Can We Predict Life Expectancy Using Historical Time Series Data?

Bhavishya Chowdary Katragadda and Nanda Vennela Kakarla

2023-11-12

Introduction

We will use the time series analysis then Random Forest Regression and Linear Regression to examine whether we can predict life expectancy given historical time series data.

Loading Libraries

```
#install.packages("randomForest")

# loading libraries
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.2

## Warning: package 'tidyr' was built under R version 4.3.2
```

```
library(forecast)
library(caret)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.2
```

```
library(gridExtra)
```

Loading the Datasets

We first need to load the data into R environment

```
# loading life expectancy data
life_ex <- read.csv("C:/Users/bhavi/OneDrive/Documents/Desktop/Applied Project/Datasets/csv/Life Expecta
head(life_ex)
```

```
##                   Country.Name Country.Code
## 1                        Aruba          ABW
## 2 Africa Eastern and Southern          AFE
```

```
## 3                  Afghanistan          AFG
## 4  Africa Western and Central          AFW
## 5                       Angola          AGO
## 6                      Albania          ALB
##                             Indicator.Name Indicator.Code    X1960    X1961
## 1 Life expectancy at birth, total (years) SP.DYN.LE00.IN 64.15200 64.53700
## 2 Life expectancy at birth, total (years) SP.DYN.LE00.IN 44.08555 44.38670
## 3 Life expectancy at birth, total (years) SP.DYN.LE00.IN 32.53500 33.06800
## 4 Life expectancy at birth, total (years) SP.DYN.LE00.IN 37.84515 38.16495
## 5 Life expectancy at birth, total (years) SP.DYN.LE00.IN 38.21100 37.26700
## 6 Life expectancy at birth, total (years) SP.DYN.LE00.IN 54.43900 55.63400
##      X1962    X1963    X1964    X1965    X1966    X1967    X1968    X1969
## 1 64.75200 65.13200 65.29400 65.50200 66.06300 66.43900 66.75700 67.16800
## 2 44.75218 44.91316 45.47904 45.49834 45.24910 45.92491 46.22310 46.43230
## 3 33.54700 34.01600 34.49400 34.95300 35.45300 35.92400 36.41800 36.91000
## 4 38.73510 39.06372 39.33536 39.61804 39.83783 39.47150 40.08568 40.35042
## 5 37.53900 37.82400 38.13100 38.49500 38.75700 39.09200 39.48400 39.82900
## 6 56.67100 57.84400 58.98300 60.01900 60.99800 61.97200 62.94600 63.92300
##      X1970    X1971    X1972    X1973    X1974    X1975    X1976    X1977
## 1 67.58300 67.97500 68.57700 69.09200 69.50300 69.76200 70.03500 70.26400
## 2 46.71848 47.19294 46.89739 47.69232 47.59806 47.75989 48.34959 48.63591
## 3 37.41800 37.92300 38.44400 39.00300 39.55000 40.10000 40.64500 41.22800
## 4 41.03476 41.55672 42.24979 42.85505 43.49768 44.20125 45.00316 45.71989
## 5 40.19000 40.55400 40.90500 41.27000 41.65200 41.19100 41.16300 41.43700
## 6 64.82400 65.61800 66.42200 67.14000 67.76900 68.32800 68.70400 69.12100
##      X1978    X1979    X1980    X1981    X1982    X1983    X1984    X1985
## 1 70.49400 70.77800 71.06600 71.72200 71.95900 72.10500 72.25100 72.38800
## 2 48.76360 49.26134 49.63654 50.05707 50.29685 48.70333 48.65266 49.01163
## 3 40.27100 39.08600 39.61800 40.16400 37.76600 38.18700 33.32900 33.55000
## 4 46.26954 46.67374 47.01524 47.29719 47.52938 47.78526 47.93192 48.02168
## 5 41.83000 42.17500 42.44900 42.77200 43.05100 42.09200 42.35300 42.64800
## 6 69.30900 69.58400 70.47800 70.73000 71.02300 71.29600 71.50200 71.65600
##      X1986    X1987    X1988    X1989    X1990    X1991    X1992    X1993
## 1 72.46200 72.78900 73.04700 73.02300 73.07600 73.10000 73.17900 73.22500
## 2 49.63972 50.07589 49.35973 50.68410 50.60773 50.39046 49.96211 50.27363
## 3 39.39600 39.84400 43.95800 45.15800 45.96700 46.66300 47.59600 51.46600
## 4 48.06676 48.23785 48.51291 48.68985 48.65000 48.66246 48.73727 48.83204
## 5 42.84300 40.91700 41.54500 41.76500 41.89300 43.81300 42.20900 42.10100
## 6 71.95000 72.35200 72.64100 72.88000 73.14400 73.37800 73.71500 73.93900
##      X1994    X1995    X1996    X1997    X1998    X1999    X2000    X2001
## 1 73.27200 73.34900 73.44800 73.45200 73.49100 73.56100 73.56900 73.64700
## 2 50.88258 51.00193 50.81069 50.97423 50.32591 51.23785 51.96448 52.18965
## 3 51.49500 52.54400 53.24300 53.63400 52.94300 54.84600 55.29800 55.79800
## 4 48.68189 48.78377 48.90628 49.07918 49.33295 49.75012 50.22195 50.56514
## 5 43.42200 45.84900 46.03300 46.30600 45.05700 45.38600 46.02400 46.59000
## 6 74.13100 74.36200 74.59200 73.90400 74.99000 75.18300 75.40400 75.63900
##      X2002    X2003    X2004    X2005    X2006    X2007    X2008    X2009
## 1 73.72600 73.75200 73.57600 73.81100 74.02600 74.21000 74.14700 74.56000
## 2 52.54079 53.02203 53.54546 54.21965 55.15055 55.93380 56.68042 57.62085
## 3 56.45400 57.34400 57.94400 58.36100 58.68400 59.11100 59.85200 60.36400
## 4 50.92785 51.40336 51.81913 52.34455 52.83213 53.25171 53.64116 54.15942
## 5 47.38600 49.61700 50.59200 51.57000 52.36900 53.64200 54.63300 55.75200
## 6 75.89000 76.14200 76.37600 76.62100 76.81600 77.54900 77.65300 77.78100
##      X2010    X2011    X2012    X2013    X2014    X2015    X2016    X2017
```

```
## 1 75.40400 75.46500 75.53100 75.63600 75.60100 75.68300 75.61700 75.90300
## 2 58.41115 59.29327 60.05078 60.70987 61.33792 61.85646 62.44405 62.92239
## 3 60.85100 61.41900 61.92300 62.41700 62.54500 62.65900 63.13600 63.01600
## 4 54.55017 55.01314 55.34056 55.67341 55.92223 56.19587 56.58168 56.88845
## 5 56.72600 57.59600 58.62300 59.30700 60.04000 60.65500 61.09200 61.68000
## 6 77.93600 78.09200 78.06400 78.12300 78.40700 78.64400 78.86000 79.04700
##      X2018    X2019    X2020    X2021 X2022
## 1 76.07200 76.24800 75.72300 74.62600    NA
## 2 63.36586 63.75568 63.31386 62.45459    NA
## 3 63.08100 63.56500 62.57500 61.98200    NA
## 4 57.18914 57.55580 57.22637 56.98866    NA
## 5 62.14400 62.44800 62.26100 61.64300    NA
## 6 79.18400 79.28200 76.98900 76.46300    NA
```

```r
# loading population data
pop_df <- read.csv("C:/Users/bhavi/OneDrive/Documents/Desktop/Applied Project/Datasets/csv/Population Da
head(pop_df)
```

```
##                  Country.Name Country.Code                     Region
## 1                       Aruba          ABW Latin America & Caribbean
## 2 Africa Eastern and Southern          AFE
## 3                 Afghanistan          AFG                South Asia
## 4  Africa Western and Central          AFW
## 5                      Angola          AGO        Sub-Saharan Africa
## 6                     Albania          ALB      Europe & Central Asia
##           IncomeGroup    Indicator.Name Indicator.Code     X1960     X1961
## 1         High income Population, total    SP.POP.TOTL     54608     55811
## 2                     Population, total    SP.POP.TOTL 130692579 134169237
## 3          Low income Population, total    SP.POP.TOTL   8622466   8790140
## 4                     Population, total    SP.POP.TOTL  97256290  99314028
## 5 Lower middle income Population, total    SP.POP.TOTL   5357195   5441333
## 6 Upper middle income Population, total    SP.POP.TOTL   1608800   1659800
##       X1962     X1963     X1964     X1965     X1966     X1967     X1968
## 1     56682     57475     58178     58782     59291     59522     59471
## 2 137835590 141630546 145605995 149742351 153955516 158313235 162875171
## 3   8969047   9157465   9355514   9565147   9783147  10010030  10247780
## 4 101445032 103667517 105959979 108336203 110798486 113319950 115921723
## 5   5521400   5599827   5673199   5736582   5787044   5827503   5868203
## 6   1711319   1762621   1814135   1864791   1914573   1965598   2022272
##       X1969     X1970     X1971     X1972     X1973     X1974     X1975
## 1     59330     59106     58816     58855     59365     60028     60715
## 2 167596160 172475766 177503186 182599092 187901657 193512956 199284304
## 3  10494489  10752971  11015857  11286753  11575305  11869879  12157386
## 4 118615741 121424797 124336039 127364044 130563107 133953892 137548613
## 5   5928386   6029700   6177049   6364731   6578230   6802494   7032713
## 6   2081695   2135479   2187853   2243126   2296752   2350124   2404831
##       X1976     X1977     X1978     X1979     X1980     X1981     X1982
## 1     61193     61465     61738     62006     62267     62614     63116
## 2 205202669 211120911 217481420 224315978 230967858 237937461 245386717
## 3  12425267  12687301  12938862  12986369  12486631  11155195  10088289
## 4 141258400 145122851 149206663 153459665 157825609 162323313 167023385
## 5   7266780   7511895   7771590   8043218   8330047   8631457   8947152
## 6   2458526   2513546   2566266   2617832   2671997   2726056   2784278
##       X1983     X1984     X1985     X1986     X1987     X1988     X1989
```

```
## 1     63683     64174     64478     64553     64450     64332     64596
## 2 252779730 260209149 267938123 276035920 284490394 292795186 301124880
## 3   9951449  10243686  10512221  10448442  10322758  10383460  10673168
## 4 171566640 176054495 180817312 185720244 190759952 195969722 201392200
## 5   9276707   9617702   9970621  10332574  10694057  11060261  11439498
## 6   2843960   2904429   2964762   3022635   3083605   3142336   3227943
##       X1990     X1991     X1992     X1993     X1994     X1995     X1996
## 1     65712     67864     70192     72360     74710     77050     79417
## 2 309890664 318544083 326933522 335625136 344418362 353466601 362985802
## 3  10694796  10745167  12057433  14003760  15455555  16418912  17106595
## 4 206739024 212172888 217966101 223788766 229675775 235861484 242200260
## 5  11828638  12228691  12632507  13038270  13462031  13912253  14383350
## 6   3286542   3266790   3247039   3227287   3207536   3187784   3168033
##       X1997     X1998     X1999     X2000     X2001     X2002     X2003
## 1     81858     84355     86867     89101     90691     91781     92701
## 2 372352230 381715600 391486231 401600588 412001885 422741118 433807484
## 3  17788819  18493132  19262847  19542982  19688632  21000256  22645130
## 4 248713095 255482918 262397030 269611898 277160097 284952322 292977949
## 5  14871146  15366864  15870753  16394062  16941587  17516139  18124342
## 6   3148281   3128530   3108778   3089027   3060173   3051010   3039616
##       X2004     X2005     X2006     X2007     X2008     X2009     X2010
## 1     93540     94483     95606     96787     97996     99212    100341
## 2 445281555 457153837 469508516 482406426 495748900 509410477 523459657
## 3  23553551  24411191  25442944  25903301  26427199  27385307  28189672
## 4 301265247 309824829 318601484 327612838 336893835 346475221 356337762
## 5  18771125  19450959  20162340  20909684  21691522  22507674  23364185
## 6   3026939   3011487   2992547   2970017   2947314   2927519   2913021
##       X2011     X2012     X2013     X2014     X2015     X2016     X2017
## 1    101288    102112    102880    103594    104257    104874    105439
## 2 537792950 552530654 567892149 583651101 600008424 616377605 632746570
## 3  29249157  30466479  31541209  32716210  33753499  34636207  35643418
## 4 366489204 376797999 387204553 397855507 408690375 419778384 431138704
## 5  24259111  25188292  26147002  27128337  28127721  29154746  30208628
## 6   2905195   2900401   2895092   2889104   2880703   2876101   2873457
##       X2018     X2019     X2020     X2021     X2022
## 1    105962    106442    106585    106537    106445
## 2 649757148 667242986 685112979 702977106 720839314
## 3  36686784  37769499  38972230  40099462  41128771
## 4 442646825 454306063 466189102 478185907 490330870
## 5  31273533  32353588  33428486  34503774  35588987
## 6   2866376   2854191   2837849   2811666   2775634
```

Preprocessing the Population Dataframe

We can the pivot the Population data set because our initial data had years as columns, this will help us process the data to change the years into rows for easier analysis.

```r
# gather columns into key-value pairs
pop_df_long <- gather(pop_df, key = "Year", value = "Population", -Country.Name, -Country.Code, -Region

# convert 'Year' to numeric (if needed)
pop_df_long$Year <- as.numeric(gsub("X", "", pop_df_long$Year))
```

```r
# drop unnecessary columns from the population data
pop_df_long <- pop_df_long %>%
  select(-Indicator.Name, -Indicator.Code)

# showing the dataframe
head(pop_df_long)
```

```
##                  Country.Name Country.Code                     Region
## 1                       Aruba          ABW Latin America & Caribbean
## 2 Africa Eastern and Southern          AFE
## 3                 Afghanistan          AFG                 South Asia
## 4  Africa Western and Central          AFW
## 5                      Angola          AGO         Sub-Saharan Africa
## 6                     Albania          ALB      Europe & Central Asia
##            IncomeGroup Year Population
## 1          High income 1960      54608
## 2                      1960  130692579
## 3           Low income 1960    8622466
## 4                      1960   97256290
## 5 Lower middle income 1960    5357195
## 6 Upper middle income 1960    1608800
```

