pdf

From grDevices v3.6.2
by R-core R-core@R-project.org

99.99th
Percentile

PDF Graphics Device

`pdf` starts the graphics device driver for producing PDF graphics.

Keywords device

Usage

```
pdf(file = if(onefile) "Rplots.pdf" else "Rplot%03d.pdf",
    width, height, onefile, family, title, fonts, version,
    paper, encoding, bg, fg, pointsize, pagecentre, colormodel,
    useDingbats, useKerning, fillOddEven, compress)
```

Arguments

file a character string giving the name of the file. If it is of the form `"|cmd"`, the

by `cmd`. If it is `NULL`, then no external

documentation

s created (effectively, no drawing occurs), but the device may still be queried (e.g., for size of text).

For use with `onefile = FALSE` give a C integer format such as `"Rplot%03d.pdf"` (the default in that case). (See `postscript` for further details.)

Tilde expansion (see `path.expand`) is done.

width, height the width and height of the graphics region in inches. The default values are `7`.

onefile logical: if true (the default) allow multiple figures in one file. If false, generate a file with name containing the page number for each page. Defaults to `TRUE`, and

	forced to true if `file` is a pipe.
family	the font family to be used, see `postscript`. Defaults to `"Helvetica"`.
title	title string to embed as the `/Title` field in the file. Defaults to `"R Graphics Output"`.
fonts	a character vector specifying R graphics font family names for additional fonts which will be included in the PDF file. Defaults to `NULL`.
version	a string describing the PDF version that will be required to view the output. This is a minimum, and will be increased (with a warning) if necessary. Defaults to `"1.4"`, but see ‘Details’.
paper	the target paper size. The choices are `"a4"`, `"letter"`, `"legal"` (or `"us"`) and `"executive"` (and these can be capitalized), or `"a4r"` and `"USr"` for rotated (‘landscape’). The default is `"special"`, which means that the `width` and `height` specify the paper size. A further choice is `"default"`; if this is selected, the papersize is taken from the option `"papersize"` if that is set and as `"a4"` if it is unset or empty. Defaults to `"special"`.
encoding	the name of an encoding file. See `postscript` for details. Defaults to `"default"`.
bg	the initial background color to be used. Defaults to `"transparent"`.
fg	the initial foreground color to be used. Defaults to `"black"`.
pointsize	the default point size to be used. Strictly speaking, in bp, that is 1/72 of an inch, but approximately in points. Defaults to `12`.
pagecentre	logical: should the device region be centred on the page? -- is only relevant for `paper != "special"`. Defaults to `TRUE`.
colormodel	a character string describing the color model: currently allowed values are `"srgb"`, `"gray"` (or `"grey"`) and `"cmyk"`. Defaults to `"srgb"`. See section ‘Color models’.
useDingbats	logical. Should small circles be rendered *via* the Dingbats font? Defaults to

`TRUE` , which produces smaller and better output. Setting this to `FALSE` can work around font display problems in broken PDF viewers: although this font is one of the 14 guaranteed to be available in all PDF viewers, that guarantee is not always honoured.

On Unix-alikes (incl. Mac), see the 'Note' for a possible fix for some viewers.

useKerning logical. Should kerning corrections be included in setting text and calculating string widths? Defaults to `TRUE` .

fillOddEven logical controlling the polygon fill mode: see `polygon` for details. Defaults to `FALSE` .

compress logical. Should PDF streams be generated with Flate compression? Defaults to `TRUE` .

Details

All arguments except `file` default to values given by `pdf.options()` . The ultimate defaults are quoted in the arguments section.

`pdf()` opens the file `file` and the PDF commands needed to plot any graphics requested are sent to that file.

The `file` argument is interpreted as a C integer format as used by `sprintf` , with integer argument the page number. The default gives files `Rplot001.pdf` , …, `Rplot999.pdf` , `Rplot1000.pdf` , ….

