Perl Script to Column-Grep CSV File

Ananth

Summary

- Problem statement
- Requirements
- Examples

Problem Statement

- Say you have a CSV file with headers : Index,Country,State,Population
- You want to (say) filter so that you only retain rows with Population > 100 million
- Assuming that the Population column has natural numbers without commas, you could do

```
$ > perl -F, -ane ‘print unless $F[3] < 100000000;’
```

- We want something more user friendly
- We want user to be able to speak using Header names

Requirement Expressed Simply

- We want to take a CSV file and retain only the Header and the rows of interest (i.e., rows that match or rows that don’t match, depending on the need)

Requirements

- Header-row is the first fully-populated row (no empty fields) and with not more than one numeric field
  - `m,,,,,n,c,d,,,,,               # is not a header row ☺`
  - `1,Name,Age,DOB,Address         # is a header row`
  - `1,2,3,4,5                      # is not a header row`
- The entire "header" (all rows before the header-row and the header-row must be output by *default*) (this defines "header")
- Full Perl Compatible regex support is required for the matching if the operator is =~ or !~

Requirements : Input Format

- Command line use model will be

  ```
  $ > script.pl file [options] [criterion1 [options] criterion2 criterion3 ..]
  ```

- If no criterion is specified, then only the header is printed
- Criterion structure is

  <column-header-name> <operator> <operand>

  E.g. : **`Country = Antigua`** # treat both = and == the same

  E.g. : **`Name ‘=~’ ‘N*’`** # operand is a regex

  # operator enclosed in quotes to hide from shell

- Multiple criteria, if specified, are always combined in AND form; there is no support for OR

Errors

- Issue meaningful error-message and exit if column-header-name matches more than one field in the Header row (Eg, `-i Date = Carol` , but you have fields named `Date` and `date`)

Operator Support

Operator	Requirement
= and ==	User is allowed to say A = something even though that is an assignment in our world. Since the script is all about matching and searching, we support this. If it is determined that the operand is NOT numeric, then, internally, be sure to replace with eq; You can use Scalar::Util::looks_like_number if it helps
<, >, <=, >=	User is required to use quotes to prevent interpretation by shell Usual quantitative comparisons
!=	Similar to ==, if determined that the operand is NOT numeric, be sure to replace with neq
=~ and !~	In this case, the operand is a regex; User will use quotes to hide from shell

Requirements - Options

Option	Requirement
`-help`	Print help message - specify how to refer to single quote and double quote within a regex (since the entire regex is wrapped in ‘’ or “”) (\047 and \032)
`-noh`	No pre-header - do not print the rows before the header row (Header row WILL be printed)
`-i`	(only) Following criterion’s column-header name is to be treated case-insensitive
`-v`	Similar to grep - suppress the rows that match
`-l`	similar to grep’s -l : just report the column headers that match

Regular Expression Operand Support

- In perl, we use m/regex/ or just /regex/ or m#regex#, etc
- If the user specifies ‘regex’ (only quotes), it is the same as m/regex/
- If the user wants to specify options, (E.g. case insensitive match) then she is permitted ‘/regex/opts’ or ‘#regex#opts’ (E.g. ‘/regex/i’ would tell the script to match using regex in a case insensitive sense; ‘#regex#i’ is the same, but now allows the ‘/’ character to be used in the regex
- Full PCRE

Examples

- `$ > script.pl input.csv Name ‘=~’ ‘/u.*/i’ -i population ‘>’ 10000000`
  - Entry in Name column matches Ukraine, Uganda, etc (case insensitive)
  - Column whose header-name matches “population” (case-insensitive)
  - So, we print all rows where Name field matches something AND population field value is > something
- `$ > script.pl -noh input.csv Corner ‘!=‘ Nominal`
  - Print all rows for which the entry in the Corner field is NOT “Nominal”
  - Also suppress the “pre-header” (rows before the header-row, if any)
  - This is the same as saying `-noh -v input.csv Corner = Nominal`

-l and NaN

Use Model

```
$ > script.pl -l <operator> <operand>
```

Eg :

`script.pl -l = NaN` # report header-names of columns with ANY **non-numeric** data

`script.pl -l ‘!=’ NaN` # report header-names of columns with ANY **numeric** (i.e., Scalar::Util looks_like_number will return 1)

`script.pl -l ‘=~’ ‘/London/i’` # report header-names of columns that satisfy Perl’s regex match `m/London/i`

Interpretation

We report the names of the columns (i.e., the Header names) for which any item in the column matches the criterion. Please note the special cases for NaN, which should not be difficult to implement since `looks_like_number` is already being used

When -l is used, there is no regular output, the output is only a **SPACE separated list** of Header names of the columns that match

Substitution Support

Use Model	`$ > script.pl in.csv <col-header-name> <operator> ‘s/regex/substitution/opts’`
	Eg :
	`$ > script.pl in.csv Date ‘=~’ ‘s/2019/2020/g’`

Interpretation	Similar to Perl code. The criterion returns 1 (i.e, True or non-zero) only if the substitution actually happened - therefore, in this case, only the fields that underwent substitution will be in the final output

Use Model	`$ > script.pl in.csv Date ‘=~’ ‘s/2019/2020/k’    # NOTE the k!!!!`
Interpretation	The “k” in the substitution options means “keep” - so, this criterion runs the substitution but ALWAYS returns 1, so that, here, substitutions happen, but all the data for this column is retained (of course, combining with other criteria will affect the final output)

Please depend on Perl to implement the substitution so that sophisticated options like /e are supported

-help Output

```
$ perl -w ~/Public/perl/column_grep.pl -help
Usage: ./script.pl [-l operator operand] [-noh] [-v] [-help] file [ [option] column-name operator operand ...]
    -help - show this help (optional)
    -noh - hide preheader (optional)
    -v - similar to grep's -v - omit rows that matched (optional)
    -l - show only matched columns (similar to grep's -l)
    -i - the next supplied column-name will be used in case-insensitive fashion
    file - file name (required)
     operators supported (use quotes to hide from shell) : = (or ==), !=, =~, !~, <,>
     operands supported (use quotes) : regular expression (when operator is =~ or !~, (PCRE)
         /regex/opts -- if options are supplied (else no need for / / )
         #regex#opts -- if it is desirable the the regex contain /
         s/regex/substituion/opts -- full fledged perl compatible subsitution
                 note that for subsitution, /k is supported - that is "keep" - normally, the operator
                 (just like Perl) will return non-zero only if a match occurred. With /k you always return 1
         NaN -- non-numeric -- not the traditional NaN, but similar :)
                 Eg. -l = NaN    will list the names of columns that contain ANY numeric data
                 Eg. -l '!=' NaN   will list the names of columns that contain ANY non-numeric data
```