Package 'R.devices'

January 20, 2025

Version 2.17.2

Depends R (>= 2.14.0)

Imports grDevices, R.methodsS3 (>= 1.8.1), R.oo (>= 1.24.0), R.utils (>= 2.10.1), base64enc (>= 0.1-2)

Suggests digest (>= 0.6.13), Cairo (>= 1.5-9), R.rsp

VignetteBuilder R.rsp

Author Henrik Bengtsson [aut, cre, cph]

Maintainer Henrik Bengtsson <henrikb@braju.com>

Title Unified Handling of Graphics Devices

Description 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.

License LGPL (>= 2.1)

URL https://henrikbengtsson.github.io/R.devices/, https://github.com/HenrikBengtsson/R.devices

BugReports https://github.com/HenrikBengtsson/R.devices/issues

LazyLoad TRUE

NeedsCompilation no

Repository CRAN

Date/Publication 2024-01-29 13:30:11 UTC

Contents

R.devices-package 2
architecture 3
capturePlot 4

devDone 5
devEval 6
devGetLabel 8
devIsInteractive 9
devIsOpen 9
devList 11
devNew 11
devOff 13
devOptions 13
devSet 15
devSetLabel 15
toNNN 16
withPar 17

Index **19**

`R.devices-package` *Package R.devices*

Description

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.

To get started

- Vignette 'R.devices overview'
- `toEPS`(), `toPDF`(), `toPNG`(), ... - evaluate graphics code and atomically save plot to a figure file.

How to cite this package

To cite this package, please use:

```
@Manual{,
  title = {R.devices: Unified Handling of Graphics Devices},
  author = {Henrik Bengtsson},
  year = {2024},
  note = {R package version 2.17.2,
https://github.com/HenrikBengtsson/R.devices},
  url = {https://henrikbengtsson.github.io/R.devices/},
}
```

License

LGPL (>= 2.1).

Author(s)

Henrik Bengtsson [aut, cre, cph].

`architecture` *Get the architecture of an object or coerce it into another*

Description

Get the architecture of an object or coerce it into another

Usage

```
architecture(x, ...)

as.architecture(
  x,
  ostype = .Platform$OS.type,
  arch = R.version$arch,
  ptrsize = .Machine$sizeof.pointer,
  endian = .Platform$endian,
  ...
)
```

Arguments

`x`	The object to be coerced.
`...`	(optional) Additional arguments passed to the underlying method.
`ostype`	A character string, e.g. `"unix"` and `"windows"`.
`arch`	A character string, e.g. `"i386"`, `"i686"` and `"x86_64"`.
`ptrsize`	The target pointer size - either `4L` or `8L`.
`endian`	The target endianess - either `"little"` or `"big"`.

Value

`architecture()` returns a named list with character element `ostype` and `arch`, integer element `ptrsize`, and character element `endian`. These elements take a missing values if they could not be inferred.

`as.architecture()` returns a coerced version of `x`. If no coercion was needed, then `x` itself is returned.

`capturePlot` *Captures a plot such that it can be redrawn later/elsewhere*

Description

Captures a plot such that it can be redrawn later/elsewhere.

This feature is only supported in R (>= 3.3.0).

Usage

```
capturePlot(expr, envir=parent.frame(), type=nulldev, ...)
```

Arguments

`expr`	The `expression` of graphing commands to be evaluated.
`envir`	The `environment` where `expr` should be evaluated.
`type`	The type of graphics device used in the background. The choice should not matter since the result should be identical regardless. All graphics is captured but any output is also voided by sending the output to a "null" file.
`...`	Additional arguments passed to the graphics device.

Details

Note that plot dimensions/aspect ratios are not recorded. This means that one does not have to worry about those when recording the plot. Instead, they are specified when setting up the graphics device(s) in which the recorded plot is replayed (see example).

Value

Returns a `recordedplot` object, which can be `replayPlot()`:ed. If replayed in an interactive session, the plot is displayed in a new window. For conveniency, the object is also replayed when `print()`:ed.

Replaying / replotting on a different architecture

In order to replay a `recordedplot` object, it has to be replayed on an architecture that is compatible with the one who created the object. If this is not the case, then `replayPlot()` will generate an *Incompatible graphics state* error. The `as.architecture()` function of this package tries to coerce between different architectures, such that one can replay across architectures using `replayPlot(as.architectures(g))`. For convenience, the recorded plot returned by `capturePlot()` is automatically coerced when `print()`:ed.

