# ColumnFileHelpers

**Version 1.0**

Carsten Kemena

September 2, 2014

This is a small and simple tool suite containing several small programs (well, currently only two :).

# Contents

# 1 colsum

*colsum* can sum up (or produce an average) of column-wise values. Several columns can be specified and it is furthermore possible to specify rows to take into account according to a given pattern.

## 1.1 Options

*colsum* supports many options to facilitate the use of it. In this section all of them explained.

### 1.1.1 General Options

This sections describes the general input and output options.

Table 1.1: General options which can be used with *colsum*.

| option | default | effect |
|---|---|---|
| `-h (--help)` | - | Produces a simple help message |
| `-i (--in)` | - | The input file. Data can be piped in as well |
| `-o (--out)` | - | The output file. If none is given, output is written to stdout |
| `-n (--no-header)` | *false* | Suppresses the printing of the header |
| `-r (--round)` | *2* | The number of digits to print after decimal point. |

#### Details

**i, in**: The file to analyze. When no file is set *colsum* reads from the pipe.

**o, out**: The output is written to the file given with this parameter. If no file is given, the output will be printed to the terminal. By default a header is printed containing the column ids.

**n, no-header**: No header information is printed.

**r, round**: The number of digits to print after decimal point. This value will be applied to all values produced.

#### Examples

The examples below show the usage of these basic input and output options (input files can be found in the test/colsum directory):

```
# use textfile as source and write to terminal
PROMPT: colsum −i test1.txt −c 2 −d " "
2
10.00

# pipe input into colsum and write result to "col1.txt"
PROMPT: less test1.txt | colsum −i test1.txt −c 2 −d " " −o col1.txt

# calculate average of column 4, print only one digit after decimal
    point and do not print a header
PROMPT: colsum −i test1.txt −d " " −c 4 −r 1 −a −−no−header
5.5
```

### 1.1.2 Input Format Options

This section described the options to manage the input format.

Table 1.2: Options of *colsum* for managing the format of the input.

| option | default | effect |
|---|---|---|
| `-d (--delimiter)` | *tab* | The delimiter to use |
| `-m (--merge)` | *false* | Consecutive delimiters are counted as only one |
| `-c (--columns)` | - | The columns to use |
| `-k (--grouping)` | 1 | The rows which should be summed up |
| `-p (--patterns)` | - | The patterns to use for summing up |
| `-l (--skip_lines)` | 0 | Skips the first $n$ lines of the file |

#### Details

**d, delimiter**: The delimiter to use, by default this is the '*tab*' character. Only one delimiter can be used at a time. Consecutive delimiters in the input are **not** merged **unless** the '-m/–merge' parameter is set. The column id is increased for each delimiter.

**m, merge**: If this option is set consecutive delimiters are treated as a single, thus the columns are only changed when encountering a non-delimiter.

**c, columns**: A string defining the columns to process. Values are separated by comma. Several consecutive columns can be set by combining the first and the last column by '-'. If a line starts with a delimiter the column after the delimiter is considered to be the second. The order of columns is unimportant, the output will **always** be ordered from first to last position.

**k, grouping**: This option defines which row to sum up into a single value. By default all rows are summed up into one value. If $k > 1$, $k$ values are computed, each containing the values of each k-th row (e.g. k=2: The first value will contain rows 1,3,5,... and the second 2,4,6,....

**p, patterns**: This parameter is used to specify rows matching a certain pattern. If several patterns are provided, the values of the rows are treated separately. If more than one patterns matches to the same row the values are used in each pattern. A pattern is a regular expression which is used to identify the rows to use. *colsum* will search for this patterns in the **whole** line. If the pattern occurs at any position the values of the specified colums will be used. The pattern can appear anywhere in the line (e.g. using 'day' as pattern, rows containing 'Friday' and 'Monday' will be summed up together even when they occur in different columns! Values of rows matching more than one pattern will be added to each pattern. Rows without any pattern are ignored. A column is added to the output identifying the group. The pattern is interpreted as a regular expression as specified in Perl. See the Boost webpage or the Perl webpage for more details. Each pattern has to be surrounded by ".

