HW 5

Enter your name and EID here: Joseph Hendrix | jlh7459

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 × 20
##   country indica…¹ `2000` `2001` `2002` `2003`  `2004`  `2005`   `2006`   `2007`
##   <chr>   <chr>     <dbl>  <dbl>  <dbl>  <dbl>   <dbl>   <dbl>    <dbl>    <dbl>
## 1 ABW     SP.URB.… 4.24e4 4.30e4 4.37e4 4.42e4 4.47e+4 4.49e+4  4.49e+4  4.47e+4
## 2 ABW     SP.URB.… 1.18e0 1.41e0 1.43e0 1.31e0 9.51e-1 4.91e-1 -1.78e-2 -4.35e-1
## 3 ABW     SP.POP.… 9.09e4 9.29e4 9.50e4 9.70e4 9.87e+4 1.00e+5  1.01e+5  1.01e+5
## 4 ABW     SP.POP.… 2.06e0 2.23e0 2.23e0 2.11e0 1.76e+0 1.30e+0  7.98e-1  3.84e-1
## 5 AFG     SP.URB.… 4.44e6 4.65e6 4.89e6 5.16e6 5.43e+6 5.69e+6  5.93e+6  6.15e+6
## 6 AFG     SP.URB.… 3.91e0 4.66e0 5.13e0 5.23e0 5.12e+0 4.77e+0  4.12e+0  3.65e+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 ¹indicator
```

The data is tidy; every variable has its own column, every observation has its own row, and every value has its own cell.

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?

```
# pull distinct countries from dataset
world_bank_pop %>%
  summarize(n_distinct(country))
```

```
## # A tibble: 1 × 1
##   `n_distinct(country)`
##                   <int>
## 1                   264
```

There are 264 distinct countries in the dataset. This doesn't make sense as there are only 189 members of the World Bank.

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`.

```
# pivot dataset to elongate year variable
myworld1 <- pivot_longer(world_bank_pop,
                         cols = c(3:20),
                         names_to = "year",
                         values_to = "value")



myworld1
```

```
## # A tibble: 19,008 × 4
##    country indicator   year  value
##    <chr>   <chr>       <chr> <dbl>
##  1 ABW     SP.URB.TOTL 2000  42444
##  2 ABW     SP.URB.TOTL 2001  43048
##  3 ABW     SP.URB.TOTL 2002  43670
##  4 ABW     SP.URB.TOTL 2003  44246
##  5 ABW     SP.URB.TOTL 2004  44669
##  6 ABW     SP.URB.TOTL 2005  44889
##  7 ABW     SP.URB.TOTL 2006  44881
##  8 ABW     SP.URB.TOTL 2007  44686
##  9 ABW     SP.URB.TOTL 2008  44375
## 10 ABW     SP.URB.TOTL 2009  44052
## # … with 18,998 more rows
```

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

