# HW 4

SDS348 Spring 2021

# Name: Santhosh Saravanan

# EID: sks3648

**This homework is due on Mar 8, 2021 at 8am. Submit 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: (9 pts)

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.

1.1 (1 pt) Save the dataset `world_bank_pop` as `myworld` and take a look at it with `head()`. Is the data tidy? Why or why not?

```
library(tidyverse)
myworld <- world_bank_pop
head(myworld)
```

```
## # 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.TO…  4.24e4  4.30e4  4.37e4  4.42e4  4.47e+4  4.49e+4   4.49e+4
## 2 ABW     SP.URB.GR…  1.18e0  1.41e0  1.43e0  1.31e0  9.51e-1  4.91e-1  -1.78e-2
## 3 ABW     SP.POP.TO…  9.09e4  9.29e4  9.50e4  9.70e4  9.87e+4  1.00e+5   1.01e+5
## 4 ABW     SP.POP.GR…  2.06e0  2.23e0  2.23e0  2.11e0  1.76e+0  1.30e+0   7.98e-1
## 5 AFG     SP.URB.TO…  4.44e6  4.65e6  4.89e6  5.16e6  5.43e+6  5.69e+6   5.93e+6
## 6 AFG     SP.URB.GR…  3.91e0  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>
```

*Each country abbreviation forms a row with information stemming from the indicator and year. Each variable (indicator, year, and country code) all form a distinct column. Hence, the data is tidy.*

1.2 (1 pt) Using pipes and `dplyr` functions, how many countries are there in the dataset?

```
# your code goes here (make sure to add comments)
myworld %>%
  summarise(count=n_distinct(country))
```

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

```
#first we group by country and use count parameter in summarise dplyr method
```

*There are 264 countries in the dataset.*

1.3 (2 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 the different values for the indicator variable are in a variable called **`value`**. In this new dataset, how many lines are there per country? Why does it make sense?

```
# your code goes here (make sure to add comments)
myworld2 <- myworld %>% pivot_longer(-c(country,indicator), names_to = "year", names_
transform = list(year= as.integer))
head(myworld2)
```

```
## # A tibble: 6 x 4
##   country indicator    year value
##   <chr>   <chr>       <int> <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
```

```
myworld2%>%count(country,indicator,sort = TRUE) # find the number of times a country
is used for every type of indicator
```

```
## # A tibble: 1,056 x 3
##    country indicator       n
##    <chr>   <chr>       <int>
##  1 ABW     SP.POP.GROW    18
##  2 ABW     SP.POP.TOTL    18
##  3 ABW     SP.URB.GROW    18
##  4 ABW     SP.URB.TOTL    18
##  5 AFG     SP.POP.GROW    18
##  6 AFG     SP.POP.TOTL    18
##  7 AFG     SP.URB.GROW    18
##  8 AFG     SP.URB.TOTL    18
##  9 AGO     SP.POP.GROW    18
## 10 AGO     SP.POP.TOTL    18
## # … with 1,046 more rows
```

```
myworld2%>%count(country,sort = TRUE) # just to confirm that the total number of rows
```

```
## # A tibble: 264 x 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
```

```
# for every country is 72.
```

