**\textsc{PCTeX}: Macros for Drawing \textsc{PCT}ures**

Michael J. Wichura
University of Chicago

**Overview**

In the preface to *The TeXbook*, Knuth describes TeX as a “typesetting system intended for the creation of beautiful books—and especially for books that contain a lot of mathematics”. \textsc{PCTeX} is a collection of \TeX macros by means of which \TeX users can easily instruct \TeX to typeset beautiful pictures as a part of their books—and especially mathematical figures, such as the one below.

Figure 1. Density $a$ and distribution function $\mathcal{A}$ of the arc sine distribution. The shaded area under the graph of $a$ is $\mathcal{A}(\beta) - \mathcal{A}(\alpha)$.

\begin{align*}
\text{Density} & \\
\quad a(t) = \frac{1}{(\pi \sqrt{t(1-t)})} \\
\text{Distribution function} & \\
\quad \mathcal{A}(x) = \frac{x}{\pi} \arcsin(\sqrt{x})
\end{align*}

That figure and the others in this article illustrate the main things you can do with \textsc{PCTeX}: place text into a \textsc{PCT}ure; construct $x$ and $y$ axes with tick marks, tick labels, and axis labels; draw rectangles and other things made out of horizontal and vertical rules; draw straight lines and curves (without recourse to special fonts); use line fills that can be [solid], [dotted], [dashed], or [otherwise]; and shade regions. In addition to these “primitive” graphics capabilities, \textsc{PCTeX} provides some “upper” level commands for drawing things like bar graphs, histograms, arrows, circles, and ellipses. Using \TeX’s powerful macro facilities, you can readily create other upper level commands that are tailored to your specific needs.

\textsc{PCTeX} has these advantages: (1) Figures become an integral part of the typesetting process. You can avoid having to leave the proper amount of space in your document for material that has to be created on some external device and later stripped into the finished product. (2) All of \TeX’s formatting capabilities are available for annotating your figures. In addition, that annotation will be done in the same fonts that you’re using in the rest of your document. (3) Just as \TeX is machine independent, so too is \textsc{PCTeX}. It doesn’t matter whether you’re working on a PC or mainframe computer. (4) Since typeset figures are embedded in the \textsc{dvi} file along with the rest of your document, all the advantages of \TeX’s device independent output accrue to them. In particular, you can revise away to your heart’s content on your local system until your \textsc{PCT}ures look just the way you want them to, and then you can have the final copy elegantly printed on a high resolution output device. (5) \textsc{PCTeX} can be extended using \TeX’s macro facilities, and can be used with \LaTeX.

On the other hand, \textsc{PCTeX} has several limitations: (1) \textsc{PCTeX} was expressly designed to facilitate the construction of pictures such as Figure 1. It simply is not the right tool for producing illustrations such as the lions that grace the title pages of *The TeXbook*. (2) Within the realm of mathematical figures, \textsc{PCTeX} itself doesn’t make 3D pictures or other complex things. Considering that \TeX provides fewer arithmetic capabilities than the simplest pocket calculator, that would be asking for too much. However \textsc{PCTeX} can be used as an interface between \TeX and a sophisticated graphics program. For example, Figure 2 was set using Fig and a Fig-to-\textsc{PCTeX} converter developed by Micah Beck. (3) \textsc{PCTeX} takes a while to draw a \textsc{PCT}ure. Figure 1 initially took 30 seconds on a Sun $3/60$, the bulk of the time going into producing the two curves. In subsequent drafts \textsc{PCTeX} used a special routine to replot the curves and did the whole \textsc{PCT}ure in 10 seconds. (By contrast, running on a Sun $3/60$, \TeX processes a page of “straight
Shakespeare used 14,376 words exactly once.

