Final Project 131

Yuanning Li

2022-11-08

Introduction

The aim of the project is to analyze the population trend among the world from 1970s to 2020s, and research about which model can best fit on the population data and predict and population. I use data from *Kaggle* and implement several techniques to answer the questions about population, growth rate, country areas, etc. I am also interested in fitting models on current data to predict the trend.

Loading Packages and Setting Up The Environment

This project uses data from *Kaggle* which records the population, land area, population density, etc. of 234 countries.

```
library(tidyverse)   # using tidyverse and tidymodels for this project mostly
library(tidymodels)
library(ggplot2)   # for most of our visualizations
library(rpart.plot)  # for visualizing trees
library(randomForest)   # for building our randomForest
```

```
# import data
df <- read.csv('data/world_population.csv')
head(df)
```

```
##   Rank CCA3 Country.Territory          Capital Continent X2022.Population
## 1   36  AFG       Afghanistan            Kabul      Asia         41128771
## 2  138  ALB           Albania           Tirana    Europe          2842321
## 3   34  DZA           Algeria          Algiers    Africa         44903225
## 4  213  ASM    American Samoa        Pago Pago   Oceania            44273
## 5  203  AND           Andorra Andorra la Vella    Europe            79824
## 6   42  AGO            Angola           Luanda    Africa         35588987
##   X2020.Population X2015.Population X2010.Population X2000.Population
## 1         38972230         33753499         28189672         19542982
## 2          2866849          2882481          2913399          3182021
## 3         43451666         39543154         35856344         30774621
## 4            46189            51368            54849            58230
## 5            77700            71746            71519            66097
## 6         33428485         28127721         23364185         16394062
##   X1990.Population X1980.Population X1970.Population Area..km..
## 1         10694796         12486631         10752971     652230
## 2          3295066          2941651          2324731      28748
## 3         25518074         18739378         13795915    2381741
## 4            47818            32886            27075        199
## 5            53569            35611            19860        468
## 6         11828638          8330047          6029700    1246700
```

```
##   Density..per.km.. Growth.Rate World.Population.Percentage
## 1           63.0587      1.0257                        0.52
## 2           98.8702      0.9957                        0.04
## 3           18.8531      1.0164                        0.56
## 4          222.4774      0.9831                        0.00
## 5          170.5641      1.0100                        0.00
## 6           28.5466      1.0315                        0.45
```

DATA DESCRIPTION

This database from *Kaggle* contains 17 variables and 234 columns. While the codebook is provided in my text file, the variables listed here are useful for understanding this report.

`Rank`="Rank of Popluation"

`CCA3`="3 Digit Country/Territories Code"

`Country`="Name of the Country/ Territories"

`Capital`="Name of the Capital"

`Continent`="Name of the Continent"

`2022_Population`="Population of the Country/Territories in the year 2022"

`2020_Population`="Population of the Country/Territories in the year 2020"

`2015_Population`="Population of the Country/Territories in the year 2015"

`2010_Population`="Population of the Country/Territories in the year 2010"

`2000.Population`="Population of the Country/Territories in the year 2000"

`1990_Population`="Population of the Country/Territories in the year 1990"

`1980_Population`="Population of the Country/Territories in the year 1980"

`1970_Population`="Population of the Country/Territories in the year 1970"

`Area`="Area size of the Country/Territories in square kilometer"

`Density`="Population Density per square kilometer"

`Growth_Rate`="Population Growth Rate by Country/Territories"

`World_Population_Percentage`="The Population percentage by each Country/Territories"

Note: a full copy of the codebook is available in text files

What is the current state of the world population?

In June 2019, the world population estimate surveyed by the U.S. Census Bureau showed that the current global population is 757,713,040 people, which is much higher than the world population of 7.2 billion in 2015. It seems that the earth is very crowded and the population is still growing very fast.

Why is analyzing and modeling the population so important?

For a growing population, since the resources on our planet are limited, the more the population, the greater the demand for various resources. When the world's population reaches a certain number, resources will be exhausted, and the loss of resources may lead to the destruction of the earth! So I hope these analysis and models can help us understand the current population situation and take corresponding measures to maintain the earth.

Project pathway

Knowing the background and the importance of the topic, I'd like to discuss how to analyze and build models. First I do some initial data manipulation and cleaning on the original data. Then I explore the data and see if there's any interesting findings about population, density, area in time series and continent perspective.

At the end, I use existing data from previous years to make the prediction. I'll split and resample the data, build the recipe and workflow, and train the model. Because this is a regression problem, I choose Ridge regression, Lasso regression, regression tree, and random forest these four models. I'd like to find the best parameters for our model using cross validation, and find which of the four models perform the best on our test dataset.

Clean Data

Our original data is quite clean, so there's not much data cleaning to do. I simply renamed the feature names to make it more clean.

```
# renamed all of the columns to make column names more clean and neat
colnames(df) <- c('Rank', 'CCA3', 'Country','Capital','Continent','Population_2022',
                  'Population_2020','Population_2015','Population_2010',
                  'Population_2000','Population_1990','Population_1980',
                  'Population_1970','Area', 'Density', 'Growth_Rate',
                  'World_Population_Percentage')
```

Exploratory Data Analysis (EDA)

The entire exploratory data analysis will be based on the entire data set with 234 observations. Each observation represents the data of a country.

Key Takeaways

– What are the countries with largest and smallest population in the world? Is China going to remain the Top 1 population country in the world in the future?

– Are the countries with small density / population has smaller growth rate than those countries with large density / population?

– In the continent perspective, which continent has largest population and density? What is the trend?

– What are the largest and smallest countries in size? The EDA part will answer the above questions.

First extract the top five countries by population from 1970 to 2022.

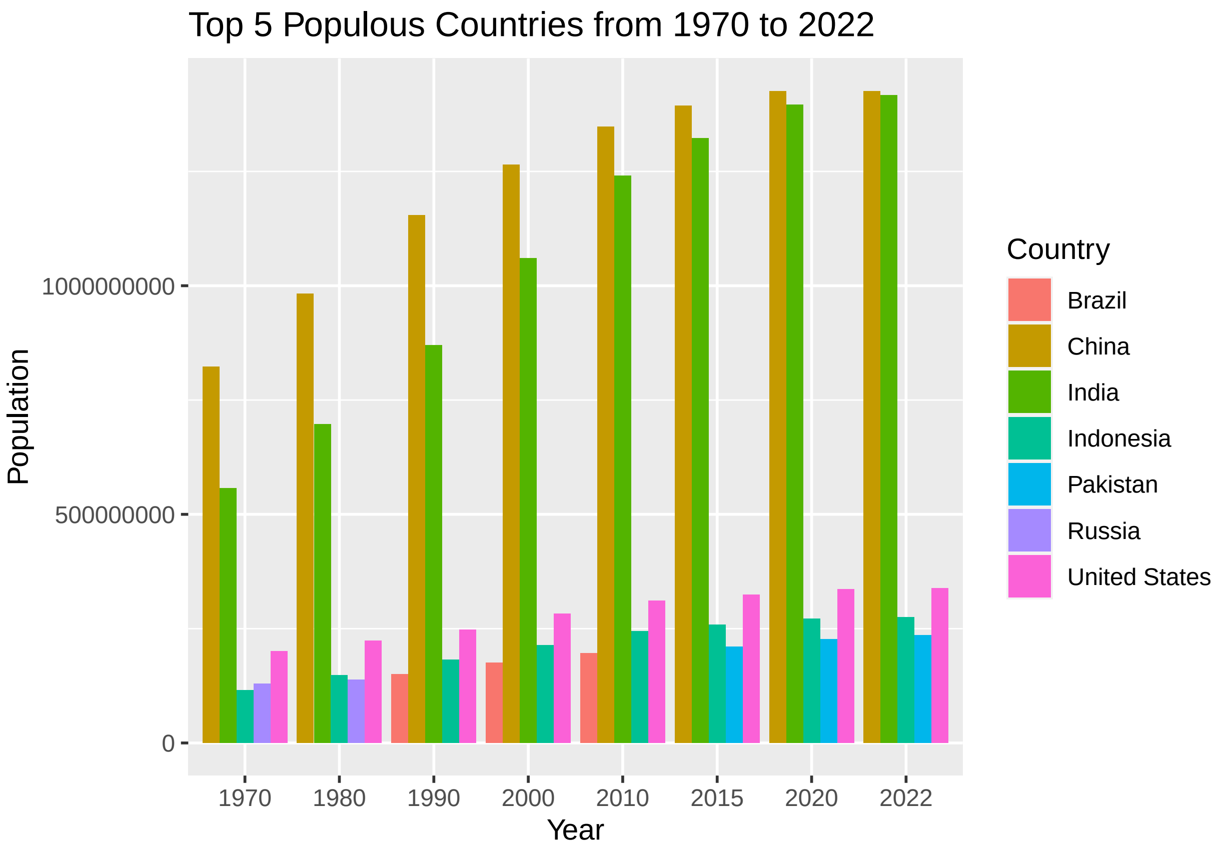
```
# for EDA, no need for more dropping or cleaning, we will clean later for model
# add column for growth rate??
# EDA
# 1. Top 5 populous countries in 1970, 1980, 1990, 2000, 2010, 2015, 2020 and 2022.
```

```r
df1 <- df
colnames(df1) <- c('Rank', 'CCA3', 'Country','Capital','Continent','2022',
                  '2020','2015','2010',
                  '2000','1990','1980',
                  '1970','Area', 'Density', 'Growth_Rate',
                  'World_Population_Percentage')

df1 <- df1 %>% select(c('Country','2022','2020',
                       '2015','2010',
                      '2000','1990',
                      '1980','1970')) %>% pivot_longer(cols=c('2022','2020',
                       '2015','2010',
                      '2000','1990',
                      '1980','1970'),
                      names_to = 'Year', values_to = 'Population') %>%
  arrange(desc(Population)) %>%
  group_by(Year) %>%
  slice(1:5)
head(df1, 10)
```

```
## # A tibble: 10 x 3
## # Groups:   Year [2]
##    Country       Year  Population
##    <chr>         <chr>      <int>
##  1 China         1970   822534450
##  2 India         1970   557501301
##  3 United States 1970   200328340
##  4 Russia        1970   130093010
##  5 Indonesia     1970   115228394
##  6 China         1980   982372466
##  7 India         1980   696828385
##  8 United States 1980   223140018
##  9 Indonesia     1980   148177096
## 10 Russia        1980   138257420
```

Then, use the above content to make a bar plot. it show the population of the top five populous countries from 1970 to 2022.

```r
ggplot(df1, aes(fill=Country, y=Population, x=Year)) +
  geom_bar(position='dodge', stat='identity') +
  ggtitle ('Top 5 Populous Countries from 1970 to 2022')
```



From the figure, We can conclude that Russia's population growth is relatively small, so that it fell out of the top five after 1990. Brazil had a large population in 1990-2010, and it was not in the top five at other times. China has always maintained the first place, but India's population has grown relatively faster and has been catching up with China. In the very near future, India's population will soon catch up with China's.