Proprocessing the Life Expectancy dataframe

We will do the same to the Life Expectancy dataframe

```r
# gathering columns into key-value pairs
life_ex_df_long <- gather(life_ex, key = "Year", value = "Life_Expectancy", -Country.Name, -Country.Code

# converting 'Year' to numeric (if needed)
life_ex_df_long$Year <- as.numeric(gsub("X", "", life_ex_df_long$Year))

# removing unnecessary columns from the life expectancy data
life_ex_df_long <- life_ex_df_long %>%
  select(-Indicator.Name, -Indicator.Code)

# showing the dataframe
head(life_ex_df_long)
```

```
##                  Country.Name Country.Code Year Life_Expectancy
## 1                       Aruba          ABW 1960        64.15200
## 2 Africa Eastern and Southern          AFE 1960        44.08555
## 3                 Afghanistan          AFG 1960        32.53500
## 4  Africa Western and Central          AFW 1960        37.84515
## 5                      Angola          AGO 1960        38.21100
## 6                     Albania          ALB 1960        54.43900
```

Merging the two dataframes

Then using the country name, Country code and Year, we can merge the two dataframes into single dataframe

```r
# merging the the datasets by 'Country.Name', 'Country.Code', and 'Year'
merged_data <- merge(pop_df_long, life_ex_df_long, by = c('Country.Name', 'Country.Code', 'Year'))

# show few instances
head(merged_data)
```

```
##   Country.Name Country.Code Year     Region IncomeGroup Population
## 1  Afghanistan          AFG 1960 South Asia  Low income    8622466
## 2  Afghanistan          AFG 1961 South Asia  Low income    8790140
## 3  Afghanistan          AFG 1962 South Asia  Low income    8969047
## 4  Afghanistan          AFG 1963 South Asia  Low income    9157465
## 5  Afghanistan          AFG 1964 South Asia  Low income    9355514
## 6  Afghanistan          AFG 1965 South Asia  Low income    9565147
##   Life_Expectancy
## 1          32.535
## 2          33.068
## 3          33.547
## 4          34.016
## 5          34.494
## 6          34.953
```

Checking and dropping the missing values

Then we need to check the missing values. It is important to note that at times, R does not read empty strings as Nulls, so the best thing is to replace empty strings with Nulls, so that if we omit all the nulls, we can omit all instances where the data point is missing

```r
# convert empty strings to NA
merged_data[merged_data == ""] <- NA

# missing values per column
missing_values <- colSums(is.na(merged_data))

# showing output
cat('Missing values Before dropping\n')
```

```
## Missing values Before dropping
```

```r
missing_values
```

```
##    Country.Name    Country.Code            Year          Region     IncomeGroup
##               0               0               0            3087            3150
##      Population Life_Expectancy
##              93             892
```

```r
# dropping the missing values
merged_data <- na.omit(merged_data)
missing_values <- colSums(is.na(merged_data))
cat('\n\nChecking the missing values After dropping\n')
```

```
##
##
## Checking the missing values After dropping
```

```r
missing_values
```

```
##    Country.Name    Country.Code            Year          Region     IncomeGroup
##               0               0               0               0               0
##      Population Life_Expectancy
##               0               0
```

```r
# shape of the data after dropping nulls
cat('\n\nChecking the shape After dropping nulls\n')
```

```
##
##
## Checking the shape After dropping nulls
```

```r
dim(merged_data)
```

```
## [1] 12828     7
```

We still have 12828 observations and 7 variables

Sorting the Merged Dataframe

```r
# Sort the dataframe by 'Country.Name', 'Country.Code', and 'Year'
merged_data <- merged_data[order(merged_data$Country.Name, merged_data$Country.Code, merged_data$Year),
```

Exploratory Data Analysis (EDA)

From This Section, we will select 3 countries, one with the largest population, another with the population closest to the median and the one with the least population. To ensure that all the 3 countries have complete data from 1960, we will first filter the countries that contains the data from all the years.

```r
# getting countries that have data for all years from 1960 to 2021
complete_years_countries <- merged_data %>%
  group_by(Country.Name) %>%
  filter(all(c(1960:2021) %in% Year))

# getting the total population for each country
country_population <- complete_years_countries %>%
  group_by(Country.Name) %>%
  summarise(Total_Population = sum(Population))

# identifying the country with the largest population
largest_population <- country_population %>%
  filter(Total_Population == max(Total_Population))
```

```r
# the country with the population closest to the median
middle_population <- country_population %>%
  arrange(abs(Total_Population - median(Total_Population))) %>%
  slice(1)

# the country with the least population
least_population <- country_population %>%
  filter(Total_Population == min(Total_Population))

# filter the merged_data to include only the selected countries
selected_countries <- merged_data %>%
  filter(Country.Name %in% c(largest_population$Country.Name,
                             middle_population$Country.Name,
                             least_population$Country.Name))
```

The countries includes, Tuvalu (with the least population), Finland (with closest to median population), and China (with the largest population)

Checking the summary statistics

```r
# Summary statistics for numeric variables
summary(selected_countries[, c("Year", "Population", "Life_Expectancy")])
```

```
##       Year        Population        Life_Expectancy
##  Min.   :1960   Min.   :5.404e+03   Min.   :33.27
##  1st Qu.:1975   1st Qu.:1.006e+04   1st Qu.:61.61
##  Median :1990   Median :5.000e+06   Median :68.09
##  Mean   :1990   Mean   :3.681e+08   Mean   :67.37
##  3rd Qu.:2006   3rd Qu.:9.124e+08   3rd Qu.:74.78
##  Max.   :2021   Max.   :1.412e+09   Max.   :81.98
```

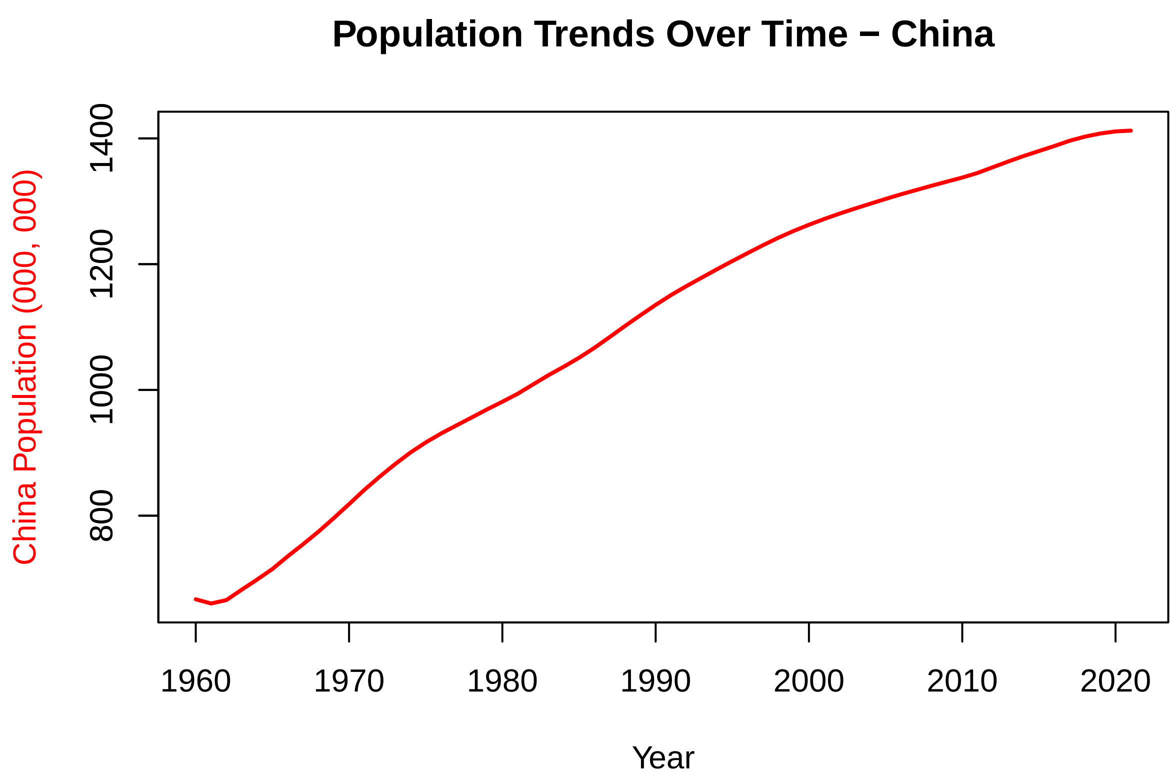
As we can see from the results above:

- The earliest year is 1960.
- 25% of the data falls below year 1975.
- The middle year is 1991.
- 75% of the data falls below year 2006.
- The latest year is 2021.
- The smallest population size is around 5,404.
- 25% of the data falls below the population size of 10,060.
- The middle population size is 5 million.
- The largest population size is 1.412 billion.
- The minimum life expectancy is 33.27 years.
- The middle life expectancy is 68.09 years.
- The average life expectancy is 67.37 years.
- The maximum life expectancy is 81.98 years.

Population Trends Over Time

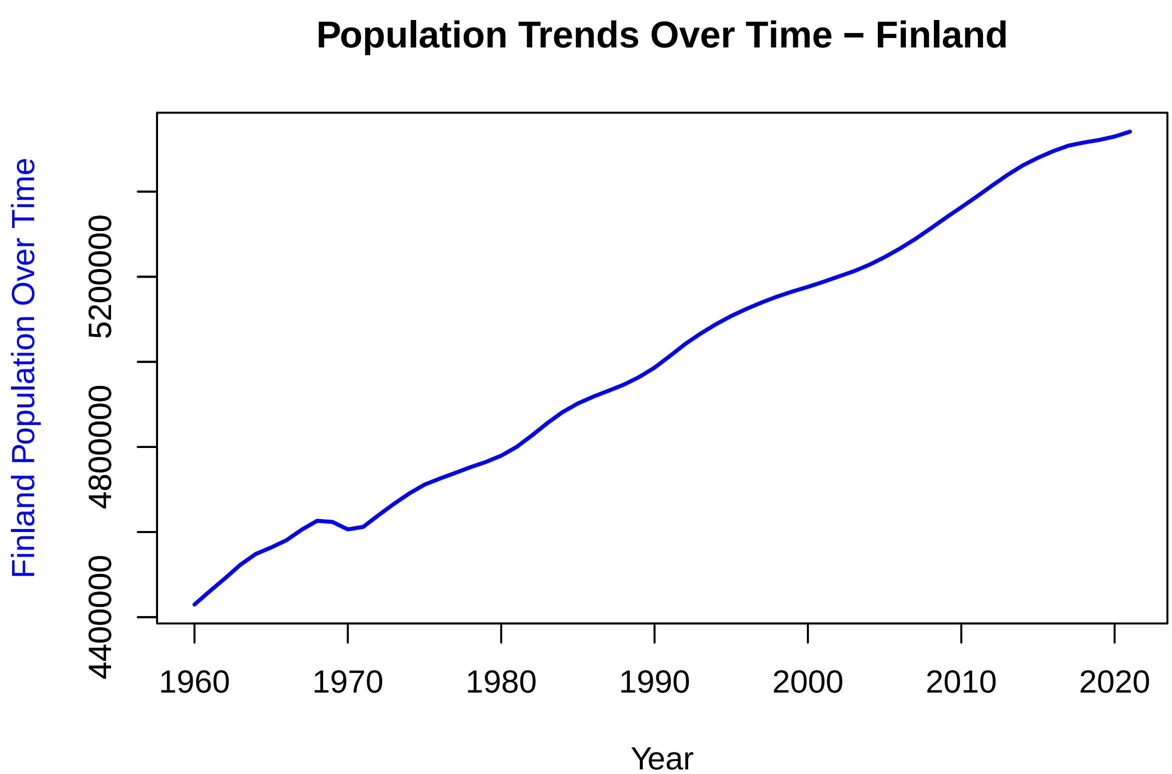
We can then check the trends in the 3 countries we selected earlier. Doing it individually makes sense so that we can have one values for each year. We cannot do it for the combined data because each year will be having multiple values which will alter our observations

```r
# filter the merged_data to include only the selected countries
china_df <- merged_data %>%
  filter(Country.Name %in% c(largest_population$Country.Name))

# Create a new plot for China with its own scale
china_plot <- plot(china_df$Year,
                   china_df$Population / 1e6,  # Convert to 'M' (millions)
                   type = "l",
                   col = "red",
                   lwd = 2,
                   xlab = "Year",
                   ylab = "",
                   main = "Population Trends Over Time - China")
mtext("China Population (000, 000)", side = 2, line = 3, col = "red")
axis(2, at = pretty(range(china_df$Population / 1e6), n = 5)) # Add y-axis with 'M' for millions
```



