`gptex2eps`
Making nice output from Gnuplot via Latex

Håvard Berland[*]

November 2003

Abstract

This document describes how to use a bash script for automating the last part of generating postscript of plots from Gnuplot, via Latex in such a way that the fonts in your plots are similar to the one in your body text, and also having the possibility of more or less latex-expressions in your plots. The output may be converted to pdf if necessary.

Contents

1 Introduction

Figures in publications, theses, reports and alike with the same typographical quality as the rest of the document, often typeset by LaTeX, is not seen too often.

A natural prerequisite is to be able to use the same font for any text in the figures as you use for the rest of your document. Also, for many purposes, special symbols are needed which correspond to symbols in your

[*]`http://www.pvv.ntnu.no/~berland`

report. These should look exactly the same in both your figures and body text.

...

2 Usage

2.1 Converting to pdf

3 Modifying the LaTeX-preamble

4 How it works

5 Source code

The source code is available for download at the web-address

`http://www.math.ntnu.no/~berland/gptex2eps`,

and is also included here for reference.

```
#!/bin/bash
#
# Script for automating the generation of eps-pictures from plots in
# gnuplot, via latex to get axis annotations and legends in the
# correct font.
#
# Assumes a file containg gnuplot commands as the input file. It is
# recommended that '.gp' as the suffix of the gnuplotfile. If the file
# has the name 'alphafunction.gp', then it assumes that this gp-file
# produces 'alphafunction.tex', and thereafter this script produces
# 'alphafunction.eps'. Consult the documentation for further information.
#
# Håvard Berland http://www.math.ntnu.no/~berland
#
# $Id: gptex2eps,v 1.4 2007/09/25 11:56:37 berland Exp $
# $Source: /home/pvv/d/berland/etc/cvsrepo/gnuplottex2eps/prog/gptex2eps,v $
#

function usage
{
    echo ""
    echo "Usage:"
    echo " gptex2eps <options> <gpcommandfile.gp>"
    echo ""
    echo " Options:  -v    Verbose, and don't delete temp files"
    echo "           -pdf  Generate pdf in addition via epstopdf"
    echo ""
    echo "Script written by Håvard Berland http://www.math.ntnu.no/~berland"
    echo "Documentation available at http://www.math.ntnu.no/~berland/gptex2eps"
    exit
}

function die
{
    echo "$1"
    echo "Exiting ... "
    exit
}

# A prefix used for outputting messages to the user
errorprefix="␣(Error)␣"
infoprefix="␣(Info)␣"

# The file where the preamble is to be written and/or found
preamblefile=gptex2eps−preamble.tex

# Option for running latex .
latexoption="−interaction=batchmode"

# Option for dvips . These are set for tex− installations based
# on teTeX, and ensures that Type1 fonts are used ( via the file
# config .pdf loaded by −Ppdf), this gives correct fonts in the
# pdf−document. Set this variable to "" if this does not work,
# and try to find other ways of getting Type1 fonts .
dvipsoptions="−Ppdf␣−G0"

# Each of these commands must be available and in $PATH !
neededcommands="gnuplot␣latex␣dvips␣gs"

for cmd in $needecommands ; do
    which $cmd >/dev/null 2>&1 \
        || die "${errorprefix}Could␣not␣find␣$cmd␣in␣your␣path!"
done



if [ −z "$1" ]; then
    echo "${errorprefix}Mandatory␣input␣arguments␣not␣provided."
    usage
fi

# Default , throw away unwanted output from commands.
out=">/dev/null"
export out

# Default option to run dvips in quiet mode:
dvipsout="−q"

# If verbose , alter the above set variables .
if [ "$1" = "−v" ] ; then
    out=""
    dvipsout=""
    latexoptions=""
    shift
fi

# Check if user wants pdf (we might be called via wrapper script )
dopdf=""
if [ "$1" = "−pdf" ] ; then
    dopdf="yes"
    shift
fi

# In case of changed order of options ( this way of dealing
# with options is certainly not scalable !!!)
if [ "$1" = "−v" ] ; then ## Do SAME as above..
    out=""
    dvipsout=""
    latexoptions=""
    shift
fi



# Check our mandatory input argument
gpfile=$1 # may or may not include .gp as an ending

if [ ! -f $gpfile ] ; then
    # Test if user just dropped the ending:
    if [ -f "$gpfile.gp" ] ; then
        gpfile="$gpfile.gp"
    else
        die "${errorprefix}Could␣not␣find␣the␣gnuplot␣command-file␣$gpfile"
    fi
fi

# If no ending, there could be both a gpfile and gpfile.gp, typically
# with data in gpfile.
if [ -f $gpfile -a -f "$gpfile.gp" ] ; then
    gpfile="$gpfile.gp"
    echo "${infoprefix}␣Assuming␣$gpfile␣is␣the␣file␣with␣gnuplot-commands"
fi

# Check that the gpfile is not empty
if [ ! -s $gpfile ] ; then
