Merging Datasets

Joining Data in Assignment 3

The three questions in Assignment 3 require you to merge different datasets. I knew this would be a bit of a jump for you all, but I'm experimenting with how big to make these jumps as you grow in ability and confidence.

In programming, there is often an elegant but hard to understand why to solve a problem as well as a kind of ugly, simpler and easier to understand way to solve a problem. If you aren't sure how to solve a problem, then the less elegant, step-by-step way is often better and you will feel more confident that you are doing things correctly.

The Problem

Joining datesets is a very powerful tool and often the way that we learn something new. However, the datasets we are using are sometimes complex and are structured in different ways. You have to spend some time exploring the datasets to understand how they are structured and how they can be used.

First, we'll load our three datasets.

ESNames is a dataset that I created that has information on the countries in the EU and their membership status.

In this dataset, each row represents a country or group of countries.

```
        ESNames <-
read_csv('https://raw.githubusercontent.com/hulseyjw/POSC644/refs/heads/main/Week3/ESNames.csv', col_types = "ffffnf") # You may need to put a full path to the file on your computer. The strange code at the end is to specify the column types. (factor, factor, factor, factor, numeric, factor)

        head(ESNames)
```

```
# A tibble: 6 × 6
  CODE      countryname    Membership    EA20  AccYr ColdWar
  <fct>     <fct>          <fct>         <fct> <dbl> <fct>
1 EU27_2020 European Union Group         No       NA <NA>
2 BE        Belgium        EUMember_2020 Yes    1957 NATO
3 BG        Bulgaria       EUMember_2020 No     2007 WP
4 CZ        Czechia        EUMember_2020 No     2004 WP
5 DK        Denmark        EUMember_2020 No     1973 NATO
6 DE        Germany        EUMember_2020 Yes    1957 NATO
```

This dataset has a few variables that we can use to join with the Eurostat data.

CODE is the Eurostat code for the country or group of countries

countryname is a more readable name for the country or group of countries

Membership is the type of membership the country has relative to the EU

EA20 is a binary variable indicating whether the country is in the Euro Area

AccYr is the year the country joined the EU (if it is a member)

ColdWar identifies whether the country was part of NATO, the Warsaw Pact, Neutral, or part of Yugoslavia during the Cold War

We're also loading two datasets from Eurostat. **GDP** and **SB**.

```
GDP <- get_eurostat_data('tipsna40')
SB <- get_eurostat_data('tps00107')
```

```
head(GDP)
```

```
Key: <unit, na_item, geo>
            unit na_item    geo   time values
          <fctr>  <fctr> <fctr> <fctr>  <num>
1: CLV15_EUR_HAB    B1GQ     AT   1995  30460
2: CLV15_EUR_HAB    B1GQ     AT   1996  31130
3: CLV15_EUR_HAB    B1GQ     AT   1997  31740
4: CLV15_EUR_HAB    B1GQ     AT   1998  32850
5: CLV15_EUR_HAB    B1GQ     AT   1999  33950
6: CLV15_EUR_HAB    B1GQ     AT   2000  35010
```

```
head(SB)
```

```
Key: <unit, spdeps, geo>
      unit spdeps    geo   time values
    <fctr> <fctr> <fctr> <fctr>  <num>
1: PPS_HAB   DISA     AL   2018 152.49
2: PPS_HAB   DISA     AL   2019 152.58
3: PPS_HAB   DISA     AL   2020 164.54
4: PPS_HAB   DISA     AL   2021 158.91
5: PPS_HAB   DISA     AT   2010 657.55
6: PPS_HAB   DISA     AT   2011 678.97
```

Joining simple datasets isn't too hard. You just need to find a common variable that you can use to join the datasets. Since ESNames has one row for each country, we can use the country code to join the datasets.

So, you didn't have too much trouble with the first question in Assignment 3, because you just needed to join the GDP dataset with the ESNames dataset. Since ESNames just has one row for each country, it is not complicated to use left_join to merge the datasets.

GDP is a somewhat more complex dataset, because it has multiple rows for each country, representing the different years that GDP Per Capita is measured. So, each unique observation of value (the variable that has GDP Per Capita in it) is defined by country (geo) and year (time). We already have practiced filtering data, which allows us to see how this works.

