pst-bar
Bar charts for pstricks

v.0.7

Alan Ristow
ristow@ee.gatech.edu

September 15, 2004

Abstract

With pst-bar, one may use pstricks to produce bar charts directly from a data file. This package is still in the beta stage — the usual caveats pertaining to beta software apply. Additional features and improved (in both content and layout) documentation will be provided as the code stabilizes.

1 Introduction

pst-bar uses the power and flexibility of pstricks [1] to draw bar charts from data stored in a comma-delimited file. There are currently two existing solutions for producing bar charts using pstricks. One is the "brute force" method, which takes advantage of the fact that a bar in a bar chart is nothing more than a colored box. Using this method, a series of \psframes are drawn with suitable heights, widths, and colors to produce the desired chart [2]. While this method works admirably and provides maximum flexibility, it is tedious and labor-intensive. The other option is the bardiag package, which essentially automates the brute force approach [3]. While this approach is effective, bardiag requires a number of external packages to function, uses LaTeX to perform its mathematical operations, and is not compatible with plain TeX.

These issues are all avoided by pst-bar. It calls only on packages that are part of the pstricks family, provides extensive customization features, relies on Postscript for nearly all mathematical operations, and should be compatible with plain TeX (untested). pst-bar provides commands for reading data from a comma-delimited file to produce and label a bar chart. Multiple series of data may be plotted in either clustered or stacked form, and bars may be plotted either horizontally or vertically. An additional option allows block charts that display the differences between subsequent rows of data in the data file (see Section 3.3).

At this stage, pst-bar should be regarded as beta software. Revisions to the code between this release and the first stable release are not guaranteed to be backward compatible.

2 Drawing a Bar Chart

Drawing a bar chart with pst-bar is a three-step process: (i) read the data file, (ii) draw the bars, and (iii) draw the axes. The first two step are accomplished directly with pst-bar; the third requires the pst-plot package, a standard part of pstricks.

The following sections address the data file format, how to load data from the file, and how to use the data to produce a bar chart.

2.1 Data File Format

The data file must be comma-delimited with each row containing the same number of entries. A header row is permitted, but no header entry may contain commas or `\par` commands (including `\\`). One of the data files used for the examples in this manual looks like

```
Set 1, Set 2, Set 3
1, 2, 3
1, 2, 3
```

In this case, `Set 1`, `Set 2`, and `Set 3` are the headers. The subsequent rows, hereafter referred to as *data rows*, contain the data associated with these headers. The file is arranged in a pseudo-columnar fashion, so that the data associated with `Set 1` are `1` and `1`, those with `Set 2` are `2` and `2`, and so on. After these data are plotted, the bars will be labeled with the text from the header row.

Warning! There are two very important conditions on the data within the file:

- All data outside of the header must be numeric. If there is no header row, all data in the file must be numeric.
- Each row must contain the same number of entries.

Failure to meet these conditions may result in Postscript errors or (less frequently) incorrectly labeled or drawn bar charts.

How these data are plotted depends on the type of plot desired. See Sections 3.1–3.3 and the examples in Section 4 for more on this.

2.2 Reading the Data File

Use the command `\readpsbardata` to read from a data file:

```
\readpsbardata[<options>]{<data>}{<filename>}
```

For the data, simply specify a macro name not currently in use.

The available options are:

Parameter	Definition
`header`	`true` if first line of data file is header row `false` otherwise Default: `true`

If `header` is set to `false`, `pst-bar` will assume that no header is present and treat the first line of the file as data.

2.3 Drawing the Bars

The command `\psbarchart` draws the bars for the bar chart.

```
\psbarchart[<options>]{<data>}
```

For `<data>`, use the macro name supplied to `\readpsbardata`. Bars are grouped in columns that are one `\psxunit` wide, and the number of bars appearing in a single column is equal to the number of data rows in the data file. It labels the bottom of each column with the headers from the data file if `header` was set to to `true` during `\readpsbardata`. It only creates the bars and the column labels — axes and frames must be provided separately.