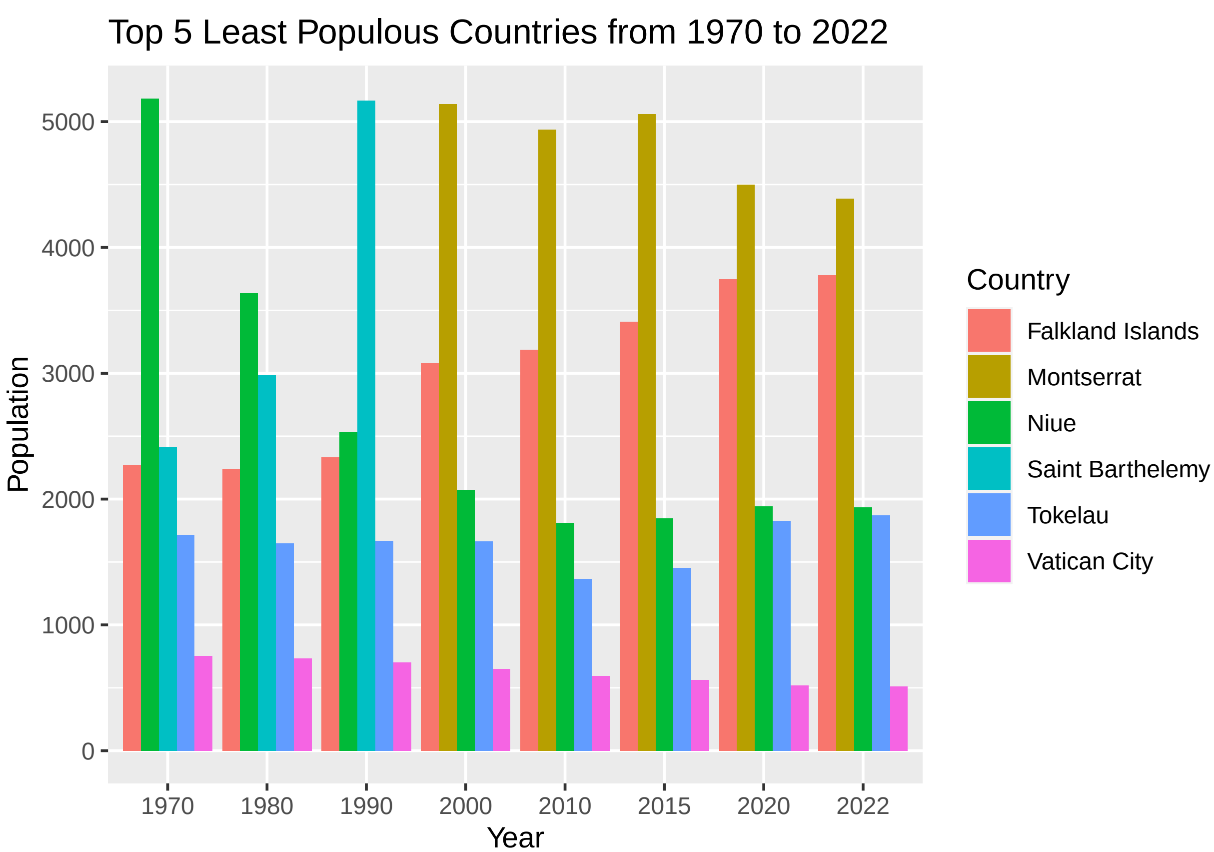
Then, we extract the least five countries by population from 1970 to 2022.

```
# 2. Top 5 populous countries in 1970, 1980, 1990, 2000, 2010, 2015, 2020 and 2022.
df2 <- df
colnames(df2) <- c('Rank', 'CCA3', 'Country','Capital','Continent','2022',
                  '2020','2015','2010',
                  '2000','1990','1980',
                  '1970','Area', 'Density', 'Growth_Rate',
                  'World_Population_Percentage')

df2 <- df2 %>% select(c('Country','2022','2020',
                         '2015','2010',
                        '2000','1990',
                        '1980','1970')) %>% pivot_longer(cols=c('2022','2020',
                         '2015','2010',
                        '2000','1990',
                        '1980','1970'),
                        names_to = 'Year', values_to = 'Population') %>%
  arrange(Population) %>%
  group_by(Year) %>%
  slice(1:5)
head(df2, 10)
```

```
## # A tibble: 10 x 3
## # Groups:   Year [2]
##    Country          Year  Population
##    <chr>            <chr>      <int>
##  1 Vatican City     1970         752
##  2 Tokelau          1970        1714
##  3 Falkland Islands 1970        2274
##  4 Saint Barthelemy 1970        2417
##  5 Niue             1970        5185
##  6 Vatican City     1980         733
##  7 Tokelau          1980        1647
##  8 Falkland Islands 1980        2240
##  9 Saint Barthelemy 1980        2983
## 10 Niue             1980        3637
```

The method I use to create a bar plot is similar to the previous question, it shows the population of the five least populous countries from 1970 to 2022.

```r
ggplot(df2, aes(fill=Country, y=Population, x=Year)) +
  geom_bar(position='dodge', stat='identity') +
  ggtitle ('Top 5 Least Populous Countries from 1970 to 2022')
```



We can conclude that the population of Niue began to decrease from 1970 until 2010, and then increased a little. Montserrat is gaining momentum, starting in 2000 as the most populous of the five least populated countries, but Falkland Island has been growing as well.

Next, we look at the population growth rate for top 5 countries.

As described above, although China's population is currently the largest in the world, India is gaining momentum. Is it possible to surpass China and become the next country with the largest population in the world?

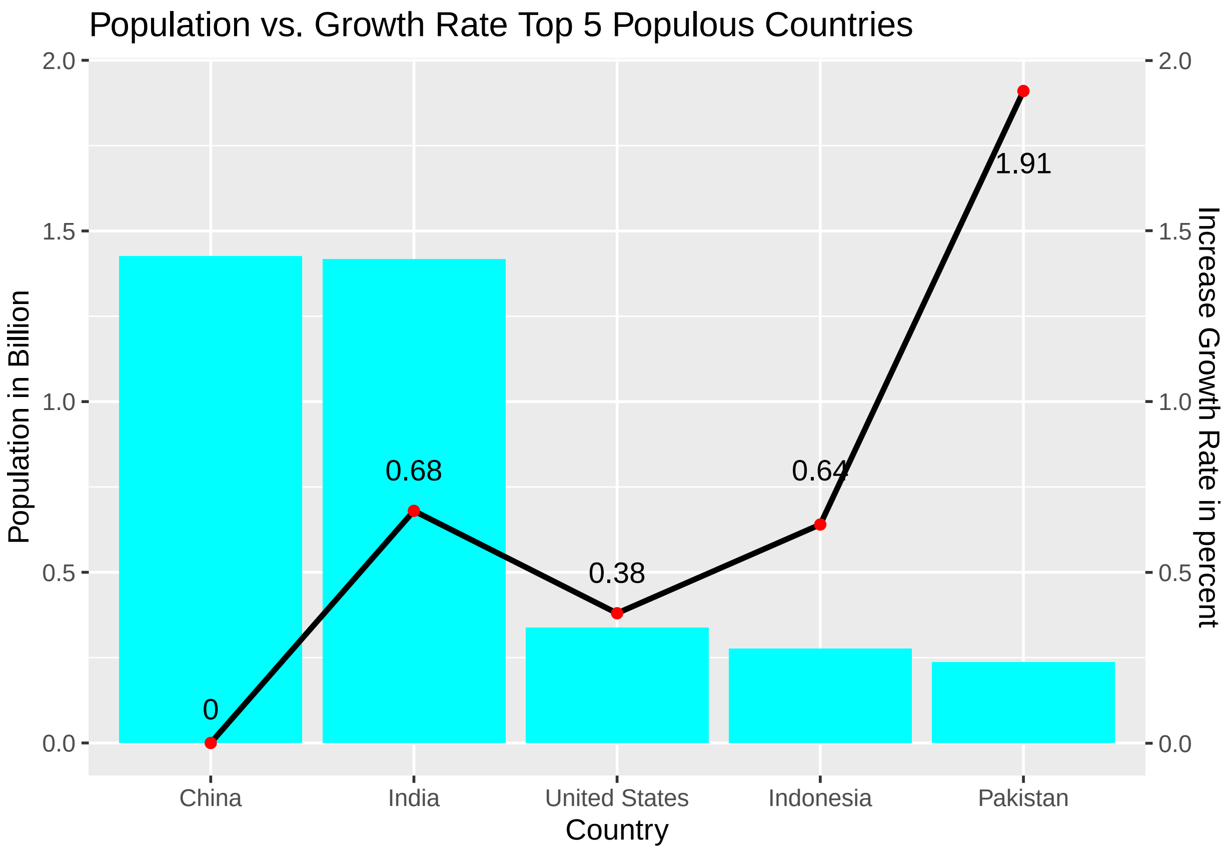
```
# 3. growth rate of the 5 most populous countries
df3 <- df1 %>% filter(Year == 2022)
df_tmp <- df %>% select(c('Country','Growth_Rate'))
df3 <- merge(df3, df_tmp, by='Country', all.x=TRUE) %>%
  select(c('Country','Population','Growth_Rate')) %>% arrange(desc(Population))
df3
```

```
##         Country Population Growth_Rate
## 1         China 1425887337      1.0000
## 2         India 1417173173      1.0068
## 3 United States  338289857      1.0038
## 4     Indonesia  275501339      1.0064
## 5      Pakistan  235824862      1.0191
```

From the data, the China has the least population growth rate and Pakistan has the highest population growth rate. Based on India's growth rate of 1.0068 is higher than China's growth rate of 1.0000, and the population until 2022, we expect India's population will have more than China's population.

Using the data `df3` obtained above, make a visualization to observe the population and growth trends of the top five countries in the world.

```
ggplot(df3, aes(x=reorder(Country, +desc(Population))))  +
  geom_bar(aes(y=Population*0.000000001, group=1),stat="identity", fill="cyan") +
  geom_line(aes(y=(Growth_Rate-1)*100, group=1),stat="identity",color="black",size=1)+
  scale_y_continuous(name = 'Population in Billion', sec.axis=sec_axis(trans = ~.*1, name="Increase Grow
  labs(title= "Population vs. Growth Rate Top 5 Populous Countries", x = 'Country') +
  geom_point(aes(y=(Growth_Rate-1)*100, group=1), col='red') +
  geom_text(x=c(1,2,3,4,5), y = c(0.1, 0.8, 0.5, 0.8, 1.7), label=c(0, 0.68, 0.38, 0.64, 1.91))
```



To better visualize the growth rate, we changed population growth rate to increase of growth rate in percentage,for example, India's growth rate is 1.0068, so the increase of growth rate is 0.68%.

From the plot, Pakistan has the highest increase of growth rate 1.91%. China has a increase of growth rate 0% in recent years, while India has a increase of growth rate 0.68%. If maintains, India's population will surpass that of China.

Then, we look at the comparison of population density. First look at the five countries with the lowest population density.

```
# 4. The 5 countries with the least population density
df4 <- df %>% arrange(Density) %>% slice(1:5) %>%
  select(c("Country","Population_2022", "Density"))
df4
```

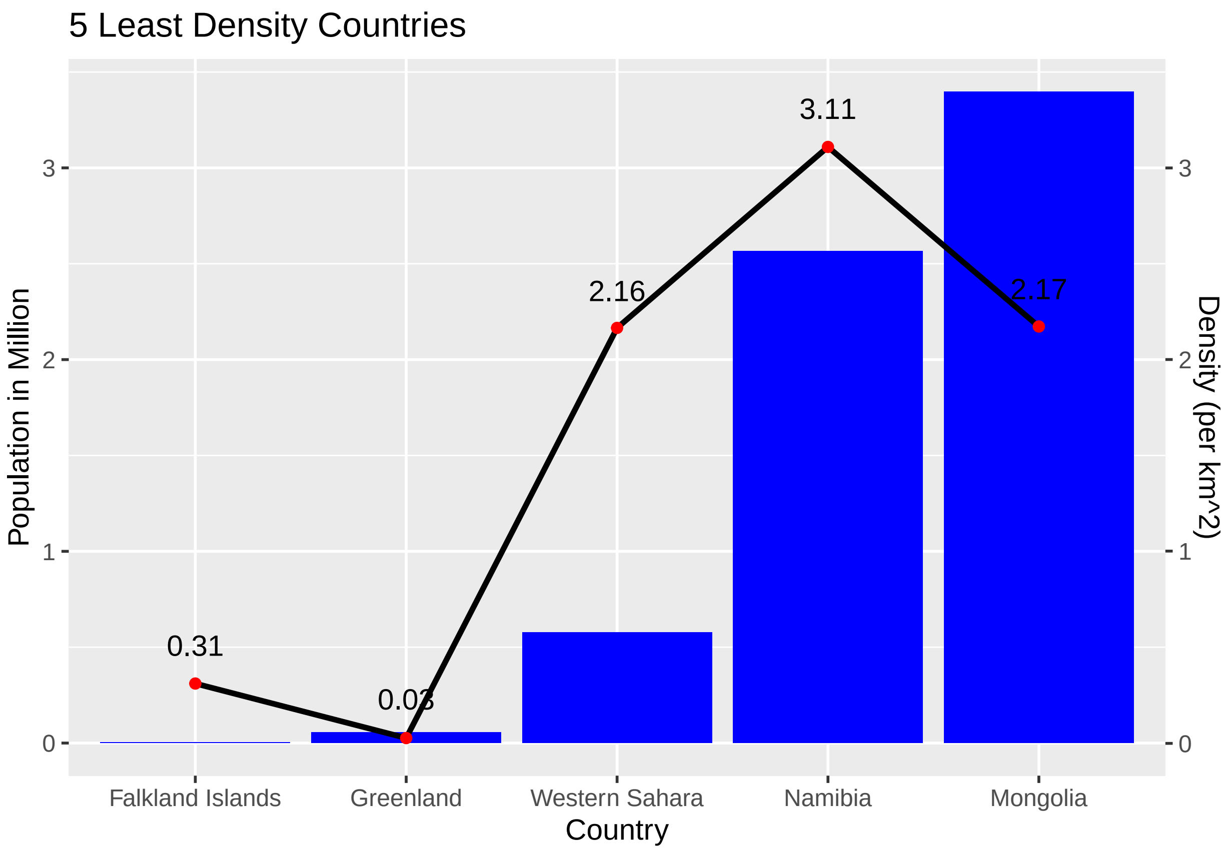
```
##             Country Population_2022 Density
## 1         Greenland           56466  0.0261
## 2 Falkland Islands            3780  0.3105
## 3   Western Sahara          575986  2.1654
## 4          Mongolia         3398366  2.1727
## 5           Namibia         2567012  3.1092
```

From the data, we know Greenland, Falkland Islands, Western Sahara, Mongolia and Namibia has the least population density.