If we use filter to show only the rows for Spain, we can see that there are multiple rows for Spain, each representing a different year.

```
        GDP %>% filter(geo == "ES")
```

```
Key: <unit, na_item, geo>
             unit na_item    geo   time values
           <fctr>  <fctr> <fctr> <fctr>  <num>
 1: CLV15_EUR_HAB    B1GQ     ES   1995  18220
 2: CLV15_EUR_HAB    B1GQ     ES   1996  18620
 3: CLV15_EUR_HAB    B1GQ     ES   1997  19210
 4: CLV15_EUR_HAB    B1GQ     ES   1998  19960
 5: CLV15_EUR_HAB    B1GQ     ES   1999  20750
 6: CLV15_EUR_HAB    B1GQ     ES   2000  21730
 7: CLV15_EUR_HAB    B1GQ     ES   2001  22470
 8: CLV15_EUR_HAB    B1GQ     ES   2002  22720
 9: CLV15_EUR_HAB    B1GQ     ES   2003  22960
10: CLV15_EUR_HAB    B1GQ     ES   2004  23310
11: CLV15_EUR_HAB    B1GQ     ES   2005  23690
12: CLV15_EUR_HAB    B1GQ     ES   2006  24260
13: CLV15_EUR_HAB    B1GQ     ES   2007  24640
14: CLV15_EUR_HAB    B1GQ     ES   2008  24420
15: CLV15_EUR_HAB    B1GQ     ES   2009  23310
16: CLV15_EUR_HAB    B1GQ     ES   2010  23230
17: CLV15_EUR_HAB    B1GQ     ES   2011  23000
18: CLV15_EUR_HAB    B1GQ     ES   2012  22330
19: CLV15_EUR_HAB    B1GQ     ES   2013  22090
20: CLV15_EUR_HAB    B1GQ     ES   2014  22500
21: CLV15_EUR_HAB    B1GQ     ES   2015  23440
22: CLV15_EUR_HAB    B1GQ     ES   2016  24100
23: CLV15_EUR_HAB    B1GQ     ES   2017  24750
24: CLV15_EUR_HAB    B1GQ     ES   2018  25230
25: CLV15_EUR_HAB    B1GQ     ES   2019  25520
26: CLV15_EUR_HAB    B1GQ     ES   2020  22610
27: CLV15_EUR_HAB    B1GQ     ES   2021  24120
28: CLV15_EUR_HAB    B1GQ     ES   2022  25380
29: CLV15_EUR_HAB    B1GQ     ES   2023  25730
             unit na_item    geo   time values
```

If we filter the data to show only the rows for 2019, we can see that there is only one row for each country.

```
        GDP %>% filter(time=="2019")
```

```
Key: <unit, na_item, geo>
             unit na_item       geo   time values
           <fctr>  <fctr>    <fctr> <fctr>  <num>
 1: CLV15_EUR_HAB    B1GQ        AT   2019  42030
 2: CLV15_EUR_HAB    B1GQ        BE   2019  38840
 3: CLV15_EUR_HAB    B1GQ        BG   2019   7430
 4: CLV15_EUR_HAB    B1GQ        CY   2019  25560
 5: CLV15_EUR_HAB    B1GQ        CZ   2019  18570
 6: CLV15_EUR_HAB    B1GQ        DE   2019  39840
 7: CLV15_EUR_HAB    B1GQ        DK   2019  51500
 8: CLV15_EUR_HAB    B1GQ      EA20   2019  33240
 9: CLV15_EUR_HAB    B1GQ        EE   2019  18580
10: CLV15_EUR_HAB    B1GQ        EL   2019  17140
11: CLV15_EUR_HAB    B1GQ        ES   2019  25520
12: CLV15_EUR_HAB    B1GQ EU27_2020   2019  29930
13: CLV15_EUR_HAB    B1GQ        FI   2019  41370
14: CLV15_EUR_HAB    B1GQ        FR   2019  34880
15: CLV15_EUR_HAB    B1GQ        HR   2019  12770
16: CLV15_EUR_HAB    B1GQ        HU   2019  13590
17: CLV15_EUR_HAB    B1GQ        IE   2019  69040
18: CLV15_EUR_HAB    B1GQ        IT   2019  29000
19: CLV15_EUR_HAB    B1GQ        LT   2019  15550
20: CLV15_EUR_HAB    B1GQ        LU   2019  96520
21: CLV15_EUR_HAB    B1GQ        LV   2019  14210
22: CLV15_EUR_HAB    B1GQ        MT   2019  26570
23: CLV15_EUR_HAB    B1GQ        NL   2019  44390
24: CLV15_EUR_HAB    B1GQ        PL   2019  13410
25: CLV15_EUR_HAB    B1GQ        PT   2019  19480
26: CLV15_EUR_HAB    B1GQ        RO   2019  10130
27: CLV15_EUR_HAB    B1GQ        SE   2019  47940
28: CLV15_EUR_HAB    B1GQ        SI   2019  21570
29: CLV15_EUR_HAB    B1GQ        SK   2019  16440
             unit na_item       geo   time values
```

