A style option for rotated objects in LaTeX

Sebastian Rahtz and Leonor Barroca

Contents

List of Figures

List of Tables

Abstract

This article documents a LaTeX style option, 'rotating.sty', which perform all the different sorts of rotation one might like, including complete figures, within the context of a PostScript driver.

1 History

Sebastian Rahtz first wrote rotation macros in 1988, and has been fighting with them since. The original trigonometry macros came from Jim Walker (Dept Mathematics, University of South Carolina); we later borrowed the trigonometry macros in psfig 1.8. This set of macros is a complete overhaul of the practice of rotated LaTeX boxes destined for a PostScript driver.

We finally decided to clean these macros up and document them to bare-bones 'doc' standard in order to avoid doing some real work in January 1992. We must thank Frank Mittelbach and Rainer Schöpf for promoting this style of literate macro writing, and inspiring the rest of us to patch up our sorry efforts. We apologize for the fact that we have not attempted to make these macros compatible with 'plain'. Life is just too short.

A modification was supplied 9/2/92 by A. Mason to handle the *Textures* driver, chosen with the `\rotdriver{TEXTURES}` option. The 'sidewaysfigure' environment was fixed on 17/3/92 after suggestions by Rainer Schöpf.

2 Usage

This style option provides three LaTeX environments:

sideways prints the contents turned through 90 degrees counterclockwise;

turn prints the contents turned through an arbitrary angle;

rotate prints the contents turned through an arbitrary angle; but does *not* leave any space for the result.

A full set of examples are given in section 7. But now we present the documented code.

3 Driver-specific macros

We try to make this style driver-independent (!) by isolating all the usage of `\special` into one case statement later, so first we declare dummy values for the two macros which vary according to the driver, in case `\rotdriver` is never called, or produces no results.

```
\def\rot@start{}
\def\rot@end{}
```

This style option (potentially!) supports a variety of dvi drivers; the user must declare the one to be used.

`\rotdriver` The user can select the specials that should be used by calling the command `\rotdriver{`*drivername*`}`. Possible choices are:

- DVItoLN03
- DVItoPS
- DVIps
- emTEX
- *Textures*

This command can only be used in the preamble of the document. The list of drivers was created for compatibility with the 'changebar' macros (version 3.0 of November 1991 by Johannes Braams), but the code only exists in this style option for 'dvips' and 'dvitops'.

The argument should be case-insensitive, so it is turned into a string containing all uppercase characters. To keep some definitions local, everything is done within a group.

```
\def\rotdriver#1{%
  \bgroup\edef\next{\def\noexpand\tempa{#1}}%
    \uppercase\expandafter{\next}%
    \def\LN{DVITOLN03}%
    \def\DVItoPS{DVITOPS}%
    \def\DVIPS{DVIPS}%
    \def\emTeX{EMTEX}%
    \def\Textures{TEXTURES}%
```

The choice has to be communicated to the macros later on that will be called from within `\document`. For this purpose the control sequence `\rot@driversetup` is used. It receives a numeric value using `\chardef`.

```
    \global\chardef\rot@driversetup=0
    \ifx\tempa\LN
        \global\chardef\rot@driversetup=0\fi
    \ifx\tempa\DVItoPS
        \global\chardef\rot@driversetup=1\fi
    \ifx\tempa\DVIPS
        \global\chardef\rot@driversetup=2\fi
    \ifx\tempa\emTeX
        \global\chardef\rot@driversetup=3\fi
    \ifx\tempa\Textures
        \global\chardef\rot@driversetup=4\fi
  \egroup
```

We use a case statement to define the macros appropriate for each driver. We will define two commands, `\rot@start` and `\rot@end`, which assume that the macro `\rot@angle` produces the angle of rotation.

```
  \ifcase\rot@driversetup
```

The first case (0) is for 'dvitoln03', for compatibility with 'changebar.sty'; we don't have access to this, so pass by on the other side.

```
  % case 0
  \typeout{WARNING! ****
   no specials for LN03  rotation}
  \or
```

First real case, James Clark's 'dvitops'. This has not been well tested with dvitops; the figures of rotated paragraphs come out oddly. Dvitops has some unusual ways of

dealing with PostScript \special commands; they are kept in a list and dealt with all together. Each time you use an effect, you number it as a block.

```
\rot@count=1
\def\rot@start{\special{dvitops: origin
    rot\the\rot@count}%
\special{dvitops: begin rot\the\rot@count}}%
\def\rot@end{\special{dvitops: end}%
\special{dvitops: rotate rot\the\rot@count \space
    \the\rot@angle}%
\global\advance\rot@count by1}%
\or
```

Case 2, Rokicki's dvips (this code works with version 5.47). We simply emit some literal PostScript (code copied from Rokicki's 'rotate.sty').

```
\def\rot@start{\special{ps:gsave currentpoint
currentpoint translate \the\rot@angle\space
rotate  neg exch neg exch translate}}%
\def\rot@end{\special{ps:currentpoint
grestore moveto}}%
```

To be consistent, lets allow for emTEX one day performing here as well.

```
\or % case 3, emTeX
\typeout{WARNING! ***
 emTeX does no rotation at this time}
```

Lastly sofar, one for a Mac TEX. The *Textures* PostScript code has been modified from code provided by:

Charles Karney	Phone: +1 609 243 2607
Plasma Physics Laboratory	Fax: +1 609 243 2662
Princeton University	MFEnet: Karney@PPC.MFEnet
PO Box 451	Internet: Karney@Princeton.edu
Princeton, NJ 08543-0451	Bitnet: KarneyPPC.MFEnet@ANLVMS.Bitnet

The following assumptions are made about the PostScript that *Textures* generates:

1. A single transform is used for all *Textures* output.
2. The PostScript \special is bracketed by gsave ... grestore.
3. Immediately after the gsave, the coordinate system is translated so the origin is at the current point; and the y axis is flipped. (The y-axis isn't flipped any more... rotations are clockwise. A.M.)
4. *Textures* doesn't leave anything on the stack for long periods. (This simplifies restoring the default coordinate system.)

```
\or
\typeout{Textures rotation}
\def\rot@start{\special{postscript
  0 0 transform                 % Get current location in device
                                % coordinates.
  grestore                      % Undoes Textures gsave.
  matrix currentmatrix          % Save current transform on stack for use
                                % by \@unrotate.
  3 1 roll                      % Put transform at back of current location.
  itransform                    % Current location in Textures coords
  dup 3 -1 roll                 % Duplicate the location; x y ==> x y x y
  dup 4 1 roll exch
  translate                     % Translate origin to current location
 % 1 -1 scale                    % Flip y coordinate
  \the\rot@angle\space rotate   % Rotate by \@rotation
```

```
 % 1 -1 scale                    % Unflip y coordinate
  neg exch neg exch translate    % Translate origin back
  gsave}}                        % To match grestore
\def\rot@end{\special{postscript
  grestore                       % Undoes Textures gsave
  setmatrix                      % Set current transform to value saved on
                                 % stack.  (Hopefully, it's still there.)
  gsave}}                        % To match grestore
\else
\typeout{WARNING! ***
 unknown  driver - no rotation}
\fi
}
```