Using the data `df4` obtained above, make a visualization to observe the population and density of the five countries with the least population in the world.

```
ggplot(df4, aes(x=reorder(Country, +Population_2022)))  +
  geom_bar(aes(y=Population_2022*0.000001, group=1),stat="identity", fill="blue") +
```

```
  geom_line(aes(y=Density, group=1),stat="identity",color="black",size=1)+
  scale_y_continuous(name = 'Population in Million', sec.axis=sec_axis(trans = ~.*1, name="Density (per
  labs(title= "5 Least Density Countries", x= 'Country') +
  geom_point(aes(y=Density, group=1), col='red') +
  geom_text(x=c(1,2,3,4,5), y = c(0.51,0.23,2.36,3.31,2.37), label=c(0.31,0.03,2.16,3.11,2.17))
```



From the figure, the blue bars represents the population and the line represents the population density. Of the five countries, Namibia has the highest population density of $3.11(people/km^2)$ and Greenland has the lowest of $0.03(people/km^2)$.

Next, we look at the population growth rate for 5 countries with least density.

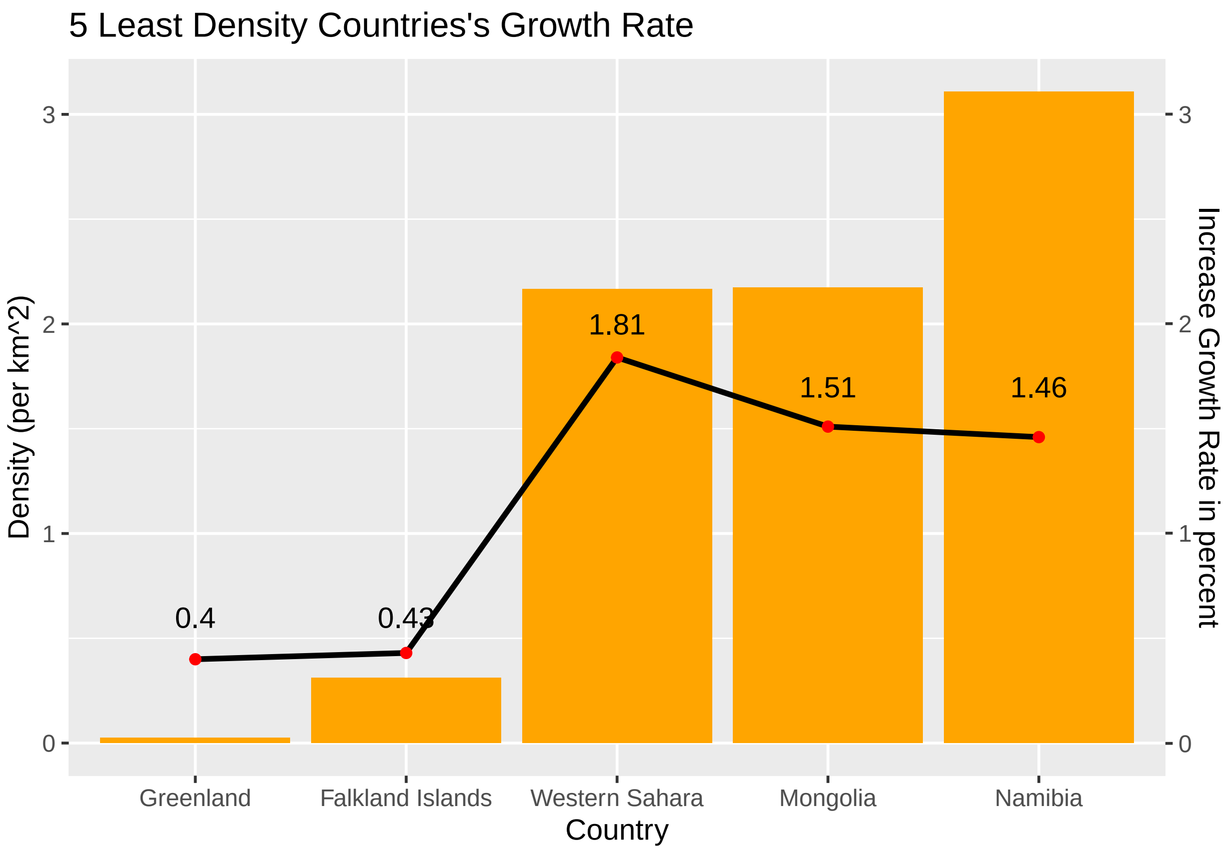
```
# Growth rate of the 5 least densely populated countries
df5 <- df %>% arrange(Density) %>% slice(1:5) %>%
  select(c("Country", "Density", "Growth_Rate"))
df5
```

```
##             Country Density Growth_Rate
## 1         Greenland  0.0261      1.0040
## 2 Falkland Islands  0.3105      1.0043
## 3   Western Sahara  2.1654      1.0184
## 4          Mongolia  2.1727      1.0151
## 5           Namibia  3.1092      1.0146
```

From the data, the Greenland has the least population growth rate and Western Sahara has the highest population growth rate.

```
ggplot(df5, aes(x=reorder(Country, +Density)))  +
  geom_bar(aes(y=Density, group=1),stat="identity", fill="orange") +
```

```
  geom_line(aes(y=(Growth_Rate-1)*100, group=1),stat="identity",color="black",size=1)+
  scale_y_continuous(name = 'Density (per km^2)', sec.axis=sec_axis(trans = ~.*1, name="Increase Growth
  labs(title= "5 Least Density Countries's Growth Rate", x = "Country") +
  geom_point(aes(y=(Growth_Rate-1)*100, group=1), col='red') +
  geom_text(x=c(1,2,3,4,5), y = c(0.6,0.6,2,1.7,1.7), label=c(0.4,0.43,1.81,1.51,1.46))
```



From the plot, the Western Sahara has the highest increase population growth rate of 1.81%, and the Greenland has the lowest increase population growth rate of 0.4%. comparing to the countries with large population and high density, the increase of growth rate of these low density countries are the similar.

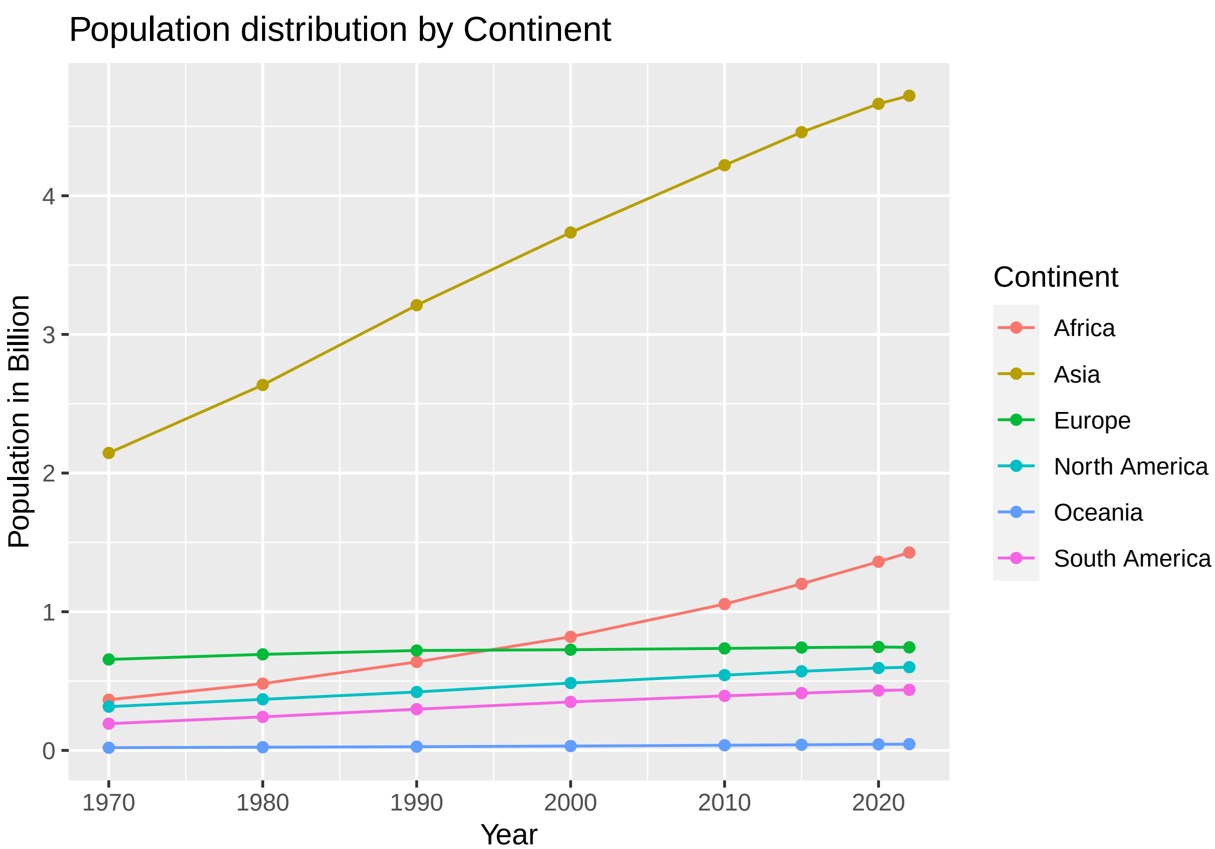
Then, Let's focus on the proportion of the population of different continents in the world from 1970 to 2022.

```
# 6. Population distribution by continent
df6 <- df

df6 <- df6 %>% select(c('Continent','Population_2022','Population_2020',
                        'Population_2015','Population_2010',
                       'Population_2000','Population_1990',
                       'Population_1980','Population_1970')) %>% group_by(Continent) %>%
  summarise(sum_2022 = sum(Population_2022),
            sum_2020 = sum(Population_2020),
            sum_2015 = sum(Population_2015),
            sum_2010 = sum(Population_2010),
            sum_2000 = sum(Population_2000),
            sum_1990 = sum(Population_1990),
            sum_1980 = sum(Population_1980),
            sum_1970 = sum(Population_1970))
```

```r
colnames(df6) <- c('Continent', '2022',
                   '2020','2015','2010',
                   '2000','1990','1980',
                   '1970')

df6 <- df6 %>% pivot_longer(cols=c('2022','2020',
                        '2015','2010',
                       '2000','1990',
                       '1980','1970'),
                       names_to = 'Year', values_to = 'Population') %>%
  arrange(Continent, Year)
df6[['Year']] <- as.numeric(df6[['Year']])
```

This time we use a line chart to do this visualization.

```r
ggplot(data = df6, aes(x=Year)) +
  geom_line(aes(y=Population / 1000000000, colour=Continent)) +
  geom_point(aes(y=Population / 1000000000, colour=Continent)) +
  labs(title= "Population distribution by Continent", y = "Population in Billion")
```



In the plot,Asia's population grows fast continuously form 1970s to 2022. And its population is much more than other continents.The other continent that has obvious population growth trend is Africa. Compare to Asia and Africa, other continents do not have a big growth throughout the years. The population of Oceania is the smallest of all continents, and the growth rate is also very small.

