Package: jasmines (via r-universe)

August 19, 2024

Title Generative Art

Version 0.0.0.9001

Description It doesn't do much, really.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports ambient, purrr, dplyr, scico, ggplot2, tidyr, stringr, tibble, viridis, grImport, animation, methods, progress, magrittr, gganimate, e1071, magick, spatstat, sp, patchwork

URL https://github.com/djnavarro/jasmines

BugReports https://github.com/djnavarro/jasmines/issues

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

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

RemoteUrl https://github.com/djnavarro/jasmines

RemoteRef HEAD

RemoteSha f5455c3e0f35043f4147b5b2af32505517c671c5

Contents

`as_hex` *Convert image to hex*

Description

Convert image to hex

Usage

```
as_hex(
  from,
  to,
  text_label = NULL,
  text_colour = "white",
  border_colour = "grey20",
  border_opacity = 60
)
```

Arguments

`from`	Path to original file
`to`	Path to destination file
`text_label`	Text annotation
`text_colour`	Colour of text annotation
`border_colour`	Colour of the border region
`border_opacity`	Opacity of the border region

entitytype *Entity types*

Description

Entity types

Usage

```
entity_circle(seed = use_seed(1), grain = 50, id = NULL, ...)

entity_line(seed = use_seed(1), grain = 50, id = NULL, ...)

entity_heart(seed = use_seed(1), grain = 50, id = NULL, ...)

entity_droplet(seed = use_seed(1), grain = 50, id = NULL, shape = 3, ...)

entity_lissajous(
  seed = use_seed(1),
  grain = 500,
  id = NULL,
  start = 0,
  end = 30,
  shape = list(a = 1, b = 1, w = 0.3, d = 1),
  ...
)

entity_gaussian(seed = use_seed(1), grain = 50, id = NULL, ...)

entity_null(seed = use_seed(1), ...)
```

Arguments

seed	Parameter specifying seed (default = NULL)
grain	The number of points that comprise the entity
id	A numeric identifier for the entity
...	Parameters to be passed to locate_entity
shape	Parameter controlling the shape of the entity (droplet, lissajous)
start	Parameter controlling a start location for a line (lissajous)
end	Parameter controlling an end location for a line (lissajous)

Details

Primitive entities in jasmines are tibbles with four columns: x and y specify co-ordinate values, the id is a number identifying the object, and the type is a character label indicating what kind of entity

it is. By default, entities are assigned a random integer as the id code, but it is often wise for the calling function to assign the id in a more predictable fashion. The shape parameter can sometimes be a list.

Value

A tibble with four columns: x, y, id and type

---

export_animation *Export animation to a file*

Description

Export animation to a file

Usage

```
export_animation(input, filename, nframes = 200, detail = 5, type = "cairo")
```

Arguments

input	art object to print
filename	filename
nframes	defaults to 200
detail	number of interpolated frames, defaults to 5
type	defaults to "cairo"

---

export_image *Export image to a file*

Description

Export image to a file

Usage

```
export_image(
  input,
  filename,
  width = 10,
  height = 10,
  dpi = 300,
  xlim = NULL,
  ylim = NULL
)
```

Arguments

input	art object to print
filename	filename
width	defaults to 3000 pixels (10in at 300dpi)
height	defaults to 3000 pixels (10in at 300dpi)
dpi	defaults to 300dpi
xlim	by default plot limits are c(-.05, 1.05), relative to data spanning c(0, 1), but can override
ylim	by default plot limits are c(-.05, 1.05), relative to data spanning c(0, 1), but can override

---

locate_entity *Locate entities*

Description

Locate entities

Usage

```
locate_entity(entity, xpos = 0, ypos = 0, size = 1, angle = 0, ...)
```

Arguments

entity	The entity to be placed
xpos	The horizontal location of the entity
ypos	The vertical location of the entity
size	Parameter controlling the size of the entity
angle	Parameter controlling the orientation of the entity
...	Other arguments are ignored

Details

When a jasmine entity is created it is implicitly assumed to be located at the origin (xpos = 0, ypos = 0), to have size 1, and to have a horizontal orientation (angle = 0). The locate_entity function allows the entity to be transformed in simple ways: translation, dilation and rotations

Value

A tibble with four columns: x, y, id and type

palette_adjust *Function factory for adjusted palettes*

Description

Function factory for adjusted palettes

Usage

```
palette_adjust(name, prefix, ...)
```

Arguments

name	Name of the base palette
prefix	Vector of colours to preprend to the named palette
...	Arguments to be passed to grDevices::adjustcolor

Value

A modified

palette_manual *Function factory for manual palettes*

Description

Function factory for manual palettes

Usage

```
palette_manual(...)
```

Arguments

...	colour names

Value

a function that takes arguments n and alpha

palette_named *Function factory for prespecified palettes*

Description

Function factory for prespecified palettes

Usage

```
palette_named(name = NULL, ...)
```

Arguments

name	name of the palette
...	arguments to be passed to other functions

Value

a function that takes arguments n and alpha

scene_bubbles *Create a scene comprised of circles of varying size and location*

Description

Create a scene comprised of circles of varying size and location

Usage

```
scene_bubbles(seed = use_seed(1), n = 2, grain = 100)
```

Arguments

seed	Seed number to attach
n	Number of circles
grain	The number of points per row