Finaly, we will need boxes to take copies of what we are rotating, and will need some registers to store sizes and angles.

```
\newsavebox{\rot@box}%
\newsavebox{\rot@tempbox}%
\newdimen\rot@temp
\newdimen\rot@width
\newdimen\rot@height
\newdimen\rot@depth
\newdimen\rot@right
\newdimen\rot@left
\newcount\rot@angle
\newcount\rot@count
\newcount\rot@circle
```

4 Rotation environments

The basic idea is to put the contents of the environment into a box, change the depth, width and height of that box (as known to TEX) if necessary, and then rotate it.

4.1 Sideways

The 'sideways' environment simply turns the box through 90°, so no trigonometry is necessary.

```
\sideways  \def\sideways{\savebox{\rot@box}\bgroup}

\endsideways  \def\endsideways{\egroup%
              \rot@angle-90
              \rot@width\ht\rot@box
              \advance\rot@width by\dp\rot@box
              \rot@height\wd\rot@box
              \wd\rot@box0pt%
              \dp\rot@box0pt%
              \ht\rot@box0pt%
              \rule{\rot@width}{0pt}%
              \rlap{\rule{0pt}{\rot@height}}%
              \rot@start\usebox{\rot@box}\rot@end%
              }
```

4.2 Rotate

In the case of the rotate environment, we are just going to turn the box without working out the space for it, so again no trigonometry.

```
\rotate  \def\rotate#1{%
         \global\rot@angle=#1
         \savebox{\rot@box}\bgroup}
```

```
\endrotate  \def\endrotate{\egroup%
            \rot@start%
            \dp\rot@box=0pt\wd\rot@box=0pt\ht\rot@box=0pt%
            \usebox{\rot@box}%
            \rot@end%
            }%
```

4.3 Turn

This is the tricky one. We rotate the box, and work out how much space to leave for it on the page. We deal with the box as a whole, i.e. both depth and height are joined to make a single height. After working out the space taken up this box after rotation, we can worry about placing it correctly in relation to the baseline.

The original philosophy was that given a box with width W and height H, then its height after rotation by R is $W \times \sin(R) + H \times \cos(R)$, and it extends $W \times \cos(R)$ to the right and $H \times \sin(R)$ to the left of the original bottom left corner (formula courtesy of Nico Poppelier). This works fine in the 'top right' quadrant, but causes problems in the other quadrants, so we adopted a rather more brute-force scheme. We consider three vertices of the original unrotated box (A, B and C in Figure 1), and calculate their x, y co-ordinates after rotation by R degrees. This deals with the *top half* of the box only, that which comes above the baseline; for the lower half (below the baseline), we deal with vertices D and E. Given original dimensions of the box as width W height H, and depth D, the formulae for calculating new positions are:

$$\begin{aligned}
Ax &= W \times \cos(R) \\
Ay &= W \times \sin(R) \\
Bx &= (W \times \cos(R)) - (H \times \sin(R)) \\
By &= (W \times \sin(R)) + (H \times \cos(R)) \\
Cx &= H \times \cos(R + 90) \\
Cy &= H \times \sin(R + 90) \\
Dx &= D \times \cos(R + 270) \\
Dy &= D \times \sin(R + 270) \\
Ex &= (D \times \cos(R + 270)) - (W \times \sin(R + 270)) \\
Ey &= (D \times \sin(R + 270)) + (W \times \cos(R + 270))
\end{aligned}$$

We could work out how far the rotated box extends to the right of the 'starting point' (S in Figure 1) by taking the largest of (Bx, Cx, Dx, Ex); how far it extends to the left by taking the smallest of (Bx, Cx, Dx, Ex); how far above the baseline with the largest of (By, Cy, Dy, Ey); and how far below the baseline with the smallest of (By, Cy, Dy, Ey). But that would be a bit slow, so we simplify matters by working out first which quadrant we are in, and then picking just the right values.

```
  \turn  \def\turn#1{%
         \global\rot@angle=#1
         \savebox{\rot@box}\bgroup}%

\endturn  \def\endturn{%
          \egroup%
```

Because PostScript works clockwise, and because we conceptualize the trigonometry in a counter-clockwise way, we temporarily reverse the direction of the angle:

```
\multiply\rot@angle by-1
```

We are going to need to know the sines and cosines of three angles: R, $R + 90$ and $R + 270$. Simplest to calculate all these now; in fact we can work it out from just two calculations.

```
\Sine{\rot@angle}%
\let\sineA\sine
```

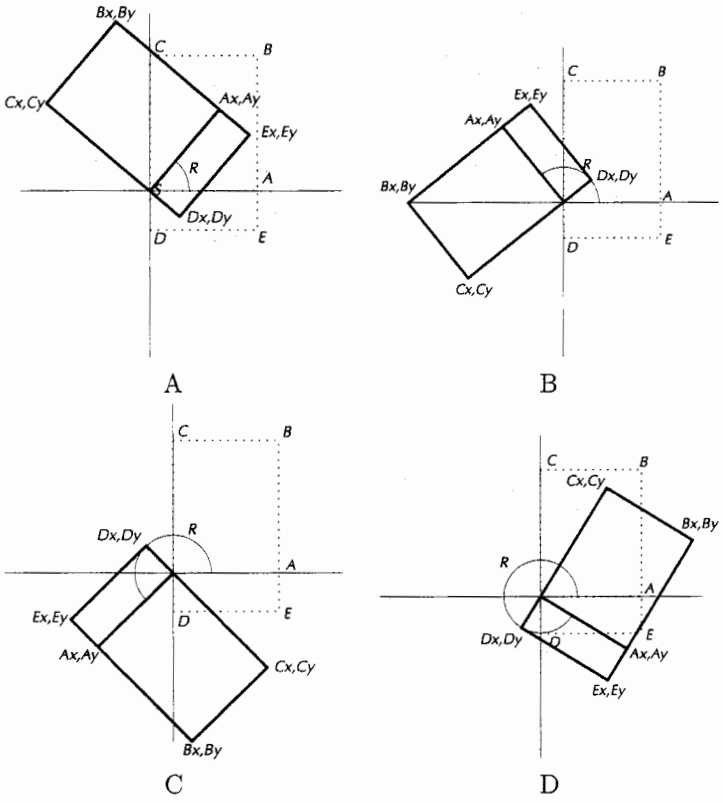


Figure 1: Working out the position of a box by considering x, y coordinates of five vertices

```
\advance\rot@angle by90%
\Sine{\rot@angle}%
\let\sineB\sine
\def\sineC{-\sine}%
\def\cosineA{-\sineC}%
\def\cosineB{-\sineA}%
\let\cosineC\sineA
\advance\rot@angle by-90%
```