Author(s)

Henrik Bengtsson

References

[1] Paul Murrell et al., *Recording and Replaying the Graphics Engine Display List*, December 2015. https://www.stat.auckland.ac.nz/~paul/Reports/DisplayList/dl-record.html

See Also

Internally `recordPlot()` is used.

Examples

```
if (getRversion() >= "3.3.0") {
oopts <- R.devices::devOptions("*", path=file.path(tempdir(), "figures"))

g <- capturePlot({
  plot(1:10)
})

## Display
print(g)

## Display with a 2/3 height-to-width aspect ratio
toDefault(aspectRatio=2/3, { print(g) })

## Redraw to many output formats using whatever PNG, EPS, and PDF
## device outputs available
devEval(c("{png}", "{eps}", "{pdf}"), aspectRatio=2/3, print(g))

R.devices::devOptions("*", path=oopts$path)
} ## if (getRversion() >= "3.3.0")
```

---

devDone *Closes zero or more open devices except screen (interactive) devices*

Description

Closes zero or more open devices except screen (interactive) devices.

Usage

```
devDone(which=dev.cur(), ...)
```

Arguments

`which`	An index (`numeric`) `vector` or a label (`character`) `vector`.
`...`	Not used.

Value

Returns (invisibly) dev.cur().

Author(s)

Henrik Bengtsson

See Also

devOff(). dev.interactive.

---

devEval *Opens a new graphics device, evaluate (graphing) code, and closes device*

Description

Opens a new graphics device, evaluate (graphing) code, and closes device.

Usage

```
devEval(type=getOption("device"), expr, initially=NULL, finally=NULL,
  envir=parent.frame(), name=NULL, tags=NULL, sep=getDevOption(type, "sep",
  default = ","), ..., ext=NULL, filename=NULL, path=getDevOption(type, "path",
  default = "figures"), field=getDevOption(type, name = "field"),
  onIncomplete=c("remove", "rename", "keep"), force=getDevOption(type, "force",
  default = TRUE), which=dev.cur(), .exprAsIs=FALSE, .allowUnknownArgs=FALSE)
```

Arguments

type	Specifies the type of graphics device to be used. The device is created and opened using devNew(). Multiple types may be specified.
expr	The expression of graphing commands to be evaluated.
initially, finally	Optional expression:s to be evaluated before and after expr. If type specifies multiple devices, these optional expression:s are only evaluated ones.
envir	The environment where expr should be evaluated.
name, tags, sep	The fullname name of the image is specified as the name with optional sep-separated tags appended.
ext	The filename extension of the image file generated, if any. By default, it is inferred from argument type.
...	Additional arguments passed to devNew().
filename	The filename of the image saved, if any. By default, it is composed of arguments name, tags, sep, and ext. See also below.
path	The directory where then image should be saved, if any.

field	An optional character string specifying a specific field of the named result list to be returned.
onIncomplete	A character string specifying what to do with an image file that was incompletely generated due to an interrupt or an error.
force	If TRUE, and the image file already exists, then it is overwritten, otherwise not.
which	A vector of devices to be copied. Only applied if argument expr is missing.
.exprAsIs, .allowUnknownArgs	(Internal use only).

Value

Returns a DevEvalFileProduct if the device generated an image file, otherwise an DevEvalProduct. If argument field is given, then the field of the DevEvalProduct is returned instead. *Note that the default return value may be changed in future releases.*

Generated image file

If created, the generated image file is saved in the directory specified by argument path with a filename consisting of the name followed by optional comma-separated tags and a filename extension given by argument ext.

By default, the image file is only created if the expr is evaluated completely. If it is, for instance, interrupted by the user or due to an error, then any incomplete/blank image file that was created will be removed. This behavior can be turned of using argument onIncomplete.

Author(s)

Henrik Bengtsson

See Also

To change default device parameters such as the width or the height, devOptions(). devNew().

