Processing of delimiter separated values in ConTEXt

Tomáš Hála

1. Department of Informatics, Faculty of Business and Economics, Mendel University, Brno, Czech Republic;
2. KONVOJ, spol. s r. o. (publishing house), Brno, Czech Republic; Email addresses: `thala@mendelu.cz, konvoj@konvoj.cz`

It happens often that data for typesetting comes from other than TEX sources. In such a situation, the use of delimiter separated values belongs to quite frequent ways of data transfer.
Methods used for processing of delimiter separated values (including external files) will be reviewed across TEX implementations, with emphasis on ConTEXt solutions (m-database, HandleCSV, etc.). The related topic – computing à la spreadsheets (m-spreadsheet) – will be also covered.
Finally, some simple algorithms for easy extensions of the repertoir of tools will be presented, such as alternative reading and writing module, some simple computations on table/columns, and simple chart drawing (with lua and MetaPost).
The aim of this paper is not only to review existing tools, but also to open discussion about data processing, about users needs, and developers possibilities.

Introduction

It is supposed that tables contain data completely prepared for publication. Therefore, typesetting systems usually focus on ways how to *design* tables. E.g., ConTEXt contains several environments for markuping and designing tables or table data — tabulate, (old) table, natural tables (TABLE), xtables. Manual use of these tools is quite time consuming because of complicated markuping structure. Any automation or simplyfication of this process is welcome. However, essential to implementation is well organised data, preferably in any simple format, e.g. like CSV. For this situation, quite a comfortable approach is offered by the module `m-database`.
However, the content of tables can be used even for computations. Therefore, this paper reviews ways how table data, especially in CSV format, can be processed, and shows some simple algorithms extending the repertoir of tools with focus on some simple computations on table/columns, and simple chart drawing with lua and MetaPost.

CSV – Comma Separated Values

Basic description

CSV (Comma Separated Values) is a simple file format used for table data exchange. Briefly, one can describe this file format as a sequence of lines consisting of pieces of data which are separated by commas.
CSV format does not depend on encoding used in the text, so it can be used also with UTF-8.

Rules describing CSV (according to RFC)

While there are various specifications and implementations for the CSV format (e.g., Repici, 2004; Edoceo, 2004 or Raymond, 2015), there exists no formal specification. The only one document is RFC (Shafranovich, 2005) which describes seven basic rules which should be followed when implementing the CSV:

1. *Each record is located on a separate line, delimited by a line break (CRLF).*

```
Nasbinal,F,2015-9th<CRLF>
Kalenberg,NL,2016-10th<CRLF>
```

2. *The last record in the file may or may not have an ending line break.*

```
Nasbinal,F,2015-9th<CRLF>
Kalenberg,NL,2016-10th
```

3. *There maybe an optional header line appearing as the first line of the file with the same format as normal record lines. This header will contain names corresponding to the fields in the file and should contain the same number of fields as the records in the rest of the file (the presence or absence of the header line should be indicated via the optional "header" parameter of this MIME type).*

```
Place,State,Year<CRLF>
Nasbinal,F,2015-9th<CRLF>
Kalenberg,NL,2016-10th
```

4. *Within the header and each record, there may be one or more fields, separated by commas. Each line should contain the same number of fields throughout the file. Spaces are considered part of a field and should not be ignored. The last field in the record must not be followed by a comma.*

5. *Each field may or may not be enclosed in double quotes. If fields are not enclosed with double quotes, then double quotes may not appear inside the fields.*

```
"Nasbinal","F","2015-9th"<CRLF>
Kalenberg,NL,2016-10th
```

6. *Fields containing line breaks (CRLF), double quotes, and commas should be enclosed in double-quotes.*

```
"Nasbinal -- horse<CRLF>
riding center","F","2015-9th"<CRLF>
"Kalenberg, in the north",NL,2016-10th
```

7. *If double-quotes are used to enclose fields, then a double-quote appearing inside a field must be escaped by preceding it with another double quote.*

```
"Nasbinal ""-- horse riding center""",
  "F","2015-9th"<CRLF>
Kalenberg,NL,2016-10th
```

TSV and DSV

Some languages use comma for delimiting integer and fraction parts of rational numbers. However, these commas clash with commas separating data fields. This was one of the reasons why other characters for separating values have been used. Very often, especially in currently used spreadsheets, we can find semicolon. Even in this case the format is still called CSV. If data fields are separated by tabulator we talk about TSV (Tab Separated Values).
We should not forget to mention the abbreviation DSV (Delimiter Separated Values) which stands for general name for all varieties of data with delimiters.