China

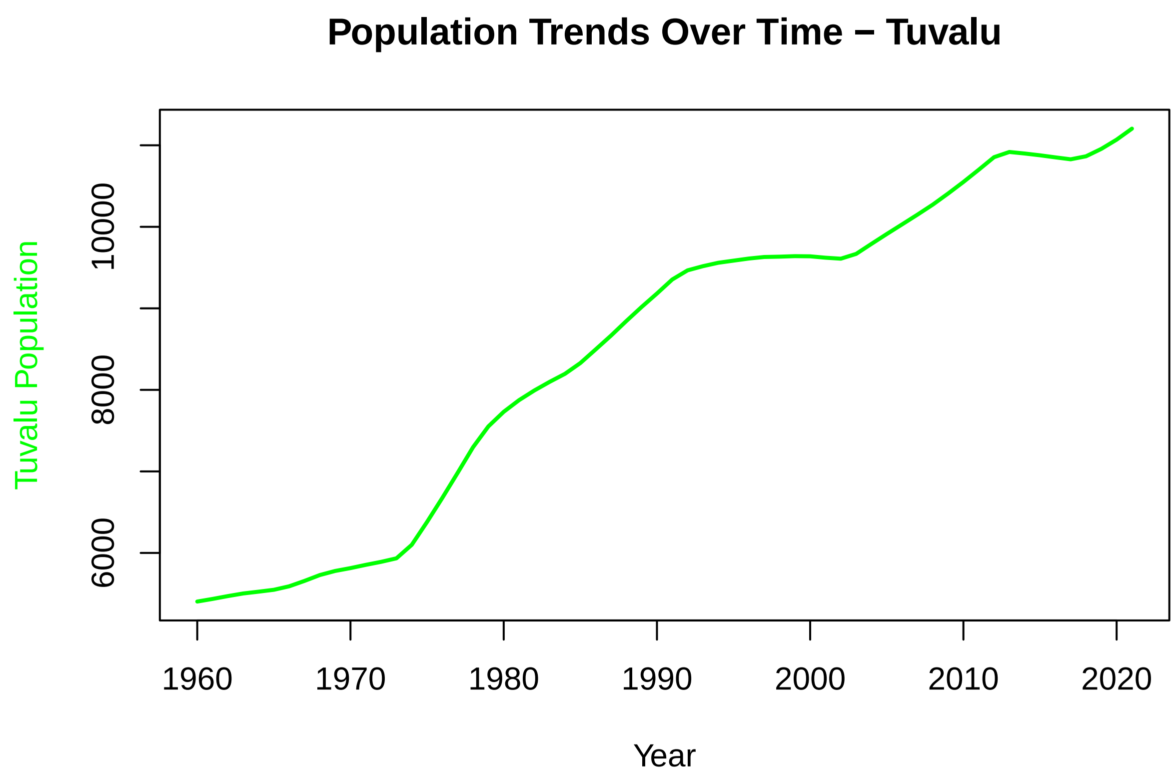
```r
finland_df <- merged_data %>%
  filter(Country.Name %in% c(middle_population$Country.Name))

# Plot Finland on the left y-axis
plot(finland_df$Year,
     finland_df$Population,
     type = "l",
     col = "blue",
     lwd = 2,
     xlab = "Year",
     ylab = "",
     main ="Population Trends Over Time - Finland",
     axes = TRUE
)
mtext("Finland Population Over Time", side = 2, line = 3, col = "blue")
axis(2, at = pretty(range(finland_df$Population), n = 5)) # Customize y-axis labels
```



Finland

```r
tuvalu_df <- merged_data %>%
  filter(Country.Name %in% c(least_population$Country.Name))

plot(tuvalu_df$Year,
```

```r
     tuvalu_df$Population,
     type = "l",
     col = "green",
     lwd = 2,
     xlab = "Year",
     ylab = "",
     axes = TRUE,
     main ="Population Trends Over Time - Tuvalu",
)
mtext("Tuvalu Population", side = 2, line = 3, col = "green")
```



Tuvalu

As we can see, for the 3 countries, the population is increasing significantly over time.

Time Series Creation from the Different Datafarmes

We will now create time series object from the different datasets like china, Finland and Tuvalu.

```r
#### Transforming the data into a time series

# china population and life expectancy time series
china_population_ts <- ts(china_df$Population, start = c(1960), end = c(2021), frequency = 1)
china_life_expectancy_ts <- ts(china_df$Life_Expectancy, start = c(1960), end = c(2021), frequency = 1)

# finland population and life expectancy time series
finland_population_ts <- ts(finland_df$Population, start = c(1960), end = c(2021), frequency = 1)
```

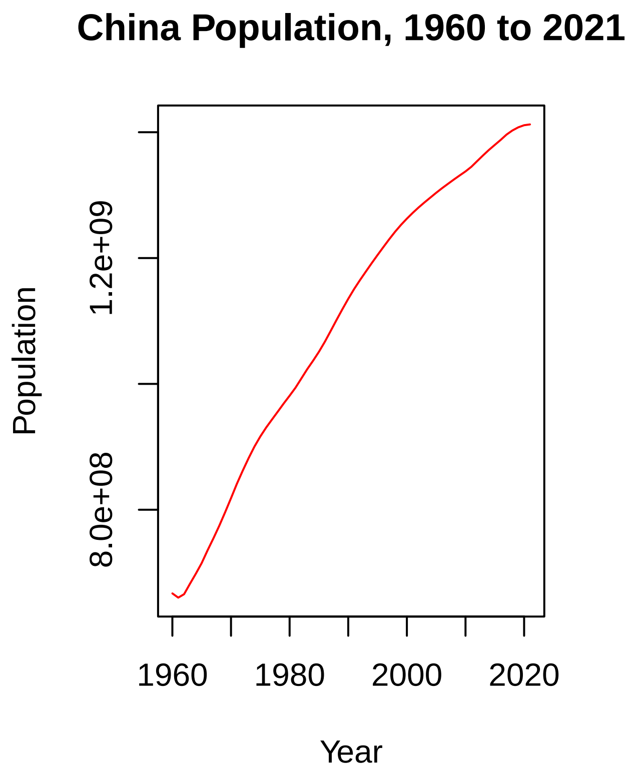
```
finland_life_expectancy_ts <- ts(finland_df$Life_Expectancy, start = c(1960), end = c(2021), frequency =

# tuvalu population and life expectancy time series
tuvalu_population_ts <- ts(tuvalu_df$Population, start = c(1960), end = c(2021), frequency = 1)
tuvalu_life_expectancy_ts <- ts(tuvalu_df$Life_Expectancy, start = c(1960), end = c(2021), frequency =
```

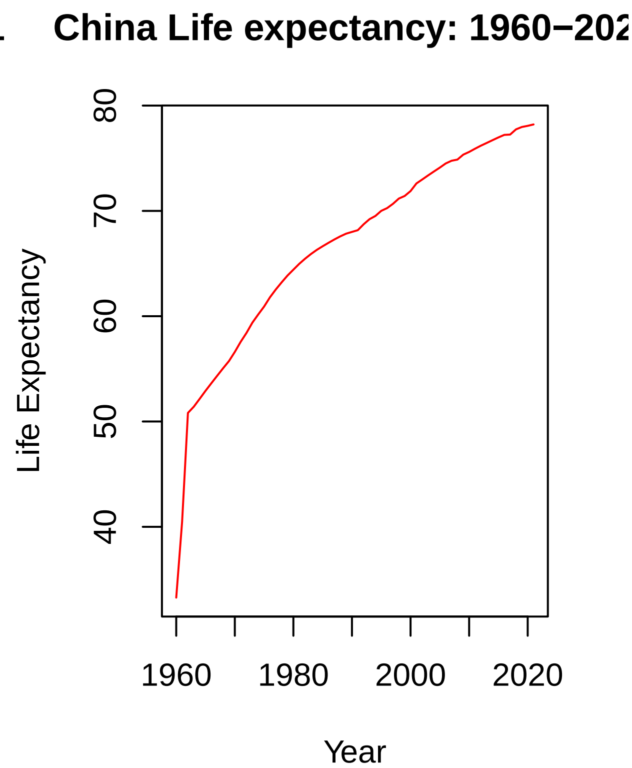
Visualiziing Time Series Trends

Visualize Time Series Data: Plot the time series data for both population and life expectancy to observe trends, seasonality, and any apparent patterns.

China

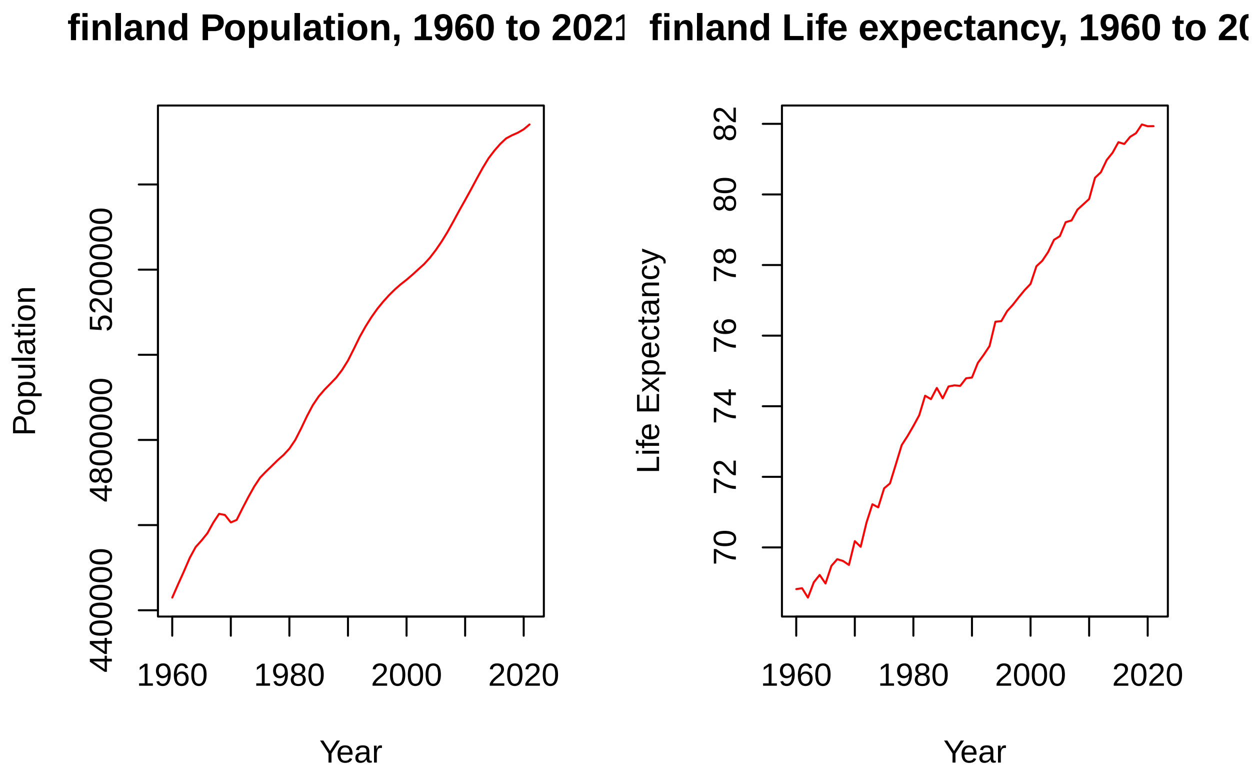
```
# plotting time series
par(mfrow=c(1,2))

plot(china_population_ts,main ="China Population, 1960 to 2021",
     col="red",
     xlab="Year",
     ylab="Population")

plot(china_life_expectancy_ts,
     main="China Life expectancy: 1960-2021",
     col="red",
     xlab="Year",
     ylab="Life Expectancy")
```





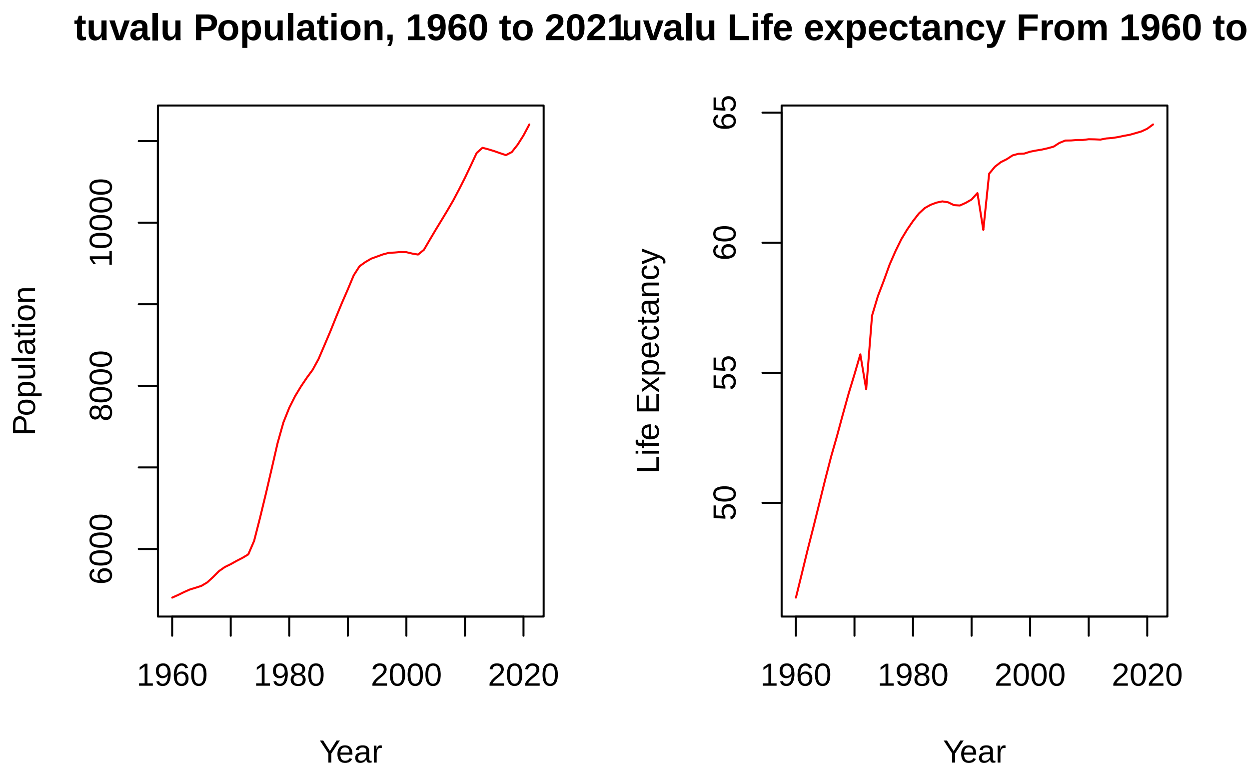
Finland

```
# plotting time series
par(mfrow=c(1,2))

plot(finland_population_ts,main ="finland Population, 1960 to 2021",
     col="red",
     xlab="Year",
     ylab="Population")

plot(finland_life_expectancy_ts,
     main="finland Life expectancy, 1960 to 2021",
     col="red",
     xlab="Year",
     ylab="Life Expectancy")
```



