HW 6

SDS322E

October 07, 2022

Ethan Chang - ehc586

Please submit as a PDF or HTML file on Canvas before the due date.

For all questions, include the R commands/functions that you used to find your answer. Answers without supporting code will not receive credit.

> Review of how to submit this assignment
>
> All homework assignments will be completed using R Markdown. These `.Rmd` files consist of >text/syntax (formatted using Markdown) alongside embedded R code. When you have completed the assignment (by adding R code inside codeblocks and supporting text outside of the codeblocks), create your document as follows (assuming you are using the edupod server and submitting HTML):

- Click the arrow next to the “Knit” button (above)
- Choose “Knit to HTML”
- Go to Files pane and put checkmark next to the correct HTML file
- Click on the blue gear icon (“More”) and click Export
- Download the file and then upload to Canvas
- To submit a PDF, open your HTML file and print it to a pdf, then upload the pdf as your submission.

Question 1 (0.5 pts)

The dataset `world_bank_pop` is a built-in dataset in `tidyverse`. It contains information about total population, population growth, and urban population for countries around the world.

Save the dataset in your environment as `myworld`. Take a look at it with `head()`. **Is the data tidy? Why or why not?**

```
# Your code goes here
myworld <- world_bank_pop
head(myworld)
```

```
## # 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
## # ℹ Use `colnames()` to see all variable names
```

Answer: The data is not tidy as it’s not really organized well, it has multiple columns for years and multiple value points that could be condensed to produce fewer columns and clearer labellings.

Question 2 (0.5 pts)

Using `dplyr` functions, count how many distinct countries there are in the dataset.

```
# Your Code here
myworld %>%
    group_by(country) %>%
    summarize(n = n()) %>%
    pull(country) %>%
    length()
```

```
## [1] 264
```

Answer: There are 264 distinct countries in the dataset.

Question 3: (1 pts)

Use one of the `pivot_` functions to create a new dataset, `myworld2` , with the years 2000 to 2017 appearing as a *numeric* variable `year` , and with the different values for the indicator variables displayed in a variable called `value` . **In this new dataset, how many lines are there per country? Why does this make sense?**

```
# your code goes below this line
myworld2 <- myworld %>%
    pivot_longer(cols = "2000":"2017", names_to = "year",
        values_to = "value") %>%
    mutate(year = as.numeric(year))
myworld2
```

```
## # A tibble: 19,008 × 4
##    country indicator    year value
##    <chr>   <chr>       <dbl> <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
## # ℹ Use `print(n = ...)` to see more rows
```

```
myworld2 %>%
    group_by(country) %>%
    summarize(n = n())
```

```
## # A tibble: 264 × 2
##    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
## # ℹ Use `print(n = ...)` to see more rows
```

Answer: There are 72 lines per country in this new dataset. This makes sense because there are 18 unique years and 4 unique indicators, so there should be 18*4 = 72 unique values/lines per country to account for every single combination of year and indicator.

Question 4 (1 pts)

Use another `pivot` function on `myworld2` to create a new dataset `myworld3` with the different categories for the indicators 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 (for example, you might use `rename`). **On this new dataset, use `dplyr` functions to find which country code had the highest population growth in 2017.**

```
# your code goes below this line
myworld3 <- myworld2 %>%
    pivot_wider(names_from = indicator) %>%
    rename(pop_growth = SP.POP.GROW, pop_urb_growth = SP.URB.GROW)
myworld3
```

```
## # A tibble: 4,752 × 6
##    country  year SP.URB.TOTL pop_urb_growth SP.POP.TOTL pop_growth
##    <chr>   <dbl>       <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
## # ℹ Use `print(n = ...)` to see more rows
```

```
myworld3 %>%
    filter(year == 2017) %>%
    slice_max(pop_growth, n = 1)
```

```
## # A tibble: 1 × 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
```

Answer: OMN had the highest population growth (pop_growth) in 2017.