Ways of data processing

Old, simple, but still usefull

For the following example let us assume these table data:

```
Brejlov:CZ:2013-7th:
Bassenge-Boirs:B:2014-8th:
Nasbinals:F:2015-9th:
Kalenberg:NL:2016-10th:
```

If the number of columns and delimiting characters is known in advance, one might prepare macros describing records (=lines):

```
\def\myline#1:#2:#3:{\bTR
  \bTD#3\eTD\bTD#2\eTD\bTD(#1)\eTD
\eTR}
```

Natural table is used in this example, but generally any other method can be used.

```
\bTABLE
\myline Brejlov:CZ:2013-7th:
\myline Bassenge-Boirs:B:2014-8th:
\myline Nasbinals:F:2015-9th:
\myline Kalenberg:NL:2016-10th:
\eTABLE
```

Result:

2013-7th	CZ	Brejlov
2014-8th	B	Bassenge-Boirs
2015-9th	F	Nasbinals
2016-10th	NL	Kalenberg

The necessity of some small pieces of manual work is a disadvantage of this method. For more comfortable processing, the last piece of data is followed by one extra delimiter. As explained later, this does not correspond to CSV definition.
On the other hand, this method enables more than one delimiter:

```
\def\myline#1:#2:#3-#4:{\bTR
   \bTD#4\eTD\bTD#3\eTD
   \bTD#2\eTD\bTD(#3)\eTD
\eTR}
```

Result:

CZ	7th	2013	Brejlov
B	8th	2014	Bassenge-Boirs
F	9th	2015	Nasbinals
NL	10th	2016	Kalenberg

Current implementation of CSV data processing in ConTEXt

Module m-database

Author	*Hans Hagen*
Available at	*TEXlive, ConTEXtGarden*
Version	*2010.08.04, lua file 1.001*
State	*works*
Development	*on demand*
Documentation	*no, only Mojca's short note [Miklavec, 2006]*

The goal of this module is to make the work with CSV files user friendly. The following example shows how this module can be used. First, we have to define the behaviour of the environment CSV:

```
\usemodule[database]
\defineseparatedlist
  [CSV]
  [separator=;,
   quotechar=",
   before=\bTABLE,
   after=\eTABLE,
   first=\bTR,
   last=\eTR,
   left=\bTD,
   right=\eTD
]
% ...............................
\startseparatedlist[CSV]
Brejlov;CZ;2013;7th
Bassenge-Boirs;B;2014;8th
Nasbinals;F;2015;9th
Kalenberg;NL;2016;10th
\stopseparatedlist
% ................... or ........
\startCSV
Brejlov;CZ;2013;7th
```

```
Bassenge-Boirs;B;2014;8th
Nasbinals;F;2015;9th
Kalenberg;NL;2016;10th
\stopCSV
```

As shown above, there are two ways how to open and close the CSV environment – `\startseparatedlist[CSV]`, and shorter (and probably more comfortable) `\startCSV` are both generated automatically. We receive the following result:

Brejlov	CZ	2013	7th
Bassenge-Boirs	B	2014	8th
Nasbinals	F	2015	9th
Kalenberg	NL	2016	10th

The module works properly, only one incompatibility with the specification of CSV has been detected – multi-line data fields are not supported (rule 6, see p. 59).