Tuvalu

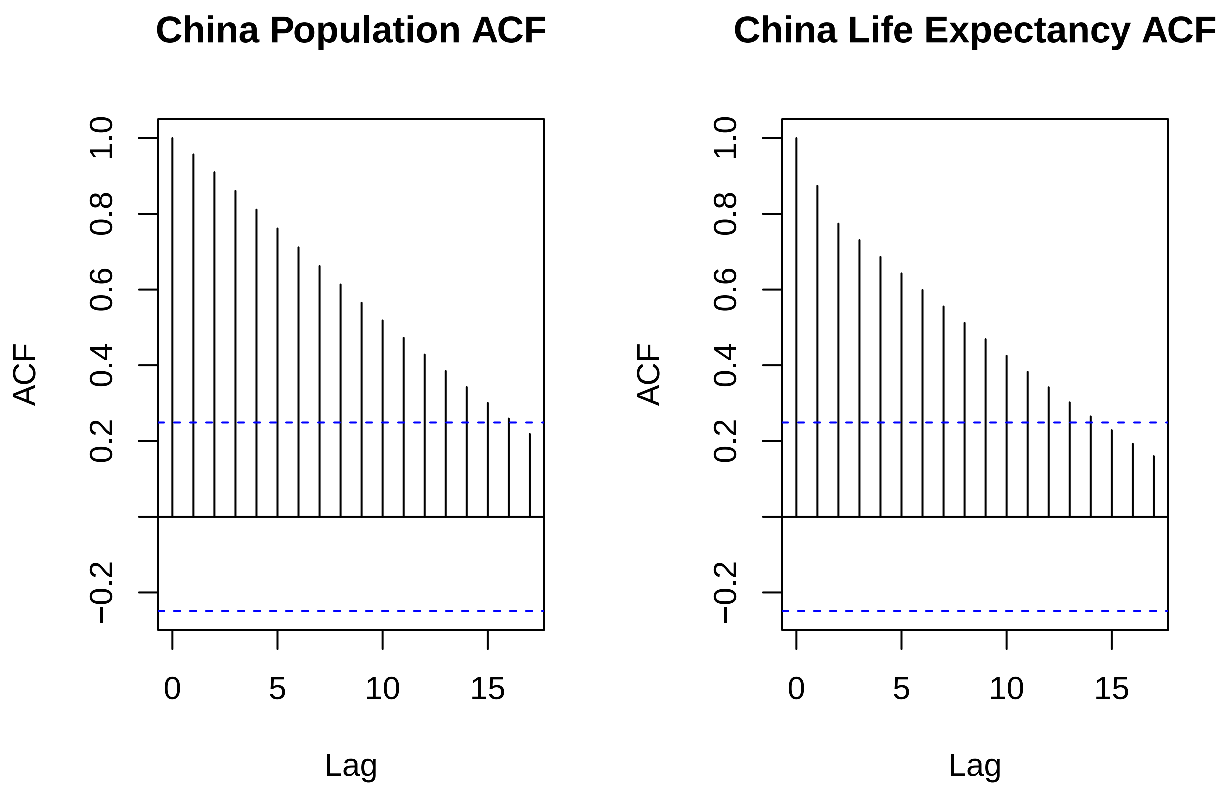
```
# plotting time series
par(mfrow=c(1,2))

plot(tuvalu_population_ts,main ="tuvalu Population, 1960 to 2021",
     col="red",
     xlab="Year",
     ylab="Population")

plot(tuvalu_life_expectancy_ts,
     main="tuvalu Life expectancy From 1960 to 2021",
     col="red",
     xlab="Year",
     ylab="Life Expectancy")
```



We can already observe strong increasing trend, with strong seasonality from all the 3 countries. But for the three, there is no evidence of any cyclic behaviour.

Visualizing Autocorrelation Function Plots

```
# ACF plots
par(mfrow=c(1,2))
acf(china_population_ts, main = "China Population ACF") # population ACF plot
acf(china_life_expectancy_ts, main = "China Life Expectancy ACF") # life expectancy ACF plot
```

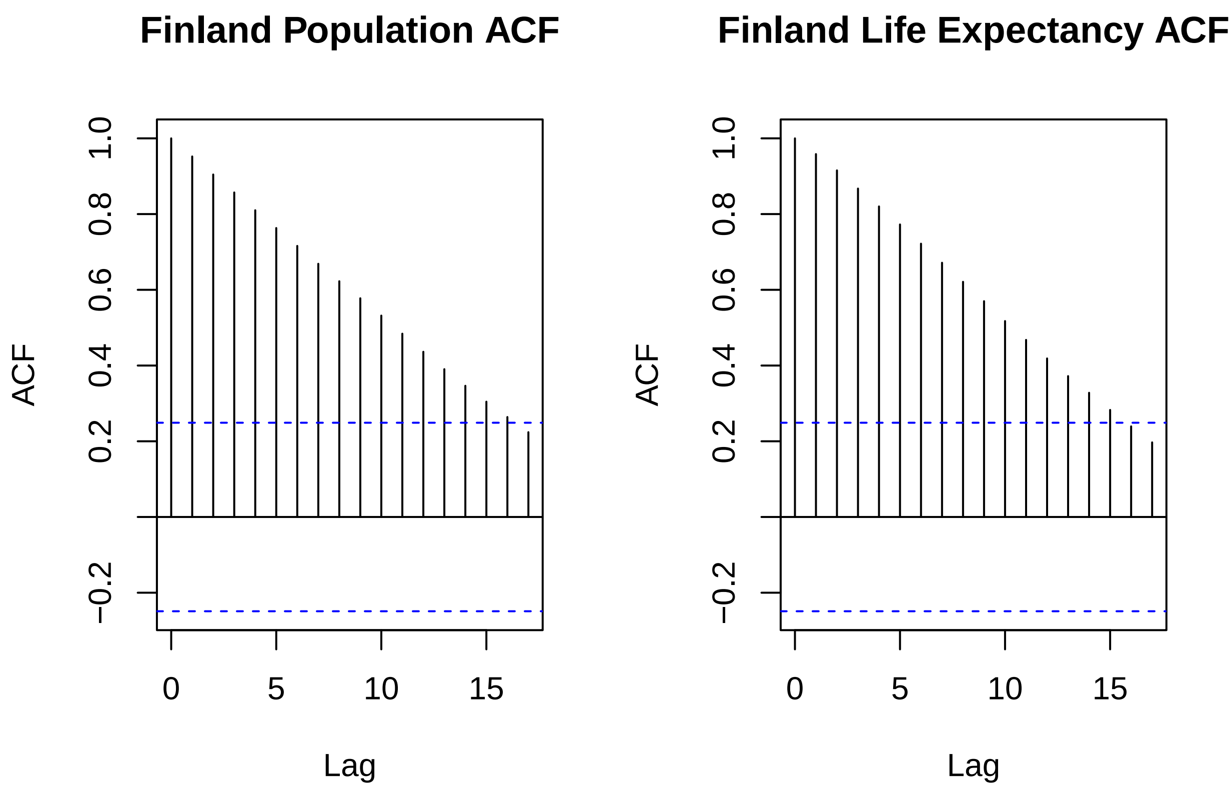


China

In China population, we can see that the ACF shows significant values up to 16 lags above the dotted line in a time series analysis, it suggests that there may be a strong autocorrelation in the data even after considering the initial seasonality.

For Life Expectancy, it shows 14 lags are significant

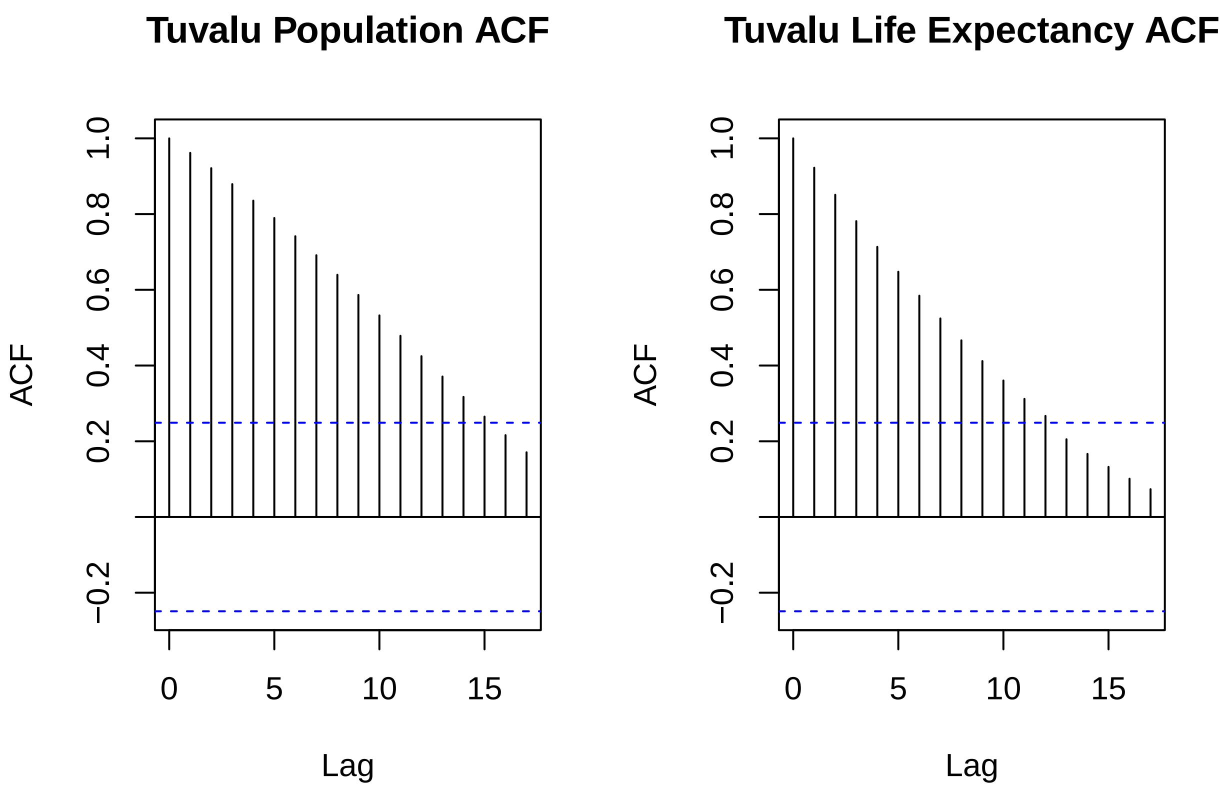
```
# ACF plots
par(mfrow=c(1,2))
acf(finland_population_ts, main = "Finland Population ACF") # population ACF plot
acf(finland_life_expectancy_ts, main = "Finland Life Expectancy ACF") # life expectancy ACF plot
```



Finland

It shows the same for the finland time seris

```
# ACF plots
par(mfrow=c(1,2))
acf(tuvalu_population_ts, main = "Tuvalu Population ACF") # population ACF plot
acf(tuvalu_life_expectancy_ts, main = "Tuvalu Life Expectancy ACF") # life expectancy ACF plot
```

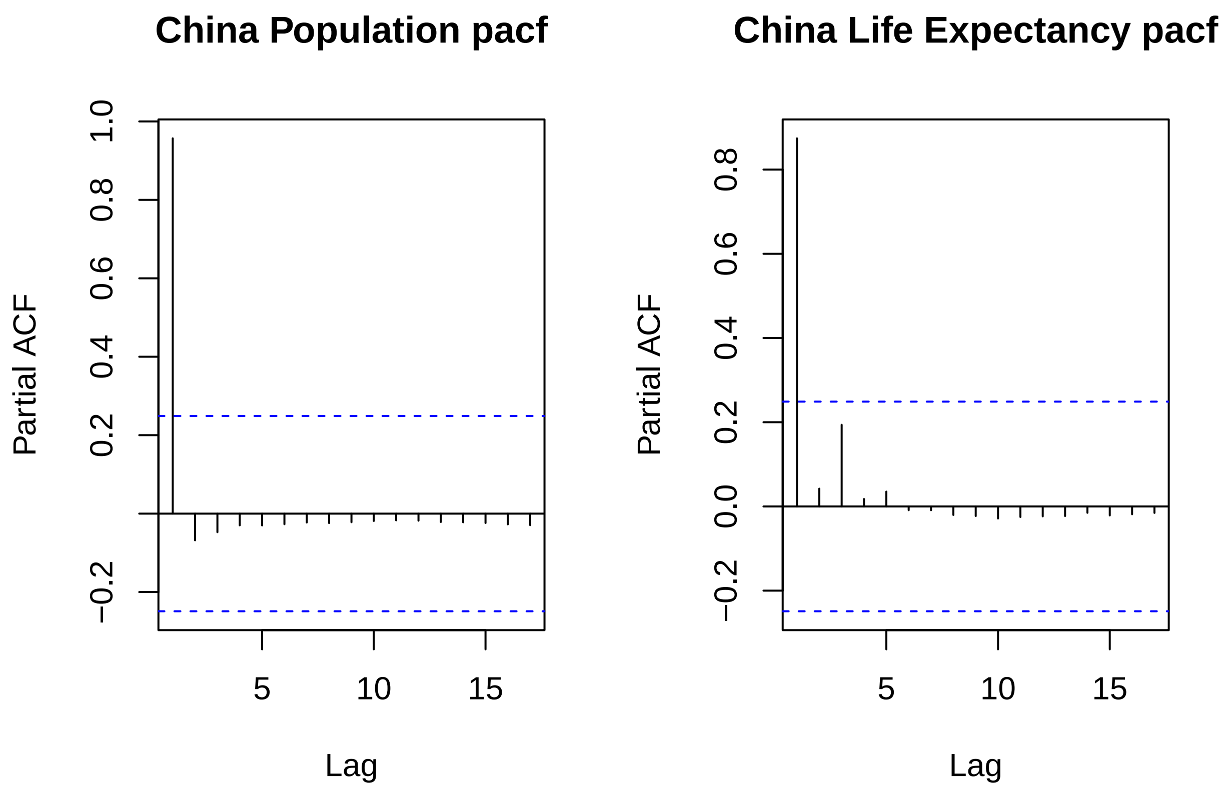


Tuvalu

For Tuvalu, it shows 15 lags above the blue line for population, and 12 for life expectancy.

