Package 'tourr'

August 2, 2017

Title Implement Tour Methods in R Code

Version 0.5.5

Author Dianne Cook <dicook@iastate.edu>, Hadley Wickham <h.wickham@gmail.com>

Maintainer Hadley Wickham <h.wickham@gmail.com>

Description Implements geodesic interpolation and basis generation functions that allow you to create new tour methods from R.

Depends R (>= 2.14)

Suggests TeachingDemos, ash, testthat, colorspace, ggplot2, reshape2

License MIT + file LICENSE

LazyData true

URL https://github.com/ggobi/tourr

BugReports https://github.com/ggobi/tourr/issues

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-08-02 05:15:17 UTC

R topics documented:

`andrews` *Compute Andrews' curves*

Description

This function takes a numeric vector of input, and returns a function which allows you to compute the value of the Andrew's curve at every point along its path from -pi to pi.

Usage

```
andrews(x)
```

Arguments

x	input a new parameter

Value

a function with single argument, theta

Examples

```
a <- andrews(1:2)
a(0)
a(-pi)
grid <- seq(-pi, pi, length = 50)
a(grid)

plot(grid, andrews(1:2)(grid), type = "l")
plot(grid, andrews(runif(5))(grid), type = "l")
```

---

animate *Animate a tour path.*

Description

This is the function that powers all of the tour animations. If you want to write your own tour animation method, the best place to start is by looking at the code for animation methods that have already implemented in the package.

Usage

```
animate(data, tour_path = grand_tour(), display = display_xy(),
  start = NULL, aps = 1, fps = 30, max_frames = Inf, rescale = TRUE,
  sphere = FALSE, ...)
```

Arguments

data	matrix, or data frame containing numeric columns
tour_path	tour path generator, defaults to 2d grand tour
display	takes the display that is suppose to be used, defaults to the xy display
start	projection to start at, if not specified, uses default associated with tour path
aps	target angular velocity (in radians per second)
fps	target frames per second (defaults to 30)
max_frames	the maximum number of bases to generate. Defaults to Inf for interactive use (must use Ctrl + C to terminate), and 1 for non-interactive use.
rescale	if true, rescale all variables to range [0,1]?
sphere	if true, sphere all variables
...	ignored

Details

See render to render animations to disk.

Value

an (invisible) list of bases visited during this tour

References

Hadley Wickham, Dianne Cook, Heike Hofmann, Andreas Buja (2011). tourr: An R Package for Exploring Multivariate Data with Projections. Journal of Statistical Software, 40(2), 1-18. http://www.jstatsoft.org/v40/i02/.

Examples

```
f <- flea[, 1:6]
animate(f, grand_tour(), display_xy())
# or in short
animate(f)
animate(f, max_frames = 30)

## Not run: animate(f, max_frames = 10, fps = 1, aps = 0.1)
```

---

center *Center a numeric vector by subtracting off its mean.*

Description

Center a numeric vector by subtracting off its mean.

Usage

```
center(x)
```

Arguments

x	numeric vector

cmass *Central mass index.*

Description

Calculates the central mass index. See Cook and Swayne (2007) Interactive and Dynamic Graphics for Data Analysis for equations.

Usage

```
cmass(mat)
```

Arguments

mat	matrix being used

dependence_tour *A dependence tour path.*

Description

The dependence tour combines a set of independent 1d tours to produce a nd tour. For the special case of 2d, this is known as a correlation tour. This tour corresponds to the multivariate method known as generalised canonical correlation, and is used to investigate dependence between groups of variables.

Usage

```
dependence_tour(pos)
```

Arguments

pos	a numeric vector describing which variables are mapped to which dimensions: 1 corresponds to first, 2 to second etc.

Details

Usually, you will not call this function directly, but will pass it to a method that works with tour paths like animate, save_history or render.

Examples

```
animate_xy(flea[, 1:3], dependence_tour(c(1, 2, 2)))
animate_xy(flea[, 1:4], dependence_tour(c(1, 2, 1, 2)))
animate_pcp(flea[, 1:6], dependence_tour(c(1, 2, 3, 2, 1, 3)))
```

display_andrews *Andrews' curves tour path animation.*

Description

Animate a nD tour path with Andrews' curves. For more details about Andrew's curves, see andrews

Usage

```
display_andrews(...)

animate_andrews(data, tour_path = grand_tour(3), ...)
```

Arguments

...	other arguments passed on to animate
data	matrix, or data frame containing numeric columns
tour_path	tour path generator, defaults to 2d grand tour

See Also

animate for options that apply to all animations

Examples

```
animate_andrews(flea[, 1:6])
animate_andrews(flea[, 1:6], grand_tour(d = 3))
animate_andrews(flea[, 1:6], grand_tour(d = 6))

# It's easy to experiment with different tour paths:
animate_andrews(flea[, 1:6], guided_tour(cmass))
```

display_depth *Display 3d projection with depth cues*

Description

Suggestion to use gray background and colour saturation (instead of gray shading) by Graham Wills.

