Package: ragg (via r-universe)

October 28, 2026

Type Package

Title Graphic Devices Based on AGG

Version 1.5.2

Maintainer Thomas Lin Pedersen <thomas.pedersen@posit.co>

Description Anti-Grain Geometry (AGG) is a high-quality and high-performance 2D drawing library. The 'ragg' package provides a set of graphic devices based on AGG to use as alternative to the raster devices provided through the 'grDevices' package.

License MIT + file LICENSE

URL https://ragg.r-lib.org, https://github.com/r-lib/ragg

BugReports https://github.com/r-lib/ragg/issues

Imports systemfonts (>= 1.0.3), textshaping (>= 0.3.0)

Suggests covr, graphics, grid, testthat (>= 3.0.0)

LinkingTo systemfonts, textshaping

Config/build/compilation-database true

Config/Needs/website ggplot2, devoid, magick, bench, tidyr, ggridges, hexbin, sessioninfo, pkgdown, tidyverse/tidytemplate

Config/testthat/edition 3

Config/usethis/last-upkeep 2025-04-25

Encoding UTF-8

RoxygenNote 7.3.3

SystemRequirements freetype2, libpng, libtiff, libjpeg, libwebp, libwebpmux

NeedsCompilation yes

Author Thomas Lin Pedersen [cre, aut] (ORCID: <https://orcid.org/0000-0002-5147-4711>), Maxim Shemanarev [aut, cph] (Author of AGG), Tony Juricic [ctb, cph] (Contributor to AGG), Milan Marusinec [ctb, cph] (Contributor to AGG), Spencer Garrett [ctb] (Contributor to AGG), Posit Software, PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>)

Repository https://cran.r-universe.dev

Date/Publication 2026-03-23 08:50:08 UTC

RemoteUrl https://github.com/cran/ragg

RemoteRef HEAD

RemoteSha b9cb6b4eb0d03a8bf96ea0ad7e3f1045925bb94b

Contents

agg_capture 2
agg_jpeg 4
agg_png 5
agg_ppm 7
agg_record 8
agg_tiff 9
agg_webp 11
agg_webp_anim 12

Index **15**

`agg_capture` *Draw to a buffer that can be accessed directly*

Description

Usually the point of using a graphic device is to create a file or show the graphic on the screen. A few times we need the image data for further processing in R, and instead of writing it to a file and then reading it back into R the `agg_capture()` device lets you get the image data directly from the buffer. In contrast to the other devices this device returns a function, that when called will return the current state of the buffer.

Usage

```
agg_capture(
  width = 480,
  height = 480,
  units = "px",
  pointsize = 12,
  background = "white",
  res = 72,
  scaling = 1,
  snap_rect = TRUE,
  bg
)
```

Arguments

width, height	The dimensions of the device
units	The unit width and height is measured in, in either pixels ('px'), inches ('in'), millimeters ('mm'), or centimeter ('cm').
pointsize	The default pointsize of the device in pt. This will in general not have any effect on grid graphics (including ggplot2) as text size is always set explicitly there.
background	The background colour of the device
res	The resolution of the device. This setting will govern how device dimensions given in inches, centimeters, or millimeters will be converted to pixels. Further, it will be used to scale text sizes and linewidths
scaling	A scaling factor to apply to the rendered line width and text size. Useful for getting the right dimensions at the resolution that you need. If e.g. you need to render a plot at 4000x3000 pixels for it to fit into a layout, but you find that the result appears to small, you can increase the scaling argument to make everything appear bigger at the same resolution.
snap_rect	Should axis-aligned rectangles drawn with only fill snap to the pixel grid. This will prevent anti-aliasing artifacts when two rectangles are touching at their border.
bg	Same as background for compatibility with old graphic device APIs

Value

A function that when called returns the current state of the buffer. The return value of the function depends on the native argument. If FALSE (default) the return value is a matrix of colour values and if TRUE the return value is a nativeRaster object.