Question 5 (1 pts)

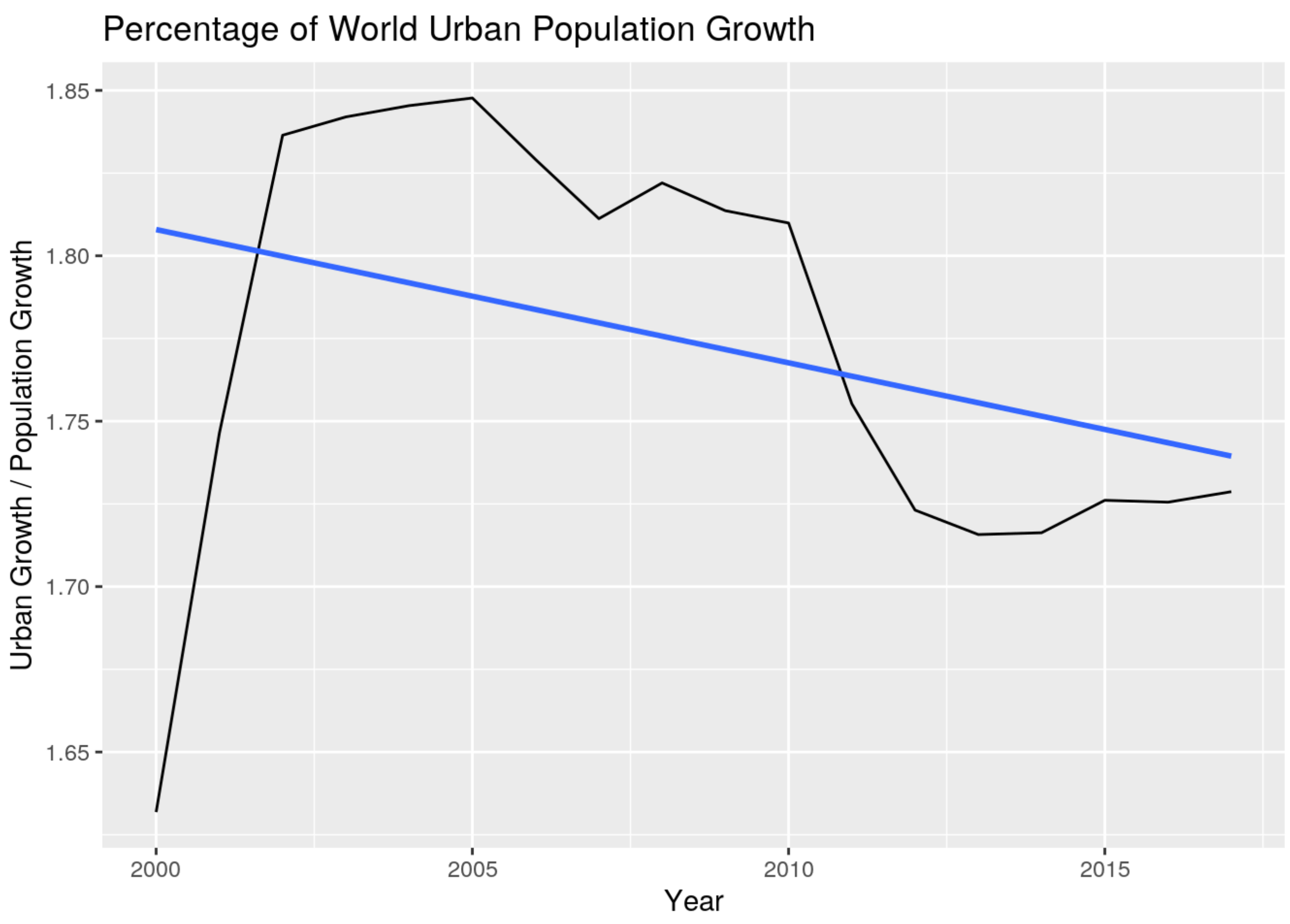
Using `dplyr` functions, find the ratio of urban growth compared to the population growth in the world for each year (*Hint: the country code `WLD` represents the entire world*). Using a visualization, describe how the percentage of urban population growth has changed over the years. **Why does your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

```
# your code goes below this line
library(ggplot2)

wld_pop_ratio <- myworld3 %>%
    filter(country == "WLD") %>%
    mutate(ratio = pop_urb_growth/pop_growth)
wld_pop_ratio
```

```
## # A tibble: 18 × 7
##    country  year SP.URB.TOTL pop_urb_growth SP.POP.TOTL pop_growth ratio
##    <chr>   <dbl>       <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
```

```
ggplot(wld_pop_ratio, aes(x = year, y = ratio)) + geom_line() +
    geom_smooth(method = "lm", se = FALSE) + xlab("Year") +
    ylab("Urban Growth / Population Growth") + ggtitle("Percentage of World Urban Populati
on Growth")
```