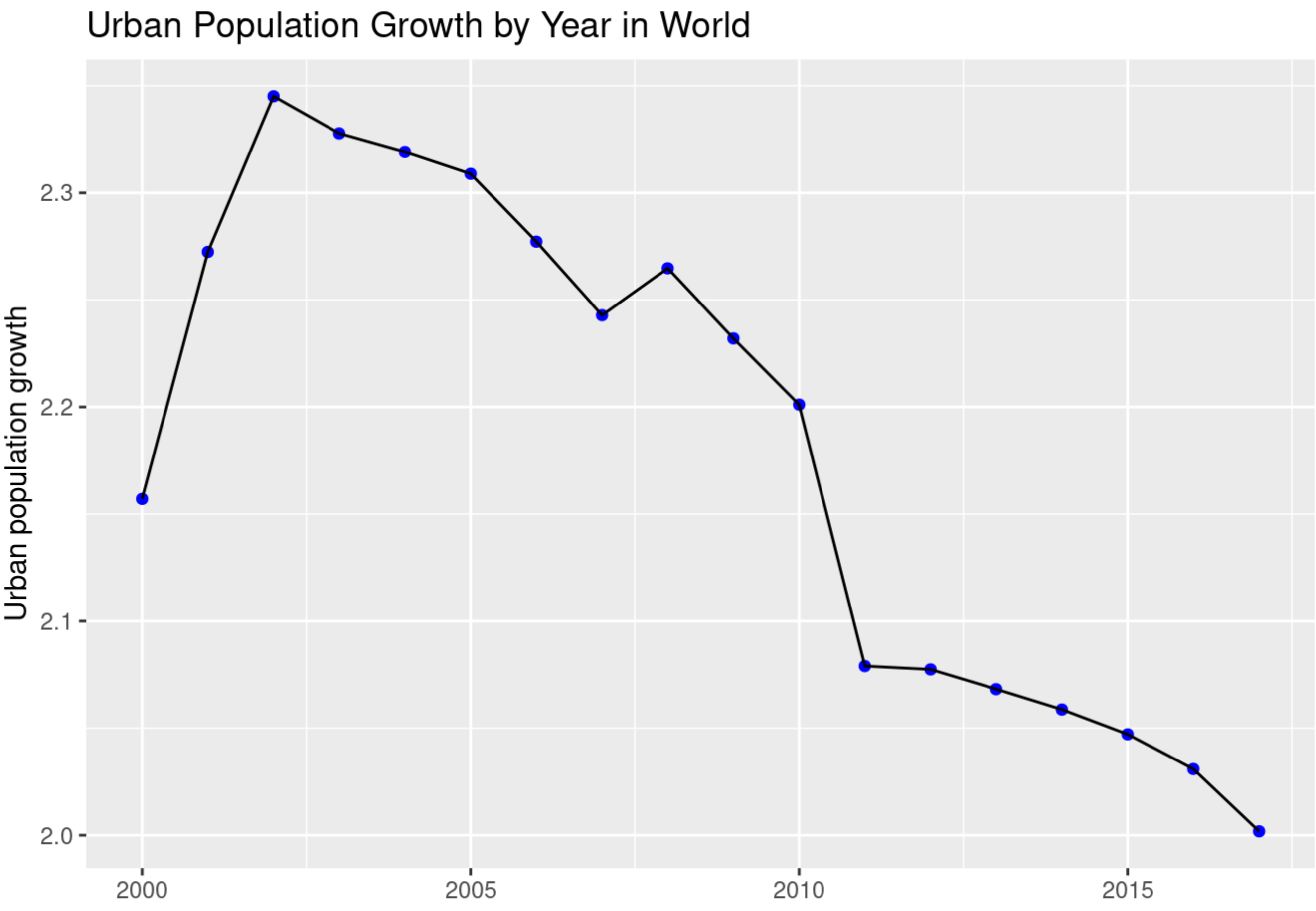
*There are about 72 lines per country. This makes sense because there are 4 indicators: population growth, total population, urban population growth, and total urban population. Given that the years range from 2000 to 2017, that’s about 17 years. 18 years per indicator gives about 72 measurements per country, so it makes sense.*

### 1.4 (3 pts) Represent the total population growth and urban population growth in the world (country code is `WLD` ) between 2000 and 2017. How has population growth changed over the years?

```
# your code goes here (make sure to add comments)
myworld2%>% filter(country=="WLD" & indicator == "SP.URB.GROW")
```