Now we can calculate the co-ordinates of the relevant vertices. To make the coding easier, we define the formulae given above as TEX macros (just the ones we ever use).

```
\def\rot@Bx{\rot@temp\cosineA\wd\rot@box
   \advance\rot@temp by -\sineA\ht\rot@box}%
\def\rot@By{\rot@temp\sineA\wd\rot@box
  \advance\rot@temp by \cosineA\ht\rot@box}%
\def\rot@Cx{\rot@temp\cosineB\ht\rot@box}%
\def\rot@Cy{\rot@temp\sineB\ht\rot@box}%
\def\rot@Dx{\rot@temp\cosineC\dp\rot@box}%
\def\rot@Dy{\rot@temp\sineC\dp\rot@box}%
\def\rot@Ex{\rot@temp\cosineC\dp\rot@box
   \advance\rot@temp by -\sineC\wd\rot@box}%
\def\rot@Ey{\rot@temp\sineC\dp\rot@box
   \advance\rot@temp by \cosineC\wd\rot@box}%
```

Now a straightforward 'if' condition to see which quadrant we are operating in; but if the angle is negative, first add 360.

```
\rot@circle\rot@angle
\ifnum\rot@circle<0
     \advance\rot@circle by 360
\fi
\ifnum\rot@circle<90
```

First quadrant: Height = By, Right = Ex, Left = Cx, Depth = Dy

```
\rot@By\rot@height\rot@temp
\rot@Ex\rot@right\rot@temp
\rot@Cx\rot@left\rot@temp
\rot@Dy\rot@depth\rot@temp
\else
     \ifnum\rot@circle<180
```

Second quadrant: Height = Ey, Right = Dx, Left = Bx, Depth = Cy

```
      \rot@Ey\rot@height\rot@temp
      \rot@Dx\rot@right\rot@temp
      \rot@Bx\rot@left\rot@temp
      \rot@Cy\rot@depth\rot@temp
     \else
           \ifnum\rot@circle<270
```

Third quadrant: Height = Dy, Right = Cx, Left = Ex, Depth = By

```
          \rot@Dy\rot@height\rot@temp
          \rot@Cx\rot@right\rot@temp
          \rot@Ex\rot@left\rot@temp
          \rot@By\rot@depth\rot@temp
           \else
```

Fourth quadrant: Height = Cy, Right = Bx, Left = Dx, Depth = Ey

```
          \rot@Cy\rot@height\rot@temp
          \rot@Bx\rot@right\rot@temp
          \rot@Dx\rot@left\rot@temp
```

```
            \rot@Ey\rot@depth\rot@temp
          \fi
      \fi
\fi
```

Put the angle back to what it was before, to pass to PostScript

```
\multiply\rot@angle by-1
```

At the end of all that nonsense, `\rot@height` contains the amount *above* the baseline we need to leave for the rotated box we are dealing with, and `\rot@depth` the amount *below* the baseline. `\rot@left` and `\rot@right` are offsets to left and right which we need to take into account. We are going to set the size of the box we are dealing with to 0 all round, and put in some struts to force TEX to leave space. We will position ourselves at the point where the bottom left-hand corner of the top half of box *would* have been, then swing the box round by that corner. Thinking about this drives you mad.

```
\wd\rot@box=0pt\ht\rot@box=0pt%
```

The left adjustment comes out negative, so be careful:

```
\ifdim\rot@left<0pt
     \rule{-\rot@left}{0pt}%
\fi
```

Put in struts (not advancing forward at all), for the height and depth.

```
\ifdim\rot@height>0pt
     \rlap{\rule{0pt}{\rot@height}}%
\fi
\ifdim\rot@depth<0pt
     \rlap{\rule[\rot@depth]{0pt}{-\rot@depth}}%
\fi
```

Finally emit the PostScript code to start rotation, output the box, end the rotation, and leave some space at the right if needed.

```
\rot@start\usebox{\rot@box}\rot@end%
\ifdim\rot@right>0pt
      \rule{\rot@right}{0pt}%
\fi
}%
```

5 Rotated tables and figures

Now we present some macros adapted from those by James Dolter (`jdolter@sawtooth.eecs.umich.edu`) which provide a complete environment for sideways figures and tables. We define two environments `sidewaysfigure` and `sidewaystable` that fit in with the normal table and figure floats. These are 'fixed' environments that just do 90 degree rotation, but it would be easy to parameterize this to do other rotations if needed (the mind boggles...).

First a generalised '`rotfloat`' environment. We have to take a copy of LATEX's float macros, in order to change the assumed width of a float being `\columnwidth`. We want it to work on a width of `\textheight` so that when we rotate the float, it comes out the right height. This is not actually very satisfactory, since what we *really* want is for rotated floats to occupy the space they actually *use*. The captions are a problem — since they can precede the figure or table, we cannot set them in a box of the right width (i.e. the *height* of the forthcoming object), because it has not happened yet. The result of these difficulties is that rotated figures always end up as full page figures.

```
\def\@rotfloat#1{\@ifnextchar[{\@xrotfloat{#1}}{%
\edef\@tempa{\noexpand\@xrotfloat{#1}[\csname fps@#1\endcsname]}\@tempa}}
```

```
\def\@xrotfloat#1[#2]{\ifhmode \@bsphack\@floatpenalty -\@Mii\else
   \@floatpenalty-\@Miii\fi\def\@captype{#1}\ifinner
      \@parmoderr\@floatpenalty\z@
    \else\@next\@currbox\@freelist{\@tempcnta\csname ftype@#1\endcsname
       \multiply\@tempcnta\@xxxii\advance\@tempcnta\sixt@@n
       \@tfor \@tempa :=#2\do
                        {\if\@tempa h\advance\@tempcnta \@ne\fi
                         \if\@tempa t\advance\@tempcnta \tw@\fi
                         \if\@tempa b\advance\@tempcnta 4\relax\fi
                         \if\@tempa p\advance\@tempcnta 8\relax\fi
         }\global\count\@currbox\@tempcnta}\@fltovf\fi
    \global\setbox\@currbox\vbox\bgroup
```

The only part changed is the setting of `\hsize` within the `\vbox` to be `\textheight` instead of `\columnwidth`.

```
   \hsize\textheight \@parboxrestore
}
```

