

# Package ‘scattermore’

May 9, 2026

**Title** Scatterplots with More Points

**Version** 1.2

**Description** C-based conversion of large scatterplot data to rasters plus other operations such as data blurring or data alpha blending. Speeds up plotting of data with millions of points.

**Imports** ggplot2, scales, grid, grDevices, graphics

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Suggests** covr, knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Tereza Kulichova [aut],  
Mirek Kratochvil [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-7356-4075>>)

**Maintainer** Mirek Kratochvil <exa.exa@gmail.com>

**Repository** CRAN

**Date/Publication** 2023-06-12 08:30:02 UTC

## Contents

apply_kernel_histogram . . . . .	2
apply_kernel_rgbwt . . . . .	3
blend_rgba_float . . . . .	3
GeomScattermore . . . . .	4
GeomScattermost . . . . .	4
geom_scattermore . . . . .	5
geom_scattermost . . . . .	6
histogram_to_rgbwt . . . . .	7
merge_rgbwt . . . . .	7
rgba_float_to_rgba_int . . . . .	8

rgba_int_to_raster . . . . .	8
rgbwt_to_rgba_float . . . . .	9
rgbwt_to_rgba_int . . . . .	9
scattermore . . . . .	10
scattermoreplot . . . . .	11
scatter_lines_histogram . . . . .	12
scatter_lines_rgbwt . . . . .	13
scatter_points_histogram . . . . .	14
scatter_points_rgbwt . . . . .	14

## Index 16

---

apply\_kernel\_histogram  
*apply\_kernel\_histogram*

---

### Description

Apply a kernel to the given histogram.

### Usage

```
apply_kernel_histogram(
    fhistogram,
    filter = "circle",
    mask = default_kernel(filter, radius, sigma),
    radius = 2,
    sigma = radius/2,
    threads = 0
)
```

### Arguments

fhistogram	Matrix or array interpreted as histogram of floating-point values.
filter	Use the pre-defined filter, either circle, square, gauss. Defaults to circle.
mask	Custom kernel used for blurring, overrides filter. Must be a square matrix of odd size.
radius	Radius of the kernel (counted without the "middle" pixel), defaults to 2. The generated kernel matrix will be a square with (2*radius+1) pixels on each side.
sigma	Radius of the Gaussian function selected by filter, defaults to radius/2.
threads	Number of parallel threads (default 0 chooses hardware concurrency).

### Value

2D matrix with the histogram processed by the kernel application.

---

apply\_kernel\_rgbwt      *apply\_kernel\_rgbwt*

---

### Description

Apply a kernel to the given RGBWT raster.

### Usage

```
apply_kernel_rgbwt(
    fRGBWT,
    filter = "circle",
    mask = default_kernel(filter, radius, sigma),
    radius = 2,
    sigma = radius/2,
    threads = 0
)
```

### Arguments

fRGBWT	RGBWT array with channels red, green, blue, weight and transparency. The dimension should be N times M times 5.
filter	Use the pre-defined filter, either circle, square, gauss. Defaults to circle.
mask	Custom kernel used for blurring, overrides filter. Must be a square matrix of odd size.
radius	Radius of the kernel (counted without the "middle" pixel"), defaults to 2. The generated kernel matrix will be a square with (2*radius+1) pixels on each side.
sigma	Radius of the Gaussian function selected by filter, defaults to radius/2.
threads	Number of parallel threads (default 0 chooses hardware concurrency).

### Value

RGBWT matrix.

---

blend\_rgba\_float      *blend\_rgba\_float*

---

### Description

Blend RGBA matrices.

### Usage

```
blend_rgba_float(fRGBA_list)
```

**Arguments**

`fRGBA_list` List of floating-point RGBA arrays with premultiplied alpha (each of the same size N-by-M-by-4). The "first" matrix in the list is the one that will be rendered on "top".

**Value**

Blended RGBA matrix.