Examples

```
cap <- agg_capture()
plot(1:10, 1:10)

# Get the plot as a matrix
raster <- cap()

# Get the plot as a nativeRaster
raster_n <- cap(native = TRUE)

dev.off()

# Look at the output
plot(as.raster(raster))
```

agg_jpeg *Draw to a JPEG file*

Description

The JPEG file format is a lossy compressed file format developed in particular for digital photography. The format is not particularly well-suited for line drawings and text of the type normally associated with statistical plots as the compression algorithm creates noticable artefacts. It is, however, great for saving image data, e.g. heightmaps etc. Thus, for standard plots, it would be better to use `agg_png()`, but for plots that includes a high degree of raster image rendering this device will result in smaller plots with very little quality degradation.

Usage

```
agg_jpeg(
  filename = "Rplot%03d.jpeg",
  width = 480,
  height = 480,
  units = "px",
  pointsize = 12,
  background = "white",
  res = 72,
  scaling = 1,
  snap_rect = TRUE,
  quality = 75,
  smoothing = FALSE,
  method = "slow",
  bg
)
```

Arguments

`filename`	The name of the file. Follows the same semantics as the file naming in `grDevices::png()`, meaning that you can provide a `sprintf()` compliant string format to name multiple plots (such as the default value)
`width, height`	The dimensions of the device
`units`	The unit `width` and `height` is measured in, in either pixels (`'px'`), inches (`'in'`), millimeters (`'mm'`), or centimeter (`'cm'`).
`pointsize`	The default pointsize of the device in pt. This will in general not have any effect on grid graphics (including ggplot2) as text size is always set explicitly there.
`background`	The background colour of the device
`res`	The resolution of the device. This setting will govern how device dimensions given in inches, centimeters, or millimeters will be converted to pixels. Further, it will be used to scale text sizes and linewidths

scaling	A scaling factor to apply to the rendered line width and text size. Useful for getting the right dimensions at the resolution that you need. If e.g. you need to render a plot at 4000x3000 pixels for it to fit into a layout, but you find that the result appears to small, you can increase the `scaling` argument to make everything appear bigger at the same resolution.
snap_rect	Should axis-aligned rectangles drawn with only fill snap to the pixel grid. This will prevent anti-aliasing artifacts when two rectangles are touching at their border.
quality	An integer between `0` and `100` defining the quality/size tradeoff. Setting this to `100` will result in no compression.
smoothing	A smoothing factor to apply before compression, from `0` (no smoothing) to `100` (full smoothing). Can also by `FALSE` (no smoothing) or `TRUE` (full smoothing).
method	The compression algorithm to use. Either `'slow'`, `'fast'`, or `'float'`. Default is `'slow'` which works best for most cases. `'fast'` should only be used when quality is below `97` as it may result in worse performance at high quality settings. `'float'` is a legacy options that calculate the compression using floating point precission instead of with integers. It offers no quality benefit and is often much slower.
bg	Same as `background` for compatibility with old graphic device APIs

Note

Smoothing is only applied if ragg has been compiled against a jpeg library that supports smoothing.

Examples

```
file <- tempfile(fileext = '.jpeg')
agg_jpeg(file, quality = 50)
plot(sin, -pi, 2*pi)
dev.off()
```

---

agg_png *Draw to a PNG file*

Description

The PNG (Portable Network Graphic) format is one of the most ubiquitous today, due to its versatiliity and widespread support. It supports transparency as well as both 8 and 16 bit colour. The device uses default compression and filtering and will not use a colour palette as this is less useful for antialiased data. This means that it might be possible to compress the resulting image even more if size is of concern (though the defaults are often very good). In contrast to `grDevices::png()` the date and time will not be written to the file, meaning that similar plot code will produce identical files (a good feature if used with version control). It will, however, write in the dimensions of the image based on the `res` argument.

