module-1.2

Megan Robertson

Downloading data from an API

We will be using the `wbstats` package from the World Bank API.

In the first part of this module, we are going to download data from the World Bank using the `wbstats` package.

```
# Load packages

library(wbstats)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(janitor)
```

```
Attaching package: 'janitor'

The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```r
# Locate indicators

#flfp_indicators <- wb_search ("female labor force")
#print(flfp_indicators, n=26)

#women_parliament <- wb_search("women in parliament")
#print(women_parliament)

# Store the list of indicators in an object
indicators <- c("flfp" = "SL.TLF.CACT.FE.ZS" , "women_rep" = "SG.GEN.PARL.ZS")

# Download the data
women_emp <- wb_data(indicators, mrv = 50) |>
select(!iso2c) |>
rename(year = date) |>
mutate(
  flfp = round_to_fraction(flfp, denominator = 100),
  women_rep = round_to_fraction(women_rep, denominator = 100))

# View the data
glimpse(women_emp)
```

```
Rows: 7,595
Columns: 5
$ iso3c     <chr> "ABW", "ABW", "ABW", "ABW", "ABW", "ABW", "ABW", "ABW", "ABW~
$ country   <chr> "Aruba", "Aruba", "Aruba", "Aruba", "Aruba", "Aruba", "Aruba~
$ year      <dbl> 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, ~
$ women_rep <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ flfp      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

Next we are moving on to filter the observations and delve into how to create new variables using the `V-Dem Dataset`. We will again be using the `filter()` and `select()` so we make it a little bit easier than just downloading the entire `vdem` dataset in one swoop. We will also want to retain `country_name year` and `country_id` for the purposes of merging these data with our World Bank data.

For even more fun, we will create a new coding called `region` to facilitate some analysis later on in the course. We plan to pipe in a `mutate()` call where we use `case_match()` function to change the `region` from a numeric variable to a string. Hopefully, if done well, we can rely on this to create visualizations later on.

And finally, all of this new data will be stored in an object called `democracy`.

It is also good to know that there are a lot of good reference materials here in case it all gets confusing –> `https://www.v-dem.net/data/the-v-dem-dataset/`

Ready, let's do this!

Filter observations, select and create new variables

In this lesson, we will work with the VDem dataset to download data from VDem and clean it.

```
# Load packages
library(vdemdata)
library(dplyr)

# Download the data
democracy <- vdem |>
  filter(year >= 1990) |>
  select(
    country = country_name,
    vdem_ctry_id = country_id,
    year,
    polyarchy = v2x_polyarchy,
    gdp_pc = e_gdppc,
    region = e_regionpol_6C
    ) |>
  mutate(
  region = case_match(region,
                      1 ~ "Eastern Europe",
                      2 ~ "Latin America",
                      3 ~ "Middle East",
                      4 ~ "Africa",
                      5 ~ "The West",
                      6 ~ "Asia"))


# View the data
glimpse(democracy)
```

```
Rows: 6,204
Columns: 6
$ country      <chr> "Mexico", "Mexico", "Mexico", "Mexico", "Mexico", "Mexico~
$ vdem_ctry_id <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, ~
```

```
$ year         <dbl> 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 199~
$ polyarchy    <dbl> 0.389, 0.412, 0.437, 0.447, 0.470, 0.480, 0.508, 0.556, 0~
$ gdp_pc       <dbl> 24.396, 25.077, 25.561, 25.967, 26.101, 25.434, 25.851, 2~
$ region       <chr> "Latin America", "Latin America", "Latin America", "Latin~
```

Add country codes to a data frame

Next we are going to really have some fun. It's June, it's beautiful out, and instead of being outside, I'm talking about continuing here in Module 1.2, and continuing to catch up on this class as quickly as possible this week.

Let's go!

Here's the mission: We're going to add country codes to our V-Dem data so we can merge them with our World Bank data.

```
# load countrycode

library(countrycode)

# Calling mutate as we are creating a new variable called iso3c. To create the new variable w

democracy <- democracy |>
  mutate(iso3c = countrycode(
         sourcevar = vdem_ctry_id,
         origin = "vdem",
         destination = "wb",
         )) |>
  relocate(iso3c, .after = vdem_ctry_id)
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `iso3c = countrycode(...)`.
Caused by warning:
! Some values were not matched unambiguously: 23, 128, 137, 139, 236
```

```
# View the data!
glimpse(democracy)
```

```
Rows: 6,204
Columns: 7
$ country      <chr> "Mexico", "Mexico", "Mexico", "Mexico", "Mexico", "Mexico~
```