Answer: Over the years, it can be seen that the percentage of urban population growth started off by increasing tremendously at 2000, and since then has started decreasing at a relatively slow rate. Despite this, the percentage has still managed to stay above 100% throughout the years. This means that while the percentage of urban population growth may be decreasing in recent years, urban population worldwide is still increasing throughout the years, just by a smaller amount proportionally when compared to population growth.

Question 6: (1 pts)

In answering question 4, we did not find out what country is represented by the three-letter code. We will now use a package that has information about the coding system used by the world bank. We will use the dataset `codelist` contained in the package `countrycode`. Run the following code to save the dataset to `mycodes`:

```
library(countrycode)
mycodes <- codelist
```

Using `dplyr` functions, modify `mycodes` as follows:

1. select only the variables `continent` , `wb` (World Bank code), and `country.name.en` (country name in English);
2. filter to keep countries in Europe only;
3. remove countries with missing `wb` code.

On this new dataset, use `dplyr` to count how many countries there are in Europe with a World Bank code.

```
# your code goes below this line
mycodes <- mycodes %>%
    select(continent, wb, country.name.en) %>%
    filter(continent == "Europe") %>%
    filter(!is.na(wb))
mycodes
```

```
## # A tibble: 46 × 3
##    continent wb    country.name.en
##    <chr>     <chr> <chr>
##  1 Europe    ALB   Albania
##  2 Europe    AND   Andorra
##  3 Europe    AUT   Austria
##  4 Europe    BLR   Belarus
##  5 Europe    BEL   Belgium
##  6 Europe    BIH   Bosnia & Herzegovina
##  7 Europe    BGR   Bulgaria
##  8 Europe    HRV   Croatia
##  9 Europe    CZE   Czechia
## 10 Europe    DNK   Denmark
## # … with 36 more rows
## # ℹ Use `print(n = ...)` to see more rows
```

```
mycodes %>%
    count()
```

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

Answer: There are 46 countries in Europe with a World Bank code

Question 7 (1 pts)

Use a `left_join()` function to create a new dataset, `myeurope` , to add data to the countries in `mycodes` dataset from `myworld3` dataset. Match the two datasets based on the World Bank code. Using `dplyr` functions, change the name of the variable containing the World Bank code to `country` . **How many rows are there in this new dataset? Why does it make sense?**

```
# your code goes below this line
mycodes <- mycodes %>%
    rename(country = wb)
myeurope <- left_join(mycodes, myworld3)
myeurope
```