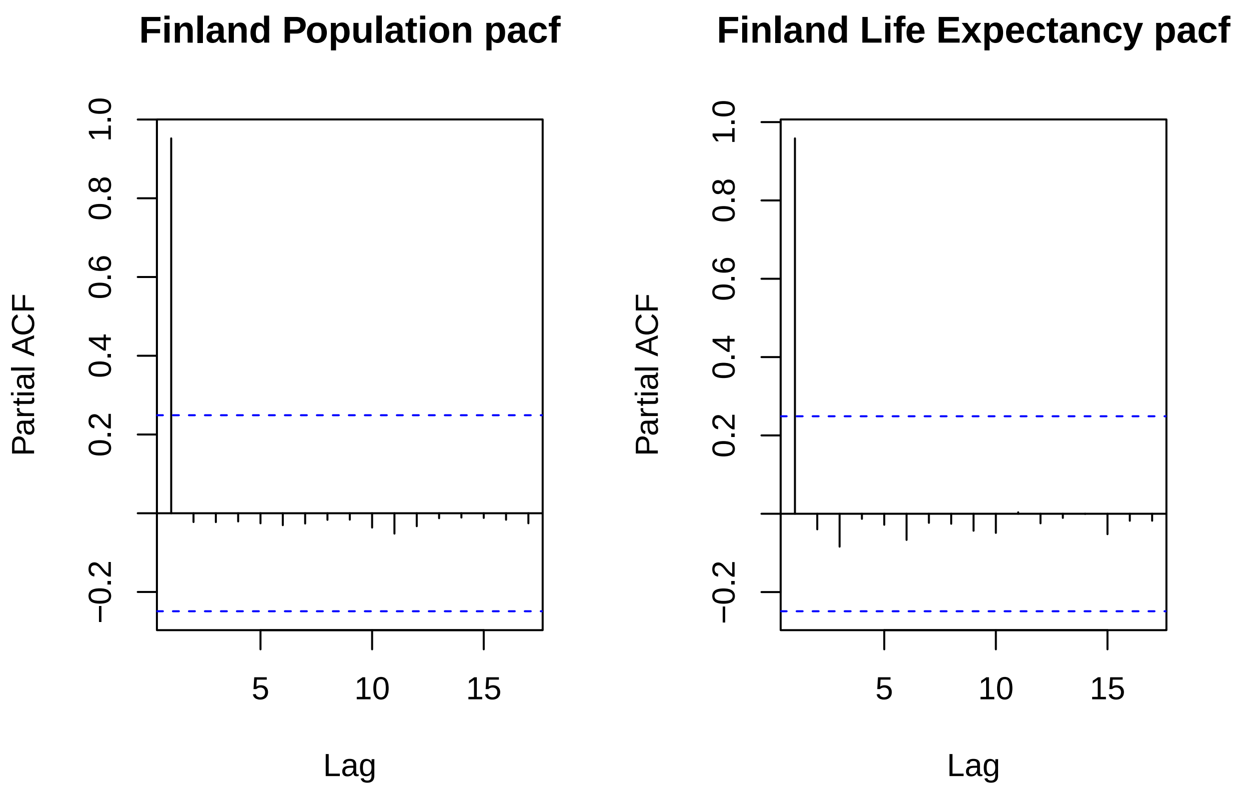
Visualizing Partial Autocorrelation Function (PACF)

```
# PACF plots
par(mfrow=c(1,2))
pacf(china_population_ts, main = "China Population pacf") # population pacf plot
pacf(china_life_expectancy_ts, main = "China Life Expectancy pacf") # life expectancy pacf plot
```



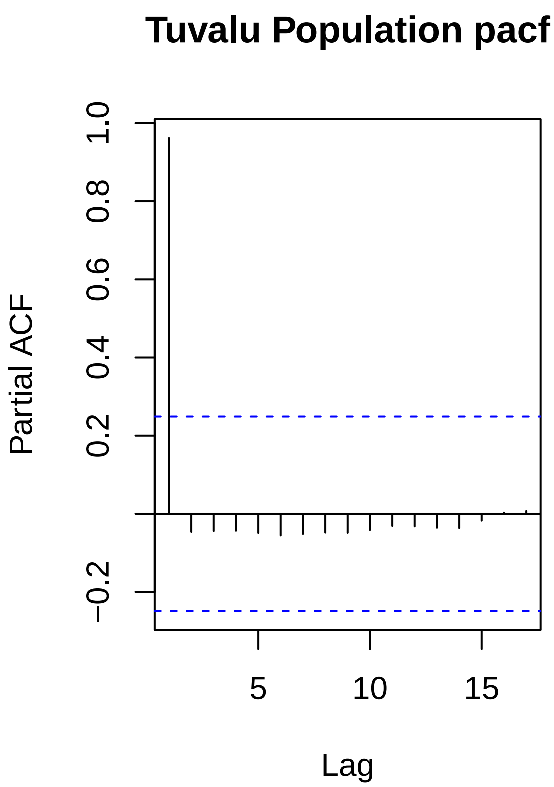
China

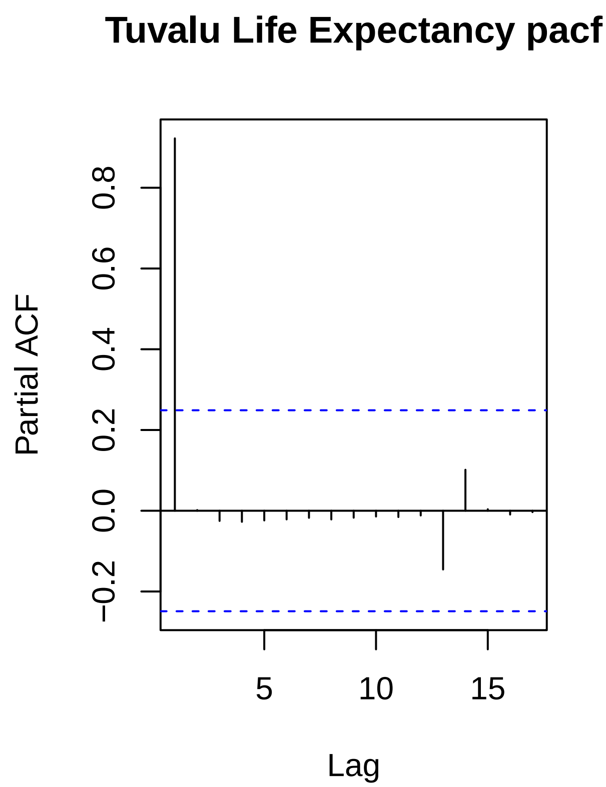
```
# pacf plots
par(mfrow=c(1,2))
pacf(finland_population_ts, main = "Finland Population pacf") # population pacf plot
pacf(finland_life_expectancy_ts, main = "Finland Life Expectancy pacf") # life expectancy pacf plot
```



Finland

```
# pacf plots
par(mfrow=c(1,2))
pacf(tuvalu_population_ts, main = "Tuvalu Population pacf") # population pacf plot
pacf(tuvalu_life_expectancy_ts, main = "Tuvalu Life Expectancy pacf") # life expectancy pacf plot
```





Tuvalu

For all the 3 countries, we can see that only the 1st lag is significant.

ARIMA Models

```
par(mfrow=c(1,2))

# create ARIMA model for China Population
china_pop_model <- auto.arima(china_population_ts)

#Summary of ARIMA model for China Population
summary(china_pop_model)
```

China Arima Model and Forecast

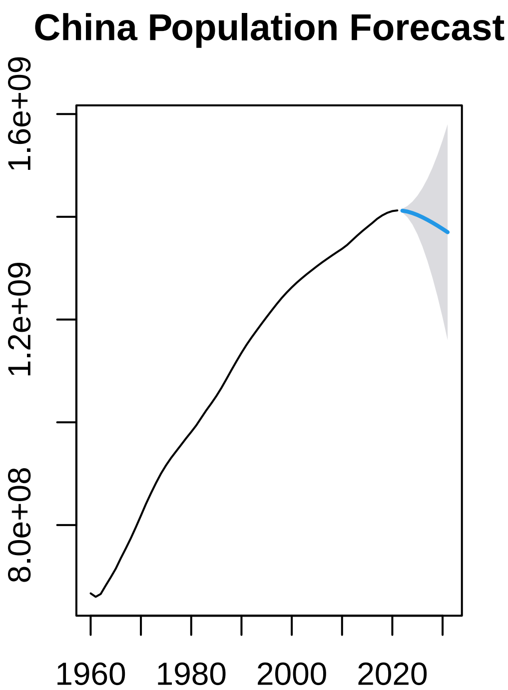
```
## Series: china_population_ts
## ARIMA(1,2,0)
##
## Coefficients:
##          ar1
##       0.8053
## s.e.  0.1204
##
## sigma^2 = 3.532e+12:  log likelihood = -951.81
```

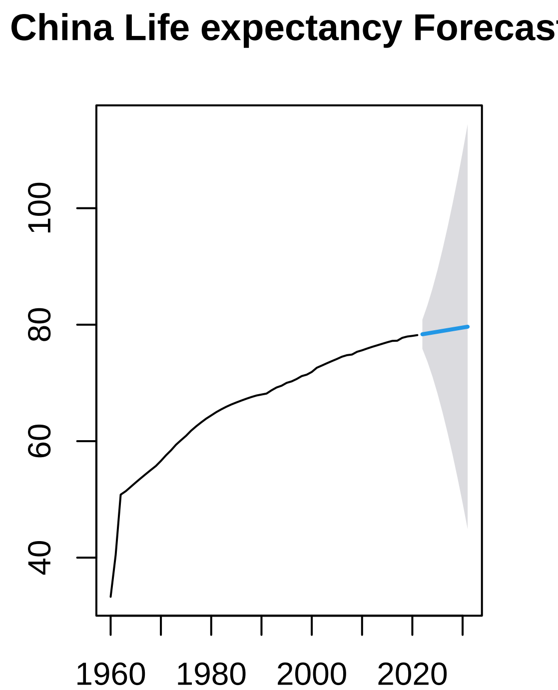
```
## AIC=1907.61   AICc=1907.82   BIC=1911.8
##
## Training set error measures:
##                      ME    RMSE      MAE          MPE      MAPE       MASE
## Training set -91891.23 1833234 983422.3 -0.006984603 0.1136123 0.07906054
##                     ACF1
## Training set -0.1681993
```

```r
china_pop_forecast <- forecast(china_pop_model, level=c(95), h=10) # h = 10, forecast for 10 years
plot(china_pop_forecast, main='China Population Forecast') # plotting the forecast

# create ARIMA model for China Life expectancy
china_lexp_model<-auto.arima(china_life_expectancy_ts)

#Summary of ARIMA model for China Life Expectancy
summary(china_lexp_model)
```

```
## Series: china_life_expectancy_ts
## ARIMA(0,2,1)
##
## Coefficients:
##           ma1
##       -0.3376
## s.e.   0.1282
##
## sigma^2 = 1.616:  log likelihood = -99.09
## AIC=202.18   AICc=202.39   BIC=206.37
##
## Training set error measures:
##                       ME     RMSE       MAE        MPE      MAPE      MASE
## Training set -0.1754817 1.240077 0.3521292 -0.3298009 0.6364184 0.4780105
##                      ACF1
## Training set -0.01267264
```

```r
china_lexp_forecast <- forecast(china_lexp_model, level=c(95), h=10) # h = 10, forecast for 10 years
plot(china_lexp_forecast, main='China Life expectancy Forecast') # plotting the forecast
```





