HW 5

Enter your name and EID here: Jongho Yoo (jy23294)

You will submit this homework assignment as a pdf file on Gradescope.

For all questions, include the R commands/functions that you used to find your answer (show R chunk). Answers without supporting code will not receive credit. Write full sentences to describe your findings.

Question 1: (1 pt)

The dataset `world_bank_pop` is a built-in dataset in `tidyverse`. It contains information about total population and population growth, overall and more specifically in urban areas, for countries around the world. Take a look at it with `head()`. Is the data tidy? Why or why not?

```
# Call tidyr, dplyr and ggplot2 packages within tidyverse
library(tidyverse)

# Take a look!
head(world_bank_pop)
```

```
## # A tibble: 6 x 20
##   country indic~1 ‘2000‘  ‘2001‘  ‘2002‘  ‘2003‘  ‘2004‘  ‘2005‘  ‘2006‘  ‘2007‘
##   <chr>   <chr>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 ABW     SP.URB~ 4.16e4 4.20e+4 4.22e+4 4.23e+4 4.23e+4 4.24e+4 4.26e+4 4.27e+4
## 2 ABW     SP.URB~ 1.66e0 9.56e-1 4.01e-1 1.97e-1 9.46e-2 1.94e-1 3.67e-1 4.08e-1
## 3 ABW     SP.POP~ 8.91e4 9.07e+4 9.18e+4 9.27e+4 9.35e+4 9.45e+4 9.56e+4 9.68e+4
## 4 ABW     SP.POP~ 2.54e0 1.77e+0 1.19e+0 9.97e-1 9.01e-1 1.00e+0 1.18e+0 1.23e+0
## 5 AFE     SP.URB~ 1.16e8 1.20e+8 1.24e+8 1.29e+8 1.34e+8 1.39e+8 1.44e+8 1.49e+8
## 6 AFE     SP.URB~ 3.60e0 3.66e+0 3.72e+0 3.71e+0 3.74e+0 3.81e+0 3.81e+0 3.61e+0
## # ... with 10 more variables: ‘2008‘ <dbl>, ‘2009‘ <dbl>, ‘2010‘ <dbl>,
## #   ‘2011‘ <dbl>, ‘2012‘ <dbl>, ‘2013‘ <dbl>, ‘2014‘ <dbl>, ‘2015‘ <dbl>,
## #   ‘2016‘ <dbl>, ‘2017‘ <dbl>, and abbreviated variable name 1: indicator
```

No, the years are separated into many different columns, and as a result, the values are also separated into different columns. The years should be listed under one column, and values should also have its own column. In addition, the indicator variable has many different categories, which should each have its own column.

Question 2: (1 pt)

Using `dplyr` functions on `world_bank_pop`, count how many distinct countries there are in the dataset. Does this makes sense? Why or why not?

```
# use summarize to count the distinct countries in the dataset
world_bank_pop %>%
  summarize(n_distinct(country))
```

```
## # A tibble: 1 x 1
##   `n_distinct(country)`
##                   <int>
## 1                   266
```

There are 266 distinct countries in the dataset. It does not make sense because there are only 195 countries that exist.

Question 3: (2 pts)

Use one of the `pivot` functions on `world_bank_pop` to create a new dataset with the years 2000 to 2017 appearing as a *numeric* variable `year`, and the different values for the indicator variable are in a variable called `value`. Save this new dataset in your environment as `myworld1`.

```
# make the different years go under one "year" variable and the values in a separate variable
myworld1 <- pivot_longer(world_bank_pop,
               cols = 3:20,
               names_to = "year",
               values_to = "value")
```

How many lines are there per country? Why does it make sense?

```
# first group by country, then find number of lines
myworld1 %>%
  group_by(country) %>%
  summarize(lines = n())
```

```
## # A tibble: 266 x 2
##    country lines
##    <chr>   <int>
##  1 ABW        72
##  2 AFE        72
##  3 AFG        72
##  4 AFW        72
##  5 AGO        72
##  6 ALB        72
##  7 AND        72
##  8 ARB        72
##  9 ARE        72
## 10 ARG        72
## # ... with 256 more rows
```

There are 72 lines per country. This makes sense because each country has the same indicator-year combinations, of which there are 72.

