The auto-pst-pdf package

Will Robertson & Johannes Große
wspr 81 at gmail dot com

2007/11/18 v0.4

1 Basic usage

This package provides a wrapper around pst-pdf to automatically accomodate for typesetting either with DVI or PDF output. With default package option `[on]`, typesetting under pdfLaTeX will automatically initiate an auxiliary compilation of LaTeX→ `dvips` → `ps2pdf` → `pdfcrop` to generate the required PDF figures for the document.

After this has been done and the figures no longer need to be re-generated, the package can be given the `[off]` option to save compilation time:

```
\usepackage[off]{auto-pst-pdf}
```

If the extension of your LaTeX document is not `.tex`, then it must be declared when the package is loaded (*e.g.*, like many others, I like to use `.ltx` to distinguish between Plain TeX and LaTeX files):

```
\usepackage[ext=ltx]{auto-pst-pdf}
```

2 Requirements

pdfTeX must be called with the `-shell-escape` option. Requires the following packages: ifplatform, pst-pdf, xkeyval.

Heiko Oberdiek's `pdfcrop` Perl script[1] must be installed for the default `crop=on` option (see section 4). Under Windows, a Perl installation[2] will also need to be installed even though `pdfcrop` itself is part of MiKTeX.

[1] `http://www.ctan.org/tex-archive/support/pdfcrop/`
[2] Freely available: `http://www.activestate.com/Products/activeperl/index.plex`

3 Provided macros for including graphics

Macros are provided to easily facilitate figures created by the MATLAB package laprint[3] and the Mathematica package MathPSfrag[4]. Also, a generic psfrag[5] wrapper is provided.

`\mathfig{`⟨*filename*⟩`}`	insert a Mathematica graphic from MathPSfrag (without `-psfrag` suffix)
`\matlabfig{`⟨*filename*⟩`}`	insert a MATLAB graphic from laprint
`\psfragfig{`⟨*filename*⟩`}`	insert an EPS with psfrag

For the `\psfragfig` command, psfrag statements are input from either or both of the files ⟨*document*⟩`-psfrag.tex` and ⟨*filename*⟩`-psfrag.tex` if they exist.

The above commands all accept an optional argument which is passed to the underlying `\includegraphics` macro.

4 Advanced package options

Better results are obtained by using `pdfcrop` in the auxiliary compilation process, and this is used by default. It is not installed by default, however, and will not always be required. Cropping can be controlled with the `crop` option:

```
\usepackage[crop=off]{auto-pst-pdf}
```

The package automatically deletes the files generated during the auxiliary LaTeX compilation. Which files are deleted are chosen by passing a list of file extensions to the `cleanup` option (no error message or warning is produced if a file is specified that does not exist). The default list is:

```
\usepackage[cleanup={log,aux,dvi,ps,pdf}]{auto-pst-pdf}
```

The options passed individually to `latex`, `dvips`, `ps2pdf`, and `pdfcrop` in the auxiliary compilation process may all be customised, if you know what you're doing. The defaults for the latter three are

```
\usepackage[dvips={-o -Ppdf},
            pspdf={-dAutoRotatePages=/None}},
            pdfcrop={}]{auto-pst-pdf}
```

The LaTeX auxiliary compilation has some hard-coded options (see the source if you're interested), and further options can be appended if you wish. For example, to run the auxiliary compilation with more information written to the console, use the following package option:

```
\usepackage[latex={-interaction=nonstopmode}]{auto-pst-pdf}
```

[3] `http://www.uni-kassel.de/fb16/rat/matlab/laprint/`
[4] `http://wwwth.mppmu.mpg.de/members/jgrosse/mathpsfrag/`
[5] `http://www.ctan.org/tex-archive/help/Catalogue/entries/psfrag.html`

5 Customisation

If you want to customise the auxiliary process, this package provides the `delay` package option that does not execute the pst-pdf processing. Then the `\app@compile` macro can be re-defined to do whatever you like and the macro `\CompilePics` will invoke the auxiliary process.