Usage

```
display_depth(center = TRUE, half_range = NULL, ...)

animate_depth(data, tour_path = grand_tour(3), ...)
```

Arguments

center	should projected data be centered to have mean zero (default: TRUE). This pins the centre of the data to the same place, and makes it easier to focus on the shape.
half_range	half range to use when calculating limits of projected. If not set, defaults to maximum distance from origin to each row of data.
...	other arguments passed on to animate
data	matrix, or data frame containing numeric columns
tour_path	tour path generator, defaults to 2d grand tour

See Also

animate for options that apply to all animations

Examples

```
animate_depth(flea[, 1:6])
```

---

display_dist *1d distribution tour path animation.*

Description

Animate a 1d tour path with a density plot or histogram.

Usage

```
display_dist(method = "density", center = TRUE, half_range = NULL,
  rug = FALSE, ...)

animate_dist(data, tour_path = grand_tour(1), ...)
```

Arguments

method	display method, histogram or density plot
center	should 1d projection be centered to have mean zero (default: TRUE). This pins the centre of distribution to the same place, and makes it easier to focus on the shape of the distribution.
half_range	half range to use when calculating limits of projected. If not set, defaults to maximum distance from origin to each row of data.
rug	draw rug plot showing position of actual data points?
...	other arguments passed on to animate
data	matrix, or data frame containing numeric columns
tour_path	tour path generator, defaults to 2d grand tour

See Also

animate for options that apply to all animations

Examples

```
animate_dist(flea[, 1:6])

# When the distribution is not centred, it tends to wander around in a
# distracting manner
animate_dist(flea[, 1:6], center = FALSE)

# Alternatively, you can display the distribution with a histogram
animate_dist(flea[, 1:6], method = "hist")
```

---

display_faces *Chernoff faces tour path animation.*

Description

Animate a nD tour path with Chernoff's faces. Can display up to 18 dimensions.

Usage

```
display_faces(...)

animate_faces(data, tour_path = grand_tour(3), ...)
```

Arguments

...	other arguments passed on to animate
data	matrix, or data frame containing numeric columns
tour_path	tour path generator, defaults to 2d grand tour

Details

This function requires the TeachingDemos package to draw the Chernoff faces. See faces2 for more details.

See Also

animate for options that apply to all animations

Examples

```
# The drawing code is fairly slow, so this animation works best with a
# limited number of cases
animate_faces(flea[1:2, 1:6])
animate_faces(flea[1:4, 1:6])

animate_faces(flea[1:2, 1:6], grand_tour(5))
```

`display_image` *Image tour path animation.*

Description

Animate a 1d tour path with an image plot. This animation requires a different input data structure, a 3d array. The first two dimensions are locations on a grid, and the 3rd dimension gives the observations to be mixed with the tour.

Usage

```
display_image(xs, ys, ...)

animate_image(data, tour_path = grand_tour(1), ...)
```

Arguments

`xs`	x limit that is used in making the size of the plot
`ys`	y limit that is used in making the size of the plot
`...`	other arguments passed on to `animate`
`data`	matrix, or data frame containing numeric columns
`tour_path`	tour path generator, defaults to 2d grand tour

See Also

`animate` for options that apply to all animations

Examples

```
str(ozone)
animate_image(ozone)
```

display_pcp *Parallel coordinates tour path animation.*

Description

Animate a nD tour path with a parallel coordinates plot.

Usage

```
display_pcp(...)

animate_pcp(data, tour_path = grand_tour(3), ...)
```

Arguments

...	other arguments passed on to animate
data	matrix, or data frame containing numeric columns
tour_path	tour path generator, defaults to 2d grand tour

Details

The lines show the observations, and the points, the values of the projection matrix.

See Also

animate for options that apply to all animations

Examples

```
animate_pcp(flea[, 1:6], grand_tour(3))
animate_pcp(flea[, 1:6], grand_tour(5))
```

display_scatmat *Scatterplot matrix tour path animation.*

Description

Animate a nD tour path with a scatterplot matrix.

Usage

```
display_scatmat(...)

animate_scatmat(data, tour_path = grand_tour(3), ...)
```

Arguments

...	other arguments passed on to animate
data	matrix, or data frame containing numeric columns
tour_path	tour path generator, defaults to 2d grand tour

Details

The lines show the observations, and the points, the values of the projection matrix.

See Also

animate for options that apply to all animations

Examples

```
animate_scatmat(flea[, 1:6], grand_tour(2))
animate_scatmat(flea[, 1:6], grand_tour(6))
```

---

display_stars *Star glyph tour path animation.*

Description

Animate a nD tour path with star glyphs.

