

# Package ‘aRtsy’

August 27, 2021

**Title** Generative Art with 'ggplot2'

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

**Version** 0.1.1

**Date** 2021-08-27

**BugReports** <https://github.com/koenderks/aRtsy/issues>

**URL** <https://koenderks.github.io/aRtsy/>, <https://github.com/koenderks/aRtsy>, [https://twitter.com/aRtsy\\_package](https://twitter.com/aRtsy_package)

**Imports** dplyr, e1071, ggplot2, kknn, randomForest, Rcpp, reshape2, 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

## R topics documented:

aRtsy-package . . . . .	2
canvas_ant . . . . .	2
canvas_blacklight . . . . .	3
canvas_circlemap . . . . .	4
canvas_collatz . . . . .	5
canvas_diamonds . . . . .	6
canvas_forest . . . . .	7
canvas_function . . . . .	7
canvas_gemstone . . . . .	8
canvas_mandelbrot . . . . .	9
canvas_mosaic . . . . .	10
canvas_planet . . . . .	11
canvas_polylines . . . . .	12
canvas_ribbons . . . . .	13
canvas_segments . . . . .	13
canvas_squares . . . . .	14

canvas_strips	15
canvas_strokes	16
canvas_turmite	17
colorPalette	18
saveCanvas	19
theme_canvas	19

<b>Index</b>	<b>21</b>
--------------	-----------

---

aRtsy-package	<i>aRtsy — Generative Art using ggplot2</i>
---------------	---

---

## 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](mailto: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	<i>Paint Langton's Ant on a Canvas</i>
------------	--

---

## Description

This function paints Langton's Ant. Langton's ant is a two-dimensional universal Turing machine with a very simple set of rules but complex emergent behavior.

## Usage

```
canvas_ant(colors, background = '#fafafa', iterations = 1e7,
           width = 200, height = 200)
```

**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	a positive integer specifying the number of iterations of the algorithm.
width	a positive integer specifying the width of the artwork in pixels.
height	a positive integer specifying the height of the artwork in pixels.

**Value**

A ggplot object containing the artwork.

**Author(s)**

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

**References**

[https://en.wikipedia.org/wiki/Langtons\\_ant](https://en.wikipedia.org/wiki/Langtons_ant)

**Examples**

```
set.seed(1)
palette <- colorPalette('random', n = 10)
canvas_ant(colors = palette)
```

---

canvas_blacklight	<i>Paint Random Blacklights on a Canvas</i>
-------------------	---

---

**Description**

This function creates an artwork from randomly generated data by running a support vector machines regression algorithm to predict the color of each pixel on the canvas.

**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	a positive integer specifying the number of pixels (resolution x resolution) of the artwork.

**Value**

A ggplot object containing the artwork.

**Author(s)**

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

**References**

[https://en.wikipedia.org/wiki/Support-vector\\_machine](https://en.wikipedia.org/wiki/Support-vector_machine)

**Examples**

```
set.seed(2)
palette <- colorPalette('random', n = 5)
canvas_blacklight(colors = palette)
```

---

canvas\_circlemap

*Paint a Circle Map on a Canvas*

---

**Description**

This function draws a circle map on the canvas.

**Usage**

```
canvas_circlemap(colors, xmin = 0, xmax = 12.56, ymin = 0, ymax = 1,
  iterations = 10, width = 1500, height = 1500)
```

**Arguments**

colors	a string or character vector specifying the color(s) used for the artwork.
xmin	a value specifying the minimum location on the x-axis.
xmax	a value specifying the maximum location on the x-axis.
ymin	a value specifying the minimum location on the y-axis.
ymax	a value specifying the maximum location on the y-axis.
iterations	a positive integer specifying the number of iterations of the algorithm.
width	a positive integer specifying the width of the artwork in pixels.
height	a positive integer specifying the height of the artwork in pixels.

**Value**

A ggplot object containing the artwork.