Question 4: (3 pts)

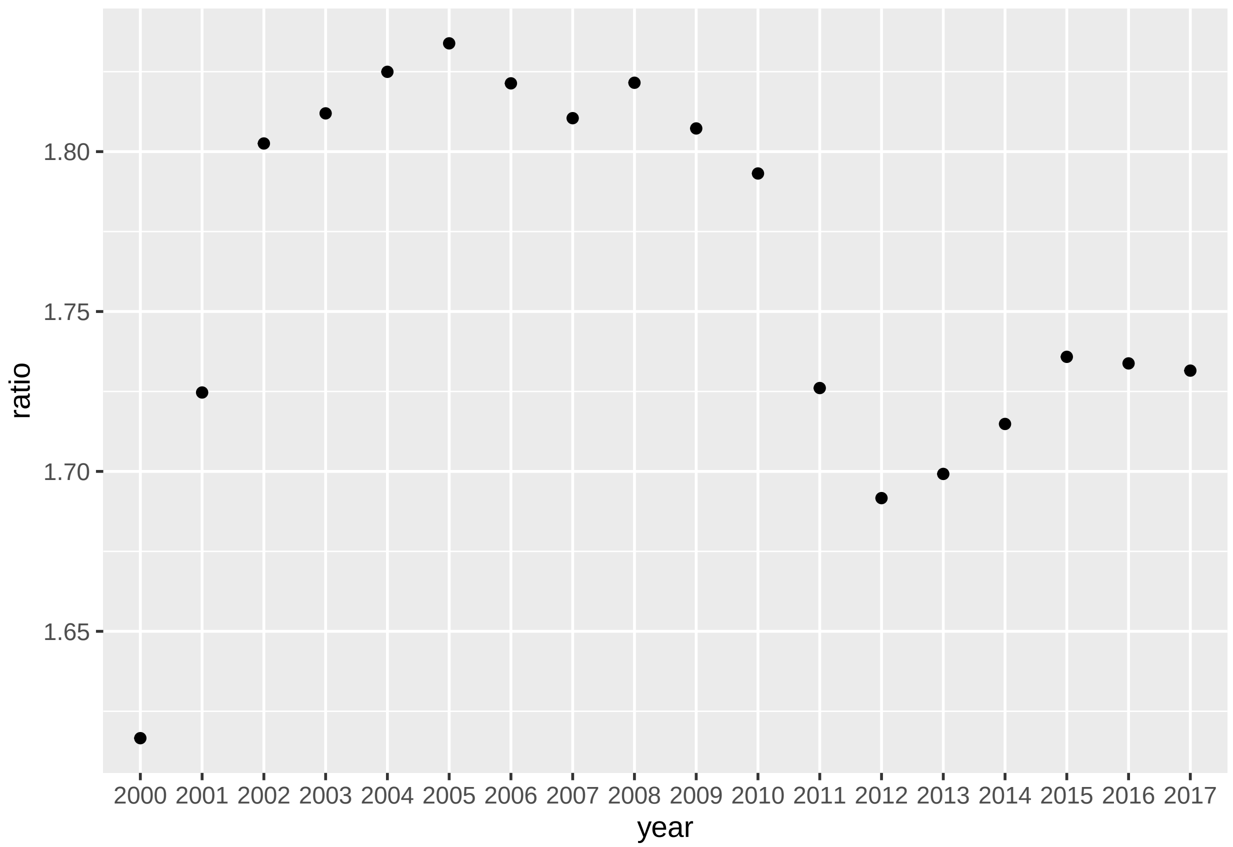
Use another pivot function on myworld1 to create a new dataset, myworld2, with the different categories for the indicator variable appearing as their own variables. Use dplyr functions to rename SP.POP.GROW and SP.URB.GROW, as pop_growth and pop_urb_growth respectively.

```
# pivot table to separate the 4 different indicators as their own variables
myworld2 <- pivot_wider(myworld1,
              names_from = indicator,
              values_from = value)

# rename variables
myworld2 <- myworld2 %>%
  rename("pop_growth" = "SP.POP.GROW",
         "pop_urb_growth" = "SP.URB.GROW")
```