We copy the '`\end@float`' macro and emend it to rotate the box we produce in a float.

```
\def\end@rotfloat{\par\vskip\z@\egroup%
   \ifnum\@floatpenalty <\z@
    \global\setbox\rot@tempbox\box\@currbox
    \global\setbox\@currbox\vbox{\centerline{\begin{turn}{-90}%
    \box\rot@tempbox\end{turn}}}%
     \@cons\@currlist\@currbox
     \typeout{Adding sideways figure to list,
        \the\ht\@currbox\space by \the\wd\@currbox}%
     \ifdim \ht\@currbox >\textheight
        \@warning{Float larger than \string\textheight}%
        \ht\@currbox\textheight \fi
     \ifnum\@floatpenalty <-\@Mii
        \penalty -\@Miv
        \@tempdima\prevdepth
        \vbox{}%
        \prevdepth \@tempdima
        \penalty\@floatpenalty
     \else \vadjust{\penalty -\@Miv
        \vbox{}\penalty\@floatpenalty}\@esphack
     \fi\fi}
```

The following definitions set up two environments, `sideways table` and `sidewaysfigure`, which uses this type of float. Naturally, users may need to change these to suit their local style. Both contribute to the normal lists of figures and tables.

```
\let\l@sidewaysfigure\l@figure
\let\c@sidewaysfigure\c@figure
\let\p@sidewaysfigure\p@figure
\let\thesidewaysfigure\thefigure
\def\fps@sidewaysfigure{tbp}
\def\ftype@sidewaysfigure{1}
\def\ext@sidewaysfigure{lof}
\def\fnum@sidewaysfigure{Figure \thefigure}
\def\sidewaysfigure{\let\make@caption\make@rcaption
\@rotfloat{sidewaysfigure}}
%
\let\endsidewaysfigure\end@rotfloat
%
\let\l@sidewaystable\l@table
\let\c@sidewaystable\c@table
```

```
\let\p@sidewaystable\p@table
\let\thesidewaystable\thetable
\def\fps@sidewaystable{tbp}
\def\ftype@sidewaystable{2}
\def\ext@sidewaystable{lot}
\def\fnum@sidewaystable{Table \thetable}
\def\sidewaystable{\let\make@caption\make@rcaption
\@rotfloat{sidewaystable}}
\let\endsidewaystable\end@rotfloat
```

We need to copy a standard `\@makecaption` to set the caption on a width of the *height* of the float (i.e. the text height). This macro is normally defined in LaTeX style files, so style file writers who change that will also need to redefine `\r@caption`.

```
\long\def\@makercaption#1#2{%
   \vskip 10\p@
   \setbox\@tempboxa\hbox{#1: #2}%
   \ifdim \wd\@tempboxa >\vsize
       #1: #2\par
   \else
       \hbox to\vsize{\hfil\box\@tempboxa\hfil}%
   \fi}%
```

5.1 Rotated captions only

Sometimes you may find that the rotation of complete figures does not give quite the right result, since they always take up the whole page. You may prefer to rotate the caption and the float contents separately within a conventional figure. Here we offer a suggestion for a `\rotcaption` command, which inserts the caption rotated by 90°. It is essentially a copy of the normal captioning code. Styles which define the `\@makecaption` command may also need to define `\@makerotcaption`.

```
\def\rotcaption{\refstepcounter\@captype\@dblarg{\@rotcaption\@captype}}
\long\def\@rotcaption#1[#2]#3{%
\addcontentsline{\csname ext@#1\endcsname}{#1}{%
 \protect\numberline{\csname the#1\endcsname}{\ignorespaces #2}}%
  \par
  \begingroup
    \@parboxrestore
    \normalsize
    \@makerotcaption{\csname fnum@#1\endcsname}{#3}%
  \endgroup}
\long\def\@makerotcaption#1#2{%
 \setbox\@tempboxa\hbox{#1: #2}%
 \ifdim \wd\@tempboxa > .8\vsize
    \begin{rotate}{-90}%
    \begin{minipage}[b]{.8\textheight}#1 #2\end{minipage}%
    \end{rotate}\par
 \else%
    \mbox{\begin{rotate}{-90}\box\@tempboxa\end{rotate}}%
 \fi
 \hspace{12pt}%
}
```

While we are doing useful new environments, why not add landscape slides?

```
\newenvironment{landscapeslide}[1]{\begin{sideways}%
\vbox\bgroup\begin{slide}{#1}}%
{\end{slide}\egroup\end{sideways}}
```

6 Trigonometry macros

Now the trigonometry macros which are borrowed from psfig1.8; the original author is not credited there, so we cannot do so either. All we have done is remove some spurious spaces which were creeping into my output (and causing havoc!), and put the comments in 'doc' style.

```
\chardef\letter = 11
\chardef\other = 12
```

Turn me on to see TEX hard at work ...

```
\newif\ifdebug%
```

don't need to compute some values

```
\newif\ifc@mpute%
```

but assume that we do

```
\c@mputetrue%
\let\then =\relax
\def\r@dian{pt }%
\let\r@dians =\r@dian
\let\dimensionless@nit =\r@dian
\let\dimensionless@nits =\dimensionless@nit
\def\internal@nit{sp }%
\let\internal@nits =\internal@nit
\newif\ifstillc@nverging
\def\Mess@ge #1{\ifdebug\then\message {#1}\fi}%
```

Things that need abnormal catcodes

```
{%
\catcode `\@ =\letter
\gdef\nodimen {\expandafter\n@dimen\the\dimen}%
\gdef\term #1 #2 #3%
```

freeze parameter 1 (count, by value)

```
{\edef\t@ {\the #1}%
```

freeze parameter 2 (dimen, by value)

```
\edef\t@@ {\expandafter\n@dimen\the #2\r@dian}%
\t@rm {\t@} {\t@@} {#3}%
}%
\gdef\t@rm #1 #2 #3%
{{%
\count 0 = 0
\dimen 0 = 1\dimensionless@nit
\dimen 2 = #2\relax
\Mess@ge {Calculating term #1 of\nodimen 2}%
\loop
\ifnum\count 0 < #1
\then\advance\count 0 by 1
\Mess@ge {Iteration\the\count 0\space}%
\Multiply\dimen 0 by {\dimen 2}%
\Mess@ge {After multiplication, term =\nodimen 0}%
\Divide\dimen 0 by {\count 0}%
\Mess@ge {After division, term =\nodimen 0}%
\repeat
\Mess@ge {Final value for term #1 of
\nodimen 2\space is\nodimen 0}%
\xdef\Term {#3 =\nodimen 0\r@dians}%
\aftergroup\Term
```

```
}}%
\catcode `\p =\other
\catcode `\t =\other
```

