Author: Atharva Mahajan

Mentor: Dr. Jamie Jennings

**Research Summary: Making CSV Safe for Spreadsheets**

**Abstract**

CSV files are ubiquitous in data storage and they are frequently used by spreadsheet applications like Microsoft Excel and Google Sheets. While it is a highly common practice to load and open CSV files into spreadsheet applications, it is rife with many issues and inconsistencies that are potentially problematic for the people working with the CSV files. These occur mainly due to the spreadsheet programs auto converting and interpreting data differently than intended which could lead to potential loss of data, incorrect data reading and unforeseen data inconsistencies which may go undetected and cause problems for people who work with the data. It is also a tedious task to look for all these problems for every element in a large data file and automating this task is certainly a good idea.

Some of the most common issues when working with CSV files are:

1. Other data like gene names being misinterpreted as dates by Excel. (Source: <https://www.biostat.wisc.edu/~kbroman/publications/dataorg.pdf>)
2. Long numbers, leading 0s being truncated resulting in loss of data. (Source: <https://www.onlinesurveys.ac.uk/help-support/common-csv-problems/>)
3. CSV files being mistaken for SYLK files. (Source: <https://superuser.com/questions/210027/why-does-excel-think-csv-files-are-sylk>)
4. Different locales having different formats and separators, particularly the Dutch use of semicolon as a separator. (Source: <https://news.ycombinator.com/item?id=12041655>)
5. CSV injection attacks due to excel formulas present in the file. (Source: <http://georgemauer.net/2017/10/07/csv-injection.html>)

For this project, we researched and found the above common CSV problems as experienced by people trying to load CSV files into Excel and we will write a program to combat these issues. We will try and find a mitigation technique that works for the data and which does not alter the data and cause any loss of data whatsoever. We will not change the value of the data at all and try to maintain it across several passes of the program. We will display an analysis to the user detailing the statistics of the file including the number of rows and columns, and the separator used in the CSV file. We will also detail the instances of each type of inconsistency and the row and column numbers of every data item which contained said inconsistency. We will also provide the user the option to either simply view the analysis or modify the file using our mitigation technique. The contents of the CSV file with all the modifications made, will be written to a different CSV file so as to preserve the original file, just in case.

**Contributions**

* Produced catalog of documented mis-representations of CSV data by Excel.
* Wrote a program to detect data subject to such mis-representation.
* Devised mitigation strategies to prevent Excel from mis-interpreting data.
* Extended our program to apply the mitigation strategies, producing a new CSV file with the same data but in a format which Excel will interpret without mis-representation.

We created a python program called datasci.py which reads in the name of the csv file along with any command-line switches and performs an analysis of any potential inconsistencies in the file that could be automatically converted by excel into an excel-specific data format. For example, excel auto converts anything that it perceives as a date format. This has been a problem with some gene-names which could be perceived as dates. This seems to be a problem in many datasets like genelists used by biologists which sometimes contain gene names like “sept54”, “march2” or “oct24” which are then interpreted as dates and converted to the appropriate format by excel, changing the data in the process which then leads to confusion and loss of data. We make use of [Rosie Pattern Language](https://rosie-lang.org/) and the inbuilt date patterns to detect any token data that could be interpreted as a date and perform an analysis on the given column to determine if the column primarily consists of dates. If it does, then we conclude that the given pattern is indeed a date type and leave it unchanged. If the column does not consist of dates, we performed a modification on the file. We set a default threshold of **0.8** for the ratio of actual dates and dates that could have been misinterpreted by excel with the total number of elements in the column. If it exceeds the given threshold, we ascertain that all the detected “misinterpretations” are actually legitimate dates and leave them unmodified. If the column does not contain any actual dates but only potentially illegitimate dates, we then have a higher threshold of **0.9** to leave them unmodified. We also convert all legitimate date formats to MM/DD/YYYY to maintain uniformity.

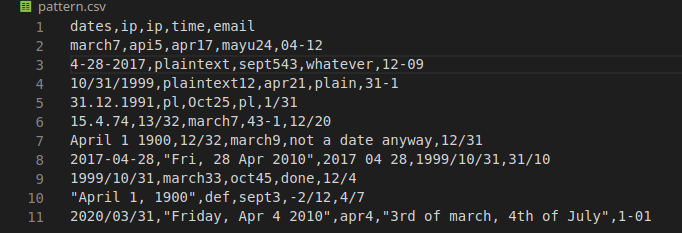
To combat the injection attack problem, we look for all items containing an ‘=’ sign and modify it so that excel does not automatically execute it when loading the csv files for the first time. This is to ensure that excel does not execute malicious injections written into the comma-separated file. If the user wishes to make any changes to the files afterwards, they can go ahead and change it back to the original formula so that excel can execute it.

Excel also tries to convert numbers longer than 15 digits into the scientific notation which is probably not something that people are looking for and it also truncates numbers with leading 0s and alters the value which results in a loss of data. We modify the data so that excel does not remove the leading 0s in the event they are essential and does not auto convert longer numbers into the scientific notation.

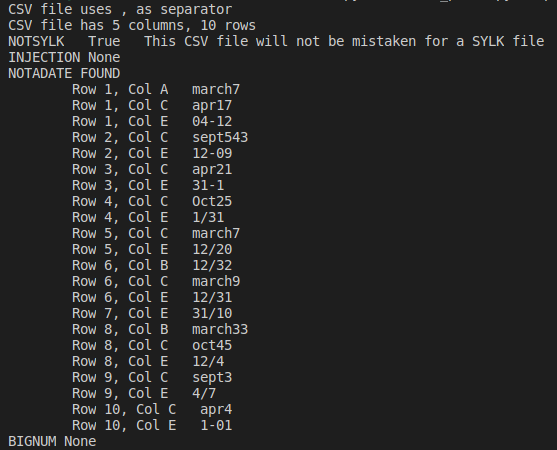
We also identify the separator being used by the csv file and include it in our analysis so the user is aware of it and knows what to select when loading the csv file into excel. Excel may misinterpret the csv file as being a SYLK file if the file begins with “ID”. We identify if this could be an issue and include that in our analysis.

When the program is run with no switch, it performs an analysis of the csv file and lists out all the inconsistencies found in the file and general information about the file. It stores this information in a separate text to retrieve every time the program is run. When the file is run with the -m switch, it performs an analysis of the file and makes modifications to the relevant data and writes to another file with the name “filename\_modified.csv”.

This is a sample file we used full of date inconsistencies:



This is the analysis produced by the program:



INJECTION refers to potential CSV injections as formulas in excel. NOTADATE refers to all the elements that are possibly not dates unless verified and BIGNUM refers to all elements that could be truncated. The mitigation technique we used to combat all these issues was to insert a “=” sign before the element and wrap quotes around it to force excel to treat it like a string while not changing its value.

**Future work**

We have identified the following extensions to this work:

* The user may want to change the thresholds used by the program to determine the best course of action for particular columns of data.
* Many other spreadsheets emulate what Excel does, and so our catalog of Excel mis-representations should be evaluated against the behavior of Google Sheets, Apple Numbers, and possibly other programs that import CSV data.
* In the long term, users may want to add their own ways of detecting certain types of data and performing transformations (mitigations) on them. This could be used for newly discovered Excel mis-representations, or for something like converting dates from an American format into a European or global one.