Usage

```
display_stars(...)

animate_stars(data, tour_path = grand_tour(3), ...)
```

Arguments

...	other arguments passed on to stars
data	matrix, or data frame containing numeric columns
tour_path	tour path generator, defaults to 2d grand tour

Details

Currently, scaling doesn't seem to be computed absolutely correctly, as centres move around as well as outside points.

See Also

animate for options that apply to all animations

Examples

```
animate_stars(flea[1:10, 1:6])
animate_stars(flea[1:10, 1:6], grand_tour(5))
animate_stars(flea[, 1:6], grand_tour(5))
animate_stars(flea[1:10, 1:6], grand_tour(5),
 col.stars = rep("grey50", 10), radius = FALSE)
```

---

display_stereo *Anaglpyh tour path animation.*

Description

Uses red-blue anaglyphs to display a 3d tour path. You'll need some red- blue glasses to get much out of this displays!

Usage

```
display_stereo(blue, red, ...)

animate_stereo(data, tour_path = grand_tour(3), blue = rgb(0, 0.91, 0.89),
  red = rgb(0.98, 0.052, 0), ...)
```

Arguments

blue	blue colour (for right eye)
red	red colour (for left eye)
...	other arguments passed on to animate
data	matrix, or data frame containing numeric columns
tour_path	tour path generator, defaults to 2d grand tour

Examples

```
animate_stereo(flea[, 1:6])
```

---

display_trails *Display tour path with trails*

Description

Animate a 2D tour path with a point trails

Usage

```
display_trails(center = TRUE, axes = "center", half_range = NULL,
  col = "black", pch = 20, past = 3, ...)

animate_trails(data, tour_path = grand_tour(), ...)
```

Arguments

center	if TRUE, centers projected data to (0,0). This pins the center of data cloud and make it easier to focus on the changing shape rather than position.
axes	position of the axes: center, bottomleft or off
half_range	half range to use when calculating limits of projected. If not set, defaults to maximum distance from origin to each row of data.
col	color to be plotted. Defaults to "black"
pch	size of the point to be plotted. Defaults to 20.
past	draw line between current projection and projection past steps ago
...	other arguments passed on to animate and display_xy
data	matrix, or data frame containing numeric columns
tour_path	tour path generator, defaults to 2d grand tour

---

display_xy *Display tour path with a scatterplot*

Description

Animate a 2D tour path with a scatterplot.

Usage

```
display_xy(center = TRUE, axes = "center", half_range = NULL,
  col = "black", pch = 20, edges = NULL, ...)

animate_xy(data, tour_path = grand_tour(), ...)
```

Arguments

center	if TRUE, centers projected data to (0,0). This pins the center of data cloud and make it easier to focus on the changing shape rather than position.
axes	position of the axes: center, bottomleft or off
half_range	half range to use when calculating limits of projected. If not set, defaults to maximum distance from origin to each row of data.
col	color to be plotted. Defaults to "black"
pch	size of the point to be plotted. Defaults to 20.

edges	A two column integer matrix giving indices of ends of lines.
...	other arguments passed on to animate and display_xy
data	matrix, or data frame containing numeric columns
tour_path	tour path generator, defaults to 2d grand tour

Examples

```
animate_xy(flea[, 1:6])
animate(flea[, 1:6], tour_path=grand_tour(), display=display_xy())
animate(flea[, 1:6], tour_path=grand_tour(),
  display=display_xy(axes = "bottomleft"))
animate(flea[, 1:6], tour_path=grand_tour(),
  display=display_xy(half_range = 0.5))
animate_xy(flea[, 1:6], tour_path=little_tour())
animate_xy(flea[, 1:3], tour_path=guided_tour(holes), sphere = TRUE)
animate_xy(flea[, 1:6], center = FALSE)

# The default axes are centered, like a biplot, but there are other options
animate_xy(flea[, 1:6], axes = "bottomleft")
animate_xy(flea[, 1:6], axes = "off")
animate_xy(flea[, 1:6], dependence_tour(c(1, 2, 1, 2, 1, 2)),
  axes = "bottomleft")
require(colorspace)
pal <- rainbow_hcl(length(levels(flea$species)))
col <- pal[as.numeric(flea$species)]
animate_xy(flea[,-7], col=col)

# You can also draw lines
edges <- matrix(c(1:5, 2:6), ncol = 2)
animate(flea[, 1:6], grand_tour(),
  display_xy(axes = "bottomleft", edges = edges))
```

Flea measurements *Flea beatle measurements*

Description

This data is from a paper by A. A. Lubischew, "On the Use of Discriminant Functions in Taxonomy", Biometrics, Dec 1962, pp.455-477.