**Author(s)**

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

**References**

[https://en.wikipedia.org/wiki/Arnold\\_tongue](https://en.wikipedia.org/wiki/Arnold_tongue)

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

## Examples

```
set.seed(3)
palette <- colorPalette('random', n = 5)
canvas_circlemap(colors = palette)
```

---

`canvas_collatz`*Paint Random Collatz Sequences on a Canvas*

---

## 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

<code>colors</code>	a string or character vector specifying the color(s) used for the artwork.
<code>background</code>	a character specifying the color used for the background.
<code>n</code>	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.
<code>angle.even</code>	a value specifying the angle (in radians) to use in bending the sequence at each odd number.
<code>angle.odd</code>	a value specifying the angle (in radians) to use in bending the sequence at each even number.
<code>side</code>	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](https://nl.wikipedia.org/wiki/Collatz_Conjecture)

## Examples

```
set.seed(4)
palette <- colorPalette('random', n = 5)
canvas_collatz(colors = palette, n = 100)
```

---

`canvas_diamonds`*Paint Random Diamonds on a Canvas*

---

### Description

This function draws many diamonds on the canvas and 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,  
                width = 500, height = 500)
```

### Arguments

<code>colors</code>	a string or character vector specifying the color(s) used for the artwork.
<code>background</code>	a character specifying the color used for the background.
<code>col.line</code>	a character specifying the color of the diamond borders.
<code>radius</code>	a positive value specifying the radius of the diamonds.
<code>alpha</code>	a value specifying the transparency of the diamonds. If NULL (the default), added layers become increasingly more transparent.
<code>p</code>	a value specifying the probability of drawing an empty diamond.
<code>width</code>	a positive integer specifying the width of the artwork in pixels.
<code>height</code>	a positive integer specifying the height of the artwork in pixels.

### Value

A ggplot object containing the artwork.

### Author(s)

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

### Examples

```
set.seed(5)  
palette <- colorPalette('random', n = 5)  
canvas_diamonds(colors = palette, radius = 10)
```

---

canvas_forest	<i>Paint a Random Forest on a Canvas</i>
---------------	--

---

**Description**

This function creates an artwork from randomly generated data by running a random forest classification algorithm to predict the color of each pixel on the canvas.

**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	a positive integer specifying the number of pixels (resolution x resolution) of the artwork.

**Value**

A ggplot object containing the artwork.

**Author(s)**

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

**Examples**

```
set.seed(6)
palette <- colorPalette('random', n = 5)
canvas_forest(colors = palette)
```

---

canvas_function	<i>Paint A Random Function on a Canvas</i>
-----------------	--

---

**Description**

This function paints functions with random parameters and mimics the functionality of the `generativeart` package.

**Usage**

```
canvas_function(color, background = '#fafafa')
```

**Arguments**

color	a string specifying the color used for the artwork.
background	a character specifying the color used for the background.

**Value**

A ggplot object containing the artwork.

**Author(s)**

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

**References**

<https://github.com/cutterkom/generativeart>

**Examples**

```
set.seed(10)
canvas_function(color = '#000000')
```

---

canvas\_gemstone

*Paint a Random Gemstone on a Canvas*

---

**Description**

This function creates an artwork from randomly generated data by running a k-nearest neighbors regression algorithm to predict the color of each pixel on the canvas.

**Usage**

```
canvas_gemstone(colors, maxk = 1, n = 1000, resolution = 500)
```

**Arguments**

colors	a string or character vector specifying the color(s) used for the artwork.
maxk	a positive integer specifying the maximum number of nearest neighbors to consider.
n	a positive integer specifying the number of random data points to generate.
resolution	a positive integer specifying the number of pixels (resolution x resolution) of the artwork.

**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](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)



## Examples

```
set.seed(7)
palette <- colorPalette('random', n = 10)
canvas_gemstone(colors = palette)
```

---

canvas_mandelbrot	<i>Paint the Mandelbrot Set on a Canvas</i>
-------------------	---

---

## Description

This function draws the Mandelbrot set on the canvas.

## Usage