**l, skip_lines**: The first $n$ lines of the files are skipped and not taken into account.

### Examples

The following examples make use of these parameters (input files can be found in the test/colsum directory):

```
# sum up only column 2
PROMPT: colsum -i test1.txt -c 2 -d " "
2
10.00

# sum up column 3,5 and 7
PROMPT: colsum -i test1.txt -c 3,5,7 -d " " -o col1.txt
3 5 7
18.00 16.00 24.00

# sum up columns 3 and 5,6,7
PROMPT: colsum -i test1.txt -c 3,5-7 -d " " -o col1.txt
3 5 6 7
18.00 16.00 20.00 24.00
```

```
# merge delimiters
PROMPT: colsum -i test2.txt -c 4 -d " " -m -n
17.00

# add up every second row
PROMPT: colsum -i test2.txt -c 4 -d " " -m -k 2
4
5.00
12.00

# Summing up rows having the same pattern. A row value can be added to
    more than one pattern!
PROMPT: colsum -i test2.txt -c 2-3 -d " " -m -p Monday Frida day -a
group 2 3
Monday 2.50 5.50
Frida 2.50 4.50
day 2.50 5.00
```

### 1.1.3 Algorithmic options

This sections describes the behaviour of the program. Only one of the available options can be choosen. If none is choosen the sum is calculated.

Table 1.3: Algorithmic options which can be used with *colsum*.

| option | default | effect |
|---|---|---|
| `-a (--average)` | *false* | Calculates the average of all values |
| `--max` | *false* | Calculates the maximum of all values |
| `--min` | *false* | Calculates the minimum of all values |

#### Details

**a, average**: If set the average is caluclated instead of the sum.

**max**: Calculates the minimum.

**min**: Calcualtes the minimum.

# 2 csv2tex

*csv2tex* is a simple program to turn a columns file into a latex table.

## 2.1 Options

Table 2.1: Options which can be used with *colsum*.

| options | default | effect |
|---|---|---|
| `-h (--help)` | - | Produces a simple help message |
| `-i (--in)` | - | The input file. Data can be piped in as well. |
| `-o (--out)` | - | The output file. If none is given, output is written to stdout. |
| `-n (--no-header)` | - | Do not print a headline |
| `-d (--delimiter)` | *tab* | Delimiter |
| `-c (--columns)` | - | The columns to use |

#### Details

**i, in**: A file containing the table data. If nono is given the data has to be piped in.

**o, out**: The output is written to the file given with this parameter. If no file is given, the output will be printed to the terminal. The output does not come in any specific order.

**n, no-header**: Doesn't print a headline.

**d, delimiter**: The file delimiter to use.

**c, columns**: A string defining the columns to extract. If none is given all columns are extracted. Values are separated by comma. Several consecutive columns can be set by combining the first and the last column by '-'. If a line starts with a delimiter the column after the delimiter is considered to be the second. The order of columns is unimportant, the output will **always** be ordered from first to last position.

## Examples

```
cat test | csv2tex -c 1-2
```

# 3 rowmerge

*rowmerge* is able to combine multiple files into a single one.

## 3.1 Options

Table 3.1: Options which can be used with *colsum*.

| options | default | effect |
|---|---|---|
| `-h (--help)` | - | Produces a simple help message |
| `-i (--in)` | - | The input file. Data can be piped in as well. |
| `-o (--out)` | - | The output file. If none is given, output is written to stdout. |
| `-c (--col)` | - | Columns in file. |
| `-d (--delimiter)` | *tab* | Delimiter. |
| `-m (--merge)` | False | Consecutive delimiters are treated as one |

#### Details

**i, in**: The files to merge, can be any number but has to be at least two.

**o, out**: The output is written to the file given with this parameter. If no file is given, the output will be printed to the terminal. The output does not come in any specific order.

**c, col**: If not set the column which is merged is used merging is detected automatically. The first line is used to find equal names between the first two files to set the name identifier for all other files. If a number is given, it idetnifies the column to used to merge the files. If a single column is given it is used for every file, else the number has to be equal to the number of input files.

**d, delimiter**: The file delimiter to use.

**m, merge**: If set, several consecutive delimiters are merged into one.

# 4 rowFilter

*rowFilter* is a simple program to extract rows from a column file that fullfills certain requirements.

## 4.1 Options

Table 4.1: Options which can be used with *colsum*.

| options | default | effect |
|---|---|---|
| `-h (--help)` | - | Produces a simple help message |
| `-i (--in)` | - | The input file. Data can be piped in as well. |
| `-o (--out)` | - | The output file. If none is given, output is written to stdout. |
| `-d (--delimiter)` | - | Delimiter |
| `-f (--filter)` | - | The filterFunction to use |

### Details

**i, in**: The files to merge, can be any number but has to be at least two.

**o, out**: The output is written to the file given with this parameter. If no file is given, the output will be printed to the terminal. The output does not come in any specific order.

**d, delimiter**: The file delimiter to use.

**f, filter**: Description of the filter to use. It consists of three parts, the columnId, the comparison function, the threshold. For example "3>3.3" means that the the values in the third column have to be larget then '3.3'.

### Examples

```
rowFilter -f "2==3.4" "3>5" -i test.txt
```