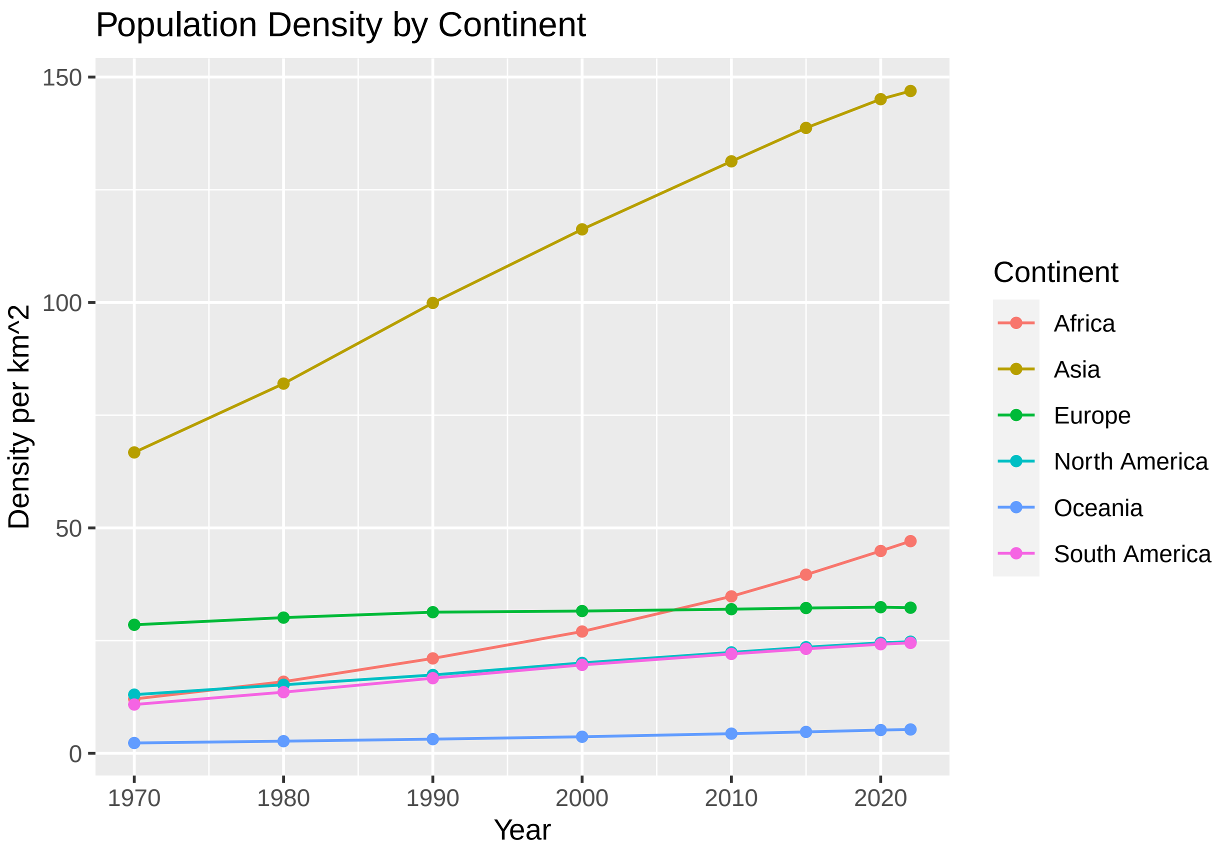
Then, Let's look at the population density of each of these continents.

```r
# 7. Population density of each continent
```

```r
df7 <- df

df7 <- df7 %>% select(c('Continent','Population_2022','Population_2020',
                       'Population_2015','Population_2010',
                      'Population_2000','Population_1990',
                      'Population_1980','Population_1970', 'Area')) %>% group_by(Continent) %>%
  summarise(density_2022 = sum(Population_2022) / sum(Area),
            density_2020 = sum(Population_2020) / sum(Area),
            density_2015 = sum(Population_2015) / sum(Area),
            density_2010 = sum(Population_2010) / sum(Area),
            density_2000 = sum(Population_2000) / sum(Area),
            density_1990 = sum(Population_1990) / sum(Area),
            density_1980 = sum(Population_1980) / sum(Area),
            density_1970 = sum(Population_1970) / sum(Area))

colnames(df7) <- c('Continent', '2022',
                  '2020','2015','2010',
                  '2000','1990','1980',
                  '1970')

df7 <- df7 %>% pivot_longer(cols=c('2022','2020',
                       '2015','2010',
                      '2000','1990',
                      '1980','1970'),
                      names_to = 'Year', values_to = 'Density') %>%
  arrange(Continent, Year)
df7[['Year']] <- as.numeric(df7[['Year']])
```

Through the above calculation and classification, visualization can now be made.

```r
ggplot(data = df7, aes(x=Year)) +
  geom_line(aes(y=Density, colour=Continent)) +
  geom_point(aes(y=Density, colour=Continent)) +
  labs(title= "Population Density by Continent", y = "Density per km^2")
```



From the visualization, Asia is still far ahead of other continents. From a population density of about 65 people per square kilometer in 1970 to about 150 people per square kilometer in 2022. Oceania still has the lowest population density. In 1970, the population density was about 1 to 2 people per square kilometer. By 2022, about 3 to 4 people per square kilometer.

By comparing population trend and population density trend from 1970 to 2020, we find the two trend shares the same pattern.

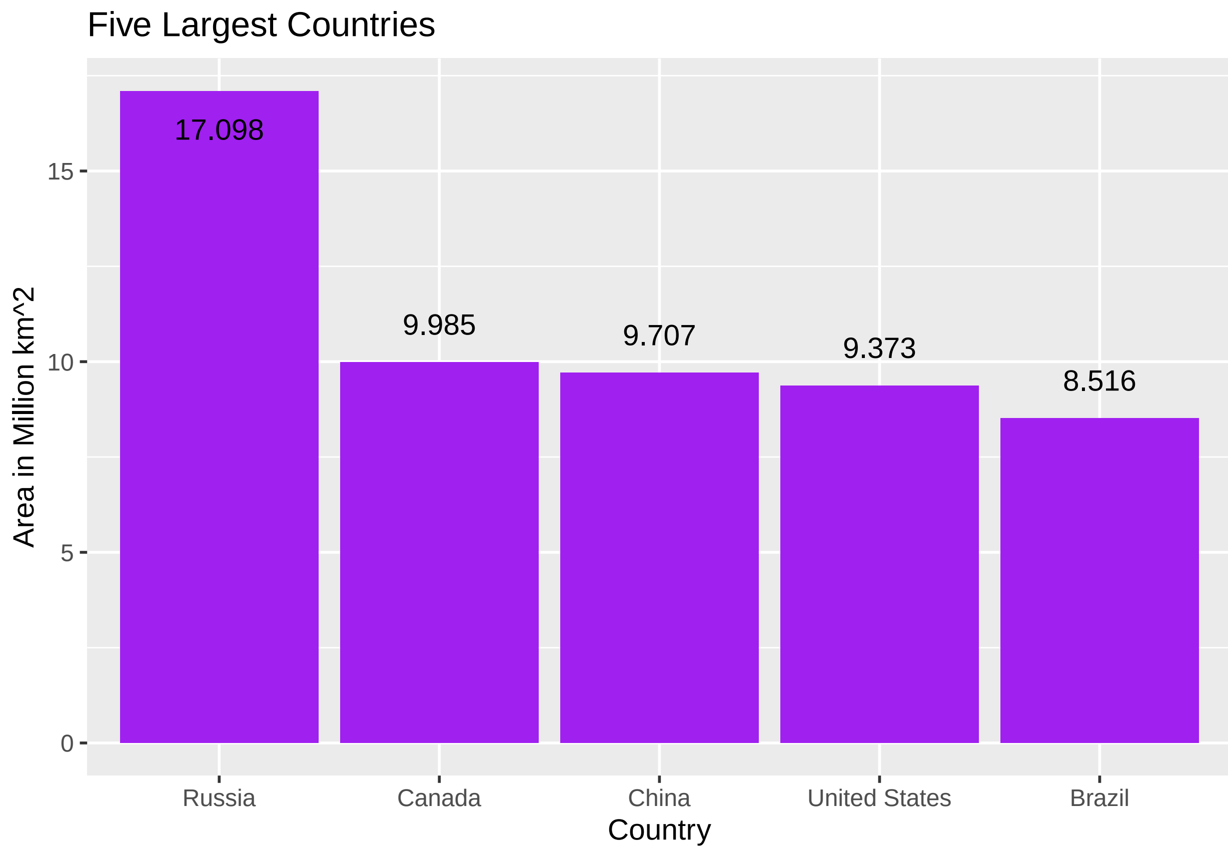
Finally, we are interest in the land area of countries. The following distribution is the largest 5 countries and the smallest 5 countries.

```
# Five largest countries
df8 <- df %>% arrange(desc(Area)) %>% slice(1:5) %>% select(c("Country", "Area"))
df8
```

```
##         Country     Area
## 1        Russia 17098242
## 2        Canada  9984670
## 3         China  9706961
## 4 United States  9372610
## 5        Brazil  8515767
```

The Largest country is Russia, the Area is 17098242 km^2.

```
ggplot(df8, aes(x=reorder(Country, +desc(Area))))  +
  geom_bar(aes(y=Area/1000000, group=1),stat="identity", fill="purple") +
  geom_text(x=c(1,2,3,4,5), y = c(16.098,10.985,10.707,10.373,9.516), label=c(17.098,9.985,9.707,9.373,8
  labs(title= "Five Largest Countries", y = "Area in Million km^2",
       x = 'Country')
```



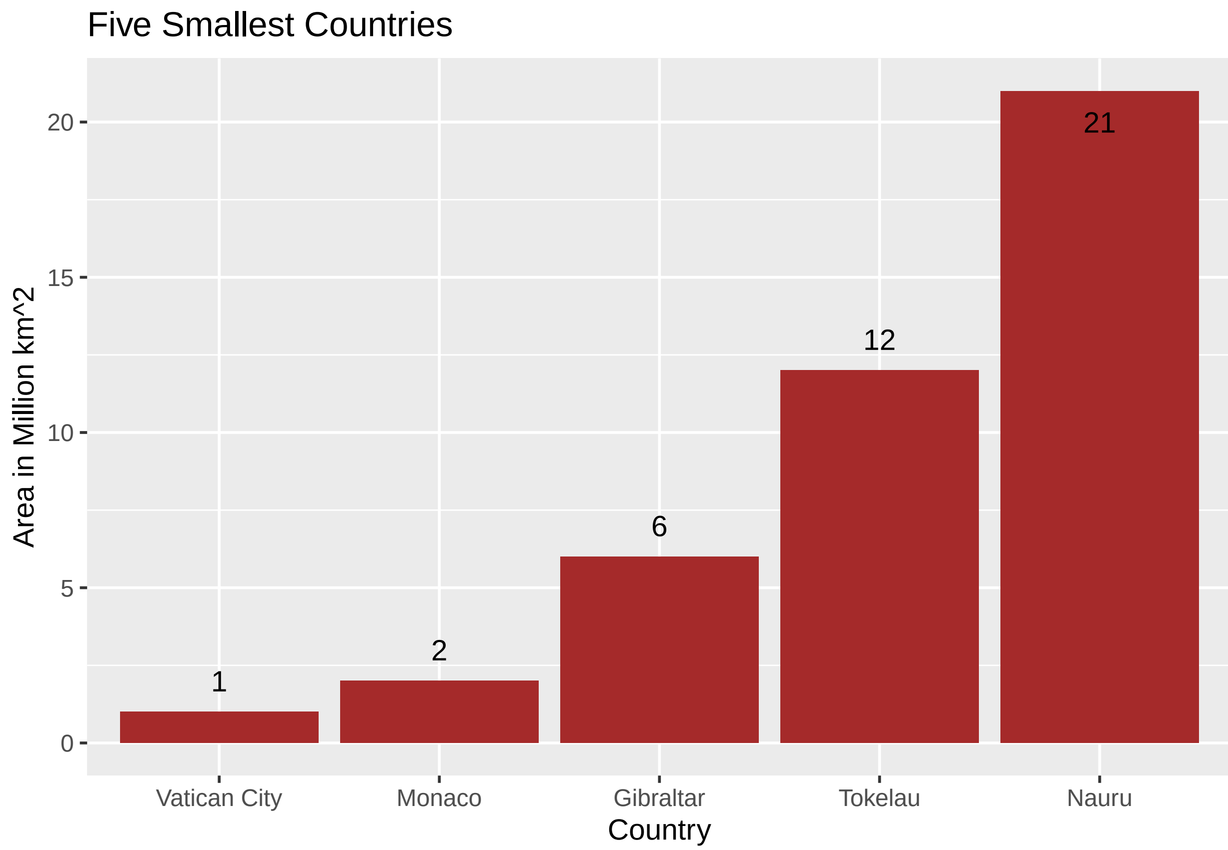
In the visualization, Russia is shown to have the largest area, roughly twice the size of Brazil, which ranks fifth. The area gap between Canada, China and the United States, which ranks second to fourth, is relatively small.

```
# Five smallest countries
df9 <- df %>% arrange(Area) %>% slice(1:5) %>% select(c("Country", "Area"))
df9
```

```
##        Country Area
## 1 Vatican City    1
## 2       Monaco    2
## 3    Gibraltar    6
## 4      Tokelau   12
## 5        Nauru   21
```

The smallest country is Vatican City, and it's area is $1km^2$.

```
ggplot(df9, aes(x=reorder(Country, +Area)))  +
  geom_bar(aes(y=Area, group=1),stat="identity", fill="brown") +
  geom_text(x=c(1,2,3,4,5), y = c(2,3,7,13,20), label=c(1,2,6,12,21)) +
  labs(title= "Five Smallest Countries", y = "Area in Million km^2",
       x = 'Country')
```



In the visualization, the Vatican is shown to have the smallest area at $1km^2$, about the size of a square. The largest of them, Nauru, is only $21km^2$.

