HW 5

Enter your name and EID here: Harini Shanmugam hs28663

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 indicator       `2000`  `2001`  `2002` `2003`  `2004`  `2005`   `2006`
##   <chr>   <chr>            <dbl>   <dbl>   <dbl>  <dbl>   <dbl>   <dbl>    <dbl>
## 1 ABW     SP.URB.TOTL   42444     4.30e4  4.37e4 4.42e4 4.47e+4 4.49e+4  4.49e+4
## 2 ABW     SP.URB.GROW       1.18  1.41e0  1.43e0 1.31e0 9.51e-1 4.91e-1 -1.78e-2
## 3 ABW     SP.POP.TOTL   90853     9.29e4  9.50e4 9.70e4 9.87e+4 1.00e+5  1.01e+5
## 4 ABW     SP.POP.GROW       2.06  2.23e0  2.23e0 2.11e0 1.76e+0 1.30e+0  7.98e-1
## 5 AFG     SP.URB.TOTL 4436299     4.65e6  4.89e6 5.16e6 5.43e+6 5.69e+6  5.93e+6
## 6 AFG     SP.URB.GROW       3.91  4.66e0  5.13e0 5.23e0 5.12e+0 4.77e+0  4.12e+0
## # ... with 11 more variables: 2007 <dbl>, 2008 <dbl>, 2009 <dbl>, 2010 <dbl>,
## #   2011 <dbl>, 2012 <dbl>, 2013 <dbl>, 2014 <dbl>, 2015 <dbl>, 2016 <dbl>,
## #   2017 <dbl>
```

No, it is not tidy because not every variable has its own column. Specifically, it would have been tidy if there was a column for year rather than each year being a column.

Question 2: (1 pt)

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

```
world_bank_pop %>%
  # count number of distinct countries in dataset
  summarize(number_of_distinct_countires = n_distinct(country))
```

```
## # A tibble: 1 x 1
##   number_of_distinct_countires
##                          <int>
## 1                          264
```

There are 264 distinct countries in the dataset. No, this doesn't make sense because there are only 197 officially recognized countries in the world.

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`.

```
myworld1 <- world_bank_pop %>%
  pivot_longer(cols = c('2000':'2017'), # turn columns into one column
               names_to = "year", # name new column as "year"
               values_to = "value") %>% # name new values as "value"
  mutate(year = as.numeric(year)) # make "year" variable numeric
```

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

```
myworld1 %>%
  group_by(country) %>% # group by country
  count(country) # count number of occurrences of each country
```

```
## # A tibble: 264 x 2
## # Groups:   country [264]
##    country     n
##    <chr>   <int>
##  1 ABW        72
##  2 AFG        72
##  3 AGO        72
##  4 ALB        72
##  5 AND        72
##  6 ARB        72
##  7 ARE        72
##  8 ARG        72
##  9 ARM        72
## 10 ASM        72
## # ... with 254 more rows
```

There are 72 lines per country. This makes sense because for each year there are 4 different indicator types and 18 years of records for each indicator type.

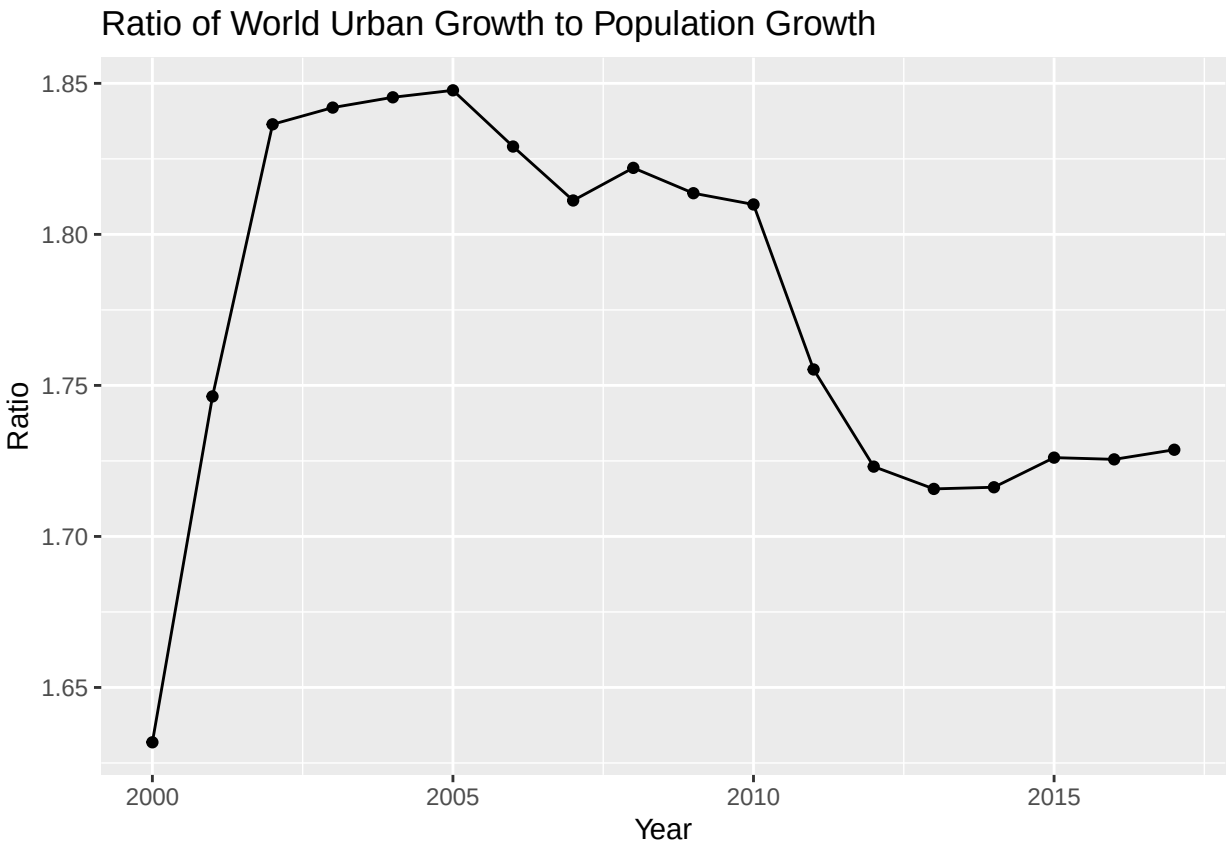
Question 4: (3 pts)

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.