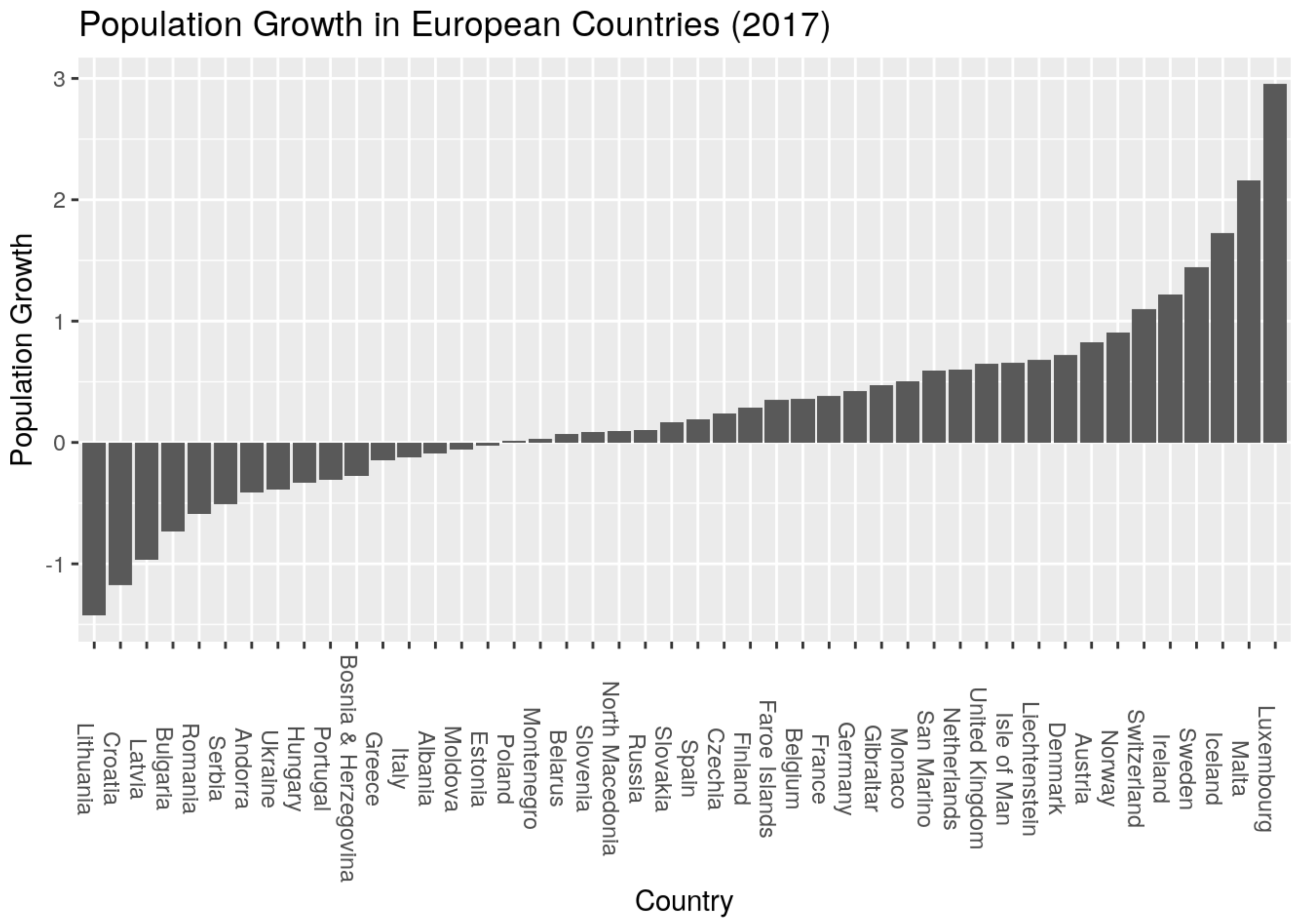
```
## # A tibble: 828 × 8
##    continent country country.name.en  year SP.URB.TOTL pop_urb…¹ SP.PO…² pop_g…³
##    <chr>     <chr>   <chr>           <dbl>       <dbl>     <dbl>   <dbl>   <dbl>
##  1 Europe    ALB     Albania          2000     1289391     0.742 3089027  -0.637
##  2 Europe    ALB     Albania          2001     1298584     0.710 3060173  -0.938
##  3 Europe    ALB     Albania          2002     1327220     2.18  3051010  -0.300
##  4 Europe    ALB     Albania          2003     1354848     2.06  3039616  -0.374
##  5 Europe    ALB     Albania          2004     1381828     1.97  3026939  -0.418
##  6 Europe    ALB     Albania          2005     1407298     1.83  3011487  -0.512
##  7 Europe    ALB     Albania          2006     1430886     1.66  2992547  -0.631
##  8 Europe    ALB     Albania          2007     1452398     1.49  2970017  -0.756
##  9 Europe    ALB     Albania          2008     1473392     1.44  2947314  -0.767
## 10 Europe    ALB     Albania          2009     1495260     1.47  2927519  -0.674
## # … with 818 more rows, and abbreviated variable names ¹pop_urb_growth,
## #   ²SP.POP.TOTL, ³pop_growth
## # ℹ Use `print(n = ...)` to see more rows
```

Answer: There are 828 rows in this dataset. This makes sense as there were 46 distinct countries in Europe, so in order for each of these countries to contain a value for each year, 18, there would have to be 46 * 18 = 828 rows.

