Package 'aRtsy'

September 26, 2021

Title Generative Art with 'ggplot2'

Description Provides algorithms for creating artworks in the 'ggplot2' language that incorporate some form of randomness.

Version 0.1.3

Date 2021-09-26

BugReports https://github.com/koenderks/aRtsy/issues

URL https://koenderks.github.io/aRtsy/,
https://github.com/koenderks/aRtsy,
https://twitter.com/aRtsy_package

Imports dplyr, e1071, ggplot2, kknn, randomForest, Rcpp, stats

LinkingTo Rcpp, RcppArmadillo

Language en-US

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Koen Derks [aut, cre] (<https://orcid.org/0000-0002-5533-9349>)

Maintainer Koen Derks <koen-derks@hotmail.com>

Repository CRAN

Date/Publication 2021-09-26 15:10:08 UTC

R topics documented:

`aRtsy-package` *aRtsy — Generative Art using* `ggplot2`

Description

`aRtsy` is an attempt at making generative art available for the masses in a simple and standardized format. The package provides various algorithms for creating artworks in ggplot2 that incorporate some form of randomness (depending on the set `seed`). Each type of artwork is implemented in a separate function.

For documentation on `aRtsy` itself, including the manual and user guide for the package, worked examples, and other tutorial information visit the package website.

Author(s)

Koen Derks (maintainer, author) <koen-derks@hotmail.com>

Please use the citation provided by R when citing this package. A BibTex entry is available from `citation("aRtsy")`.

See Also

Useful links:

- The twitter feed to check the artwork of the day.
- The issue page to submit a bug report or feature request.

`canvas_ant` *Draw Langton's Ant*

Description

This function draws Langton's Ant on a canvas. Langton's ant is a two-dimensional universal Turing machine with a very simple set of rules. These simple rules can lead to complex emergent behavior.

Usage

```
canvas_ant(colors, background = "#fafafa", iterations = 50000,
           resolution = 500)
```

Arguments

`colors`	a character (vector) specifying the color(s) used for the artwork.
`background`	a character specifying the color used for the background.
`iterations`	a positive integer specifying the number of iterations of the algorithm.
`resolution`	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Details

The algorithm for Langton's Ant involves repeating the following rules: 1) on a non-colored block: turn 90 degrees clockwise, un-color the block, move forward one block; 2) On a colored block: turn 90 degrees counter-clockwise, color the block, move forward one block; 3) If a certain number of iterations has passed, choose a different color which corresponds to a different combination of these rules.

Value

A `ggplot` object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://en.wikipedia.org/wiki/Langtons_ant

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_ant(colors = colorPalette("house"))
```

---

canvas_blacklight *Draw Blacklights*

Description

This function draws the predictions from a support vector machine algorithm trained on randomly generated continuous data.

Usage

```
canvas_blacklight(colors, n = 1000, resolution = 500)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
n	a positive integer specifying the number of random data points to generate.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://en.wikipedia.org/wiki/Support-vector_machine

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_blacklight(colors = colorPalette("tuscany2"))
```

`canvas_circlemap` *Draw a Circle Map*

Description

This function draws a circle map on the canvas. A circle map models the dynamics of a physical system consisting of two rotors or disks, one free to spin, and another one attached to a motor, with a long (weak) spring connecting the two.

Usage

```
canvas_circlemap(colors, left = 0, right = 12.56, bottom = 0, top = 1,
                 iterations = 10, resolution = 1500)
```

Arguments

`colors`	a string or character vector specifying the color(s) used for the artwork.
`left`	a value specifying the minimum location on the x-axis.
`right`	a value specifying the maximum location on the x-axis.
`bottom`	a value specifying the minimum location on the y-axis.
`top`	a value specifying the maximum location on the y-axis.
`iterations`	a positive integer specifying the number of iterations of the algorithm.
`resolution`	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Value

A `ggplot` object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://en.wikipedia.org/wiki/Arnold_tongue

https://linas.org/art-gallery/circle-map/circle-map.html

See Also

colorPalette

Examples

```
canvas_circlemap(colors = colorPalette("dark2"))
```

---

canvas_collatz *Draw Collatz Sequences*

Description

This function draws the Collatz conjecture on the canvas.

Usage

```
canvas_collatz(colors, background = "#fafafa", n = 200,
               angle.even = 0.0075, angle.odd = 0.0145, side = FALSE)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
background	a character specifying the color used for the background.
n	a positive integer specifying the number of random starting integers to use for the lines. Can also be a vector of numbers to use as starting numbers.
angle.even	a value specifying the angle (in radials) to use in bending the sequence at each odd number.
angle.odd	a value specifying the angle (in radials) to use in bending the sequence at each even number.
side	logical. Whether to put the artwork on its side.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://nl.wikipedia.org/wiki/Collatz_Conjecture