Value

A tibble with four columns: x, y, id and type

scene_delaunay *Create a scene using Delaunay triangulation*

Description

Create a scene using Delaunay triangulation

Usage

```
scene_delaunay(seed = use_seed(1), n = 20, grain = 50)
```

Arguments

seed	The RNG seed
n	Number of vertices
grain	Number of points along each line

Value

A tibble with four columns: x, y, id and type

scene_discs *Create a scene comprised of concentric circles*

Description

Create a scene comprised of concentric circles

Usage

```
scene_discs(seed = use_seed(1), points = 100, rings = 3, size = 2)
```

Arguments

seed	Seed number to attach
points	Total number of interior points
rings	How many rings to spread across?
size	Diameter of the outermost ring

scene_grid *Create a scene with entities on a grid*

Description

Create a scene with entities on a grid

Usage

```
scene_grid(
  seed = use_seed(1),
  xpos = 1:3,
  ypos = 1:3,
  entity = "circle",
  grain = 50,
  size = 1,
  shape = 3,
  angle = 0
)
```

Arguments

seed	Seed number to attach
xpos	Numeric vector specifying horizontal locations
ypos	Numeric vector specifying vertical locations
entity	The entity type (e.g., "line", "circle")
grain	The number of points per entity
size	The size of each entity
shape	The shape of each entity
angle	The angle of each entity

Details

The scene_grid() function allows multiple entities to be included in the initial object, laying out items in grid.

Value

A tibble with four columns: x, y, id and type

scene_mix *Create a scene with entities placed randomly*

Description

Create a scene with entities placed randomly

Usage

```
scene_mix(
  seed = use_seed(1),
  n = 5,
  xpos = (1:20)/4,
  ypos = (1:20)/4,
  entity = c("circle", "line", "heart", "droplet"),
  grain = 100,
  size = (10:20)/20,
  shape = 3,
  angle = seq(0, 2 * pi, length.out = 20)
)
```

Arguments

seed	Seed number to attach
n	Number of entities
xpos	Numeric vector specifying possible horizontal locations
ypos	Numeric vector specifying possible vertical locations
entity	Character vector specifying possible entity types (e.g., "line", "circle")
grain	Numeric vector specifying possible grains
size	Numeric vector specifying possible sizes
shape	Numeric vector specifying possible shapes
angle	Numeric vector specifying possible angles

Value

A tibble with four columns: x, y, id and type

scene_rows *Create a scene comprised of horizontal or vertical lines*

Description

Create a scene comprised of horizontal or vertical lines

Usage

```
scene_rows(seed = use_seed(1), n = 10, grain = 100, vertical = FALSE)
```

Arguments

seed	Seed number to attach
n	Number of rows
grain	The number of points per row
vertical	Flip the x/y co-ords to produce columns?

Value

A tibble with columns: x, y, id, type, seed

scene_sticks *Create a scene comprised of lines of varying length and orientation*

Description

Create a scene comprised of lines of varying length and orientation

Usage

```
scene_sticks(seed = use_seed(1), n = 10, grain = 100)
```

Arguments

seed	Seed number to attach
n	how many sticks
grain	how many points along each stick

Value

a tibble with columns x, y and id

style_overlay *Adds an overlay to the existing plot*

Description

Adds an overlay to the existing plot

Usage

```
style_overlay(pic, border = NULL, fill = NULL, linewidth = 1, data = NULL)
```

Arguments

pic	Existing plot
border	Colour of border
fill	Colour of fill
linewidth	Width of border
data	Data to be shown in overlay (if NULL, taken from pic)

style_pop *Style as a pop art image*

Description

Style as a pop art image

Usage

```
style_pop(
  data,
  palette = "base",
  colour = "ind",
  alpha = 0.3,
  fade = 0,
  background = "warhol",
  adjust = 0.7,
  panels = 4,
  ...
)
```

Arguments

data	data frame with x, y, order, id, time
palette	function generating a palette (or string naming the palette)
colour	name of variable to use to specify the colour aesthetic
alpha	length two numeric, first element is the initial alpha, (optional) second is the decay rate for alpha
background	colour of the background in the plot
...	arguments to pass to geom

Value

Returns a ggplot2 object

---

style_ribbon *Style as a ribbon image*

Description

Style as a ribbon image

Usage

```
style_ribbon(
  data,
  palette = "viridis",
  colour = "order",
  alpha = c(0.3, 0),
  background = "black",
  discard = 0,
  type = "segment",
  ...
)
```

Arguments

data	data frame with x, y, order, id, time
palette	function generating a palette (or string naming the palette)
colour	name of variable to use to specify the colour aesthetic
alpha	length two numeric, first element is the initial alpha, (optional) second is the decay rate for alpha
background	colour of the background in the plot
discard	how many iterations should we discard before drawing?
type	type of geom to use ("segment", "curve" or "point")
...	arguments to pass to geom