Format

A 74 x 7 numeric array

Details

- tars1, width of the first joint of the first tarsus in microns (the sum of measurements for both tarsi)
- tars2, the same for the second joint
- head, the maximal width of the head between the external edges of the eyes in 0.01 mm
- ade1, the maximal width of the aedeagus in the fore-part in microns
- ade2, the front angle of the aedeagus (1 unit = 7.5 degrees)
- ade3, the aedeagus width from the side in microns
- species, which species is being examined - concinna, heptapotamica, heikertingeri

Examples

```
head(flea)
animate_xy(flea[,-7])
animate_xy(flea[,-7],col=flea[,7])
```

---

frozen_guided_tour *The frozen guided tour*

Description

The frozen guided tour

Usage

```
frozen_guided_tour(frozen, index_f, d = 2, max.tries = 25)
```

Arguments

frozen	matrix of frozen variables, as described in freeze
index_f	the index function to optimise.
d	target dimensionality
max.tries	the maximum number of unsuccessful attempts to find a better projection before giving up

See Also

cmass, holes and lda_pp for examples of index functions. The function should take a numeric matrix and return a single number, preferrably between 0 and 1.

Examples

```
frozen <- matrix(NA, nrow = 4, ncol = 2)
frozen[3, ] <- .5
animate_xy(flea[, 1:4], frozen_guided_tour(frozen, holes))
```

`frozen_tour` *A frozen tour path.*

Description

A frozen tour fixes some of the values of the orthonormal projection matrix and allows the others to vary freely according to any of the other tour methods. This frozen tour is a frozen grand tour. See `frozen_guided_tour` for a frozen guided tour.

Usage

```
frozen_tour(d = 2, frozen)
```

Arguments

`d`	target dimensionality
`frozen`	matrix of frozen variables, as described in `freeze`

Details

Usually, you will not call this function directly, but will pass it to a method that works with tour paths like `animate`, `save_history` or `render`.

Examples

```
frozen <- matrix(NA, nrow = 4, ncol = 2)
frozen[3, ] <- .5
animate_xy(flea[, 1:4], frozen_tour(2, frozen))

## Not run:
  # Doesn't work - a bug?
  frozen <- matrix(NA, nrow = 4, ncol = 2)
  frozen[1, 1] <- 0.5
  animate_xy(flea[, 1:4], frozen_tour(2, frozen))

  # Doesn't work - a bug?
  frozen <- matrix(NA, nrow = 4, ncol = 2)
  frozen[, 1] <- 1/2
  animate_xy(flea[, 1:4], frozen_tour(2, frozen))

  # Doesn't work - a bug?
  frozen[3, ] <- c(0, 1)
  animate_xy(flea[, 1:4], frozen_tour(2, frozen))

  # Doesn't move, which is correct - no free variables
  frozen[4, ] <- .2
  animate_xy(flea[, 1:4], frozen_tour(2, frozen))

  # Doesn't work - a bug?
```

```
  frozen <- matrix(NA, nrow = 4, ncol = 2)
  frozen[, 1] <- 1/2
  animate_xy(flea[, 1:4], frozen_tour(2, frozen))

## End(Not run)
# Two frozen variables in five 5.
frozen <- matrix(NA, nrow = 5, ncol = 2)
frozen[3, ] <- .5
frozen[4, ] <- c(-.2, .2)
animate_xy(flea[, 1:5], frozen_tour(2, frozen))
```

---

grand_tour *A grand tour path.*

Description

This method generates target bases by randomly sampling on the space of all d-dimensional planes in p-space.

Usage

```
grand_tour(d = 2)
```

Arguments

d	target dimensionality

Details

Usually, you will not call this function directly, but will pass it to a method that works with tour paths like animate, save_history or render.

Examples

```
# All animation methods use the grand tour path by default
animate_dist(flea[, 1:6])
animate_xy(flea[, 1:6])
animate_pcp(flea[, 1:6])
animate_pcp(flea[, 1:6], grand_tour(4))

# The grand tour is a function:
tour2d <- grand_tour(2)
is.function(tour2d)

# with two parameters, the previous projection and the data set
args(tour2d)
# if the previous projection is null, it will generate a starting
# basis, otherwise the argument is ignored
tour2d(NULL, mtcars)
# the data argument is just used to determine the correct dimensionality
```

```
# of the output matrix
tour2d(NULL, mtcars[, 1:2])
```

---

guided_tour *A guided tour path.*

Description

Instead of choosing new projections at random like the grand tour, the guided tour always tries to find a projection that is more interesting than the current projection.

Usage

```
guided_tour(index_f, d = 2, alpha = 0.5, cooling = 0.99, max.tries = 25,
  search_f = search_geodesic, ...)
```

Arguments

index_f	the index function to optimise.
d	target dimensionality
alpha	the initial size of the search window, in radians
cooling	the amount the size of the search window should be adjusted by after each step
max.tries	the maximum number of unsuccessful attempts to find a better projection before giving up
search_f	the search strategy to use
...	arguments sent to the search_f

Details

Currently the index functions only work in 2d.