Using dplyr functions, find the ratio of urban growth compared to the population growth in the world for each year. *Hint: the country code WLD represents the entire world.* Create a ggplot to display how the percentage of urban population growth has changed over the years. Why does your graph not contradict the fact that the urban population worldwide is increasing over the years?

```
# filter for WLD, add ratio using mutate, then use ggplot to create scatterplot
myworld2 %>%
  filter(country == "WLD") %>%
  mutate(ratio = pop_urb_growth/pop_growth) %>%
  ggplot(aes(x = year, y = ratio)) +
  geom_point()
```



Even though the graph has ups and downs, the ratio at each year is above one. This means for each year, urban growth is greater than population growth, and thus does not contradict increasing worldwide urban population.

Question 5: (1 pt)

In `myworld2`, which country code had the highest population growth in 2017?

```
# filter for year 2017, arrange highest to lowest, then return first value
myworld2 %>%
  filter(year == 2017) %>%
  arrange(desc(pop_growth)) %>%
  head(1)
```

```
## # A tibble: 1 x 6
##   country year  SP.URB.TOTL pop_urb_growth SP.POP.TOTL pop_growth
##   <chr>   <chr>       <dbl>          <dbl>       <dbl>      <dbl>
## 1 QAT     2017      2686753           4.46     2711755       4.39
```

QAT had the highest population growth in 2017.

Question 6: (1 pt)

When answering the previous, we only reported the three-letter code and (probably) have no idea what the actual country is. We will now use the package `countrycode` with a built-in dataset called `codelist` that has information about the coding system used by the World bank:

```
# Paste and run the following into your console (NOT HERE): install.packages("countrycode")

# Call the countrycode package
library(countrycode)

# Create a list of codes with matching country names
mycodes <- codelist %>%
  select(continent, wb, country.name.en) %>%
  na.omit(wb)
```

Using `dplyr` functions, modify `mycodes` above to only keep the variables `continent`, `wb` (World Bank code), and `country.name.en` (country name in English). Then remove countries with missing `wb` code.

How many countries are there in `mycodes`?

```
# find how many distinct countries are in mycodes
mycodes %>%
  n_distinct
```

```
## [1] 216
```

There are 216 countries in mycodes

Question 7: (1 pt)

Use a `left_join()` function to add the information of the country codes **to** `myworld2` dataset. Match the two datasets based on the World Bank code. *Note: the World Bank code does not have the same name in each dataset.* Using `dplyr` functions, only keep the data available for Europe and for the year 2017. Save this new dataset as `myeurope`.

```
# left join mycodes onto myworld2, with country and wb as the same keyword
myeurope <- left_join(myworld2, mycodes, by = c("country" = "wb")) %>%
  # filter to only keep data on europe in 2017
  filter(continent == "Europe", year == 2017)
```

How many rows are there in `this new dataset`myeurope‘? What does each row represent?

```
# use nrow to find how many rows exist
nrow(myeurope)
```