Question 8 (1 pts)

Using `dplyr` functions, 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 highest population growth in 2017?**

```
# your code goes below this line
ggplot(myeurope %>%
    filter(year == 2017), aes(x = reorder(country.name.en,
    +pop_growth), y = pop_growth)) + geom_bar(stat = "identity") +
    xlab("Country") + ylab("Population Growth") + ggtitle("Population Growth in European C
ountries (2017)") +
    theme(axis.text.x = element_text(angle = -90))
```



Answer: The country Luxembourg (LUX) had the highest population growth in Europe in 2017.

Question 9 (0.5 pts)

When dealing with location data, we can actually visualize information on a map if we have geographic information such as latitude and longitude.

Let's 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` with `glimpse()`. **What variable could we use to join this dataset with `myeurope` dataset?**

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

```
# your code goes below this line
glimpse(mapWorld)
```

```
## Rows: 99,338
## Columns: 6
## $ long      <dbl> -69.89912, -69.89571, -69.94219, -70.00415, -70.06612, -70.0…
## $ lat       <dbl> 12.45200, 12.42300, 12.43853, 12.50049, 12.54697, 12.59707, …
## $ group     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, …
## $ order     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 1…
## $ region    <chr> "Aruba", "Aruba", "Aruba", "Aruba", "Aruba", "Aruba", "Aruba…
## $ subregion <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …
```

Answer: Based on the variables available in this dataset, it looks like `region` would be the best variable to join this dataset with `myeurope`.

Question 10 (1 pts)

Only keep the year 2017 in the dataset `myeurope`. Then use a `left_join()` to add data to the countries in `myeurope` dataset from `mapWorld` dataset, matching the two datasets based on the country name. If we then use `dplyr` functions, we can identify some missing values for `lat` and `long` in the new dataset. Indeed, some countries such as United Kingdom did not have a match. **Why do you think this happened?**

```
# your code goes below this line
mapWorld <- mapWorld %>%
    rename(country.name.en = region)
myeuropemapWorld <- left_join(myeurope %>%
    filter(year == 2017), mapWorld)
myeuropemapWorld
```

```
## # A tibble: 19,828 × 13
##    continent country country…¹  year SP.UR…² pop_u…³ SP.PO…⁴ pop_g…⁵  long   lat
##    <chr>     <chr>   <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl> <dbl>
##  1 Europe    ALB     Albania    2017 1706345    1.54 2873457 -0.0920  20.1  42.5
##  2 Europe    ALB     Albania    2017 1706345    1.54 2873457 -0.0920  20.1  42.5
##  3 Europe    ALB     Albania    2017 1706345    1.54 2873457 -0.0920  20.2  42.4
##  4 Europe    ALB     Albania    2017 1706345    1.54 2873457 -0.0920  20.2  42.3
##  5 Europe    ALB     Albania    2017 1706345    1.54 2873457 -0.0920  20.3  42.3
##  6 Europe    ALB     Albania    2017 1706345    1.54 2873457 -0.0920  20.4  42.3
##  7 Europe    ALB     Albania    2017 1706345    1.54 2873457 -0.0920  20.5  42.2
##  8 Europe    ALB     Albania    2017 1706345    1.54 2873457 -0.0920  20.5  42.2
##  9 Europe    ALB     Albania    2017 1706345    1.54 2873457 -0.0920  20.6  42.0
## 10 Europe    ALB     Albania    2017 1706345    1.54 2873457 -0.0920  20.6  41.9
## # … with 19,818 more rows, 3 more variables: group <dbl>, order <int>,
## #   subregion <chr>, and abbreviated variable names ¹country.name.en,
## #   ²SP.URB.TOTL, ³pop_urb_growth, ⁴SP.POP.TOTL, ⁵pop_growth
## # ℹ Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