Usually, you will not call this function directly, but will pass it to a method that works with tour paths like animate, save_history or render.

See Also

cmass, holes and lda_pp for examples of index functions. The function should take a numeric matrix and return a single number, preferrably between 0 and 1. search_geodesic, search_better, search_better_random for different search strategies

Examples

```
animate_xy(flea[, 1:3], guided_tour(holes), sphere = TRUE)
animate_xy(flea[, 1:6], guided_tour(holes), sphere = TRUE)
animate_dist(flea[, 1:6], guided_tour(holes, 1), sphere = TRUE)

# save_history is particularly useful in conjunction with the
# guided tour as it allows us to look at the tour path in many different
# ways
f <- flea[, 1:3]
tries <- replicate(5, save_history(f, guided_tour(holes)), simplify = FALSE)
```

---

holes *Holes index.*

Description

Calculates the holes index. See Cook and Swayne (2007) Interactive and Dynamic Graphics for Data Analysis for equations.

Usage

```
holes(mat)
```

Arguments

mat	matrix being used

---

interpolate *Interpolate geodesically between bases.*

Description

This function takes a set of bases and produces a tour by geodesically interpolating between each basis

Usage

```
interpolate(basis_set, angle = 0.05)
```

Arguments

basis_set	input basis set
angle	target distance (in radians) between bases

Examples

```
t1 <- save_history(flea[, 1:6], grand_tour(1), max = 10)
dim(t1)
dim(interpolate(t1, 0.01))
dim(interpolate(t1, 0.05))
dim(interpolate(t1, 0.1))
```

`Laser measurements` *Turnable laser measurements from Bellcore*

Description

This data came from an investigation of an experimental laser at Bellcore. It was a tunable laser, in the sense that both its wavelength and power output were controllable.

Format

A 64 x 4 numeric array

Details

Rotation helped the experimental physicists to characterize the laser, which turned out not to be a very good one, due to its unstable operating region.

This data initially came to the statistics research group when Janette Cooper asked Paul Tukey to help her analyze the data she had collected to describe the laser.

- ifront, current applied to the front of the laser
- iback, current applied to the back of the laser
- power, output power
- lambda, output wavelength

Examples

```
head(laser)
animate_xy(laser[,-4])
```

lda_pp *LDA projection pursuit index.*

Description

Calculate the LDA projection pursuit index. See Cook and Swayne (2007) Interactive and Dynamic Graphics for Data Analysis for equations.

Usage

```
lda_pp(cl)
```

Arguments

cl	class to be used. Such as "color"

little_tour *A little tour path.*

Description

The little tour is a planned tour that travels between all axis parallel projections. (John McDonald named this type of tour.)

Usage

```
little_tour(d = 2)
```

Arguments

d	target dimensionality

Details

Usually, you will not call this function directly, but will pass it to a method that works with tour paths like animate, save_history or render.

Examples

```
animate_xy(flea[, 1:6], little_tour())
animate_pcp(flea[, 1:6], little_tour(3))
animate_scatmat(flea[, 1:6], little_tour(3))
animate_pcp(flea[, 1:6], little_tour(4))
```

local_tour *A local tour path.*

Description

The local tour alternates between the starting position and a nearby random projection.

Usage

```
local_tour(start, angle = pi/4)
```

Arguments

start	initial projection matrix
angle	distance in radians to stay within

Details

Usually, you will not call this function directly, but will pass it to a method that works with tour paths like animate, save_history or render.

Examples

```
animate_xy(flea[, 1:3], local_tour(basis_init(3, 2)))
animate_xy(flea[, 1:3], local_tour(basis_init(3, 2), 0.2))
animate_xy(flea[, 1:3], local_tour(basis_random(3, 2), 0.2))
```

Olive oil measurements *Olive oil samples from Italy*

Description

This data is from a paper by Forina, Armanino, Lanteri, Tiscornia (1983) Classification of Olive Oils from their Fatty Acid Composition, in Martens and Russwurm (ed) Food Research and Data Anlysis. We thank Prof. Michele Forina, University of Genova, Italy for making this dataset available.

Format

A 572 x 10 numeric array

Details

- region Three super-classes of Italy: North, South and the island of Sardinia
- area Nine collection areas: three from North, four from South and 2 from Sardinia
- palmitic, palmitoleic, stearic, oleic, linoleic, linolenic, arachidic, eicosenoic fatty acids percent x 100

Examples

```
head(olive)
animate_xy(olive[,c(7,9,10)])
animate_xy(olive[,c(7,9,10)],col=olive[,1])
```

Ozone measurements *Monthly ozone measurements over Central America*

Description

This data set is a subset of the data from the 2006 ASA Data expo challenge, http://stat-computing.org/dataexpo/2006/. The data are monthly ozone averages on a very coarse 24 by 24 grid covering Central America, from Jan 1995 to Dec 2000. The data is stored in a 3d area with the first two dimensions representing latitude and longitude, and the third representing time.

