Demographic Data

In the last module, we used a variety of functions to calculate metrics from Census of Agriculture data. By now, you may be used to how that data is formatted. Now, we will apply those functions to data from a different source, the American Community survey, to calculate population demographics. These data are not all in the same format and will require data cleaning steps before we can calculate metrics.

Objectives:

1. Clean data by extracting column names from a multi-row header, filtering with `%in%` statements, and matching variable codes to their meanings.
2. Increase reproducibility by using the `tidycensus` package to pull data from the online American Community Survey data portal.
3. Calculate employment metrics (percent residents employed in agriculture, manufacturing, hospitality, and technology), residents over retirement age, and net migration rate.

Getting Started

(1) Open RStudio and your project file. (2) Create a new Quarto file.

(3) Download the following `.csv` files into your `data` folder from this GitHub folder: `ID_employment.csv` and `ID_county-to-county-2013-2017-ins-outs-nets-gross.csv`.

Employment metrics

(4) Create a new code chunk and use `read.csv` to read in the data from `ID_employment.csv`.
(5) Use `head` and at least one other function from Module 2 to inspect the data.

This is data from the American Community Survey for the year 2017. It has information on the gender and employment sector of residents in each county, but this information is contained in codes in the variable column. The meanings of these codes are stored in `ID_employment_codes.csv` in the GitHub folder. The `moe` column stands for "margin of error" – an important estimate of uncertainty. However, we will not be using it for these case studies.

(6) Filter the employment data to your county.

To calculate percentages, we also need the total population of your county. (7) Refer to the shared Google sheet and save the value you previously found for total population in a new object in this report.

We need to calculate the percent of county residents employed in (1) agriculture, (2) manufacturing/warehousing, (3) hospitality, and (4) technology. To do this, we need the corresponding codes.

(8) Look at the `ID_employment_codes.csv` file in the GitHub folder.

(9) Make a table of the codes that apply to each category in your report. Since codes refer to employment and gender, there will be at least two codes for each category.

To make a table, use the website https://www.tablesgenerator.com/markdown_tables. Fill in the correct values in the editable table at the top of the page, generate the table, and copy the generated table into your report *outside* a code chunk. It will be formatted similar to the table below.

Sector	Employment Category	Codes
Agriculture	Agriculture, forestry, fishing and hunting	C24030_004, C?????_???
Manufacturing/Warehousing	Manufacturing	2 codes
Manufacturing/Warehousing	Transportation and warehousing	2 codes

Sector	Employment Category	Codes
Hospitality	Arts, entertainment, and recreation, and accommodation and food services	2 codes
Technology	Information	2 codes
Technology	Professional, scientific, and technical services	2 codes

Next, we will use four `subset` operations to find the number of people employed in each sector of interest. A comparison we can use in `subset` is `%in%`. When this logical expression is used, `subset` will keep any row with a value in the vector following `%in%`.

(10) Modify the code below to filter to the agricultural codes.

```
my_county_ag_employed <- subset(my_county_employ,
                                variable %in% c("C24030_004", "C?????_???"))
```

(11) Find the sum of the `estimate` column to calculate the total number of residents employed in the agricultural sector.

(12) Divide the number of residents employed in agriculture by the total population of your county. Round the percentage of residents employed in agriculture to one decimal and record it in the shared Google sheet.

(13) Repeat steps (10) through (12) for the other three sectors, recording your results.

Retirement

Retirees represent two potential pathways to farmland loss: farmers themselves retiring and not having successors willing to take over farm operations and wealthy retirees seeking to retire to the country. While we cannot differentiate between the two from census data, we can track how the number of people over 65 outside the labor force has changed.

We are going to pull this data directly from the census data website using a package called `tidycensus`. This increases reproducibility by allowing anyone to see the source of our data.

(14) In the Console, install the `tidycensus` package:

```
install.packages('tidycensus')
```

(15) In your report, import the `tidycensus` library.