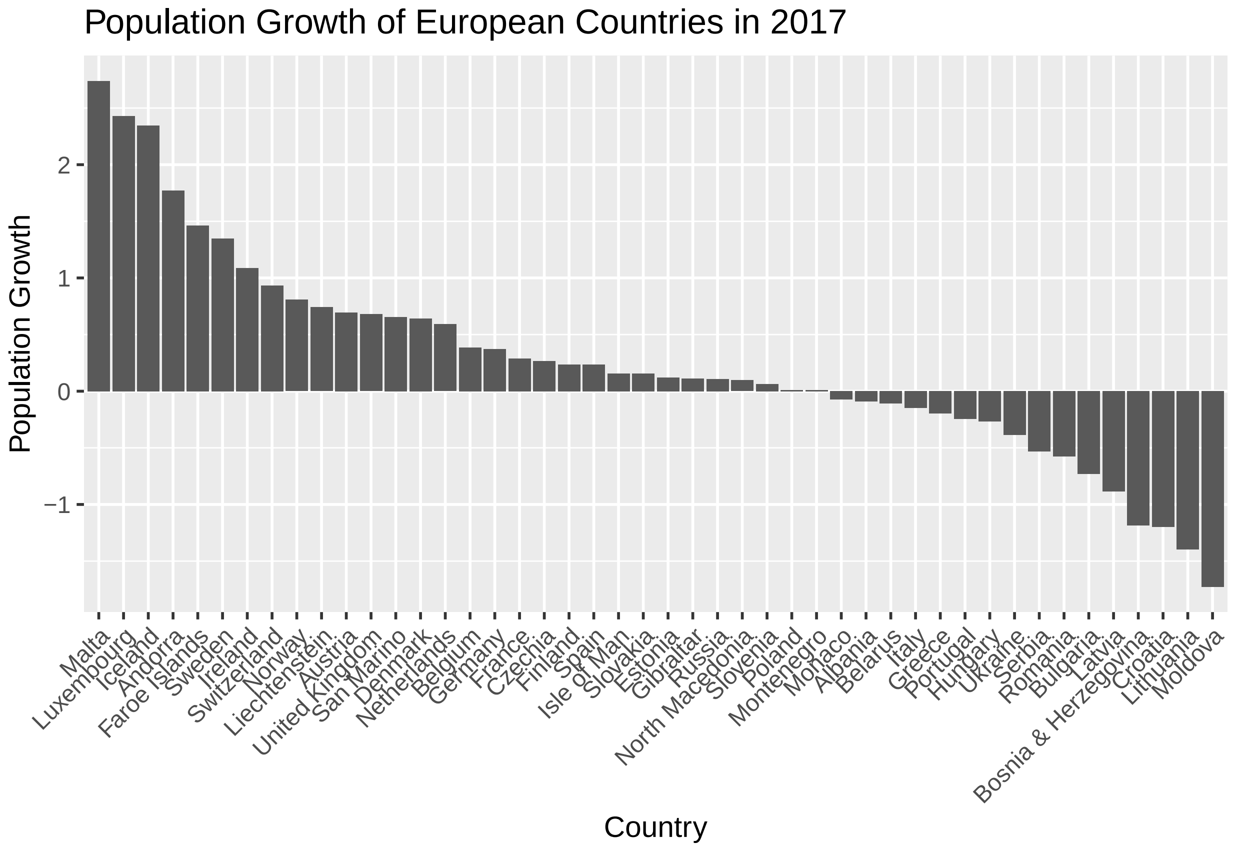
```
## [1] 46
```

There are 46 rows in the myeurope dataset. Each row represents the statistics for a given country.

Question 8: (2 pts)

Using dplyr functions on myeurope, only keep information for the population growth in 2017 then compare the population growth per country with ggplot using geom_bar(). Make sure to order countries in order of population growth. Which country in Europe had the lowest population growth in 2017?

```r
# first select country name and pop_growth
myeurope %>%
  select(country.name.en, pop_growth) %>%
  # arrange highest to lowest
  arrange(desc(pop_growth)) %>%
  # use reorder function to order by highest population growth to lowest
  ggplot(aes(x = reorder(country.name.en, -pop_growth), y = pop_growth)) +
  geom_bar(stat = "summary", fun = "mean") +
  # tilt country names to fit on x-axis
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Population Growth of European Countries in 2017",
       x = "Country",
       y = "Population Growth")
```



Moldova had the lowest population growth in 2017.