```
canvas_mandelbrot(colors, iterations = 100, zoom = 1, xmin = -1.7, xmax = -0.2,
  ymin = -0.2999, ymax = 0.8001, width = 500, height = 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.
xmin	a value specifying the minimum location on the x-axis.
xmax	a value specifying the maximum location on the x-axis.
ymin	a value specifying the minimum location on the y-axis.
ymax	a value specifying the maximum location on the y-axis.
width	a positive integer specifying the width of the artwork in pixels.
height	a positive integer specifying the height of the artwork in pixels.

## Value

A ggplot object containing the artwork.

## Author(s)

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

## References

[https://en.wikipedia.org/wiki/Mandelbrot\\_set](https://en.wikipedia.org/wiki/Mandelbrot_set)

## Examples

```
set.seed(8)
palette <- colorPalette('random', n = 6)
canvas_mandelbrot(colors = palette, zoom = 10)
```

---

`canvas_mosaic`*Paint a Random Mosaic on a Canvas*

---

### Description

This function paints a mosaic from randomly generated data by running a k-nearest neighbors classification algorithm to predict the color of each pixel on the canvas. Low values of `maxk` produce a mosaic like artwork, while higher values produce a more smooth decision boundary.

### Usage

```
canvas_mosaic(colors, maxk = 1, n = 1000, resolution = 500)
```

### Arguments

<code>colors</code>	a string or character vector specifying the color(s) used for the artwork.
<code>maxk</code>	a positive integer specifying the maximum number of nearest neighbors to consider.
<code>n</code>	a positive integer specifying the number of random data points to generate.
<code>resolution</code>	a positive integer specifying the number of pixels (resolution x resolution) of the artwork.

### 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](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)

### Examples

```
set.seed(9)
palette <- colorPalette('random', n = 10)
canvas_mosaic(colors = palette)
```

---

canvas_planet	<i>Paint a Random Planet on a Canvas</i>
---------------	--

---

## 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, width = 1500, height = 1500)
```

## Arguments

colors	a character specifying the colors used for the planet(s). Can also be a list where each entry is a vector of colors for each 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.
width	a positive integer specifying the width of the artwork in pixels.
height	a positive integer specifying the height of the artwork in pixels.

## 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

```
# Sun behind Earth and Moon
set.seed(1)
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

*Paint Random Polygons and Lines on a Canvas*


---

## 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,
  alpha = NULL, size = 0.1, width = 500, height = 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.
alpha	a value specifying the transparency of the polygons. If NULL (the default), added layers become increasingly more transparent.
size	a positive value specifying the size of the borders.
width	a positive integer specifying the width of the artwork in pixels.
height	a positive integer specifying the height of the artwork in pixels.

## Value

A ggplot object containing the artwork.

## Author(s)

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

## Examples

```
set.seed(11)
palette <- colorPalette('random', n = 10)
canvas_polylines(colors = palette)
```

---

`canvas_ribbons`*Paint Random Ribbons on a Canvas*

---

**Description**

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

**Usage**

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

**Arguments**

<code>colors</code>	a string or character vector specifying the color(s) used for the artwork. The number of colors determines the number of ribbons.
<code>background</code>	a character specifying the color of the background.
<code>triangle</code>	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>

**Examples**

```
set.seed(12)
palette <- colorPalette('random', n = 4)
canvas_ribbons(colors = palette)
```

---

`canvas_segments`*Paint Random Line Segments on a Canvas*

---

**Description**

This function draws many random line segments on the canvas.