Data split & cross validation

For model training, we only keep the relevant features, including all the previous year's population, area, and continent. We delete features such as density and growth rate because they can be used to calculate population of 2022 directly (correlation is almost 1).

The data was split in 70% training, 30% testing split. Stratified sampling was used as the `Continent`.

```
# keep only the relevant variables
data <- df %>% select(c(Continent, Area, Population_1970,
                        Population_1980, Population_1990, Population_2000,
                        Population_2010, Population_2015, Population_2020,
                        Population_2022))
head(data)
```

```
##   Continent    Area Population_1970 Population_1980 Population_1990
## 1      Asia  652230        10752971        12486631        10694796
## 2    Europe   28748         2324731         2941651         3295066
## 3    Africa 2381741        13795915        18739378        25518074
## 4   Oceania     199           27075           32886           47818
## 5    Europe     468           19860           35611           53569
## 6    Africa 1246700         6029700         8330047        11828638
##   Population_2000 Population_2010 Population_2015 Population_2020
## 1        19542982        28189672        33753499        38972230
## 2         3182021         2913399         2882481         2866849
```

```
## 3        30774621        35856344        39543154        43451666
## 4           58230           54849           51368           46189
## 5           66097           71519           71746           77700
## 6        16394062        23364185        28127721        33428485
##   Population_2022
## 1        41128771
## 2         2842321
## 3        44903225
## 4           44273
## 5           79824
## 6        35588987
```

```
# initial train test split
pop_split <- initial_split(data, strata = Continent, prop = 0.7)
pop_split
```

```
## <Analysis/Assess/Total>
## <162/72/234>
```

The training data set has 161 observations and the testing data set has 73 observations.

Then we use the cross validation resampling method to fold the training data into 10 folds with 5 repeats.

```
pop_train <- training(pop_split)
pop_test <- testing(pop_split)
# 5 fold cross validation
pop_folds <- vfold_cv(pop_train, v = 5)
```

Model Building

Steps we use to build and analysis the model:

1. Build the recipe and workflow for each of the model.
2. Use cross validation to tune the model parameters, and find the best parameter for the model.
3. Use the best parameters found in step2, fit the model on test set and calculate model performance.
4. Compare the performance for each model. Find the best model in the four models.

Building the Recipe

set up preprocess recipe for all models.

```
recipe <-
  recipe(formula = Population_2022 ~ ., data = pop_train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_normalize(all_predictors())
```

After many considerations, I decided to build the following four models.

1. Ridge Regression
2. Lasso Regression
3. Regression Tree

4. Random Forest

Ridge Regression

Loaded the required object that I saved in my script, set `mode` to `"regression"`,tuned `penality`, and used the `glmnet` engine. I stored this model and `recipe` in workflow.

```
set.seed(1234)
ridge_spec <- linear_reg(mixture = 0, penalty = tune()) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

ridge_workflow <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(ridge_spec)
```

Next, I set up the adjustment grid and updated the penalty parameters. I tune the penalty, in the range of -10 to 10 with the level 50.

```
ridge_penalty_grid <- grid_regular(penalty(range = c(-10, 10)), levels = 50)

ridge_tune_res <- tune_grid(
  ridge_workflow,
  resamples = pop_folds,
  grid = ridge_penalty_grid)
```

Lasso Regression

In a similar process, I set the model with tuning parameter penalty. Set the engine as glmnet and created a workflow.

The difference between Lasso Regression and Ridge Regression is Lasso regression 's mixture parameter equals 1 and Ridge regression's mixture parameter equals 0. This differs the two models from $L1$ to $L2$.

```
set.seed(1234)
lasso_spec <-
  linear_reg(penalty = tune(), mixture = 1) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

lasso_workflow <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(lasso_spec)
```

Next, set up the adjustment grid and updated the parameters. Tune the penalty, in the range of -10 to 10 with the level 50.

```
lasso_penalty_grid <- grid_regular(penalty(range = c(-10, 10)), levels = 50)

lasso_tune_res <- tune_grid(
  ridge_workflow,
  resamples = pop_folds,
  grid = lasso_penalty_grid)
```

Regression Tree

In this process, Set `mode` to `"regression"` and used the `rpart` engine. Stored this model and `recipe` in workflow.

```
set.seed(1234)
reg_tree_spec <- decision_tree() %>%
  set_engine("rpart") %>%
  set_mode("regression")

reg_tree_wf <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(reg_tree_spec %>% set_args(cost_complexity = tune()))
```

As above, Set up the adjustment grid and updated the parameters. Tune the cost complexity, in the range of -10 to -1 with the level 50.

```
reg_tree_param_grid <- grid_regular(cost_complexity(range = c(-10, -1)), levels = 50)

reg_tree_tune_res <- tune_grid(
  reg_tree_wf,
  resamples = pop_folds,
  grid = reg_tree_param_grid
)
```

```
## ! Fold1: internal: A correlation computation is required, but `estimate` is const...
```

Random Forest

To prepare, load the required objects that I saved in my script, tuned `mtry`, where `mtry` is the number of levels of the trees. Set mode to `"regression"` , and used the random Forest engine. Stored this model and my recipe in a workflow.

```
set.seed(1234)
rf_spec <- rand_forest(mtry = tune()) %>%
  set_engine("randomForest", importance = TRUE) %>%
  set_mode("regression")

rf_wf <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(rf_spec)
```

Next, set up the tuning grid, and updated the parameters to tune the number of level of the trees. The minimum number of level is 1 and the maximum number of level is 9.

```
rf_grid <- grid_regular(parameters(rf_spec) %>%
 update(mtry = mtry(range= c(1, 9))), levels = 9)
```

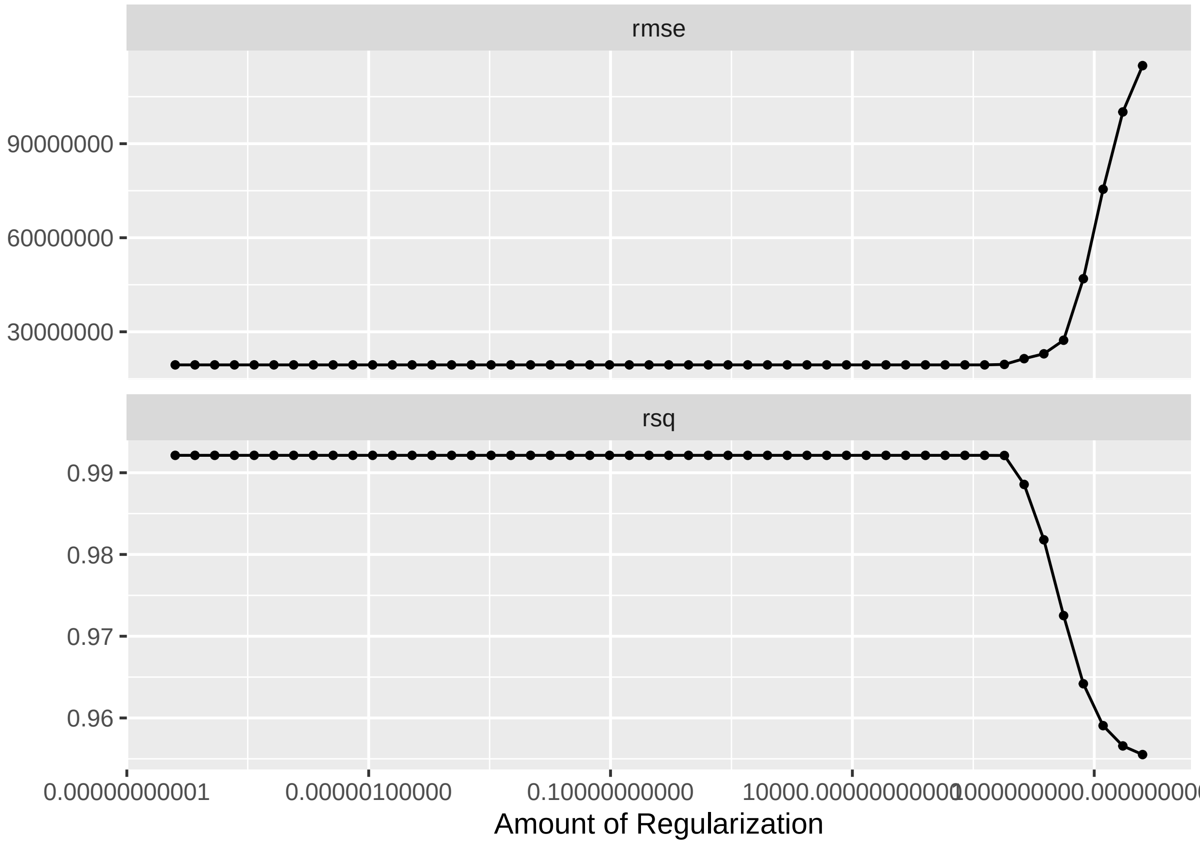
```
## Warning: `parameters.model_spec()` was deprecated in tune 0.1.6.9003.
## Please use `hardhat::extract_parameter_set_dials()` instead.
```

```
rf_tune_res <- tune_grid(
  rf_wf,
  resamples = pop_folds,
  grid = rf_grid
)
```

Repeated Cross Validation Parameter Tuning

Ridge Regression

```
autoplot(ridge_tune_res)
```



Use the `autoplot()`, we can observe the change of rmse and rsq when I tune the penalty of the ridge model. From the plot, we can see the rmse is small and stable, and rsq is large and stable at the beginning and has a sudden drop of rmse and sudden rise of rsq at the final. Therefore we pick the initial value as the best penalty value for the model.