Websites that allow users to pull data want to make sure that someone isn't spamming them and clogging their servers. To prevent this, they require that users get an API key. **(16)** Request a census API key at http://api.census.gov/data/key_signup.html. **(17)** Once you receive your key by email, import it into your report:

```
census_api_key("YOUR API KEY GOES HERE")
```

You should keep your API key private. While I have no security concerns with this particular project, it is good practice. **(18)** To hide your key from the PDF report but leave it active to run your code, modify the top of your code chunk from `{r}` to `{r, include=FALSE}`. This will run the code when your PDF is created, but it will not be printed.

Within the `tidycensus` library, we are going to use the `get_acs` function to get American Community Survey data. It requires us to specify the location of interest (state and county) and a table or variable code. In the next module, you will explore how to find codes of interest for yourself. For now, we are going to pull a few variables from a table with data on employment by age and gender. These are the codes we are interested in:

- B23001_077: Total Male, 65 to 69 years, Not in labor force
- B23001_163: Total Female, 65 to 69 years, Not in labor force
- B23001_082: Total Male, 70 to 74 years, Not in labor force
- B23001_168: Total Female, 70 to 74 years, Not in labor force
- B23001_087: Total Male, 75 years and over, Not in labor force
- B23001_173: Total Female, 75 years and over, Not in labor force

(19) Modify the code below to pull retiree data for your county. The `survey` argument specifies that we want estimates based on 5 years of data, since one year of data is not enough for estimates in small counties.

```
my_county_retire <- get_acs(geography = "county",
                            state = "ID",
                            county = "Bear Lake",
                            year = 2017,
                            survey = "acs5",
                            variables = c("B23001_077", "B23001_163",
                                          "B23001_082", "B23001_168",
                                          "B23001_087", "B23001_173"))
```

```
Getting data from the 2013-2017 5-year ACS
```

(20) Calculate the sum of all retirees in your county. **(21)** Calculate the percent of county residents who are retirees over 65, round to one decimal place, and record in the shared Google sheet.

If you want to calculate other metrics from this table, you can find all the codes at https://api.census.gov/data/2009/acs/acs5/groups/B23001.html.

Net migration rate

The net migration rate is calculated with this equation:

$NetMigration = \frac{NumEntering - NumLeaving}{Population} \cdot 1000$

Net migration per 1,000 residents is the number of people entering the county minus the number of people leaving divided by the total population in the middle of the time period.

In `ID_county-to-county-2013-2017-ins-outs-nets-gross.csv`, we have information on the counties that people moved to and moved out of between 2013 and 2017. We need to calculate (1) the number of people that moved to your county in that time period, (2) the number of people who moved out of your county in that time period, and (3) the county population in 2015 to calculate the net migration rate.

(22) Create a new code chunk and use `read.csv` to read in the data from `ID_county-to-county-2013-2017-ins`
(23) Use `colnames` to inspect the data.

These column names are not descriptive at all! This is because this dataset has multiple header columns, as does a lot of other Census data. We need to extract the most useful header row for our column names, and then read in the data separately.

(24) Open up the data frame in your Environment. If the current column names shown here are assumed to be row 1, then notice that the best row for column names is actually row 2 and the data starts on row 4. Additionally, the data stops at column 16, but `R` reads in 40 columns. This is an artifact of converting an Excel file to a csv, and it's helpful to know how to deal with this.

(25) Create a new `read.csv` command and save the results in an object called `migration_columns`. Use `skip=1` to skip the first row of data and use the second row of data as column names, and use `nrows=1` to read in a single row of data that is also a header row and not a data row.

```
migration_columns <-
  read.csv("data/ID_county-to-county-2013-2017-ins-outs-nets-gross.csv",
           skip = 1, nrows = 1)
```

(26) Print `migration_columns`.