We can see that China's population in the next 10 years is expected to drop, while Life expectancy rate is expected to rise.

```
par(mfrow=c(1,2))

# create ARIMA model for Finland Population
finland_pop_model<-auto.arima(finland_population_ts)

#Summary of ARIMA model for Finland Population
summary(finland_pop_model)
```

finland Arima Model and Forecast

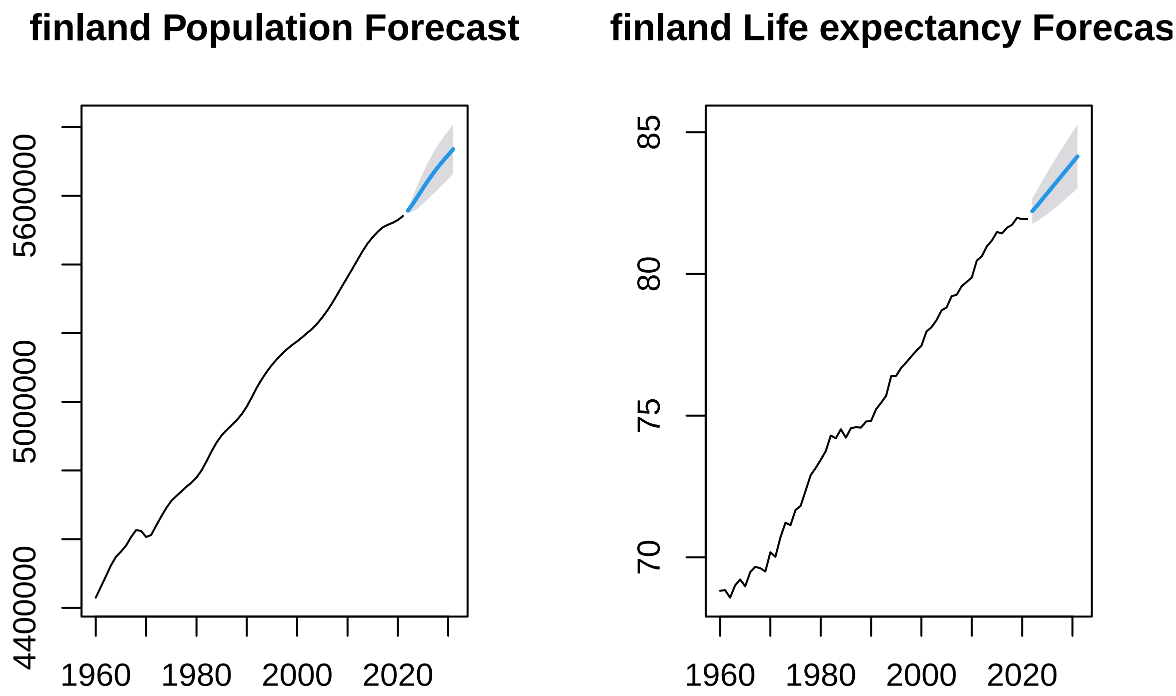
```
## Series: finland_population_ts
## ARIMA(4,1,0) with drift
##
## Coefficients:
##          ar1      ar2     ar3      ar4     drift
##       1.6234  -1.6285  1.0941  -0.4711  18428.43
## s.e.  0.1155   0.1925  0.1911   0.1164   1272.76
##
## sigma^2 = 15687418:  log likelihood = -590.24
## AIC=1192.48   AICc=1194.03   BIC=1205.14
```

```
##
## Training set error measures:
##                     ME    RMSE      MAE         MPE       MAPE      MASE
## Training set 52.71012 3764.21 2646.972 0.001586491 0.05475399 0.1401965
##                     ACF1
## Training set 0.07554526
```

```
finland_pop_forecast <- forecast(finland_pop_model, level=c(95), h=10) # h = 10, forecast for 10 years
plot(finland_pop_forecast, main='finland Population Forecast') # plotting the forecast

# create ARIMA model for finland Life expectancy
finland_lexp_model<-auto.arima(finland_life_expectancy_ts)

#Summary of ARIMA model for Finland Life expectancy
summary(finland_lexp_model)
```

```
## Series: finland_life_expectancy_ts
## ARIMA(1,1,0) with drift
##
## Coefficients:
##           ar1   drift
##       -0.2962  0.2165
## s.e.   0.1226  0.0223
##
## sigma^2 = 0.05255:  log likelihood = 4.31
## AIC=-2.63   AICc=-2.21   BIC=3.71
##
## Training set error measures:
##                        ME      RMSE       MAE           MPE      MAPE      MASE
## Training set 0.0003287302 0.2236262 0.1689953 -0.0002329704 0.2287908 0.6450125
##                     ACF1
## Training set 0.01207243
```

```
finland_lexp_forecast <- forecast(finland_lexp_model, level=c(95), h=10) # h = 10, forecast for 10 year
plot(finland_lexp_forecast, main='finland Life expectancy Forecast') # plotting the forecast
```



In Finland, both the population and life expectancy rate are expected rise in the next 10 years.

```r
par(mfrow=c(1,2))

# create ARIMA model for Tuvalu Population
tuvalu_pop_model<-auto.arima(tuvalu_population_ts)

#Summary of ARIMA model for Tuvalu Population
summary(tuvalu_pop_model)
```

tuvalu Arima Model and Forecast

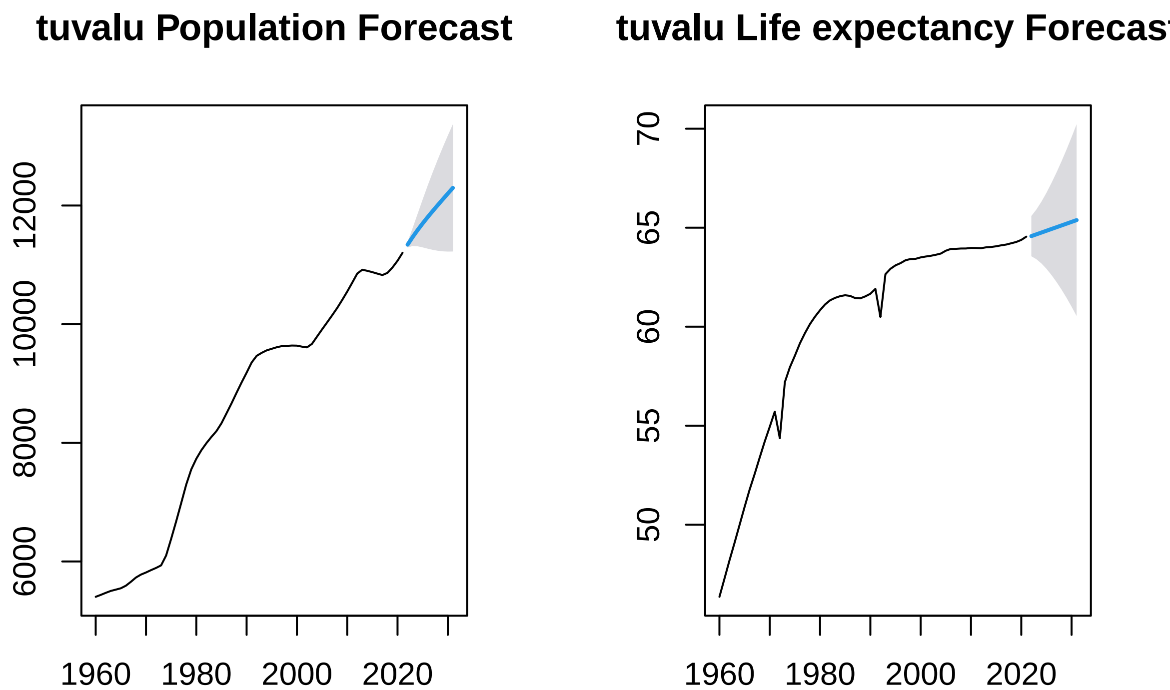
```
## Series: tuvalu_population_ts
## ARIMA(2,1,1) with drift
##
## Coefficients:
##          ar1      ar2     ma1    drift
##       1.0529  -0.2535  0.8897  93.8622
## s.e.  0.1452   0.1444  0.1165  27.4521
##
## sigma^2 = 611.2:  log likelihood = -282.36
## AIC=574.71   AICc=575.8   BIC=585.27
##
```

```
## Training set error measures:
##                     ME     RMSE      MAE       MPE      MAPE      MASE
## Training set 0.6538791 23.70555 14.70139 0.0138038 0.1799776 0.1484252
##                      ACF1
## Training set -0.006693502
```

```r
tuvalu_pop_forecast <- forecast(tuvalu_pop_model, level=c(95), h=10) # h = 10, forecast for 10 years
plot(tuvalu_pop_forecast, main='tuvalu Population Forecast') # plotting the forecast

# create ARIMA model for tuvalu Life expectancy
tuvalu_lexp_model<-auto.arima(tuvalu_life_expectancy_ts)

#Summary of ARIMA model for Tuvalu Life expectancy
summary(tuvalu_lexp_model)
```

```
## Series: tuvalu_life_expectancy_ts
## ARIMA(0,2,2)
##
## Coefficients:
##           ma1     ma2
##       -1.2762  0.4643
## s.e.   0.1111  0.1132
##
## sigma^2 = 0.2685:  log likelihood = -45.62
## AIC=97.23   AICc=97.66   BIC=103.52
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE     MASE
## Training set -0.07105173 0.5011502 0.2781292 -0.1237143 0.464748 0.705149
##                     ACF1
## Training set -0.05012284
```

```r
tuvalu_lexp_forecast <- forecast(tuvalu_lexp_model, level=c(95), h=10) # h = 10, forecast for 10 years
plot(tuvalu_lexp_forecast, main='tuvalu Life expectancy Forecast') # plotting the forecast
```



Even for Tuvalu, both the population and life expectancy rates are expected to rise.

Random Forest and Linear Regression Model Development

Train-Test Split: Split the time series data into training and testing sets.

```
# Divide the data into a training set and a testing set
set.seed(123)  # for reproducibility
trainIndex <- createDataPartition(selected_countries$Life_Expectancy, p = .8, list = FALSE)
train <- selected_countries[trainIndex, ]
test <- selected_countries[-trainIndex, ]
```

Training Random Forest Model

```
# Create random forest for regression
RF_model <- randomForest (Life_Expectancy~ Population + IncomeGroup + Region,
                          data = train,
                          mtry = 3,
                         importance = TRUE,
                         na.action = na.omit)

RF_model
```

```
##
## Call:
##  randomForest(formula = Life_Expectancy ~ Population + IncomeGroup +      Region, data = train, mtry
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.3165332
##                     % Var explained: 99.57
```

```r
#Summary of RF Model
summary(RF_model)
```

```
##                 Length Class  Mode
## call              6    -none- call
## type              1    -none- character
## predicted       150    -none- numeric
## mse             500    -none- numeric
## rsq             500    -none- numeric
## oob.times       150    -none- numeric
## importance        6    -none- numeric
## importanceSD      3    -none- numeric
## localImportance   0    -none- NULL
## proximity         0    -none- NULL
## ntree             1    -none- numeric
## mtry              1    -none- numeric
## forest           11    -none- list
## coefs             0    -none- NULL
## y               150    -none- numeric
## test              0    -none- NULL
## inbag             0    -none- NULL
## terms             3    terms  call
```

Random Forest Evaluation

```r
# Making predictions on the test data
RF_preds <- predict(RF_model, newdata = test)
cat("Making predictions on test data: \n",RF_preds,"\n")
```

```
## Making predictions on test data:
##  51.94881 51.94881 52.44559 62.15865 66.36517 67.93416 71.42097 72.7921 73.63193 74.33321 74.86693 7
```

```r
# Calculate performance metrics
RF_mse <- mean((RF_preds - test$Life_Expectancy)^2)
RF_rmse <- sqrt(RF_mse)
cat("Mean Square Error:",RF_mse,"\n")
```

```
## Mean Square Error: 13.65553
```

```r
cat("Root Mean Square Error: ",RF_rmse, "\n")
```

```
## Root Mean Square Error:  3.695339
```

```r
RF_mae <- mean(abs(RF_preds - test$Life_Expectancy))
cat("Mean Absolute Error: ",RF_mae, "\n")
```

```
## Mean Absolute Error:  1.142909
```

```r
RF_r2 <- cor(RF_preds, test$Life_Expectancy)^2
cat("R^2 Value: ",RF_r2, "\n")
```

```
## R^2 Value:  0.9098648
```

```r
RF_adjR2 <- 1 - ((1 - RF_r2) * (nrow(test) - 1) / (nrow(test) - ncol(train)))
cat("Adj R^2 Value: ",RF_adjR2, "\n")
```

```
## Adj R^2 Value:  0.8912161
```

Linear Regression Model (Base Model)

```r
# Linear Regression
LR_model <- lm(Life_Expectancy~ Population + IncomeGroup + Region, data = train)

# model
summary(LR_model)
```

```
##
## Call:
## lm(formula = Life_Expectancy ~ Population + IncomeGroup + Region,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.899  -3.854   1.145   4.607   7.094
##
## Coefficients: (1 not defined because of singularities)
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     7.519e+01  7.537e-01  99.754  < 2e-16 ***
## Population                      8.571e-09  9.271e-10   9.246  2.5e-16 ***
## IncomeGroupUpper middle income -1.617e+01  1.053e+00 -15.358  < 2e-16 ***
## RegionEurope & Central Asia            NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.329 on 147 degrees of freedom
## Multiple R-squared:  0.6205, Adjusted R-squared:  0.6153
## F-statistic: 120.2 on 2 and 147 DF,  p-value: < 2.2e-16
```

Linear Regression Model with log(Population)

```
# Linear Regression
LR_model_1 <- lm(Life_Expectancy~ log(Population) + IncomeGroup + Region, data = train)

# model
summary(LR_model_1)
```

```
##
## Call:
## lm(formula = Life_Expectancy ~ log(Population) + IncomeGroup +
##     Region, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.349  -4.032   1.280   3.815  10.567
##
## Coefficients: (1 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     65.451      1.771  36.964  < 2e-16 ***
## log(Population)                  0.634      0.101   6.276 3.68e-09 ***
## IncomeGroupUpper middle income -11.165      1.032 -10.816  < 2e-16 ***
## RegionEurope & Central Asia         NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.952 on 147 degrees of freedom
## Multiple R-squared:  0.5267, Adjusted R-squared:  0.5202
## F-statistic: 81.78 on 2 and 147 DF,  p-value: < 2.2e-16
```