throw away the "pt"

```
\gdef\n@dimen #1pt{#1}%
}%
```

just a synonym

```
\def\Divide #1by #2{\divide #1 by #2}%
```

allows division of a dimen by a dimen

```
\def\Multiply #1by #2%
```

should really freeze parameter 2 (dimen, passed by value)

```
{{%
\count 0 = #1\relax
\count 2 = #2\relax
\count 4 = 65536
\Mess@ge {Before scaling, count 0 =\the\count 0\space and
count 2 =\the\count 2}%
```

do our best to avoid overflow

```
\ifnum\count 0 > 32767%
\then\divide\count 0 by 4
\divide\count 4 by 4
\else\ifnum\count 0 < -32767
\then\divide\count 0 by 4
\divide\count 4 by 4
\else
\fi
\fi
```

while retaining reasonable accuracy

```
\ifnum\count 2 > 32767%
\then\divide\count 2 by 4
\divide\count 4 by 4
\else\ifnum\count 2 < -32767
\then\divide\count 2 by 4
\divide\count 4 by 4
\else
\fi
\fi
\multiply\count 0 by\count 2
\divide\count 0 by\count 4
\xdef\product {#1 =\the\count 0\internal@nits}%
\aftergroup\product
}}%
```

$\sin(x + 90) = \sin(180 - x)$

```
\def\r@duce{\ifdim\dimen0 > 90\r@dian\then%
\multiply\dimen0 by -1
\advance\dimen0 by 180\r@dian
\r@duce
```

$\sin(-x) = \sin(360 + x)$

```
\else\ifdim\dimen0 < -90\r@dian\then%
\advance\dimen0 by 360\r@dian
\r@duce
```

```
\fi
\fi}%
\def\Sine#1%
{{%
       \dimen 0 = #1\r@dian
       \r@duce
       \ifdim\dimen0 = -90\r@dian\then
              \dimen4 = -1\r@dian
              \c@mputefalse
       \fi
       \ifdim\dimen0 = 90\r@dian\then
              \dimen4 = 1\r@dian
              \c@mputefalse
       \fi
       \ifdim\dimen0 = 0\r@dian\then
       \dimen4 = 0\r@dian
       \c@mputefalse
\fi
%
\ifc@mpute\then
```

convert degrees to radians

```
%
\divide\dimen0 by 180
\dimen0=3.141592654\dimen0
%
```

a well-known constant

```
\dimen 2 = 3.1415926535897963\r@dian%
```

we only deal with $-\pi/2 : \pi/2$

```
\divide\dimen 2 by 2%
\Mess@ge {Sin: calculating Sin of\nodimen 0}%
```

see power-series expansion for sine

```
\count 0 = 1%
\dimen 2 = 1\r@dian%
\dimen 4 = 0\r@dian%
\loop
```

then we've done

```
\ifnum\dimen 2 = 0%
\then\stillc@nvergingfalse
\else\stillc@nvergingtrue
\fi
\ifstillc@nverging%
```

then calculate next term

```
\then\term {\count 0} {\dimen 0} {\dimen 2}%
\advance\count 0 by 2
\count 2 =\count 0
\divide\count 2 by 2
```

signs alternate

```
\ifodd\count 2%
\then\advance\dimen 4 by\dimen 2
\else\advance\dimen 4 by -\dimen 2
\fi
\repeat
```

```
\fi
\xdef\sine {\nodimen 4}%
}}%
```

Now the Cosine can be calculated easily by calling `\Sine`

```
%
\def\Cosine#1{\ifx\sine\UnDefined\edef\Savesine{\relax}\else
\edef\Savesine{\sine}\fi
{\dimen0=#1\r@dian\advance\dimen0 by 90\r@dian
\Sine{\nodimen 0}%
\xdef\cosine{\sine}%
\xdef\sine{\Savesine}}}
% end of trig stuff
```

And that's the end of the trigonometry macros. Finally, we'll set up a default for the driver:

```
\rotdriver{dvips}
```

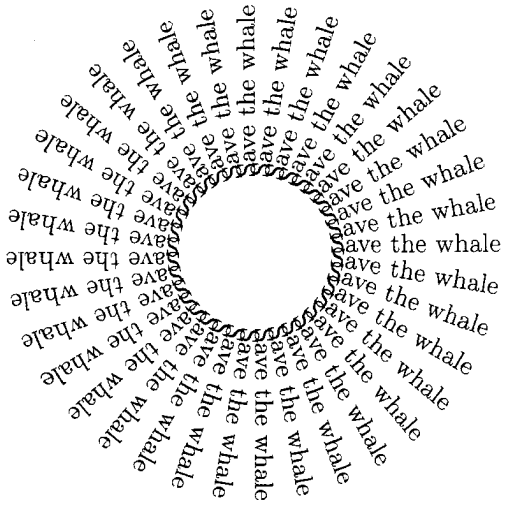
7 Examples

'Rotate' provides a generalised rotation environment, where the text will be rotated (clockwise, as is normal in PostScript) by the number of degrees specified as a parameter to the environment, but no special arrangement is made to find space for the result. Note the % at the end of `\begin{rotate}{56}` — this is vital to prevent a space getting into the rotated text.

Start here End here
Save whales

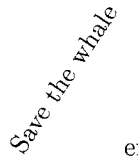
```
Start here
  \begin{rotate}{56}%
   Save whales
  \end{rotate}
End here
```

A complete example of rotating text without leaving space would the 'Save the whale' text written at 10 degree intervals round the compass. We use 'rlap' to ensure that all the texts are printed at the same point. Just to show that TEX can handle PostScript muckings-about properly...




```
\newcount\wang
\newsavebox{\wangtext}
\newdimen\wangspace
\def\wheel#1{\savebox{\wangtext}{#1}%
\wangspace\wd\wangtext
\advance\wangspace by 1cm%
\centerline{%
\rule{0pt}{\wangspace}%
\rule[-\wangspace]{0pt}{\wangspace}%
\wang=-180\loop\ifnum\wang<180
\rlap{\begin{rotate}{\the\wang}%
\rule{1cm}{0pt}#1\end{rotate}}%
\advance\wang by 10\repeat}}
\wheel{Save the whale}
```

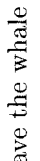
If the user desires LaTeX to leave space for the rotated box, then 'turn' is used:




```
Start here \begin{turn}{-56}%
  Save the whale
 \end{turn} end here
```

Start here end here

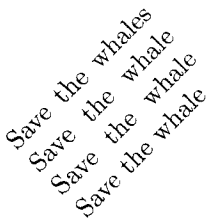
The environment 'Sideways' is a special case, setting the rotation to -90, and leaving the correct space for the rotated box.




```
Start here
\begin{sideways}%
Save the whale
\end{sideways}
End here
```