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_collatz(colors = colorPalette("tuscany3"))
```

---

canvas_diamonds *Draw Diamonds*

Description

This function draws diamonds on a canvas and (optionally) places two lines behind them. The diamonds can be transparent or have a random color sampled from the input.

Usage

```
canvas_diamonds(colors, background = "#fafafa", col.line = "black",
                radius = 10, alpha = 1, p = 0.2, resolution = 500)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
background	a character specifying the color used for the background.
col.line	a character specifying the color of the diamond borders.
radius	a positive value specifying the radius of the diamonds.
alpha	a value specifying the transparency of the diamonds. If NULL (the default), added layers become increasingly more transparent.
p	a value specifying the probability of drawing an empty diamond.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_diamonds(colors = colorPalette("tuscany1"))
```

---

canvas_flow *Draw A Flow Field*

Description

This function draws flow fields on a canvas.

Usage

```
canvas_flow(colors, background = "#fafafa", lines = 500, lwd = 0.05,
            iterations = 100, resolution = 100, angles = NULL)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
background	a character specifying the color used for the background.
lines	the number of lines to draw.
lwd	expansion factor for the line width.
iterations	the maximum number of iterations for each line.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.
angles	optional, a matrix containing the angles of the flow field . If NULL (default), angles are set according to the predictions of a supervised learning algorithm.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://tylerxhobbs.com/essays/2020/flow-fields

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_flow(colors = colorPalette("dark2"))

# Advanced example
angles <- matrix(rnorm(200 * 200), nrow = 200, ncol = 200)
canvas_flow(colors = colorPalette("tuscany1"), angles = angles)
```

---

canvas_forest *Draw a Random Forest*

Description

This function draws the predictions from a random forest algorithm trained on randomly generated categorical data.

Usage

```
canvas_forest(colors, n = 1000, resolution = 500)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
n	a positive integer specifying the number of random data points to generate.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://en.wikipedia.org/wiki/Random_forest

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_forest(colors = colorPalette("jungle"))
```

---

canvas_function *Draw Functions*

Description

This function paints functions with random parameters on a canvas.

Usage

```
canvas_function(color, background = "#fafafa", formula = NULL)
```

Arguments

color	a string specifying the color used for the artwork.
background	a character specifying the color used for the background.
formula	optional, a named list with 'x"and 'y' as structured in the example. If NULL (default), chooses a function with random parameters.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://github.com/cutterkom/generativeart

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_function(color = "navyblue")

# Advanced example
formula <- list(
  x = quote(x_i^2 - sin(y_i^2)),
  y = quote(y_i^3 - cos(x_i^2))
)
canvas_function(color = "firebrick", formula = formula)
```

---

canvas_gemstone *Draw Gemstones*

Description

This function draws the predictions from a k-nearest neighbors algorithm trained on randomly generated continuous data.

Usage

```
canvas_gemstone(colors, n = 1000, resolution = 500)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
n	a positive integer specifying the number of random data points to generate.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_gemstone(colors = colorPalette("dark3"))
```

---

canvas_mandelbrot *Draw the Mandelbrot Set*

Description

This function draws the Mandelbrot set on the canvas.

Usage

```
canvas_mandelbrot(colors, iterations = 100, zoom = 1, left = -1.7, right = -0.2,
                  bottom = -0.2999, top = 0.8001, resolution = 500)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
iterations	a positive integer specifying the number of iterations of the algorithm.
zoom	a positive value specifying the amount of zoom to apply.
left	a value specifying the minimum location on the x-axis.
right	a value specifying the maximum location on the x-axis.
bottom	a value specifying the minimum location on the y-axis.
top	a value specifying the maximum location on the y-axis.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://en.wikipedia.org/wiki/Mandelbrot_set

See Also

colorPalette

Examples

```
canvas_mandelbrot(colors = colorPalette("tuscany1"))
```

---

canvas_mosaic *Draw Moisaics*

Description

This function draws the predictions from a k-nearest neighbors algorithm trained on randomly generated categorical data.

Usage

```
canvas_mosaic(colors, n = 1000, resolution = 500)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
n	a positive integer specifying the number of random data points to generate.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_mosaic(colors = colorPalette("retro2"))
```

---

canvas_nebula *Draw Nebulas*

Description

This function creates an artwork from randomly generated k-nearest neighbors noise.

Usage