Value

Returns a ggplot2 object

`style_walk` *Style as animated points*

Description

Style as animated points

Usage

```
style_walk(
  data = unfold_meander(),
  wake_length = 0.1,
  palette = palette_scico(palette = "berlin"),
  background = "black",
  ...
)
```

Arguments

`data`	tibble specifying the time series
`wake_length`	length of the tail
`palette`	function generating palette values
`background`	colour of the background
`...`	other arguments to pass to shadow_wake
	The `style_walk()` function generates an animation as an output. The input data takes the form of a tibble with variables `x` and `y` specifying co-ordinate values, a `id` variable identifying each point and a `time` variable specifying the time

Value

The output is a gganim object

`unfold_breeze` *Unfold a scene with a swirl operation*

Description

Unfold a scene with a swirl operation

Usage

```
unfold_breeze(
  data = scene_sticks(),
  iterations = 6,
  scale = 0.02,
  drift = 0.01,
  noise = NULL,
  fractal = NULL,
  octaves = 8,
  output = "time",
  ...
)
```

Arguments

`data`	data frame with x, y, id, and more
`iterations`	how many times should we iterate?
`scale`	how large is each step?
`drift`	gaussian noise to inject at each step
`noise`	noise function (default is ambient::gen_simplex)
`fractal`	fractal function (default is ambient::billow)
`octaves`	default = 8
`output`	name of the unfolding variable to add (e.g., time)
`...`	arguments to pass to ambient::fracture

Value

a "tempest" ribbon, data frame with x, y, order, time and id

unfold_inside *Unfold a scene with an inside operation*

Description

Unfold a scene with an inside operation

Usage

```
unfold_inside(data, output = "inside")
```

Arguments

data	Data
output	String specifying a Column name

Value

Returns the original data tibble with a new column added

unfold_loop *Unfold a scene with a loop operation*

Description

Unfold a scene with a loop operation

Usage

```
unfold_loop(data, points = 20, radius = 1)
```

Arguments

data	data
points	number of time points
radius	radius of the circle

Value

a tibble with x, y, id and time

unfold_meander *Unfold a scene with a meander operation*

Description

Unfold a scene with a meander operation

Usage

```
unfold_meander(
  data = entity_circle(),
  iterations = 100,
  smoothing = 6,
  endpause = 0,
  output1 = "time",
  output2 = "series"
)
```

Arguments

data	data frame with x, y, etc
iterations	number of time points in the time series
smoothing	number of smoothing iterations
endpause	length of pause at the end
output1	name of the primary unfolding variable to add (e.g., time)
output2	name of the secondary unfolding variable to add (e.g., series)

Value

tibble with columns series, time, x, y

unfold_slice *Unfold a scene with a slice operation*

Description

Unfold a scene with a slice operation

Usage

```
unfold_slice(
  data = scene_sticks(),
  iterations = 6,
  scale = 0.2,
  scatter = FALSE,
  output1 = "time",
  output2 = "order"
)
```

Arguments

data	data frame with x, y, id, and more
iterations	how many times should we iterate the curl noise?
scale	how large is each curl step?
scatter	should the noise seed be "scattered"?
output1	name of the primary unfolding variable to add (e.g., time)
output2	name of the secondary unfolding variable to add (e.g., order)

Value

a "tempest" ribbon, data frame with x, y, order, time and id

---

unfold_tempest *Unfold a scene with a tempest operation*

Description

Unfold a scene with a tempest operation

Usage

```
unfold_tempest(
  data = scene_sticks(),
  iterations = 6,
  scale = 0.02,
  scatter = FALSE,
  output1 = "time",
  output2 = "order"
)
```

Arguments

data	data frame with x, y, id, and more
iterations	how many times should we iterate the curl noise?
scale	how large is each curl step?
scatter	should the noise seed be "scattered"?
output1	name of the primary unfolding variable to add (e.g., time)
output2	name of the secondary unfolding variable to add (e.g., order)

Value

a "tempest" ribbon, data frame with x, y, order, time and id

---

unfold_warp *Unfold a scene with a warp operation*

Description

Unfold a scene with a warp operation

Usage

```
unfold_warp(
  data = scene_sticks(),
  iterations = 6,
  scale = 0.02,
  scatter = FALSE,
  output = "time"
)
```

Arguments

data	data frame with x, y, id, and more
iterations	how many times should we iterate the noise?
scale	how large is each step?
scatter	should the noise seed be "scattered"?
output	name of the output variable (default = time)

Value

thing

unfold_worley *Unfold a scene with a worley operation*

Description

Unfold a scene with a worley operation

Usage

```
unfold_worley(data, scatter = FALSE, output = "order", ...)
```

Arguments

data	the data object
scatter	should the noise seed be "scattered"?
output	name of the primary unfolding variable to add (e.g., order)
...	arguments to pass to ambient::genworley()

use_seed *Wrapper to set.seed*

Description

Wrapper to set.seed

Usage

```
use_seed(seed = 1)
```

Arguments

seed	the seed value

Index