```
myworld1 %>%
  filter(country == "ABW") %>%
  summarize(n())
```

```
## # A tibble: 1 × 1
##   `n()`
##   <int>
## 1    72
```

There are 72 rows for each country. This makes sense! There's 18 total years and 4 indicators for each year, for a total of 72.

Question 4: (3 pts)

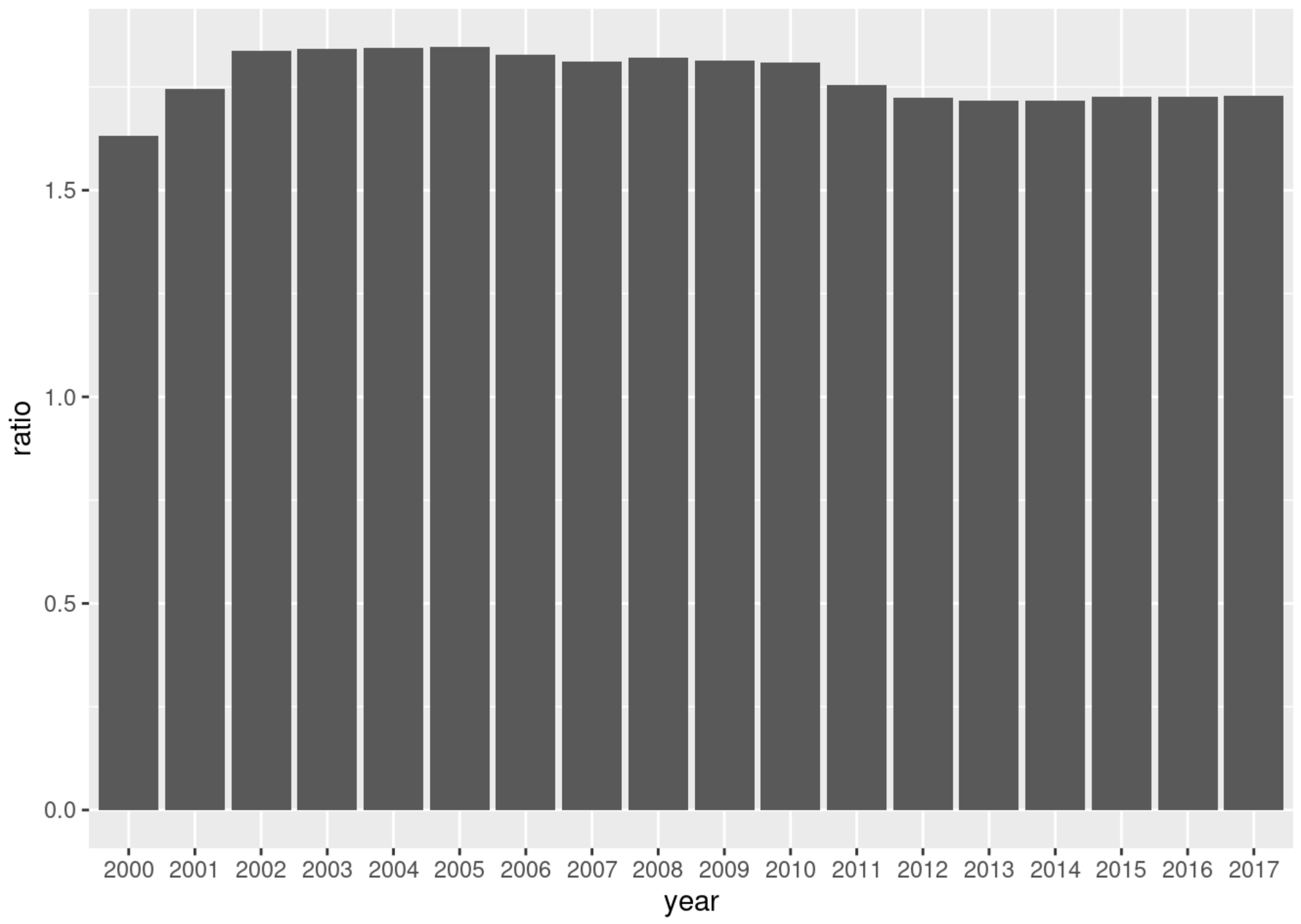
Use another `pivot` function on `myworld1` to create a new dataset, `myworld2`, with the different categories for the indicator variable appearing as their own variables. Use `dplyr` functions to rename `SP.POP.GROW` and `SP.URB.GROW`, as `pop_growth` and `pop_urb_growth` respectively.

```
# define codes for observation refactoring
codes = list("SP.URB.TOTL" = "pop_urb_total", "SP.URB.GROW" = "pop_urb_grow", "SP.POP.TOTL" = "p
op_total", "SP.POP.GROW" = "pop_growth")

myworld2 <- myworld1 %>%
  mutate(indicator = recode(indicator, !!!codes)) %>%    # refactor indicators
  pivot_wider(
    names_from = indicator,
    values_from = value
  )