**Usage**

```
canvas_segments(colors, background = '#fafafa', n = 100,
  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 vectical 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>

**Examples**

```
set.seed(13)
palette <- colorPalette('random', n = 4)
canvas_segments(colors = palette)
```

---

canvas_squares	<i>Paint Random Squares on a Canvas</i>
----------------	---

---

**Description**

This function paints random squares. 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,
               width = 100, height = 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 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.
width	a positive integer specifying the width of the artwork in pixels.
height	a positive integer specifying the height of the artwork in pixels.

**Value**

A ggplot object containing the artwork.

**Author(s)**

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

**Examples**

```
set.seed(14)
palette <- colorPalette('random', n = 4)
canvas_squares(colors = palette)
```

---

canvas\_stripes

*Paint Random Stripes on a Canvas*

---

**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>

**Examples**

```
set.seed(15)
palette <- colorPalette('random', n = 4)
canvas_stripes(colors = palette)
```

---

`canvas_strokes`*Paint Random Strokes on a Canvas*

---

**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,  
               width = 500, height = 500, side = FALSE)
```

**Arguments**

<code>colors</code>	a string or character vector specifying the color(s) used for the artwork.
<code>neighbors</code>	a positive integer specifying the number of neighbors a block considers when taking over a color. More neighbors fades the artwork.
<code>p</code>	a value specifying the probability of selecting a new color at each block. A higher probability adds more noise to the artwork.
<code>iterations</code>	a positive integer specifying the number of iterations of the algorithm. More iterations generally apply more fade to the artwork.
<code>width</code>	a positive integer specifying the width of the artwork in pixels.
<code>height</code>	a positive integer specifying the height of the artwork in pixels.
<code>side</code>	logical. Whether to put the artwork on its side.

**Value**

A ggplot object containing the artwork.

**Author(s)**

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

**Examples**

```
set.seed(16)  
palette <- colorPalette('random', n = 6)  
canvas_strokes(colors = palette)
```



---

canvas\_turmite*Paint a Random Turmite on a Canvas*

---

## 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. The algorithm is simple: 1) turn on the spot (left, right, up, down) 2) change the color of the square 3) move forward one square.

## Usage

```
canvas_turmite(color, background = '#fafafa', p = 0.5, iterations = 1e7,  
               width = 1500, height = 1500)
```

## Arguments

color	a character specifying the color used for the artwork.
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.
width	a positive integer specifying the width of the artwork in pixels.
height	a positive integer specifying the height of the artwork in pixels.

## Value

A ggplot object containing the artwork.