Table 1 lists the code that was used to construct Figure 3, in which the common logarithm of the number of words Shakespeare used exactly \( n \) times is graphed versus \( n \) for \( n = 1, 2, \ldots, 100 \). In tabular form the data are as follows:

\[
\begin{array}{ccc}
n & \nu_n & \lambda_n = \log_{10}(\nu_n) \\
1 & 14,376 & 4.1576 \\
2 & 4,343 & 3.6378 \\
3 & 2,292 & 3.3602 \\
\vdots & \vdots & \vdots \\
99 & 7 & 1.1761 \\
100 & 5 & 0.6990 \\
\end{array}
\]

Figure 3 has \( n \) on the horizontal (\( x \)) axis, and \( \lambda_n \) on the vertical (\( y \)) axis.
(4) \TeX draws coordinate axes along the edges of a prespecified rectangular “plot area”. The command on lines 13 - 14 sets up such a plot area, running from 0 to 100 along the x axis and from 0 to 4.301 = \log_{10}(20,000) along the y axis. The command on line 15 draws an axis along the bottom edge of this plot area. The analogous command on line 19 creates an axis along the left edge of the plot area; however this axis is subsequently shifted to lie along the line x = -5.

Some features of \TeX that emerged in the preceding example merit further discussion. The coordinate system is a key element of any \TeXture, since it governs the placement of everything that’s put into that \TeXture. You can set and reset the coordinate system at will, moving the location of the origin and changing the lengths of the x and y units; this facilitates work on \TeXtures like Figure 1 having several components. \TeX’s \texttt{\put} command and its relatives have options that make it easy to specify exactly how an object should be positioned relative to a given coordinate. You can choose between various horizontal (centered, left, right) and vertical (centered, above, below, baseline) orientations, and you can offset objects horizontally and vertically by amounts that don’t depend on the current coordinate system. \texttt{\axis} is \TeX’s most versatile command. You can: choose between bottom, left, top, and right axes; freely specify where ticks are to be placed; use any combination of unlabeled, numbered, or user-labeled ticks; have the ticks marks point out from the plot area or into it, or even extend all the way across it; specify axis labels and a plot heading; govern the length and thickness of axes and tick marks; and adjust the spacing between the various components of the graph framework. In general you can fine tune any part of a \TeXture; a few minor adjustments may make the difference between a figure that is merely presentable and one that is a work of art. One last point: you don’t have to prespecify to \TeX how much room a \TeXture will take up. \TeX automatically determines the size of a \TeXture and passes this information on to so that the \PICture can be positioned appropriately in the page layout.

What about lines and curves? \TeX has four interpolation modes, two of which are piecewise linear interpolation and piecewise quadratic interpolation. (The other two modes generate histograms and bar graphs.) Moreover, \TeX has four modes for line fill: solid, dotted, dashed, and user-specified. Given a list of coordinate points, \TeX’s \texttt{\plot} command connects those points using the current interpolation and line fill modes. Every option in \TeX has a default: the defaults for \texttt{\plot}ing are piecewise linear interpolation with solid line fill.

For example, in Figure 4 the edges of the simplex were drawn simply with
\begin{verbatim}
\plot 0 6 6 0 8.5 0 6 /
\end{verbatim}
while the y3 axis was created with
\begin{verbatim}
\setdashes
\plot 0 0 8.5 5.3125 /
\end{verbatim}
In Figure 1 the left-half of the arc sine density \(a\) was created by first placing the origin of the coordinate system at the point \(t = 0.5\) on the horizontal axis, and by then entering
\begin{verbatim}
\setquadratic
\inboundscheck
\plot -.485 2.6187 -.475 2.0388
-1.465 1.7320 -.465 1.7320
-.44 1.3403 -.40 1.0610
-.36 0.9174 -.32 0.8285
-.27 0.7564 -.22 0.7089
-.12 0.6558 0 0.6378 /
\end{verbatim}
Note that \(a(-.5-485) = 2.6187, a(-.475) = 2.0388,\) and so on. The “in bounds check” feature kept \TeX from plotting any points with vertical coordinates greater than 2.5. To plot the right half of \(a\) the same commands were used, but with the signs of the horizontal coordinates changed to ‘+’.

In general, to \texttt{\plot} any given curve you have to provide \TeX with a list of coordinate values, as opposed to, say, some mathematical formula.