Question 9: (1 pt)

When dealing with location data, we can actually visualize information on a map if we have geographic information such as latitude and longitude. Next, we will use a built-in function called `map_data()` to get geographic coordinates about countries in the world (see below). Take a look at the dataset `mapWorld`. What variables could we use to join `mapWorld` and `myeurope`? *Note: the variables do not have the same name in each dataset but they contain the same information.*

```
# Geographic coordinates about countries in the world
mapWorld <- map_data("world")
```

The region in mapWorld and the country name in myeurope can be used to join the two datasets.

Question 10: (2 pts)

Use a joining function to check if any information from `myeurope` is not contained in `mapWorld`, matching the two datasets based on the country name.

```
# use anti-join to check information from myeurope not contained in mapWorld
anti_join(myeurope, mapWorld, by = c("country.name.en" = "region"))
```

```
## # A tibble: 4 x 8
##   country year  SP.URB.TOTL pop_urb_growth SP.POP.TOTL pop_gro~1 conti~2 count~3
##   <chr>   <chr>       <dbl>          <dbl>       <dbl>     <dbl> <chr>   <chr>
## 1 BIH     2017      1646947         -0.433     3440027    -1.18  Europe  Bosnia~
## 2 CZE     2017      7805452          0.408    10594438     0.266 Europe  Czechia
## 3 GBR     2017     54923317          0.989    66058859     0.679 Europe  United~
## 4 GIB     2017        32602          0.114       32602     0.114 Europe  Gibral~
## # ... with abbreviated variable names 1: pop_growth, 2: continent,
## #   3: country.name.en
```

Some countries such as United Kingdom did not have a match. Why do you think this happened? *Hint: find the distinct country names in `mapWorld`, arrange them in alphabetical order, and scroll through the names. Can you find any of these countries with no match in a slightly different form?*

```
# first get the distinct country names, then arrange in alphabetical order
mapWorld %>%
  distinct(region) %>%
  arrange(region)
```

```
##                                   region
## 1                            Afghanistan
## 2                                Albania
## 3                                Algeria
## 4                         American Samoa
## 5                                Andorra
## 6                                 Angola
## 7                               Anguilla
## 8                            Antarctica
## 9                               Antigua
## 10                            Argentina
## 11                              Armenia
## 12                                Aruba
## 13                     Ascension Island
## 14                            Australia
## 15                              Austria
## 16                           Azerbaijan
## 17                               Azores
## 18                              Bahamas
## 19                              Bahrain
## 20                           Bangladesh
## 21                             Barbados
## 22                              Barbuda
## 23                              Belarus
## 24                              Belgium
## 25                               Belize
## 26                                Benin
## 27                              Bermuda
## 28                               Bhutan
## 29                              Bolivia
## 30                              Bonaire
## 31               Bosnia and Herzegovina
## 32                             Botswana
## 33                               Brazil
## 34                               Brunei
## 35                             Bulgaria
## 36                         Burkina Faso
## 37                              Burundi
## 38                             Cambodia
## 39                             Cameroon
## 40                               Canada
## 41                       Canary Islands
## 42                           Cape Verde
## 43                       Cayman Islands
## 44             Central African Republic
## 45                                 Chad
## 46                   Chagos Archipelago
## 47                                Chile
## 48                                China
## 49                     Christmas Island
## 50                        Cocos Islands
## 51                             Colombia
## 52                              Comoros
## 53                         Cook Islands
## 54                           Costa Rica
## 55                              Croatia
## 56                                 Cuba
## 57                              Curacao
## 58                               Cyprus
## 59                       Czech Republic
## 60     Democratic Republic of the Congo
## 61                              Denmark
## 62                            Djibouti
## 63                            Dominica
## 64                  Dominican Republic
## 65                             Ecuador
## 66                               Egypt
## 67                         El Salvador
## 68                   Equatorial Guinea
## 69                             Eritrea
## 70                             Estonia
## 71                            Ethiopia
## 72                    Falkland Islands
## 73                       Faroe Islands
## 74                                Fiji
## 75                             Finland
## 76                              France
## 77                       French Guiana
## 78                    French Polynesia
## 79  French Southern and Antarctic Lands
## 80                               Gabon
## 81                              Gambia
## 82                             Georgia
## 83                             Germany
## 84                               Ghana
## 85                              Greece
## 86                           Greenland
## 87                             Grenada
## 88                          Grenadines
## 89                          Guadeloupe
## 90                                Guam
## 91                           Guatemala
## 92                            Guernsey
## 93                              Guinea
## 94                       Guinea-Bissau
## 95                              Guyana
## 96                               Haiti
## 97                        Heard Island
## 98                            Honduras
## 99                             Hungary
## 100                            Iceland
##  [ reached 'max' / getOption("max.print") -- omitted 152 rows ]
```

