R.devices overview

Henrik Bengtsson

NA

Abstract

The *R.devices* package provides functions for creating plots and image files in a unified way regardless of output format (EPS, PDF, PNG, SVG, TIFF, WMF, etc.). Default device options as well as scales and aspect ratios are controlled in a uniform way across all device types. Switching output format requires minimal changes in code. This package is ideal for large-scale batch processing, because it will never leave open graphics devices or incomplete image files behind, even on errors or user interrupts.

Keywords: devices, graphics, plots, figures

This vignette is distributed as part of the R.devices package, which is available on CRAN (https://cran.r-project.org/). Feedback is very much appreciated.

Contents

1 Creating image files

When creating image files using one of the built-in R device functions (e.g. `pdf()`) several device specific arguments need to the specified. For instance, when creating a PDF file with aspect ratio 0.6, one will do something like

```
pdf("GaussianDensity.pdf", width=7, height=0.6*7)
curve(dnorm, from=-5, to=+5)
dev.off()
```

If one later wish to output a PNG file instead, one has to change (i) the name of the function, (ii) filename, (iii) the `width` and (iv) the `height` arguments, e.g.

```
png("GaussianDensity.png", width=480, height=0.6*480)
curve(dnorm, from=-5, to=+5)
dev.off()
```

Changing output formats is not only tedious but also error prone, e.g. you may forget to change the filename extension or create images with ridiculously large or small dimensions because units are not the same across devices types.

1.1 devEval()

To overcome the above and other hurdles, the `devEval()` function was created. When using `devEval()` it is only the argument that specify the image format that needs to be modified. For instance,

```
devEval("pdf", name="GaussianDensity", aspectRatio=0.6, {
  curve(dnorm, from=-5, to=+5)
})
```

creates a PDF file named *GaussianDensity.pdf* that is 7.0 inches wide and 4.2 inches tall (for default dimensions see Section 2), whereas

```
devEval("png", name="GaussianDensity", aspectRatio=0.6, {
  curve(dnorm, from=-5, to=+5)
})
```

creates a PNG file named *GaussianDensity.png* using `grDevices::png()`. The created PNG has an height-to-width aspect ratio of 6:10 (Section 2). Since the `png()` function might not be supported on all platforms, you can specify a set of *alternatives* as:

```
devEval("png|cairo_png|CairoPNG|png2", name="GaussianDensity", aspectRatio=0.6, {
  curve(dnorm, from=-5, to=+5)
})
```

which will the attempt each of them in order. A shortcut for the above is to use the *alias* specification:

```
devEval("{png}", name="GaussianDensity", aspectRatio=0.6, {
  curve(dnorm, from=-5, to=+5)
})
```

By specifying the `scale` argument, it is possible to create an image file with a smaller or a larger dimension relative to that of the default, e.g.

```
devEval("png", name="GaussianDensity,large", aspectRatio=0.6, scale=2, {
  curve(dnorm, from=-5, to=+5)
})
```

creates a PNG file named *GaussianDensity,large.png* that is twice as tall and twice as wide as the previous PNG image. Note also how in none of the above examples there is a need for closing the device via `dev.off()`, which is sometimes forgotten, particularly by newcomers to R. The graphical device opened is also guaranteed to be closed by `devEval()` regardless of errors or interrupts (Section 1.8).

1.2 Creating multiple image files of different formats in one call

The `devEval()` function can also be used to generate image files or different formats in one call. For instance,

```
devEval(c("pdf", "svg"), name="GaussianDensity", aspectRatio=0.6,
  curve(dnorm, from=-5, to=+5)
)
```

creates image files *GaussianDensity.pdf*, and *GaussianDensity.svg*. To adjust the default image dimensions for each format, see Section 2.

1.3 Creating one image file trying different device types

Some image formats, or more precisely some device types, are only supported on certain systems. If that is the case, and it does not matter exactly which of many image formats is used, then `devEval()` can be used to generate an image file based on the first supported device type. For instance,

```
devEval("png", name="GaussianDensity", aspectRatio=0.6,
  curve(dnorm, from=-5, to=+5)
)
```

will first try to create a PNG file and if that does not work it will try to create a PDF file and as a last resort an EPS file. It is only if none of them work that an error is generated.

1.4 toEPS(), toPDF(), toPNG() etc.