---

`GeomScattermore` *The actual geom for scattermore*

---

**Description**

The actual geom for scattermore

**Usage**

`GeomScattermore`

**Format**

An object of class `GeomScattermore` (inherits from `Geom`, `ggproto`, `gg`) of length 6.

---

`GeomScattermost` *The actual geom for scattermost*

---

**Description**

The actual geom for scattermost

**Usage**

`GeomScattermost`

**Format**

An object of class `GeomScattermost` (inherits from `Geom`, `ggproto`, `gg`) of length 4.

---

geom\_scattermore      *geom\_scattermore*

---

## Description

`ggplot2::ggplot()` integration. This cooperates with the rest of `ggplot` (so you can use it to e.g. add rasterized scatterplots to vector output in order to reduce PDF size). Note that the `ggplot` processing overhead still dominates the plotting time. Use `geom_scattermost()` to tradeoff some niceness and circumvent `ggplot` logic to gain speed.

## Usage

```
geom_scattermore(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "identity",  
  ...,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE,  
  interpolate = FALSE,  
  pointsize = 0,  
  pixels = c(512, 512)  
)
```

## Arguments

<code>mapping</code> , <code>data</code> , <code>stat</code> , <code>position</code> , <code>inherit.aes</code> , <code>show.legend</code> , ...	passed to <code>ggplot2::layer()</code>
<code>na.rm</code>	Remove NA values, just as with <code>ggplot2::geom_point()</code> .
<code>interpolate</code>	Default FALSE, passed to <code>grid::rasterGrob()</code> .
<code>pointsize</code>	Radius of rasterized point. Use 0 for single pixels (fastest).
<code>pixels</code>	Vector with X and Y resolution of the raster, default <code>c(512, 512)</code> .

## Details

Accepts aesthetics `x`, `y`, `colour` and `alpha`. Point size is fixed for all points. Due to rasterization properties it is often beneficial to try non-integer point sizes, e.g. 3.2 looks much better than 3.

## Examples

```
library(ggplot2)  
library(scattermore)  
ggplot(data.frame(x = rnorm(1e6), y = rexp(1e6))) +  
  geom_scattermore(aes(x, y, color = x),  
    pointsize = 3,
```

```

    alpha = 0.1,
    pixels = c(1000, 1000),
    interpolate = TRUE
  ) +
  scale_color_viridis_c()

```

---

```

geom_scattermost      geom_scattermost

```

---

## Description

Totally non-ggplotish version of [geom\\_scattermore\(\)](#), but faster. It avoids most of the ggplot processing by bypassing the largest portion of data around any ggplot functionality, leaving only enough data to set up axes and limits correctly. If you need to break speed records, use this.

## Usage

```

geom_scattermost(
  xy,
  color = "black",
  interpolate = FALSE,
  pointsize = 0,
  pixels = c(512, 512)
)

```

## Arguments

<code>xy</code>	2-column object with data, as in <a href="#">scattermore()</a> .
<code>color</code>	Color vector (or a single color).
<code>interpolate</code>	Default FALSE, passed to <a href="#">grid::rasterGrob()</a> .
<code>pointsize</code>	Radius of rasterized point. Use 0 for single pixels (fastest).
<code>pixels</code>	Vector with X and Y resolution of the raster, default <code>c(512, 512)</code> .

## Examples

```

library(ggplot2)
library(scattermore)
d <- data.frame(x = rnorm(1000000), y = rnorm(1000000))
x_rng <- range(d$x)
ggplot() +
  geom_scattermost(cbind(d$x, d$y),
    color = heat.colors(100, alpha = .01)
    [1 + 99 * (d$x - x_rng[1]) / diff(x_rng)],
    pointsize = 2.5,
    pixels = c(1000, 1000),
    interpolate = TRUE
  )

```

histogram\_to\_rgbwt      *histogram\_to\_rgbwt*

**Description**

Colorize given histogram with input palette.