If we filter by both Spain and 2019, we can see that there is only one row for Spain in 2019. So, the **key** for this GDP dataset is a combination of geo and time. In other words, there is one row for each country and year.

```
GDP %>% filter(time=="2019", geo=="ES")
```

```
Key: <unit, na_item, geo>
            unit na_item    geo   time values
          <fctr>  <fctr> <fctr> <fctr>  <num>
1: CLV15_EUR_HAB    B1GQ     ES   2019  25520
```

GDP has a key of geo and time, ESNames has a key of CODE, and the contents of GDP/$geoandESNames/$CODE match (ie. they use the came country codes and are formatted in the same way). So, we can use left_join to merge the datasets. Unlike in my example for the homework, I'm putting the resulting dataset in a new dataset called GDP_ESN. This avoids the problem many of you were having of repeatedly joining things to one dataset and getting values.x and values.y.

We're just merging on one variable from each dataset, so for all of the rows for each country in GDP (regardless of year), we're adding the information from ESNames.

```
GDP_ESN <- GDP %>%
  left_join(ESNames, by = c("geo" = "CODE"))

head(GDP_ESN)
```

```
Key: <unit, na_item, geo>
            unit na_item    geo   time values countryname    Membership   EA20
          <fctr>  <fctr> <fctr> <fctr>  <num>      <fctr>        <fctr> <fctr>
1: CLV15_EUR_HAB    B1GQ     AT   1995  30460     Austria EUMember_2020    Yes
2: CLV15_EUR_HAB    B1GQ     AT   1996  31130     Austria EUMember_2020    Yes
3: CLV15_EUR_HAB    B1GQ     AT   1997  31740     Austria EUMember_2020    Yes
4: CLV15_EUR_HAB    B1GQ     AT   1998  32850     Austria EUMember_2020    Yes
5: CLV15_EUR_HAB    B1GQ     AT   1999  33950     Austria EUMember_2020    Yes
6: CLV15_EUR_HAB    B1GQ     AT   2000  35010     Austria EUMember_2020    Yes
   AccYr ColdWar
   <num>  <fctr>
1:  1995 Neutral
2:  1995 Neutral
3:  1995 Neutral
4:  1995 Neutral
5:  1995 Neutral
6:  1995 Neutral
```

You can use this new dataset GDP_ESN to answer the first question in Assignment 3.

Working with the SB dataset

For SB, the values represent social benefits per capita for each country, year and type of benefit (OLD, SICK, FAM, UNEMP, DISAB, HOUS, and OTH).So SB has three keys: geo, time and spdeps.

If we just filter by Spain, then it shows us all of the rows for Spain, regardless of year or type of benefit.

```
SB %>% filter(geo=="ES")
```

```
Key: <unit, spdeps, geo>
        unit   spdeps    geo   time values
      <fctr>   <fctr> <fctr> <fctr>  <num>
  1: PPS_HAB     DISA     ES   2010 400.89
  2: PPS_HAB     DISA     ES   2011 404.03
  3: PPS_HAB     DISA     ES   2012 405.49
  4: PPS_HAB     DISA     ES   2013 417.64
  5: PPS_HAB     DISA     ES   2014 424.77
 ---
104: PPS_HAB UNEMPLOY     ES   2017 451.51
105: PPS_HAB UNEMPLOY     ES   2018 442.76
```

```
106: PPS_HAB UNEMPLOY     ES   2019 464.04
107: PPS_HAB UNEMPLOY     ES   2020 847.35
108: PPS_HAB UNEMPLOY     ES   2021 664.85
```

If we filter by Spain and 2019, then it shows us all of the rows for Spain in 2019, regardless of type of benefit.

```
        SB %>% filter(geo=="ES", time=="2019")
```

```
Key: <unit, spdeps, geo>
      unit           spdeps    geo   time  values
    <fctr>           <fctr> <fctr> <fctr>   <num>
1: PPS_HAB             DISA     ES   2019  449.19
2: PPS_HAB            EXCLU     ES   2019   65.60
3: PPS_HAB              FAM     ES   2019  367.10
4: PPS_HAB            HOUSE     ES   2019   30.71
5: PPS_HAB              OLD     ES   2019 2676.56
6: PPS_HAB             SICK     ES   2019 1812.62
7: PPS_HAB SPBENEFNOREROUTE     ES   2019 6498.85
8: PPS_HAB           SURVIV     ES   2019  633.03
9: PPS_HAB         UNEMPLOY     ES   2019  464.04
```