myworld2
```

```
## # A tibble: 4,752 × 6
##    country year  pop_urb_total pop_urb_grow pop_total pop_growth
##    <chr>   <chr>         <dbl>        <dbl>     <dbl>      <dbl>
##  1 ABW     2000          42444       1.18       90853     2.06
##  2 ABW     2001          43048       1.41       92898     2.23
##  3 ABW     2002          43670       1.43       94992     2.23
##  4 ABW     2003          44246       1.31       97017     2.11
##  5 ABW     2004          44669       0.951      98737     1.76
##  6 ABW     2005          44889       0.491     100031     1.30
##  7 ABW     2006          44881      -0.0178    100832     0.798
##  8 ABW     2007          44686      -0.435     101220     0.384
##  9 ABW     2008          44375      -0.698     101353     0.131
## 10 ABW     2009          44052      -0.731     101453     0.0986
## # … with 4,742 more rows
```

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") %>%
  mutate(ratio = pop_urb_grow / pop_growth) %>%    # calculate growth ratio
  ggplot(aes(x = year, y = ratio)) +
  geom_col()
```



```
myworld2 %>%
  filter(country == "WLD") %>%
  mutate(ratio = pop_urb_grow / pop_growth)
```

```
## # A tibble: 18 × 7
##    country year  pop_urb_total pop_urb_grow  pop_total pop_growth ratio
##    <chr>   <chr>         <dbl>        <dbl>      <dbl>      <dbl> <dbl>
##  1 WLD     2000     2858130756         2.16 6121682736       1.32  1.63
##  2 WLD     2001     2923079567         2.27 6201340258       1.30  1.75
##  3 WLD     2002     2991628819         2.35 6280530065       1.28  1.84
##  4 WLD     2003     3061267131         2.33 6359899296       1.26  1.84
##  5 WLD     2004     3132261863         2.32 6439825381       1.26  1.85
##  6 WLD     2005     3204583153         2.31 6520298763       1.25  1.85
##  7 WLD     2006     3277558354         2.28 6601476541       1.25  1.83
##  8 WLD     2007     3351069648         2.24 6683223772       1.24  1.81
##  9 WLD     2008     3426964053         2.26 6766296679       1.24  1.82
## 10 WLD     2009     3503454963         2.23 6849569339       1.23  1.81
## 11 WLD     2010     3580569790         2.20 6932869743       1.22  1.81
## 12 WLD     2011     3655009162         2.08 7014983968       1.18  1.76
## 13 WLD     2012     3730938916         2.08 7099557649       1.21  1.72
## 14 WLD     2013     3808101888         2.07 7185137526       1.21  1.72
## 15 WLD     2014     3886498272         2.06 7271322821       1.20  1.72
## 16 WLD     2015     3966059373         2.05 7357559450       1.19  1.73
## 17 WLD     2016     4046606978         2.03 7444157356       1.18  1.73
## 18 WLD     2017     4127612962         2.00 7530360149       1.16  1.73
```

The data shows a pretty consistent ratio of urban growth to population growth. This is consistent with an increase in global urban population over the years.

Question 5: (1 pt)

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

```
# sort myworld2 by population growth from 2017
myworld2 %>%
  filter(year == 2017) %>%
  group_by(country) %>%
  arrange(desc(pop_growth))
```

```
## # A tibble: 264 × 6
## # Groups:   country [264]
##    country year  pop_urb_total pop_urb_grow pop_total pop_growth
##    <chr>   <chr>         <dbl>        <dbl>     <dbl>      <dbl>
##  1 OMN     2017        3874061         5.95   4636262       4.67
##  2 BHR     2017        1331176         4.73   1492584       4.62
##  3 NRU     2017          13649         4.50     13649       4.50
##  4 NER     2017        3511546         4.18  21477348       3.82
##  5 GNQ     2017         908248         4.42   1267689       3.71
##  6 AGO     2017       19311773         4.38  29784193       3.31
##  7 UGA     2017        9942492         5.76  42862958       3.26
##  8 COD     2017       35691987         4.57  81339988       3.25
##  9 BDI     2017        1380411         5.72  10864245       3.18
## 10 TZA     2017       18942681         5.28  57310019       3.08
## # … with 254 more rows
```

Oman (OMN) has the highest population growth in 2017 at 4.669.

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 # continue this code...
```

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`?

```
# refactor mycodes to cull irrelevant info
mycodes = mycodes %>%
  select(continent, wb, country.name.en) %>%
  filter(wb != "")

mycodes
```

```
## # A tibble: 218 × 3
##    continent wb    country.name.en
##    <chr>     <chr> <chr>
##  1 Asia      AFG   Afghanistan
##  2 Europe    ALB   Albania
##  3 Africa    DZA   Algeria
##  4 Oceania   ASM   American Samoa
##  5 Europe    AND   Andorra
##  6 Africa    AGO   Angola
##  7 Americas  ATG   Antigua & Barbuda
##  8 Americas  ARG   Argentina
##  9 Asia      ARM   Armenia
## 10 Americas  ABW   Aruba
## # … with 208 more rows
```

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` .

