Data publishing

Siep Kroonenberg
Economics Department, Groningen University

Abstract
This paper demonstrates how commercial software and \LaTeX can work together in data publishing.

Keywords: database, spreadsheet, table

Introduction
One of (\LaTeX)'s strong suits is automation. Every \LaTeX-user knows about \LaTeX's features for automatic cross-references and automatic tables of contents. But some people may not realize how convenient \LaTeX can be for publishing data from commercial programs. Therefore I present here a couple of real-life examples.

CCSO, the economic research group with which I am associated specializes in the building of econometric models, for forecasting, policy analysis and theoretical exercises. The data for a model typically consist of several hundreds of time series. Now and then we publish these databases, both the data themselves and associated descriptive information.

1 The descriptive database
The descriptive database contains for each time series a description in economic terms and a definition which contains either its source or a definition in terms of other series. A series taken directly from a published source (in this case, the 'CBS Kwartaalrekeningen') might have the following database entry:

```
The descriptive database

CNAME COGRABS
TEXNAME C_o
DESCRIPTION Consumption of general government at current prices
DEFINITION 1000 * CBS KR 1.3.1 row 5

For a constructed series, we might have

CNAME CP
TEXNAME \dot{C}_p
DESCRIPTION Growth rate of consumption of households at constant prices
DEFINITION 100 * (CPKLABS / CPKLABS(-1) - 1)

CNAME indicates the name used in the computer models and the spreadsheets; TEXNAME the \TeX code used in papers; DESCRIPTION is the definition in economic terms, and DEFINITION specifies either the source or how it is computed from other series.

We wish to emulate and improve upon a 'native' dBase report, in which each field has its own column, and long fields are wrapped within their column.

There are two problems associated with using the tabular environment for this purpose: first, a tabular environment justifies text even within a narrow column, which can be very ugly.

The first problem is solved by using Theo Jurriën's `\supertabular environment, which is defined in the supertab style file or package. This environment also allows the specification of a table header which is repeated on every page.

We solve the second problem in a less elegant way: we wrap each long field in a parbox of appropriate width:

\newcommand{px}{\parbox[t]{1in}{\raggedright #1}}
\newcommand{py}{\parbox[t]{1.3in}{\raggedright #1}}

The following dBase commands create a file db94tex.txt containing a listing in \LaTeX format:
```
set alternate to db94tex
set talk off
set console off
set alternate on
go top
dowhile .not. eof()
dobrec
skip
enddo
set alternate off
close alternate
set console on
set talk on
```

This is the procedure dbrec that was called above:
```
procedure dbrec
?'$'+trim(texname)+'$ &
?'tt '+trim(cname)+' &
?\px{'+trim(descrp)+'} &
?\py{'+trim(definition)+'} \\
?
return
```

A fragment of output from these commands:
```
\begin{verbatim}
$C_o$ & COGRABS & Consumption of general government at current prices & 1000 * CBS KR 1.3.1 row 5 \\
$\dot{C}_p$ & CP & Growth rate of consumption of households at constant prices & 100 * (CPKLABS / CPKLABS(-1) - 1) \\
\end{verbatim}
```

Preprint MAPS#12 (94.1); May 1994 Dutch \TeX Users Group (NTG), P.O. Box 394, 1740 AJ Schagen, The Netherlands
The following \LaTeX code will process the file db94tex.txt (remember to include supertab as a documentstyle option):

\begin{verbatim}
{\renewcommand{\arraystretch}{2}
% for extra vertical space between records;
% a pair of braces limits this redefinition
% to the table
\tablehead{& \bf Series & \bf Description & \%
\bf Source/definition\}
\begin{supertabular}{@{}p{0.3in}p{0.5in}p{1in}p{1.3in}@{}}
\input db94tex.txt
\end{supertabular}
}
\end{verbatim}

And we get:

<table>
<thead>
<tr>
<th>Series</th>
<th>Description</th>
<th>Source/definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_0$</td>