File I
auto-pst-pdf implementation

6 Setup code

This is the package.

```
\ProvidesPackage{auto-pst-pdf}[2007/11/18 v0.4 Wrapper for pst-pdf]
```

Change History

Required packages pst-pdf is loaded later on.

```
\RequirePackage{ifpdf,xkeyval,ifplatform}
```

Things we need

```
\newif\if@app@off@
\newif\if@app@delay@
\newif\if@app@crop@
\def\app@suffix{autopp}
\edef\app@jobname{\jobname-\app@suffix}
\edef\app@pics{\jobname-pics.pdf}
```

Option processing

```
\DeclareOptionX{off}[]{\@app@off@true}
\define@choicekey{auto-pst-pdf.sty}{crop}[\@tempa\@tempb]{on,off}{%
  \ifcase\@tempb\relax
    \@app@crop@true
  \or
    \@app@crop@false
  \fi}
\DeclareOptionX{on}[]{\@app@off@false}
\DeclareOptionX{ext}{\def\app@ext{#1}}
\DeclareOptionX{delay}{\@app@delay@true}
\DeclareOptionX{latex}{%
  \def\app@latex@opts{%
```

```
    \ifwindows
      -disable-write18
    \else
      -no-shell-escape
    \fi
    -jobname="\app@jobname"
    -interaction=batchmode
    #1}}
\DeclareOptionX{dvips}{\def\app@dvips@opts{#1}}
\DeclareOptionX{pspdf}{\def\app@pspdf@opts{#1}}
\DeclareOptionX{pdfcrop}{\def\app@pdfcrop@opts{#1}}
\DeclareOptionX{cleanup}{%
  \let\app@rm@files\@empty
  \@for\@ii:=#1\do{%
    \edef\app@rm@files{\app@rm@files,\app@jobname.\@ii}}}
\ExecuteOptionsX{%
    ext=tex,
    crop=on,
    latex={},
    dvips={-Ppdf},
    pspdf={},
    pdfcrop={},
    cleanup={log,aux,dvi,ps,pdf}}
\ifwindows
  \ExecuteOptionsX{pspdf={}}
\else
  \ExecuteOptionsX{pspdf={-dAutoRotatePages=/None}}
\fi
\ProcessOptionsX
```

Shorthands

```
\def\app@exe{\immediate\write18}
\def\app@nl{^^J\space\space\space\space}
\newcommand\app@PackageError[2]{%
  \PackageError{auto-pst-pdf}{\app@nl #1^^J}{#2}}
\newcommand\app@PackageWarning[1]{%
  \PackageWarning{auto-pst-pdf}{\app@nl #1^^JThis warning occured}}
\newcommand\app@PackageInfo[1]{\PackageInfo{auto-pst-pdf}{#1}}
```

These are cute:

```
\newcommand\OnlyIfFileExists[2]{\IfFileExists{#1}{#2}{}}
\newcommand\NotIfFileExists[2]{\IfFileExists{#1}{}{#2}}
```

`\app@convert` #1 : command name
#2 : source file

`#3` : destination file

Check if the source file exists and calls the command to generate the destination file. If the final file is not created, generate an error.

```
\def\app@convert#1#2#3{%
  \OnlyIfFileExists{#2}{%
    \app@exe{\csname app@cmd@#1\endcsname{#2}{#3}}%
    \NotIfFileExists{#3}{\app@PackageError{Creation of #3 failed.}}}}
```

`\app@compile` First we define the entire `latex` → `dvips` → `ps2pdf` (→ `pdfcrop`) command sequence. The actual call to the compilation macro follows thereafter. This macro contains the actual creation of the pdf container. Each processing step is in a separate macro to allow simple modification.

```
\def\app@compile{%
  \app@cleanup
  \app@remove@container
  \app@convert{latex}{\jobname.\app@ext}{\app@jobname.dvi}%
  \app@convert{dvips}{\app@jobname.dvi}{\app@jobname.ps}%
  \if@app@crop@
    \app@convert{pstopdf}{\app@jobname.ps}{\app@jobname.pdf}%
    \app@convert{pdfcrop}{\app@jobname.pdf}{\app@pics}%
  \else
    \app@convert{pstopdf}{\app@jobname.ps}{\app@pics}%
  \fi
  \IfFileExists{\app@pics}
    {\app@cleanup}
    {\app@PackageWarning{Could not create \app@pics.
      Auxiliary files not deleted.}}}
```

Command-line program to delete files:

```
\edef\app@rm{\ifwindows del \else rm -- \fi}
```

`\app@try@rm` Macro to delete files (comma-separated) if they exist:

```
\newcommand\app@try@rm[1]{%
  \@for\@tempa:=#1\do{%
    \OnlyIfFileExists{\@tempa}{\app@exe{\app@rm "\@tempa"}}}}
```

Remove pdf picture container:

```
\def\app@remove@container{\app@try@rm{\app@pics}}
```

Clean up auxiliary files: (`\app@rm@files` defined by the `cleanup` package option)

```
\def\app@cleanup{\app@try@rm{\app@rm@files}}
```