```
# join both datasets
myeurope = myworld2 %>%
  filter(year == 2017) %>%    # only from 2017
  left_join((mycodes %>%
               filter(continent == "Europe") %>%    # only from Europe
               select(wb, country.name.en)), by = c("country" = "wb")) %>%    # join on country
name
  filter(country.name.en != "") %>%    # get rid of missing non-Europe values
  select(country.name.en, country, everything())    # reorder dataset

myeurope
```

```
## # A tibble: 46 × 7
##    country.name.en      country year  pop_urb_total pop_urb_grow pop_t…¹ pop_g…²
##    <chr>                <chr>   <chr>         <dbl>        <dbl>   <dbl>   <dbl>
##  1 Albania              ALB     2017        1706345        1.54   2.87e6 -0.0920
##  2 Andorra              AND     2017          67845       -0.520  7.70e4 -0.410
##  3 Austria              AUT     2017        5117624        1.15   8.81e6  0.827
##  4 Belgium              BEL     2017       11140192        0.401  1.14e7  0.358
##  5 Bulgaria             BGR     2017        5283572       -0.273  7.08e6 -0.730
##  6 Bosnia & Herzegovina BIH     2017        1679019        0.472  3.51e6 -0.279
##  7 Belarus              BLR     2017        7428883        0.674  9.51e6  0.0667
##  8 Switzerland          CHE     2017        6244619        1.13   8.47e6  1.10
##  9 Czechia              CZE     2017        7803157        0.379  1.06e7  0.236
## 10 Germany              DEU     2017       63890984        0.468  8.27e7  0.420
## # … with 36 more rows, and abbreviated variable names ¹pop_total, ²pop_growth
```

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

```
# pull row count from myeurope
myeurope %>%
  summarize(n())
```

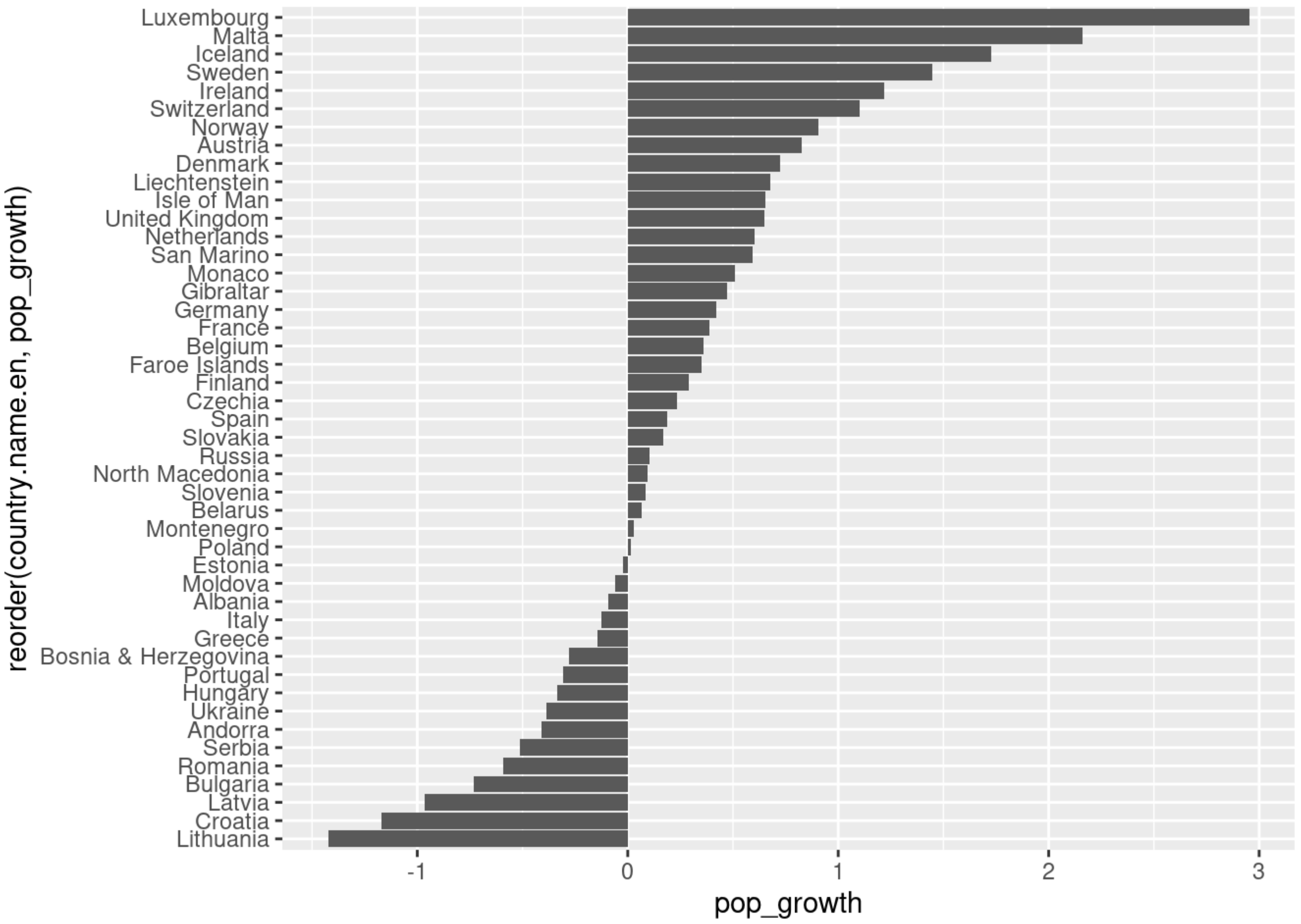
```
## # A tibble: 1 × 1
##   `n()`
##   <int>
## 1    46
```

There are 46 rows in myeurope, each row representing a 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?