Figure 4. Trinomial sample space and probabilities for \(m = 6\) and \(\pi = (1/3, 1/3, 1/3)\). For nonnegative integers \(y_1, y_2,\) and \(y_3\) summing to \(m\), the probability at the point \((y_1, y_2, y_3)\) is \[
\frac{m!}{y_1!y_2!y_3!} \pi_1^{y_1} \pi_2^{y_2} \pi_3^{y_3}.
\] (The y3 axis recedes into the plane of the page.)
However, you can create such a list using Fortran or C or whatever, store it in a file, and have `\plot` take its input from that file.

\TeX\ draws lines and curves by placing lots of periods close together. This takes a lot of time, because \TeX\ repeatedly has to lead the \TeX\ step by simple arithmetic step through complex calculations, like finding the distance between two points in a coordinate plane, for which it has no primitives. (\TeX\ can only do fixed point addition, subtraction, and multiplication, and division by integers.) And all those periods take up a lot of room in \TeX's memory. That's the down side; the up side is that there are no restrictions on the slope of lines and the curvature of arcs. By contrast, \LaTeX's picture environment draws lines and circles by piecing together characters from specially designed fonts. This is fast and uses little storage space, but the choice of slopes and radii is quite limited. Fortunately, this is one instance where you can have your cake and eat it too.

\begin{itemize}
\item \LaTeX\ knows how to use \LaTeX's picture objects, and \LaTeX\'s macros can be used inside \LaTeX\.
\end{itemize}

(Don't try this without first reading the relevant section from the \LaTeX\ manual.)

\begin{figure}
\centering
\includegraphics[width=\textwidth]{fig5}
\caption{Diagram showing how the probabilities for the response categories I, II, III, and IV in a certain proportional hazards model vary with $\eta$. Probabilities are shown as shaded areas; higher numbered categories have greater shade density.}
\end{figure}

\LaTeX\ shades a region by placing a “shading symbol” at every point of a “shading lattice” that falls within the region. You have control over both the symbol and lattice. In Figure 5 three different shading lattices were used, each with a `\fiverm` period as the shading symbol.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{fig6}
\caption{Suicide rates in western Europe per 100,000 population per year for the years (19xx) indicated.}
\end{figure}

\TeX\'s picture environment draws lines and circles by piecing together characters from specially designed fonts. This is fast and uses little storage space, but the choice of slopes and radii is quite limited. Fortunately, this is one instance where you can have your cake and eat it too.

\begin{itemize}
\item \LaTeX\ knows how to use \LaTeX's picture objects, and \LaTeX\'s macros can be used inside \LaTeX\.
\end{itemize}

(Don't try this without first reading the relevant section from the \LaTeX\ manual.)

**Extending \LaTeX**

Consider now how you might go about drawing Figure 6. To get started you could enter

\begin{verbatim}
\begin{picture}
\setcoordinatesystem units <5pt,11pt>
\setplotarea x from 0 to 25,
\y from 0 to 0
\linethickness=.15pt
{\eightpoint \axis top
ticks numbered from 0 to 25 by 5 / \%}
\linethickness=2pt
\end{picture}
\end{verbatim}

These commands establish the coordinate system, draw the axis, and set \LaTeX's line thickness parameter to 2 points for drawing the black bars. The problem that needs addressing is how to place the bars and their associated labels into the \LaTeX\ picture. You could do this easily with \LaTeX\'s bar graph command, or even with \TeX\'s `\align`. However the point of this section is to illustrate a technique that can often be put to good use when there isn't an upper level command that does what you want to do. Let's agree then that you have to solve this exercise using just \LaTeX\'s primitive commands `\put` and `\putrule`. The most direct solution would be

\begin{verbatim}
\putrule from 0 -1 to 24.1 -1
\put {Austria \sevenrm 75}

{\Br} <-5pt,-2pt> at 0 -1
\putrule from 0 -2 to 23.8 -2
\put {Denmark \sevenrm 73}

{\Br} <-5pt,-2pt> at 0 -2
\end{verbatim}

and a bunch of similar commands. (The `\Br`’s above stand for ‘baseline right’ orientation.)

It would be much less tedious to enter just the bare essentials, say in the form

\begin{verbatim}
\placebars
Austria 5 24.1
\end{verbatim}