If we filter by 2019, and OLD, then it shows us the the spending on Old Age benefits for all countries in 2019.

```
        SB %>% filter(spdeps =="OLD", time=="2019")
```

```
Key: <unit, spdeps, geo>
       unit spdeps       geo   time  values
     <fctr> <fctr>    <fctr> <fctr>   <num>
 1: PPS_HAB    OLD        AL   2019  593.93
 2: PPS_HAB    OLD        AT   2019 4843.82
 3: PPS_HAB    OLD        BE   2019 3925.02
 4: PPS_HAB    OLD        BG   2019 1202.78
 5: PPS_HAB    OLD        CH   2019 4743.21
 6: PPS_HAB    OLD        CY   2019 2388.83
 7: PPS_HAB    OLD        CZ   2019 2466.49
 8: PPS_HAB    OLD        DE   2019 4056.28
 9: PPS_HAB    OLD        DK   2019 4498.70
10: PPS_HAB    OLD      EA19   2019 3716.11
11: PPS_HAB    OLD      EA20   2019 3693.47
12: PPS_HAB    OLD        EE   2019 1686.08
13: PPS_HAB    OLD        EL   2019 2719.89
14: PPS_HAB    OLD        ES   2019 2676.56
15: PPS_HAB    OLD EU27_2020   2019 3463.90
16: PPS_HAB    OLD        FI   2019 4351.46
17: PPS_HAB    OLD        FR   2019 4159.16
18: PPS_HAB    OLD        HR   2019 1469.27
19: PPS_HAB    OLD        HU   2019 1703.69
20: PPS_HAB    OLD        IE   2019 2185.89
21: PPS_HAB    OLD        IS   2019 2599.98
22: PPS_HAB    OLD        IT   2019 4071.84
```

```
23: PPS_HAB    OLD        LT   2019 1807.18
24: PPS_HAB    OLD        LU   2019 4866.81
25: PPS_HAB    OLD        LV   2019 1515.92
26: PPS_HAB    OLD        ME   2019  952.38
27: PPS_HAB    OLD        MT   2019 2014.55
28: PPS_HAB    OLD        NL   2019 3978.80
29: PPS_HAB    OLD        NO   2019 4515.47
30: PPS_HAB    OLD        PL   2019 2335.59
31: PPS_HAB    OLD        PT   2019 2735.91
32: PPS_HAB    OLD        RO   2019 1639.14
33: PPS_HAB    OLD        RS   2019 1141.89
34: PPS_HAB    OLD        SE   2019 4256.94
35: PPS_HAB    OLD        SI   2019 2402.54
36: PPS_HAB    OLD        SK   2019 1548.10
37: PPS_HAB    OLD        TR   2019 1219.05
       unit spdeps       geo   time  values
```

In order to isolate one row, we have to filter by all three keys.

```
SB %>% filter(geo=="ES", time=="2019", spdeps=="OLD")
```

```
Key: <unit, spdeps, geo>
      unit spdeps    geo   time  values
    <fctr> <fctr> <fctr> <fctr>   <num>
1: PPS_HAB    OLD     ES   2019 2676.56
```

To answer question 2 in Assignment 3, you will need to merge the SB dataset with the ESNames dataset. Since there is a unique row for each country in ESNames, you can use left_join to merge the datasets, and all of the rows for each country in SB will be merged with the information from ESNames.

```
SB_ESN <- SB %>%
  left_join(ESNames, by = c("geo" = "CODE"))

head(SB_ESN)
```

```
Key: <unit, spdeps, geo>
      unit spdeps    geo   time values countryname    Membership   EA20 AccYr
    <fctr> <fctr> <fctr> <fctr>  <num>      <fctr>        <fctr> <fctr> <num>
1: PPS_HAB   DISA     AL   2018 152.49        <NA>          <NA>   <NA>    NA
2: PPS_HAB   DISA     AL   2019 152.58        <NA>          <NA>   <NA>    NA
3: PPS_HAB   DISA     AL   2020 164.54        <NA>          <NA>   <NA>    NA
4: PPS_HAB   DISA     AL   2021 158.91        <NA>          <NA>   <NA>    NA
5: PPS_HAB   DISA     AT   2010 657.55     Austria EUMember_2020    Yes  1995
6: PPS_HAB   DISA     AT   2011 678.97     Austria EUMember_2020    Yes  1995
   ColdWar
    <fctr>
1:    <NA>
2:    <NA>
3:    <NA>
4:    <NA>
```

```
5: Neutral
6: Neutral
```