<td>COGRABS</td>
<td>Consumption of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>general government row 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>at current prices</td>
</tr>
<tr>
<td>$C_p$</td>
<td>CP</td>
<td>Growth rate of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>consumption of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>households at</td>
</tr>
<tr>
<td></td>
<td></td>
<td>constant prices</td>
</tr>
</tbody>
</table>

The alternative: tabbing

The example can be easily adapted for a tabbing environment; in fact, that is what we actually did, ignorant of the blessings of supertabular. We even managed to press the page header into service as a repeating table header. But now we know better.

2 The timeseries spreadsheet

Our timeseries are stored in spreadsheets, such as

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>BPRGRABS</td>
<td>BPRKLABS</td>
<td>COGRABS</td>
<td>COKLABS</td>
</tr>
<tr>
<td>2</td>
<td>$&amp;$</td>
<td>$SB!P&amp;!R&amp;$</td>
<td>$bpr&amp;$</td>
<td>$C_o&amp;$</td>
<td>$c_o&amp;$</td>
</tr>
<tr>
<td>3</td>
<td>1980Q1</td>
<td>80500</td>
<td>83831</td>
<td>14100</td>
<td>15041</td>
</tr>
<tr>
<td>4</td>
<td>1980Q2</td>
<td>86400</td>
<td>87945</td>
<td>17700</td>
<td>18233</td>
</tr>
<tr>
<td>5</td>
<td>1980Q3</td>
<td>82000</td>
<td>82501</td>
<td>13700</td>
<td>13842</td>
</tr>
<tr>
<td>6</td>
<td>1980Q4</td>
<td>87800</td>
<td>87800</td>
<td>14700</td>
<td>14700</td>
</tr>
<tr>
<td>7</td>
<td>1981Q1</td>
<td>84800</td>
<td>84250</td>
<td>15000</td>
<td>15643</td>
</tr>
<tr>
<td>8</td>
<td>1981Q2</td>
<td>89700</td>
<td>87945</td>
<td>18500</td>
<td>18780</td>
</tr>
</tbody>
</table>

The first column contains dates; 1981Q1 indicates the first quarter of 1981. The first row contains the names used in the computer model; the second their \LaTeX equivalents (in practice, we add the \LaTeX names only when we are going to print the database; for this, we use the descriptive database discussed above).

Generating \LaTeX code from such a spreadsheet requires fewer tricks than in the dBase case since we don’t have to wrap lines within cells. However, it can’t be automated as effectively because, in spite of their macro capabilities, spreadsheets are not really geared towards batch processing. Maybe some day I’ll write a Pascal program that does it right.

Meanwhile, we’ll let Lotus’ print facility generate the output. With some suitable macros (not detailed here), it isn’t too much of a job to massage the spreadsheet into shape: we add ‘s alignment symbols with a macro, and a column of ‘\’ new-line symbols by hand. Printing a block to a file timesers.prn we get something like

\begin{verbatim}
& $SB!P\&!R\&$ ... & $\&$ \dot{C}_p \\
1980Q1 & ... & -4.4301 \\
1980Q2 & ... & 0.7075 \\
1980Q3 & ... & -0.5809 \\
1980Q4 & ... & 3.4624 \\
1981Q1 & ... & -5.3955 \\
1981Q2 & ... & 0.1937 \\
\end{verbatim}

The lines

\begin{verbatim}
\begin{tabular}{@{}lrrrrr@{}}
\input timesers.prn
\end{tabular}
\end{verbatim}

produce the table below:

<table>
<thead>
<tr>
<th>BPR</th>
<th>Bp</th>
<th>$C_0$</th>
<th>$c_o$</th>
<th>$\dot{C}_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1980Q1</td>
<td>80500</td>
<td>83831</td>
<td>14100</td>
<td>15041</td>
</tr>
<tr>
<td>1980Q2</td>
<td>86400</td>
<td>87945</td>
<td>17700</td>
<td>18233</td>
</tr>
<tr>
<td>1980Q3</td>
<td>82000</td>
<td>82501</td>
<td>13700</td>
<td>13842</td>
</tr>
<tr>
<td>1980Q4</td>
<td>87800</td>
<td>87800</td>
<td>14700</td>
<td>14700</td>
</tr>
<tr>
<td>1981Q1</td>
<td>84800</td>
<td>84250</td>
<td>15000</td>
<td>15643</td>
</tr>
<tr>
<td>1981Q2</td>
<td>89700</td>
<td>87945</td>
<td>18500</td>
<td>18780</td>
</tr>
</tbody>
</table>

3 Caveats

Watch out for characters in the database or spreadsheet which are also markup codes for \LaTeX, such as percent- or ampersand characters. In spreadsheets, texts should fit within their columns, or they will be truncated without warning. Also, if the block to be printed exceeds the declared linelength of the printfile, the table will be split horizontally.