Linear Regression Evaluation

```
# Making predictions on the test data
LR_preds <- predict(LR_model, newdata = test)

# Calculate performance metrics
LR_mse <- mean((LR_preds - test$Life_Expectancy)^2)
LR_rmse <- sqrt(LR_mse)

LR_mae <- mean(abs(LR_preds - test$Life_Expectancy))
LR_r2 <- cor(LR_preds, test$Life_Expectancy)^2
LR_adjR2 <- 1 - ((1 - LR_r2) * (nrow(test) - 1) / (nrow(test) - ncol(train)))
```

Model Comparisons

Accuracy Metrics

```
cat('Random Forest\n')
```

```
## Random Forest
```

```r
print(paste("MSE:", RF_mse))
```

```
## [1] "MSE: 13.6555334236835"
```

```r
print(paste("RMSE:", RF_rmse))
```

```
## [1] "RMSE: 3.69533941927984"
```

```r
print(paste("MAE:", RF_mae))
```

```
## [1] "MAE: 1.14290869775423"
```

```r
print(paste("R Squared:", RF_r2))
```

```
## [1] "R Squared: 0.909864765877473"
```

```r
print(paste("Adj. R²:", RF_adjR2))
```

```
## [1] "Adj. R²: 0.891216096748674"
```

```r
cat('\n\nLinear Regression\n')
```

```
##
##
## Linear Regression
```

```r
print(paste("MSE:", LR_mse))
```

```
## [1] "MSE: 65.4692262197067"
```

```r
print(paste("RMSE:", LR_rmse))
```

```
## [1] "RMSE: 8.09130559426022"
```

```r
print(paste("MAE:", LR_mae))
```

```
## [1] "MAE: 5.36487464166902"
```

```r
print(paste("R Squared:", LR_r2))
```

```
## [1] "R Squared: 0.493934967157817"
```

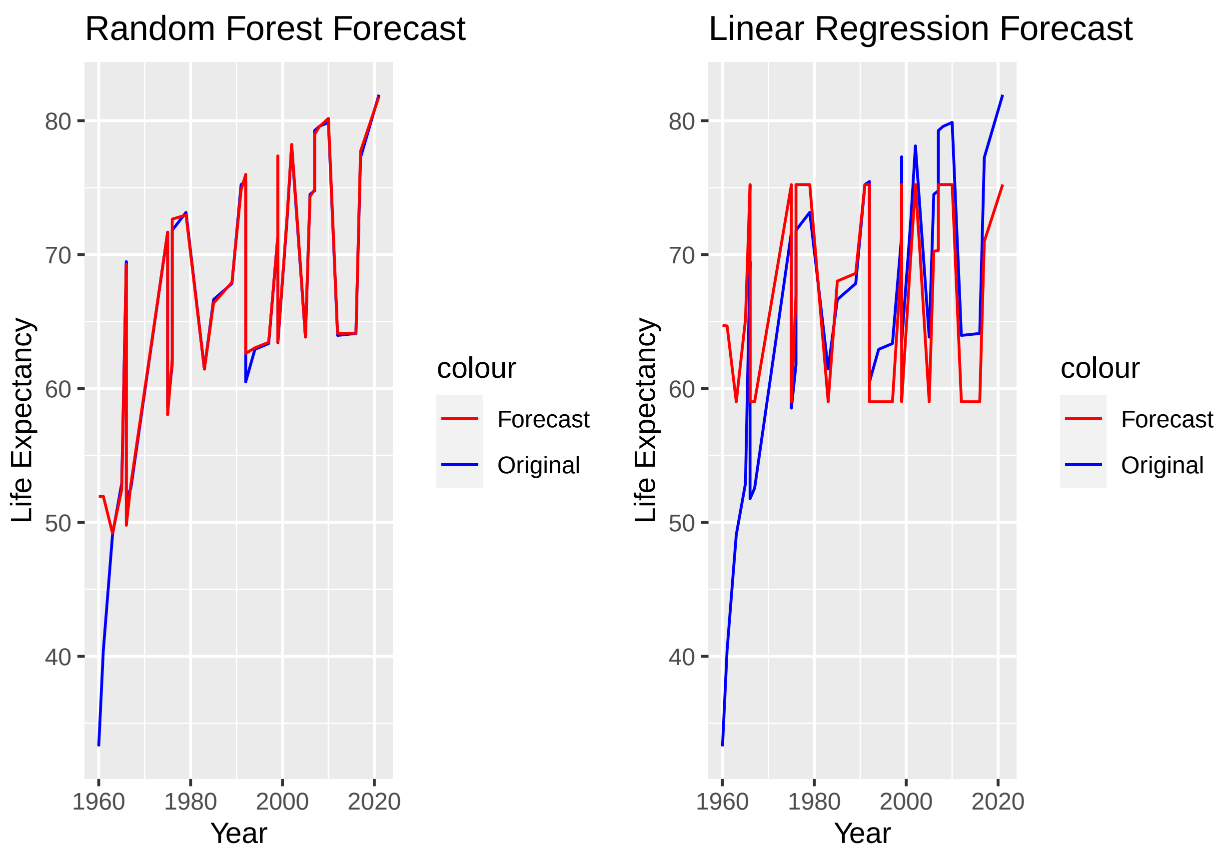
```r
print(paste("Adj. R²:", LR_adjR2))
```

```
## [1] "Adj. R²: 0.389231856914607"
```

From the evaluation metrics above, we can see that Random Forest model outperforms the Linear Regression model in terms of all the evaluated metrics. The Random Forest model has a significantly lower MSE, RMSE, and MAE, indicating better accuracy and precision in predicting the target variable. The r squared value for the Random Forest model is also higher, indicating a better fit to the data compared to the Linear Regression model. In general, the Random Forest still demonstrates a better fit.

Predictions Vs Reference

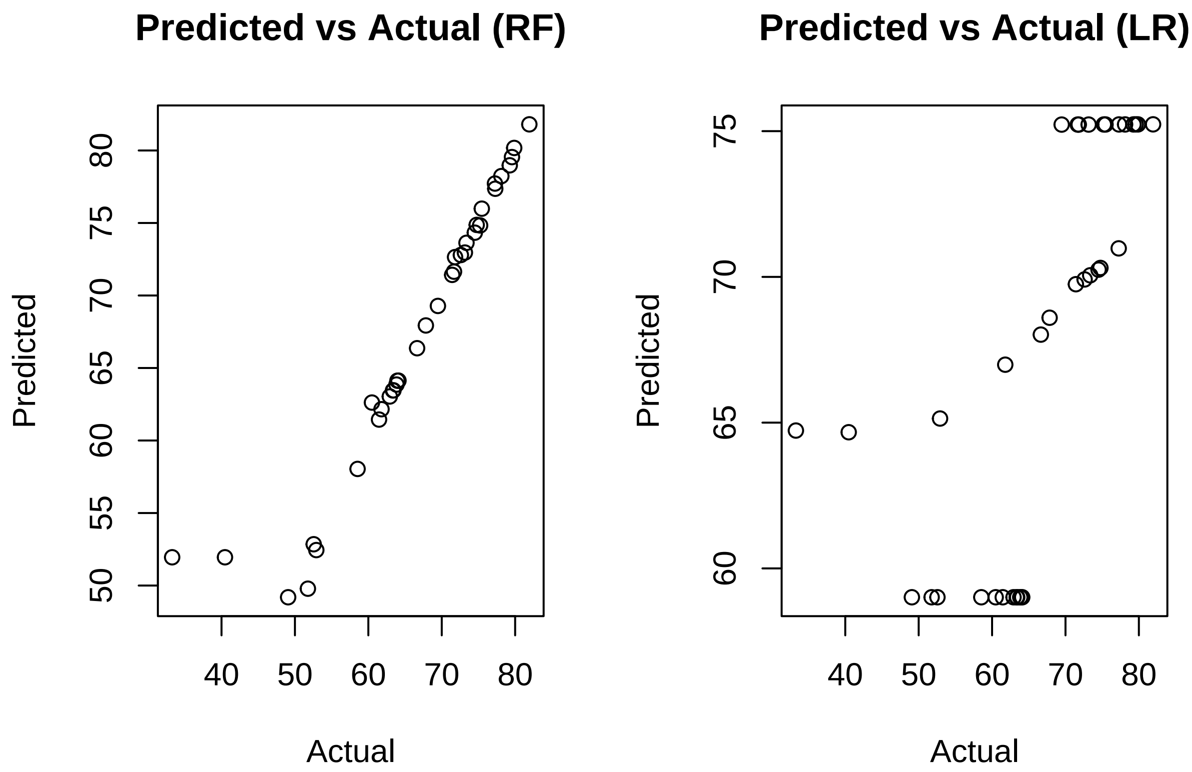
```r
# Plot the original time series and the random forest forecast
plot_rf <- ggplot(test) +
  geom_line(aes(x = Year, y = Life_Expectancy, color = "Original")) +
  geom_line(data = test, aes(x = Year, y = RF_preds, color = "Forecast")) +
  scale_color_manual(values = c("Original" = "blue", "Forecast" = "red")) +
  labs(title = "Random Forest Forecast", y = "Life Expectancy")

# Plot the original time series and the linear regression forecast
plot_lr <- ggplot(test) +
  geom_line(aes(x = Year, y = Life_Expectancy, color = "Original")) +
  geom_line(data = test, aes(x = Year, y = LR_preds, color = "Forecast")) +
  scale_color_manual(values = c("Original" = "blue", "Forecast" = "red")) +
  labs(title = "Linear Regression Forecast", y = "Life Expectancy")

# Arrange the plots side by side using grid.arrange
final_plot <- grid.arrange(plot_rf, plot_lr, ncol = 2)
```



From the forecast life expectancy rate, we can also see that the Random forest model perfprms better than the linear regression model. This can be observed from the blue lines indicating the actual value and the red line indicating the predicted.

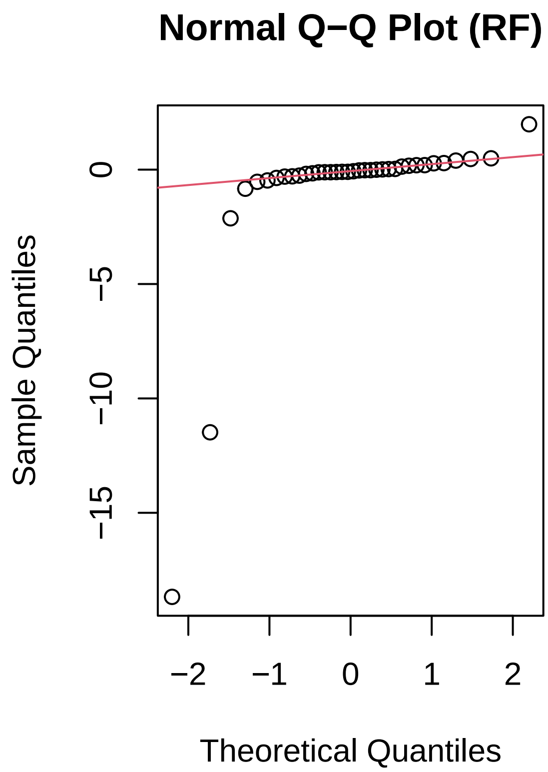
Predictions Vs Actual

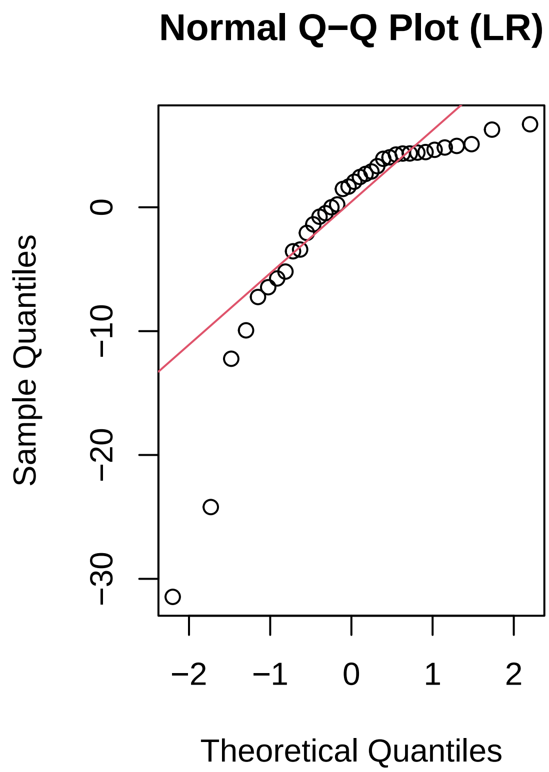
```
par(mfrow=c(1,2))

# Random Forest
plot(test$Life_Expectancy, RF_preds, main = "Predicted vs Actual (RF)", xlab = "Actual", ylab = "Predic

# Linear Regression
plot(test$Life_Expectancy, LR_preds, main = "Predicted vs Actual (LR)", xlab = "Actual", ylab = "Predic
```



Even for this plots, we can see that the Random Forest model has a better fit than the Linear Regression model.

