Package 'transformr'

February 26, 2024

Type Package

Title Polygon and Path Transformations

Version 0.1.5

Maintainer Thomas Lin Pedersen <thomasp85@gmail.com>

Description In order to smoothly animate the transformation of polygons and paths, many aspects needs to be taken into account, such as differing number of control points, changing center of rotation, etc. The 'transformr' package provides an extensive framework for manipulating the shapes of polygons and paths and can be seen as the spatial brother to the 'tweenr' package.

License MIT + file LICENSE

Encoding UTF-8

Imports tweenr, rlang, sf, lpSolve, vctrs

Suggests covr, magrittr

LinkingTo cpp11

RoxygenNote 7.3.1

URL https://github.com/thomasp85/transformr

BugReports https://github.com/thomasp85/transformr/issues

NeedsCompilation yes

Author Thomas Lin Pedersen [cre, aut]
(<https://orcid.org/0000-0002-5147-4711>)

Repository CRAN

Date/Publication 2024-02-26 14:30:02 UTC

R topics documented:

`simple_shapes` *Some different geometries to play with*

Description

These functions are provided to allow you to play with somee simple shapes as you explore `transformr` and are also used in the examples for the different tween functions. All geometries can be returned as either a standard `data.frame` with `x`, `y`, and `id` column, or as an sf geometry of the appropriate type.

Usage

```
poly_circle(st = FALSE, detail = 360)

poly_circles(st = FALSE, n = 3, r = 0.25, detail = 360)

poly_star(st = FALSE, n = 5, r1 = 0.5)

poly_star_hole(st = FALSE, n = 5, r1 = 0.5)

path_spiral(st = FALSE, windings = 5)

path_waves(st = FALSE, w1 = 7, w2 = 11)

point_random(st = FALSE, n = 10)

point_grid(st = FALSE, dim = 5)
```

Arguments

`st`	Logical. Should the geometry be returned as an `sf` feature?
`detail`	The number of points defining the shape
`n`	For `poly_circles` the number of circles, for `poly_star` and `poly_star_hole` the number of 'arms', and for `point_random` the number of points
`r, r1`	The radius of the geometry. `r` gives the radius of the circles in `poly_circles` and `r1` gives the inner radius for `poly_star`/`poly_star_hole`, thus determining how pointy it is
`windings`	The number of revolutions in the spiral
`w1, w2`	The frequency for the two sine waves
`dim`	the number of rows and columns in the grid

Value

Either a data.frame or an sf feature depending on the value of `st`

Examples

```
# Create a 7-pointed star
poly_star(n = 7)
```

`st_normalize` *Normalise a geometry to fit inside a unit square*

Description

This is a small helper function that will take an sf geometry and fit it inside the unit square (a square centered on 0 and ranging from -1 to 1 in both dimensions). The function will retain the aspect ratio of the geometry and simply scale it down until it fits.

Usage

```
st_normalize(st)
```

Arguments

`st`	An sf geometry such as `sf`, `sfc`, or `sfg`

Value

An object of the same type as `st`

Examples

```
library(sf)
nc <- st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)
st_bbox(nc)

nc_norm <- st_normalize(nc)
st_bbox(nc_norm)
```

`tween_path` *Transition between path data frames*

Description

This function is equivalent to `tweenr::tween_state()` but expects the data to have an `x` and `y` column and encode paths.

Usage

```
tween_path(
  .data,
  to,
  ease,
  nframes,
  id = NULL,
  enter = NULL,
  exit = NULL,
  match = TRUE
)
```

Arguments

.data	A data.frame to start from. If .data is the result of a prior tween, only the last frame will be used for the tween. The new tween will then be added to the prior tween
to	A data.frame to end at. It must contain the same columns as .data (exluding .frame)
ease	The easing function to use. Either a single string or one for each column in the data set.
nframes	The number of frames to calculate for the tween
id	The column to match observations on. If NULL observations will be matched by position. See the *Match, Enter, and Exit* section for more information.
enter, exit	functions that calculate a start state for new observations that appear in to or an end state for observations that are not present in to. If NULL the new/old observations will not be part of the tween. The function gets a data.frame with either the start state of the exiting observations, or the end state of the entering observations and must return a modified version of that data.frame. See the *Match, Enter, and Exit* section for more information.
match	Should polygons be matched by id? If FALSE then polygons will be matched by shortest distance and if any state has more polygons than the other, the other states polygons will be chopped up so the numbers match.

Value

A data.frame containing intermediary states

Aligning paths

There is less work required to align paths than there is to align polygons, simply because no rotation is possible/required, and the notion of clockwise winding order is not meaningful in the scope of paths. Still, paths need to be matched and the number of points in each pair of matched paths must be equal. Paths are matched based on relative length rather than on position and seek to minimize the change in length during transition. This is chosen from the point of view that huge elongation or contraction are much more distracting than longer travel distances.

Cutting paths

If the number of paths to transition between is not even, some of the paths need to be cut in order to succesfully match the paths. The cuts are distributed based on the same algorithm that distributes cuts in polygons and seek to cut the lines into as even-length pieces as possible.

Multipaths

It is possible to encode multiple paths with the same id be separating them with a `NA` row, much in the same way as holes are encoded in polygons. If paths are not matched based on id (`match = FALSE`) then multipaths will simply be split into separate paths. On the other hand, if paths are matched by id all paths within a multipath will transition into the (multi)path that has the same id in the other state.

