Package ‘aRtsy’

November 26, 2021

Title  Generative Art with ‘ggplot2’
Description  Provides algorithms for creating artworks in the ‘ggplot2’ language that incorporate some form of randomness.
Version  0.1.5
Date  2021-11-26
BugReports  https://github.com/koenderks/aRtsy/issues
URL  https://koenderks.github.io/aRtsy/,
    https://github.com/koenderks/aRtsy,
    https://twitter.com/aRtsy_package
Imports  e1071, ggplot2, kknn, randomForest, Rcpp, stats
LinkingTo  Rcpp, RcppArmadillo
Language  en-US
License  GPL-3
Encoding  UTF-8
RoxygenNote  7.1.2
Suggests  testthat (>= 3.0.0)
Config/testthat/edition  3
NeedsCompilation  yes
Author  Koen Derks [aut, cre]
Maintainer  Koen Derks <koen-derks@hotmail.com>
Repository  CRAN
Date/Publication  2021-11-26 15:50:07 UTC

R topics documented:

  aRtsy-package .................................................. 2
  canvas_ant ..................................................... 3
  canvas_blacklight .......................................... 4
aRtsy — Generative Art using ggplot2

Description

aRtsy aims to make generative art accessible to the general public in a straightforward and standardized manner. The package provides algorithms for creating artworks that incorporate some form of randomness and are dependent on the set seed. Each algorithm is implemented in a separate function with its own set of parameters that can be tweaked.

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

\textit{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

- \texttt{colors}: a character (vector) specifying the color(s) used for the artwork.
- \texttt{background}: a character specifying the color used for the background.
- \texttt{iterations}: a positive integer specifying the number of iterations of the algorithm.
- \texttt{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

canvas_chladni

See Also

colorPalette

Examples

set.seed(1)

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

Description

This function draws Chladni figures on a canvas.

Usage

canvas_chladni(colors, waves = 5, resolution = 500)

Arguments

colors a string or character vector specifying the color(s) used for the artwork.
waves a character specifying the number of randomly sampled waves, or an integer vector of waves to be summed.
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

```r
set.seed(1)
# Simple example
canvas_chladni(colors = colorPalette("lava"))

# Advanced example
canvas_chladni(colors = colorPalette("lava"), waves = c(1, 2, 3, 9))
```

describe canvas_circlemap

**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**

```r
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>
canvas_cobweb

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"))

Description

This function draws many Fibonacci spirals shifted by random noise from a normal distribution.

Usage

canvas_cobweb(colors, background = "#fafafa", lines = 300,
iterations = 100)

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.
iterations the number of iterations of the algorithm.

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_cobweb(colors = colorPalette("neon1"), background = "black")

---

**canvas_collatz**

*Draw Collatz Sequences*

**Description**

This function draws the Collatz conjecture on the canvas.

**Usage**

```r
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
canvas_diamonds

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 = "#fafa", 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>
canvas_flow

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. The algorithm simulates the flow of points through a field of angles which can be set manually or generated from the predictions of a supervised learning method (i.e., knn, svm, random forest) trained on randomly generated data.

Usage

canvas_flow(colors, background = "#fafafa", lines = 500, lwd = 0.05,
            iterations = 100, stepmax = 0.01, 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.
stepmax | the maximum proportion of the canvas covered in each iteration.
angles | optional, a 200 x 200 matrix containing the angles in 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>
### canvas_forest

**Draw a Random Forest**

#### Description

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

#### Usage

```r
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.

---

#### Examples

```r
set.seed(1)

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

# Advanced example
angles <- matrix(0, 200, 200)
angles[1:100, ] <- seq(from = 0, to = 2 * pi, length = 100)
angles[101:200, ] <- seq(from = 2 * pi, to = 0, length = 100)
angles <- angles + rnorm(200 * 200, sd = 0.1)
canvas_flow(colors = colorPalette("tuscany1"), background = "black",
  angles = angles, lwd = 0.4, lines = 1000, stepmax = 0.001)
```

---

### References


### See Also

colorPalette
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

```r
set.seed(1)

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

Description

This function paints functions with random parameters on a canvas.

Usage

```r
canvas_function(colors, background = "#fafafa", by = 0.01,
polar = TRUE, formula = NULL)
```

Arguments

- `colors`: a string specifying the color used for the artwork.
- `background`: a character specifying the color used for the background.
- `by`: a value specifying the step size between consecutive points.
- `polar`: logical. Whether to draw the function with polar coordinates.
- `formula`: optional, a named list with ‘x’ and ‘y’ as structured in the example. If NULL (default), chooses a function with random parameters.
canvas_gemstone

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(10)

# Simple example
canvas_function(colors = colorPalette("tuscany1"))

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


See Also

colorPalette

Examples

```r
set.seed(1)

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

```
canvas_mandelbrot  Draw the Mandelbrot Set
```

Description

This function draws the Mandelbrot set on the canvas.

Usage

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

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_maze

Description

This function draws a maze on a canvas.

Usage

canvas_maze(color = "#fafe", walls = "black", background = "#fafe", resolution = 20, polar = FALSE)
canvas_mosaic

Arguments

- **color**: a character specifying the color used for the artwork.
- **walls**: a character specifying the color used for the walls of the maze.
- **background**: a character specifying the color used for the background.
- **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.
- **polar**: logical, whether to use polar coordinates. Warning, this increases display and saving time dramatically.

Value

A ggplot object containing the artwork.

Author(s)

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

References

https://github.com/matfmc/mazegenerator

See Also

colorPalette

Examples

```r
set.seed(1)

# Simple example
canvas_maze(color = "#fafafa")
```

Description

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

Usage

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

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


See Also

colorPalette

Examples

```r
set.seed(1)

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

**Description**

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

**Usage**

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

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_phyllotaxis(colors, background = '#fafafa', iterations = 10000, angle = 137.5, size = 0.01, alpha = 1, p = 0.5)

Description

This function draws a phyllotaxis which resembles the arrangement of leaves on a plant stem.