```
myworld2 <- myworld1 %>%
  pivot_wider(names_from = "indicator", # differentiate by type for new columns
              values_from = "value") %>%  # keep values under respective variable
  rename(pop_growth = SP.POP.GROW, # rename variables
         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?

```
myworld2 %>%
  filter(country=="WLD") %>%
  # create new variable holding ratio of world urban growth to pop growth
  mutate(ratio = pop_urb_growth/pop_growth) %>%
  ggplot(aes(x=year, y=ratio)) + # plot of year vs ratio
  geom_point() +
  geom_line() +
  labs(title="Ratio of World Urban Growth to Population Growth",
       x="Year",
       y="Ratio")
```



Although a part of the graph is decreasing, it does not contradict the fact that the urban population worldwide is increasing over the years because the ratio is above 1 (aka 100%), meaning the population of urban growth is more than the current population growth.

Question 5: (1 pt)

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

```
myworld2 %>%
  filter(year==2017) %>% # only consider rows with year 2017
  slice_max(pop_growth) # row with maximum value of pop_growth
```

```
## # A tibble: 1 x 6
##   country  year SP.URB.TOTL pop_urb_growth SP.POP.TOTL pop_growth
##   <chr>   <dbl>       <dbl>          <dbl>       <dbl>      <dbl>
## 1 OMN      2017     3874061           5.95     4636262       4.67
```

OMN 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 %>%
  # only display these variables/columns
  select(continent, wb, country.name.en) %>%
  filter(!is.na(wb)) # get rid of na values
```

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`?

```
# count number of countries/ rows of observations
mycodes %>%
  count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   218
```

There are 218 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`.

```
# set it to new object "myeurope"
myeurope <- myworld2 %>%
  # join myworld2 and mycodes by country names under
  # columns "country" and "wb" respectively.
  left_join(mycodes, by=c("country" = "wb")) %>%
  # only show thoes in the Europe continent and for year 2017
  filter(continent=="Europe", year==2017)
```

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