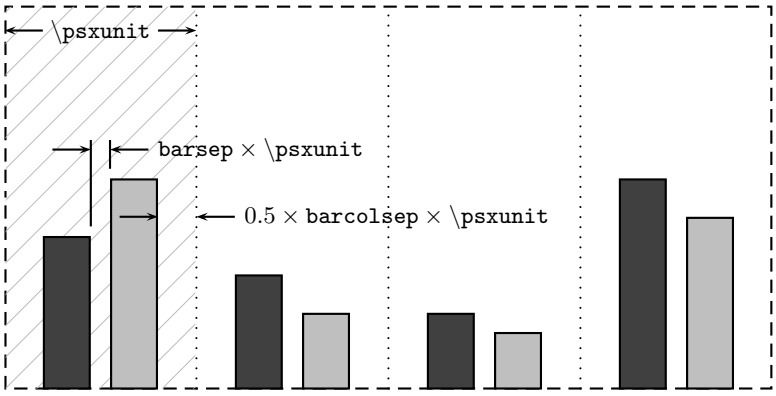


Figure 1: Schematic diagram for the layout of the bar chart. The hatched area marks the region in which data from the first column of the data file will be plotted. Bar width is equal to $[(1 - \texttt{barsep} - 0.5 \times \texttt{barcolsep}) \times \texttt{\psxunit}]/N$, where N is the number of bars in the column.

The available options are:

Parameter	Definition
`chartstyle`	`cluster` produces a cluster of bars `stack` stacks the cluster into a single column `block` prints a "floating" bar with a nonzero minimum Default: `cluster`
`barstyle`	Name(s) of bar style(s) to use for each bar Default: `{black,darkgray,gray,lightgray,white,red,green,blue}`
`barcolsep`	Factor determining whitespace between columns Default: `0.4`
`barsep`	Factor determining whitespace between bars (`cluster` and `block` charts) Default: `0.0`
`barlabelrot`	Angle of rotation of the column labels Default: `0`
`orientation`	`vertical` for vertically drawn bars `horizontal` for horizontally drawn bars Default: `vertical`

The number of `barstyle`s specified must equal or exceed the number of bar segments in each column; otherwise, a Postscript error will result. For the `cluster` and `stack` `chartstyle`s, this is equal to the number of data rows in the data file; for the `block` `chartstyle`, it is equal to half the number of data rows. Furthermore, when specifying the list of `barstyle`s, the styles must be listed between curly braces, e.g.,

```
\psbarchart[barstyles={black,gray,white},...]{...}
```

The `barstyle`s available by default are `red`, `green`, `blue`, `black`, `white`, `gray`, `lightgray`, and `darkgray`. Each of these plots a bar of the named color with square corners and a black outline. New `barstyle`s may be defined using the `\newpsbarstyle` command described in Section 2.4.

Figure 1 shows how the `\psbarchart` lays out the bars for the default `chartstyle`, `cluster`. For the `stack` `chartstyle`, the `barsep` option is ignored.

2.4 Customizing the Chart

To add a custom bar style, use the command `\newpsbarstyle`,

```
\newpsbarstyle{<name>}{<definition>}
```

The name is a text string not currently used by any other `barstyle`, and the definition may consist of any `pstricks` key or group of keys applicable to `\psframe`. For example, the `red barstyle` is defined as

```
\newpsbarstyle{red}{fillcolor=red,fillstyle=solid,framearc=0}
```

To customize the type used to set the column labels, redefine `\psbarlabel`. For example, to have the label set in small italics, use

```
\renewcommand*{\psbarlabel}{\small\itshape}
```

By default, `\psbarlabel` sets the column label in the current text font.

To adjust the spacing between the bars and the column labels, redefine `\psbarlabelsep`. Note that it is defined as a command, not a length, and should be redefined using `\renewcommand*`. By default it is `0pt`.

Finally, the data from the file may be scaled and manipulated using the command `\psbarscale`,

```
\psbarscale(<scale>){<Postscript code>}
```

inspired by Herbert Voß' `\pstScale` from `pstricks-add` [4]. The data are scaled by the value in parentheses and may be further manipulated with Postscript code. For example, to plot the logarithm of the input data one would use `\psbarscale(1){log}`. The Postscript code is applied to the data prior to scaling.

3 The Chart Styles

As described in Section 2.3, there are three different `chartstyle` options. The default, `cluster` is the common bar chart with bars clustered into groups of related data. The second option, `stack`, draws a series of bars stacked atop one another instead of sitting side-by-side in a cluster. The third, `block`, draws clustered bars between two data points, thus displaying a range of values instead of a single value.

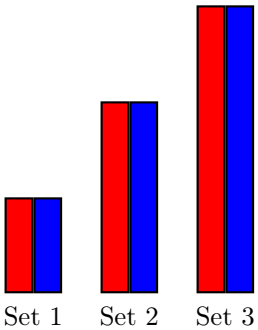
The following sections describe each `chartstyle` in detail, including how it uses the input data. Examples of use are provided in each case. Full examples, including axes, appear in Section 4.