## Author(s)

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

## References

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

## Examples

```
set.seed(17)  
canvas_turmite(color = '#000000')
```

colorPalette

*Color palette generator.***Description**

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

**Usage**

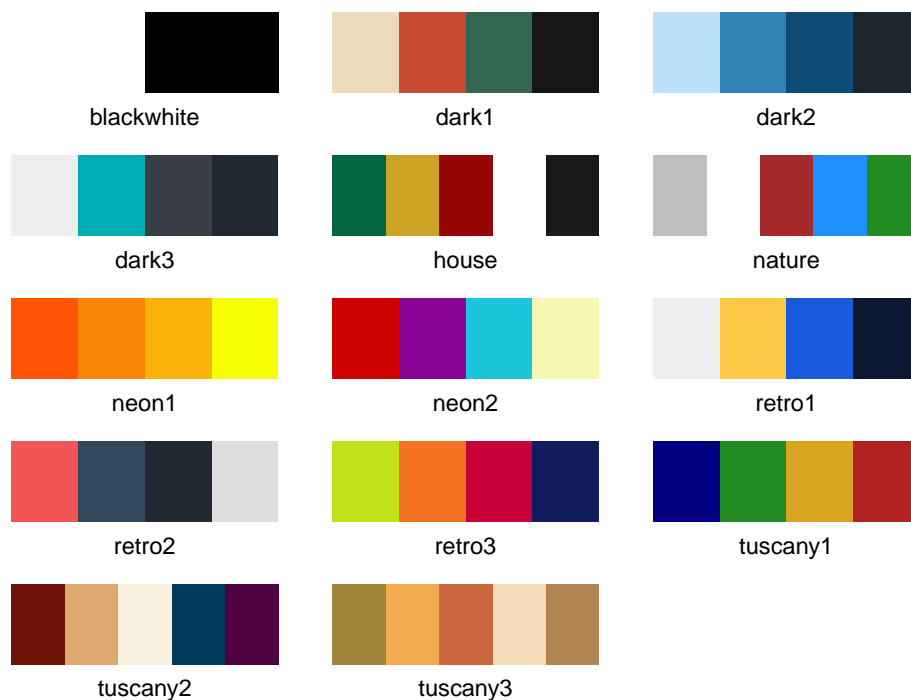
```
colorPalette(name, n = NULL)
```

**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	<i>Save a canvas to an external device.</i>
------------	---

---

**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	<i>Canvas theme for ggplot2 objects</i>
--------------	---

---

**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 = '#fafafa', margin = -1.25)
```

**Arguments**

<code>x</code>	a <code>ggplot2</code> object.
<code>background</code>	a character specifying the color used for the empty canvas.
<code>margin</code>	margins of the plot.

**Value**

A `ggplot` object containing the artwork.

**Author(s)**

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

# Index

- \* **aRtsy**
    - aRtsy-package, [2](#)
  - \* **artwork**
    - canvas\_ant, [2](#)
    - canvas\_blacklight, [3](#)
    - canvas\_circlemap, [4](#)
    - canvas\_collatz, [5](#)
    - canvas\_diamonds, [6](#)
    - canvas\_forest, [7](#)
    - canvas\_function, [7](#)
    - canvas\_gemstone, [8](#)
    - canvas\_mandelbrot, [9](#)
    - canvas\_mosaic, [10](#)
    - canvas\_planet, [11](#)
    - canvas\_polylines, [12](#)
    - canvas\_ribbons, [13](#)
    - canvas\_segments, [13](#)
    - canvas\_squares, [14](#)
    - canvas\_strips, [15](#)
    - canvas\_strokes, [16](#)
    - canvas\_turmite, [17](#)
  - \* **canvas**
    - canvas\_ant, [2](#)
    - canvas\_blacklight, [3](#)
    - canvas\_circlemap, [4](#)
    - canvas\_collatz, [5](#)
    - canvas\_diamonds, [6](#)
    - canvas\_forest, [7](#)
    - canvas\_function, [7](#)
    - canvas\_gemstone, [8](#)
    - canvas\_mandelbrot, [9](#)
    - canvas\_mosaic, [10](#)
    - canvas\_planet, [11](#)
    - canvas\_polylines, [12](#)
    - canvas\_ribbons, [13](#)
    - canvas\_segments, [13](#)
    - canvas\_squares, [14](#)
    - canvas\_strips, [15](#)
    - canvas\_strokes, [16](#)
    - canvas\_turmite, [17](#)
    - colorPalette, [18](#)
    - saveCanvas, [19](#)
    - theme\_canvas, [19](#)
  - \* **package**
    - aRtsy-package, [2](#)
  - \* **palette**
    - colorPalette, [18](#)
  - \* **save**
    - saveCanvas, [19](#)
  - \* **theme**
    - theme\_canvas, [19](#)
- aRtsy (aRtsy-package), [2](#)
- aRtsy-package, [2](#)
- canvas\_ant, [2](#)
- canvas\_blacklight, [3](#)
- canvas\_circlemap, [4](#)
- canvas\_collatz, [5](#)
- canvas\_diamonds, [6](#)
- canvas\_forest, [7](#)
- canvas\_function, [7](#)
- canvas\_gemstone, [8](#)
- canvas\_mandelbrot, [9](#)
- canvas\_mosaic, [10](#)
- canvas\_planet, [11](#)
- canvas\_polylines, [12](#)
- canvas\_ribbons, [13](#)
- canvas\_segments, [13](#)
- canvas\_squares, [14](#)
- canvas\_strips, [15](#)
- canvas\_strokes, [16](#)
- canvas\_turmite, [17](#)
- colorPalette, [18](#)
- saveCanvas, [19](#)
- theme\_canvas, [19](#)