csvtools.sty v1.1 : A LaTeX 2ε Package Providing Access to Data Saved in a CSV File

Nicola Talbot

19 October 2004

Contents

1 Introduction **2**

2 Mail Merging and Similar Applications **2**

3 Converting CSV file into a tabular environment **4**

4 Converting CSV file into longtable environment **6**

5 Associated Counters **7**

6 Cross-Referencing **8**

7 Saving Entries **11**

8 The csvtools.pl Perl Script **13**
8.1 Notes 13

9 Bugs/Drawbacks/"Features" **14**

10 Contact Details **15**

List of Examples

1 Mail Merging **2**

2 Multiple Figures **3**

3 Mail Merging using `\field` **3**

4 Aligning Data from a CSV file **4**

5 Adding Lines **5**

6 Added Complexity **5**

7 Using a longtable environment **6**

8 Stripy Table **7**

9 More Mail Merging **8**

10 Labelling within `\applyCSVfile` **8**

11 Labelling within `\applyCSVfile` **8**

12 Labelling within `CSVtotabular` **9**

13 Labelling within `CSVtotabular` **10**

14 Saving Entries **11**

15 csvtools.pl — Aligning Data **13**

1 Introduction

The `csvtools` package allows you to repeatedly perform a set of LaTeX commands on data in each row of a comma separated variable (CSV) file. This can be used for mail merging, generating tables etc.

2 Mail Merging and Similar Applications

`\applyCSVfile` `\applyCSVfile[`⟨*n*⟩`]{`⟨*filename*⟩`}{`⟨*text*⟩`}`

Letters can be generated using data given in each line from ⟨*filename*⟩. If the CSV file contains a header row, the unstarred version of `\applyCSVfile` should be used, otherwise the starred version `\applyCSVfile*` should be used. The optional argument ⟨*n*⟩ specifies on which line the actual data (not header line) starts. The unstarred version defaults to line 2 (the header row is always assumed to be on line 1) and the starred version defaults to 1.

`\insert...` With the unstarred version, the entries in the header row are used to generate commands of the form `\insert`⟨*identifier*⟩[1] to access corresponding elements in the row currently being processed. For example, suppose the first line of the CSV file looks like:

```
Name,Address,Time,Date
```

`\insertbyname` then the commands `\insertName`, `\insertAddress`, `\insertTime` and `\insertDate` are created, allowing you to use the entries in the first, second, third and fourth columns of the current row. If the header text contains non-alphabetical characters, e.g. `Full Name`, then you will need to use `\insertbyname{`⟨*text*⟩`}`, e.g. `\insertbyname{Full Name}`.

Example 1 (Mail Merging)

Suppose there is a file called `details.csv` that has the following contents:

```
Name,Address,Time,Date
```

[1]See Note 1 in Section 9

```
Miss A. Person,1 The Road\\The Town\\AB1 2XY,15.00,4th May 2004
Mr A. N. Other,2 The Road\\The Town\\AB1 2XY,15.30,11th May 2004
```

then the following code can be used to generate a letter for each person in the CSV file:

```
\applyCSVfile{details.csv}{%
\begin{letter}{\insertName\\\insertAddress}
\opening{Dear \insertName}

You are invited to an interview at \insertTime\ on the \insertDate.

\closing{Yours Sincerely}
\end{letter}}
```

Note that you could also use `\insertbyname{Name}` etc instead of `\insertName` etc. Also note that you need to specify the file extension when specifying the filename.

Example 2 (Multiple Figures)

Suppose `details.csv` looks like:

```
File,Caption
circle.ps,A Circle
rectangle.ps,A Rectangle
triangle.ps,A Triangle
```

Assuming that the files `circle.ps`, `rectangle.ps` and `triangle.ps` exist, then the following code will generate a figure for each graphics file[2]:

```
\applyCSVfile{sample3.csv}{
\begin{figure}
\centerline{\includegraphics{\insertFile}}
\caption{\insertCaption}
\end{figure}}
```

Note that in this example, you can't use `\insertbyname{File}`. (See Note 3 in Section 9.)

`\applyCSVfile*`

`\applyCSVfile*[`⟨*n*⟩`]{`⟨*filename*⟩`}{`⟨*text*⟩`}`

`\field`

In this case the CSV file has no header row, so there are no `\insert`⟨*identifier*⟩ or `\insertbyname{`⟨*label*⟩`}` commands available, instead, the command `\field{`⟨*col*⟩`}` should be used, where ⟨*col*⟩ is the column number.