For conveniency, there exists a set of `toNNN()` functions that basically are wrappers for `devEval()`. For instance, instead of calling `devEval("pdf", ...)` one can use `toPDF(...)` as

```
toPDF("GaussianDensity,large", aspectRatio=0.6, scale=2, {
  curve(dnorm, from=-5, to=+5)
})
```

The following `toNNN()` functions are currently available: `toBMP()`, `toCairoWin()`, `toCairoX11()`, `toDefault()`, `toEMF()`, `toEPS()`, `toFavicon()`, `toNullDev()`, `toPDF()`, `toPNG()`, `toQuartz()`, `toRStudioGD()`, `toSVG()`, `toTIFF()`, `toWMF()`, and `toWindows()`.

1.5 Setting default output directory

All figures created by `devEval()/toNNN()` are by default written to the *figures/* directory (created if missing), which can be overridden by passing argument `path` to `devEval()`. The default figure path can be change by setting "global" device option `"path"`, e.g.

```
devOptions("*", path="figures/col/")
```

1.6 Names and comma-separated tags

The filename used by `devEval()/toNNN()`, is made up of argument `name`, followed by comma-separated argument `tags` (an optional character vector) and a filename extension (specified by the device type). Argument `tags` provides a convenient way to adjust the filename, e.g.

```
for (ar in c(0.6, 0.8)) {
  arTag <- sprintf("aspect=%g", ar)
  for (sc in 2:4) {
    scaleTag <- sprintf("scale=%d", sc)
    toEPS("GaussianDensity", tags=c(arTag, scaleTag), aspectRatio=ar, scale=sc, {
      curve(dnorm, from=-5, to=+5)
    })
  }
})
```

which creates six images files (*GaussianDensity,aspect=0.6,scale=2.eps*, *GaussianDensity,aspect=0.6,scale=3.eps*, ..., and *GaussianDensity,aspect=0.8,scale=4.eps*). To use a different separator specify argument `sep`, or change the global device option, e.g.

```
devOptions("*", sep="_")
```

1.7 Overwriting existing figure files

By default, existing figure files created by `devEval()/toNNN()` are overwritten without notice. By passing argument `force=FALSE` to `devEval()`, existing figure files will be skipped. To change the default, do:

```
devOptions("*", force=FALSE)
```

Note that whenever a figure is skipped this way, it also means that none of the expressions in `devEval(..., {<exprs>})` are executed. This will speed up the processing, but it also means that the rest of your code must not rely on such code being executed.

1.8 No more incomplete image files

The `devEval()/toNNN()` functions create image files atomically. When creating image files by opening a device, calling a set of plot functions and then closing the device (`pdf(...); {...}; dev.off()`), there is a risk of creating an incomplete file whenever an error or an interrupt occurs while plotting. By contrast, `devEval()/toNNN()` is fault tolerant and guarantees that the image file created is complete; if an error or an interrupt occurs, then the default is to remove the incomplete image. For instance, the following will not result in an image file:

```
toPDF("GaussianDensity", {
  curve(dnorm, from=-5, to=+5)
  abline(v=log("a"))
})
```

because the last plot statement generates an error. To further lower the risk for incomplete image files, for instance due to abrupt power failures, all image files are first written to a temporary file which is renamed to the final file only when the plotting is complete. This is useful when for instance running large non-interactive batch jobs that creates hundreds or thousands of image files.

1.9 Including images in RSP-embedded LaTeX documents

By using RSP markup, image files can be included in for instance LaTeX, Sweave and knitr documents in a very clean fashion, while keeping full control of all image formatting. For instance, the plot in Figure 1 was included as:

```
\includegraphics{<%=toPDF("MyGaussianDensity", aspectRatio=0.6, {
  curve(dnorm, from=-5, to=+5)
})%>}
```

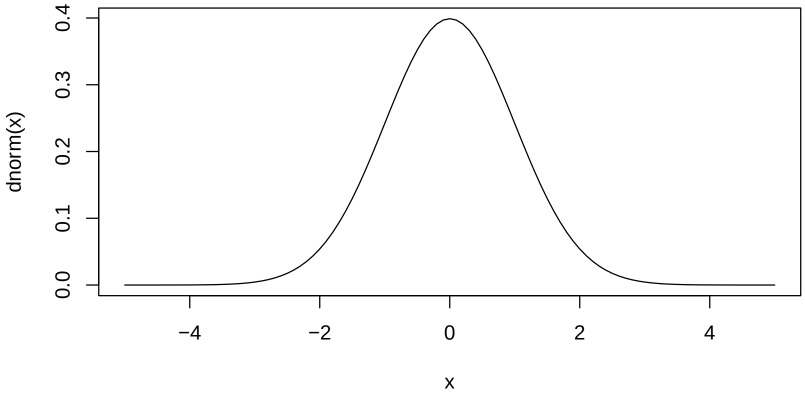

Figure 1: This graph was generated using `toPDF()` and then include into the LaTeX document using RSP.