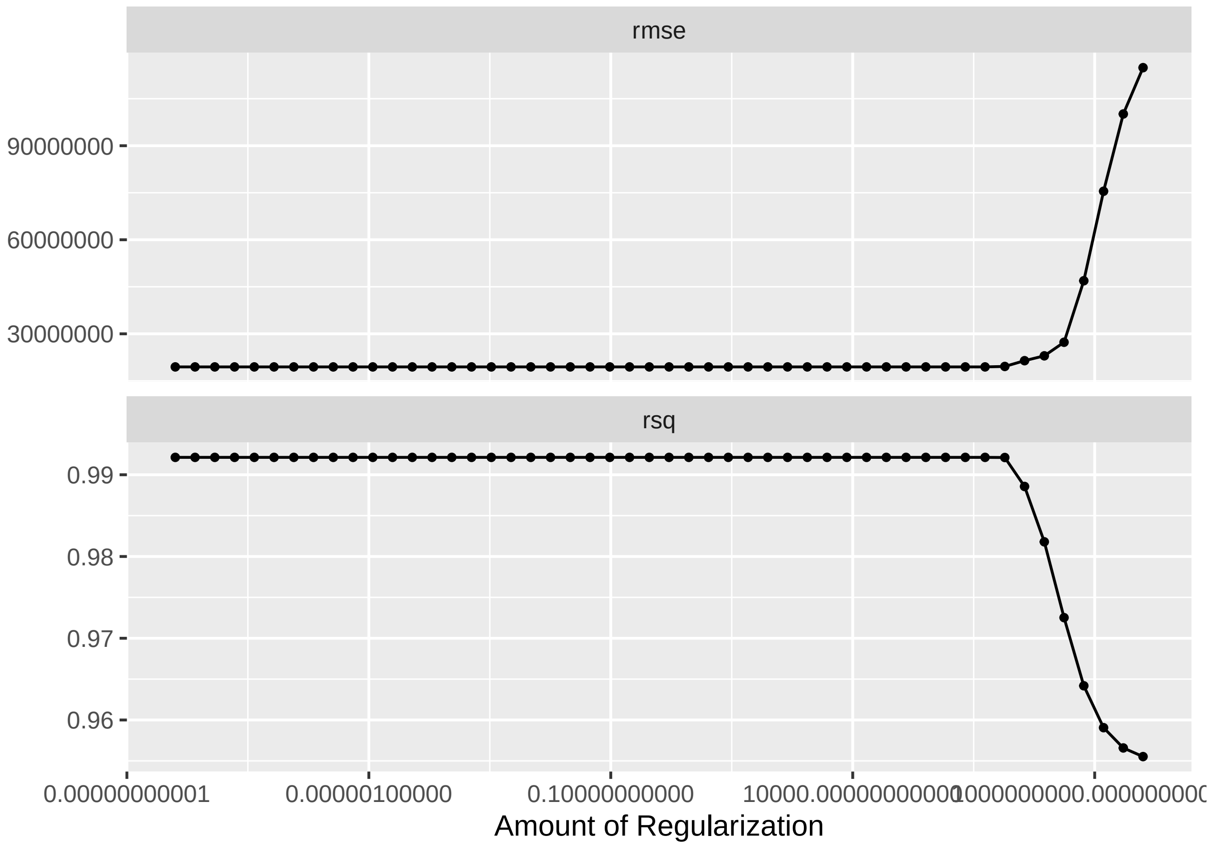
```
ridge_best <- select_best(ridge_tune_res, metric = "rsq")
ridge_best
```

```
## # A tibble: 1 x 2
##        penalty .config
##          <dbl> <chr>
## 1 0.0000000001 Preprocessor1_Model01
```

Using the `select_best()` function, the best value of penalty is 1e-10.

Lasso Regression

```
autoplot(lasso_tune_res)
```



The trend of this graph is similar to the one above.

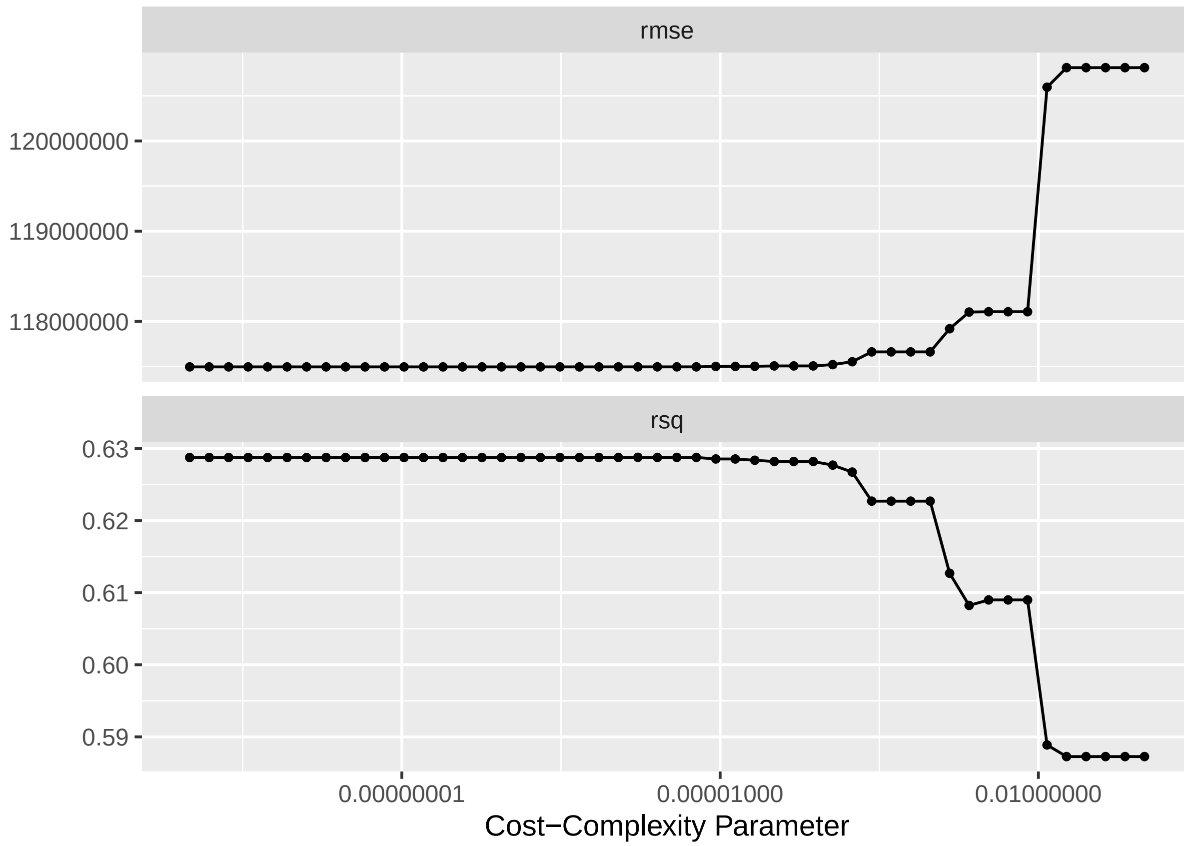
```
lasso_best <- select_best(lasso_tune_res, metric = "rsq")
lasso_best
```

```
## # A tibble: 1 x 2
##        penalty .config
##          <dbl> <chr>
## 1 0.0000000001 Preprocessor1_Model01
```

Also using the `select_best()` function, the best value of penalty is 1e-10.

Regression Tree

```
autoplot(reg_tree_tune_res)
```



For regression tree model, we tune the cost complexity parameter. From the plot, similar to ridge and lasso regression models, we can see the rmse is small and stable, and rsq is large and stable at the beginning and has a sudden drop of rmse and sudden rise of rsq at the final. Therefore we also pick the initial value as the best penalty value for the model.

```
reg_tree_best <- select_best(reg_tree_tune_res, metric = "rmse")
reg_tree_best
```

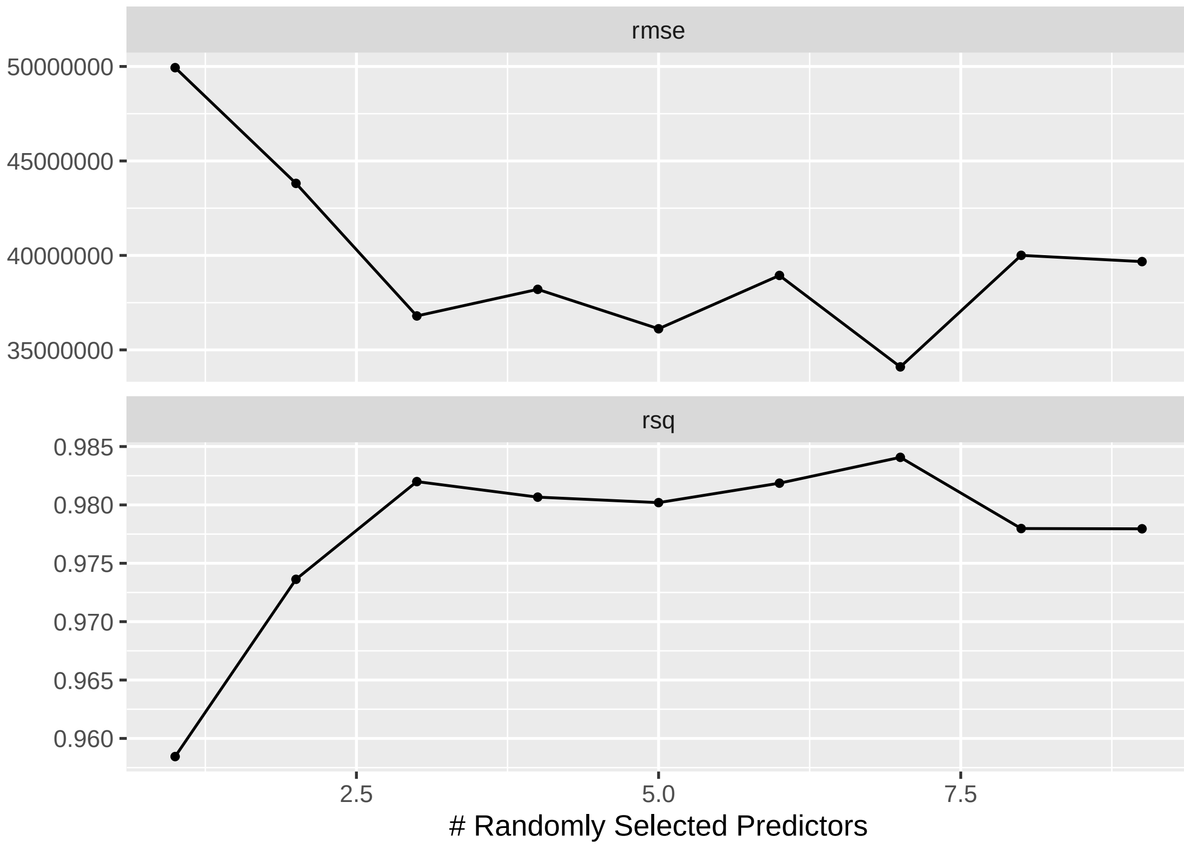
```
## # A tibble: 1 x 2
##   cost_complexity .config
##             <dbl> <chr>
## 1    0.0000000001 Preprocessor1_Model01
```

As same before, using the `select_best()` function, the best value of cost complexity is 1e-10.

Random Forest

The autoplot shows when the number of levels of tree increase, the rmse decrease and the rsq increase. However, the best value of rmse appears at level 5 and best value of rsq appears at level 5.

```
autoplot(rf_tune_res)
```



From the `autoplot()` we see the trend of rmse and rsq is getting better when we add layer to the trees. The best value of rmse and rsq appear when number of levels of trees is about 5.

```
rf_best <- select_best(rf_tune_res, metric = "rmse")
rf_best
```

```
## # A tibble: 1 x 2
##    mtry .config
##   <int> <chr>
## 1     7 Preprocessor1_Model7
```

Using the `show_best()` function, the best value of number of trees is config is 5 with $mtry = 5$.

Model performance evaluation and select best model

Create a workflow with an adjusted name so I can identify it. Use the $fit()$ function for each model to run the models on the test set and find the best model by comparing r squared value.

Final Ridge Regression Model

```
ridge_final <- finalize_workflow(ridge_workflow, ridge_best)
ridge_final_fit <- fit(ridge_final, data = pop_train)
augment(ridge_final_fit, new_data = pop_test) %>%
  rsq(truth = Population_2022, estimate = .pred)
```

```
## # A tibble: 1 x 3
```

```
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.980
```

Final Lasso Regression Model

```
lasso_final <- finalize_workflow(lasso_workflow, lasso_best)
lasso_final_fit <- fit(lasso_final, data = pop_train)
augment(lasso_final_fit, new_data = pop_test) %>%
  rsq(truth = Population_2022, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.998
```

Final Regression Tree Model

```
reg_tree_final <- finalize_workflow(reg_tree_wf, reg_tree_best)
reg_tree_final_fit <- fit(reg_tree_final, data = pop_train)
#reg_tree_final_fit %>%
  #extract_fit_engine() %>%
  #rpart.plot()
augment(reg_tree_final_fit, new_data = pop_test) %>%
  rsq(truth = Population_2022, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.845
```

Final Random Forest Model

```
rf_final <- finalize_workflow(rf_wf, rf_best)
rf_final_fit <- fit(rf_final, data = pop_train)
augment(rf_final_fit, new_data = pop_test) %>%
  rsq(truth = Population_2022, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.950
```

By comparison, Lasso Regression is the best model because it's r squared value is the largest.