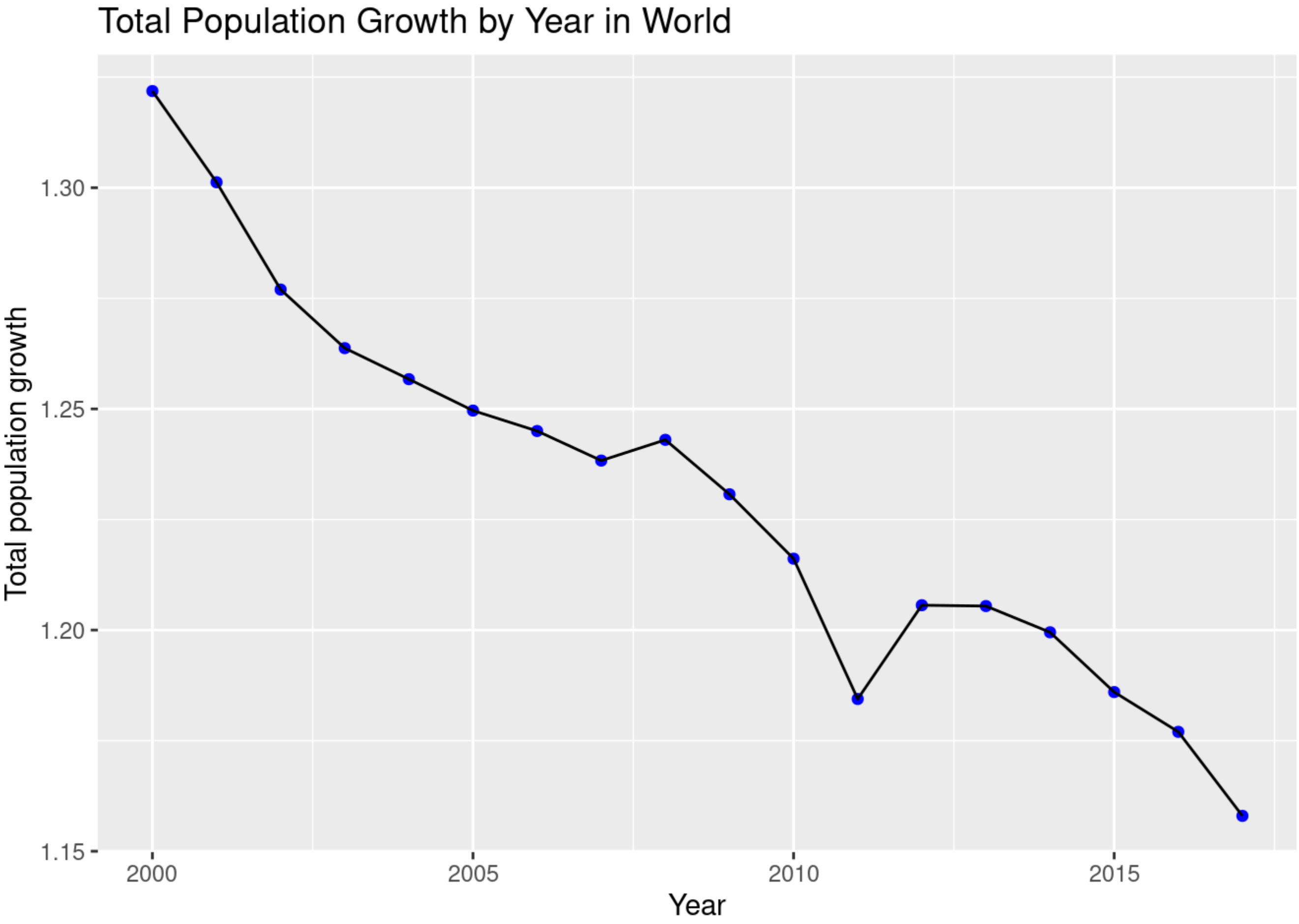
```
## # A tibble: 18 x 4
##    country indicator    year value
##    <chr>   <chr>       <int> <dbl>
##  1 WLD     SP.URB.GROW  2000  2.16
##  2 WLD     SP.URB.GROW  2001  2.27
##  3 WLD     SP.URB.GROW  2002  2.35
##  4 WLD     SP.URB.GROW  2003  2.33
##  5 WLD     SP.URB.GROW  2004  2.32
##  6 WLD     SP.URB.GROW  2005  2.31
##  7 WLD     SP.URB.GROW  2006  2.28
##  8 WLD     SP.URB.GROW  2007  2.24
##  9 WLD     SP.URB.GROW  2008  2.26
## 10 WLD     SP.URB.GROW  2009  2.23
## 11 WLD     SP.URB.GROW  2010  2.20
## 12 WLD     SP.URB.GROW  2011  2.08
## 13 WLD     SP.URB.GROW  2012  2.08
## 14 WLD     SP.URB.GROW  2013  2.07
## 15 WLD     SP.URB.GROW  2014  2.06
## 16 WLD     SP.URB.GROW  2015  2.05
## 17 WLD     SP.URB.GROW  2016  2.03
## 18 WLD     SP.URB.GROW  2017  2.00
```

```
ggplot(myworld2%>% filter(country=="WLD" & indicator == "SP.URB.GROW"),aes(x= `year`,
y= `value`)) + geom_point(size=1.5,color="Blue") + geom_line() + labs(y="Urban popula
tion growth",x="Year") + ggtitle("Urban Population Growth by Year in World")
```