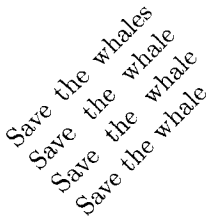
Start here End here

If you deal with whole paragraphs of text, you realize that TeX boxes are not as simple as they sometimes look: they have a height *and* a depth. So when you rotate, you rotate about the point on the left-hand edge of the box that meets the baseline. The results can be unexpected, as shown in the full set of paragraph rotations in Figures 2 and 3. If you really want to turn a paragraph so that it appears to rotate about the *real* bottom of the TeX box, you have to adjust the box in the normal LaTeX way:



Start End

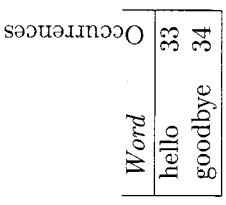
```
\newsavebox{\foo}
\savebox{\foo}{\parbox{1in}{Save
the whales Save the whale
Save the whale
Save the whale}}%
Start
\begin{turn}{-45}\usebox{\foo}\end{turn}
End
```



Start End

```
\savebox{\foo}{\parbox[b]{1in}{Save
the whales Save the whale
Save the whale
Save the whale}}%
Start
\begin{turn}{-45}\usebox{\foo}\end{turn}
End
```

We can set tabular material in this way; at the same time, we demonstrate that the rotation can be nested:



Word	Occurrences
hello	33
goodbye	34

```
\begin{sideways}
\rule{1in}{0pt}
\begin{tabular}{|lr|}
\em Word & \begin{rotate}{-90}%
Occurrences\end{rotate}
\\
\hline
hello & 33\\
goodbye & 34\\
\hline
\end{tabular}
\end{sideways}
```