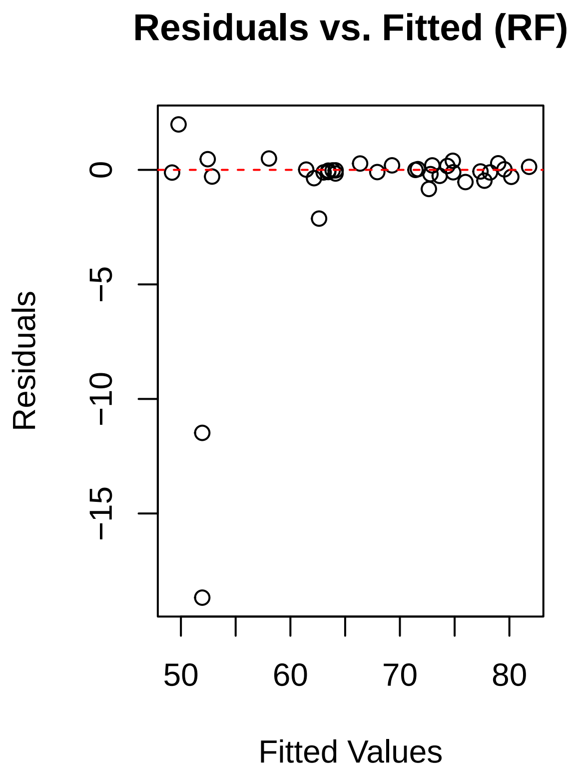
```r
par(mfrow=c(1,2))

# RF QQ Plot
rf_residuals <- test$Life_Expectancy - RF_preds
qqnorm(rf_residuals, main='Normal Q-Q Plot (RF)')
qqline(rf_residuals, col = 2)

# Linear Regression
lm_residuals <- test$Life_Expectancy - LR_preds
qqnorm(lm_residuals, main='Normal Q-Q Plot (LR)')
qqline(lm_residuals, col = 2)
```

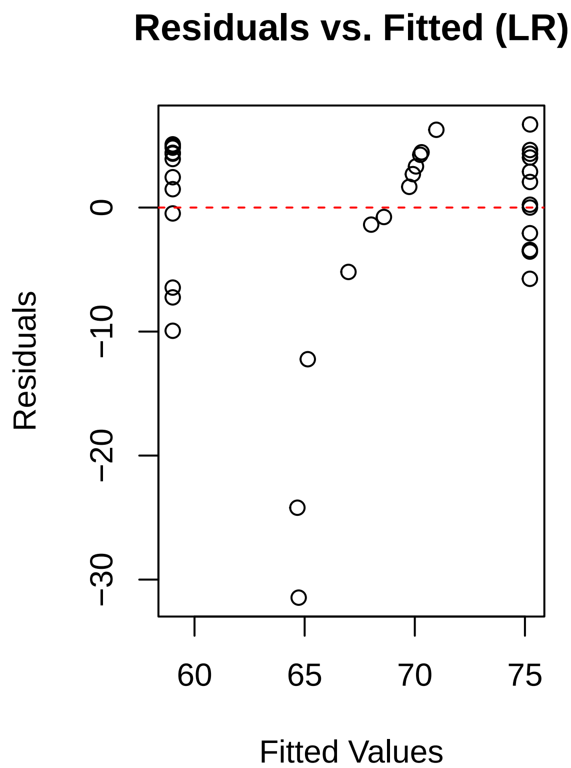




The normal Q-Q plot as well shows that the Random Forest model performs better and the points follows the line though there are some deviations, they are not as bad as the one for Linear Regression model.

```r
# Create Residuals vs. Fitted Plots
par(mfrow = c(1, 2))

# Residuals vs. Fitted Plot for Random Forest
plot(RF_preds, rf_residuals, main = "Residuals vs. Fitted (RF)",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)


# Residuals vs. Fitted Plot for Linear Regression
plot(LR_preds, lm_residuals, main = "Residuals vs. Fitted (LR)",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)
```





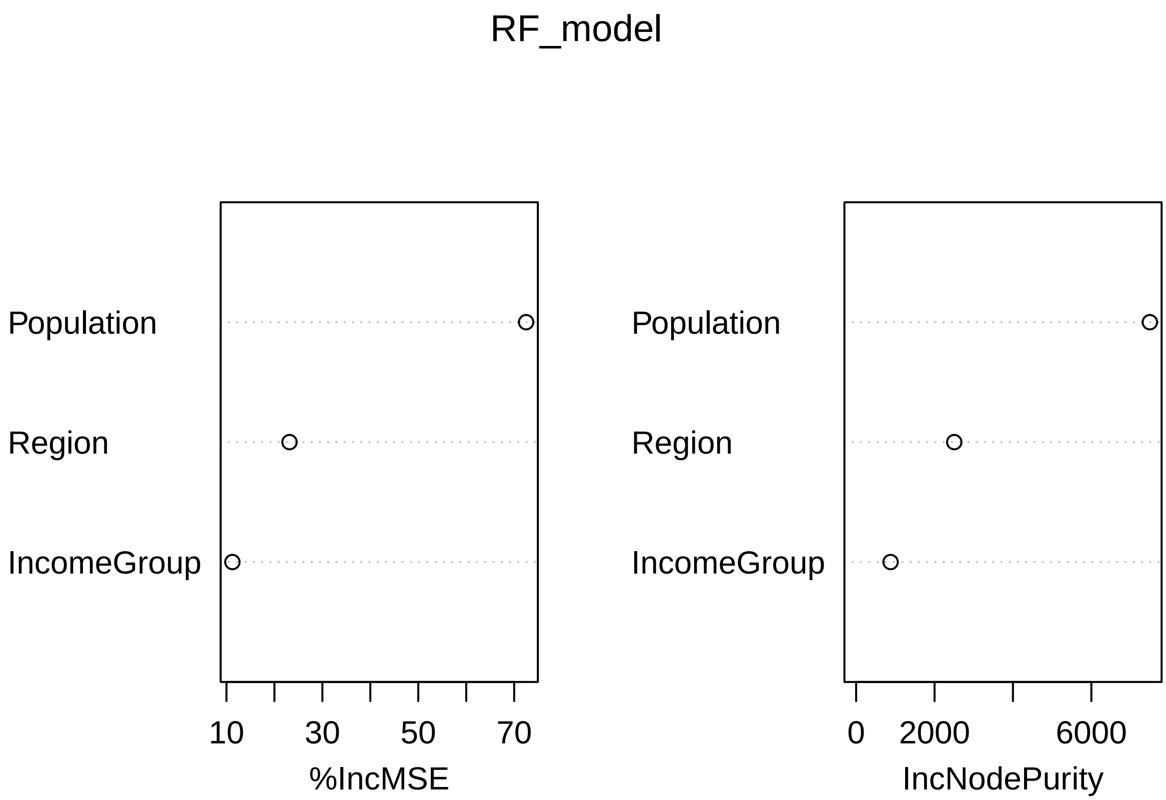

```r
# Reset the plotting layout
par(mfrow = c(1, 1))
```

The Random Forest model shows it outperforms the Linear Regression model.

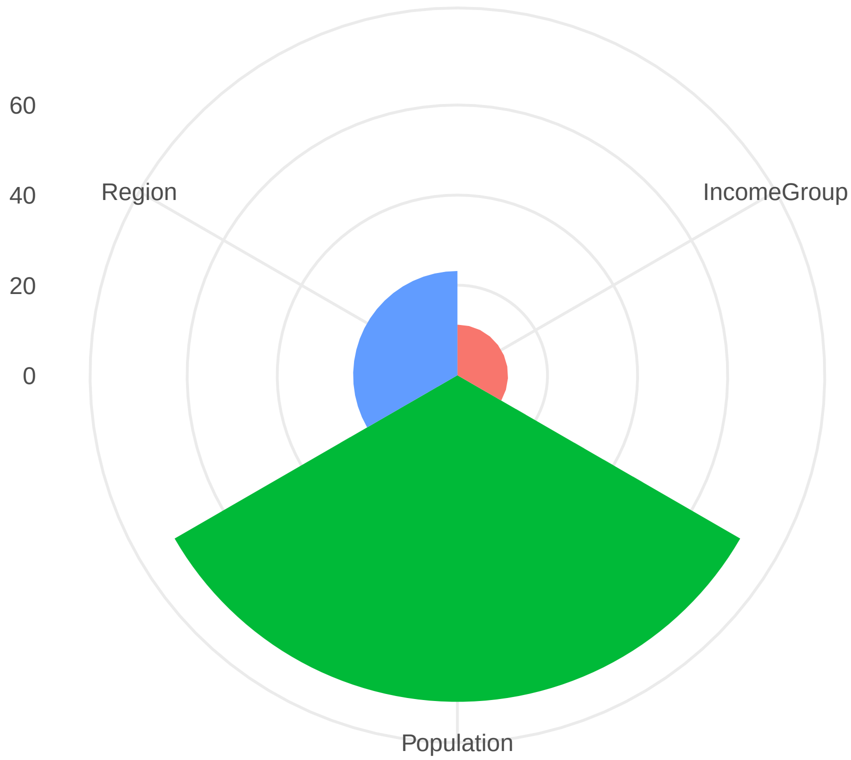
Therefore, we can say that we choose the Random Forest method for us life expectancy prediction.

The Most Important Variable

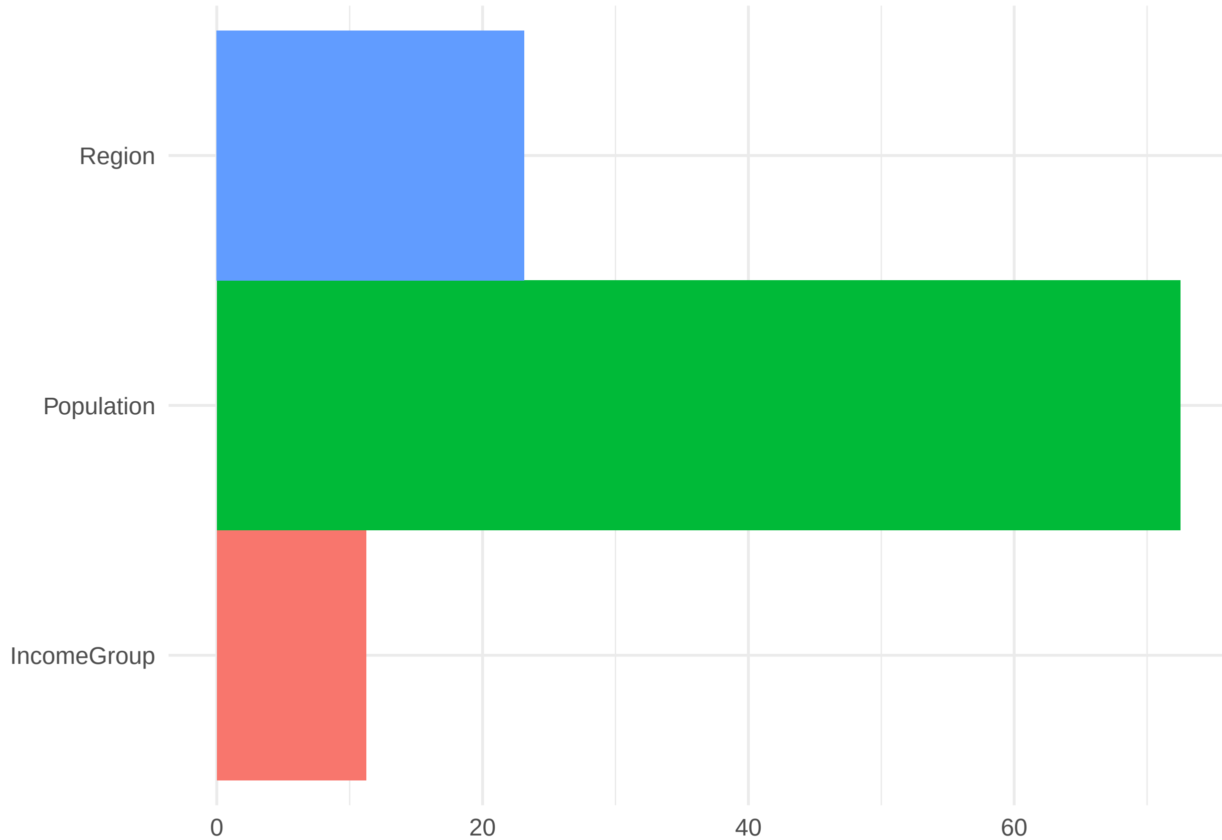
```r
# Random Forest
varImpPlot(RF_model)
```




```
#Conditional=True, adjusts for correlations between predictors.
i_scores <- varImp(RF_model, conditional=TRUE)

#Gathering rownames.
i_scores <- i_scores %>% tibble::rownames_to_column("var")
i_scores$var<- i_scores$var %>% as.factor()

#Plotting the bar and polar charts for comparing variables
i_bar <- ggplot(data = i_scores) +
  geom_bar(stat = "identity",
           mapping = aes(x = var, y=Overall, fill = var), show.legend = FALSE, width = 1) +
  labs(x = NULL, y = NULL)
i_bar + coord_polar() + theme_minimal()
```




```
i_bar + coord_flip() + theme_minimal()
```



In this project, we wanted to see and investigate whether life expectancy can be predicted based on historical time series data. To do that, we developed three different modeling approaches:

- Auto ARIMA,
- Random Forest Regression,
- Linear Regression.

For Auto Arima model, we use different datasets from different countries like China, Tuvalu and Finland. We selected the countries based on the population. The most populated country, the country whose population was nearest to the median population, and the least populated country. We developed auto Arima model for both population and Life expectancy. All the models showed predictive capabilities and showed us that indeed, we can use historical data to predict life expectancy and population.

For Random Forest and Linear Regression, we trained our models with a combination of the three countries. The data included population size, region, and income group as potential predictors. We compared the models performance based on the performance metrics and it revealed that Random forest Regression was the best performing model.

Also, from the two models, indicate that life expectancy can indeed be predicted with a high level of accuracy, especially using the Random Forest model.

The findings from the feature importance of the Random Forest model is that the most important feature or predictor for the life expectancy rate is population, which according to the value importance plot, contributes to about 60% of the predictive power. That shows population has the most influence on the life expectancy rate of any country or region. The variable with the least contributing power is the income group with around 11%.

Key Predictors: Population size emerged as the most critical predictor of life expectancy.