3.1 `cluster`

For a `cluster` chart, each comma-delimited variable within a given row represents a bar in a different column. Each row represents a new set of bars. Thus the file

```
Set 1, Set 2, Set 3
1, 2, 3
1, 2, 3
```

produces the bars



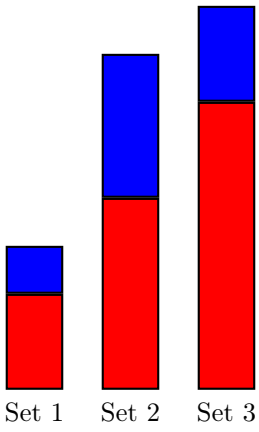

```
\psset{unit=0.5in}%
\begin{pspicture}(0,-0.5)(3,3)%
  \readpsbardata{\data}{example1.csv}%
  \psbarchart[barstyle={red,blue}]{\data}%
\end{pspicture}
```

3.2 stack

For a `stack` chart, each column of the chart has only one bar. This bar consists of as many segments as the data file has data rows, with the data from each row stacked onto the previous row. Thus, the file

```
Set 1, Set 2, Set 3
1, 2, 3
0.5, 1.5, 1
```

produces the bars




```
\psset{unit=0.5in}%
\begin{pspicture}(0,-0.5)(3,4)%
  \readpsbardata{\data}{example2.csv}%
  \psbarchart[barstyle={red,blue},%
    chartstyle=stack]{\data}%
\end{pspicture}
```

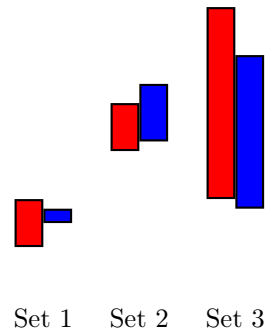
Notice that these bars are wider than those produced by `[chartstyle=cluster]` because the bars are stacked vertically instead of being clustered along the bottom axis. A bar width similar to that of the `cluster` chart in the previous section could be obtained by setting `xunit=0.25in`.

3.3 block

For a `block` chart, each bar represents a range of values. As such, each bar requires two data lines from the data file, one denoting the upper limit and the other the lower limit. If there are multiple pairs of data lines, they are plotted in a clustered fashion. If there are an odd number of data lines, the last line of the data file is ignored. Thus, the file

```
Set 1, Set 2, Set 3
1, 2, 3
0.5, 1.5, 1
0.75, 1.6, 0.9
0.9, 2.2, 2.5
```

produces the bars

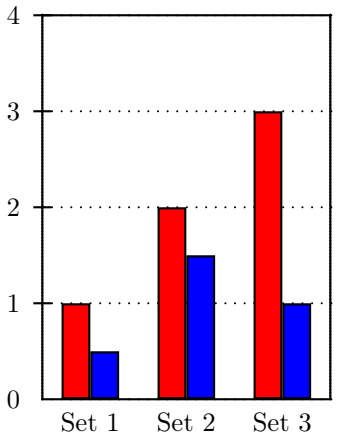



```
\psset{unit=0.5in}%
\begin{pspicture}(0,-0.5)(3,3.5)%
  \readpsbardata{\data}{example3.csv}%
  \psbarchart[barstyle={red,blue},%
    chartstyle=block]{\data}%
\end{pspicture}
```

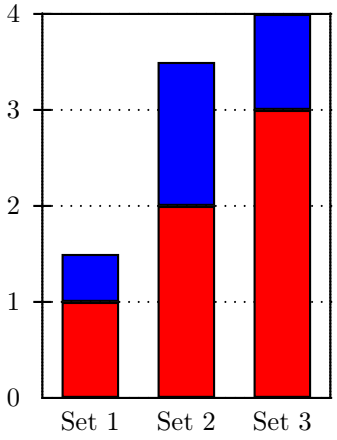
4 Examples

Basic examples of each type including axes and gridlines:

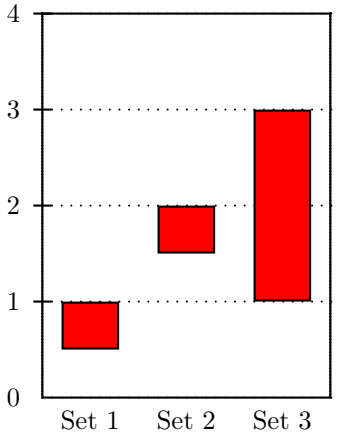
```
\begin{filecontents*}{example2.csv}
  Set 1, Set 2, Set 3
  1, 2, 3
  0.5, 1.5, 1
\end{filecontents*}
```




```
\psset{unit=0.5in}%
\begin{pspicture}(0,-0.5)(3,4.5)%
  \psgrid[xunit=1.5in,gridlabels=0,%
    subgriddiv=0,griddots=30](0,0)(1,4)%
  \psaxes[axesstyle=frame,Ox=0,Dx=1,labels=y,%
    ticks=y](0,0)(3,4)%
  \readpsbardata{\data}{example2.csv}%
  \psbarchart[barstyle={red,blue}]{\data}%
\end{pspicture}
```

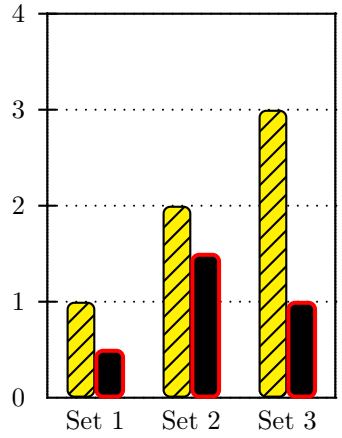



```
\psset{unit=0.5in}%
\begin{pspicture}(0,-0.5)(3,4.5)%
  \psgrid[xunit=1.5in,gridlabels=0,%
    subgriddiv=0,griddots=30](0,0)(1,4)%
  \psaxes[axesstyle=frame,Ox=0,Dx=1,labels=y,%
    ticks=y](0,0)(3,4)%
  \readpsbardata{\data}{example2.csv}%
  \psbarchart[barstyle={red,blue},%
    chartstyle=stack]{\data}%
\end{pspicture}
```



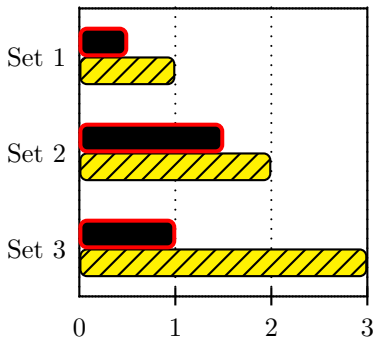

```
\psset{unit=0.5in}%
\begin{pspicture}(0,-0.5)(3,4.5)%
  \psgrid[xunit=1.5in,gridlabels=0,%
    subgriddiv=0,griddots=30](0,0)(1,4)%
  \psaxes[axesstyle=frame,Ox=0,Dx=1,labels=y,%
    ticks=y](0,0)(3,4)%
  \readpsbardata{\data}{example2.csv}%
  \psbarchart[barstyle={red,blue},%
    chartstyle=block]{\data}%
\end{pspicture}
```

Using `\newpsbarstyle`:




```
\psset{unit=0.5in}%
\newpsbarstyle{yellowhatch}{framearc=0.5,%
  fillstyle=hlines*,rot=45,fillcolor=yellow}%
\newpsbarstyle{redoutline}{framearc=0.5,%
  fillcolor=black,linecolor=red,%
  linewidth=1.5pt}%
\begin{pspicture}(0,-0.5)(3,4.5)%
  \psgrid[xunit=1.5in,gridlabels=0,%
    subgriddiv=0,griddots=30](0,0)(1,4)%
  \psaxes[axesstyle=frame,Ox=0,Dx=1,labels=y,%
    ticks=y](0,0)(3,4)%
  \readpsbardata{\data}{example2.csv}%
  \psbarchart[barstyle={yellowhatch,%
    redoutline}]{\data}%
\end{pspicture}
```

Using `[orientation=horizontal]`:




```
\psset{unit=0.5in}%
\newpsbarstyle{yellowhatch}{framearc=0.5,%
  fillstyle=hlines*,rot=45,fillcolor=yellow}%
\newpsbarstyle{redoutline}{framearc=0.5,%
  fillcolor=black,linecolor=red,%
  linewidth=1.5pt}%
\begin{pspicture}(0,-0.5)(3,4.5)%
  \psgrid[xunit=1.5in,gridlabels=0,%
    subgriddiv=0,griddots=30](0,0)(1,4)%
  \psaxes[axesstyle=frame,Ox=0,Dx=1,labels=y,%
    ticks=y](0,0)(3,4)%
  \readpsbardata{\data}{example2.csv}%
  \psbarchart[barstyle={yellowhatch,%
    redoutline}]{\data}%
\end{pspicture}
```

Sophisticated effects are possible using multiple calls to `\psbarchart` in a single chart. For example, a particular important bar may be highlighted in green by splitting the data into two files, one containing a zero for the data value to be highlighted, and the other containing zeros for all of the data *except* the value to be highlighted.