Usage

```
agg_png(
  filename = "Rplot%03d.png",
  width = 480,
  height = 480,
  units = "px",
  pointsize = 12,
  background = "white",
  res = 72,
  scaling = 1,
  snap_rect = TRUE,
  bitsize = 8,
  bg
)
```

Arguments

filename	The name of the file. Follows the same semantics as the file naming in grDevices::png(), meaning that you can provide a sprintf() compliant string format to name multiple plots (such as the default value)
width, height	The dimensions of the device
units	The unit width and height is measured in, in either pixels ('px'), inches ('in'), millimeters ('mm'), or centimeter ('cm').
pointsize	The default pointsize of the device in pt. This will in general not have any effect on grid graphics (including ggplot2) as text size is always set explicitly there.
background	The background colour of the device
res	The resolution of the device. This setting will govern how device dimensions given in inches, centimeters, or millimeters will be converted to pixels. Further, it will be used to scale text sizes and linewidths
scaling	A scaling factor to apply to the rendered line width and text size. Useful for getting the right dimensions at the resolution that you need. If e.g. you need to render a plot at 4000x3000 pixels for it to fit into a layout, but you find that the result appears to small, you can increase the scaling argument to make everything appear bigger at the same resolution.
snap_rect	Should axis-aligned rectangles drawn with only fill snap to the pixel grid. This will prevent anti-aliasing artifacts when two rectangles are touching at their border.
bitsize	Should the device record colour as 8 or 16bit
bg	Same as background for compatibility with old graphic device APIs

Examples

```
file <- tempfile(fileext = '.png')
agg_png(file)
plot(sin, -pi, 2*pi)
dev.off()
```

agg_ppm *Draw to a PPM file*

Description

The PPM (Portable Pixel Map) format defines one of the simplest storage formats available for image data. It is basically a raw 8bit RGB stream with a few bytes of information in the start. It goes without saying, that this file format is horribly inefficient and should only be used if you want to play around with a simple file format, or need a file-based image stream.

Usage

```
agg_ppm(
  filename = "Rplot%03d.ppm",
  width = 480,
  height = 480,
  units = "px",
  pointsize = 12,
  background = "white",
  res = 72,
  scaling = 1,
  snap_rect = TRUE,
  bg
)
```

Arguments

filename	The name of the file. Follows the same semantics as the file naming in grDevices::png(), meaning that you can provide a sprintf() compliant string format to name multiple plots (such as the default value)
width, height	The dimensions of the device
units	The unit width and height is measured in, in either pixels ('px'), inches ('in'), millimeters ('mm'), or centimeter ('cm').
pointsize	The default pointsize of the device in pt. This will in general not have any effect on grid graphics (including ggplot2) as text size is always set explicitly there.
background	The background colour of the device
res	The resolution of the device. This setting will govern how device dimensions given in inches, centimeters, or millimeters will be converted to pixels. Further, it will be used to scale text sizes and linewidths
scaling	A scaling factor to apply to the rendered line width and text size. Useful for getting the right dimensions at the resolution that you need. If e.g. you need to render a plot at 4000x3000 pixels for it to fit into a layout, but you find that the result appears to small, you can increase the scaling argument to make everything appear bigger at the same resolution.

snap_rect	Should axis-aligned rectangles drawn with only fill snap to the pixel grid. This will prevent anti-aliasing artifacts when two rectangles are touching at their border.
bg	Same as background for compatibility with old graphic device APIs

Examples

```
file <- tempfile(fileext = '.ppm')
agg_ppm(file)
plot(sin, -pi, 2*pi)
dev.off()
```

---

agg_record *Capture drawing instructions without rendering*

Description

While the point of a graphics device is usually to render the graphics, there are a few situations where you are instead interested in only capturing the instructions required to render the graphics. While all graphics devices can be retrofitted for that using dev.control(), they would still render to their internal buffer even if you are only interested in the recorded instructions, thus adding a performance penalty. agg_record() is a device that does no rendering whatsoever, but has recording turned on by default making it a no-overhead solution for plot recording.

Usage

```
agg_record(
  width = 480,
  height = 480,
  units = "px",
  pointsize = 12,
  background = "white",
  res = 72,
  scaling = 1,
  snap_rect = TRUE,
  bg
)
```