Since questions 1 and 2 just involve adding information from ESNames to the GDP and SB datasets, they are relatively simple joins Where things get more complicated is in merging the GDP and SB datasets, because GDP has two keys (geo and time) and SB has three keys (geo, time, and spdeps). Some of you already had difficulty with question two because you were trying to merge GDP with the ESNames variables to SB. This isn't wrong, but it does make question 2 harder than it needs to be.

Question 3 asks whether wealthier countries in the EU spend more on social benefits for families than poorer countries. To answer this question, you will need to merge the GDP and SB datasets. Since GDP has two keys (geo and time) and SB has three keys (geo, time, and spdeps), the way we've been joining above won't work. It worked when ESNames was the dataset we were adding to GDP or SB, because it had only one key, so we only had to join on one shared variable by=c("geo"="CODE").

There are two ways to deal with this problem. We can call it the "inelegant but works" way and the "elegant but hard to understand" way. Let's do the "inelegant but works" way first.

Inelegant but Works

This way of dealing with the problem is possible because to answer question 3, we don't need all of the information in the two datasets, so we can make them simpler by filtering them and then merge the filtered, simpler datasets. First, we can filter the SB_ESN dataset to only include the information that we need. I'm using SB_ESN because it has the ColdWar vairable in it, which we need to answer question 3.

To answer our question, we need to know the family benefits for each country in 2021, the most recent year they have data. So, we can filter the SB_ESN dataset to only include the rows for 2021 and FAM. Since we are filtering two of the keys, the resulting dataset will have only one key (geo).

```
SB_FAM <- SB_ESN %>% filter(time=="2021", spdeps=="FAM")
```

In the same way, we onlly need GDP for 2021, so we can filter the GDP database to only include the rows for 2021. (I wouldn't use GDP_ESN because it duplicates the ESNames variables we already have from SB_ESN.) I'm also using select to choose only the variables that we need and using rename to rename the values variable to GDP, since we already have a values in the SB_FAM dataset, but that values has different values and a different meaning (it's the family spending)

```
GDP_2021 <- GDP %>% filter(time=="2021") %>%
  select(geo, values) %>%
  rename(GDP = values)
```

Now we have two datasets that have only one key (geo) and we can merge them using left_join.

```
GDP_SB_FAM <- GDP_2021 %>%
  left_join(SB_FAM, by = "geo")

head(GDP_SB_FAM)
```

```
Key: <unit>
      geo   GDP    unit spdeps   time  values countryname    Membership   EA20
   <fctr> <num>  <fctr> <fctr> <fctr>   <num>      <fctr>        <fctr> <fctr>
1:     AT 40560 PPS_HAB    FAM   2021 1057.19     Austria EUMember_2020    Yes
2:     BE 39000 PPS_HAB    FAM   2021  802.94     Belgium EUMember_2020    Yes
3:     BG  7790 PPS_HAB    FAM   2021  310.20    Bulgaria EUMember_2020     No
4:     CY 26570 PPS_HAB    FAM   2021  301.69      Cyprus EUMember_2020    Yes
5:     CZ 18380 PPS_HAB    FAM   2021  558.14     Czechia EUMember_2020     No
6:     DE 39570 PPS_HAB    FAM   2021 1450.56     Germany EUMember_2020    Yes
   AccYr ColdWar
   <num>  <fctr>
1:  1995 Neutral
2:  1957    NATO
3:  2007      WP
4:  2004    NATO
5:  2004      WP
6:  1957    NATO
```

Your new dataset GDP_SB_FAM has the information you need to answer question 3.