Examples

```
# Plot to PNG using one whatever PNG device driver is available
res <- devEval("{png}", name="MyPlot", tags=c("10", "rnd"), aspectRatio=0.7, {
  plot(1:10)
})
print(res$pathname)
# [1] "figures/MyPlot,10,rnd.png"

str(res$dataURI)
# chr "..."

## Plot to PDF using grDevices::pdf()
res <- devEval("pdf", name="MyPlot", tags=c("10", "rnd"), aspectRatio=0.7, {
  plot(1:10)
})
print(res$pathname)
```

```
# [1] "figures/MyPlot,10,rnd.pdf"


## Plot to EPS using R.devices::eps()
res <- devEval("eps", name="MyPlot", tags=c("10", "rnd"), aspectRatio=0.7, {
  plot(1:10)
})
print(res$pathname)
# [1] "figures/MyPlot,10,rnd.eps"
```

---

devGetLabel *Gets the labels of zero or more devices*

Description

Gets the labels of zero or more devices.

Usage

```
devGetLabel(which=dev.cur(), ...)
```

Arguments

which	An index (numeric) vector or a label (character) vector.
...	Not used.

Value

Returns a character vector. If a device does not exist, an error is thrown.

Author(s)

Henrik Bengtsson

See Also

devSetLabel() and devList().

devIsInteractive *Checks whether a device type is interactive or not*

Description

Checks whether a device type is interactive or not.

Usage

```
devIsInteractive(types, ...)
```

Arguments

types	A character vector.
...	Not used.

Value

Returns a logical vector with TRUE if the device type is interactive, otherwise FALSE.

Author(s)

Henrik Bengtsson

See Also

Internally, deviceIsInteractive is used.

devIsOpen *Checks if zero or more devices are open or not*

Description

Checks if zero or more devices are open or not.

Usage

```
devIsOpen(which=dev.cur(), ...)
```

Arguments

which	An index (numeric) vector or a label (character) vector.
...	Not used.

Value

Returns a named logical vector with TRUE if a device is open, otherwise FALSE.

Author(s)

Henrik Bengtsson

Examples

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Use devices for conditional processing of code.
# Close devices to rerun code.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
cat("Currently opened device:\n")
print(devList())

# Alt A: Use device index counter (starting with the 16:th)
fig <- 15
if (!devIsOpen(fig <- fig + 1)) {
  devSet(fig)
  cat("Figure", fig, "\n")
  plot(1:10)
}
cat("Currently opened device:\n")
print(devList())

if (!devIsOpen(fig <- fig + 1)) {
  devSet(fig)
  cat("Figure", fig, "\n")
  plot(1:10)
}
cat("Currently opened device:\n")
print(devList())

# Alt B: Use device labels
if (!devIsOpen(label <- "part 1")) {
  devSet(label)
  cat("Part 1\n")
  plot(1:10)
}
cat("Currently opened device:\n")
print(devList())


if (!devIsOpen(label <- "part 2")) {
  devSet(label)
  cat("Part 2\n")
  plot(1:10)
}
cat("Currently opened device:\n")
print(devList())
```

devList *Lists the indices of the open devices named by their labels*

Description

Lists the indices of the open devices named by their labels.

Usage

```
devList(interactiveOnly=FALSE, dropNull=TRUE, ...)
```

Arguments

interactiveOnly	If TRUE, only open interactive/screen devices are returned.
dropNull	If TRUE, the "null" device (device index 1) is not returned.
...	Not used.

Value

Returns a named integer vector.

Author(s)

Henrik Bengtsson

See Also

dev.list().

devNew *Opens a new device*

Description

Opens a new device.

Usage

```
devNew(type=getOption("device"), ..., scale=1, aspectRatio=1, par=NULL, label=NULL)
```

Arguments

type	A character string specifying the type of device to be opened. This string should match the name of an existing device function.
...	Additional arguments passed to the device function, e.g. width and height. If not given, the are inferred from devOptions().
scale	A numeric scalar factor specifying how much the width and the height should be rescaled.
aspectRatio	A numeric ratio specifying the aspect ratio of the image. See below.
par	An optional named list of graphical settings applied, that is, passed to par, immediately after opening the device.
label	An optional character string specifying the label of the opened device.

Value

Returns the device index of the opened device.

Width and heights

The default width and height of the generated image is specific to the type of device used. There is not straightforward programmatic way to infer these defaults; here we use devOptions(), which in most cases returns the correct defaults.

Aspect ratio

The aspect ratio of an image is the height relative to the width. If argument height is not given (or NULL), it is calculated as aspectRatio*width as long as they are given. Likewise, if argument width is not given (or NULL), it is calculated as width/aspectRatio as long as they are given. If neither width nor height is given, then width defaults to devOptions(type)$width. If both width and height are given, then aspectRatio is ignored.

Author(s)

Henrik Bengtsson

See Also

devDone() and devOff(). For simplified generation of image files, see devEval().

devOff *Closes zero or more devices*

Description

Closes zero or more devices.

Usage