`tween_polygon` *Transition between polygon data.frames*

Description

This function is equivalent to `tweenr::tween_state()` except that data is interpeted as encoding polygons. Data is expected to have an `x` and `y` column encoding the location of corners in the polygon.

Usage

```
tween_polygon(
  .data,
  to,
  ease,
  nframes,
  id = NULL,
  enter = NULL,
  exit = NULL,
  match = TRUE
)
```

Arguments

`.data`	A data.frame to start from. If `.data` is the result of a prior tween, only the last frame will be used for the tween. The new tween will then be added to the prior tween
`to`	A data.frame to end at. It must contain the same columns as .data (exluding `.frame`)
`ease`	The easing function to use. Either a single string or one for each column in the data set.
`nframes`	The number of frames to calculate for the tween

`id`	The column to match observations on. If `NULL` observations will be matched by position. See the *Match, Enter, and Exit* section for more information.
`enter, exit`	functions that calculate a start state for new observations that appear in `to` or an end state for observations that are not present in `to`. If `NULL` the new/old observations will not be part of the tween. The function gets a data.frame with either the start state of the exiting observations, or the end state of the entering observations and must return a modified version of that data.frame. See the *Match, Enter, and Exit* section for more information.
`match`	Should polygons be matched by id? If `FALSE` then polygons will be matched by shortest distance and if any state has more polygons than the other, the other states polygons will be chopped up so the numbers match.

Value

A data.frame containing intermediary states

Aligning polygons

`transformr` performs a lot of work to try to ensure the transition between different shapes are as smooth and direct as possible. The first operation is to ensure that the two end states of the polygon are both drawn clockwise, so that the transition will not contain an inversion. Second, we need to make sure that each end state is drawn with the same number of points. If not, the less detailed polygon will get points inserted at the longest edges so that the number is even between the two states. Third, we rotate the last state so as to minimize the cumulative distance between all point pairs, thus ensuring that the transition will involve a minimum of rotation.

Cutting polygons

If the transition involves changing the number of polygons, there are two strategies: Making polygons appear/disappear to even out the number, or cutting up the polygons in the state with the fewest in order to create the same number of polygons for the transition. In the latter case, a choice have to be made with regards to which polygons to cut, into how many, and where to cut it. `transformr` will distribute the number of cuts among candidate polygons based on their relative area, ensuring that it is not necessarily the largest polygon that gets all the cuts, but that divisions are distributed as fairly as possible. For deciding on where to cut the polygons they are triangulated and the triangles are then reassembled into the number of pieces needed by always adding to the smallest piece.

Polygon with holes

`transformr` support polygons with any number of holes. Holes are encoded by adding an `NA` row to the main enclosing polygon and appending the hole after that. Multiple holes are likewise added by simply separating them with `NA` rows. A hole might get cut up and disappear during transition if the polygon needs to be divided. When transitioning between polygons with holes the holes are matched by position to minimize the travel distance. If there is a mismatch between the number of holes in each end state then new zero-area holes are inserted in the centroid of the polygon with the fewest to even out the number.

Examples

```
library(magrittr)
star <- poly_star_hole()
circle <- poly_circle()
circles <- poly_circles()

tween_polygon(circle, star, 'cubic-in-out', 20) %>%
  tween_polygon(circles, 'cubic-in-out', 20)
```

`tween_sf` *Transition between data.frames containing sfc columns*

Description

This function is equivalent to `tweenr::tween_state()` except that it understands `sf::sfc` columns, as defined by the `sf` package. An `sfc` column is a column containing simple features and can this hold both points, lines polygons and more. `tween_sf` currently has support for (multi)point, (multi)linestring, and (multi)polygon types and requires that the transition is between compatible types (points-to-points, linestring-to-linestring, polygon-to-polygon). For (multi)linestring and (multi)polygon, the behavior is similar to `tween_path()` and `tween_polygon()` respectively, with each feature beeing run through the respective function with `match = FALSE`. For (multi)points it behaves more or less like `tweenr::tween_state()` except additional points are added as needed to make the to stages contain the same number of points. Points are added on top of existing points so it appears as if the points are divided into more.

Usage

```
tween_sf(.data, to, ease, nframes, id = NULL, enter = NULL, exit = NULL)
```

Arguments

`.data`	A data.frame to start from. If `.data` is the result of a prior tween, only the last frame will be used for the tween. The new tween will then be added to the prior tween
`to`	A data.frame to end at. It must contain the same columns as .data (exluding `.frame`)
`ease`	The easing function to use. Either a single string or one for each column in the data set.
`nframes`	The number of frames to calculate for the tween
`id`	The column to match observations on. If `NULL` observations will be matched by position. See the *Match, Enter, and Exit* section for more information.
`enter, exit`	functions that calculate a start state for new observations that appear in `to` or an end state for observations that are not present in `to`. If `NULL` the new/old observations will not be part of the tween. The function gets a data.frame with either the start state of the exiting observations, or the end state of the entering observations and must return a modified version of that data.frame. See the *Match, Enter, and Exit* section for more information.

Value

A data.frame containing intermediary states

Examples

```
library(magrittr)
star_hole <- poly_star_hole(st = TRUE)
circles <- poly_circles(st = TRUE)
spiral <- path_spiral(st = TRUE)
waves <- path_waves(st = TRUE)
random <- point_random(st = TRUE)
grid <- point_grid(st = TRUE)
df1 <- data.frame(geo = sf::st_sfc(star_hole, spiral, random))
df2 <- data.frame(geo = sf::st_sfc(circles, waves, grid))

tween_sf(df1, df2, 'linear', 30)
```

Index