Example 3 (Mail Merging using `\field`)

Suppose there is a file called `details.csv` that has the following contents:

```
Miss A. Person,1 The Road\\The Town\\AB1 2XY,15.00,4th May 2004
Mr A. N. Other,2 The Road\\The Town\\AB1 2XY,15.30,11th May 2004
```

[2]The `graphicx` package will be needed.

then the following code can be used to generate a letter for each person in the CSV file:

```
\applyCSVfile*{details.csv}{%
\begin{letter}{\field{1}\\\field{2}}
\opening{Dear \field{1}}

You are invited to an interview at \field{3}\ on the \field{4}.

\closing{Yours Sincerely}
\end{letter}}
```

---

3 Converting CSV file into a tabular environment

`\CSVtotabular` `\CSVtotabular{`⟨*filename*⟩`}{`⟨*col-align*⟩`}{`⟨*first*⟩`}{`⟨*middle*⟩`}{`⟨*last*⟩`}`

⟨*filename*⟩ is the name of the CSV file which must have a header row on line 1, ⟨*col-align*⟩ is the column alignment argument that gets passed to the `tabular` environment, ⟨*first*⟩ is the code for the first line, ⟨*middle*⟩ is the code for the middle lines and ⟨*last*⟩ is the code for the last line. This is best demonstrated with an example.

Example 4 (Aligning Data from a CSV file)

Suppose the file `sample.csv` looks like:

```
Name,Assignment 1,Assignment 2,Total
A. Smith,80,70,150
B. Jones,60,80,140
J. Doe,85,75,160
,75,75,150
```

then the following code can be used to align the data:

```
\CSVtotabular{sample.csv}{lccc}{%
\bfseries Name &
\bfseries Assignment 1&
\bfseries Assignment 2&
\bfseries Total\\}{%
\insertName &
\insertbyname{Assignment 1} &
\insertbyname{Assignment 2} &
\insertTotal\\}{%
 &
\insertbyname{Assignment 1} &
\insertbyname{Assignment 2} &
\insertTotal}
```

The result of this code is shown in Table 1[3].

`\ifnextrowlast` `\ifnextrowlast{`⟨*last-code*⟩`}{`⟨*not-last-code*⟩`}`

[3]Note that `\CSVtotabular` only puts the data in a `tabular` environment not in a table

Table 1: Example 4

Name	**Assignment 1**	**Assignment 2**	**Total**
A. Smith	80	70	150
B. Jones	60	80	140
J. Doe	85	75	160
	75	75	150

Table 2: Example 5

Name	**Assignment 1**	**Assignment 2**	**Total**
A. Smith	80	70	150
B. Jones	60	80	140
J. Doe	85	75	160
	75	75	150

The command `\ifnextrowlast` can be used to vary what happens on the last but one row. The following example illustrates this by placing `\hline\hline` after the penultimate row.

Example 5 (Adding Lines)

```
\CSVtotabular{sample.csv}{|l|ccc|}{%
\hline\bfseries Name &
\bfseries Assignment 1&
\bfseries Assignment 2&
\bfseries Total\\\hline\hline}{%
\insertName &
\insertbyname{Assignment 1} &
\insertbyname{Assignment 2} &
\insertTotal
\ifnextrowlast{\\\hline\hline}{\\}}{%
 &
\insertbyname{Assignment 1} &
\insertbyname{Assignment 2} &
\insertTotal\\\hline}
```

This result of this code is shown in Table 2.

Example 6 (Added Complexity)

In this example, `\multicolumn` is used to override the column specifier for the first column in the last row.

```
\CSVtotabular{sample2.csv}{|l|ccc|}{%
\hline\bfseries Name &
\bfseries Assignment 1 &
\bfseries Assignment 2 &
\bfseries Total\\\hline\hline
```

Table 3: Example 6

Name	**Assignment 1**	**Assignment 2**	**Total**
A. Smith	80	70	150
B. Jones	60	80	140
J. Doe	85	75	160
	75	75	150

```
}{%
\insertName &
\insertbyname{Assignment 1} &
\insertbyname{Assignment 2} &
\insertTotal
\ifnextrowlast{\\\hline\multicolumn{1}{l|}{}}{\\}
}{%
 &
\insertbyname{Assignment 1} &
\insertbyname{Assignment 2} &
\insertTotal\\\cline{2-4}
}
```

Notice that instead of placing `\multicolumn{1}{l|}{}` at the start of the final argument, it is instead placed in the first argument to `\ifnextrowlast`[4]. The result of this code is shown in Table 3.