Final model fitting and analysis

Compare the predictions of the four models across countries, actual values, and append them into a data frame for comparison.

```r
# append country, actual value, predict result from 4 models together in one data frame
data_pred_result <- df %>% select('Country', 'Population_2022')

data_pred_result['reg_tree_pred'] <- augment(reg_tree_final_fit, new_data = data)['.pred']

data_pred_result['rf_pred'] <- augment(rf_final_fit, new_data = data)['.pred']

data_pred_result['ridge_pred'] <- augment(ridge_final_fit, new_data = data)['.pred']

data_pred_result['lasso_pred'] <- augment(lasso_final_fit, new_data = data)['.pred']

data_pred_result
```

```
##                              Country Population_2022 reg_tree_pred
## 1                        Afghanistan        41128771   43945079.65
## 2                            Albania         2842321    2576001.67
## 3                            Algeria        44903225   43945079.65
## 4                     American Samoa           44273      45548.29
## 5                            Andorra           79824     102367.82
## 6                             Angola        35588987   43945079.65
## 7                           Anguilla           15857       8000.60
## 8                Antigua and Barbuda           93763     102367.82
## 9                          Argentina        45510318   43945079.65
## 10                           Armenia         2780469    2576001.67
## 11                             Aruba          106445     102367.82
## 12                         Australia        26177413   24063625.15
## 13                           Austria         8939617    7507049.91
## 14                        Azerbaijan        10358074   10751702.19
## 15                           Bahamas          409984     301374.29
## 16                           Bahrain         1472233    2576001.67
## 17                        Bangladesh       171186372  555085699.29
## 18                          Barbados          281635     301374.29
## 19                           Belarus         9534954   10751702.19
## 20                           Belgium        11655930   10751702.19
## 21                            Belize          405272     301374.29
## 22                             Benin        13352864   16328847.00
## 23                           Bermuda           64184      45548.29
## 24                            Bhutan          782455     830537.44
## 25                           Bolivia        12224110   10751702.19
## 26            Bosnia and Herzegovina         3233526    2576001.67
## 27                          Botswana         2630296    2576001.67
## 28                            Brazil       215313498  555085699.29
## 29            British Virgin Islands           31305      45548.29
## 30                            Brunei          449002     830537.44
## 31                          Bulgaria         6781953    7507049.91
## 32                      Burkina Faso        22673762   24063625.15
## 33                           Burundi        12889576   10751702.19
## 34                          Cambodia        16767842   16328847.00
## 35                          Cameroon        27914536   24063625.15
## 36                            Canada        38454327   43945079.65
## 37                        Cape Verde          593149     830537.44
## 38                    Cayman Islands           68706      45548.29
## 39          Central African Republic         5579144    4956415.73
## 40                              Chad        17723315   16328847.00
```

```
## 41                          Chile     19603733   24063625.15
## 42                          China   1425887337  555085699.29
## 43                       Colombia     51874024   43945079.65
## 44                        Comoros       836774     830537.44
## 45                   Cook Islands        17011       8000.60
## 46                     Costa Rica      5180829    4956415.73
## 47                        Croatia      4030358    4956415.73
## 48                           Cuba     11212191   10751702.19
## 49                        Curacao       191163     301374.29
## 50                         Cyprus      1251488     830537.44
## 51                 Czech Republic     10493986   10751702.19
## 52                        Denmark      5882261    4956415.73
## 53                       Djibouti      1120849     830537.44
## 54                       Dominica        72737     102367.82
## 55             Dominican Republic     11228821   10751702.19
## 56                       DR Congo     99010212   98478049.29
## 57                        Ecuador     18001000   16328847.00
## 58                          Egypt    110990103   98478049.29
## 59                    El Salvador      6336392    7507049.91
## 60              Equatorial Guinea      1674908    2576001.67
## 61                        Eritrea      3684032    4956415.73
## 62                        Estonia      1326062     830537.44
## 63                       Eswatini      1201670     830537.44
## 64                       Ethiopia    123379924   98478049.29
## 65               Falkland Islands         3780       8000.60
## 66                  Faroe Islands        53090      45548.29
## 67                           Fiji       929766     830537.44
## 68                        Finland      5540745    4956415.73
## 69                         France     64626628   98478049.29
## 70                  French Guiana       304557     301374.29
## 71               French Polynesia       306279     301374.29
## 72                          Gabon      2388992    2576001.67
## 73                         Gambia      2705992    2576001.67
## 74                        Georgia      3744385    4956415.73
## 75                        Germany     83369843   98478049.29
## 76                          Ghana     33475870   43945079.65
## 77                      Gibraltar        32649      45548.29
## 78                         Greece     10384971   10751702.19
## 79                      Greenland        56466      45548.29
## 80                        Grenada       125438     102367.82
## 81                     Guadeloupe       395752     301374.29
## 82                           Guam       171774     301374.29
## 83                      Guatemala     17843908   16328847.00
## 84                       Guernsey        63301      45548.29
## 85                         Guinea     13859341   16328847.00
## 86                  Guinea-Bissau      2105566    2576001.67
## 87                         Guyana       808726     830537.44
## 88                          Haiti     11584996   10751702.19
## 89                       Honduras     10432860   10751702.19
## 90                      Hong Kong      7488865    7507049.91
## 91                        Hungary      9967308   10751702.19
## 92                        Iceland       372899     301374.29
## 93                          India   1417173173  555085699.29
## 94                      Indonesia    275501339  555085699.29
```

```
## 95                           Iran      88550570   98478049.29
## 96                           Iraq      44496122   43945079.65
## 97                        Ireland       5023109    4956415.73
## 98                    Isle of Man         84519     102367.82
## 99                         Israel       9038309    7507049.91
## 100                         Italy      59037474   43945079.65
## 101                   Ivory Coast      28160542   24063625.15
## 102                       Jamaica       2827377    2576001.67
## 103                         Japan     123951692   98478049.29
## 104                        Jersey        110778     102367.82
## 105                        Jordan      11285869   10751702.19
## 106                    Kazakhstan      19397998   24063625.15
## 107                         Kenya      54027487   43945079.65
## 108                      Kiribati        131232     102367.82
## 109                        Kuwait       4268873    4956415.73
## 110                    Kyrgyzstan       6630623    7507049.91
## 111                          Laos       7529475    7507049.91
## 112                        Latvia       1850651    2576001.67
## 113                       Lebanon       5489739    4956415.73
## 114                       Lesotho       2305825    2576001.67
## 115                       Liberia       5302681    4956415.73
## 116                         Libya       6812341    7507049.91
## 117                 Liechtenstein         39327      45548.29
## 118                     Lithuania       2750055    2576001.67
## 119                    Luxembourg        647599     830537.44
## 120                         Macau        695168     830537.44
## 121                    Madagascar      29611714   24063625.15
## 122                        Malawi      20405317   24063625.15
## 123                      Malaysia      33938221   43945079.65
## 124                      Maldives        523787     830537.44
## 125                          Mali      22593590   24063625.15
## 126                         Malta        533286     830537.44
## 127              Marshall Islands         41569      45548.29
## 128                    Martinique        367507     301374.29
## 129                    Mauritania       4736139    4956415.73
## 130                     Mauritius       1299469     830537.44
## 131                       Mayotte        326101     102367.82
## 132                        Mexico     127504125   98478049.29
## 133                    Micronesia        114164     102367.82
## 134                       Moldova       3272996    2576001.67
## 135                        Monaco         36469      45548.29
## 136                      Mongolia       3398366    2576001.67
## 137                    Montenegro        627082     830537.44
## 138                    Montserrat          4390       8000.60
## 139                       Morocco      37457971   43945079.65
## 140                    Mozambique      32969517   24063625.15
## 141                       Myanmar      54179306   43945079.65
## 142                       Namibia       2567012    2576001.67
## 143                         Nauru         12668       8000.60
## 144                         Nepal      30547580   24063625.15
## 145                   Netherlands      17564014   16328847.00
## 146                 New Caledonia        289950     301374.29
## 147                   New Zealand       5185288    4956415.73
## 148                     Nicaragua       6948392    7507049.91
```

```
## 149                           Niger  26207977  24063625.15
## 150                         Nigeria 218541212  98478049.29
## 151                            Niue      1934     8000.60
## 152                     North Korea  26069416  24063625.15
## 153                 North Macedonia   2093599   2576001.67
## 154        Northern Mariana Islands     49551    45548.29
## 155                          Norway   5434319   4956415.73
## 156                            Oman   4576298   4956415.73
## 157                        Pakistan 235824862 555085699.29
## 158                           Palau     18055    45548.29
## 159                       Palestine   5250072   4956415.73
## 160                          Panama   4408581   4956415.73
## 161                Papua New Guinea  10142619  10751702.19
## 162                        Paraguay   6780744   7507049.91
## 163                            Peru  34049588  43945079.65
## 164                     Philippines 115559009  98478049.29
## 165                          Poland  39857145  43945079.65
## 166                        Portugal  10270865  10751702.19
## 167                     Puerto Rico   3252407   2576001.67
## 168                           Qatar   2695122   2576001.67
## 169           Republic of the Congo   5970424   4956415.73
## 170                         Reunion    974052    830537.44
## 171                         Romania  19659267  24063625.15
## 172                          Russia 144713314 555085699.29
## 173                          Rwanda  13776698  16328847.00
## 174               Saint Barthelemy     10967     8000.60
## 175           Saint Kitts and Nevis     47657    45548.29
## 176                     Saint Lucia    179857   301374.29
## 177                    Saint Martin     31791    45548.29
## 178       Saint Pierre and Miquelon      5862     8000.60
## 179 Saint Vincent and the Grenadines    103948   102367.82
## 180                           Samoa    222382   301374.29
## 181                      San Marino     33660    45548.29
## 182           Sao Tome and Principe    227380   301374.29
## 183                    Saudi Arabia  36408820  43945079.65
## 184                         Senegal  17316449  16328847.00
## 185                          Serbia   7221365   7507049.91
## 186                      Seychelles    107118   102367.82
## 187                    Sierra Leone   8605718   7507049.91
## 188                       Singapore   5975689   4956415.73
## 189                    Sint Maarten     44175    45548.29
## 190                        Slovakia   5643453   4956415.73
## 191                        Slovenia   2119844   2576001.67
## 192                 Solomon Islands    724273   830537.44
## 193                         Somalia  17597511  16328847.00
## 194                    South Africa  59893885  43945079.65
## 195                     South Korea  51815810  43945079.65
## 196                     South Sudan  10913164  10751702.19
## 197                           Spain  47558630  43945079.65
## 198                       Sri Lanka  21832143  24063625.15
## 199                           Sudan  46874204  43945079.65
## 200                        Suriname    618040    830537.44
## 201                          Sweden  10549347  10751702.19
## 202                     Switzerland   8740472   7507049.91
```

```
## 203                         Syria        22125249   24063625.15
## 204                        Taiwan        23893394   24063625.15
## 205                    Tajikistan         9952787   10751702.19
## 206                      Tanzania        65497748   98478049.29
## 207                      Thailand        71697030   98478049.29
## 208                   Timor-Leste         1341296     830537.44
## 209                          Togo         8848699    7507049.91
## 210                       Tokelau            1871       8000.60
## 211                         Tonga          106858     102367.82
## 212           Trinidad and Tobago         1531044    2576001.67
## 213                       Tunisia        12356117   10751702.19
## 214                        Turkey        85341241   98478049.29
## 215                  Turkmenistan         6430770    7507049.91
## 216      Turks and Caicos Islands           45703      45548.29
## 217                        Tuvalu           11312       8000.60
## 218                        Uganda        47249585   43945079.65
## 219                       Ukraine        39701739   43945079.65
## 220          United Arab Emirates         9441129   10751702.19
## 221                United Kingdom        67508936   98478049.29
## 222                 United States       338289857  555085699.29
## 223   United States Virgin Islands           99465     102367.82
## 224                       Uruguay         3422794    2576001.67
## 225                    Uzbekistan        34627652   43945079.65
## 226                       Vanuatu          326740     301374.29
## 227                  Vatican City             510       8000.60
## 228                     Venezuela        28301696   24063625.15
## 229                       Vietnam        98186856   98478049.29
## 230              Wallis and Futuna           11572       8000.60
## 231                Western Sahara          575986     830537.44
## 232                         Yemen        33696614   43945079.65
## 233                        Zambia        20017675   24063625.15
## 234                      Zimbabwe        16320537   16328847.00
##             rf_pred   ridge_pred lasso_pred
## 1      38498025.920   33672224.7   38244672
## 2       2782363.963    2187351.6    4292551
## 3      45171477.550   40014858.7   43206887
## 4         45398.560     627468.9    1479620
## 5         73319.664   -1253675.9    1506739
## 6      33414289.910   27202352.1   32692585
## 7         11919.553    1043171.0    1446756
## 8         93225.922    1124053.3    1522878
## 9      46040757.278   44786518.4   45599617
## 10      2784505.989    9047235.6    4253514
## 11       106438.041    1135049.8    1536892
## 12     40270990.455   16565230.5   26287050
## 13      9268569.876    8469234.7   10200830
## 14     10340911.531   15546392.0   11512534
## 15       433506.156    1407875.5    1831151
## 16      1464938.329    6568664.1    2858832
## 17    156669138.987  162066112.2  164486089
## 18       351939.351    1351653.7    1710168
## 19      9888133.614   10328493.4   11050913
## 20     11506733.907   11454886.5   12825343
## 21       388147.251    1331918.2    1811507
```

```
## 22    12806557.622   14052685.0   13367665
## 23       64175.459    1102081.2    1495066
## 24      812996.556    6152248.4    2189748
## 25    12465122.347   12399137.2   12994387
## 26     3344750.333    3288318.9    4815052
## 27     2598849.300    5962619.8    3875138
## 28   231687638.696  203207927.4  210676841
## 29       30621.430    1056032.6    1461832
## 30      476163.093    5847446.4    1863765
## 31     7116266.674    8124489.5    8508113
## 32    21834804.676   20908562.5   21791483
## 33    12632318.859   13999127.5   13029671
## 34    17554875.908   19964902.7   17385540
## 35    27584762.929   24619601.9   26481416
## 36    50810061.104   28113323.8   38337502
## 37      606492.735    4997571.8    2000196
## 38       52532.606    1081396.2    1496223
## 39     5480379.710    8510310.3    6552035
## 40    17039318.353   15266323.0   17048716
## 41    19837773.673   20666414.4   20104385
## 42  1245388044.158 1490537719.9 1408491018
## 43    51219474.301   49097941.6   50692127
## 44      807271.749    5130645.4    2205390
## 45       16782.095     591529.4    1448854
## 46     5435149.876    5741410.9    6446243
## 47     4773977.252    3979473.1    5571918
## 48    11450250.359   13934852.7   12700349
## 49      214891.922    1224489.9    1612597
## 50     1364936.739    -142890.5    2644799
## 51    10484124.880   11084218.2   11916067
## 52     5952241.285    5182691.3    7176480
## 53     1094717.540    5323907.1    2485158
## 54       78693.008    1113264.6    1502576
## 55    11148890.260   11350722.6   12158314
## 56    73028155.273   71215435.4   88465966
## 57    18299332.392   18238805.7   18414133
## 58   119639438.476   95334015.1  104722026
## 59     6950975.461    7791664.0    7674607
## 60     1523670.092    5516458.0    2925735
## 61     3560168.374    7524221.2    4890115
## 62     1461606.944     302227.2    2749898
## 63     1309019.155    5580493.3    2589508
## 64   117519377.287   94407499.0  112502461
## 65        5371.131    2759647.9    1435243
## 66       50880.861   -1270710.8    1482461
## 67      963994.325    1536734.8    2347003
## 68     5741434.126    4584723.8    6918980
## 69    66310462.460   69981931.5   65387068
## 70      274393.477    2900653.9    1708410
## 71      340647.536     860920.5    1728428
## 72     2223309.862    5951576.9    3613630
## 73     2581504.867    6405158.7    3872143
## 74     3873929.123   10596330.8    5183891
## 75   110371032.496   96013028.0   83933571
```

```
## 76    32860980.380   30608333.3   32211739
## 77       34444.975   -1289852.1    1464142
## 78    10778254.723   11061355.1   12012829
## 79       57968.705   -1238732.4    1487384
## 80      124231.685    1161901.6    1552998
## 81      441191.284    1494861.5    1826957
## 82      202308.077     749255.4    1599689
## 83    17907651.918   15887354.3   18201090
## 84       63113.304   -1254917.8    1493728
## 85    13103081.769   15067282.0   13977561
## 86     1960447.314    6066897.0    3352251
## 87      866184.974    3453759.5    2209390
## 88    11337342.366   11428511.9   12407073
## 89    10307197.005    9466584.5   11194734
## 90     7557573.438   13250750.9    8862712
## 91    10066559.422   10798053.6   11177517
## 92      424682.645   -1082252.1    1783158
## 93  1242618639.349 1303430449.0 1363876925
## 94   231115711.805  263491368.2  267273791
## 95    95171407.202   83031238.8   86254332
## 96    44250570.344   38130297.0   41975081
## 97     5336926.309    3537406.2    6249903
## 98       86600.719   -1235316.5    1515164
## 99     8912473.792   13088146.7    9865810
## 100   65270394.541   68392534.4   60965062
## 101   28470982.094   25461614.9   26901582
## 102    2751546.425    4089170.8    4230403
## 103  136940071.402  153973656.8  126923357
## 104     106873.750   -1216641.8    1536541
## 105   10808957.910   12919121.3   11765962
## 106   19714255.425   22260005.0   19894425
## 107   49416932.561   45392874.7   51236424
## 108     122138.202     684857.8    1553845
## 109    4268413.659    8645212.6    5601093
## 110    6414009.021   11336067.3    7634669
## 111    7292482.097   11814283.9    8517022
## 112    2038837.655    1321969.3    3357123
## 113    5753407.410   10971143.6    7352682
## 114    2304659.879    6680119.7    3624484
## 115    5145697.083    8447303.0    6316196
## 116    6839044.651    8739625.7    7881225
## 117      38428.051   -1286824.7    1469700
## 118    2834170.017    2538504.1    4295096
## 119     627446.231    -750005.9    2036078
## 120     682164.356    6027666.3    2081751
## 121   28887415.066   25967194.0   28248162
## 122   19478944.984   19715087.1   19795090
## 123   33994742.398   34092360.0   33678014
## 124     519307.219    5844849.4    1913763
## 125   21695822.474   19540501.9   21376834
## 126     549505.356    -828509.0    1922361
## 127      43571.431     624717.9    1477168
## 128     451034.664    1484549.4    1805514
## 129    4551499.344    6909485.1    5700148
```

```
## 130    1399704.881    5856627.9    2722283
## 131     245503.636    4656405.1    1714511
## 132  128963081.179  118188384.0  124667909
## 133     112218.012     690520.0    1542269
## 134    3307370.121    3166453.9    4577344
## 135      37038.678   -1286910.5    1468347
## 136    3024364.137    6772641.4    4586149
## 137     684802.202    -597194.2    2059955
## 138       5849.719    1037837.7    1436336
## 139   38746444.083   38770748.3   37199298
## 140   30762534.851   28080193.7   30821286
## 141   54485531.376   58691301.5   53890570
## 142    2565412.399    5731072.8    3831404
## 143      12142.190     582839.2    1443472
## 144   30299074.781   33676754.6   29993005
## 145   18406127.621   17803288.3   18643134
## 146     329707.256     822413.1    1715563
## 147    5293824.751    5146102.5    6291658
## 148    6806157.770    7097428.3    7984442
## 149   24049754.048   20434549.3   24054696
## 150  176233809.147  170666107.0  199581366
## 151       3764.586     573485.5    1433539
## 152   29881516.213   32588176.3   26958317
## 153    2256045.174    1086413.2    3532745
## 154      50816.795     627715.7    1481759
## 155    5688363.840    3991720.5    6716597
## 156    4467154.204    8428216.4    5821586
## 157  212349786.569  198309842.0  221473615
## 158      18252.738     590359.4    1449470
## 159    5116783.345    9447424.9    6225514
## 160    4491163.570    4757984.2    5579101
## 161    9941170.398    7732418.1   10732234
## 162    6753465.609    8407976.4    7854147
## 163   34213706.428   32792359.5   33605642
## 164  121432414.019  103446636.0  109650837
## 165   41425828.694   43670764.3   39747969
## 166   10359868.406   10780980.8   11712745
## 167    3341607.147    5180675.7    4776001
## 168    2404842.444    7091637.4    4048210
## 169    5682454.215    8552000.3    6866031
## 170     971965.320    5387117.6    2371961
## 171   20807358.856   23581351.6   20970420
## 172  169872122.714  153226955.2  146080326
## 173   12974748.573   15406735.2   13948004
## 174      10735.859    1038218.8    1441880
## 175      47888.680    1084295.0    1479141
## 176     204567.351    1216588.4    1608750
## 177      34499.150    1062184.7    1465002
## 178       5381.366    1036477.6    1437550
## 179     118520.157    1159300.4    1536546
## 180     274458.652     792554.0    1641330
## 181      34373.043   -1292203.0    1465339
## 182     201823.923    4634458.1    1642669
## 183   36988987.680   31786035.6   36036379
```

```
## 184   17163110.685   17355265.0   17003448
## 185    7492034.006    7801442.7    8820851
## 186     104196.520    4546158.0    1534325
## 187    8386257.730   11172955.5    9279606
## 188    5802090.704   10662000.7    7217494
## 189      40318.507    1065646.3    1473773
## 190    5783875.356    4972037.4    6853296
## 191    2227712.954    1071445.6    3526402
## 192     685309.914    1098963.6    2089905
## 193   17259489.930   16362850.4   16838657
## 194   60510309.889   58654868.1   58869537
## 195   54421242.355   60896181.1   52735358
## 196   11101178.996   13059127.7   12218814
## 197   49124249.301   49507576.9   48241660
## 198   22417237.494   28049315.6   22911623
## 199   44915949.610   37963110.9   43288147
## 200     632543.318    3195000.5    2024093
## 201   10346899.388    9293815.6   11558642
## 202    9216770.847    7653309.8    9897759
## 203   22833927.808   24884231.4   21518029
## 204   26685408.267   31337493.6   25035676
## 205    9830782.935   13458215.0   10545152
## 206   59364988.920   50718008.6   59373510
## 207   77716458.576   78516316.2   72158043
## 208    1357406.381    6604945.4    2690188
## 209    8364320.252   11158769.4    9468156
## 210       3933.439     572659.9    1433318
## 211     109587.897     692785.7    1536789
## 212    1553367.675    2634666.7    2921291
## 213   12590070.092   15975375.7   13311672
## 214   86446239.475   83558843.0   83499239
## 215    6302338.514   10533547.2    7470865
## 216      33227.756    1058051.2    1472772
## 217      11864.629     582286.6    1442590
## 218   44192054.497   37766932.6   43000632
## 219   52530696.942   54976865.7   45569310
## 220    9478587.184   11897404.6   10538568
## 221   69638818.894   72334719.8   67509998
## 222  241047924.042  334457003.5  331639277
## 223     108883.757    1148531.6    1532569
## 224    3418355.881    6481530.8    4836416
## 225   34490867.313   35398746.5   33832609
## 226     304780.631     813455.1    1728541
## 227       3520.225   -1325267.8    1432176
## 228   30594388.430   30703427.1   30584029
## 229  119203747.042   98727948.7   95972696
## 230      12696.058     585581.4    1443449
## 231     544164.258    4559827.7    1960799
## 232   33194655.646   29971036.4   32139667
## 233   18427524.021   17964745.8   19255244
## 234   17810879.839   18078317.5   16456481
```