```
# build population growth bar graph
myeurope %>%
  select(country.name.en, pop_growth) %>%
  arrange(pop_growth) %>%
  ggplot(aes(x = reorder(country.name.en, pop_growth), y = pop_growth)) +
  geom_bar(stat = "identity") +
  coord_flip()
```



Lithuania has the lowest population growth.

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")

mapWorld
```

```
##         long      lat group order      region subregion
## 1  -69.89912 12.45200     1     1       Aruba      <NA>
## 2  -69.89571 12.42300     1     2       Aruba      <NA>
## 3  -69.94219 12.43853     1     3       Aruba      <NA>
## 4  -70.00415 12.50049     1     4       Aruba      <NA>
## 5  -70.06612 12.54697     1     5       Aruba      <NA>
## 6  -70.05088 12.59707     1     6       Aruba      <NA>
## 7  -70.03511 12.61411     1     7       Aruba      <NA>
## 8  -69.97314 12.56763     1     8       Aruba      <NA>
## 9  -69.91181 12.48047     1     9       Aruba      <NA>
## 10 -69.89912 12.45200     1    10       Aruba      <NA>
## 12  74.89131 37.23164     2    12 Afghanistan      <NA>
## 13  74.84023 37.22505     2    13 Afghanistan      <NA>
## 14  74.76738 37.24917     2    14 Afghanistan      <NA>
## 15  74.73896 37.28564     2    15 Afghanistan      <NA>
## 16  74.72666 37.29072     2    16 Afghanistan      <NA>
## 17  74.66895 37.26670     2    17 Afghanistan      <NA>
##  [ reached 'max' / getOption("max.print") -- omitted 99322 rows ]
```

```
myeurope
```

```
## # A tibble: 46 × 7
##    country.name.en      country year  pop_urb_total pop_urb_grow pop_t…¹ pop_g…²
##    <chr>                <chr>   <chr>         <dbl>        <dbl>   <dbl>   <dbl>
##  1 Albania              ALB     2017        1706345        1.54   2.87e6 -0.0920
##  2 Andorra              AND     2017          67845       -0.520  7.70e4 -0.410
##  3 Austria              AUT     2017        5117624        1.15   8.81e6  0.827
##  4 Belgium              BEL     2017       11140192        0.401  1.14e7  0.358
##  5 Bulgaria             BGR     2017        5283572       -0.273  7.08e6 -0.730
##  6 Bosnia & Herzegovina BIH     2017        1679019        0.472  3.51e6 -0.279
##  7 Belarus              BLR     2017        7428883        0.674  9.51e6  0.0667
##  8 Switzerland          CHE     2017        6244619        1.13   8.47e6  1.10
##  9 Czechia              CZE     2017        7803157        0.379  1.06e7  0.236
## 10 Germany              DEU     2017       63890984        0.468  8.27e7  0.420
## # … with 36 more rows, and abbreviated variable names ¹pop_total, ²pop_growth
```