4 Converting CSV file into longtable environment

`\CSVtolongtable` The command `\CSVtolongtable` works in the same way as `\CSVtotabular` but creates a `longtable` environment instead of a `tabular` environment.

Example 7 (Using a longtable environment)

Suppose the CSV file in the previous example, contains, say, 100 entries. This will no longer fit onto one page, so it would be better to use `\CSVtolongtable` instead. For example:

```
\CSVtolongtable{sample.csv}{|l|ccc|}{%
\caption{Student Marks}\label{tab:students}\\
\hline
\bfseries Name &
\bfseries Assignment 1 &
\bfseries Assignment 2 &
\bfseries Total\\\hline
\endfirsthead
\caption[]{Student Marks}\\
\hline
```

[4]See Note 4 in Section 9

```
\bfseries Name &
\bfseries Assignment 1 &
\bfseries Assignment 2 &
\bfseries Total\\\hline
\endhead
\hline
\multicolumn{3}{r}{\em Continued on next page}
\endfoot
\hline
\endlastfoot}{%
\insertName &
\insertbyname{Assignment 1} &
\insertbyname{Assignment 2} &
\insertTotal
\ifnextrowlast{\\\hline\hline}{\\}}{%
 & \insertbyname{Assignment 1} &
 \insertbyname{Assignment 2} &
\insertTotal\\}
```

---

5 Associated Counters

csvlinenum
csvrownumber

Within the `\CSVtotabular`, `\CSVtolongtable` and `\applyCSVfile` commands, there are two counters, `csvlinenum` and `csvrownumber`. The former, `csvlinenum`, is the current line number in the CSV file, whereas the latter, `csvrownumber`, is the current data row. Of the two counters, `csvrownumber` is likely to be the most useful.

Example 8 (Stripy Table)

The package `colortbl` defines the command `\rowcolor` which enables you to specify the row colour. Suppose you want a stripy table[5], this can be achieved as follows:

```
\CSVtotabular{sample2.csv}{lccc}{%
\rowcolor{green}\bfseries Name &
\bfseries Assignment 1 &
\bfseries Assignment 2 &
\bfseries Total\\\rowcolor{blue}
}{%
\insertName &
\insertbyname{Assignment 1} &
\insertbyname{Assignment 2} &
\insertTotal
\ifthenelse{\isodd{\value{csvrownumber}}}{%
\\\rowcolor{blue}}{\\\rowcolor{green}}
}{%
 &
\insertbyname{Assignment 1} &
```

[5]This is designed as an example of how to use the package, not incouragement to produce garish tables!

```
\insertbyname{Assignment 2} &
\insertTotal
}
```

Example 9 (More Mail Merging)

This is an example of mail merging where the letter reference is generated from the value of `csvrownumber`. The CSV file is as used in Example 1 on page 2.

```
\applyCSVfile{details.csv}{%
\begin{letter}{\insertName\\\insertAddress}
\opening{Dear \insertName}

\textbf{Ref : } interview.\thecsvrownumber

You are invited to an interview at \insertTime\ on the \insertDate.

\closing{Yours Sincerely}
\end{letter}}
```

---

6 Cross-Referencing

Labels can be generated using the standard `\label` command, but you will need some way to make each label unique. Example 10 does this by using `\thecsvrownumber`, whereas Example 11 uses `\insert`⟨*identifier*⟩.

Example 10 (Labelling within `\applyCSVfile`)

Example 2 on page 3 can be modified to label each figure:

```
\applyCSVfile{sample3.csv}{
\begin{figure}
\centerline{\includegraphics{\insertFile}}
\caption{\insertCaption}
\label{fig:pic\thecsvrownumber}
\end{figure}}
```

This example uses `\label{fig:pic\thecsvrownumber}`, so the first figure generated by this `\applyCSVfile` command will have the label `fig:pic1`, the second `fig:pic2` etc.

Example 11 (Labelling within `\applyCSVfile`)

Modifying the previous example, we now have:

```
\applyCSVfile{sample3.csv}{
\begin{figure}
\centerline{\includegraphics{\insertFile}}
\caption{\insertCaption}
\label{fig:\insertFile}
\end{figure}}
```

The labels for each figure are now: `fig:circle.ps`, `fig:rectangle.ps` and `fig:triangle.ps`, respectively.