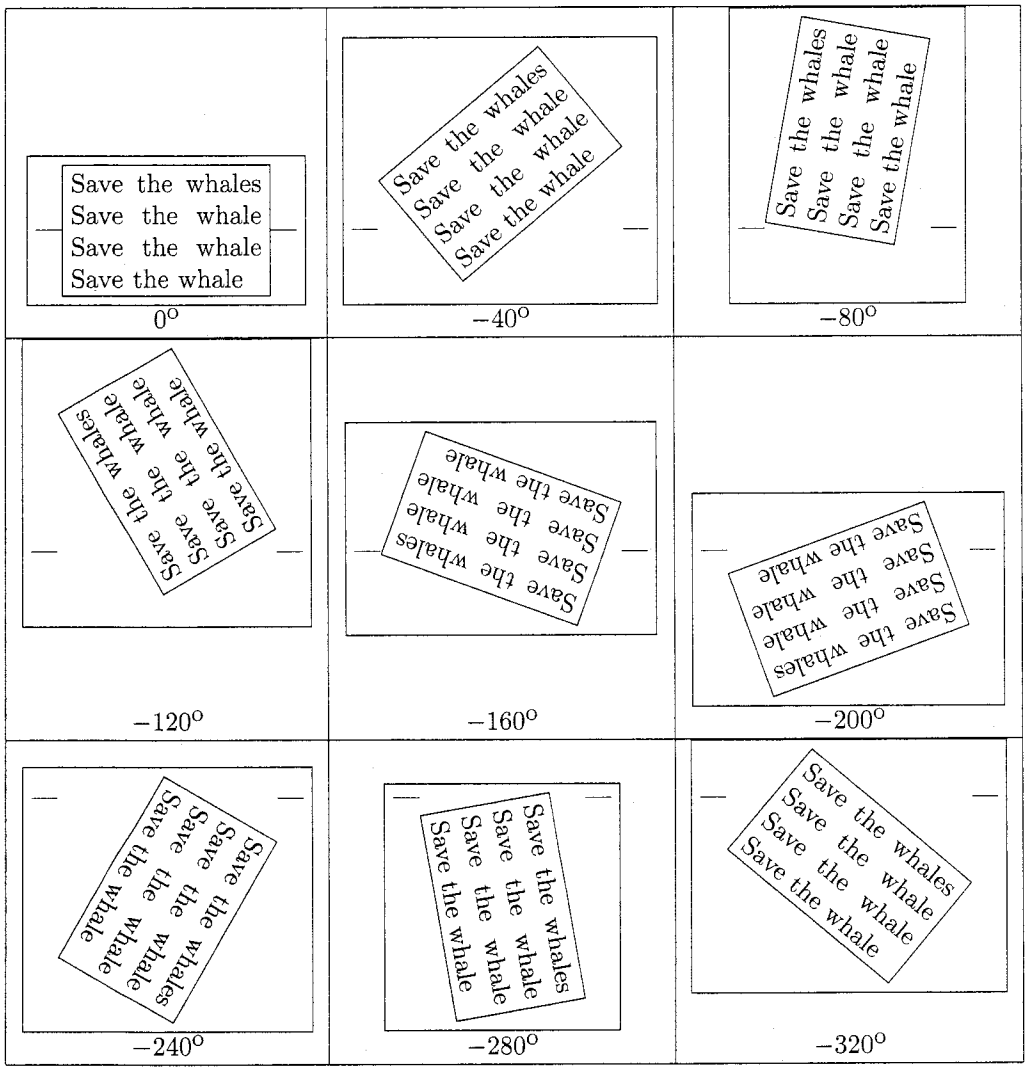


Figure 2: Rotation of paragraphs between 0 and -320°

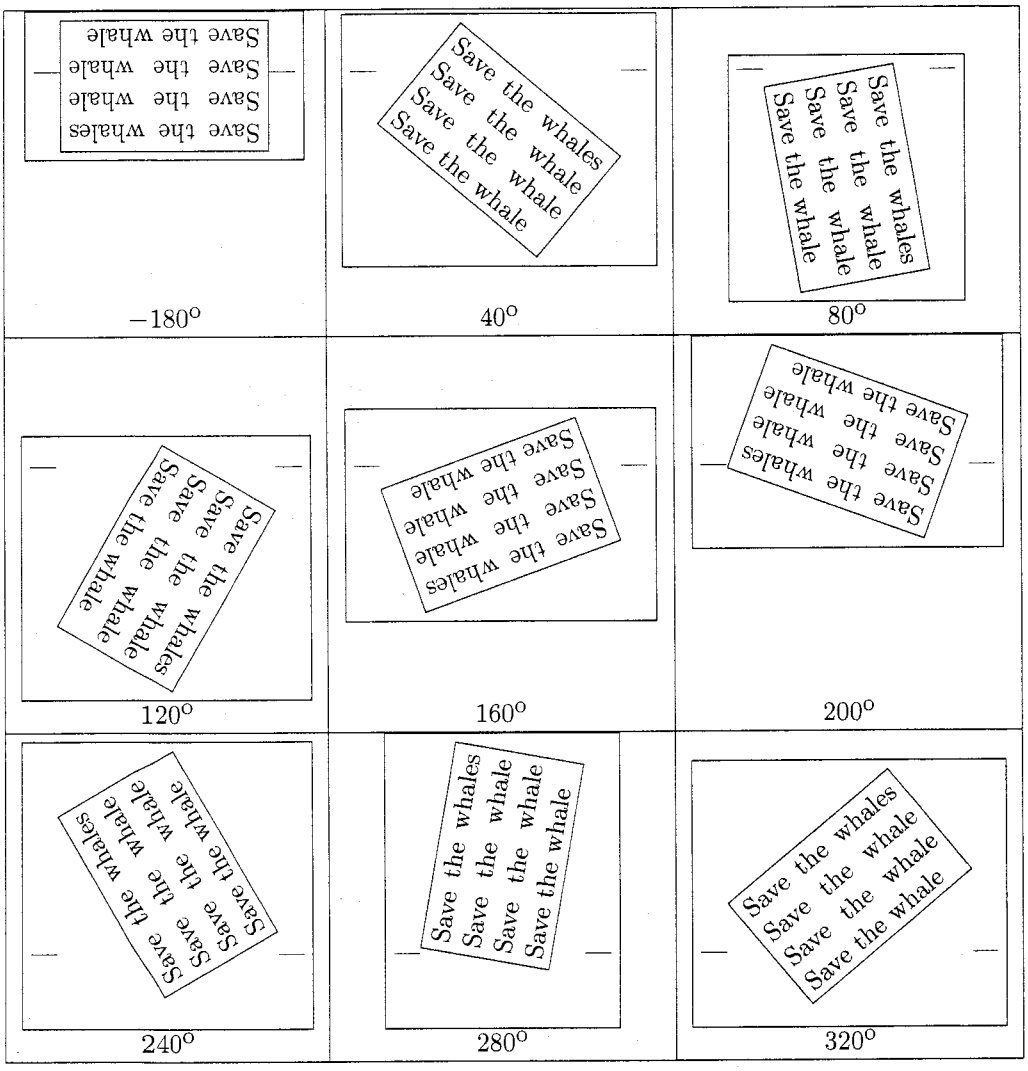


Figure 3: Rotation of paragraphs between 0 and 320°

Column 1	Column 2	Column 3
1	2	3
4	5	6
7	8	9

```
\begin{quote}
\rule{0pt}{1.5in}\begin{tabular}{rrr}
\begin{rotate}{-45}Column 1\end{rotate}&
\begin{rotate}{-45}Column 2\end{rotate}&
\begin{rotate}{-45}Column 3\end{rotate}\\
\hline
1& 2& 3\\
4& 5& 6\\
7& 8& 9\\
\hline
\end{tabular}
\end{quote}
```

Column 1	Column 2	Column 3
1	2	3
4	5	6
7	8	9

```
\begin{quote}
\begin{tabular}{rrr}
\begin{turn}{-45}Column 1\end{turn}&
\begin{turn}{-45}Column 2\end{turn}&
\begin{turn}{-45}Column 3\end{turn}\\
\hline
1& 2& 3\\
4& 5& 6\\
7& 8& 9\\
\hline
\end{tabular}
\end{quote}
```

Column 1	Column 2	Column 3
1	2	3
4	5	6
7	8	9

```
\begin{quote}
\rule{0pt}{1.5in}\begin{tabular}{rrr}
\begin{rotate}{-45}Column 1\end{rotate}
\rule{.5cm}{0pt}&
\begin{rotate}{-45}Column 2\end{rotate}
\rule{.5cm}{0pt}&
\begin{rotate}{-45}Column 3\end{rotate}
\rule{.5cm}{0pt}\\
\hline
1& 2& 3\\
4& 5& 6\\
7& 8& 9\\
\hline
\end{tabular}
\end{quote}
```

PRIMARY UNITS	STUDY AREA	NUMBER OF SITES					ACCEPT or REJECT NULL HYPOTH
			IN BOUNDARY ZONE				
				EXPECTED			
		TOT	OBS	FROM	TO		
	FULL SAMPLE	41	31	10.3	27.0		REJECT
	SAMPLE AREA 1	23	16	4.3	16.7		ACCEPT
	SAMPLE AREA 2	18	15	2.8	13.7		REJECT
	RUSHEN	13	9	1.2	10.4		ACCEPT
	ARBORY	10	7	0.6	8.8		ACCEPT
	MAROWN	10	8	0.4	8.6		ACCEPT
	SANTON	8	7	0.0	7.3		ACCEPT

```
\begin{sideways}
\begin{tabular}{|l|c|c|c|c|c|p{1in}|}
\hline
&&\multicolumn{4}{c}{NUMBER OF SITES}\vline &ACCEPT or\\
\cline{3-6} &STUDY AREA&&\multicolumn{3}{c}{%
IN BOUNDARY ZONE}\vline&REJECT\\
\cline{4-6}&&&&\multicolumn{2}{c}{EXPECTED}
\vline&NULL\\
\cline{5-6}&&TOT&OBS&FROM&TO&HYPOTH\\
\cline{2-7}
&FULL SAMPLE&41&31&10.3&27.0&REJECT\\
&SAMPLE AREA 1&23&16&4.3&16.7&ACCEPT\\
&SAMPLE AREA 2&18&15&2.8&13.7&REJECT\\
&RUSHEN&13&9&1.2&10.4&ACCEPT\\
&ARBORY&10&7&0.6&8.8&ACCEPT\\
&MAROWN&10&8&0.4&8.6&ACCEPT\\
\rule{0.5cm}{0pt}
\begin{rotate}{-90}PRIMARY UNITS%
\end{rotate}\rule{0.5cm}{0pt}
&SANTON&8&7&0.0&7.3&ACCEPT\\
\hline
\end{tabular}
\end{sideways}
```

If you are interested in setting rotated material in tables or figures, this presents no problem. Figure 4 shows how PostScript files which are being incorporated using `psfig` can be rotated at will, while Figure 5 shows, in contrast, how `psfig` itself handles rotation. It is also possible to rotate the whole of the figure environment, including caption, by using the `sidewaysfigure` and `sidewaystable` environments in place of `figure` and `table`. The code used to produce figures 1–6 is as follows:

Figure 1 `\begin{sidewaystable}`

```
\centering
\caption{This is a narrow  table, which should be centred vertically
on the final page.\label{rotfloat1}}
  \begin{tabular}{|ll|}
\hline
    a & b \\
    c & d \\
    e & f \\
    g & h \\
    i & j \\
\hline
  \end{tabular}
\end{sidewaystable}
```

Figure 2 `\begin{sidewaystable}`

```
\centering
\begin{tabular}{|llllllllp{1in}lp{1in}|}
```