```
devOff(which=dev.cur(), ...)
```

Arguments

which	An index (numeric) vector or a label (character) vector.
...	Not used.

Value

Returns dev.cur().

Author(s)

Henrik Bengtsson

See Also

devDone(). Internally, dev.off() is used.

devOptions *Gets the default device options*

Description

Gets the default device options as given by predefined devices options adjusted for the default arguments of the device function.

Usage

```
devOptions(type=NULL, custom=TRUE, special=TRUE, inherits=FALSE, drop=TRUE,
  options=list(), ..., reset=FALSE)
```

Arguments

type	A character string or a device function specifying the device to be queried. May also be a vector, for querying device options for multiple devices.
custom	If TRUE, also the default settings specific to this function is returned. For more details, see below.
special	A logical. For more details, see below.
inherits	If TRUE, the global option is used if the type-specific is not set (or NULL).
drop	If TRUE and only one device type is queried, then a list is returned, otherwise a matrix.
options	Optional named list of settings.
...	Optional named arguments for setting new defaults. For more details, see below.
reset	If TRUE, the device options are reset to R defaults.

Details

If argument special is TRUE, then the 'width' and 'height' options are adjusted according to the rules explained for argument 'paper' in pdf, postscript, and xfig.

Value

If drop=TRUE and a single device is queries, a named list is returned, otherwise a matrix is returned. If a requested device is not implemented/supported on the current system, then "empty" results are returned. If options were set, that is, if named options were specified via ..., then the list is returned invisibly, otherwise not.

Setting new defaults

When setting device options, the getOption("devOptions")[[type]] option is modified. This means that for such options to be effective, any device function needs to query also such options, which for instance is done by devNew().

Also, for certain devices (eps, pdf, postscript, quartz, windows and x11), builtin R device options are set.

Author(s)

Henrik Bengtsson

Examples

```
# Tabulate some of the default settings for known devices
print(devOptions()[,c("width", "height", "bg", "fg", "pointsize")])
```

devSet *Activates a device*

Description

Activates a device.

Usage

```
devSet(which=dev.next(), ...)
```

Arguments

which	An index (numeric) or a label (character). If neither, then a label corresponding to the checksum of which as generated by digest is used.
...	Not used.

Value

Returns what dev.set() returns.

Author(s)

Henrik Bengtsson

See Also

devOff() and devDone(). Internally, dev.set() is used.

devSetLabel *Sets the label of a device*

Description

Sets the label of a device.

Usage

```
devSetLabel(which=dev.cur(), label, ...)
```

Arguments

which	An index (numeric) or a label (character).
label	A character string.
...	Not used.

Value

Returns nothing.

Author(s)

Henrik Bengtsson

See Also

devGetLabel() and devList().

---

toNNN *Methods for creating image files of a specific format*

Description

Methods for creating image files of a specific format.

Usage

```
toBMP(name, ...)
toPDF(name, ...)
toPNG(name, ...)
toSVG(name, ...)
toTIFF(name, ...)
toEMF(name, ..., ext="emf")
toWMF(name, ..., ext="wmf")

toFavicon(..., name="favicon", ext="png",
          field=getDevOption("favicon", "field", default="htmlscript"))


toDefault(name, ...)
toQuartz(name, ...)
toX11(name, ...)
toWindows(name, ...)


toCairoWin(name, ...)
toCairoX11(name, ...)



toRStudioGD(name, ..., .allowUnknownArgs = TRUE)
```

Arguments

name	A character string specifying the name of the image file.
..., .allowUnknownArgs	Additional arguments passed to devEval(), e.g. tags and aspectRatio.
ext, field	Passed to devEval().

Value

Returns by default the DevEvalProduct. For toFavicon() the default return value is a character string.

Windows Metafile Format

Both toEMF() and toWMF() use the exact same graphics device (win.metafile()) and settings. They only differ by filename extension. The win.metafile() device function exists on Windows only; see the **grDevices** package for more details.

Author(s)

Henrik Bengtsson

See Also

These functions are wrappers for devEval(). See devOptions() to change the default dimensions for a specific device type.

---

withPar *Evaluate an R expression with graphical parameters set temporarily*

Description

Evaluate an R expression with graphical parameters set temporarily.

Usage