Format

A 24 x 24 x 72 numeric array

References

http://stat-computing.org/dataexpo/2006/

Examples

```
example(display_image)
```

path_curves *Draw the path that the geodesics took.*

Description

This computes the projected values of each observation at each step, and allows you to recreate static views of the animated plots.

Usage

```
path_curves(history, data = attr(history, "data"))
```

Arguments

history	list of bases produced by save_history (or otherwise)
data	dataset to be projected on to bases

Examples

```
path1d <- save_history(flea[, 1:6], grand_tour(1), 10)
path2d <- save_history(flea[, 1:6], grand_tour(2), 10)

if (require("ggplot2")) {
plot(path_curves(path1d))
plot(path_curves(interpolate(path1d)))

plot(path_curves(path2d))
plot(path_curves(interpolate(path2d)))

# Instead of relying on the built in plot method, you might want to
# generate your own.  Here are few examples of alternative displays:

df <- path_curves(path2d)
qplot(step, value, data = df, group = obs:var, geom = "line", colour=var) + facet_wrap( ~ obs)

library(reshape2)
qplot(`1`, `2`, data = dcast(df, ... ~ var)) +
  facet_wrap( ~ step) +
  coord_equal()
}
```

path_dist *Compute distance matrix from bases.*

Description

Compute distance matrix from bases.

Usage

```
path_dist(history)
```

Arguments

history	history of the plots

Examples

```
## Not run:
 grand <- interpolate(save_history(flea[, 1:6], max = 50), 0.2)
 # The grand tour  ----------------------------
 # Look at the tour path in a tour, how well does it cover a sphere
 # Using MDS
 d <- path_dist(grand)
 ord <- as.data.frame(MASS::isoMDS(d)$points)
 qplot(V1, V2, data = ord, geom="path") +
 coord_equal() + labs(x = NULL, y = NULL)

## End(Not run)

# 5 guided tours  ----------------------------
holes1d <- guided_tour(holes, 1)
tries <- replicate(5, save_history(flea[, 1:6], holes1d, max = 10),
  simplify = FALSE)
tries2 <- lapply(tries, interpolate, 0.2)

bases <- unlist(lapply(tries2, as.list), recursive = FALSE)
class(bases) <- "history_list"
index_values <- paths_index(tries2, holes)
d <- path_dist(bases)
ord <- as.data.frame(cmdscale(d, 2))

info <- cbind(ord, index_values)
if (require("ggplot2")) {
qplot(step, value, data = info, geom="line", group = try)
qplot(V1, V2, data = info, geom="path", group = try) +
  geom_point(aes(size = value)) +
  coord_equal()
last_plot() + facet_wrap(~ try)
}
```

path_index *Compute index values for a tour history.*

Description

Compute index values for a tour history.

Usage

```
path_index(history, index_f, data = attr(history, "data"))
```

Arguments

history	list of bases produced by save_history (or otherwise)
index_f	index function to apply to each basis
data	dataset to be projected on to bases

See Also

save_history for options to save history

Examples

```
fl_holes <- save_history(flea[, 1:6], guided_tour(holes), sphere = TRUE)
path_index(fl_holes, holes)
path_index(fl_holes, cmass)

plot(path_index(fl_holes, holes), type = "l")
plot(path_index(fl_holes, cmass), type = "l")

# Use interpolate to show all intermediate bases as well
hi <- path_index(interpolate(fl_holes), holes)
hi
plot(hi)
```

pda_pp *PDA projection pursuit index.*

Description

Calculate the PDA projection pursuit index. See Lee and Cook (2009) A Projection Pursuit Index for Large p, Small n Data

Usage

```
pda_pp(cl, lambda = 0.2)
```

Arguments

`cl`	class to be used. Such as "color"
`lambda`	shrinkage parameter (0 = no shrinkage, 1 = full shrinkage)

`Places Ratings` *Ratings of different locations across North America*

Description

The "places data" were distribed to interested ASA members a few years ago so that they could apply contemporary data analytic methods to describe these data and then present results in a poster session at the ASA annual conference. Latitude and longitude have been added by Paul Tukey.

Format

A 329 x 14 numeric array

Details

The first dataset is taken from the Places Rated Almanac, by Richard Boyer and David Savageau, copyrighted and published by Rand McNally. This book order (SBN) number is 0-528-88008-X, and it retails for $14.95 . The data are reproduced on disk by kind permission of the publisher, and with the request that the copyright notice of Rand McNally, and the names of the authors appear in any paper or presentation using these data.