Elegant but Harder to Understand Method

It is possible to merge the GDP and SB datasets without filtering them first. This is a more elegant solution, but it is harder to understand and verify that you are doing what you think you are doing. The key to doing this is to use the by argument in left_join to specify the variables that you are joining on.

```
        SB_GDP <- SB_ESN %>%
          left_join(GDP, by = c("geo", "time"))

        head(SB_GDP)
```

```
    unit.x spdeps    geo   time values.x countryname    Membership   EA20 AccYr
    <fctr> <fctr> <fctr> <fctr>    <num>      <fctr>        <fctr> <fctr> <num>
1: PPS_HAB   DISA     AL   2018   152.49        <NA>          <NA>   <NA>    NA
2: PPS_HAB   DISA     AL   2019   152.58        <NA>          <NA>   <NA>    NA
3: PPS_HAB   DISA     AL   2020   164.54        <NA>          <NA>   <NA>    NA
4: PPS_HAB   DISA     AL   2021   158.91        <NA>          <NA>   <NA>    NA
5: PPS_HAB   DISA     AT   2010   657.55     Austria EUMember_2020    Yes  1995
6: PPS_HAB   DISA     AT   2011   678.97     Austria EUMember_2020    Yes  1995
   ColdWar        unit.y na_item values.y
    <fctr>        <fctr>  <fctr>    <num>
1:    <NA>          <NA>    <NA>       NA
2:    <NA>          <NA>    <NA>       NA
3:    <NA>          <NA>    <NA>       NA
4:    <NA>          <NA>    <NA>       NA
5: Neutral CLV15_EUR_HAB    B1GQ    39070
6: Neutral CLV15_EUR_HAB    B1GQ    40080
```

