NAME

ExtractFromTextFiles.pl - Extract specific data from TextFile(s)

SYNOPSIS

ExtractFromTextFiles.pl TextFile(s)...

ExtractFromTextFiles.pl [-c, --colmode colnum | collabel] [--categorycol number | string] [--columns "colnum,[colnum]..." | "collabel,[collabel]..."] [-h, --help] [--indelim *comma* | *semicolon*] [-m, --mode *columns* | *rows* | *categories*] [-o, --overwrite] [--outdelim *comma* | *tab* | *semicolon*] [-q, --quote *yes* | *no*] [--rows "colid,value,criteria..." | "colid,value..." | "colid,mincolvalue,maxcolvalue" | "rownum,rownum,..." | colid | "minrownum,maxrownum"] [--rowsmode rowsbycolvalue | rowsbycolvaluelist | rowsbycolvaluerange | rowbymincolvalue | rowbymaxcolvalue | rownums | rownumrange] [-r, --root *rootname*] [-w, --workingdir *dirname*] TextFile(s)...

DESCRIPTION

Extract column(s)/row(s) data from *TextFile(s)* identified by column numbers or labels. Or categorize data using a specified column category. During categorization, a summary text file is generated containing category name and count; an additional text file, containing data for for each category, is also generated. The file names are separated by space. The valid file extensions are *.csv* and *.tsv* for comma/semicolon and tab delimited text files respectively. All other file names are ignored. All the text files in a current directory can be specified by **.csv*, **.tsv*, or the current directory name. The --indelim option determines the format of *TextFile(s)*. Any file which doesn't correspond to the format indicated by --indelim option is ignored.

OPTIONS

-c, --colmode *colnum* | *collabel*

Specify how columns are identified in *TextFile(s)*: using column number or column label. Possible values: *colnum or collabel*. Default value: *colnum*.

--categorycol *number* | *string*

Column used to categorize data. Default value: First column.

For *colnum* value of -c, --colmode option, input value is a column number. Example: *1*.

For *collabel* value of -c, --colmode option, input value is a column label. Example: *Mol_ID*.

--columns *"colnum,[colnum]..."* | *"collabel,[collabel]..."*

List of comma delimited columns to extract. Default value: First column.

For *colnum* value of -c, --colmode option, input values format is: *colnum,colnum,...*. Example: *1,3,5*

For *collabel* value of -c, --colmode option, input values format is: *collabel,collabel,...*. Example: *Mol_ID,MolWeight*

-h, --help

Print this help message.

--indelim *comma* | *semicolon*

Input delimiter for CSV *TextFile(s)*. Possible values: *comma or semicolon*. Default value: *comma*. For TSV files, this option is ignored and *tab* is used as a delimiter.

-m, --mode *columns* | *rows* | *categories*

Specify what to extract from *TextFile(s)*. Possible values: *columns, rows, or categories*. Default value: *columns*.

For *columns* mode, data for appropriate columns specified by --columns option is extracted from *TextFile(s)* and placed into new text files.

For *rows* mode, appropriate rows specified in conjuction with --rowsmode and rows options are extracted from *TextFile(s)* and placed into new text files.

For *categories* mode, coulmn specified by --categorycol is used to categorize data, and a summary text file is generated containing category name and count; an additional text file, containing data for for each category, is also generated.

-o, --overwrite

Overwrite existing files.

--outdelim *comma* | *tab* | *semicolon*.

Output text file delimiter. Possible values: *comma, tab, or semicolon*. Default value: *comma*

-q, --quote *yes* | *no*

Put quotes around column values in output text file. Possible values: *yes or no*. Default value: *yes*.

-r, --root *rootname*

New file name is generated using the root: <Root>.<Ext>. Default for new file names: <TextFile>CategoriesSummary.<Ext>, <TextFile>ExtractedColumns.<Ext>, and <TextFile>ExtractedRows.<Ext> for *categories*, *columns*, and *rows* mode respectively. And <TextFile>Category<CategoryName>.<Ext> for each category retrieved from each text file. The output file type determines <Ext> value: csv and tsv for CSV, and TSV files respectively.