```
myeuropemapWorld %>%
    filter(is.na(lat))
```

```
## # A tibble: 4 × 13
##   continent country country.…¹  year SP.UR…² pop_u…³ SP.PO…⁴ pop_g…⁵  long   lat
##   <chr>     <chr>   <chr>      <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl> <dbl>
## 1 Europe    BIH     Bosnia & …  2017  1.68e6   0.472  3.51e6  -0.279    NA    NA
## 2 Europe    CZE     Czechia     2017  7.80e6   0.379  1.06e7   0.236    NA    NA
## 3 Europe    GIB     Gibraltar   2017  3.46e4   0.473  3.46e4   0.473    NA    NA
## 4 Europe    GBR     United Ki…  2017  5.49e7   0.958  6.60e7   0.648    NA    NA
## # … with 3 more variables: group <dbl>, order <int>, subregion <chr>, and
## #   abbreviated variable names ¹country.name.en, ²SP.URB.TOTL, ³pop_urb_growth,
## #   ⁴SP.POP.TOTL, ⁵pop_growth
## # ℹ Use `colnames()` to see all variable names
```

Answer: These specific European countries likely do not have matching data in the mapWorld dataset as they were either named differently between both data sets (United Kingdom and UK), or there just wasn't any data about it in the the mapWorld dataset in the first place.

Question 11 (0.5 pts)

To identify all countries in 2017 that did not have an exact match, do an `anti_join()` instead of `left_join()` in the previous question. **How many countries did not have an exact match?** *Note: using `anti_join()` is a very useful function to identify differences between datasets.*

```
# your code goes below this line
anti_join(myeurope %>%
    filter(year == 2017), mapWorld)
```

