Making Publication Quality Plots and Graphics for LaTeX

Sigvald Marholm

22.11.19

0 Introduction

It is not uncommon to see technical reports and scientific articles with figures of different style than the rest of the document. The text in the figures may be too small, the font may be wrong, the lines in plots so thin it is barely visible, or pictures may appear pixelated when printed. In this document we explain some useful techniques for making high quality graphics, in particular for LaTeX-users, although the ideas are universal. Although there is much to be said about the artistic aspects of figure making, for instance on color palettes and proportions, we shall focus on the more technical aspects.

There is actually a very simple recipe to consistent, high quality figures:

1. Obtain the correct figure size and font from the document where the figure is to be inserted.

2. Create the figure using the right size and font from the start. For consistency between figures you may use the same line width, color palettes, etc. in all your figures.

3. Imort the image into the document without rescaling it.

The reader familiar with vector graphics should note that vector graphics is no replacement for the above recipe. Although vector graphics can be scaled to fit inside any document without loss of quality, this scaling also affects the text such that it quickly becomes too small or too large. You would also not get consistent line widths, arrow sizes, etc., between your figures should you attempt that, because they may be scaled by different factors in different figures.

In what follows, we go through each step in detail. In step 2 there is obviously endless possibilities in which software to use, and how to use them, and we do not cover it all. We do, however, provide some general information, as well as some relevant details on getting the right figure size in a few selected tools: InkScape, Gimp, and Matplotlib.

1 Obtain the Correct Size and Font from LaTeX

1.1 Figure Size

We shall start by determining the right size of the figure. For two-column articles one may for instance let the figure fill the entire column width, whereas for single column articles perhaps a lower fraction of the width is more suitable, unless the figure has a very wide aspect ratio. The author may of course

experiment with this, but in any case, it is a good start to know what the column width is.

The column width is not the same on all documents, but fortunately, it is rather easy to extract by inserting the following LaTeX command somewhere in your document:

```
\the\columnwidth
```

When the document is compiled, the width of the column will be written in place of this command, for instance "221.0pt" for a typical two-column document. In case you have a very wide figure and would like it to span both columns, you should instead insert

```
\the\textwidth
```

The text width accounts for both columns, as well as the spacing in between. For single column documents the text width and the column width are usually the same.

Although many programs allow you to specify the size of a figure directly in typographic points (pt), one should be aware that slightly different definitions of the pt exists. This usually isn't noticable when specifying the text size, which is usually around 10pt. However, if a figure ends up slightly wider than a column, you will get an "overfull hbox" warning by LaTeX. You may therefore wish to convert the column width to for instance millimeters or inches before creating the figure. In LaTeX, the conversion is given by

$$1\,\mathrm{pt} = 0.351\,\mathrm{mm} = 0.0138\,\mathrm{in}. \tag{1}$$

A column width of for instance 221 pt thus converts to 77.6 mm. It pays off to round downwards, for instance to 77 mm, in order to avoid an "overfull hbox". The height of the figure is simply chosen to get an aspect ratio deemed appropriate for the figure.

1.2 Font

The next thing we need is to determine the font family and text size. This can be found by inserting the following into your document:

```
\makeatletter
\f@family~\f@size pt
\makeatother
```

The macro `\f@family` contains the font name and the macro `\f@size` the text size. The @ symbol is frequently used in macro names internal to LaTeX packages to prevent them from being overwritten by the end user. The reason this works is because @ behaves as a normal letter inside packages and that it can therefore be used in macro names. During normal usage, however, @ is a special symbol. Nevertheless, we can temporarily turn @ into a normal letter with `\makeatletter` to get access to these macros, and then turn it back to a symbol again using `\makeatother`. This is what we have done in the above code. To make it a bit more convenient, the following code can be put in the preamble of your document:

```
\makeatletter
\newcommand{\getfont}{\f@family~\f@size pt}
\makeatother
```

Now you can just insert `\getfont` where you want to determine the font.

The font families in LaTeX often do not have nice names such as "Times New Roman" but may instead come out as something like "cmr". "cmr" is short for Computer Modern, the standard font in LaTeX. Unfortunately the font format used in LaTeX is a different kind than the TrueType or OpenType formats understood by most programs, and it may therefore be difficult to find the exact same font for use in your graphics software. Computer Modern, however, often have good replica available for download on the internet. If you are unable to find the exact same font, you may instead use something similar. Computer Modern or variants of Times or Times New Roman are often used fonts in scientific journals.