Example 12 (Labelling within `CSVtotabular`)

This example is slightly more complicated. The CSV file, `data.csv` looks like:

```
Incubation Temperature,Incubation Time,Time to Growth
40,120,40
40,90,60
35,180,20
```

The following code generates a table using the data with an additional column that generates the experiment number. (See note 9.)

```
\begin{table}
\caption{Time to Growth Experiments}
\label{tab:exp}
\vspace{10pt}
\centering
\CSVtotabular{data.csv}{cccc}{%
 % Header Row
\bfseries Experiment &
\bfseries \begin{tabular}{c}Incubation\\Temperature\end{tabular} &
\bfseries \begin{tabular}{c}Incubation\\Time\end{tabular} &
\bfseries \begin{tabular}{c}Time\\to\\Growth\end{tabular}\\}{%
 % Middle Rows
\label{exp:\insertbyname{Incubation Temperature}:\insertbyname{Incubation Time}}
\thecsvrownumber &
\insertbyname{Incubation Temperature} &
\insertbyname{Incubation Time} &
\insertbyname{Time to Growth} \\}{%
 % Final Row
\label{exp:\insertbyname{Incubation Temperature}:\insertbyname{Incubation Time}}
\thecsvrownumber &
\insertbyname{Incubation Temperature} &
\insertbyname{Incubation Time} &
\insertbyname{Time to Growth}}
\par
\end{table}

It can be seen from Table~\ref{tab:exp}, that
Experiment~\ref{exp:35:180} had the shortest time to growth.
```

In this example, each experiment has the corresponding label `exp:`⟨*Incubation Temperature*⟩`:`⟨*Incubation Time*⟩ so the first experiment has label `exp:40:120`, the second experiment has the label `exp:40:90` and the third experiment has the label `exp:35:180`.

Table 4 shows the resulting table for this example.

Table 4: Time to Growth Experiments

Experiment	**Incubation Temperature**	**Incubation Time**	**Time to Growth**
1	40	120	40
2	40	90	60
3	35	180	20

The following example is more refined in that it takes advantage of the fact that the time to growth data consists of integers only, so the experiment with the maximum growth can be determined by LaTeX.

Example 13 (Labelling within `CSVtotabular`)

```
\newcounter{maxgrowth}
\newcounter{incT} % incubation temperature
\newcounter{inct} % incubation time

\begin{table}
\caption{Time to Growth Experiments}
\label{tab:exp}
\vspace{10pt}
\centering
\CSVtotabular{data.csv}{cccc}{%
 % Header row
\bfseries Experiment &
\bfseries \begin{tabular}{c}Incubation\\Temperature\end{tabular} &
\bfseries \begin{tabular}{c}Incubation\\Time\end{tabular} &
\bfseries \begin{tabular}{c}Time\\to\\Growth\end{tabular}\\}{%
 % Middle rows
\label{exp:\insertbyname{Incubation Temperature}:\insertbyname{Incubation Time}}
\thecsvrownumber &
\insertbyname{Incubation Temperature} &
\insertbyname{Incubation Time} &
\insertbyname{Time to Growth}%
\ifthenelse{\value{maxgrowth}<\insertbyname{Time to Growth}}{%
\setcounter{maxgrowth}{\insertbyname{Time to Growth}}%
\setcounter{incT}{\insertbyname{Incubation Temperature}}%
\setcounter{inct}{\insertbyname{Incubation Time}}}{}%
\\}{%
 % Last row
\label{exp:\insertbyname{Incubation Temperature}:\insertbyname{Incubation Time}}
\thecsvrownumber &
\insertbyname{Incubation Temperature} &
\insertbyname{Incubation Time} &
\insertbyname{Time to Growth}%
\ifthenelse{\value{maxgrowth}<\insertbyname{Time to Growth}}{%
\setcounter{maxgrowth}{\insertbyname{Time to Growth}}%
\setcounter{incT}{\insertbyname{Incubation Temperature}}%
\setcounter{inct}{\insertbyname{Incubation Time}}}{}%
}
```

```
\par
\end{table}

As can be seen from Table~\ref{tab:exp},
Experiment~\ref{exp:\theincT:\theinct}
had the maximum time to growth, with
incubation time \theinct,
incubation temperature \theincT\ and
time to growth, \themaxgrowth.
```

---

7 Saving Entries

Entries can be saved using the command:

`\csvSaveEntry` `\csvSaveEntry[`⟨*counter*⟩`]{`⟨*identifier*⟩`}`

where ⟨*counter*⟩ is a LaTeX counter, by default `csvrownumber`, and ⟨*identifier*⟩ is the header entry. The entry can then be used with the command:

`\csvGetEntry` `\csvGetEntry{`⟨*counter*⟩`}{`⟨*identifier*⟩`}`

The following example illustrates the use of these commands.