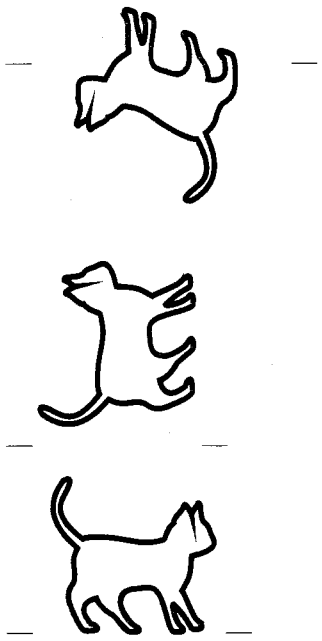
```
\hline
Context   &Length   &Breadth/   &Depth   &Profile   &Pottery   &Flint
   &Animal   &Stone   &Other    &C14 Dates \\
          &         &Diameter   &        &          &          &
   &Bones&&&\\
\hline
&&&&&&&&&&\\
\multicolumn{10}{|l|}{\bf Grooved Ware}&\\
784       &---        &0.9m       &0.18m   &Sloping U &P1       &$\times$46
  &  $\times$8      &&       $\times$2 bone&  2150$\pm$ 100 BC\\
785       &---        &1.00m      &0.12    &Sloping U &P2--4    &$\times$23
  &  $\times$21     & Hammerstone &---&---\\
962       &---        &1.37m      &0.20m   &Sloping U &P5--6    &$\times$48
  &  $\times$57*    & ---&     ---&1990 $\pm$ 80 BC (Layer 4) 1870 $\pm$90 BC (Layer 1)\\
983       &0.83m      &0.73m      &0.25m   &Stepped U &---      &$\times$18
  &  $\times$8      & ---& Fired clay&---\\
&&&&&&&&&&\\
\multicolumn{10}{|l|}{\bf Beaker}&\\
552       &---        &0.68m      &0.12m   &Saucer    &P7--14   &---
  & ---       & ---       &---        &---\\
790       &---        &0.60m      &0.25m   &U         &P15      &$\times$12
  & ---       & Quartzite-lump&---    &---\\
794       &2.89m      &0.75m      &0.25m   &Irreg.    &P16      &$\times$3
  & ---       & ---       &---        &---\\
\hline
\end{tabular}

\caption[Grooved Ware and Beaker Features, their Finds and
Radiocarbon Dates]{Grooved Ware and Beaker Features, their
Finds and Radiocarbon Dates; For a breakdown of the Pottery
Assemblages see Tables I and III; for
the Flints see Tables II and IV; for the
Animal Bones see Table V.}\label{rotfloat2}
\end{sidewaystable}
```

Figure 3 `\begin{table}`

```
\centering
\rotcaption{Minimum number of individuals; effect of rotating table
and caption separately}\label{rotfloat3}%
\begin{sideways}
\begin{tabular}[b]{ccccccccp{1cm}}
\hline
Phase&Total&Cattle&Sheep&Pig&Red Deer&Horse&Dog&Goat&Other\\
\hline
&1121&54&12&32&1&1&1&1&1 polecat\\
3&8255&58&6&35&1&1&1&1&1 roe deer, 1 hare, 1 cat, 1 otter\\
4&543&45&6&45&4&1&1&---&---\\
\hline
&9919&157&24&112&6&3&3&2&5\\
\hline
\end{tabular}
\end{sideways}
\end{table}
```

Figure 6 `\begin{sidewaysfigure}`

```
  \centerline{\psfig{figure=cat.eps,width=.8\textheight,height=.4\textwidth}}
\caption{A pathetically squashed rotated pussycat}\label{rotfloat4}
\end{sidewaysfigure}
```

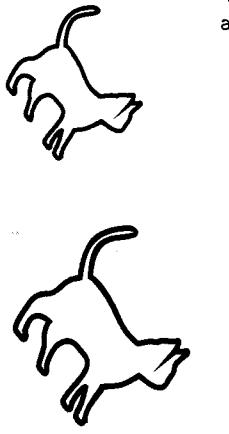


```
---\begin{turn}{-156}
\psfig{figure=cat.eps,width=1in}
\end{turn}---
```

```
---\begin{sideways}
\psfig{figure=cat.eps,width=1in}
\end{sideways}---
```

```
---\psfig{figure=cat.eps,width=1in}---
```

Figure 4: Turned, normal, and sideways, pictures within a figure



```
\psfig{figure=cat.eps,width=1in,%
angle=-56}
```

```
\psscalefirst
\psfig{figure=cat.eps,width=1in,%
angle=-56}
```

Figure 5: Figures rotated with 'psfig'

Table 1: This is a narrow table, which should be centred vertically on the final page.

a	b
c	d
e	f
g	h
i	j

Context	Length	Breadth/ Diameter	Depth	Profile	Pottery	Flint	Animal Bones	Stone	Other	C14 Dates
Grooved Ware										
784	—	0.9m	0.18m	Sloping U	P1	×46	×8		×2 bone	2150± 100 BC
785	—	1.00m	0.12	Sloping U	P2–4	×23	×21	Hammerstone	—	—
962	—	1.37m	0.20m	Sloping U	P5–6	×48	×57*	—	—	1990 ± 80 BC (Layer 4) 1870 ±90 BC (Layer 1)
983	0.83m	0.73m	0.25m	Stepped U	—	×18	×8	—	Fired clay	—
Beaker										
552	—	0.68m	0.12m	Saucer	P7–14	—	—	—	—	—
790	—	0.60m	0.25m	U	P15	×12	—	Quartzite-lump	—	—
794	2.89m	0.75m	0.25m	Irreg.	P16	×3	—	—	—	—

Table 2: Grooved Ware and Beaker Features, their Finds and Radiocarbon Dates; For a breakdown of the Pottery Assemblages see Tables I and III; for the Flints see Tables II and IV; for the Animal Bones see Table V.

Table 3: Minimum number of individuals; effect of rotating table and caption separately

Phase	Total	Cattle	Sheep	Pig	Red Deer	Horse	Dog	Goat	Other
	1121	54	12	32	1	1	1	1	1 polecat
3	8255	58	6	35	1	1	1	1	1 roe deer, 1 hare, 1 cat, 1 otter
4	543	45	6	45	4	1	1	—	—
	9919	157	24	112	6	3	3	2	5

Figure 6: A pathetically squashed rotated pussycat

◇ Sebastian Rahtz
ArchaeoInformatica
12 Cygnet Street
York Y02 1AG
spqr@uk.ac.york.minster

◇ Leonor Barroca
Department of Computer Science
University of York
Heslington
York Y01 5DD
lmb@uk.ac.york.minster