In order to look at this complex dataset and see if it is what you think it is, you can filter it to only include the rows for 2021 and FAM.

```r
SB_GDP %>% filter(time=="2021", spdeps=="FAM")
```

```
Key: <unit, spdeps, geo>
     unit.x spdeps       geo   time values.x    countryname    Membership
     <fctr> <fctr>    <fctr> <fctr>    <num>         <fctr>        <fctr>
 1: PPS_HAB    FAM        AL   2021    83.56           <NA>          <NA>
 2: PPS_HAB    FAM        AT   2021  1057.19        Austria EUMember_2020
 3: PPS_HAB    FAM        BE   2021   802.94        Belgium EUMember_2020
 4: PPS_HAB    FAM        BG   2021   310.20       Bulgaria EUMember_2020
 5: PPS_HAB    FAM        CH   2021   707.61    Switzerland          EFTA
 6: PPS_HAB    FAM        CY   2021   301.69         Cyprus EUMember_2020
 7: PPS_HAB    FAM        CZ   2021   558.14        Czechia EUMember_2020
 8: PPS_HAB    FAM        DE   2021  1450.56        Germany EUMember_2020
 9: PPS_HAB    FAM        DK   2021  1291.64        Denmark EUMember_2020
10: PPS_HAB    FAM      EA19   2021   789.17           <NA>          <NA>
11: PPS_HAB    FAM      EA20   2021   785.64      Euro Area         Group
12: PPS_HAB    FAM        EE   2021   609.22        Estonia EUMember_2020
13: PPS_HAB    FAM        EL   2021   289.02         Greece EUMember_2020
14: PPS_HAB    FAM        ES   2021   403.37          Spain EUMember_2020
15: PPS_HAB    FAM EU27_2020   2021   776.54 European Union         Group
16: PPS_HAB    FAM        FI   2021  1056.55        Finland EUMember_2020
17: PPS_HAB    FAM        FR   2021   765.45         France EUMember_2020
18: PPS_HAB    FAM        HR   2021   441.32        Croatia EUMember_2020
19: PPS_HAB    FAM        HU   2021   464.89        Hungary EUMember_2020
20: PPS_HAB    FAM        IE   2021   657.07        Ireland EUMember_2020
21: PPS_HAB    FAM        IS   2021  1025.27        Iceland          EFTA
22: PPS_HAB    FAM        IT   2021   378.83          Italy EUMember_2020
23: PPS_HAB    FAM        LT   2021   598.47      Lithuania EUMember_2020
24: PPS_HAB    FAM        LU   2021  2357.76     Luxembourg EUMember_2020
25: PPS_HAB    FAM        LV   2021   490.96         Latvia EUMember_2020
26: PPS_HAB    FAM        ME   2021   111.96     Montenegro  EU_Candidate
27: PPS_HAB    FAM        MT   2021   306.21          Malta EUMember_2020
28: PPS_HAB    FAM        NL   2021   526.75    Netherlands EUMember_2020
29: PPS_HAB    FAM        NO   2021  1403.65         Norway          EFTA
30: PPS_HAB    FAM        PL   2021   909.00         Poland EUMember_2020
31: PPS_HAB    FAM        PT   2021   310.46       Portugal EUMember_2020
32: PPS_HAB    FAM        RO   2021   496.92        Romania EUMember_2020
33: PPS_HAB    FAM        RS   2021   174.34         Serbia  EU_Candidate
34: PPS_HAB    FAM        SE   2021  1034.83         Sweden EUMember_2020
35: PPS_HAB    FAM        SI   2021   523.62       Slovenia EUMember_2020
36: PPS_HAB    FAM        SK   2021   436.45       Slovakia EUMember_2020
37: PPS_HAB    FAM        TR   2021   104.26         Turkey  EU_Candidate
     unit.x spdeps       geo   time values.x    countryname    Membership
       EA20 AccYr ColdWar        unit.y na_item values.y
     <fctr> <num>  <fctr>        <fctr>  <fctr>    <num>
 1:    <NA>    NA    <NA>          <NA>    <NA>       NA
 2:     Yes  1995 Neutral CLV15_EUR_HAB    B1GQ    40560
 3:    Yes  1957    NATO CLV15_EUR_HAB    B1GQ    39000
 4:     No  2007      WP CLV15_EUR_HAB    B1GQ     7790
 5:     No    NA Neutral          <NA>    <NA>       NA
 6:    Yes  2004    NATO CLV15_EUR_HAB    B1GQ    26570
 7:     No  2004      WP CLV15_EUR_HAB    B1GQ    18380
 8:    Yes  1957    NATO CLV15_EUR_HAB    B1GQ    39570
 9:     No  1973    NATO CLV15_EUR_HAB    B1GQ    53970
10:   <NA>    NA    <NA>          <NA>    <NA>       NA
11:     No    NA    <NA> CLV15_EUR_HAB    B1GQ    33100
12:    Yes  2004      WP CLV15_EUR_HAB    B1GQ    19250
13:    Yes  1981    NATO CLV15_EUR_HAB    B1GQ    16980
14:    Yes  1986    NATO CLV15_EUR_HAB    B1GQ    24120
15:     No    NA    <NA> CLV15_EUR_HAB    B1GQ    30000
16:    Yes  1995 Neutral CLV15_EUR_HAB    B1GQ    41290
17:    Yes  1957    NATO CLV15_EUR_HAB    B1GQ    34260
18:    Yes  2013    Yugo CLV15_EUR_HAB    B1GQ    13570
19:     No  2004      WP CLV15_EUR_HAB    B1GQ    13990
20:    Yes  1973 Neutral CLV15_EUR_HAB    B1GQ    83910
21:     No    NA    NATO          <NA>    <NA>       NA
22:    Yes  1957    NATO CLV15_EUR_HAB    B1GQ    29080
23:    Yes  2004      WP CLV15_EUR_HAB    B1GQ    16440
24:    Yes  1957    NATO CLV15_EUR_HAB    B1GQ    99360
25:    Yes  2004      WP CLV15_EUR_HAB    B1GQ    14870
26:     No    NA    Yugo          <NA>    <NA>       NA
27:    Yes  2004 Neutral CLV15_EUR_HAB    B1GQ    28340
28:    Yes  1957    NATO CLV15_EUR_HAB    B1GQ    44870
29:     No    NA    NATO          <NA>    <NA>       NA
30:     No  2004      WP CLV15_EUR_HAB    B1GQ    14140
31:    Yes  1986    NATO CLV15_EUR_HAB    B1GQ    18880
32:     No  2007      WP CLV15_EUR_HAB    B1GQ    10460
33:     No    NA    Yugo          <NA>    <NA>       NA
34:     No  1995 Neutral CLV15_EUR_HAB    B1GQ    49110
35:    Yes  2004    Yugo CLV15_EUR_HAB    B1GQ    22230
36:    Yes  2004      WP CLV15_EUR_HAB    B1GQ    16690
37:     No    NA    NATO          <NA>    <NA>       NA
      EA20 AccYr ColdWar        unit.y na_item values.y
```