Example 14 (Saving Entries)

This example illustrates how you can use one CSV file to access data in other CSV files. This example has several CSV files:

File `index.csv`:

```
File,Temperature,NaCl,pH
exp25a.csv,25,4.7,0.5
exp25b.csv,25,4.8,1.5
exp30a.csv,30,5.12,4.5
```

File `exp25a.csv`:

```
Time,Logcount
0,3.75
23,3.9
45,4.0
```

File `exp25b.csv`:

```
Time,Logcount
0,3.6
60,3.8
120,4.0
```

File `exp30a.csv`:

```
Time,Logcount
0,3.73
23,3.67
60,4.9
```

It is not possible to nest `\CSVtotabular`, `\CSVtolongtable` and `\applyCSVfile`, so if you need to go through `index.csv` and use each file named in there, you can first go through `index.csv` storing the information using `\csvSaveEntry` as follows:

```
\newcounter{maxexperiments}
\applyCSVfile{sample5.csv}{%
\stepcounter{maxexperiments}
\csvSaveEntry{File}
\csvSaveEntry{Temperature}
\csvSaveEntry{NaCl}
\csvSaveEntry{pH}
}
```

The counter `maxexperiments` simply counts the number of entries in `index.csv`. The entries can now be used to generate a table for each file listed in `index.csv` (the `\whiledo` command is defined in the `ifthen` package):

```
\newcounter{experiment}
\whiledo{\value{experiment}<\value{maxexperiments}}{%
\stepcounter{experiment}
\begin{table}
\caption{Temperature = \protect\csvGetEntry{experiment}{Temperature},
NaCl = \protect\csvGetEntry{experiment}{NaCl},
pH = \protect\csvGetEntry{experiment}{pH}}
\vspace{10pt}
\centering
\CSVtotabular{\csvGetEntry{experiment}{File}}{ll}{%
Time & Log Count\\}{%
\insertTime & \insertLogcount\\}{%
\insertTime & \insertLogcount}

\end{table}
}
```

Note that `\csvGetEntry` needs to be `\protect`ed within the `\caption` command.

This example can be modified if, say, you only want the tables where the temperature is 25:

```
\setcounter{experiment}{0}
\whiledo{\value{experiment}<\value{maxexperiments}}{%
\stepcounter{experiment}
\ifthenelse{\equal{\csvGetEntry{experiment}{Temperature}}{25}}{%
\begin{table}
\caption{Temperature = \protect\csvGetEntry{experiment}{Temperature},
NaCl = \protect\csvGetEntry{experiment}{NaCl},
pH = \protect\csvGetEntry{experiment}{pH}}
\vspace{10pt}
\centering
\CSVtotabular{\csvGetEntry{experiment}{File}}{ll}{%
Time & Log Count\\}{%
\insertTime & \insertLogcount\\}{%
\insertTime & \insertLogcount}\par
\end{table}}{}
}
```

---

8 The csvtools.pl Perl Script

Suppose you have several large CSV files, and you have included the information into your document using `\applyCSVfile`, `\CSVtolongtable` or `\CSVtotabular`, which has made life so much easier for you, but you are now required by a journal to submit your source code in a single `.tex` file. They don't want all your CSV files, so what do you do? If you have Perl installed on your system you can use the `csvtools.pl` Perl script. This has the following syntax:

`csvtools.pl` ⟨*in-file*⟩ ⟨*out-file*⟩

where ⟨*in-file*⟩ is the name of your file that contains the `\applyCSVfile`, `\CSVtotabular` etc commands, and ⟨*out-file*⟩ is a new file which will be created by `csvtools.pl`. This new file will be the same as ⟨*in-file*⟩ except that all occurances of `\applyCSVfile`, `\CSVtolongtable` and `\CSVtotabular` will be replaced by the relevant data extracted from the named CSV files.

Example 15 (csvtools.pl — Aligning Data)

Suppose the file `mydoc.tex` contains the code given in Example 4, with the associated CSV file `sample.csv` also given in that example. Then if you do:

```
csvtools.pl mydoc.tex mydocnew.tex
```

the file `mydocnew.tex` will be created which will be identical to `mydoc.tex` except the lines containing the code `\CSVtotabular{sample.csv}{lccc}{...}{...}{...}` will be replaced with the lines:

```
% \CSVtotabular{sample.csv}... converted using csvtools.pl
%>> START INSERT
\begin{tabular}{lccc}
\bfseries Name &
\bfseries Assignment 1 &
\bfseries Assignment 2 &
\bfseries Total\\
A. Smith&80&70&150\\
B. Jones&60&80&140\\
J. Doe&85&75&160\\
 &75&75&150
\end{tabular}%<< END INSERT
```

Similarly, `csvtools.pl` will substitute all occurrances of `\CSVtolongtable` and `\applyCSVfile`.