Application of m-database: Conversion of LaTeX Tables

If line organisation of tables is used in a source code, LaTeX tables can also be considered as a CSV data because data fields are separated by character &.
The following example is taken from the textbook which was published in 1999 and 2003, and typeset in LaTeX 2.09 and LaTeX2ϵ, respectively. The new Czech edition which is just under the preparation will, of course, be typeset in ConTeXt. Therefore, all ca 50 tables should be converted from LaTeX to ConTeXt. However, as explained above, the content (rows and cells) of LaTeX of tables behaves as CSV data. This is the reason why the module m-database can be used in this situation and thus avoiding greater part of exhausting manual work.
The original source code looks as follows (`\hp` is a shortcut for `\hphantom`):

```
\begTAB{Longest-lived isotopes of transfermium elements}{trfe}{%
\begin{center}\setlength{\tabcolsep}{1mm}\def\arraystretch{1.2}
\small\sf\begin{tabular}{|c|l|l|c|l|} \hline
\strut
Z   & Nucleon numbers          & isotop with  & T (s)               & Produced in\\
    & of known isotopes (A)    & the longest half-life (T)         & & the reaction\\[2.5mm]\hline\hline
101 & 248--259                 & \(^{258}\)Md & 55 days             & \(^{255}\)Es(\(\alpha\),n)\\\hline
102 & 250--259                 & \(^{255}\)No & 185\hp{,00000}      & \(^{244}\)Pu(\(^{16}\)O,~5n)\\\hline
103 & 252--262                 & \(^{256}\)Lr & \hp{0}45\hp{,00000} & \(^{243}\)Am(\(^{18}\)O,~5n)\\\hline
104 & 253--262                 & \(^{261}\)Rf & \hp{0}65\hp{,00000} & \(^{248}\)Cm(\(^{18}\)O,~5n)\\\hline
105 & 255--258, 260--263       & \(^{262}\)Db & \hp{0}34\hp{,00000} & \(^{249}\)Bk(\(^{18}\)O,~5n)\\\hline
106 & 258--261, 263, 265, 266  & \(^{263}\)Sg & \hp{00}0,9\hp{0000} & \(^{249}\)Cf(\(^{18}\)O,~4n)\\\hline
107 & 261, 262, 264            & \(^{262}\)Bh & \hp{00}0,0061\hp{0} & \(^{209}\)Bi(\(^{54}\)Cr,~2n)\\\hline
108 & 264, 265, 267, 269       & \(^{269}\)Hs & \hp{0}19,7\hp{0000} & from \(\alpha\) decay\(^{273}\)Ds\\\hline
109 & 266, 268                 & \(^{266}\)Mt & \hp{00}0,0034\hp{0} & \(^{209}\)Bi(\(^{59}\)Fe,~n)\\\hline
110 & 269, 271--273            & \(^{269}\)Ds & \hp{00}0,0027\hp{0} & \(^{208}\)Pb(\(^{62}\)Ni,~n)\\\hline
111 & 272                      & \(^{272}\)111& \hp{00}0,0015\hp{0} & \(^{209}\)Bi(\(^{64}\)Ni,~n)\\\hline
112 & 272                      & \(^{272}\)112& \hp{00}0,00028      & \(^{208}\)Pb(\(^{70}\)Zn,~n)\\\hline
\end{tabular}\end{center}}{}
\endTAB
```

In the given example, there are some LaTeX macros which have to be supressed:

```
\def\hline{}
\def\\{}
```

In the LaTeX editions of the textbook, the mathematical mode was actived by macros `\(` and `\)` because these macros can be in LaTeX simply redefined. Unfortunately, both macros are specific for LaTeX only and for use in ConTeXt have to be redefined:

```
\def\({\startluacode
      context("$") \stopluacode}
\def\){\startluacode
       context("$") \stopluacode}
```

Let us prepare the definition of separated list for LaTeX tables:

```
\usemodule[database]
\defineseparatedlist[LTX][
   separator=&,
   before=\bTABLE,after=\eTABLE,
   first=\bTR,last=\eTR,
   left=\bTD,right=\eTD,
   setups=unix
]
```

Even so, some specific macros will have to be manually erased or deactivated (first four and last two lines).

After conversion and typesetting, the table looks as follows:

Z	Nucleon numbers of known isotopes (A)	isotop with longest half-life (T)	T (s)	Produced in the reaction
101	248–259	^{258}Md	55 days	^{255}Es(α,n)
102	250–259	^{255}No	185	^{244}Pu(^{16}O, 5n)
103	252–262	^{256}Lr	45	^{243}Am(^{18}O, 5n)
104	253–262	^{261}Rf	65	^{248}Cm(^{18}O, 5n)
105	255–258, 260–263	^{262}Db	34	^{249}Bk(^{18}O, 5n)
106	258–261, 263, 265, 266	^{263}Sg	0,9	^{249}Cf(^{18}O, 4n)
107	261, 262, 264	^{262}Bh	0,0061	^{209}Bi(^{54}Cr, 2n)
108	264, 265, 267, 269	^{269}Hs	19,7	from α decay^{273}Ds
109	266, 268	^{266}Mt	0,0034	^{209}Bi(^{59}Fe, n)
110	269, 271–273	^{269}Ds	0,0027	^{208}Pb(^{62}Ni, n)
111	272	^{272}111	0,0015	^{209}Bi(^{64}Ni, n)
112	272	^{272}112	0,00028	^{208}Pb(^{70}Zn, n)

