Assignment 2 2024

Submission deadline: 11:59pm, Monday 20 May 2024.

Value: 20% of CITS4407.

To be done individually.

No AI tools are to be used for this assignment.

This assignment will involve creating two Shell scripts, which will use Unix tools and/or calls to other Shell scripts. The top-level scripts has been given specific names. Please make sure you use those script names, as these are the names which the testing software will use to test your script.

Put the top level scripts, plus any other scripts which you have created, into a directory, and then zip the directory, so your submission is a single package consisting of a zip file. An alternative is to use the Linux program tar to create a tar file from the directory. Submit the zip or tar file via *cssubmit*. **No** other file format will be accepted.

Measuring World Happiness

The idea of measuring the happiness across a nation started in Bhutan, which added Gross National Happiness, as a goal of government, to its constitution in 2008, as a response to the conventional Gross Domestic Product]. This was supported by a resolution of the United Nations in 2011, inviting Member States to share policies which aim to promote this goal. Since 2012, the Gallup organisation has been measuring happiness, understood as "satisfaction with one's life", via a series of world-wide polls, which ask respondents to rate their life satisfaction on a ladder from 0 (living worst life) to 10 (living best life) - the Cantril ladder.

The Data for this Assignment

Our World in Data is a truly excellent resource for high quality data across a wide range of topics. For example, during acute phase of the Covid19 pandemic, Our World in Data created and hosted many analyses of Covid19 disease data.

Relevant to this assignment are three, linked datasets from the Gallup world happiness surveys, averaged by country for a range of years, and then aligned with data from national sources related to GDP, homicide rates per 100,000 population, size of the population, and life-expectancy. Linked below, lightly cleaned tab-separated file (.tsv) versions

- gdp-vs-happiness.tsv
- homicide-rate-unodc.csv
- life-satisfaction-vs-life-expectancy.tsv

What you are to implement

This assignment mirrors some of the steps required for any data-science analysis. In particularly, you will be implementing a data-cleaning stage and an analysis stage.

Data Cleaning

You are to create a top level Bash script, called `cantril_data_cleaning` (Note: No suffix!), which will use Bash plus Shell tools, to clean the data.

`cantril_data_cleaning` expects three .tsv files corresponding to the three files from Our World in Data. The input files may be in any order. The output is expected to be a tab-separated data directed, as before, to standard output.

The overall program should, for a given datafile:

- Based on the header (i.e. top) line, make sure that the file is a tab-separated format file
- Also based on the header line, report any lines that do not have the same number of cells. (Cells are allowed be empty.)
- Remove the column with header Continent, which is sparsely populated and is not present in one of the files.
- Ignore the rows that do not represent countries (the country code field is empty)

- Ignore the rows for years outside those for which we have at least some Cantril data. (Cantril data may be absent in certain years within the range of those for which there otherwise is data; those cells should be retained.)
-

The output file sent to stdout should have rows with the data in the following order (tab separated):
<*Entity/Country*> <*Code*> <*Year*> <*GDP per capita*> <*Population*> <*Homicide Rate*> <*Life Expectancy*> <*Cantril Ladder score*>
While the contents of the input files may change, and the order that they are provided to cantril_data_cleaning may vary, you can assume that the order of the columns in the various input files will not change.
Hint: You will notice that the country year combination of cells is unique within each of the three input files.
Here is a sample output file: sample2.tsv

Finding the best predictor of world happiness.
You are to create a program where the top-level script is called `best_predictor`. `best_predictor` has a single input, the cleaned datafile produced by `cantril_data_cleaning`. What the program is seeking is the best predictor of Cantril-ladder life-satisfaction scores, based on the other data, which are thought to generally influence life-satisfaction across the community: GDP per capita, Population, Homicide Rate per 100,000 and Life Expectancy. (In the case of homicide rate per 100,000 population, it would presumably be an inverse correlation.) The method is: for each predictor, for each country, compute the correlation (also known as the Pearson correlation) between the predictor and the associated Cantril Ladder score. (The link provided includes both a definition of correlation, and a worked example.) The mean correlation for a given predictor is then computed across all the countries, and the best of the four predictors will have the largest absolute value.

Please bear in mind that, for the range of years of interest, the Cantril data is none-the-less missing for some countries, wholly or in part. That being the case, to include a correlation for any country there be at least 3 predictor-value, Cantril-value pairs. For example, `sample2.tsv` includes the data from Afghanistan, UAE, Bahamas and Oman, but only data from the first two countries should be included.

For example:

```
% ./best_predictor sample2.tsv
Mean correlation of Homicide Rate with Cantril ladder is
0.061
Mean correlation of GDP with Cantril ladder is -0.110
Mean correlation of Population with Cantril ladder is -0.835
Mean correlation of Life Expectancy with Cantril ladder is -
0.208
Most predictive mean correlation with the Cantril ladder is
Population (r = -0.835)
```

Marking Your Program

Runtime Testing

The program will be marked out of 20. Of the 20 marks, 14 (70%) will be awarded based on how the program deals with different types of input, both input that conforms to expectations and error state input that anti-bugging should catch. Please note that your program will be tested automatically. However, a human marker will be assessing your program's outputs for the range of tests, which means that the output format your program uses does not much matter for the auto-testing so long at the expected data is present.

You can assume that cleaned data-file submitted best_predictor is, in fact, clean (which is the point of that exercise).

Using Git

There will be a maximum of 2 marks for be for appropriate use of Git. The rubric is:

$0 \leq x < 1$	No use of Git or just token commands at one time
$1 \leq x < 2$	Git being used, but not informative, or just over one day

2	Multiple commits over time with appropriate/relevant commit messages

Style Rubric

The remaining 4 marks (20%) will be for style/maintainability. Programs are written as much for human as for computers. As such, it is important that your code be readable and maintainable. Similarly, outputs should aim to be informative (but never verbose). Much of this has been discussed in classes, but includes comments, meaningful variable names for significant variables (i.e. not throw away variables such as loop variables), and sensible anti-bugging. It also includes making sure your program removes any temporary files that were created along the way.

For the style/maintainability mark, the rubric is:

$0 \leq x < 1$	Gibberish, impossible to understand
$1 \leq x < 2$	Style is really poor, but can see where the train of thought may be heading
$2 \leq x < 3$	Style is acceptable with some lapses
$3 \leq x < 4$	Style is good or very good, with small lapses
4	Excellent style, really easy to read and follow

Important things to Note

- Automated testing is being used so that all submitted programs are being tested the same way. Sometimes it happens that there is one mistake in a program that means that no tests are passed. If the human marker is able to spot the cause and fix it readily, then they are allowed to do that, and your - now fixed - program will score whatever it scores from the tests, minus 2 marks, because other students will not have had the benefit of marker intervention. That's way better than getting zero for the run-time tess, right? (On the other hand, if the bug is too hard to fix, the marker needs to move on to other submissions.)
- **Most Important.** It's okay to develop your code on some other version of Linux, but make sure you test the final version, prior to submission, using the class Docker image or UniApps (Linux Lab).