**Usage**

```

histogram_to_rgbwt(
  fhistogram,
  RGBA = grDevices::col2rgb(col, alpha = T),
  col = grDevices::hcl.colors(10),
  zlim = c(min(fhistogram), max(fhistogram))
)

```

**Arguments**

fhistogram	Matrix or 2D array with the histogram of values.
RGBA	4-by-N matrix floating-point R, G, B and A channels for the palette. Overrides col.
col	Colors to use for coloring.
zlim	Values to use as extreme values of the histogram

**Value**

RGBWT matrix.

merge\_rgbwt      *merge\_rgbwt*

**Description**

Merge RGBWT matrices.

**Usage**

```

merge_rgbwt(fRGBWT_list)

```

**Arguments**

fRGBWT_list	List of RGBWT arrays. The order of the matrices does not matter (except for negligible floating-point rounding and other robustness errors).
-------------	--

**Value**

Merged RGBWT matrix.

---

rgba\_float\_to\_rgba\_int  
*rgba\_float\_to\_rgba\_int*

---

**Description**

Convert a float RGBA bitmap with pre-multiplied alpha to integer RGBA bitmap.

**Usage**

```
rgba_float_to_rgba_int(fRGBA)
```

**Arguments**

fRGBA            RGBA bitmap in N-by-M-by-4 array.

**Value**

RGBA matrix. The output *is not premultiplied* by alpha.

---

rgba\_int\_to\_raster    *rgba\_int\_to\_raster*

---

**Description**

Create a raster from the given RGBA matrix.

**Usage**

```
rgba_int_to_raster(i32RGBA)
```

**Arguments**

i32RGBA            Integer RGBA matrix (with all values between 0 and 255).

**Value**

The matrix converted to raster.

---

rgbwt\_to\_rgba\_float    *rgbwt\_to\_rgba\_float*

---

**Description**

Convert RGBWT matrix to floating-point RGBA matrix, suitable for alpha-blending.

**Usage**

```
rgbwt_to_rgba_float(fRGBWT)
```

**Arguments**

fRGBWT            The RGBWT matrix.

**Value**

RGBA matrix, output *is premultiplied* by alpha.

---

rgbwt\_to\_rgba\_int    *rgbwt\_to\_rgba\_int*

---

**Description**

Convert a RGBWT matrix to an integer RGBA matrix.

**Usage**

```
rgbwt_to_rgba_int(fRGBWT)
```

**Arguments**

fRGBWT            The RGBWT matrix.

**Value**

A RGBA matrix. The output *is not premultiplied* by alpha.

---

 scattermore

*scattermore*


---

## Description

Convert points to raster scatterplot rather quickly.

## Usage

```
scattermore(
  xy,
  size = c(512, 512),
  xlim = c(min(xy[, 1]), max(xy[, 1])),
  ylim = c(min(xy[, 2]), max(xy[, 2])),
  rgba = c(0L, 0L, 0L, 255L),
  cex = 0,
  output.raster = TRUE
)
```

## Arguments

<code>xy</code>	2-column float matrix with point coordinates. As usual with rasters in R, X axis grows right, and Y axis grows DOWN. Flipping <code>ylim</code> causes the usual mathematical behavior.
<code>size</code>	2-element vector integer size of the result raster, defaults to <code>c(512, 512)</code> .
<code>xlim, ylim</code>	Float limits as usual (position of the first pixel on the left/top, and the last pixel on the right/bottom). You can easily flip the top/bottom to the "usual" mathematical system by flipping the <code>ylim</code> vector.
<code>rgba</code>	4-row matrix with color values of 0-255, or just a single 4-item vector for <code>c(r, g, b, a)</code> . Best created with <code>col2rgb(..., alpha=TRUE)</code> .
<code>cex</code>	Additional point radius in pixels, 0=single-pixel dots (fastest)
<code>output.raster</code>	Output R-style raster (as.raster)? Default TRUE. Raw array output can be used much faster, e.g. for use with <code>png::writePNG</code> .

## Value

Raster with the result.

## Examples

```
library(scattermore)
plot(scattermore(cbind(rnorm(1e6), rnorm(1e6)), rgba = c(64, 128, 192, 10)))
```

---

scattermoreplot	scattermoreplot
-----------------	-----------------

---

## Description

Convenience base-graphics-like layer around scattermore. Currently only works with linear axes!