The `family` argument can be used to specify a PDF-specific font family as the initial/default font for the device. If additional font families are to be used they should be included in the `fonts` argument.

If a device-independent R graphics font family is specified (e.g., via `par(family = )` in the graphics package), the PDF device makes use of the PostScript font mappings to convert the R graphics font family to a PDF-specific font family description. (See the documentation for `pdfFonts` .)

This device does *not* embed fonts in the PDF file, so it is only straightforward to use mappings to the font families that can be assumed to be available in any PDF viewer: `"Times"` (equivalently `"serif"`), `"Helvetica"` (equivalently `"sans"`) and `"Courier"` (equivalently `"mono"`). Other families may be specified, but it is the user's responsibility to ensure that these fonts are available on the system and third-party software (e.g., Ghostscript) may be required to embed the fonts so that the PDF can be included in other documents (e.g., LaTeX): see `embedFonts`. The URW-based families described for `postscript` can be used with viewers, platform dependently:

on Unix-alikes viewers set up to use URW fonts, which is usual with those based on `xpdf` or Ghostscript.

on Windows viewers such as GSView which utilise URW fonts.

Since `embedFonts` makes use of Ghostscript, it should be able to embed the URW-based families for use with other viewers.

See `postscript` for details of encodings, as the internal code is shared between the drivers. The native PDF encoding is given in file `PDFDoc.enc`.

The PDF produced is fairly simple, with each page being represented as a single stream (by default compressed and possibly with references to raster images). The R graphics model does not distinguish graphics objects at the level of the driver interface.

The `version` argument declares the version of PDF that gets produced. The version must be at least 1.2 when compression is used, 1.4 for semi-transparent output to be understood, and at least 1.3 if CID fonts are to be used: if any of these features are used the version number will be increased (with a warning). (PDF 1.4 was first supported by Acrobat 5 in 2001; it is very unlikely not to be supported in a current viewer.)

Line widths as controlled by `par(lwd = )` are in multiples of 1/96 inch. Multiples less than 1 are allowed. `pch = "."` with `cex = 1` corresponds to a square of side 1/72 inch, which is also the 'pixel' size assumed for graphics parameters such as `"cra"`.

The `paper` argument sets the `/MediaBox` entry in the file, which defaults to `width` by `height`. If it is set to something other than `"special"`, a device region of the specified size is (by default) centred on the rectangle given by the paper size: if either `width` or `height` is less than `0.1` or too large to give a total margin of 0.5 inch, it is reset to the corresponding paper dimension minus 0.5. Thus if you want the default behaviour of `postscript` use `pdf(paper = "a4r", width = 0, height = 0)` to centre the device region on a landscape A4 page with 0.25 inch margins.

When the background colour is fully transparent (as is the initial default value), the PDF produced does not paint the background. Most PDF viewers will use a white canvas so the visual effect is if the background were white. This will not be the case when printing onto coloured paper, though.

Note

If you see problems with PDF output, do remember that the problem is much more likely to be in your viewer than in R. Try another viewer if possible. Symptoms for which the viewer has been at fault are apparent grids on image plots (turn off graphics anti-aliasing in your viewer if you can) and missing or incorrect glyphs in text (viewers silently doing font substitution).

Unfortunately the default viewers on most Linux and macOS systems have these problems, and no obvious way to turn off graphics anti-aliasing.

Acrobat Reader does not use the fonts specified but rather emulates them from multiple-master fonts. This can be seen in imprecise centering of characters, for example the multiply and divide signs in Helvetica. This can be circumvented by embedding fonts where possible. Most other viewers substitute fonts, e.g.URW fonts for the standard Helvetica and Times fonts, and these too often have different font metrics from the true fonts.

Acrobat Reader can be extended by ‘font packs’, and these will be needed for the full use of encodings other than Latin-1 (although they may be offered for download as needed).