For more details on RSP, see the vignettes of the *R.rsp* package (available on CRAN).

2 Default device options

2.1 devOptions()

The `devOptions()` function provides a unified interface to getting and settings common options for the various graphical devices available in R. When using one of the `toNNN()` functions, `devEval()` or `devNew()`, the device options used are given by `devOptions()`. For example, to see the current settings used by PDF device, do:

```
> str(devOptions("pdf"))
List of 22
 $ width      : num 7
 $ height     : num 7
 $ onefile    : logi TRUE
 $ family     : chr "Helvetica"
 $ title      : chr "R Graphics Output"
 $ fonts      : NULL
 $ version    : chr "1.4"
 $ paper      : chr "special"
 $ encoding   : chr "default"
 $ bg         : chr "transparent"
 $ fg         : chr "black"
 $ pointsize  : num 12
 $ pagecentre : logi TRUE
 $ colormodel : chr "srgb"
 $ useDingbats: logi FALSE
 $ useKerning : logi TRUE
 $ fillOddEven: logi FALSE
 $ compress   : logi TRUE
 $ timestamp  : logi TRUE
 $ producer   : logi TRUE
 $ author     : chr ""
 $ file       : language if (onefile) "Rplots.pdf" else "Rplot%03d.pdf"
```

To change one or several options, do:

```
> devOptions("pdf", width = 5, bg = "lightblue")
```

To reset the options back to the built-in defaults, do:

```
> devOptions("pdf", reset = TRUE)
```

To get an overview of a set of common options for all supported devices, do:

```
> devOptions()[, c("width", "height", "bg", "fg", "pointsize")]
           width height bg            fg      pointsize
*          NULL  NULL   NULL          NULL    NULL
CairoJPEG  480   480    "white"       NULL    12
CairoPDF   6     6      "transparent" "black" 12
CairoPNG   840   480    "white"       NULL    12
CairoPS    8.27  11.7   "transparent" "black" 12
CairoSVG   6     6      "transparent" NULL    12
CairoTIFF  480   480    "white"       NULL    12
CairoX11   7     7      "transparent" NULL    12
X11        NA    NA     "transparent" NULL    12
bmp        480   480    "white"       NULL    12
cairo_pdf  7     7      "white"       NULL    12
cairo_ps   7     7      "white"       NULL    12
eps        7     7      "transparent" "black" 12
favicon    32    ?      "transparent" NULL    12
jpeg       480   480    "white"       NULL    12
jpeg2      480   480    "transparent" "black" 12
nulldev    480   480    "white"       ?       12
pdf        7     7      "transparent" "black" 12
pictex     5     4      "white"       "black" NULL
png        840   480    "white"       NULL    12
png2       840   480    "transparent" "black" 12
```

```
postscript 8.27  11.7   "transparent" "black" 12
quartz     7     7      "transparent" NULL    12
svg        7     7      "white"       NULL    12
tiff       480   480    "white"       NULL    12
x11        NA    NA     "transparent" NULL    12
xfig       8.27  11.7   "transparent" "black" 12
```

2.2 Under the hood (advanced)

The `devOptions()` function tries as far as possible to infer the default options from the default arguments of the device function and any additional options for that device, e.g. `formals(pdf)` and `pdf.options()`. Likewise, when setting an option it uses the standard interfaces to do so, whenever possible. This means that for instance `pdf()` will also be affected by `devOptions("pdf", width=5)`. Note that this may not be the case for all devices, because their options cannot be set. Instead they are all specified as arguments when opening the device, e.g. `png()` will *not* be affected by `devOptions("png", width=1024)`. This is why we recommend to always use `devNew()` in place of `dev.new()`, or better, `devEval()` or the corresponding `toNNN()` function, which all respects the options set via `devOptions()`.

Appendix

Session information

- R version 4.5.2 (2025-10-31), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Time zone: `America/Los_Angeles`
- TZcode source: `system (glibc)`
- Running under: `Ubuntu 24.04.3 LTS`
- Matrix products: default
- BLAS: `/home/hb/shared/software/CBI/_ubuntu24_04/R-4.5.2-gcc13/lib/R/lib/libRblas.so`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Cairo 1.7-0, R.devices 2.17.3, R.methodsS3 1.8.2, R.oo 1.27.1
- Loaded via a namespace (and not attached): R.cache 0.17.0, R.rsp 0.46.0, R.utils 2.13.0, base64enc 0.1-3, compiler 4.5.2, digest 0.6.39, tools 4.5.2

This report was automatically generated using `rfile()` of the R.rsp package.