Usage

canvas PHYLLOTAXIS(colors, background = '#fafafa', iterations = 10000, angle = 137.5, size = 0.01, alpha = 1, p = 0.5)
canvas_planet

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.
- **iterations**: the number of iterations of the algorithm.
- **angle**: the angle at which to place the artwork.
- **size**: the size of the lines.
- **alpha**: transparency of the points.
- **p**: probability of drawing a point on each iteration.

Value

A ggplot object containing the artwork.

Author(s)

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

References

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

See Also

colorPalette

Examples

```r
set.seed(1)

# Simple example
canvas_phyllotaxis(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

```r
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  

**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**

```r
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**

```r
set.seed(1)

# Simple example
canvas_polylines(colors = colorPalette("retro1"))
```
canvas_recaman  Draw Recaman's Sequence

Description
This function draws Recaman’s sequence on a canvas. The algorithm takes increasingly large steps backward on the positive number line, but if it is unable to it takes a step forward.

Usage
```
canvas_recaman(colors, background = "#fafafa", iterations = 100, start = 0,
              increment = 1, curvature = 1, angle = 0, size = 0.1,
              closed = 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.
- `iterations` the number of iterations of the algorithm.
- `start` the starting point of the algorithm.
- `increment` the increment of each step.
- `curvature` the curvature of each line.
- `angle` the angle at which to place the artwork.
- `size` the size of the lines.
- `closed` logical. Whether to plot a curve from the end of the sequence back to the starting point.

Value
A ggplot object containing the artwork.

Author(s)
Koen Derks, <koen-derks@hotmail.com>

References
https://mathworld.wolfram.com/RecamansSequence.html

See Also
colorPalette
Examples

```r
set.seed(1)

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

---

canvas_ribbons | Draw Ribbons

Description

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

Usage

```r
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

```r
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**

```r
canvas_segments(colors, background = "#fafa", 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**

```r
set.seed(1)
# Simple example
canvas_segments(colors = colorPalette("dark1"))
```
canvas_squares  

**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**

```r
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**

```r
set.seed(1)

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

\textit{Draw Stripes}

\textbf{Description}

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

\textbf{Usage}

\texttt{canvas\_stripes(colors, n = 300, H = 1, burnin = 1)}

\textbf{Arguments}

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

\textbf{Value}

A \texttt{ggplot} object containing the artwork.

\textbf{Author(s)}

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

\textbf{See Also}

\texttt{colorPalette}

\textbf{Examples}

\begin{verbatim}
set.seed(1)

# Simple example
canvas_stripes(colors = colorPalette("random", n = 10))
\end{verbatim}
canvas_strokes

Draw Strokes

Description

This function creates an artwork that resembles paint 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 chance 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

```r
set.seed(1)

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

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

```r
canvas_turmite(colors, background = "#fafa", 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>
canvas_watercolors

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

See Also
colorPalette

Examples

set.seed(1)

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

---

**Description**

This function paints watercolors on a canvas.

**Usage**

```r
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>
colorPalette

References


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, complement for complementing colors, divergent for equally spaced colors, or random-palette for a random palette, 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', name = 'complement', or name = 'divergent'. 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("divergent", 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>

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

* aRtsy
  aRtsy-package, 2
* artwork
  canvas_ant, 3
canvas_blacklight, 4
canvas_chladni, 5
canvas_circlemap, 6
canvas_cobweb, 7
canvas_collatz, 8
canvas_diamonds, 9
canvas_flow, 10
canvas_forest, 11
canvas_function, 12
canvas_gemstone, 13
canvas_mandelbrot, 14
canvas_maze, 15
canvas_mosaic, 16
canvas_nebula, 17
canvas_phyllotaxis, 18
canvas_planet, 19
canvas_polylines, 21
canvas_recaman, 22
canvas_ribbons, 23
canvas_segments, 24
canvas_squares, 25
canvas_stripes, 26
canvas_strokes, 27
canvas_turmite, 28
canvas_watercolors, 29
canvas_function, 12
canvas_gemstone, 13
canvas_mandelbrot, 14
canvas_maze, 15
canvas_mosaic, 16
canvas_nebula, 17
canvas_phyllotaxis, 18
canvas_planet, 19
canvas_polylines, 21
canvas_recaman, 22
canvas_ribbons, 23
canvas_segments, 24
canvas_squares, 25
canvas_stripes, 26
canvas_strokes, 27
canvas_turmite, 28
canvas_watercolors, 29

* package
  aRtsy-package, 2
* palette
  colorPalette, 30
* save
  saveCanvas, 31
* theme
  theme_canvas, 32

aRtsy (aRtsy-package), 2
aRtsy-package, 2

canvas_ant, 3
canvas_blacklight, 4
canvas_chladni, 5
canvas_circlemap, 6
canvas_cobweb, 7
canvas_collatz, 8
canvas_diamonds, 9
canvas_flow, 10
canvas_forest, 11
canvas_forest, 11
canvas_function, 12
canvas_gemstone, 13
canvas_mandelbrot, 14
canvas_maze, 15
canvas_mosaic, 16
canvas_nebula, 17
canvas_phyllotaxis, 18
canvas_planet, 19
canvas_polylines, 21
canvas_recaman, 22
canvas_ribbons, 23
canvas_segments, 24
canvas_squares, 25
canvas_stripes, 26
canvas_strokes, 27
canvas_turmite, 28
canvas_watercolors, 29
colorPalette, 30

saveCanvas, 31

theme_canvas, 32