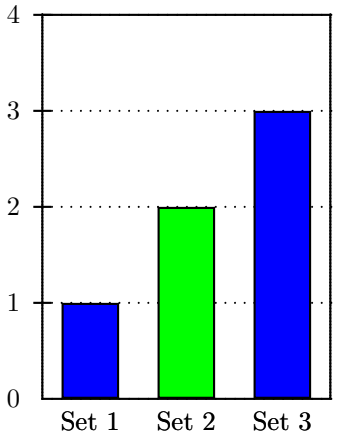
File 1 (`example4.csv`):

```
Set 1, Set 2, Set 3
1, 0, 3
```

File 2 (`example5.csv`):

```
0, 2, 0
```

Notice that the second file has no header row in order to prevent the bar labels from being printed twice.




```
\psset{unit=0.5in}%
\begin{pspicture}(0,-0.5)(3,4.5)%
  \psgrid[xunit=1.5in,gridlabels=0,%
    subgriddiv=0,griddots=30](0,0)(1,4)%
  \psaxes[axesstyle=frame,Ox=0,Dx=1,labels=y,%
    ticks=y](0,0)(3,4)%
  \readpsbardata{\data}{example4.csv}%
  \psbarchart[barstyle={blue}]{\data}%
  \readpsbardata[header=false]{\data}%
    {example5.csv}%
  \psbarchart[barstyle={green}]{\data}%
\end{pspicture}
```

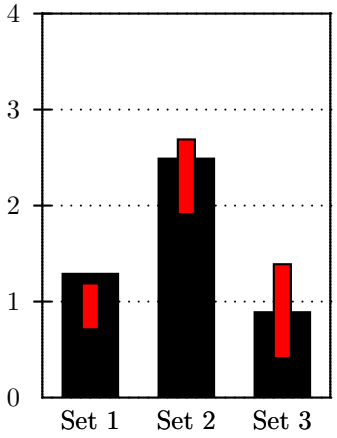
Different chart styles may also be combined in a single bar chart.

File 1 (`example6.csv`):

```
Set 1, Set 2, Set 3
1.3, 2.5, 0.9
```

File 2 (`example7.csv`):

```
0.7, 1.9, 0.4
1.2, 2.7, 1.4
```




```
\psset{unit=0.5in}%
\begin{pspicture}(0,-0.5)(3,4.5)%
  \psgrid[xunit=1.5in,gridlabels=0,%
    subgriddiv=0,griddots=30](0,0)(1,4)%
  \psaxes[axesstyle=frame,Ox=0,Dx=1,labels=y,%
    ticks=y](0,0)(3,4)%
  \readpsbardata{\data}{example6.csv}%
  \psbarchart[barstyle={black}]{\data}%
  \readpsbardata[header=false]{\data}%
    {example7.csv}%
  \psbarchart[barstyle={red},chartstyle=block,%
    barcolsep=0.8]{\data}%
\end{pspicture}
\end{pspicture}
```

5 To Do

- Provide commands to facilitate legend creation and placement.
- Allow the automatic labeling of bars with values from the data file.
- Allow the automatic labeling of bars with labels of the user's choice.
- Improve header parsing to allow commas within header entries.
- Add error-checking to ensure that each row of the data file contains the same number of entries, throwing an error in TEX or LATEX if not.
- Improve documentation.

References

[1] Timothy van Zandt. *PSTricks: PostScript Macros for Generic TEX*. `CTAN:/graphics/pstricks`, 1993.

[2] Herbert Voß. Pstricks — charts. `http://www.pstricks.de/Examples/Charts/chart.phtml#barI`, 2004.

[3] R. Stepanyan. *The `bardiag` Package*. `CTAN:/graphics/bardiag`, 2003.

[4] Herbert Voß. *`pstricks-add`: Additional Macros for `pstricks`*. `CTAN:/graphics/pstricks/contrib/pstricks-add`, 2004.