Producing graphs with MetaPost

multiple aligned graphs and error bars

Keywords
MetaPost, graphs, error-bars

Abstract
MetaPost is an interesting companion for generating figures for documents written in TEX or one of its derivatives. This article focuses on generating graphs in MetaPost, and more specifically on two problems one can encounter when creating graphs: multi part graphs and error bars.

Introduction

In Maps 29 Karel Wesseling described how several MetaPost graphs can be aligned relative to each other, when including them in a `\startcombination[1*2]` command in ConTEXt. Here I describe a different approach to the same problem: aligning multiple graphs in a single figure.

The method proposed in that article required a solid, *near white* background to be printed behind the graphs. The lightness of this background ensured that it appeared white when printed, but somehow it still feels as a 'dirty' trick. Having the graph and the axes in a fixed location to the bounding box of the figure can be highly desirable for some effects though. One such situation occurs when one wants to align all graphs with the vertical axis on the left margin throughout the document (so that the axis labels stick out in the left margin).

When that is not required, a different solution becomes possible. Note that I use LaTEX instead of ConTEXt, but the same technique should work in other TEX and MetaPost combinations.

Another issue that is often encountered in generating (scientific) graphs is the inclusion of error bars in the graph. The second part of this note provides a possible solution for that.

Multiple graphs in a single figure

This is the annotated code to produce the graph shown in figure 1. The code generates a graph with multiple panes, in a single figure. The alignment is not accomplished in TEX but in MetaPost. We use the graph module written by John Hobby. This example uses LaTEX for its labels, and this temporary document requires some preamble material. This is also where you change the typefaces should you want to do so, and include other packages.

Since this example uses pdfLaTEX instead of TEX, the invocation of `mpost` is slightly more elaborate:

```
mpost -tex=latex file.mp
```

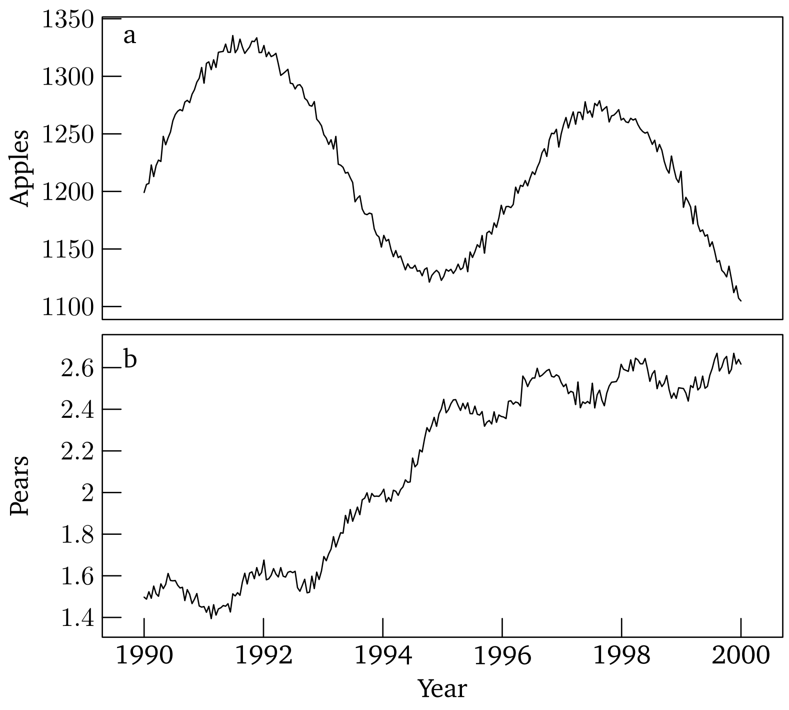

Figure 1. The sample figure with multiple graphs in a single figure. The MetaPost code is shown in the first listing. Note that this not really is about Apples and Pears, the data is just some goniometric functions with some added Gaussian noise.

```
mptopdf file.0
mv file-0.pdf file.pdf
```

This loads the graph package for MetaPost and sets up latex, so that it uses the correct fonts – here we show how to use times, but in the figures themselves, the Maps fonts have been used. After the fonts are changed, the numbers for the graph package have to be initialised again, which is done by the init_numbers function.

```
input graph;

verbatimtex
\documentclass[10pt]{article}
\usepackage{amsmath}
\usepackage{txfonts}
\begin{document}
etex

init_numbers(btex$-$etex, btex$1$etex, btex${\times}10$etex,
  btex${}^-$etex, btex${}^2$etex) ;
```

Start the figure itself. Some variables are declared here: a **pair** corner; to hold the corner of the final image for placing the axis labels, a few numerics to hold the size of the graphs, and an array **picture** thepart[]; to store the parts of the figure. In this example there are only two parts, more parts are left as an exercise for the reader.

```
beginfig(0);
  pair corner;
```

```
numeric width, height, gap; width := 90mm; height := 40mm; gap := 2mm;
picture thepart[];
```