```
withPar(expr, ..., args=list(), envir=parent.frame())
```

Arguments

expr	The R expression to be evaluated.
...	Named options to be used.
args	(optional) Additional named options specified as a named list.
envir	The environment in which the expression should be evaluated.

Details

Upon exit (also on errors), this function will reset *all* (modifiable) graphical parameters to the state of options available upon entry. This means any parameters *modified* from evaluating `expr` will also be undone upon exit.

Value

Returns the results of the expression evaluated.

Author(s)

Henrik Bengtsson

See Also

Internally, `eval()` is used to evaluate the expression, and `par` to set graphical parameters.

Examples

```
withPar({
  layout(1:4)

  withPar({
    plot(1:10)
    plot(10:1)
  }, pch=4)

  withPar({
    plot(1:10)
    plot(10:1)
  }, pch=0, bg="yellow")
}, mar=c(2,2,1,1))
```

Index

∗ **IO**
withPar, 17
∗ **device**
capturePlot, 4
devDone, 5
devEval, 6
devGetLabel, 8
devIsInteractive, 9
devIsOpen, 9
devList, 11
devNew, 11
devOff, 13
devOptions, 13
devSet, 15
devSetLabel, 15
toNNN, 16
∗ **documentation**
toNNN, 16
∗ **package**
R.devices-package, 2
∗ **programming**
withPar, 17
∗ **utilities**
devDone, 5
devEval, 6
devGetLabel, 8
devIsInteractive, 9
devIsOpen, 9
devList, 11
devNew, 11
devOff, 13
devOptions, 13
devSet, 15
devSetLabel, 15
toNNN, 16

architecture, 3
as.architecture, *4*
as.architecture *(*architecture*)*, 3
asDataURI *(*toNNN*)*, 16

capturePlot, 4
character, *5*, *7–9*, *12–15*, *17*

dev.cur, *6*, *13*
dev.interactive, *6*
dev.list, *11*
dev.off, *13*
dev.set, *15*
devDone, 5, *12*, *13*, *15*
devDump *(*devEval*)*, 6
devEval, 6, *12*, *17*
DevEvalFileProduct, *7*
DevEvalProduct, *7*, *17*
devGetLabel, 8, *16*
deviceIsInteractive, *9*
devIsInteractive, 9
devIsOpen, 9
devList, *8*, 11, *16*
devNew, *6*, *7*, 11, *14*
devOff, *6*, *12*, 13, *15*
devOptions, *7*, *12*, 13, *17*
devSet, 15
devSetLabel, *8*, 15
digest, *15*

environment, *4*, *6*, *17*
eval, *18*
expression, *4*, *6*

FALSE, *9*, *10*
function, *12*, *14*

getDevOption *(*devOptions*)*, 13

integer, *11*

list, *7*, *12*, *14*, *17*
logical, *9*, *10*, *14*

matrix, *14*

NULL, *12*, *14*
numeric, *5*, *8*, *9*, *12*, *13*, *15*

par, *12*, *18*
pdf, *14*
postscript, *14*

R.devices *(*R.devices-package*)*, 2
R.devices-package, 2
recordPlot, *5*
replayPlot, *4*

suppressGraphics *(*devEval*)*, 6

toBMP *(*toNNN*)*, 16
toCairoWin *(*toNNN*)*, 16
toCairoX11 *(*toNNN*)*, 16
toDefault *(*toNNN*)*, 16
toEMF *(*toNNN*)*, 16
toEPS, *2*
toEPS *(*toNNN*)*, 16
toFavicon *(*toNNN*)*, 16
toNNN, 16
toNullDev *(*toNNN*)*, 16
toPDF, *2*
toPDF *(*toNNN*)*, 16
toPNG, *2*
toPNG *(*toNNN*)*, 16
toQuartz *(*toNNN*)*, 16
toRStudioGD *(*toNNN*)*, 16
toSVG *(*toNNN*)*, 16
toTIFF *(*toNNN*)*, 16
toWindows *(*toNNN*)*, 16
toWMF *(*toNNN*)*, 16
toX11 *(*toNNN*)*, 16
TRUE, *7*, *9–11*, *14*

vector, *5*, *7–11*, *13*, *14*

withPar, 17

xfig, *14*