From the Table, In many countries, the model predictions are quite different from the actual values. For example, the actual value of American Samoa's population is 44273, but its Lasso Regression prediction result is 1507434; the actual population value of Anguilla is 15857, but its Lasso Regression prediction result is 1476729. These predictions are much larger than actual values, so the predictions are not accurate.

Advantage of lasso model: the value of its metric r squared value is the highest and closest to the actual value among the four models. In the other three models, there are negative numbers in the prediction results, but Lasso model's predict result stays positive in all its predicted values.

Conclusion

Through research, testing and analysis, Lasso Regression performed best in my models, Ridge Regression and Random forest are not bad, and Regression Tree did not perform well. Although the value of the r squared value of Lasso Regression, Random Forest and Ridge Regression is very close to 1, there is still a gap between the prediction results of many countries and their real population values. Because a lot of factors such as people's life expectancy, fertility rate and death rate are all related to population growth, there is no way to predict the exact future world population.

In addition, the regression models has its limitation that it predicts a similar pattern for each continents or level of size of countries. However, from EDA at the beginning, even in the same continent or shares the similar value of areas, the country's growth rate differs. Therefore, for future research I propose to find more features to improve our model, such that gross national happiness index, and GDP, etc. We can also try more complicated models such as neuro network model.

Overall, this World Population Modeling project provided me with a great opportunity to gain experience and improve my skills through data analysis and machine learning techniques.