Begin the first part of the figure, for the top of the final figure. A fairly simple graph, all auto scaled and without horizontal markers as they will appear on the bottom graph. More fancy examples for individual graphs can be found in the `mpgraph` manual. The label for this part of the figure is drawn outside the graph routines, so the physical units can be used for positioning the label in the top left corner of the graph.

```
draw begingraph(width,height);
  gdraw "Apples.dat";
  autogrid( ,itick.lft);
endgraph;

label.llft( btex a etex, (2mm, height−1mm) );
```

This is the crucial part: here the current graph is captured into the array, and the currentpicture is wiped to start with the other part of the graph.

```
thepart[1] := currentpicture;
currentpicture := nullpicture;
```

Here we start all over again, drawing the second graph, and storing it in the array. The labels at the bottom are done manually to make sure the full year is used – and not rounded to three digits, as MetaPost might do.

```
draw begingraph(width,height);
  gdraw "Pears.dat";
  autogrid( ,itick.lft);
  itick.bot( btex 1990 etex, 1990); itick.bot( btex 1992 etex, 1992);
  itick.bot( btex 1994 etex, 1994); itick.bot( btex 1996 etex, 1996);
  itick.bot( btex 1998 etex, 1998); itick.bot( btex 2000 etex, 2000);
endgraph;
label.llft( btex b etex, (2mm, height−1mm) );

thepart[2] := currentpicture;
currentpicture := nullpicture;
```

Assemble the graph: draw both parts, with the required gap in between. In this case the last wipe is not needed, as the bottom part will end up in the same location. I would still advise to follow this method, as it avoids many bugs and frustration when you decide later on to add parts.

```
draw thepart[1] shifted (0,height+gap);
draw thepart[2] shifted (0,0);
```

Figure out what the overall lower left corner of the current canvas is.

```
corner := llcorner currentpicture;
```

Draw the axis labels. This is not done using the standard MetaPost graph methods, as they *include* the axis labels in determining the width of the graph. This has the effect of drawing the label off centre. Since we already know the length of the axis, determining the centre is easy. That leaves the offset, which can be determined from the lower left corner of the assembled graph. Note that the labels for the vertical axes are at `height`/2 and 3`height`/2 + `gap`.

The 'Apples' and 'Pears' are smashed, to make sure the baselines are at the same distance from the axis. Yes, I'm aware that this can be done more easily in MetaFun, but I've never managed to get it to work outside of ConTEXt.

```
  label.lft( btex $\smash[b]{\text{Pears}}$ etex rotated 90,
    (xpart corner, height/2) );
  label.lft( btex $\smash[b]{\text{Apples}}$ etex rotated 90,
    (xpart corner, gap + 3*height/2) );
  label.bot( btex Year etex,
    (width/2, ypart corner) );
endfig;
```

Close the LaTEX document used for typesetting the labels.

```
verbatimtex
\end{document}
etex
end
```

I'm sure there are a dozen more solutions, and one may be more suitable than the other, depending on one's needs. This one worked rather well for me.

Error bars in MetaPost graphs

This second part of this note shows how to add error bars to a graph. Many graphs in scientific papers require the use of error indicators. This is the annotated code to produce the graph shown in figure 2. Again, we use the graph module written by John Hobby. Just like the previous example, this one uses LaTEX for its labels.

```
input graph;

verbatimtex
\documentclass[10pt]{article}
\usepackage{amsmath}
\usepackage{txfonts}
\begin{document}
etex

init_numbers(btex$-$etex, btex$1$etex, btex${\times}10$etex,
  btex${}^-$etex, btex${}^2$etex) ;
```

Define a few new macros for producing the error bars themselves. There are three versions: one with vertical error bars (errorbary), one with horizontal error bars (errorbarx) and one with indicators on both directions (errorbarxy). All three work in much the same fashion: define an empty **path** p;, and add a line from $(x, y+\Delta y)$ to $(x, y-\Delta y)$ to this path and draw it. Add the end caps, which are just pictures, and a marker to the data point itself. At the end of the routine, the path is declared

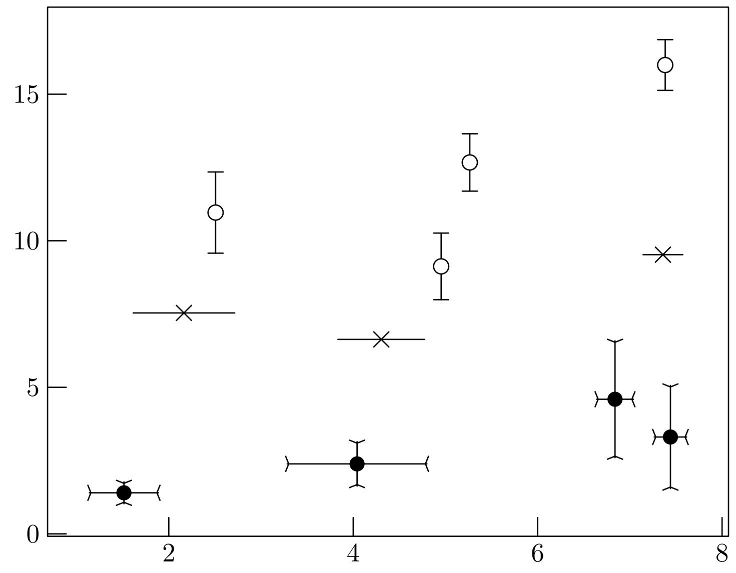

Figure 2. A sample graph with error bars. The MetaPost code is shown in the second listing.

again, which empties the path for the next point. The endcap pictures are rotated, so only a horizontal line is needed in most cases.