LaTeX:

```
\def\app@cmd@latex#1#2{latex \app@latex@opts\space
  "\let\noexpand\APPmakepictures\noexpand\empty\noexpand\input #1"}
```

dvips:

```
\def\app@cmd@dvips#1#2{dvips \app@dvips@opts\space -o "#2" "#1"}
```

ps2pdf:

```
\def\app@cmd@pstopdf#1#2{ps2pdf \app@pspdf@opts\space "#1" "#2"}
```

pdfcrop:

```
\def\app@cmd@pdfcrop#1#2{pdfcrop \app@pdfcrop@opts\space "#1" "#2"}
```

6.1 Base functionality

For compilation, we use the [notightpage] option of pst-pdf and the pdfcrop Perl script because Matlab-created EPS figures (can) have elements that extend ouside their bounding boxes, and end up with clipped content after ps2pdf. Otherwise the script ps4pdf would be sufficient.

pdfLaTeX compilation Requires supplementary processing with pst-pdf:

```
\ifpdf
  \if@app@off@\else
    \newcommand\CompilePics{%
      \ifshellescape
        \app@exe{echo "  "}
        \app@exe{echo "-----------------------------------------------"}
        \app@exe{echo "auto-pst-pdf: Auxiliary LaTeX compilation"}
        \app@exe{echo "-----------------------------------------------"}
        \app@compile
        \app@exe{echo "-----------------------------------------------"}
        \app@exe{echo "auto-pst-pdf: End auxiliary LaTeX compilation"}
        \app@exe{echo "-----------------------------------------------"}
      \else
        \app@PackageError{%
          "shell escape" (or "write18") is not enabled:\app@nl
          auto-pst-pdf will not work!}
         {You need to run LaTeX with the equivalent of
          "pdflatex -shell-escape"\app@nl
          Or turn off auto-pst-pdf.}
      \fi}
```

Execute the compilation process unless the user is daring and wishes to do so themselves:

```
    \if@app@delay@\else\CompilePics\fi
  \fi
  \if@app@crop@
    \PassOptionsToPackage{notightpage}{pst-pdf}
  \fi
```

LaTeX compilation Either we're calling `latex` from within a pdfLaTeX run (see above) or the document is being compiled as usual.

```
\else
```

LaTeX compilation induced by this package:

```
  \ifdefined\APPmakepictures
    \if@app@crop@
      \PassOptionsToPackage{notightpage}{pst-pdf}
    \fi
```

LaTeX compilation from scratch (as in '`latex` ⟨*document*⟩`.tex`') — here the `postscript` environment does nothing and document is processed 'normally':

```
  \else
    \PassOptionsToPackage{inactive}{pst-pdf}
  \fi
```

Bug fix for `pst-pdf` (apparently):

```
  \AtBeginDocument{%
    \gdef\ppf@other@extensions{.png,.pdf,.jpg,.jpeg,.PNG,.PDF,.JPG,.JPEG}
    \edef\Gin@extensions{\Gin@extensions,.mps}
    \DeclareGraphicsRule{.mps}{eps}{*}{}}
\fi
```

After the requisite package options have been declared depending on the execution mode, it's now time to load the package:

```
\RequirePackage{pst-pdf}
```

6.2 Extras for external packages

Commands are provided that mirror `\includegraphics` (and similarly accept an optional argument) for the output of different psfrag-related packages. This provides a consistent and easy way to include such figures in the document.

Please suggest wrappers for other packages that output psfrag (for example: SciLab, R, Maple, LabView, ... ?)

`\matlabfig` We need to disable the scaling that laprint applies to `\includegraphics` in here, because otherwise labels that extend outside the bounding box of the generated PostScript file will change the intended width of the graphic.

```
\let\app@ig\includegraphics
\newcommand\matlabfig[2][]{%
  \begin{postscript}
    \renewcommand\includegraphics[2][]{\app@ig[#1]{##2}}
    \input{#2}
  \end{postscript}}
```

`\mathfig` For Mathematica's MathPSfrag output

```
\newcommand\mathfig[2][]{%
  \begin{postscript}
    \input{#2-psfrag}%
    \includegraphics[#1]{#2-psfrag}%
  \end{postscript}}
```

`\psfragfig` EPS graphics via psfrag. Include your psfrag commands in the files ⟨*document*⟩-psfrag.tex and/or ⟨*figname*⟩-psfrag.tex, where ⟨*document*⟩ is the filename of the main document and ⟨*figname*⟩ is the filename of the graphics inserted.

```
\newcommand\psfragfig[2][]{%
  \begin{postscript}
    \InputIfFileExists{#2-psfrag}{}{}%
    \includegraphics[#1]{#2}%
  \end{postscript}}
```

Finally, input any psfrag commands associated with the document: (actually, does this work?!?)

```
\InputIfFileExists{\jobname-psfrag}{}{}
```