On some Unix-alike systems the default plotting character `pch = 1` was displayed in some PDF viewers incorrectly as a `"q"` character. (These seem to be viewers based on the `poppler` PDF rendering library). This may be due to incorrect or incomplete mapping of font names to those used by the system. Adding the following lines to `~/.fonts.conf` or `/etc/fonts/local.conf` may circumvent this problem, although this has largely been corrected on the affected systems.

```
<fontconfig>
<alias binding="same">
  <family>ZapfDingbats</family>
  <accept><family>Dingbats</family></accept>
</alias>
</fontconfig>
```

Some further workarounds for problems with symbol fonts on viewers using 'fontconfig' are given in the 'Cairo Fonts' section of the help for `X11` .

On Windows: The TeXworks PDF viewer was one of those which has been seen to fail to display Dingbats (used by e.g. `pch = 1`) correctly. Whereas on other platforms the problems seen were incorrect output, on Windows points were silently omitted: however recent versions seem to manage to display Dingbats.

There is a different font bug in the `pdf.js` viewer included in Firefox 19 and later: that maps Dingbats to the Symbol font and so displays symbols such `pch = 1` as lambda.

Color Models

The default color model (`"srgb"`) is sRGB. Model `"gray"` (or `"grey"`) maps sRGB colors to greyscale using perceived luminosity (biased towards green). `"cmyk"` outputs in CMYK colorspace. The simplest possible conversion from sRGB to CMYK is used (https://en.wikipedia.org/wiki/CMYK_color_model#Mapping_RGB_to_CMYK), and raster images are output in RGB.

Also available for backwards compatibility is model `"rgb"` which uses uncalibrated RGB and corresponds to the model used with that name in R prior to 2.13.0. Some viewers may render some plots in that colorspace faster than in sRGB, and the plot files will be smaller.

Conventions

This section describes the implementation of the conventions for graphics devices set out in the "R Internals Manual".

- The default device size is 7 inches square.
- Font sizes are in big points.
- The default font family is Helvetica.
- Line widths are as a multiple of 1/96 inch, with a minimum of 0.01 enforced.
- Circles of any radius are allowed. Unless `useDingbats = FALSE` , opaque circles of less than 10 big points radius are rendered using char 108 in the Dingbats font: all semi-transparent and larger circles using a B<U+00E9>zier curve for each quadrant.

- Colours are by default specified as sRGB.

At very small line widths, the line type may be forced to solid.

Printing

Except on Windows it is possible to print directly from `pdf` by something like (this is appropriate for a CUPS printing system):

```
    pdf("|lp -o landscape", paper = "a4r")
```

This forces `onefile = TRUE` .

See Also

`pdfFonts` , `pdf.options` , `embedFonts` , `Devices` , `postscript` .

`cairo_pdf` and (on macOS only) `quartz` for other devices that can produce PDF.

More details of font families and encodings and especially handling text in a non-Latin-1 encoding and embedding fonts can be found in

Paul Murrell and Brian Ripley (2006) Non-standard fonts in PostScript and PDF graphics. *R News*, 6(2):41--47. https://www.r-project.org/doc/Rnews/Rnews_2006-2.pdf.

Examples

script.R R Console

```
# NOT RUN {
## Test function for encodings
TestChars <- function(encoding = "ISOLatin1", ...)
{
    pdf(encoding = encoding, ...)
    par(pty = "s")
    plot(c(-1,16), c(-1,16), type = "n", xlab = "", ylab = "",
         xaxs = "i", yaxs = "i")
         title(paste("Centred chars in encoding", encoding))
    grid(17, 17, lty = 1)
    for(i in c(32:255)) {
```

Run

Community examples

Looks like there are no examples yet.

Post a new example:

~~~~
## New example
Use markdown to format your example

R code blocks are runnable and interactive:
```r
a <- 2
print(a)
```

You can also display normal code blocks
```
var a = b
```
~~~~

Submit your example