```
canvas_nebula(colors, k = 50, n = 500, resolution = 500)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
k	a positive integer specifying the number of nearest neighbors to consider.
n	a positive integer specifying the number of random data points to generate.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_nebula(colors = colorPalette("tuscany1"))
```

---

canvas_planet *Draw Planets*

Description

This function paints one or multiple planets and uses a cellular automata to fill their surfaces.

Usage

```
canvas_planet(colors, threshold = 4, iterations = 200,
              starprob = 0.01, fade = 0.2,
              radius = NULL, center.x = NULL, center.y = NULL,
              light.right = TRUE, resolution = 1500)
```

Arguments

colors	a character specifying the colors used for a single planet. Can also be a list where each entry is a vector of colors for a planet.
threshold	a character specifying the threshold for a color take.
iterations	a positive integer specifying the number of iterations of the algorithm.
starprob	a value specifying the probability of drawing a star in outer space.
fade	a value specifying the amount of fading to apply.
radius	a numeric (vector) specifying the radius of the planet(s).
center.x	the x-axis coordinate(s) for the center(s) of the planet(s).
center.y	the y-axis coordinate(s) for the center(s) of the planet(s).
light.right	whether to draw the light from the right or the left.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://fronkonstin.com/2021/01/02/neighborhoods-experimenting-with-cyclic-cellular-automata/

Examples

```
set.seed(1)

# Simple example
canvas_planet(colors = colorPalette("retro3"))

# Advanced example
colors <- list(
  c("khaki1", "lightcoral", "lightsalmon"),
  c("dodgerblue", "forestgreen", "white"),
  c("gray", "darkgray", "beige")
)
canvas_planet(colors,
  radius = c(800, 400, 150),
  center.x = c(1, 500, 1100),
  center.y = c(1400, 500, 1000),
  starprob = 0.005
)
```

---

canvas_polylines *Draw Polygons and Lines*

Description

This function draws many points on the canvas and connects these points into a polygon. After repeating this for all the colors, the edges of all polygons are drawn on top of the artwork.

Usage

```
canvas_polylines(colors, background = "#fafafa", ratio = 0.5, iterations = 1000,
                 size = 0.1, resolution = 500)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
background	a character specifying the color used for the lines.
ratio	a positive value specifying the width of the polygons. Larger ratios cause more overlap.
iterations	a positive integer specifying the number of iterations of the algorithm.
size	a positive value specifying the size of the borders.

resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_polylines(colors = colorPalette("retro1"))
```

---

canvas_ribbons *Draw Ribbons*

Description

This function paints random ribbons and (optionally) a triangle in the middle.

Usage

```
canvas_ribbons(colors, background = "#fdf5e6", triangle = TRUE)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork. The number of colors determines the number of ribbons.
background	a character specifying the color of the background.
triangle	logical. Whether to draw the triangle that breaks the ribbon polygons.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_ribbons(colors = colorPalette("retro1"))
```

---

canvas_segments *Draw Segments*

Description

This function draws line segments on a canvas. The length and direction of the line segments is determined randomly.

Usage

```
canvas_segments(colors, background = "#fafafa", n = 250,
                p = 0.5, H = 0.1, size = 0.2)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
background	a character specifying the color used for the background.
n	a positive integer specifying the number of line segments to draw.
p	a value specifying the probability of drawing a vertical line segment.
H	a positive value specifying the scaling factor for the line segments.
size	a positive value specifying the size of the line segments.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_segments(colors = colorPalette("dark1"))
```

---

canvas_squares *Draw Squares and Rectangles*

Description

This function paints random squares and rectangles. It works by repeatedly cutting into the canvas at random locations and coloring the area that these cuts create.

Usage

```
canvas_squares(colors, background = "#000000", cuts = 50, ratio = 1.618,
               resolution = 200, noise = FALSE)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
background	a character specifying the color used for the borders of the squares.
cuts	a positive integer specifying the number of cuts to make.
ratio	a value specifying the 1:1 ratio for each cut.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.
noise	logical. Whether to add k-nn noise to the artwork. Note that adding noise increases computation time significantly in large dimensions.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_squares(colors = colorPalette("retro2"))
```

---

canvas_stripes *Draw Stripes*

Description

This function creates a brownian motion on each row of the artwork and colors it according to the height of the motion.

Usage

```
canvas_stripes(colors, n = 300, H = 1, burnin = 1)
```

Arguments

colors	a string or character vector specifying the color(s) used for the artwork.
n	a positive integer specifying the length of the brownian motion (effectively the width of the artwork).
H	a positive value specifying the square of the standard deviation of each step in the motion.
burnin	a positive integer specifying the number of steps to discard before filling each row.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_stripes(colors = colorPalette("random", n = 10))
```

`canvas_strokes` *Draw Strokes*

Description

This function creates an artwork that resembles paints strokes. The algorithm is based on the simple idea that each next point on the grid has a chance to take over the color of an adjacent colored point but also has a change of generating a new color.