Year

```
ggplot(myworld2%>% filter(country=="WLD" & indicator == "SP.POP.GROW"),aes(x= `year`,
y= `value`)) + geom_point(size=1.5,color="Blue") + geom_line() + labs(y="Total popula
tion growth",x="Year") + ggtitle("Total Population Growth by Year in World")
```



*Urban population growth reached its peak above 2.3 at around 2002-2003 and has declined ever since until past 2005. After 2005, there was a momentary spike upward at around 2007-2007 but has declined until experiencing a sharp decline after 2010. Ever since, the rate of decrease is still present, but a lot slower after 2011. For total population growth, it's been steadily decreasing till a a short while after 2010. There was an all-time low at the time at 2011, but it steadily gre back and has similarily, the rate of decrease is still present, but it's a bit higher than the urban population growth after 2011.*

1.5 (2 pts) Use one of the `pivot` functions to create a new dataset, `myworld3`, 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. What is the country code that had the highest population growth in 2017?

```
myworld3 <- myworld2 %>% pivot_wider( names_from= indicator,values_from= value)
myworld3 <- myworld3 %>% rename(pop_growth = SP.POP.GROW, pop_urb_growth = SP.URB.GRO
W)
myworld3 %>% filter(year==2017) %>% arrange(desc(pop_growth))
```

```
## # A tibble: 264 x 6
##    country  year SP.URB.TOTL pop_urb_growth SP.POP.TOTL pop_growth
##    <chr>   <int>       <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
```

*OMN has the highest population growth in 2017.*

# Question 2: (10 pts)

From answering the previous question, we have no idea what actual countries are represented by the codes. We will now use a package that has information about the coding system used by the World bank.

2.1 (2 pts) Install the package `countrycode` . We will use a built-in dataset called `codelist` . Make sure to upload the library and save this dataset as `mycodes` . Using `dplyr` functions, modify `mycodes` to: 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. How many countries are there in Europe with a World Bank code?

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

library(countrycode)

mycodes <- codelist
mycodes <- mycodes %>% select(continent, wb, country.name.en) %>% filter(continent ==
"Europe") %>% drop_na(wb) # select countries based on continent, wb, and country name
in English
# filter with continent == "Europe" and drop_na values in wb column
nrow(mycodes)
```

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

```
# your code goes here (make sure to add comments)
```

*There are 46 countries in Europe with a World Bank Code.*

2.2 (2 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`.