ScanCSV

Author	*Jaroslav Hajtmar*
Presented at	*TEXperience 2012, Morávka, Czech Republic*
	ConTEXt Meeting 2013, Brejlov, Czech Republic
Published	*Zpravodaj CSTUG, 2012, vol. 2*
Available	*hajtmar.com?*
State	*obsolete*
Use	*private Jarda's projects for his needs at work (gymnasium)*
Use now	*rather not*

ScanCSV is a library written in lua (t-scancsv.lua).[1] For simplifying user's work, the direct use of lua is not required (compare with m-spreadsheet, page 64).

The principle of work is based on parsing input data and its subsequent automated transformation to TEX macros. These macros enable access to each cell (column) of a given row. They are very similar to spreadsheet notation of columns – `\cA`, `\cB`, `\cC`, etc.

Therefore, user has to prepare only his own macro called `\lineaction` designing the output. In this macro it is possible to use automatically generated macros. After processing the line, the next line is read and transformed, and this procedure is repeated until the whole table is processed.

HandleCSV

Authors	*Pablo Rodriguez + Jaroslav Hajtmar*
Presented	*not, 'private project', GitLab*
State	*under the development*
Documentation	*not published*
Use	*private projects of authors*

The library HandleCSV, also written in lua (files t-handlecsv.lua and t-handlecsv-tools.lua), is the successor of ScanCSV.

Because of close relation between libraries ScanCSV and HandleCSV the following example is common for both.

```
Place,State,Year,Number
Nasbinal,F,2015,9th
Kalenberg,NL,2016,10th
```

```
\usemodule[handlecsv]
\setheader
\opencsvfile{a.csv}
\starttext
\startbuffer[loop]
\cD\ \ConTeXt\ Meeting \cC\
  \crlf\qquad took place at \cA\
(\cB).\crlf
\stopbuffer
\doloop{\getbuffer[loop]\nextrow%
       \ifEOF\exitloop\fi}
\stoptext
```

Macro `\setheader` indicates that the first row contains no data, but headings of the table.

Result:

9th ConTEXt Meeting 2015
    took place at Nasbinals (F).
10th ConTEXt Meeting 2016
    took place at Kalenberg (NL).

[1] Before writing the t-scancsv.lua file for ConTEXt MkIV, the library was written generally for ConTEXt as well as for LuaLATEXand LuaTEX.

Module m-spreadsheet

Author	*Hans Hagen*
Available	*TEXlive, ConTEXtGarden*
Version	*2011.02.21, lua file 1.001*
State	*works, maintained*
Development	*on demand*
Documentation	*spreadsheets-mkiv.pdf*

```
\usemodule[spreadsheet]
```

While the ScanCSV and HandleCSV work only with data and do not require Lua code as a content of cells, the module m-spreadsheet supposes that Lua code will appear for definition of all cells. As shown below (Hagen, 2016), then column notation is taken from speadsheet (A, B, etc.) whereas row numbers have to be used as indexes because of inner array data structure.

```
\startspreadsheettable[test]
  \startrow
    \startcell 1.1       \stopcell
    \startcell 2.1       \stopcell
    \startcell A[1] + B[1] \stopcell

  \stoprow
  \startrow
    \startcell 2.1       \stopcell
    \startcell 2.2       \stopcell
    \startcell A[2] + B[2] \stopcell
  \stoprow
  \startrow
    \startcell A[1] + B[1] \stopcell
    \startcell A[2] + B[2] \stopcell
    \startcell A[3] + B[3] \stopcell
  \stoprow
\stopspreadsheettable
```