Usage

```
canvas_strokes(colors, neighbors = 1, p = 0.01, iterations = 1,
               resolution = 500, side = FALSE)
```

Arguments

`colors`	a string or character vector specifying the color(s) used for the artwork.
`neighbors`	a positive integer specifying the number of neighbors a block considers when taking over a color. More neighbors fades the artwork.
`p`	a value specifying the probability of selecting a new color at each block. A higher probability adds more noise to the artwork.
`iterations`	a positive integer specifying the number of iterations of the algorithm. More iterations generally apply more fade to the artwork.
`resolution`	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.
`side`	logical. Whether to put the artwork on its side.

Value

A `ggplot` object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_strokes(colors = colorPalette("tuscany1"))
```

---

canvas_turmite *Draw Turmites*

Description

This function paints a turmite. A turmite is a Turing machine which has an orientation in addition to a current state and a "tape" that consists of a two-dimensional grid of cells.

Usage

```
canvas_turmite(colors, background = "#fafafa", p = 0.5, iterations = 1e6,
               resolution = 500, noise = FALSE)
```

Arguments

colors	a character specifying the color used for the artwork. The number of colors determines the number of turmites.
background	a character specifying the color used for the background.
p	a value specifying the probability of a state switch within the turmite.
iterations	a positive integer specifying the number of iterations of the algorithm.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.
noise	logical. Whether to add k-nn noise to the artwork. Note that adding noise increases computation time significantly in large dimensions.

Details

The turmite algorithm consists of the following steps: 1) turn on the spot (left, right, up, down) 2) change the color of the square 3) move forward one square.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://en.wikipedia.org/wiki/Turmite

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_turmite(colors = colorPalette("dark2"))
```

---

canvas_watercolors *Draw Watercolors*

Description

This function paints watercolors on a canvas.

Usage

```
canvas_watercolors(colors, background = "#fafafa", layers = 50,
                   depth = 2, resolution = 250)
```

Arguments

colors	a string specifying the color used for the artwork.
background	a character specifying the color used for the background.
layers	the number of layers of each color.
depth	the maximum depth of the recursive algorithm.
resolution	resolution of the artwork in pixels per row/column. Increasing the resolution increases the quality of the artwork but also increases the computation time exponentially.

Value

A ggplot object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

References

https://tylerxhobbs.com/essays/2017/a-generative-approach-to-simulating-watercolor-paints

See Also

colorPalette

Examples

```
set.seed(1)

# Simple example
canvas_watercolors(colors = colorPalette("tuscany2"))
```

---

colorPalette *Color Palette Generator*

Description

This function creates a random color palette, or allows the user to select a pre-implemented palette.

Usage

```
colorPalette(name, n = NULL)
```

Arguments

name	name of the color palette. Can be random for random colors, but can also be the name of a pre-implemented palette. See the details section for a list of pre-implemented palettes.
n	the number of colors to select from the palette. Required if name = 'random'. Otherwise, if NULL, automatically selects all colors from the chosen palette.

Details

The following color palettes are implemented:

Value

A vector of colors.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

Examples

```
colorPalette("random", 5)
```

---

saveCanvas *Save a Canvas to an External Device*

Description

This function is a wrapper around `ggplot2::ggsave`. It provides a suggested export with square dimensions for a canvas created using the `aRtsy` package.

Usage

```
saveCanvas(plot, filename, width = 7, height = 7, dpi = 300)
```

Arguments

`plot`	a ggplot2 object to be saved.
`filename`	the filename of the export.
`width`	the width of the artwork in cm.
`height`	the height of the artwork in cm.
`dpi`	the `dpi` (dots per inch) of the file.

Value

No return value, called for saving plots.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

---

theme_canvas *Canvas Theme for ggplot2 Objects*

Description

Add a canvas theme to the plot. The canvas theme by default has no margins and fills any empty canvas with a background color.

Usage

```
theme_canvas(x, background = NULL, margin = 0)
```

Arguments

`x`	a ggplot2 object.
`background`	a character specifying the color used for the empty canvas.
`margin`	margins of the canvas.

Value

A `ggplot` object containing the artwork.

Author(s)

Koen Derks, <koen-derks@hotmail.com>

Index