Bosnia & Herzegovina is written with “and”, not the “&” symbol. Czechia is written as Czech Republic. United Kingdom is written has UK. These countries did not have a match because they were written slightly differently. Gibraltar was the only one that did not have a match because it was not in the mapWorld dataset, as it is a British territory, not a separate country.

Question 11: (1 pt)

Consider the `myeurope` dataset. Recode some of the country names so that the countries with no match from the previous question (with the exception of Gibraltar which is not technically a country anyway) will have a match. *Hint: use `recode()` inside `mutate()` as described in this article https://www.statology.org/recode-*

dplyr/. Then add a pipe and use a `left_join()` function to add the geographic information in `mapWorld` to the countries in `myeurope`. Save this new dataset as `mymap`.

```r
# rename the country names to match using mutate
mymap <- myeurope %>%
  mutate(recode(country.name.en,
              "Bosnia & Herzegovina" = "Bosnia and Herzegovina",
              "Czechia" = "Czech Republic",
              "United Kingdom" = "UK")) %>%
  # add mapWorld data onto myeurope using country.name.en/region as the key
  left_join(mapWorld, by = c("country.name.en" = "region"))
```

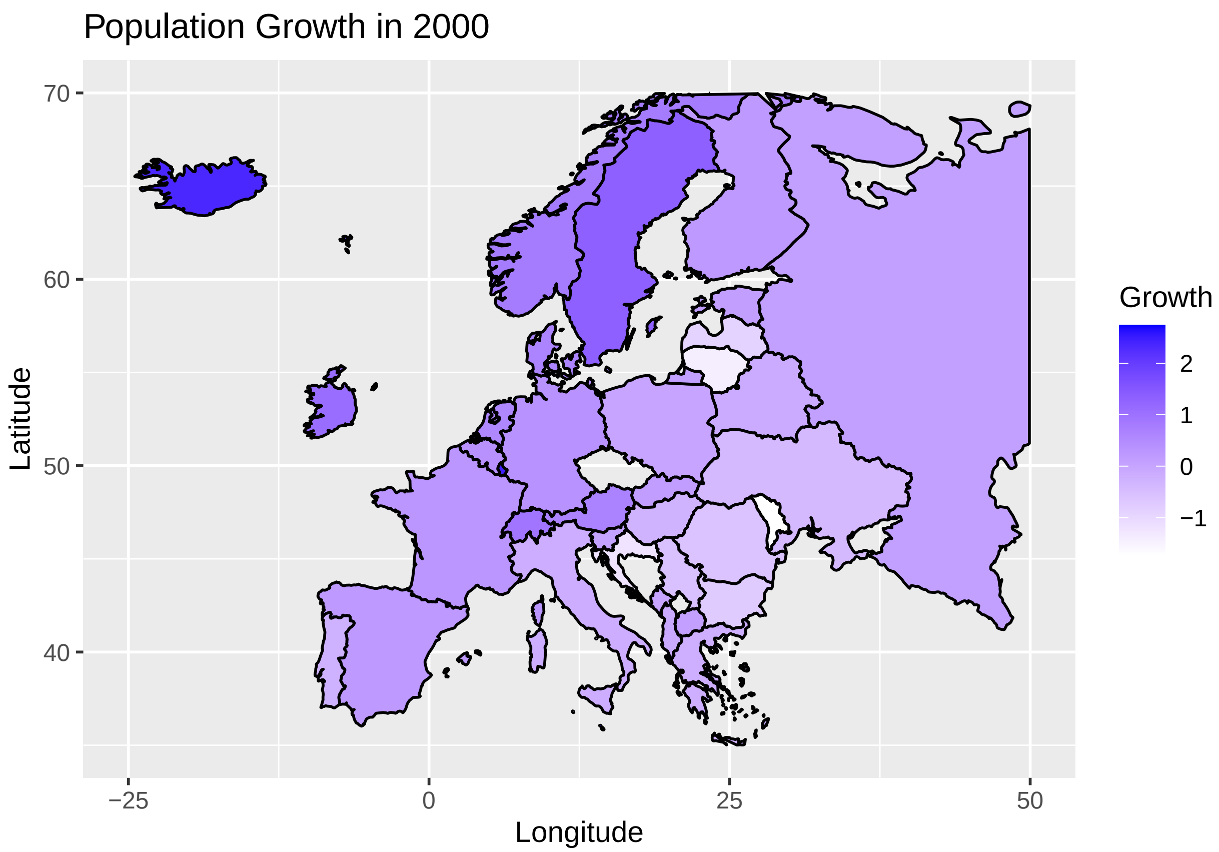
Question 12: (2 pts)

Let's visualize how population growth varies across European countries in 2017 with a map. With the package `ggmap`, use the R code provided below. Add a comment after each `#` to explain what each component of this code does. *Note: it would be a good idea to run the code piece by piece to see what each layer adds to the plot.*

```r
# Paste and run the following into your console (NOT HERE): install.packages("ggmap")

# Call the ggmap package
library(ggmap)

# Build a map!
mymap %>%
  # define aesthetics: x as longitude, y as latitude, group countries by their group
  # code (which is unique for each country), and fill color by population growth
  ggplot(aes(x = long, y = lat, group = group, fill = pop_growth)) +
  # make outline/border of countries black
  geom_polygon(colour = "black") +