```
# your code goes here (make sure to add comments)
mycodes <- rename(mycodes,c("country" = "wb"))
myeurope <- mycodes %>%
left_join(myworld3, by= c("country"))
```

2.3 (1 pt) Using `dplyr` functions, what was the total population in European countries in 2017? Give your answer in million (round to the next million).

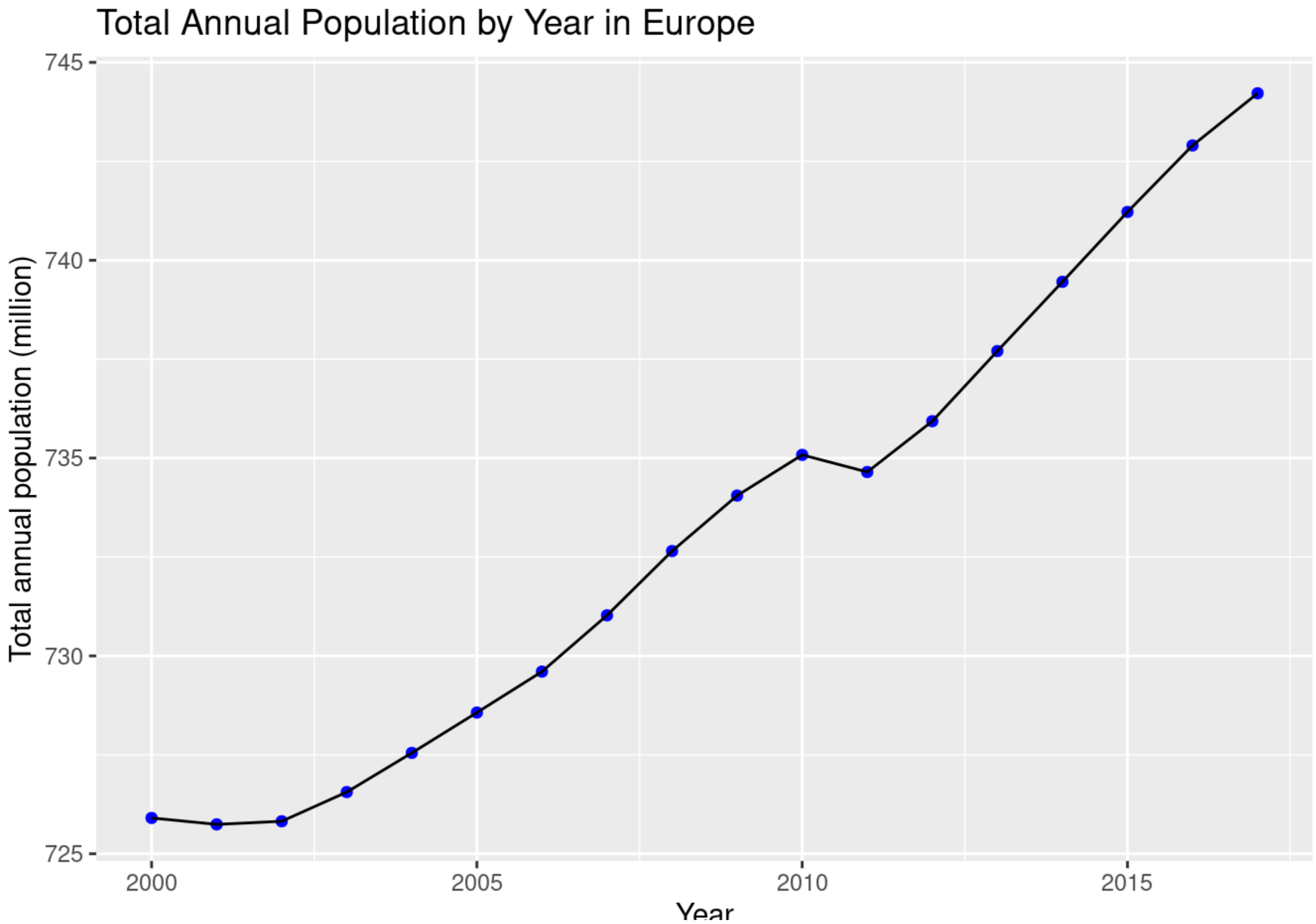
```
  myeurope1 <- myeurope %>% filter(continent == 'Europe' & year == 2017)
  sum(myeurope1$SP.POP.TOTL)
```

```
## [1] 744218594
```

*There are 744 million people in Europe in 2017.*

2.4 (2 pts) Represent the annual total population in European countries between 2000 and 2017. Express the total population in million. How has European population changed over the years?