Arguments

width, height	The dimensions of the device
units	The unit width and height is measured in, in either pixels ('px'), inches ('in'), millimeters ('mm'), or centimeter ('cm').
pointsize	The default pointsize of the device in pt. This will in general not have any effect on grid graphics (including ggplot2) as text size is always set explicitly there.
background	The background colour of the device

res	The resolution of the device. This setting will govern how device dimensions given in inches, centimeters, or millimeters will be converted to pixels. Further, it will be used to scale text sizes and linewidths
scaling	A scaling factor to apply to the rendered line width and text size. Useful for getting the right dimensions at the resolution that you need. If e.g. you need to render a plot at 4000x3000 pixels for it to fit into a layout, but you find that the result appears to small, you can increase the scaling argument to make everything appear bigger at the same resolution.
snap_rect	Should axis-aligned rectangles drawn with only fill snap to the pixel grid. This will prevent anti-aliasing artifacts when two rectangles are touching at their border.
bg	Same as background for compatibility with old graphic device APIs

Examples

```
# Capture drawing instructions
agg_record()
plot(1:10, 1:10)
rec <- recordPlot()
dev.off()

# Replay these on another device
file <- tempfile(fileext = '.png')
agg_png(file)
replayPlot(rec)
dev.off()
```

---

agg_tiff *Draw to a TIFF file*

Description

The TIFF (Tagged Image File Format) format is a very versatile raster image storage format that supports 8 and 16bit colour mode, true transparency, as well as a range of other features not relevant to drawing from R (e.g. support for different colour spaces). The storage mode of the image data is not fixed and different compression modes are possible, in contrast to PNGs one-approach-fits-all. The default (uncompressed) will result in much larger files than PNG, and in general PNG is a better format for many of the graphic types produced in R. Still, TIFF has its purposes and sometimes this file format is explicetly requested.

Usage

```
agg_tiff(
  filename = "Rplot%03d.tiff",
  width = 480,
  height = 480,
```

```
  units = "px",
  pointsize = 12,
  background = "white",
  res = 72,
  scaling = 1,
  snap_rect = TRUE,
  compression = "none",
  bitsize = 8,
  bg
)
```

Arguments

`filename`	The name of the file. Follows the same semantics as the file naming in `grDevices::png()`, meaning that you can provide a `sprintf()` compliant string format to name multiple plots (such as the default value)
`width, height`	The dimensions of the device
`units`	The unit `width` and `height` is measured in, in either pixels (`'px'`), inches (`'in'`), millimeters (`'mm'`), or centimeter (`'cm'`).
`pointsize`	The default pointsize of the device in pt. This will in general not have any effect on grid graphics (including ggplot2) as text size is always set explicitly there.
`background`	The background colour of the device
`res`	The resolution of the device. This setting will govern how device dimensions given in inches, centimeters, or millimeters will be converted to pixels. Further, it will be used to scale text sizes and linewidths
`scaling`	A scaling factor to apply to the rendered line width and text size. Useful for getting the right dimensions at the resolution that you need. If e.g. you need to render a plot at 4000x3000 pixels for it to fit into a layout, but you find that the result appears to small, you can increase the `scaling` argument to make everything appear bigger at the same resolution.
`snap_rect`	Should axis-aligned rectangles drawn with only fill snap to the pixel grid. This will prevent anti-aliasing artifacts when two rectangles are touching at their border.
`compression`	The compression type to use for the image data. The standard options from the `grDevices::tiff()` function are available under the same name.
`bitsize`	Should the device record colour as 8 or 16bit
`bg`	Same as `background` for compatibility with old graphic device APIs

Transparency

TIFF have support for true transparency, meaning that the pixel colour is stored in pre-multiplied form. This is in contrast to pixels being stored in plain format, where the alpha values more function as a mask. The utility of this is not always that important, but it is one of the benefits of TIFF over PNG so it should be noted.

Note

'jpeg' compression is only available if ragg is compiled with a version of libtiff where jpeg support has been turned on.

Examples

```
file <- tempfile(fileext = '.tiff')
# Use jpeg compression
agg_tiff(file, compression = 'lzw+p')
plot(sin, -pi, 2*pi)
dev.off()
```

---

agg_webp *Draw to a WebP file*

Description

The WebP format is a raster image format that provides improved lossless (and lossy) compression for images on the web. Transparency is supported.