## Usage

```
scattermoreplot(
  x,
  y,
  xlim,
  ylim,
  size,
  col = grDevices::rgb(0, 0, 0, 1),
  cex = 0,
  pch = NULL,
  xlab,
  ylab,
  ...
)
```

## Arguments

x, y, xlim, ylim, xlab, ylab, ...	used as in <a href="#">graphics::plot()</a> or forwarded to <a href="#">graphics::plot()</a>
size	forwarded to <a href="#">scattermore()</a> , or auto-derived from device and plot size if missing (the estimate is not pixel-perfect on most devices, but gets pretty close)
col	point color(s)
cex	forwarded to <a href="#">scattermore()</a>
pch	ignored (to improve compatibility with <a href="#">graphics::plot()</a> )

## Examples

```
# plot an actual rainbow
library(scattermore)
d <- data.frame(s = qlogis(1:1e6 / (1e6 + 1), 6, 0.5), t = rnorm(1e6, pi / 2, 0.5))
scattermoreplot(
  d$s * cos(d$t),
  d$s * sin(d$t),
  col = rainbow(1e6, alpha = .05)[c((9e5 + 1):1e6, 1:9e5)],
  main = "scattermore demo"
)
```

---

```
scatter_lines_histogram  
    scatter_lines_histogram
```

---

### Description

Render lines into a histogram.

### Usage

```
scatter_lines_histogram(  
    xy,  
    xlim = c(min(xy[, c(1, 3)]), max(xy[, c(1, 3)])),  
    ylim = c(min(xy[, c(2, 4)]), max(xy[, c(2, 4)])),  
    out_size = c(512L, 512L),  
    skip_start_pixel = FALSE,  
    skip_end_pixel = TRUE  
)
```

### Arguments

<code>xy</code>	4-column matrix with point coordinates. Each row contains X and Y coordinates of line start and X and Y coordinates of line end, in this order.
<code>xlim, ylim</code>	2-element vector of rendered area limits (position of the first pixel on the left/top, and the last pixel on the right/bottom). You can flip the image coordinate system by flipping the <code>*lim</code> vectors.
<code>out_size</code>	2-element vector size of the result raster, defaults to <code>c(512L, 512L)</code> .
<code>skip_start_pixel</code>	TRUE if the start pixel of the lines should be omitted, defaults to FALSE.
<code>skip_end_pixel</code>	TRUE if the end pixel of a line should be omitted, defaults to TRUE. (When plotting long ribbons of connected lines, this prevents counting the connecting pixels twice.)

### Value

Histogram with the rendered lines.

---

 scatter\_lines\_rgbwt    *scatter\_lines\_rgbwt*


---

### Description

Render lines into a RGBWT bitmap.

### Usage

```
scatter_lines_rgbwt(
  xy,
  xlim = c(min(xy[, c(1, 3)]), max(xy[, c(1, 3)])),
  ylim = c(min(xy[, c(2, 4)]), max(xy[, c(2, 4)])),
  out_size = c(512L, 512L),
  RGBA = c(0, 0, 0, 255),
  skip_start_pixel = FALSE,
  skip_end_pixel = TRUE
)
```

### Arguments

<code>xy</code>	4-column matrix with point coordinates. Each row contains X and Y coordinates of line start and X and Y coordinates of line end, in this order.
<code>xlim, ylim</code>	2-element vector of rendered area limits (position of the first pixel on the left/top, and the last pixel on the right/bottom). You can flip the image coordinate system by flipping the <code>*lim</code> vectors.
<code>out_size</code>	2-element vector size of the result raster, defaults to <code>c(512L, 512L)</code> .
<code>RGBA</code>	Vector of 4 elements with integral RGBA color for the lines, defaults to <code>c(0, 0, 0, 255)</code> .
<code>skip_start_pixel</code>	TRUE if the start pixel of the lines should be omitted, defaults to FALSE.
<code>skip_end_pixel</code>	TRUE if the end pixel of a line should be omitted, defaults to TRUE. (When plotting long ribbons of connected lines, this prevents counting the connecting pixels twice.)