This module can be recommended for some simple computations, e.g., invoices, costings, quotations, etc. The documentation brings more sophisticated examples – cells containing longer Lua code with definition of function(s) or cells with formatting the output etc.
The m-spreadsheet module enables the use of computation tools in natural tables. In this case it is not possible to write pieces of lua code directly (natural tables works in TEX mode), and the user has to put lua code as a parameter of special macro:

```
... \bTD \getspr{A[1] + B[1]} \eTD
...
```

However, the processing of table runs from the top down and from left to right which restricts user to put data in left columns and computations right from data. Therefore, the following example will not work because while processing the first line, the values B[1] and C[1] have not been defined yet:

```
\startrow
  \startcell B[1] + C[1] \stopcell
  \startcell 2.1        \stopcell
  \startcell 2.2        \stopcell
\stoprow
```

CSV Data: data stream from a parameter

Some three years ago, I had to prepare the annual report for one non-profit organisation. Among other things, this report contained table and graphic presentation of selected economic data.
At the begining the customer was not sure of either which data will be presented, or how will be presented.
For practical use, some general structure, with simple access, e.g., array with indexes, and comfortable for future editing was requiered. Moreover, simple access must make computations for the graphical representation possible. This requirement excluded the use of m-database.[2]

[2] Module m-spreadsheet should possibly be used but the author (of this paper) did not know about it at that time.

The former idea how data should be processed was based on reading of twodimensional array.

```
\tabulka[title="Department A",
data=
RS 2517000 2515000 2386000 1954000
PU 4.57 4.62 4.65 4.05
PKL 361 491 392 268
PK 6699 5508 9475 6012
]
```

This experiment was not succesful. While reading data, the TEX input processor changes character `<lf>` for a space, so it is not important how data are written in this case. One might use the following with the same result:

```
\tabulka[
title="Department A",
data=RS
2517000
2515000
2386000
1954000
PU
4.57
4.62
4.65
4.05
PKL
361
491
392
268
...]
```

Fortunately, in this task all tables had the same number of columns. This means that – regardless to the organisation of input data – we are able to read the structure correctly.

```
\def\tabulka[#1]{
  \ctxlua{userdata.tabulka('#1')}}
```

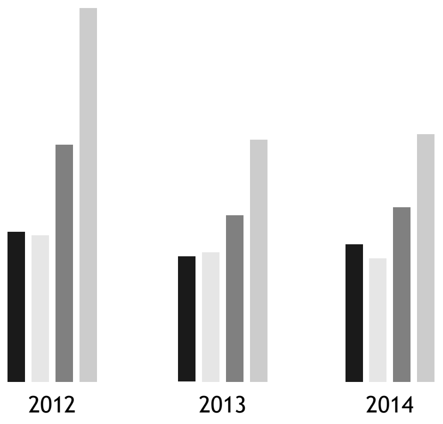
For drawing, the following macros have been prepared:

```
\def\grafwidth{0.5}
\def\sqrwidth{0.75}
\def\sqrindent{\null\kern\sqrwidth
cc\ }
\def\compute#1{\ctxlua{context(#1)}}
```

```
\def\graf#1#2{%
  \def\koef{\csname#1koef\endcsname}
  \startcolor[#1color]
  \vrule width\grafwidth cc height
#2dd\
  \stopcolor}

\def\ctverec#1{\startcolor[#1]
   \vrule width\sqrwidth cc height
0.625cc depth0.125cc
   \stopcolor}
```

Department A



	2012	2013	2014
■ Service budget	2 386 000	1 954 000	2 171 000
□ Number of workloads	4,65	4,05	3,84
▨ Number of clients	392	268	282
▨ Number of contacts	9 475	6 012	6 162

Data on an external file

While the previous part dealt with data written as a parameter, this part shows data stored on an external file.
The presented modules enable processing of data files. For HandleCSV (and ScanCSV as well) we can extract from the complete example (p. 63):

```
\opencsvfile{filename.csv}
```

Module m-database uses a similarly easy access available (Miklavec, 2006):

```
\processseparatedfile
  [TSV][filename.csv]
```

Because of some restrictions discussed above and of the requierement of simple access to data, I have decided to write my own set of tools for the following task with graphical presentation.