Usage

```
agg_webp(
  filename = "Rplot%03d.webp",
  width = 480,
  height = 480,
  units = "px",
  pointsize = 12,
  background = "white",
  res = 72,
  scaling = 1,
  snap_rect = TRUE,
  lossy = FALSE,
  quality = 80,
  bg
)
```

Arguments

filename	The name of the file. Follows the same semantics as the file naming in grDevices::png(), meaning that you can provide a sprintf() compliant string format to name multiple plots (such as the default value)
width, height	The dimensions of the device
units	The unit width and height is measured in, in either pixels ('px'), inches ('in'), millimeters ('mm'), or centimeter ('cm').

pointsize	The default pointsize of the device in pt. This will in general not have any effect on grid graphics (including ggplot2) as text size is always set explicitly there.
background	The background colour of the device
res	The resolution of the device. This setting will govern how device dimensions given in inches, centimeters, or millimeters will be converted to pixels. Further, it will be used to scale text sizes and linewidths
scaling	A scaling factor to apply to the rendered line width and text size. Useful for getting the right dimensions at the resolution that you need. If e.g. you need to render a plot at 4000x3000 pixels for it to fit into a layout, but you find that the result appears to small, you can increase the `scaling` argument to make everything appear bigger at the same resolution.
snap_rect	Should axis-aligned rectangles drawn with only fill snap to the pixel grid. This will prevent anti-aliasing artifacts when two rectangles are touching at their border.
lossy	Use lossy compression. Default is `FALSE`.
quality	An integer between `0` and `100` defining either the quality (if using lossy compression) or the compression effort (if using lossless).
bg	Same as `background` for compatibility with old graphic device APIs

Examples

```
file <- tempfile(fileext = '.webp')
agg_webp(file)
plot(sin, -pi, 2*pi)
dev.off()
```

`agg_webp_anim` *Draw an animation to a WebP file*

Description

The WebP format is a raster image format that provides improved lossless (and lossy) compression for images on the web. Transparency is supported.

Usage

```
agg_webp_anim(
  filename = "Ranim.webp",
  width = 480,
  height = 480,
  units = "px",
  pointsize = 12,
  background = "white",
  res = 72,
```

```
  scaling = 1,
  snap_rect = TRUE,
  lossy = FALSE,
  quality = 80,
  delay = 100L,
  loop = 0L,
  bg
)
```

Arguments

filename	The name of the file. This function does not perform page number substitution as the other devices since it cannot produce multiple pages.
width, height	The dimensions of the device
units	The unit width and height is measured in, in either pixels ('px'), inches ('in'), millimeters ('mm'), or centimeter ('cm').
pointsize	The default pointsize of the device in pt. This will in general not have any effect on grid graphics (including ggplot2) as text size is always set explicitly there.
background	The background colour of the device
res	The resolution of the device. This setting will govern how device dimensions given in inches, centimeters, or millimeters will be converted to pixels. Further, it will be used to scale text sizes and linewidths
scaling	A scaling factor to apply to the rendered line width and text size. Useful for getting the right dimensions at the resolution that you need. If e.g. you need to render a plot at 4000x3000 pixels for it to fit into a layout, but you find that the result appears to small, you can increase the scaling argument to make everything appear bigger at the same resolution.
snap_rect	Should axis-aligned rectangles drawn with only fill snap to the pixel grid. This will prevent anti-aliasing artifacts when two rectangles are touching at their border.
lossy	Use lossy compression. Default is FALSE.
quality	An integer between 0 and 100 defining either the quality (if using lossy compression) or the compression effort (if using lossless).
delay	Per-frame delay in milliseconds (single integer)
loop	Number of loops (0 = infinite)
bg	Same as background for compatibility with old graphic device APIs

See Also

agg_webp() for static WebP images

Examples

```
file <- tempfile(fileext = '.webp')
agg_webp_anim(file, delay = 100, loop = 0)
for(i in 1:10) {
```

```
  plot(sin(1:100 + i/10), type = 'l', ylim = c(-1, 1))
  dev.flush()
}
dev.off()
```

Index

agg_capture, 2
agg_jpeg, 4
agg_png, 5
agg_png(), *4*
agg_ppm, 7
agg_record, 8
agg_tiff, 9
agg_webp, 11
agg_webp(), *13*
agg_webp_anim, 12

dev.control(), *8*

grDevices::png(), *4–7*, *10*, *11*
grDevices::tiff(), *10*

sprintf(), *4*, *6*, *7*, *10*, *11*