```
## # A tibble: 4 × 8
##   continent country country.name.en       year SP.URB.…¹ pop_u…² SP.PO…³ pop_g…⁴
##   <chr>     <chr>   <chr>                <dbl>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 Europe    BIH     Bosnia & Herzegovina  2017   1679019   0.472  3.51e6  -0.279
## 2 Europe    CZE     Czechia               2017   7803157   0.379  1.06e7   0.236
## 3 Europe    GIB     Gibraltar             2017     34571   0.473  3.46e4   0.473
## 4 Europe    GBR     United Kingdom        2017  54892898   0.958  6.60e7   0.648
## # … with abbreviated variable names ¹SP.URB.TOTL, ²pop_urb_growth,
## #   ³SP.POP.TOTL, ⁴pop_growth
```

Answer: From the table above, it can be seen that 4 countries did not have an exact match.

Question 12 (0.5 pts)

Joining datasets by variables containing names often leads to a mismatch because spelling can vary from one dataset to another. Sometimes we need to manually fix spelling in order to be able to match values. Consider the code given below. Replace the name of United Kingdom so that its name in `myeurope` dataset corresponds to the name given in `mapWorld` dataset. **Following this code, add a pipe and use a `left_join()` function to create the new dataset, `mymap`, adding data to the countries in `myeurope` dataset from `mapWorld` dataset.**

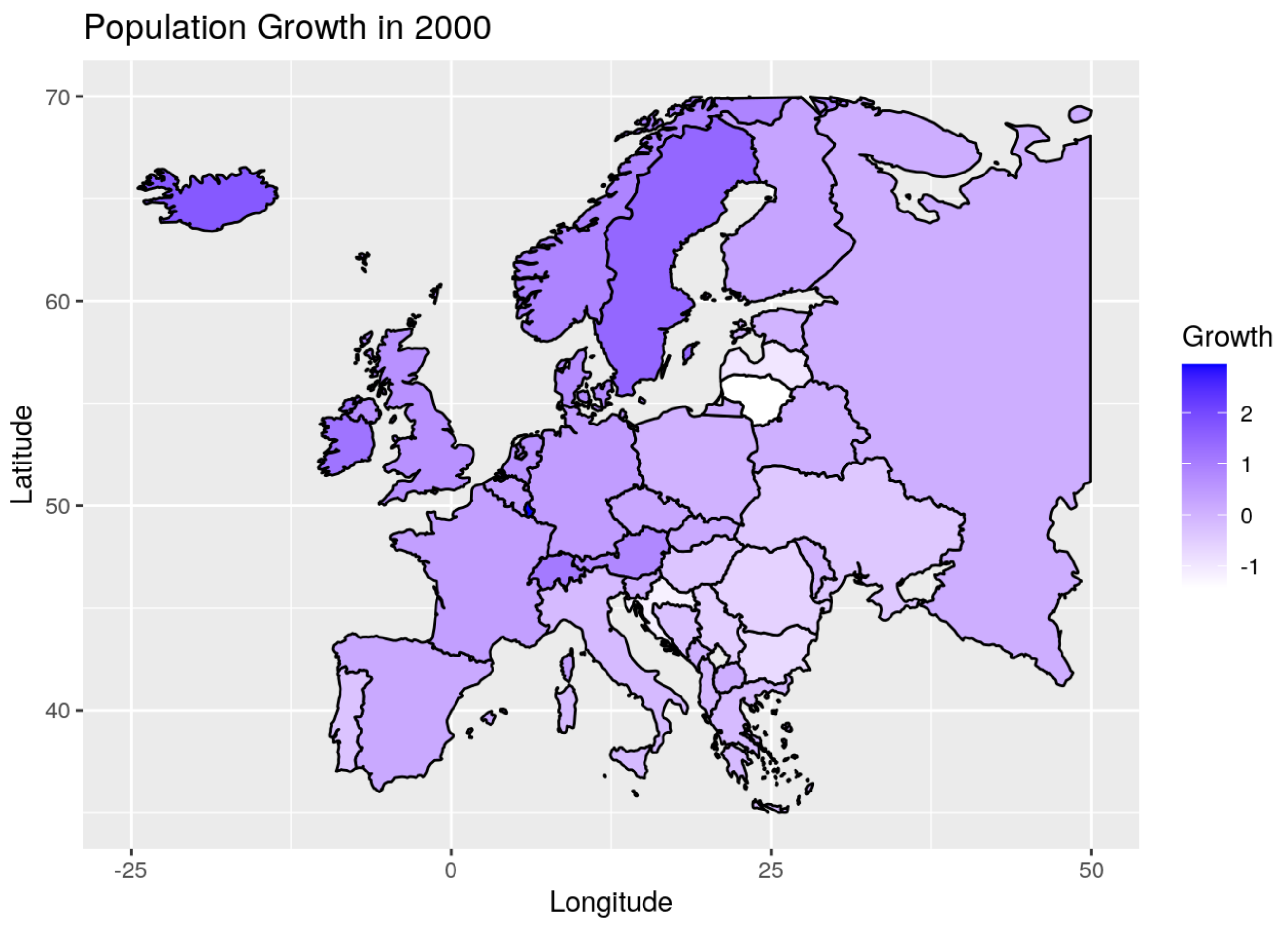
```
# Remove `eval = FALSE` to run the code
mapWorld <- mapWorld %>% rename(country_clean = country.name.en)
mymap <- myeurope %>%
  filter(year == 2017) %>%
  mutate(country_clean = recode(country.name.en,
                              'United Kingdom' = 'UK', # Modify code HERE
                              'Bosnia & Herzegovina' = 'Bosnia and Herzegovina',
                              'Czechia' = 'Czech Republic')) %>% left_join(mapWorld)
```

Question 13 (0.5 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")
# When you are ready to run the code, remove `eval = FALSE` in the markdown
# Call the ggmap package
library(ggmap)

mymap %>%
  # Sets up the conditions of a figure to be plotted, setting longitude as x, latitude as
 y, grouping all the data by the group, and filling in colors of the shapes based on the p
opulation growth
  ggplot(aes(x = long, y = lat, group = group, fill = pop_growth)) +
  # Plots the shapes produced by the mymap dataset given the conditions specified in the g
gplot, setting the color of the outline of the produced shapes as black
  geom_polygon(colour = "black") +
  # Changes the fill color of the polygons to be a gradient between white and blue (ultima
tely producing a purple/violet hue) based on its population growth, and setting the legend
for it as a continuous colorbar.
  scale_fill_gradient(low = "white", high = "blue",
                      guide = "colorbar") +
  # Labels the fill legend as "Growth", the title as "Population Growth in 2000", the x-ax
is as "Longitude", and the y-axis as "Latitude"
  labs(fill = "Growth" ,
       title = "Population Growth in 2000",
       x = "Longitude", y = "Latitude") +
  # Sets limits on the x and y axes so that the longitude (x-axis) is between -25 and 50,
 and the latitude (y-axis) is between 35 and 70.
  xlim(-25,50) + ylim(35,70)
```



```
##                                                 sysname
##                                                 "Linux"
##                                                 release
##                                     "4.15.0-193-generic"
##                                                 version
## "#204-Ubuntu SMP Fri Aug 26 19:20:21 UTC 2022"
##                                                nodename
##                             "educcomp02.ccbb.utexas.edu"
##                                                 machine
##                                                "x86_64"
##                                                   login
##                                               "unknown"
##                                                    user
##                                                "ehc586"
##                                          effective_user
##                                                "ehc586"
```