This option is ignored for multiple input files.

--rows *"colid,value,criteria..."* | *"colid,value..."* | *"colid,mincolvalue,maxcolvalue"* | *"rownum,rownum,..."* | *colid* | *"minrownum,maxrownum"*

This value is --rowsmode specific. In general, it's a list of comma separated column ids and associated mode specific value. Based on Column ids specification, column label or number, is controlled by -c, --colmode option.

First line containing column labels is always written out. And value comparisons assume numerical column data.

For *rowsbycolvalue* mode, input value format contains these triplets: *colid,value, criteria...*. Possible values for criteria: *le, ge or eq*. Examples:

```
MolWt,450,le
MolWt,450,le,LogP,5,le,SumNumNO,10,le,SumNHOH,5,le
```

For *rowsbycolvaluelist* mode, input value format is: *colid,value...*. Examples:

```
Mol_ID,20
Mol_ID,20,1002,1115
```

For *rowsbycolvaluerange* mode, input value format is: *colid,mincolvalue,maxcolvalue*. Examples:

```
MolWt,100,450
```

For *rowbymincolvalue, rowbymaxcolvalue* modes, input value format is: *colid*.

For *rownum* mode, input value format is: *rownum*. Default value: *2*.

For *rownumrange* mode, input value format is: *minrownum, maxrownum*. Examples:

```
10,40
```

--rowsmode *rowsbycolvalue* | *rowsbycolvaluelist* | *rowsbycolvaluerange* | *rowbymincolvalue* | *rowbymaxcolvalue* | *rownums* | *rownumrange*

Specify how to extract rows from *TextFile(s)*. Possible values: *rowsbycolvalue, rowsbycolvaluelist, rowsbycolvaluerange, rowbymincolvalue, rowbymaxcolvalue, rownum, rownumrange*. Default value: *rownum*.

Use --rows option to list rows criterion used for extraction of rows from *TextFile(s)*.

-w, --workingdir *dirname*

Location of working directory. Default: current directory.

EXAMPLES

To extract first column from a text file and generate a new CSV text file NewSample1.csv, type:

```
% ExtractFromTextFiles.pl -r NewSample1 -o Sample1.csv
```

To extract columns Mol_ID, MolWeight, and NAME from Sample1.csv and generate a new textfile NewSample1.tsv with no quotes, type:

```
% ExtractFromTextFiles.pl -m columns -c collabel --columns "Mol_ID,
  MolWeight,NAME" --outdelim tab --quote no -r NewSample1
```

```
    -o Sample1.csv
```

To extract rows containing values for MolWeight column of less than 450 from Sample1.csv and generate a new textfile NewSample1.csv, type:

```
% ExtractFromTextFiles.pl -m rows --rowsmode rowsbycolvalue
  -c collabel --rows MolWeight,450,le -r NewSample1
  -o Sample1.csv
```

To extract rows containing values for MolWeight column between 400 and 500 from Sample1.csv and generate a new textfile NewSample1.csv, type:

```
% ExtractFromTextFiles.pl -m rows --rowsmode rowsbycolvaluerange
  -c collabel --rows MolWeight,450,500 -r NewSample1
  -o Sample1.csv
```

To extract a row containing minimum value for column MolWeight from Sample1.csv and generate a new textfile NewSample1.csv, type:

```
% ExtractFromTextFiles.pl -m rows --rowsmode rowbymincolvalue
  -c collabel --rows MolWeight -r NewSample1
  -o Sample1.csv
```

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

JoinTextFiles.pl, MergeTextFilesWithSD.pl, ModifyTextFilesFormat.pl, SplitTextFiles.pl

COPYRIGHT