(27) We only need the first 16 column names. Use `colnames` to get the column names we imported without the extra data and brackets to extract the first 16 entries.

```
migration_column_names <- colnames(migration_columns)[1:16]
```

(28) Print `migration_column_names` to check the result. Don't worry about the columns with names like `X`, `X.1`, etc. Those are margin of error columns and will not be used for this analysis.

(29) Create a new code chunk and use `read.csv` to read in the data from ID_county-to-county-2013-2017-ins-outs-nets-gross.csv again. This time, use the arguments `skip=3, header=FALSE` to only read in the data.

(30) Use `data_frame[, 1:16]` to get the first 16 columns of data and remove all the empty columns. Save this data frame in a new object.

(31) Add the column names you saved to this data frame by assigning `colnames(migration)` the names you saved earlier.

```
colnames(migration) <- migration_column_names
```

(32) Print the `head` of the migration data to check that the data has column names.

(33) Now that the data is formatted correctly, `subset` the dataset to rows where the column `County.Name.of.Geography.A` is equal to your county.

(34) Click on the subset you just created in the Environment tab. Take a look at the `State.U.S..Island.Area.Foreign.Region.of.Geography.B` column to see where people are moving from into your county. Did any location surprise you?

(35) Check the `class` of the columns `Flow.from.Geography.B.to.Geography.A` and `Counterflow.from.Geography.A.to.Geography.B1`. If the data type is `"character"`, use `str_remove_all` and `as.numeric` to remove commas and change the data type to numeric.

(36) Calculate the number of people that moved to your county between 2013-2017 (hint: use one of the columns from step (35) in a function that will add all the values together). Save this value in an object called `NumEntering`.

(37) Calculate the number of people that moved out of your county between 2013-2017 (hint: use the other column from step (35)). Save this value in an object called `NumLeaving`.

(38) Print `NumEntering` and `NumLeaving` to check your work.

We have two out of the three pieces of data needed to calculate net migration. The last piece of information we need is the total population in 2015 (the middle of the time period). We are going to pull this number from the US Census.

(39) Modify the code below to save the total population in your county for the year 2015. The variable code for total population is B01001_001.

```
my_county_population_2015 <- get_acs(geography = "______",
                                     state = "__",
                                     county = "_________",
                                     year = 2015,
                                     survey = "acs5",
                                     variables = "_______")
```

(40) Save just the `estimate` column in an object called `total_population_2015`. Print the result to check your work.

(41) Use the three objects you created in steps (36), (37), and (40) to calculate net migration according to the following equation:

$NetMigration = \frac{NumEntering - NumLeaving}{totalpopulation2015} \cdot 1000$

(42) Round the net migration rate you calculated to the ones place (no decimal points) and print the net migration you calculated. If the number is positive, this is the amount of people your county is growing by per 1000 people. If it is negative, that is how fast your county is shrinking per 1000 people.

For example, the net migration rate for Bear Lake County is -23 per 1,000 people. This means that for every 1,000 people in Bear Lake County in 2013, 23 people left by 2017. The population of Bear Lake County is shrinking.

(43) Use the statement above as a template to write about the net migration rate for your county. **(44)** Record the net migration rate of your county in the shared Google sheet.

Finishing up

(45) At the end of your report, copy this list and paste it outside of a code chunk to create bullet points. Fill in the data you calculated for your county. Make sure that all of this information is also in the team Google Sheet for comparison with other counties. If you calculated anything extra that not on this list, be sure to add it so you have a record for later.

```
- Percent employed in agriculture, 2017:
- Percent employed in manufacturing/warehousing, 2017:
- Percent employed in hospitality, 2017:
- Percent employed in technology, 2017:
- Percent retirees over 65, 2017:
- Net migration per 1,000 residents, 2013-2017:
```

(46) Go back through your report and add short explanations for what each code chunk does in your own words if you haven't done so already. **(47)** Render your report to a PDF and email it to [INSERT EMAIL HERE].

Statement of original and referenced work:

The entirety of this module is original work authored by Carolyn Koehn.

License