  # change fill color of countries on a scale gradient of white(low) to blue(high)
  scale_fill_gradient(low = "white", high = "blue") +
  # create labels for the graph
  labs(fill = "Growth" ,title = "Population Growth in 2000",
       x ="Longitude", y ="Latitude") +
  # set x-axis and y-axis boundaries (set window) to focus on a specific part of the map
  xlim(-25,50) + ylim(35,70)
```



Which country had the highest population growth in Europe in 2017? *Hint: it's very tiny and very close to where I'm from! You can refer to this map for European geography: https://www.wpmap.org/europe-map-hd-with-countries/*

Luxembourg had the highest population growth in Europe in 2017.

Formatting: (2 pts)

Comment your code, write full sentences, and knit your file!

```
##                                                               sysn
##                                                              "Darw
##                                                               rele
##                                                              "21.3
##                                                               vers
## "Darwin Kernel Version 21.3.0: Wed Jan  5 21:37:58 PST 2022; root:xnu-8019.80.24~20/RELEASE_ARM64_T8
##                                                              noden
##                                                        "Stevens-MBP-2.
##                                                               mach
##                                                               "arm
##                                                                 lo
##                                                                                                          "re
##                                                                                                            u
##                                                                                                     "stevenj
##                                                                                                  effective_u
##                                                                                                     "stevenj
```