SI206 Project 1
Winter 2023

Introduction:

In this project, you will work with Comma-Separated Value (CSV) files. CSV files are very common and these foundational skills will enable more complex data analysis.

On data science and social justice: Data scientists use data to create actionable insights. Social justice is a broad term for several movements based on furthering equality and ending socioeconomic oppression. We can use data science in the pursuit of social justice by uncovering inequity so that we can act to correct it.

Data Description:

One data source for this assignment is the SAT Suite of Assessments for 2021**.** We extracted the race and ethnicity data listing the number of exam takers for the SAT exam. The columns of the data are of the different race and ethnicity groups that were measured, while the rows are of the different regions.

The other data is from the American Community Survey done by the US Census Bureau. The exact query and resulting data table can be found here. We reduced the data to just the columns you need.

The instructions, starter code, SAT data, Census data, and data dictionary are included when you clone the GitHub repository. These files can also be found on the Canvas site under Files > Projects > Project 1. The first row of data for both the SAT and Census data is header information.

Assignment:

You will create the following five functions and four corresponding test functions, and run them in *main()* in order to load, store, and analyze these data.

1. ***load_csv("filename")***

   load_csv takes one argument: a string that represents the name of a file. The function returns a dictionary in which each key is a region and each value is another dictionary. Each inner dictionary will use demographic categories as the keys and either the number of exam takers or number of people of that category in that region as the values. You must convert the numbers from strings to integers.

Example output:
When run on the SAT data it should produce a dictionary like this:
{“west”: {“AMERICAN INDIAN/ALASKA NATIVE”: 2091, “ASIAN”: ...},...}
When run on the Census data it should produce a dictionary like this:
{“west”: {“AMERICAN INDIAN/ALASKA NATIVE”: 1253113, “ASIAN”: ...}….}

test_load_csv()
tests ***load_csv***
Write a test case that checks for the length of the dictionary.
Write another test case that checks the number of test takers and/or the number of people for a region.

2. ***calc_pct(dict)***
   calc_pct will take in a dictionary of dictionaries. The function will iterate through that dictionary of dictionaries and return a dict of dicts where the inner key values are the proportion that each demographic category is of the region’s population.

   For the Census data, include each demographic’s regional population percent. For the SAT data, include each demographic’s regional test-taker percent. Remember that each inner dictionary value is the count of that demographic in that data set. Round your percentages to two decimal points.

   An example of this in general terms is:
   (White population of region/Region Totals) * 100 = percentage of the region’s population that is White

   test_calc_pct tests ***calc_pct***
   Write a test case that checks the calculated percentage for at least one race/ethnicity.

3. ***calc_diff(sat_dict, census_dict)***
   calc_diff will take two arguments. The first is a dictionary with the SAT data; the second is a dictionary with the Census data. For each demographic category in each region, you will calculate the difference between the two datasets by subtracting the values in the second dict from the values in the first dict. This function will return a double nested dictionary which contains the difference between each “cell” of the data. As a reminder, the SAT data contains a column

that won't be found in the Census data ("NO RESPONSE") so you'll have to ignore that. Round your percentages to two decimal points.

An example of this in general terms is:
(% of test takers of SAT that are white) - (% of population of region that is white) = the percentage difference between the population and test takes for that demographic.

test_calc_diff tests ***calc_diff***
Write a test case that checks the difference for at least one race/ethnicity.
Write a test case that checks if you're returning the absolute value of the difference.

4. ***write_csv(dict, "filename")***
   write_csv will take two arguments. The first is the dictionary that was produced through ***calc_diff*** and the second is the name for the output file in the format "proj1-yourlastname.csv"). The function will write the data from the dictionary into a csv file. The first column should contain the regions and the rest of the columns the percentages for each respective demographic category separated by commas. The first line of the file should be the header information and each row of data should be on a new line.

5. ***min_max(data)***
   min_max will take the argument of a dictionary. Use the provided ***min_max_mutate*** function to reformat the data into an easier-to-sort format. Your goal is to create a triple nested dictionary that will contain the regions with the largest and smallest differences between their demographics and the demographics of SAT test takers for each demographic. The function will return this new dictionary; please print it in *main()*. It will look like this:

   {"min": {"demographic": {"region": value}, ...},
     "max": {"demographic": {"region": value}, ...}...}

   Hint: you might use the *sorted() function* to make your job easier!

   test_min_max tests ***min_max***
   Write a test case that checks for the region with the largest difference for at least one race/ethnicity.
   Write a second test case that checks for the region with the smallest difference for at least one race/ethnicity.

6. **Questions:**

   Once you've completed the coding portion of this assignment, use the data you produced to answer the following questions. Data scientists think critically about how to turn data into actionable information – the programming and quantitative pieces are only part of the job. Turn in your answers to these questions as a PDF file in your Github repo along with your code and output. A few sentences for each question is fine.

   a. What story can you tell with this information? Does this story differ from your expectations? Why or why not?
   b. Think about the data sources for this information. Are there any limitations to the information based on these data sources? What information may be missing? What, if anything, is too generalized in these datasets?
   c. Think about potential audiences for your story (perhaps College Board, state/federal education departments, journalists, and more). How could you use this information to advocate for individual, organizational, and/or policy change?

7. **Extra Credit:**
   Create a pair of functions to calculate the national percentages for each dataset and compute the difference between them. Your output should not include the 'region totals' category.

   nat_pct(dict, col_list)
   Note: *col_list refers to the headers in the original spreadsheets loaded at the beginning of the homework. Most of these column names can be extracted from dict.*

   nat_pct will take in a dict (sat_data or census_data) and a list of columns (col_list) and then calculate the demographic percentages at the national level.

   nat_diff(dict1, dict2)
   nat_diff will take in the two percentage dictionaries you created with the ***nat_pct*** function and calculate the difference between each value in them.

   As with the ***calc_diff*** function above, make sure you know which values you subtract from which in order to appropriately interpret your results later. For example: if you subtract Census values from SAT values, a positive difference will mean the percentage of people in the given region of the given race was *higher* than the percentage of people in the given region of the given race who took the SAT. That would mean fewer than expected people of that race in that region took the SAT, and we can think about what may cause that difference.

Rubric:

As the syllabus states, each regular project is worth 200 points. The following rubric breaks down how you can earn each of these points.

Item	Points
load_csv + ***test_load_csv***	50 + 10
write_csv	40
calc_pct + ***test_calc_pct***	14 + 6
calc_diff + ***test_calc_diff***	14 + 6
min_max + ***test_min_max***	24 + 6
Reflection shows critical thought	30
Extra credit	20