### Value

Lines plotted in RGBWT bitmap.

---

```
scatter_points_histogram
    scatter_points_histogram
```

---

**Description**

Render a 2D histogram with given points

**Usage**

```
scatter_points_histogram(
    xy,
    xlim = c(min(xy[, 1]), max(xy[, 1])),
    ylim = c(min(xy[, 2]), max(xy[, 2])),
    out_size = c(512L, 512L)
)
```

**Arguments**

<code>xy</code>	2-column matrix with point coordinates (X and Y).
<code>xlim, ylim</code>	2-element vector of rendered area limits (position of the first pixel on the left/top, and the last pixel on the right/bottom). You can flip the image coordinate system by flipping the <code>*lim</code> vectors.
<code>out_size</code>	2-element vector size of the result raster, defaults to <code>c(512L, 512L)</code> .

**Value**

2D histogram with the points "counted" in appropriate pixels.

---

```
scatter_points_rgbwt  scatter_points_rgbwt
```

---

**Description**

Render colored points into a RGBWT bitmap

**Usage**

```
scatter_points_rgbwt(
    xy,
    xlim = c(min(xy[, 1]), max(xy[, 1])),
    ylim = c(min(xy[, 2]), max(xy[, 2])),
    out_size = c(512, 512),
    RGBA = c(0, 0, 0, 255),
    map = NULL,
    palette = NULL
)
```

**Arguments**

<code>xy</code>	2-column matrix with N point coordinates (X and Y) in rows.
<code>xlim, ylim</code>	2-element vector of rendered area limits (position of the first pixel on the left/top, and the last pixel on the right/bottom). You can flip the image coordinate system by flipping the <code>*lim</code> vectors.
<code>out_size</code>	2-element vector size of the result raster, defaults to <code>c(512L, 512L)</code> .
<code>RGBA</code>	Point colors. Either a 4-element vector that specifies the same color for all points, or 4-by-N matrix that specifies color for each of the individual points. Color is specified using integer RGBA; i.e. the default black is <code>c(0, 0, 0, 255)</code> .
<code>map</code>	Vector with N integer indices to <code>palette</code> . Overrides RGBA-based coloring.
<code>palette</code>	Matrix 4-by-K matrix of RGBA colors used as a palette lookup for the <code>map</code> that gives the point colors. K is at least <code>max(map)</code> . Notably, using a palette may be faster than filling and processing the whole RGBA matrix.

**Value**

A RGBWT array with the rendered points.

# Index

## \* datasets

GeomScattermore, [4](#)

GeomScattermost, [4](#)

apply\_kernel\_histogram, [2](#)

apply\_kernel\_rgbwt, [3](#)

blend\_rgba\_float, [3](#)

geom\_scattermore, [5](#)

geom\_scattermore(), [6](#)

geom\_scattermost, [6](#)

geom\_scattermost(), [5](#)

GeomScattermore, [4](#)

GeomScattermost, [4](#)

ggplot2::geom\_point(), [5](#)

ggplot2::ggplot(), [5](#)

ggplot2::layer(), [5](#)

graphics::plot(), [11](#)

grid::rasterGrob(), [5, 6](#)

histogram\_to\_rgbwt, [7](#)

merge\_rgbwt, [7](#)

rgba\_float\_to\_rgba\_int, [8](#)

rgba\_int\_to\_raster, [8](#)

rgbwt\_to\_rgba\_float, [9](#)

rgbwt\_to\_rgba\_int, [9](#)

scatter\_lines\_histogram, [12](#)

scatter\_lines\_rgbwt, [13](#)

scatter\_points\_histogram, [14](#)

scatter\_points\_rgbwt, [14](#)

scattermore, [10](#)

scattermore(), [6, 11](#)

scattermoreplot, [11](#)