We could use region from mapWorld to join on country.name.en in 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.

```
# check where both datasets don't match
myeurope %>%
  anti_join(mapWorld, by = c("country.name.en" = "region"))
```

```
## # A tibble: 4 × 7
##   country.name.en      country year  pop_urb_total pop_urb_grow pop_to…¹ pop_g…²
##   <chr>                <chr>   <chr>         <dbl>        <dbl>    <dbl>   <dbl>
## 1 Bosnia & Herzegovina BIH     2017        1679019        0.472  3507017  -0.279
## 2 Czechia              CZE     2017        7803157        0.379 10591323   0.236
## 3 United Kingdom       GBR     2017       54892898        0.958 66022273   0.648
## 4 Gibraltar            GIB     2017          34571        0.473    34571   0.473
## # … with abbreviated variable names ¹pop_total, ²pop_growth
```

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?*

```
# sort mapWorld by aplhabetical region
mapWorld %>%
  group_by(region) %>%
  summarize(mean(group))
```

```
## # A tibble: 252 × 2
##    region         `mean(group)`
##    <chr>                  <dbl>
##  1 Afghanistan             2
##  2 Albania                 6
##  3 Algeria               486
##  4 American Samoa         24
##  5 Andorra                10
##  6 Angola                  3.12
##  7 Anguilla                5
##  8 Antarctica            100.
##  9 Antigua               137
## 10 Argentina              19.8
## # … with 242 more rows
```

It looks like the data from myeurope is older than mapWorld. Czechia in myeurope is now called Czech Republic, which is how it shows up in mapWorld. There's also small syntactic differences, like an ampersand instead of the word 'and'.

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/ (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`.