CTX-CSV

Author	*TH*
Available at	*????*
Version	*0.01, 2016.01.17*
State	*works*
Development	*as needed and on demand*
Documentation	*will be available later*

Basic Manipulation with CSV

Module t-csv contains four functions for basic work with CSV data (kw=key words, kv=key values):

```
function userdata.CSV.Read (kw, kv)
function userdata.CSV.Write (kw, kv)
function userdata.CSV.TABLE (kw, kv)
function userdata.CSV.NewColumn (kw,
kv)
```

The following examples demonstrate their use.

```
\CSVRead[][data=input.csv,separator=;]
\CSVWrite[][output=out.csv,
  enclosechar="]
\CSVTABLE[][enclosechar="]
\CSVNewColumn[][table=1,column=23]
```

For real values, some languages use decimal comma instead of decimal dot as in English. It will not be not user friendly if one has to convert data manually before processing. Therefore, use of both, comma and dot, is possible. Two functions which are used as key values in the first parameter, are defined for the required conversion:

```
function comma2dot(inputstr)
function dot2comma(inputstr)

\CSVRead[comma2dot][
       data=input.csv,separator=;]
\CSVWrite[dot2comma][
       output=out.csv,enclosechar="]
```

Assume the following CSV data:

```
F;;Kate;2;45;40,0;75,0
M;;Luke;1,5;43;30,0;71,7
M;;Bernie;3;39;60,0;65,0
M;;David;1,5;48,5;30,0;80,8
F;;Lis;1;44;20,0;73,3
...
```

Graphic representation of data is closely related with statistics. At first, some basic pieces of one-dimensional statistical data are computed (e.g., from column 4):

```
\CSVComputeColumn[][table=1,
   column=4,
   meze={0,5},
   startrow=3]
```

The distribution of these values shows the following command:

```
\CSVDrawDistribution[][
   table=1,
   column=4]
```

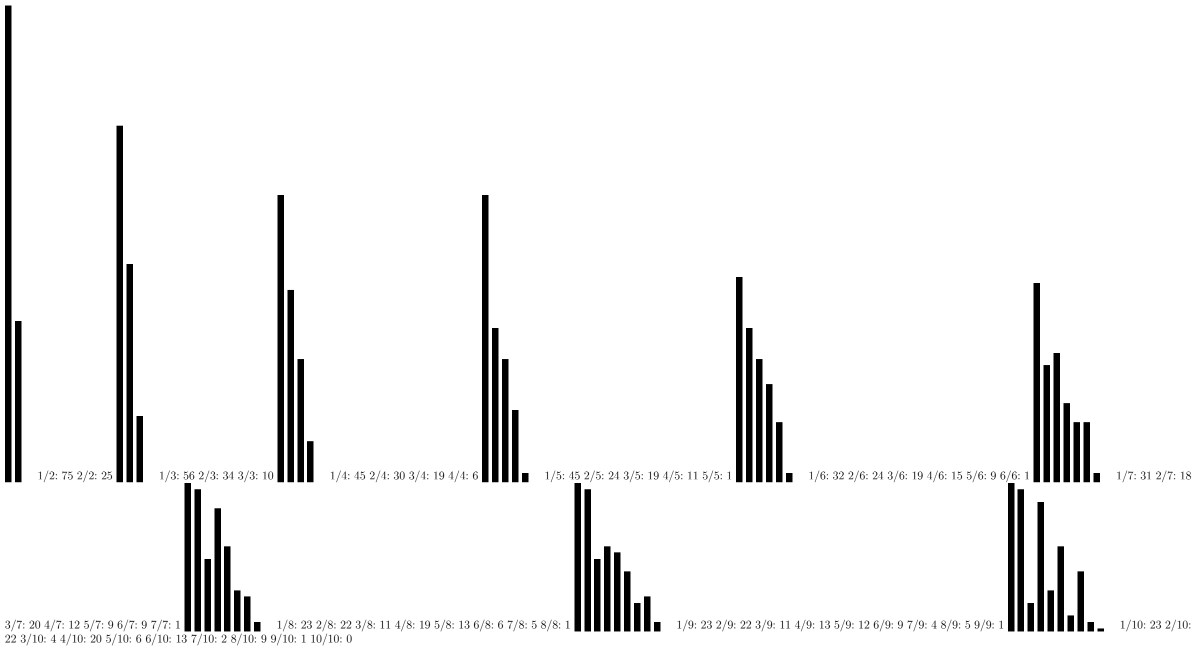
The two-dimensional distribution gives:

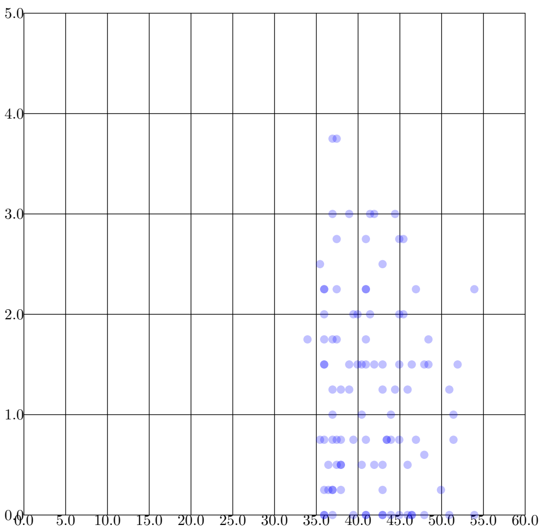
```
\CSVDrawCluster[][
   table=1,
   columns={4,5},
   startrow=3]
\CSVDrawCluster[sort,pergroups][
   table=1,
   columns={4,5},
   startrow=3,
   groupby=1,
   palette={red,black}]
```

Colors are taken from the predefined palette, but as shown above, users can set their own set of colors.

CTX-STAT: Output

Sum	125.1
Count	101
Average	1.2386138613861
Modus	0
Median	1.25
Min	0
Max	3.75
Variance	0.92113273208509
Standard deviation	0.95975660043841
Coefficient of variation	77.486344240031





Conclusion

This paper reviewed existing tools for CSV data processing. Some of them are publicly available and well documented, some of them are not because they have been created as private projects. Even so, they contain interesting tools for data processing.

The disadvantage is that the tools are not communicating with each other, so for cases with special requirements one has to write his/her own solutions. This was the reason why the project ctx-csv has been opened and special tools for data manipulation and graphical presentation have been developed. Typically for private projects, only necessary elements exists Other kinds of charts will be added on demand.

The question is whether the mentioned approaches can be generalised and their development coordinated. For easy user's work the one common way might to be better than many tools with different syntax or handling.

References

Edoceo, Inc.. *CSV Standard File Format* [on-line]. 2004. [cit. 2015-01-09 20:51]. Available at: http://www.edoceo.com/utilis/csv-file-format.php.

Hagen, Hans. *Simple Spreadsheets* [on-line]. Hasselt (NL) : Pragma ADE, 2016-05-12. [cit. 2017-01-30]. 11 pp. Available at: http://www.pragma-ade.nl/general/manuals/spreadsheets-mkiv.pdf.

Hajtmar, Jaroslav. [ScanCSV – Lua Library for processing of CSV files in ConTEXt and LuaLATEX]. *Zpravodaj CSTUG*, 2012, Vol. 2, p. 76–90. (ISSN 1211-6661.)

Miklavec, Mojca. Creating tables using CSV (comma–separated values) [on-line]. *My Way*, 2006, July 26, p. 1–7.. [cit. 2016-08-03]. Available at: http://dl.contextgarden.net/myway/csv.pdf.

Raymond, E. *The Art of Unix Programming*. Boston : Addison-Wesley, September 2003, p. 1–7 [cit. 2015-01-09]. Chapter 5 Raymond, E. *Data File Metaformats* [on-line]. Available at: http://www.catb.org/~esr/writings/taoup/html/ch05s02.html.

Repici, J. *HOW-TO: The Comma Separated Value (CSV) File Format* [on-line]. 2004. [cit. 2016-09-01]. Available at: http://www.creativyst.com/Doc/Articles/CSV/CSV01.htm.

Rodríguez, Pablo; Hajtmar, Jaroslav. *HandleCSV* [on-line]. 2016. [cit. 2016-06-29]. Available at: http://www.handlecsv.tk.

Shafranovich, Y. *RFC 4180 – Common Format and MIME Type for Comma-Separated Values (CSV) Files* [on-line]. October 2005. [cit. 2016-09-01]. Available at: https://tools.ietf.org/html/rfc4180#page-2.