    die "${errorprefix}The␣file␣$gpfile␣is␣empty."
fi

base="${gpfile%.gp}"
outfile=$base.eps

# The user is also allowed to provide an output file if really necessary:
if [ ! -z "$2" ]; then
        outfile=$2
fi


## Make a preamble file if it does not exist

if [ ! -s "$preamblefile" ] ; then
    echo "${infoprefix}Generating␣preamble␣file␣$preamblefile"
    echo "${infoprefix}You␣may␣edit␣to␣suit␣your␣needs␣if␣necessary␣and␣rerun"

    # CVS tip if the directory CVS exists.
    if [ -d "CVS" ]; then
        echo "${infoprefix}You␣might␣want␣to␣do␣a␣'cvs␣add␣$preamblefile'␣as␣well"
    fi
    touch $preamblefile \
        || die "${errorprefix}Could␣not␣write␣to␣$preamblefile,␣check␣your␣permissions"


    ## This could have been redone with a HERE document..
    echo "%" > $preamblefile
    echo "%␣This␣is␣a␣preamble␣file␣for␣'gptex2eps'.␣You␣may␣edit␣things" \
        >> $preamblefile
    echo "%␣here␣if␣necessary.␣Typically␣you␣might␣want␣to␣change␣the␣font" \
        >> $preamblefile
    echo "%␣size,␣the␣font␣or␣add␣some␣more␣packages␣for␣your␣latex␣commands" \
        >> $preamblefile
    echo "%␣in␣your␣figures.␣" >> $preamblefile
    echo "%" >> $preamblefile
    echo "%␣If␣you␣make␣errors␣in␣here,␣rerun␣gptex2eps␣with␣'−v'␣(verbose)␣and" \
        >> $preamblefile
    echo "%␣check␣the␣error␣messages␣from␣latex,␣and␣then␣fix␣here." >> $preamblefile
    echo "%␣You␣may␣also␣just␣delete␣this␣file␣if␣you␣are␣in␣trouble" >> $preamblefile
    echo "%␣and␣a␣new␣default␣one␣will␣be␣generated" >> $preamblefile
    echo "%" >> $preamblefile
    echo "\documentclass[12pt]{article}" >> $preamblefile
    echo "" >> $preamblefile
    echo "%␣Packages␣for␣most␣mathematical␣latex␣commands:" >> $preamblefile
    echo "\usepackage{amsfonts}" >> $preamblefile
    echo "\usepackage{amsmath}" >> $preamblefile
    echo "\usepackage{amssymb}" >> $preamblefile
    echo "" >> $preamblefile
    echo "\usepackage{ae}␣␣␣␣␣␣␣%␣This␣is␣in␣case␣you␣also␣want␣to␣make␣pdf␣afterwards" \
        >> $preamblefile
    echo "" >> $preamblefile
    echo "%␣You␣might␣want␣the␣palatino␣font␣instead,␣then␣uncomment␣the␣following" \
        >> $preamblefile
    echo "%␣two␣lines,␣and␣do␣not␣use␣the␣ae␣package␣above" >> $preamblefile
    echo "%\usepackage{palatino}" >> $preamblefile
    echo "%\usepackage{palatcm}␣%␣Palatino␣math␣fonts␣" >> $preamblefile
    echo "" >> $preamblefile
    echo "\usepackage[dvips]{color}" >> $preamblefile
fi ## Preamble done.


##############################
# Then run GNUPLOT on $gpfile

if [ ! −z "$out" ] ; then   # (Gnuplot is normally very quiet, so no difference her)
        # Not verbose:
        gnuplot $gpfile  || die "${errorprefix}Gnuplot␣failed"
else
        # verbose
        gnuplot $gpfile  || die "${errorprefix}Gnuplot␣failed"
fi

# Check that gnuplot has written to a file $base.tex
# and that $base.tex is newer than $gpfile ( == is also ok, so we check with −ot)
if [ "$base.tex" −ot "$gpfile" ] ; then
    echo "${errorprefix}No␣newly␣generated␣$base.tex␣found␣after␣running␣gnuplot."
    echo "${errorprefix}Maybe␣gnuplot␣failed␣or␣your␣gnuplot␣code␣did␣not␣contain"
    die "${errorprefix}␣␣set␣output␣'$base.tex'"
fi

cat $preamblefile > $base−gptex.tex

echo "\begin{document}" >> $base−gptex.tex
echo "\pagestyle{empty}" >> $base−gptex.tex
echo "\input{$base.tex}" >> $base−gptex.tex
echo "\end{document}" >> $base−gptex.tex

##########################
# Run latex

if [ ! −z "$out" ] ; then
    # Not verbose:
    latex $latexoptions $base−gptex.tex >/dev/null \
    || die "${errorprefix}Latex␣failed,␣rerun␣with␣'−v'␣(verbose)"
else
    # Verbose:
    latex $base−gptex.tex || die "${errorprefix}Latex␣failed"
```

```
fi

dvips -E $dvipsoptions $base-gptex.dvi $dvipsout -o $outfile \
    || die "${errorprefix}dvips failed, rerun with '-v' (verbose)"


##########################################################
# Use 'gs' to determine bounding box.
# This is sometimes necessary, and does not hurt to do always.
# (credit: Per Kristian Hove)
(gs -sDEVICE=bbox -dBATCH -dNOPAUSE $outfile 2>&1 >/dev/null) > $outfile.bbox
(echo "/%%BoundingBox:"
 echo -en "d\ni\n"
 cat $outfile.bbox
 echo -en ".\nw\nq\n"
) | ed $outfile >/dev/null


# Make pdf if user wants to.
if [ ! -z "$dopdf" ]; then
    cmd="epstopdf"
    which $cmd >/dev/null 2>&1 \
        || die "${errorprefix}Could not find $cmd in your path, only eps generated!"
    $cmd $base.eps --outfile=$base.pdf
fi


# delete if not verbose
if [ ! -z "$out" ]; then
    rm -f $base-gptex.tex $base.tex $base-gptex.aux \
        $base-gptex.log $base-gptex.dvi $outfile.bbox
fi
```