8.1 Notes

1. The `csvtools.pl` file is bundled up with the code and documentation in the file `csvtools.dtx`. It will be extracted along with the code when you LaTeX the installation script `csvtools.ins`. If you are using UNIX or Linux etc, you will need to set the permissions so that the file can be executed:

   ```
   chmod u+x csvtools.pl
   ```

If perl is located in a directory other than `/usr/bin/` you will need to edit the first line of `csvtools.pl` as appropriate. You can find the location using the command:

```
which perl
```

2. If you can't directly execute a Perl script, you can do:

   `perl csvtools.pl` ⟨*in-file*⟩ ⟨*out-file*⟩

3. You must first LaTeX your document before using `csvtools.pl` as it checks the log file for any counters that have been defined.

4. `csvtools.pl` only knows about a very limited set of LaTeX commands. It should be able to understand:

   ```
   \CSVtotabular{\csvGetEntry{experiment}{File}}{ll}{...
   ```

   (see Example 14), but it won't be able to understand, say,

   ```
   \newcommand{\filename}{\csvGetEntry{experiment}{File}}
   \CSVtotabular{\filename}{ll}{...
   ```

   It can pick up on `\addtocounter`, `\stepcounter`, `\refstepcounter` and `\setcounter` but only if they are used explicitly in the named `.tex` file. (It ignores any files that have been included using `\input`, `\include` etc.)

5. This Perl script has only been tested under Linux, but it ought to work under other systems.

9 Bugs/Drawbacks/"Features"

1. The package doesn't check to see whether `\insert`⟨*identifier*⟩ exists, otherwise you would not be able to use multiple CSV files with the same headers, as in Example 14. Therefore it is recommended that you check to make sure that the command does not already exist. For example, the TeX commands `\insert` and `\insertpenalties` already exist, so a blank header or a header named `penalties` would cause problems. (These two will now cause an error as from version 1.1, but it's something bear in mind.)

2. Note also that `\insertbyname` doesn't check if you've given a valid label, so if no text appears, check you've spelt it correctly, checking punctuation, spaces and case.

3. Note that in Example 2, replacing line 3 with:

   ```
   \centerline{\includegraphics{\insertbyname{File}}}
   ```

   will cause an error, as `\insertbyname{File}` doesn't get fully expanded by the time it gets passed to `\includegraphics`, and will prevent `\includegraphics` from finding the file. It is possible to get around this using TeX's `\edef` command:

   ```
   \edef\psfilename{\insertbyname{File}}
   \centerline{\includegraphics{\psfilename}}
   ```

4. You can't have commands like `\hline`, `\cline` and `\multicolumn` in the first column of the ⟨*middle*⟩ or ⟨*last*⟩ code of `\CSVtotabular` or `\CSVtolongtable`. If you do, it will generate a `misplaced \noalign` error, instead you need to put it at the end of the ⟨*first*⟩ or ⟨*middle*⟩ code. (See Example 6.)

5. You can't have nested `\applyCSVfile`, `\CSVtolongtable` and `\CSVtotabular` commands. (See Example 14)

6. If the CSV file has a header row, it must be on the first line.

7. It is possible for TEX to run out of memory if you use `\csvSaveEntry` on a large file.

8. Commas within an entry in a CSV file may cause problems. You will need to enclose the entry in braces, e.g. `{Joe Smith, Jr}`.

9. In version 1.0, there was an inconsistency with `csvrownumber` within `\applyCSVfile` and `\CSVtotabular`. In the former it excluded the header row, whereas the latter included it. This has been changed in version 1.1 so that within `\applyCSVfile`, `\CSVtotabular` and `\CSVtolongtable`, `csvrownumber` refers to the data row (excluding header row.) I hope this doesn't cause problems, but it makes more sense that they should be consistent. So if you have no blank lines in your CSV file, `csvrownumber` should always be 1 more than `csvlinenumber`.

10 Contact Details

Dr Nicola Talbot
School of Computing Sciences
University of East Anglia
Norwich. NR4 7TJ. England.

http://theoval.cmp.uea.ac.uk/~nlct/