We can play around with filtering in different ways and looking at the dataset to verify that it has joined the dataset in the way that we expect. The new dataset has the same keys as the original SB dataset, so we can use the same methods to filter it and look at it. Since GDP just had two keys, all of the rows that have the same value for geo and time will have the same GDP information for that country and year.

```r
SB_GDP %>% filter(geo=="ES", time=="2019")
```

```
Key: <unit, spdeps, geo>
    unit.x           spdeps    geo   time values.x countryname    Membership
    <fctr>           <fctr> <fctr> <fctr>    <num>      <fctr>        <fctr>
1: PPS_HAB             DISA     ES   2019   449.19       Spain EUMember_2020
2: PPS_HAB           EXCLU      ES   2019    65.60       Spain EUMember_2020
3: PPS_HAB              FAM     ES   2019   367.10       Spain EUMember_2020
```

```
4: PPS_HAB            HOUSE     ES   2019    30.71       Spain EUMember_2020
5: PPS_HAB              OLD     ES   2019  2676.56       Spain EUMember_2020
6: PPS_HAB             SICK     ES   2019  1812.62       Spain EUMember_2020
7: PPS_HAB SPBENEFNOREROUTE     ES   2019  6498.85       Spain EUMember_2020
8: PPS_HAB           SURVIV     ES   2019   633.03       Spain EUMember_2020
9: PPS_HAB         UNEMPLOY     ES   2019   464.04       Spain EUMember_2020
     EA20 AccYr ColdWar        unit.y na_item values.y
   <fctr> <num>  <fctr>        <fctr>  <fctr>    <num>
1:    Yes  1986    NATO CLV15_EUR_HAB    B1GQ    25520
2:    Yes  1986    NATO CLV15_EUR_HAB    B1GQ    25520
3:    Yes  1986    NATO CLV15_EUR_HAB    B1GQ    25520
4:    Yes  1986    NATO CLV15_EUR_HAB    B1GQ    25520
5:    Yes  1986    NATO CLV15_EUR_HAB    B1GQ    25520
6:    Yes  1986    NATO CLV15_EUR_HAB    B1GQ    25520
7:    Yes  1986    NATO CLV15_EUR_HAB    B1GQ    25520
8:    Yes  1986    NATO CLV15_EUR_HAB    B1GQ    25520
9:    Yes  1986    NATO CLV15_EUR_HAB    B1GQ    25520
```

One thing that we can see is that we have a number of variables that we don't need and we have some variables with suffixes .x and .y. This is because the variables in the two datasets had the same name, so R added the suffixes to distinguish them. We can use select to choose only the variables that we need and rename to rename the variables that have suffixes. Most important, the values variable from the GDP dataset values.y must be renamed to GDP, because we already have a values variable in the SB dataset (values.x) and it has a different meaning.

```
        SB_GDP <- SB_GDP %>%
          select(geo, time, values.y, spdeps, countryname, Membership, EA20, AccYr, 
ColdWar) %>%
          rename(GDP = values.y)

        head(SB_GDP)
```

```
      geo   time   GDP spdeps countryname    Membership   EA20 AccYr ColdWar
   <fctr> <fctr> <num> <fctr>      <fctr>        <fctr> <fctr> <num>  <fctr>
1:     AL   2018    NA   DISA        <NA>          <NA>   <NA>    NA    <NA>
2:     AL   2019    NA   DISA        <NA>          <NA>   <NA>    NA    <NA>
3:     AL   2020    NA   DISA        <NA>          <NA>   <NA>    NA    <NA>
4:     AL   2021    NA   DISA        <NA>          <NA>   <NA>    NA    <NA>
5:     AT   2010 39070   DISA     Austria EUMember_2020    Yes  1995 Neutral
6:     AT   2011 40080   DISA     Austria EUMember_2020    Yes  1995 Neutral
```

Now we can use this dataset to answer question 3. We'll still have to filter it when we use it in order to make our plots and answer the question.

Inelegant vs. Elegant

Both of these methods work to answer the question. So, you should use the one that makes sense to you. For me, the advante of the inelegant method of filtering and then merging is that it is easier for me to verify that I am doing what I think I am doing. I can look at the filtered datasets and see that they are what I expect.

The advantage of the elegant method of merging and then filtering is that it is more efficient and doesn't require you to create new datasets. I can just use the complex, new dataset and filter it for what I need when I need it.

If I were going to go on and make plots for all of the different kinds of social benefits or different combinations of years over time, then the elegant way would be a lot faster. The inelegant way would still work, but I'd have to go back and step by step make the new datasets for each new plot.

Homework

For each question, create a figure that explores the question and write a brief explanation of what you see. Clearly label your plots. Submit the .qmd file and the .html file to Canvas.

1. Do countries who joined the EU earlier tend to be wealthier or poorer than those who joined later? Create a plot that shows the relationship.
2. Do former communist countries spend more or less on social benefits for the old and sick than other countries? Create a plot that shows the relationship.(spdeps in SB has OLD and SICK)
3. Do wealthier countries in the EU spend more on social benefits for families (FAM) than poorer countries? Create a plot that shows the relationship.