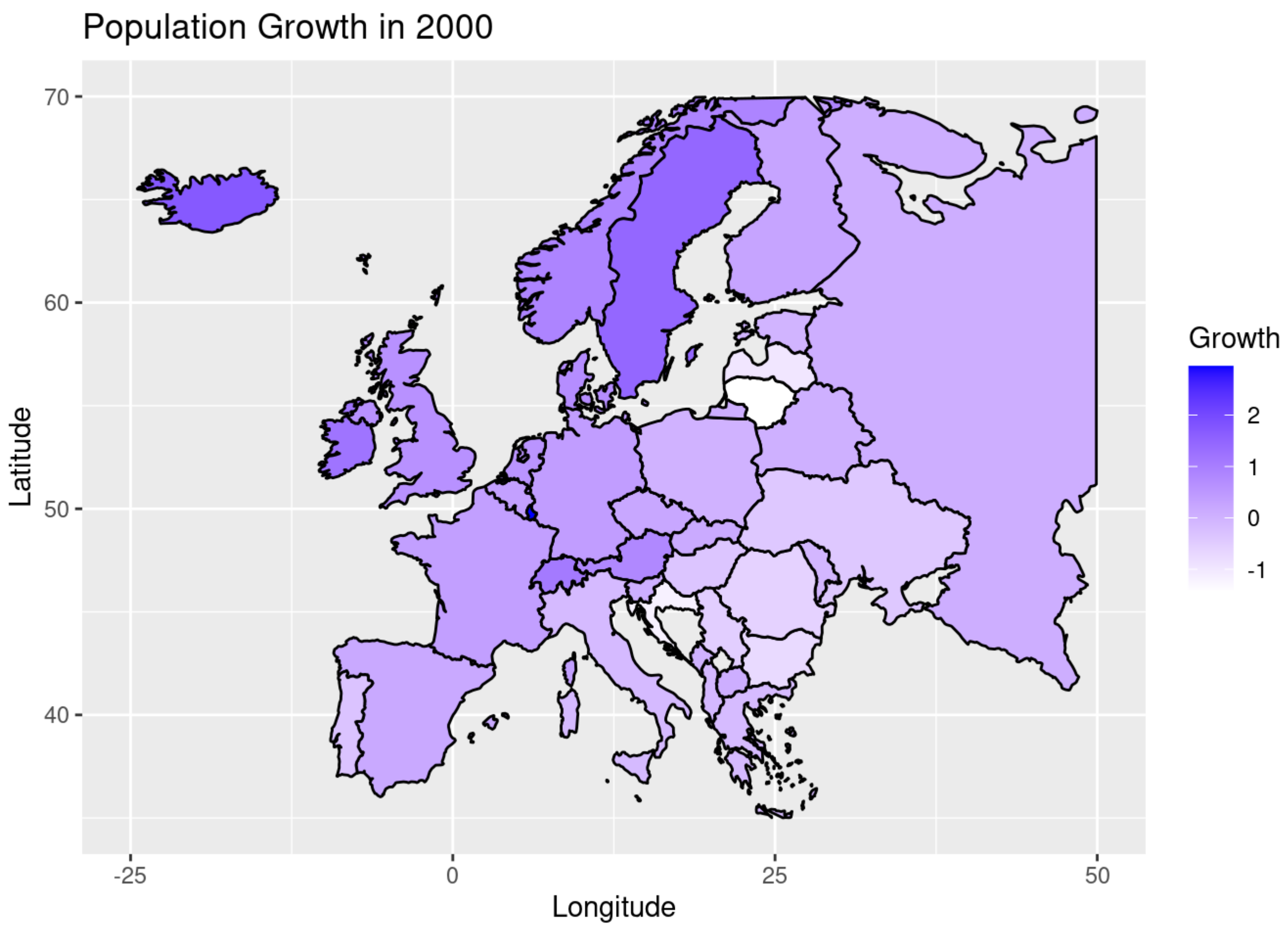
```
mymap = myeurope %>%
  # recode missing variables
  mutate(country.name.en=recode(country.name.en, "Bosnia & Herzegovina"="Boznia and Herzegovin
a", "Czechia"="Czech Republic", "United Kingdom"="UK")) %>%
  # combine datasets
  left_join(mapWorld, by = c("country.name.en" = "region"))

mymap %>% arrange(desc(pop_growth))
```

```
## # A tibble: 21,056 × 12
##    count…¹ country year  pop_u…² pop_u…³ pop_t…⁴ pop_g…⁵  long   lat group order
##    <chr>   <chr>   <chr>   <dbl>   <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <int>
##  1 Luxemb… LUX     2017   543862    3.25  599449    2.95  6.12  50.1   958 59705
##  2 Luxemb… LUX     2017   543862    3.25  599449    2.95  6.11  50.1   958 59706
##  3 Luxemb… LUX     2017   543862    3.25  599449    2.95  6.11  50.0   958 59707
##  4 Luxemb… LUX     2017   543862    3.25  599449    2.95  6.14  50.0   958 59708
##  5 Luxemb… LUX     2017   543862    3.25  599449    2.95  6.20  49.9   958 59709
##  6 Luxemb… LUX     2017   543862    3.25  599449    2.95  6.26  49.9   958 59710
##  7 Luxemb… LUX     2017   543862    3.25  599449    2.95  6.32  49.8   958 59711
##  8 Luxemb… LUX     2017   543862    3.25  599449    2.95  6.44  49.8   958 59712
##  9 Luxemb… LUX     2017   543862    3.25  599449    2.95  6.49  49.8   958 59713
## 10 Luxemb… LUX     2017   543862    3.25  599449    2.95  6.49  49.8   958 59714
## # … with 21,046 more rows, 1 more variable: subregion <chr>, and abbreviated
## #   variable names ¹country.name.en, ²pop_urb_total, ³pop_urb_grow, ⁴pop_total,
## #   ⁵pop_growth
```

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.*

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

# Call the ggmap package
library(ggmap)

# Build a map!
mymap %>%
  # assign data from mymap to ggplot
  ggplot(aes(x = long, y = lat, group = group, fill = pop_growth)) +
  # draw countries
  geom_polygon(colour = "black") +
  # assign fill color based on population growth
  scale_fill_gradient(low = "white", high = "blue") +
  # add title and axes labels
  labs(fill = "Growth" ,title = "Population Growth in 2000",
       x ="Longitude", y ="Latitude") +
  # limit spn of graph to make it look prettier
  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/ (https://www.wpmap.org/europe-map-hd-with-countries/)*

Luxembourg had the highest European population growth in 2017!

Formatting: (2 pts)

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

```
##                                                 sysname 
##                                                 "Linux" 
##                                                 release 
##                                     "5.15.0-58-generic" 
##                                                 version 
## "#64~20.04.1-Ubuntu SMP Fri Jan 6 16:42:31 UTC 2023" 
##                                                nodename 
##                            "educcomp01.ccbb.utexas.edu" 
##                                                 machine 
##                                                "x86_64" 
##                                                   login 
##                                               "unknown" 
##                                                    user 
##                                               "jlh7459" 
##                                          effective_user 
##                                               "jlh7459" 
```