```
annualNumbers = myeurope %>% group_by(year) %>% summarise(totalPop=sum(SP.POP.TOTL)/1000000)
ggplot(annualNumbers,aes(x= `year`, y=totalPop )) + geom_point(size=1.5,color="Blue") + geom_line() + labs(y="Total annual population (million)",x="Year") + ggtitle("Total Annual Population by Year in Europe")
```



*Population seems to be strictly increasing except for 2011, where there was a slight dip. After 2011, it seems to continue to increase.*

### 2.5 (2 pts) Create a new dataset `myeurope2017` by filtering the data for the `year` 2017, dropping the variable `year` , and creating a new variable `prop_urb` which is the proportion of urban population for each country. Which European country had the lowest proportion of urban population in 2017?

```
myeurope2017 <- myeurope %>%filter(year == 2017)%>% select(-year) %>% mutate(prop_urb
= SP.URB.TOTL/SP.POP.TOTL) %>% arrange(prop_urb)
myeurope2017
```

```
## # A tibble: 46 x 8
##    continent country country.name.en      SP.URB.TOTL pop_urb_growth SP.POP.TOTL
##    <chr>     <chr>   <chr>                      <dbl>          <dbl>       <dbl>
##  1 Europe    LIE     Liechtenstein               5429         0.777        37922
##  2 Europe    FRO     Faroe Islands              20659         0.675        49290
##  3 Europe    MDA     Moldova                  1510667         0.0461     3549750
##  4 Europe    BIH     Bosnia & Herzegovina     1679019         0.472      3507017
##  5 Europe    IMN     Isle of Man                44215         0.875        84287
##  6 Europe    SVK     Slovakia                 2923996         0.0650     5439892
##  7 Europe    ROU     Romania                 10564196        -0.523     19586539
##  8 Europe    SVN     Slovenia                 1121686         0.550      2066748
##  9 Europe    SRB     Serbia                   3928397        -0.276      7022268
## 10 Europe    HRV     Croatia                  2337910        -0.705      4125700
## # … with 36 more rows, and 2 more variables: pop_growth <dbl>, prop_urb <dbl>
```

*Liechtenstein had the lowest proportion of urban population in 2017*

### 2.6 (1 pt) Using `dplyr` functions, find the top 3 countries in terms of their total population in 2017.

```
topmyeurope <- myeurope %>%filter(year==2017)
arrange(topmyeurope,desc(SP.POP.TOTL))
```

```
## # A tibble: 46 x 8
##    continent country country.name.en  year SP.URB.TOTL pop_urb_growth
##    <chr>     <chr>   <chr>           <int>       <dbl>          <dbl>
##  1 Europe    RUS     Russia           2017   107348258          0.278
##  2 Europe    DEU     Germany          2017    63890984          0.468
##  3 Europe    FRA     France           2017    53815732          0.715
##  4 Europe    GBR     United Kingdom   2017    54892898          0.958
##  5 Europe    ITA     Italy            2017    42473185          0.287
##  6 Europe    ESP     Spain            2017    37294880          0.489
##  7 Europe    UKR     Ukraine          2017    31043784         -0.253
##  8 Europe    POL     Poland           2017    22825379         -0.106
##  9 Europe    ROU     Romania          2017    10564196         -0.523
## 10 Europe    NLD     Netherlands      2017    15604089          1.09
## # … with 36 more rows, and 2 more variables: SP.POP.TOTL <dbl>,
## #   pop_growth <dbl>
```

*Russia, Germany, and France are the top 3 countries in terms of total population in 2017*

# Question 3: (6 pts)

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

3.1 (1 pt) 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` with `glimpse()`. What variable could we use to join this dataset with `myeurope2017` dataset?