One thing in particular to pay attention to is whether or not the fonts have *serifs*, i.e., tiny feet on letters such as "i" and "f":

Fonts may be classified into serif fonts or sans (without) serif fonts according to this feature. It is interesting that the serifs almost touches one another at the base of each line, thereby creating the illusion of a continuous line on which the letters sit. There is also the illusion of a line on top of the "low" letters such as "e" and "r". These lines are believed to help guide the eye along the text, preventing you from losing track of which line you're currently reading. Serif fonts are therefore the norm when it comes to printed books and articles, perhaps except for headings, or where there are only small fragments of text. Sans serif fonts, on the other hand, can be used with lower resolution, and are predominant on web pages, and other digital

resources. Consequently, plotting tools such as Matplotlib uses a sans serif font by default, whereas scientific articles are more likely to use a Serif font. Choosing any serif font in your plots goes a long way for a more consistent look.

You should also be aware that the text size (or sometimes even the font) may be different on different parts of your document. The footnotes, for instance, are typically smaller. The figure captions are also sometimes a bit smaller than the body text. If that is the case, it would make sense for the text inside your figures to match the size of the figure captions rather than the body text. You may also deliberately choose a slightly smaller text size inside your figures as part of your document's style, e.g. 9pt instead of 11pt. This is fine as long as you do it consistently throughout your document.

2 Create the Figure

Let us first discuss what is common regardless of what software you choose for creating your figures. First you need to choose a suitable image format with respect to both quality and size. Some journals have a file size limit on the submitted documents, but if you habitually use the correct format ind image size this should not pose any problem. We present a summary of formats below:

BMP This is one of the simplest formats, where an image is represented as a grid of pixels (a *raster*) with each pixel stored as a fixed number of bytes. This leads to comparatively large files. Use PNG or JPG instead.

PNG Similarly to BMP, this is also a raster format, but contrary to BMP, it uses a *lossless* compression scheme, meaning that it takes less space with no loss of quality. Use compression level 9 in your graphics program for the smallest files. The compression used in PNG is efficient for images where large areas have the same color, typically graphical illustrations, schematics and plots. The compression is not as efficient for photographs, although it do lead to significantly smaller files than BMP.

JPG Yet another raster format, but this one with a *lossy* compression, leading to a quality reduction. For JPG you can tune a quality factor between 0 and 100, where a higher number means better quality, but larger files. JPG files are very well suited for photographs, and even a quality factor of 90 or 95 will produce files much smaller than PNG

files at still an excellent quality. The JPG compression do not handle sharp transition between colors very well, however, leading to "foggy" artifacts near sharp edges. This makes it particularly unsuited for graphical illustrations, schematics and plots.

SVG Contrary to the other formats, SVG is a *vector* format, meaning that it consits of geometric entities such as lines and curves being described by their end points, curvature, thickness, etc. Whereas the pixels would become visible if you scale up a raster format, the vector formats can be scaled up indefinitely without getting pixelated. For this reason, vector graphics is excellent for graphical illustrations and schematics, but the figures must be created using a dedicated programs (such as InkScape) from the very start. You cannot convert a raster image into a vector format. Although it is possible to store plots in vector formats, this means storing every single line segment in the plot. This sometimes causes unnecessarily large files, and for this reason it is often easier to use PNG files for this. Unfortunately, LaTeX cannot import SVG files, so they must be converted to PDF files prior to being used in LaTeX. It is nevertheless a good idea to also keep an SVG file, since this gives you full flexibility if you later need to change the figure.

PDF In addition to full fledged documents, PDF can also be used to store vector graphics, as described above.

EPS If you compile your documents using the command `latex`, none of the above formats can be used, and you must instead resort to EPS. EPS can also store raster and vector graphics, but we recommend rather compiling your documents with `pdflatex` if possible.

Images formats can be classified into *bitmap/raster* or *vector* graphics. Bitmap images consists of a grid of pixels of different colors. If they are scaled up too much, or are of too low resolution, they get pixelated. Vector graphics, on the other hand, consists of geometric entities such as lines, circular arcs, bezier curves, etc., and can be scaled up indefinitely.

Regardless of which software is used to create the figures, many principles are the same. The principles are largely the same regardless of what software you use to create your figures; you must specify the right size of the images from the start, use a suitable file format, and a sufficient resolution.

2.1 Image Formats and Resolution

2.2 InkScape

2.3 Gimp

2.4 Matplotlib

3 Inserting the figure

Finally comes the point of inserting the figure into your LaTeX document, which is done in the simplest way possible:

```
\includegraphics{file.png}
```

(given that the file is named `file.png`). Note that we do not use any options for scaling the figure, like `[width=0.7\columnwidth]` or `[scale=0.23]`. The default is already a scaling factor of one, and the figure should already have the right size.

That's all, good luck!