The nine rating criteria used by Places Rated Almanac are: Climate and Terrain Housing Health Care and Environment Crime Transportation Education The Arts Recreation Economics

For all but two of the above criteria, the higher the score, the better. For Housing and Crime, the lower the score the better.

The scores are computed using the following component statistics for each criterion (see the Places Rated Almanac for details):

Climate and Terrain: very hot and very cold months, seasonal temperature variation, heating- and cooling-degree days, freezing days, zero-degree days, ninety-degree days.

Housing: utility bills, property taxes, mortgage payments.

Health Care and Environment: per capita physicians, teaching hospitals, medical schools, cardiac rehabilitation centers, comprehensive cancer treatment centers, hospices, insurance/hospitalization costs index, flouridation of drinking water, air pollution.

Crime: violent crime rate, property crime rate.

Transportation: daily commute, public transportation, Interstate highways, air service, passenger rail service.

Education: pupil/teacher ratio in the public K-12 system, effort index in K-12, accademic options in higher education.

The Arts: museums, fine arts and public radio stations, public television stations, universities offering a degree or degrees in the arts, symphony orchestras, theatres, opera companies, dance companies, public libraries.

Recreation: good restaurants, public golf courses, certified lanes for tenpin bowling, movie theatres, zoos, aquariums, family theme parks, sanctioned automobile race tracks, pari-mutuel betting attractions, major- and minor- league professional sports teams, NCAA Division I football and basketball teams, miles of ocean or Great Lakes coastline, inland water, national forests, national parks, or national wildlife refuges, Consolidated Metropolitan Statistical Area access.

Economics: average household income adjusted for taxes and living costs, income growth, job growth.

Examples

```
head(places)
animate_xy(places[,1:9])
```

`planned_tour` *A planned tour path.*

Description

The planned tour takes you from one basis to the next in a set order. Once you have visited all the planned bases, you either stop or start from the beginning once more (if `cycle = TRUE`).

Usage

```
planned_tour(basis_set, cycle = FALSE)
```

Arguments

`basis_set`	the set of bases as a list of projection matrices or a 3d array
`cycle`	cycle through continuously (`TRUE`) or stop after first pass (`FALSE`)

Details

Usually, you will not call this function directly, but will pass it to a method that works with tour paths like `animate`, `save_history` or `render`.

See Also

The `little_tour`, a special type of planned tour which cycles between all axis parallel projections.

Examples

```
twod <- save_history(flea[, 1:3], max = 5)
str(twod)
animate_xy(flea[, 1:3], planned_tour(twod))
animate_xy(flea[, 1:3], planned_tour(twod, TRUE))

oned <- save_history(flea[, 1:6], grand_tour(1), max = 3)
animate_dist(flea[, 1:6], planned_tour(oned))
```

---

proj_dist *Calculate the distance between two bases.*

Description

Computes the Frobenius norm between two bases, in radians. This is equals to the Euclidean norm of the vector of principal angles between the two subspaces.

Usage

```
proj_dist(x, y)
```

Arguments

x	projection matrix a
y	projection matrix b

Rat CNS *Rat CNS Gene Expression*

Description

Columns:

Format

A 112 x 11 numeric array

Details

e11 e13 e15 e18 e21 p0 p7 p14 a class1 class2

- e11, an ebryonic timepoint from the original data with the number corresponding to the day
- e13, an ebryonic timepoint from the original data with the number corresponding to the day
- e15, an ebryonic timepoint from the original data with the number corresponding to the day
- e18, an ebryonic timepoint from the original data with the number corresponding to the day
- e21, an ebryonic timepoint from the original data with the number corresponding to the day
- p0, a postnatal timpoint from the original data with the number corresponding to the day
- p7, a postnatal timpoint from the original data with the number corresponding to the day
- p14, a postnatal timpoint from the original data with the number corresponding to the day
- a, a postnatal timpoint from the original data. It is equivalent to p90.
- class1, is the high-level class: its range is 1:4
- class2, breaks down the high-level classes, so its range is 1:14

Rows: Each case is a gene (or gene family?) And each cell is the gene expression level for that gene at time t, averaging a few measured values and normalizing using the maximum expression value for that gene.

Reference (available on the web at pnas.org): Large-scale temporal gene expression mapping of central nervous system development by X. Wen, S. Fuhrman, G. S. Michaels, D. B. Carr, S. Smith, J. L. Barker, R. Somogyi in the Proceedings of the National Academy of Science, Vol 95, pp. 334-339, January 1998

References

http://www.pnas.org

Examples

```
head(ratcns)
animate_xy(ratcns[,1:8],col=ratcns[,10])
```

---

render *Render frames of animation to disk*

Description

Render frames of animation to disk

Usage

```
render(data, tour_path, display, dev, ..., apf = 1/10, frames = 50,
  rescale = TRUE, sphere = FALSE, start = NULL)
```

Arguments

data	matrix, or data frame containing numeric columns
tour_path	tour path generator
display	the method used to render the projected data, e.g. display_xy, display_pcp
dev	name of output device to use (e.g. png, pdf)
...	other options passed to output device
apf	angle (in radians) per frame
frames	number of frames in output
rescale	if true, rescale all variables to range [0,1]
sphere	if true, sphere all variables
start	starting projection. If NULL, uses path default.