```
# geographic coordinates about countries in the world
mapWorld <- map_data("world")
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, …
```

*We could use the region variable to join the dataset with the myeurope2017 dataset.*

3.2 (1 pt) We want to use a `left_join()` function to create a new dataset, `mymap`, to add data to the countries in `myeurope2017` 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?

```
myeurope2017_1 <- rename(myeurope2017,c("region" = "country.name.en"))
mymap <- myeurope2017_1 %>%
left_join(mapWorld, by= c("region"))
```

*For the mapWorld dataset, the country United Kingdom is saved as UK under the region. For the myeurope2017 dataset, the country United Kingdom is saved as United Kingdom. When you left-join these two datasets, the names don't match.*

3.3 (1 pt) To identify all countries that did not have an exact match, do an `anti_join()` and display only distinct country names. How many countries did not have an exact match? *Note: using anti_join() is a very usedul function to identify differences between datasets.*

```
anti_join(myeurope2017_1,mapWorld,by="region")
```

```
## # A tibble: 5 x 8
##   continent country region     SP.URB.TOTL pop_urb_growth SP.POP.TOTL pop_growth
##   <chr>     <chr>   <chr>            <dbl>          <dbl>       <dbl>      <dbl>
## 1 Europe    BIH     Bosnia & …     1679019          0.472     3507017    -0.279
## 2 Europe    MKD     North Mac…     1202983          0.415     2083160     0.0938
## 3 Europe    CZE     Czechia        7803157          0.379    10591323     0.236
## 4 Europe    GBR     United Ki…    54892898          0.958    66022273     0.648
## 5 Europe    GIB     Gibraltar        34571          0.473       34571     0.473
## # … with 1 more variable: prop_urb <dbl>
```

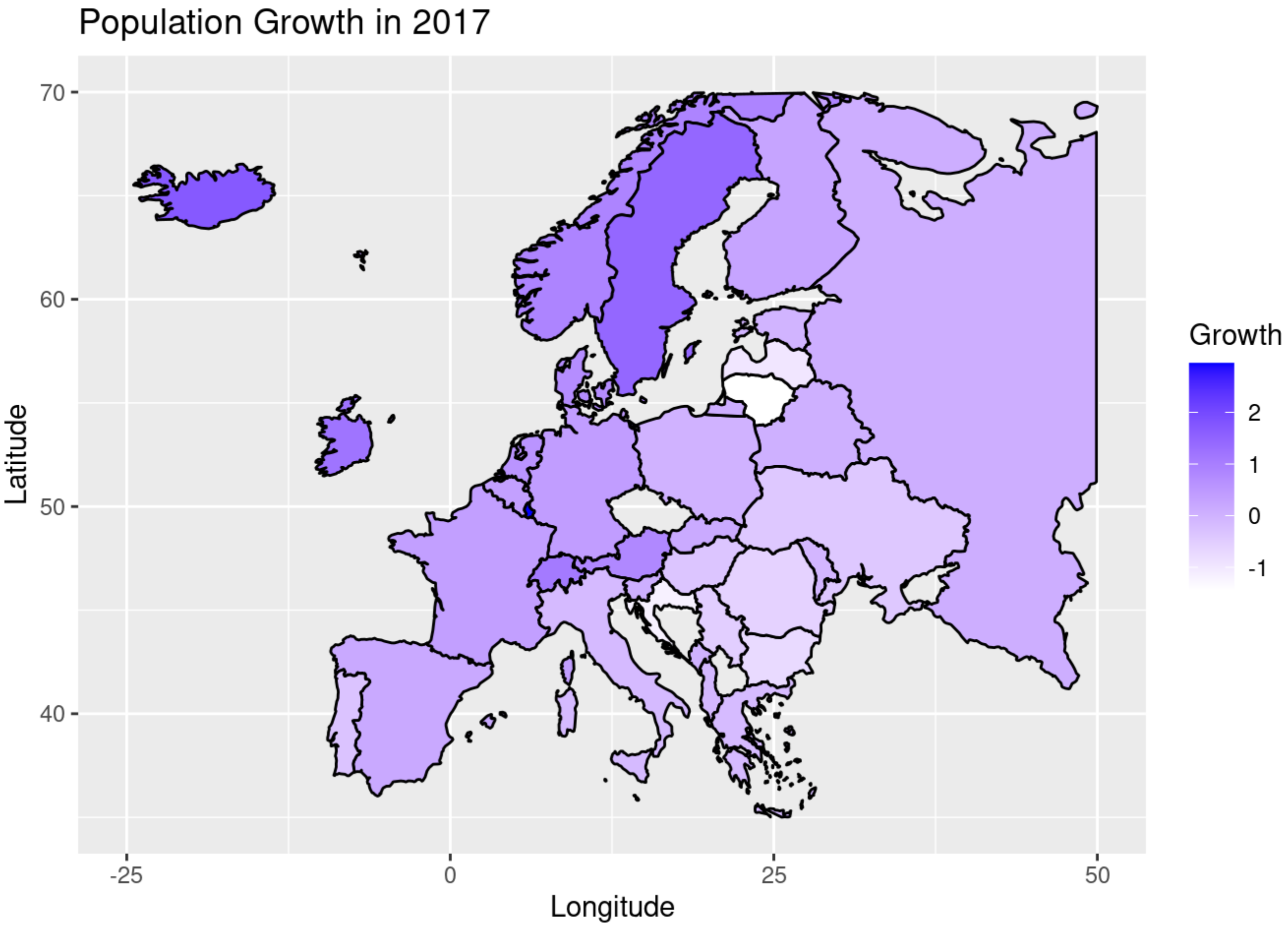
*5 countries did not have a match.*

3.4 (1 pt) 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 `myeurope2017` 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.