```
$ vdem_ctry_id <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, ~
$ iso3c        <chr> "MEX", "MEX", "MEX", "MEX", "MEX", "MEX", "MEX", "MEX", "~
$ year         <dbl> 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 199~
$ polyarchy    <dbl> 0.389, 0.412, 0.437, 0.447, 0.470, 0.480, 0.508, 0.556, 0~
$ gdp_pc       <dbl> 24.396, 25.077, 25.561, 25.967, 26.101, 25.434, 25.851, 2~
$ region       <chr> "Latin America", "Latin America", "Latin America", "Latin~
```

```
# After fiddling with this, we can see that the iso3c was moved to the right of country code
```

Merge two data sets

What we've done so far:
• We downloaded some data from the World Bank on women's empowerment indicators, specifically women's participation in the labor force and their representation in parliament.

• We then downloaded some data from VDem, namely their famous polyarchy score and GDP.
• We and then made our own region labels so we can merge the two sets of data.

Now is the day we've all been waiting for: let's merge the data!

The joy of having a common country code is that we can now merge the two data sets. A **mutating join** adds observations from one dataset to another, and **filtering joins** filter out observations based on their presence or absence in another dataset. We will use a **mutating join** called `left_join()` to merge the two datasets.

What is a **left join**, you ask? A left join keeps all of the observations from the first data frame (`x`) and only matching observations in the second data frame (`y`).

The `left_join()` is from `dplyr`.

Once you do a first round on this, we see have two country name columns, and we want to just have one. So we're going to go back and add a line to rename it, and get rid of the other one.

Let's rock.

```
# Load readr, the necessary package for this
library(readr)

# Make sure we're in the right working directory, since we got some error messages about that
setwd("~/Desktop/DataViz_2102")

#Perform left join using common iso3c variable and year.
dem_women <- left_join(democracy, women_emp, by = c("iso3c", "year"))|>
```

```
  rename(country = country.x)|>
  select(!country.y)

# Does that data folder exist in a place we can write to?
if(!dir.exists("data")) {
  dir.create("data")
}


# Save as a .csv for future use so we can use it in the next module for visualizations.  We'l
write_csv(dem_women, "data/dem_women.csv")

# View the data
glimpse(dem_women)
```

```
Rows: 6,204
Columns: 9
$ country      <chr> "Mexico", "Mexico", "Mexico", "Mexico", "Mexico", "Mexico~
$ vdem_ctry_id <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, ~
$ iso3c        <chr> "MEX", "MEX", "MEX", "MEX", "MEX", "MEX", "MEX", "MEX", "~
$ year         <dbl> 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 199~
$ polyarchy    <dbl> 0.389, 0.412, 0.437, 0.447, 0.470, 0.480, 0.508, 0.556, 0~
$ gdp_pc       <dbl> 24.396, 25.077, 25.561, 25.967, 26.101, 25.434, 25.851, 2~
$ region       <chr> "Latin America", "Latin America", "Latin America", "Latin~
$ women_rep    <dbl> NA, NA, NA, NA, NA, NA, NA, 14.20, 17.40, 18.20, 16.00, 1~
$ flfp         <dbl> 33.96, 34.21, 34.99, 35.84, 36.37, 37.61, 37.68, 39.64, 3~
```

Group, summarize and arrange data

Now we have wrangled the data, we can now group the data by region, summarize the data and then arrange it in descending (or ascending order) based on the values of a particular variable.

```
# group, summarize, arrange
dem_summary<- dem_women |>
  group_by(region) |>
  summarize(
    polyarchy = mean(polyarchy, na.rm = TRUE),
    gdp_pc = mean(gdp_pc, na.rm = TRUE),
    flfp = mean(flfp, na.rm = TRUE),
    women_rep = mean(women_rep, na.rm = TRUE)
```

```
  )|>
  arrange(desc(women_rep))

glimpse(dem_summary)
```

```
Rows: 6
Columns: 5
$ region    <chr> "The West", "Latin America", "Eastern Europe", "Africa", "As~
$ polyarchy <dbl> 0.8691655, 0.6332560, 0.5367554, 0.3901770, 0.4083480, 0.242~
$ gdp_pc    <dbl> 85.849149, 22.281340, 27.315742, 8.679578, 21.891156, 45.425~
$ flfp      <dbl> 53.31588, 48.51694, 51.14573, 57.17447, 51.04393, 26.69013
$ women_rep <dbl> 28.72443, 22.22819, 18.69017, 18.02494, 14.66381, 10.66862
```

This gives us a new dataframe, `dem_summary`, which has variables for each region in the dataset. You can also go back and adjust `mean` to max, min and median if you wanted to play around with how we look at the data in a table.

We rendered the document as `.html` which opened up in a web browser, and it looked good. But this time, let's render it as a `.pdf` file.