```r
# count number of countries/ rows of observations
myeurope %>%
  count()
```

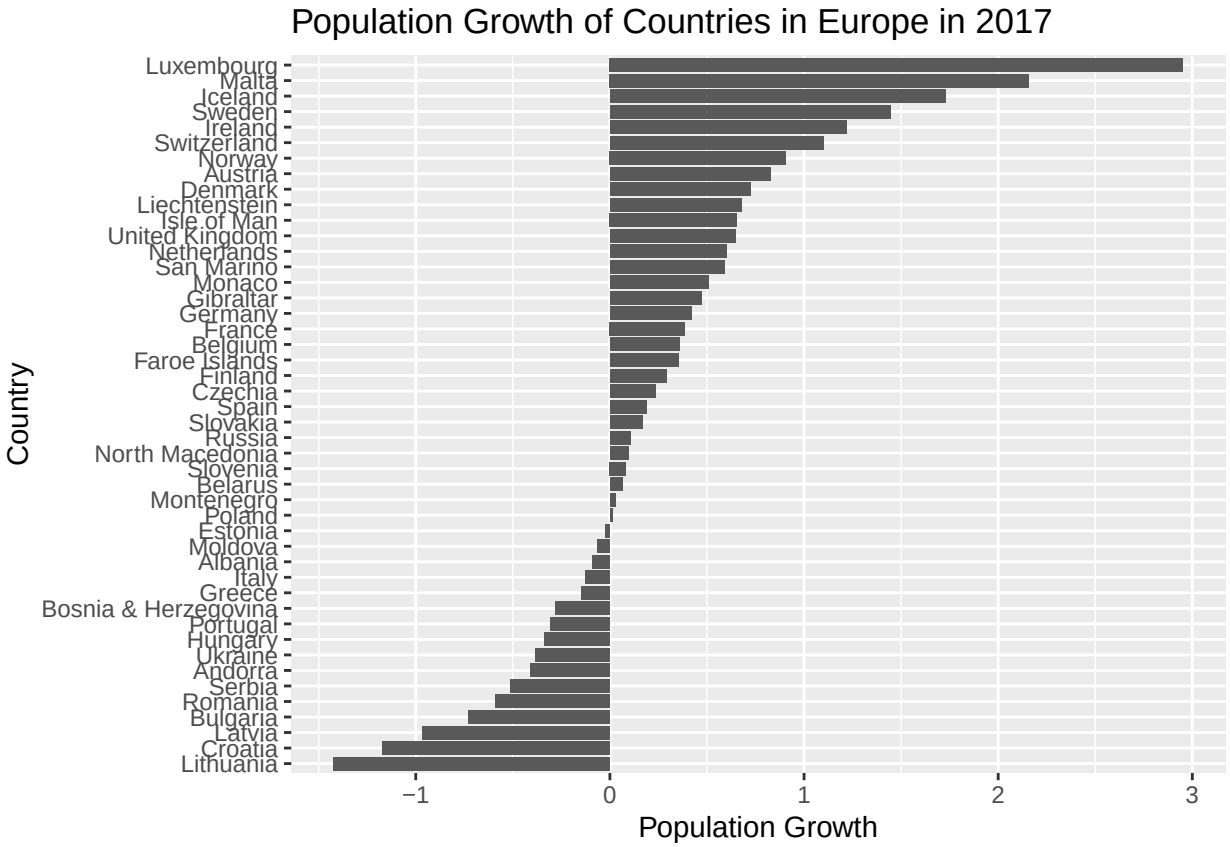
```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    46
```

There are 46 rows in the new dataset. Each row represents a different 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
myeurope %>%
  # only display population growth from year 2017
  filter(year==2017) %>%
  # arrange in ascending order
  arrange(pop_growth) %>%
  # bar graph of popultion growth for all counties
  ggplot(aes(y=reorder(country.name.en,pop_growth), x=pop_growth)) +
  geom_bar(stat="identity") +
  labs(title="Population Growth of Countries in Europe in 2017",
       y="Country",
       x="Population Growth")
```



Lithuania (LTU) was the country with the lowest population growth in Europe 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")
```

We could join "region" from mapWorld and "country.name.en" from myeurope.

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.

```
# return all observations in myeurpoe that is missing in mapWorld
myeurope %>%
  anti_join(mapWorld, by=c("country.name.en" = "region"))
```

```
## # A tibble: 4 x 8
##   country  year SP.URB.TOTL pop_urb_growth SP.POP.TOTL pop_growth continent
##   <chr>   <dbl>       <dbl>          <dbl>       <dbl>      <dbl> <chr>
## 1 BIH      2017     1679019          0.472     3507017     -0.279 Europe
## 2 CZE      2017     7803157          0.379    10591323      0.236 Europe
## 3 GBR      2017    54892898          0.958    66022273      0.648 Europe
## 4 GIB      2017       34571          0.473       34571      0.473 Europe
## # ... with 1 more variable: country.name.en <chr>
```

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?*

```
mapWorld %>%
  distinct(region) %>% # list distinct country names
  arrange(region) # arrange in alphabetical order
```

```
##                              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 ]
```

The United Kingdom may not have had a match because it was listed by its abbreviation UK instead of it being fully spelled out, unlike a country like the UAE which was fully spelled out.

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`.