References

Hadley Wickham, Dianne Cook, Heike Hofmann, Andreas Buja (2011). tourr: An R Package for Exploring Multivariate Data with Projections. Journal of Statistical Software, 40(2), 1-18. http://www.jstatsoft.org/v40/i02/.

Examples

```
render(flea[, 1:4], grand_tour(), display_xy(), "pdf", "test.pdf")
render(flea[, 1:4], grand_tour(), display_xy(), "png", "test-%03d.png")
```

---

rescale *Rescale a matrix or data frame*

Description

Standardise each column to have range [0, 1]

Usage

```
rescale(df)
```

Arguments

df	data frame or matrix

save_history *Save tour history.*

Description

Save a tour path so it can later be displayed in many different ways.

Usage

```
save_history(data, tour_path = grand_tour(), max_bases = 100,
  start = NULL, rescale = TRUE, sphere = FALSE, step_size = Inf)
```

Arguments

data	matrix, or data frame containing numeric columns
tour_path	tour path generator
max_bases	maximum number of new bases to generate. Some tour paths (like the guided tour) may generate less than the maximum.
start	starting projection, if you want to specify one
rescale	if true, rescale all variables to range [0,1]?
sphere	if true, sphere all variables
step_size	distance between each step - defaults to Inf which forces new basis generation at each step.

References

Hadley Wickham, Dianne Cook, Heike Hofmann, Andreas Buja (2011). tourr: An R Package for Exploring Multivariate Data with Projections. Journal of Statistical Software, 40(2), 1-18. http://www.jstatsoft.org/v40/i02/.

Examples

```
# You can use a saved history to replay tours with different visualisations

t1 <- save_history(flea[, 1:6], max = 3)
animate_xy(flea[, 1:6], planned_tour(t1))
##andrews_history(t1)
##andrews_history(interpolate(t1))

t1 <- save_history(flea[, 1:6], grand_tour(4), max = 3)
animate_pcp(flea[, 1:6], planned_tour(t1))
animate_scatmat(flea[, 1:6], planned_tour(t1))

t1 <- save_history(flea[, 1:6], grand_tour(1), max = 3)
animate_dist(flea[, 1:6], planned_tour(t1))

testdata <- matrix(rnorm(100*3), ncol=3)
```

```
testdata[1:50, 1] <- testdata[1:50, 1] + 10
testdata <- sphere(testdata)
t2 <- save_history(testdata, guided_tour(holes, max.tries = 100),
  max = 5, rescale=FALSE)
animate_xy(testdata, planned_tour(t2))

# Or you can use saved histories to visualise the path that the tour took.
plot(path_index(interpolate(t2), holes))
plot(path_curves(interpolate(t2)))
```

`search_geodesic` *Search for most interesting projection along random geodesics.*

Description

This is a novel method for finding more interesting projections for the guided tour. It works by first taking a small step in `n` random directions, and then picking the direction that looks most promising (based on the height of the index function). Once the best direction is found, it performs a linear search along the geodesic in that direction, traveling up to half way around the sphere.

Usage

```
search_geodesic(current, alpha = 1, index, max.tries = 5, n = 5)
```

Arguments

`current`	starting projection
`alpha`	maximum distance to travel (currently ignored)
`index`	interestingness index function
`max.tries`	maximum number of failed attempts before giving up
`n`	number of random steps to take to find best direction

Details

You should not to have call this function directly, but should supply it to the `guided_tour` as a search strategy.

sphere *Sphere a matrix (or data frame) by transforming variables to principal components.*

Description

Sphering is often useful in conjunction with the guided tour, as it removes simpler patterns that may conceal more interesting findings.

Usage

```
sphere(df)
```

Arguments

df	data frame or matrix

Tropical Atmosphere Ocean *Tropical Atmosphere Ocean data*

Description

This is a subset of data taken from the NOAA web site http://www.pmel.noaa.gov/tao/. The data is generated from recording instruments on a grid of buoys laid out over the Pacific Ocean. The grid was setup to monitor El Nino and La Nina events. This subset contains measurements from 5 locations (0deg/110W, 2S/110W, 0deg/95W,2S/95W,5S/95W) and two time points Nov-Jan 1993 (normal), 1997 (El Nino). There are missing values in this data set, which need to be removed, or imputed before running a tour.

Format

A 736 x 8 numeric array

References

http://www.pmel.noaa.gov/tao/

Index