The data files are just ASCII text files, with the columns x, y, and Δy in the version with vertical error indicators. The columns are white space separated. The version with horizontal indicators, the columns are x, y, Δx. The final version with indicators on both axes, the order is a bit different: x, Δx, y, and Δy.

```
def errorbary(expr file,marker,endcap) =
  path p;
  gdata( file, s,
   augment.p(scantokens(s1), scantokens(s2)−scantokens(s3));
   augment.p(scantokens(s1), scantokens(s2)+scantokens(s3));
   glabel(endcap, scantokens(s1), scantokens(s2)−scantokens(s3) ) ;
   glabel(endcap rotated 180, scantokens(s1), scantokens(s2)+scantokens(s3));
   gdraw p;
   path p;
   glabel(marker, s1, s2 ) ;
  )
enddef;

def errorbarx(expr file,marker,endcap) =
  path p;
  gdata( file, s,
   augment.p(scantokens(s1)−scantokens(s3), scantokens(s2));
   augment.p(scantokens(s1)+scantokens(s3), scantokens(s2));
   glabel(endcap rotated 270, scantokens(s1)−scantokens(s3), scantokens(s2));
   glabel(endcap rotated 90, scantokens(s1)+scantokens(s3), scantokens(s2));
   gdraw p;
   path p;
   glabel(marker, s1, s2 ) ;
  )
enddef;
```

```
def errorbarxy(expr file,marker,endcap) =
  path p;
  gdata( file, s,
   augment.p(scantokens(s1)−scantokens(s2), scantokens(s3));
   augment.p(scantokens(s1)+scantokens(s2), scantokens(s3));
   glabel(endcap rotated 270, scantokens(s1)−scantokens(s2), scantokens(s3));
   glabel(endcap rotated 90, scantokens(s1)+scantokens(s2), scantokens(s3));
   gdraw p;
   path p;
   augment.p(scantokens(s1), scantokens(s3)−scantokens(s4));
   augment.p(scantokens(s1), scantokens(s3)+scantokens(s4));
   glabel(endcap, scantokens(s1), scantokens(s3)−scantokens(s4) ) ;
   glabel(endcap rotated 180, scantokens(s1), scantokens(s3)+scantokens(s4));
   gdraw p;
   path p;
   glabel(marker, s1, s3 ) ;
  )
enddef;
```

Here we build the graph. We declare some placeholders for the end caps and the marker symbols.

```
beginfig(0);
  numeric width, height;
  width := 90mm; height := 70mm;
  picture marker[], endcap[] ;
  numeric Symbolsize; Symbolsize := 2mm;
```

Create the end caps. Two versions are shown here: a simple line, and a wedge like shape.

```
  draw ((−0.5,0) −− (0.5,0)) scaled Symbolsize ;
  endcap[1] := currentpicture ;
  currentpicture := nullpicture ;

  draw ((−0.5,−0.2) −− origin −− (0.5,−0.2)) scaled Symbolsize ;
  endcap[2] := currentpicture ;
  currentpicture := nullpicture ;
```

Here we create three markers for the three different data sets: an open circle, a closed circle and a diagonal cross.

```
  fill fullcircle scaled Symbolsize withcolor white ;
  draw fullcircle scaled Symbolsize ;
  marker[1] := currentpicture ;
  currentpicture := nullpicture;

  fill fullcircle scaled Symbolsize ;
  marker[2] := currentpicture ;
  currentpicture := nullpicture;

  draw ((−0.5,−0.5) −− (0.5,0.5)) scaled Symbolsize ;
  draw ((0.5,−0.5) −− (−0.5,0.5)) scaled Symbolsize ;
  marker[3] := currentpicture ;
  currentpicture := nullpicture;
```

Finally: the graph itself. All three modes are shown, with different end caps and markers.

```
  draw begingraph(width,height);
    errorbary("oranges.dat", marker[1], endcap[1])
    errorbarx("oranges.dat", marker[3], nullpicture)
    errorbarxy("oranges.dat", marker[2], endcap[2])

    autogrid(itick.bot,itick.lft);
  endgraph ;
endfig ;
```

Close the LaTeX document used for typesetting the labels.

```
verbatimtex
\end{document}
etex
end
```

(Meta) Post Script

In the article that started this all, Karel Wesseling complains about the manual of the graph package. I can see his point, I'm not a fan of the documentation of metapost in general. There are other manuals, including a nice one written by André Heck: "Learning MetaPost by Doing", a link is provided below. Another manual that contains a lot of general MetaPost advice is the MetaFun manual – although some parts are obviously beyond plain MetaPost.

`http://remote.science.uva.nl/~heck/Courses/mptut.pdf`

Maarten Sneep
Atoom & Laserfysica
Vrije Universiteit
Amsterdam
sneep@nat.vu.nl