```
mymap <- myeurope %>%
  # change some specific set country names to a new spelling
  mutate(recode(country.name.en,
                "United Kingdom" = "UK",
                "Bosnia & Herzegovina" = "Bosnia and Herzegovina",
                "Chezchia" = "Czech Republic",
                "North Macedonia" = "Macedonia")) %>%
  left_join(mapWorld, by = c("country.name.en"="region")) # adding to mapWorld
```

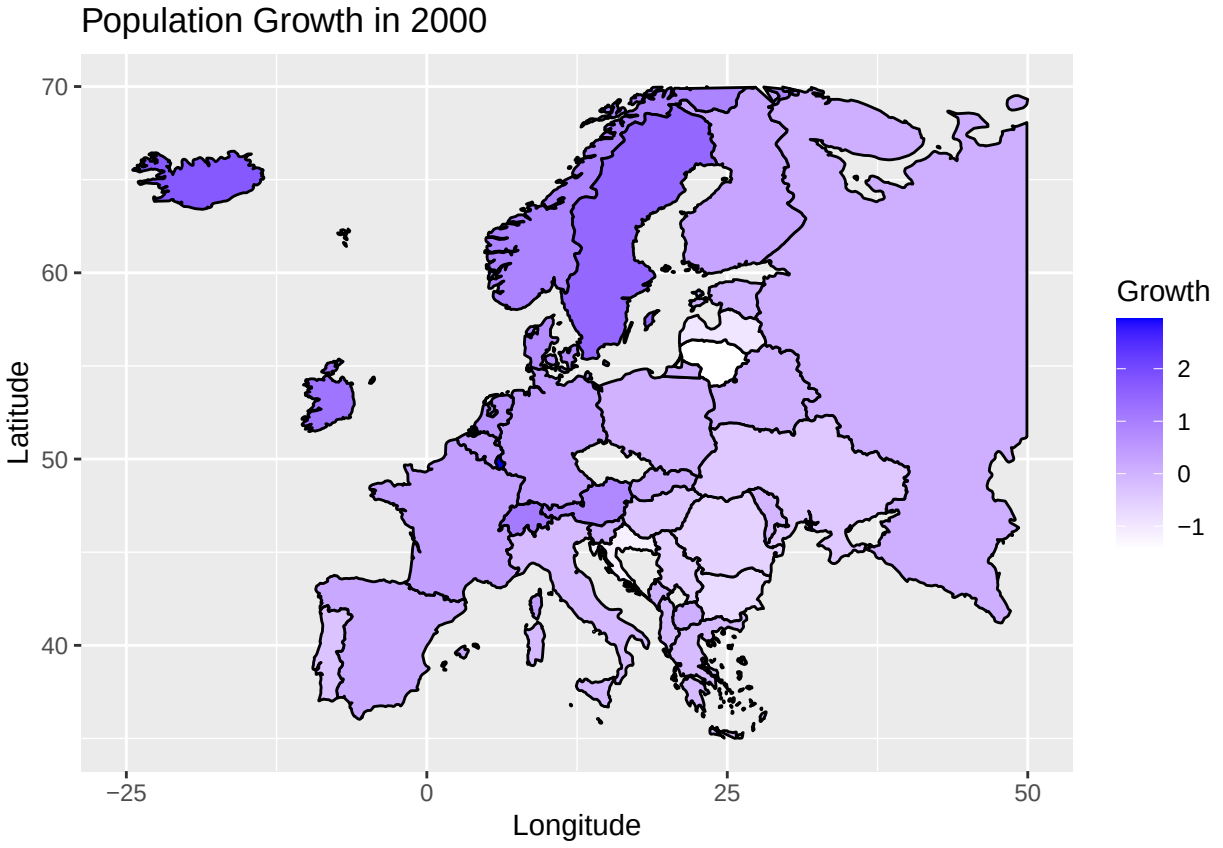
Question 12: (2 pts)

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 %>%
  # Set the aesthetics
  ggplot(aes(x = long, y = lat, group = group, fill = pop_growth)) +
  # Display the country borders in black
  geom_polygon(colour = "black") +
  # choose how the growth is displayed: color from white to blue, between -3 and 3
  scale_fill_gradient(low = "white", high = "blue") +
  # Give a title, label axes
  labs(fill = "Growth" ,title = "Population Growth in 2000",
       x ="Longitude", y ="Latitude") +
  # Center the map in Europe
  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!

```
##                                                                                                                 sysna
##                                                                                                                "Darwi
##                                                                                                                 relea
##                                                                                                                "21.6
##                                                                                                                 versi
## "Darwin Kernel Version 21.6.0: Wed Aug 10 14:28:35 PDT 2022; root:xnu-8020.141.5~2/RELEASE_ARM64_T810
##                                                                                                              nodena
##                                                                                "wireless-10-146-198-6.public.utexas.ed
##                                                                                                                 machi
##                                                                                                                 "arm6
##                                                                                                                   log
##                                                                                                                  "roo
##                                                                                                                    us
##                                                                                                "harinishanmuga
##                                                                                                   effective_us
##                                                                                                "harinishanmuga
```