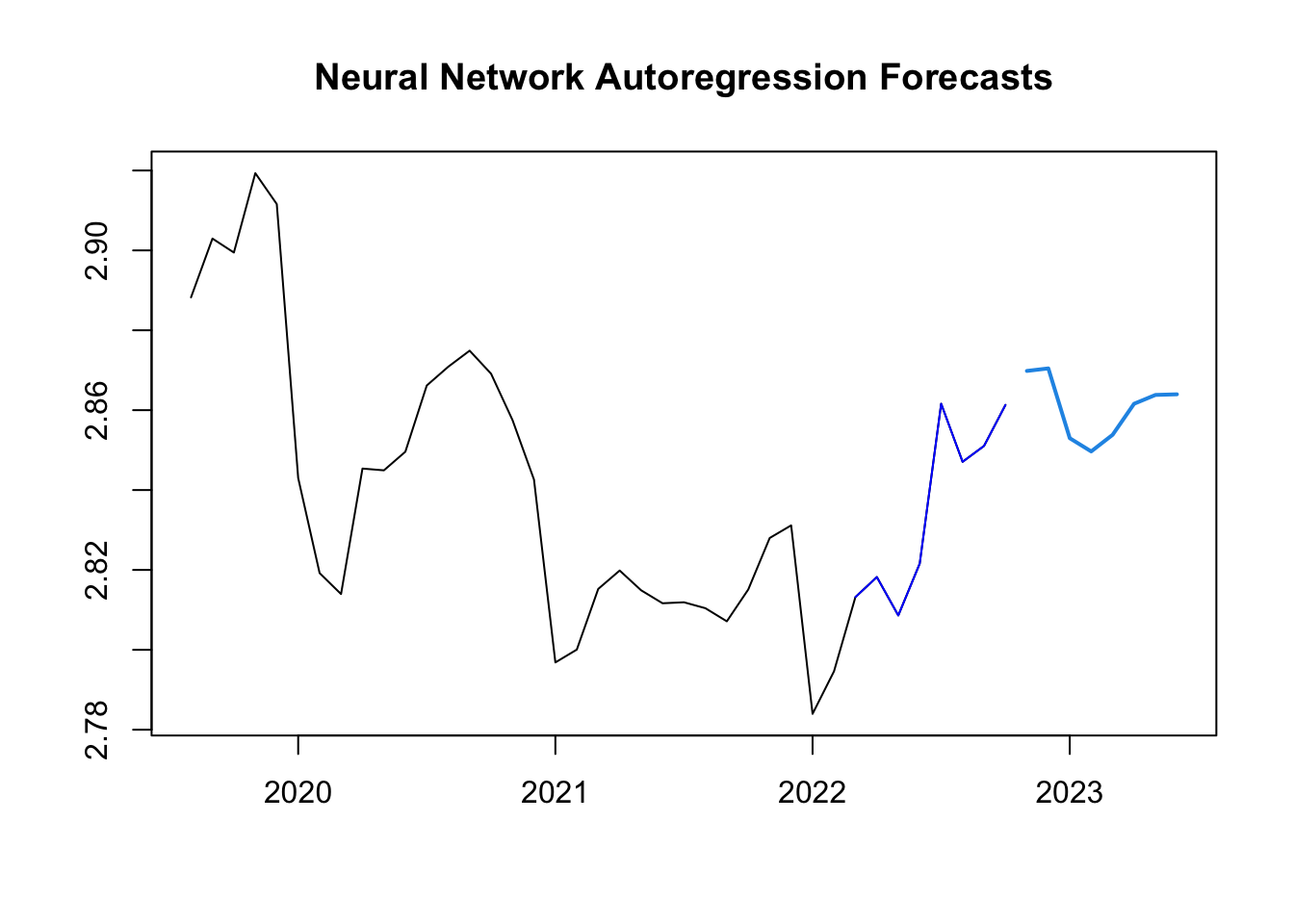
legend("bottomleft", legend = c("Forecast", "Actual"), col = c("red", "blue"), lty = 1)



# Plot forecasts and actual values

plot(forecast\_data, main = "Neural Network Autoregression Forecasts")

lines(test\_data, col = "blue")



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