```
# myeurope2017_1 <- rename(myeurope2017,c("region" = "country.name.en"))
# mymap <- myeurope2017_1 %>%
# left_join(mapWorld, by= c("region"))
mymap <- myeurope2017 %>%
  mutate(country_clean = recode(country.name.en,
                             'United Kingdom' = 'UK',
                             'Bosnia & Herzegovina' = 'Bosnia and Herzegovina',
                             'Czechia' = 'Czech Republic',
                             'North Macedonia' = 'Macedonia'))%>% left_join(mapWorl
d,by= c("country.name.en" = "region"))



#
```

3.5 (2 pts) Let's visualize how population growth varies across European countries. Install the package `ggmap`, call the corresponding library, and use the R code provided below. Try to identify what each component of the graph does by completing the code with comments.

```
#install.packages("ggmap")
library(ggmap)

mymap %>%
  ggplot(aes(x=long, y=lat, group = group, fill = pop_growth)) +
  # Plots each individual country based on latitude and longitude in the ggplot and
outlines each country with a black color
   geom_polygon(colour = "black") +
  # Creates a smooth gradient between white and blue and colorcodes the bar relating
to      # pop_growth. Higher pop_growth has colors close to blue. Lower pop_growth ha
s colors      # close to white.
   scale_fill_gradient(low = "white", high = "blue", guide="colorbar") +
  # Adds title to graph, axis names to x and y axis and fills in plot with values in
Growth
   labs(fill = "Growth", title = "Population Growth in 2017", x="Longitude", y="Latit
ude")+
  # Sets scale limits
   xlim(-25,50) + ylim(35,70)
```



```
##                                        sysname 
##                                        "Linux" 
##                                        release 
##                             "5.4.0-66-generic" 
##                                        version 
## "#74-Ubuntu SMP Wed Jan 27 22:54:38 UTC 2021" 
##                                       nodename 
##                           "machina128-linux64" 
##                                        machine 
##                                       "x86_64" 
##                                          login 
##                                   "machinadmz" 
##                                           user 
##                                   "machinadmz" 
##                                 effective_user 
##                                   "machinadmz"
```