Package ‘tourr’

August 25, 2023

Title  Tour Methods for Multivariate Data Visualisation
Version  1.1.0
Description  Implements geodesic interpolation and basis
generation functions that allow you to create new tour
methods from R.
Depends  R (>= 3.6)
Imports  MASS, tibble, dplyr, stats, utils, grDevices
Suggests  TeachingDemos, ash, energy, testthat, colorspace, ggplot2,
          mgcv, gifski, geozoo, knitr, rmarkdown, tidyr, covr, plotly
License  MIT + file LICENSE
LazyData  true
URL  https://github.com/ggobi/tourr
BugReports  https://github.com/ggobi/tourr/issues
RoxygenNote  7.2.3
Encoding  UTF-8
VignetteBuilder  knitr
NeedsCompilation  no
Author  Hadley Wickham [aut, ctb] (<https://orcid.org/0000-0003-4757-117X>),
        Dianne Cook [aut, cre] (<https://orcid.org/0000-0002-3813-7155>),
        Nick Spyrison [ctb] (<https://orcid.org/0000-0002-8417-0212>),
        Ursula Laa [ctb] (<https://orcid.org/0000-0002-0249-6439>),
        H. Sherry Zhang [ctb] (<https://orcid.org/0000-0002-7122-1463>),
        Stuart Lee [ctb] (<https://orcid.org/0000-0003-1179-8436>)
Maintainer  Dianne Cook <dicook@monash.edu>
Repository  CRAN
Date/Publication  2023-08-24 23:40:02 UTC
R topics documented:

- anchored_orthogonal_distance ........................................ 3
- andrews ................................................................. 4
- angular_breaks ........................................................... 4
- animate ................................................................. 5
- areColors ............................................................... 6
- center ................................................................. 6
- cmass ................................................................. 7
- dcor2d .................................................................. 7
- dependence_tour .......................................................... 7
- display_andrews .......................................................... 8
- display_density2d ........................................................ 9
- display_depth ............................................................ 10
- display_dist ............................................................. 11
- display_faces ............................................................ 13
- display_groupxy .......................................................... 14
- display_idx .............................................................. 15
- display_image ............................................................ 17
- display_pca .............................................................. 18
- display_pcp .............................................................. 19
- display_sage ............................................................. 20
- display_scatmat .......................................................... 21
- display_slice ............................................................. 22
- display_stars ............................................................. 23
- display_stereo ............................................................ 24
- display_trails ............................................................. 25
- display_xy ............................................................... 26
- draw_tour_axes ............................................................ 28
- estimate_eps ............................................................... 28
- Flea measurements ....................................................... 29
- frozen_guided_tour ....................................................... 30
- frozen_tour .............................................................. 30
- grand_tour ............................................................... 31
- guided_section_tour ..................................................... 32
- guided_tour ............................................................... 34
- holes ................................................................. 35
- interpolate ............................................................. 36
- Laser measurements ..................................................... 36
- lda_pp ................................................................. 37
- linear_breaks ............................................................ 37
- little_tour ............................................................... 38
- local_tour .............................................................. 38
- manual_slice ............................................................. 39
- mapColors .............................................................. 40
- mapShapes .............................................................. 41
- norm_bin ............................................................... 41
- Olive oil measurements ................................................ 42
anchored_orthogonal_distance

Calculate orthogonal distances

Description

For each datapoint this function calculates the orthogonal distance from the anchored projection plane.

Usage

anchored_orthogonal_distance(plane, data, anchor = NULL)

Arguments

plane          matrix specifying the projection plane
data          data frame or matrix
anchor         A vector specifying the reference point to anchor the plane If NULL (default) the slice will be anchored at the origin.
**andrews**  
*Compute Andrews' curves*

**Value**

distance vector

**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**

```r
andrews(x)
```

**Arguments**

`x`  
input a new parameter

**Value**

a function with single argument, theta

**Examples**

```r
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")
```

---

**angular_breaks**  
*Returns n equidistant bins between -pi and pi*

**Description**

Returns n equidistant bins between -pi and pi

**Usage**

```r
angular_breaks(n)
```

**Arguments**

`n`  
number of bins
**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**

```r
animate(
  data,
  tour_path = grand_tour(),
  display = display_xy(),
  start = NULL,
  aps = 1,
  fps = 10,
  max_frames = Inf,
  rescale = FALSE,
  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 15, to accommodate RStudio graphics device)
- `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` Default FALSE. 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

Examples

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

animate(f, max_frames = 10, fps = 1, aps = 0.1)

areColors

Test if all entries are colors

Description

Test if all entries are colors

Usage

areColors(x)

Arguments

x  vector

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

Usage
cmass()


dcor2d

Distance correlation index.

Description
Computes the distance correlation based index on 2D projections of the data.

Usage
dcor2d()

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(col = "black", palette = "Zissou 1", ...)
animate_andrews(data, tour_path = grand_tour(3), col = "black", ...)

Arguments

col color to be plotted. Defaults to "black"
palette name of color palette for point colour, used by hcl.colors, default "Zissou 1"
... 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(cmss()))
Description

Animate a 2D tour path with density contour(s) and a scatterplot.

Usage

```r
display_density2d(
  center = TRUE,
  axes = "center",
  half_range = NULL,
  col = "black",
  pch = 20,
  cex = 1,
  contour_quartile = c(0.25, 0.5, 0.75),
  edges = NULL,
  palette = "Zissou 1",
  ...
)

animate_density2d(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 use for points, can be a vector or hexcolors or a factor. Defaults to "black".

pch shape of the point to be plotted. Defaults to 20.

cex size of the point to be plotted. Defaults to 1.

contour_quartile Vector of quartiles to plot the contours at. Defaults to 5.

edges A two column integer matrix giving indices of ends of lines.

palette name of color palette for point colour, used by hcl.colors, default "Zissou 1"

... other arguments passed on to animate and display_density2d
data matrix, or data frame containing numeric columns
tour_path tour path generator, defaults to 2d grand tour
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), ...)
**display_dist**

*1d distribution tour path animation.*

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

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

```r
display_dist(
    method = "density",
    center = TRUE,
    half_range = NULL,
    col = "black",
    rug = FALSE,
    palette = "Zissou 1",
    density_max = 3,
    bw = 0.2,
    scale_density = FALSE,
    ...
)
```

```r
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.
col: color to use for points, can be a vector or hexcolors or a factor. Defaults to "black".
rug: draw rug plot showing position of actual data points?
palette: name of color palette for point colour, used by `hcl.colors`, default "Zissou 1"
density_max: allow control of the y range for density plot
bw: binwidth for histogram and density, between 0-1, default 0.2
scale_density: Height of density is scaled at each projection, default FALSE
... 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

```r
animate_dist(flea[, 1:6])
# Change inputs, to color by group, fix y axis, change bin width # and scale bar height or density at each projection
animate_dist(flea[, 1:6], col=flea$species, density_max=5)
animate_dist(flea[, 1:6], col=flea$species, density_max=5, bw=0.1)
animate_dist(flea[, 1:6], col=flea$species, scale_density=TRUE)

# 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  
flea_s <- rescale(flea[,1:6])  
animate_faces(flea_s[1:2, 1:6])  
animate_faces(flea_s[1:4, 1:6])  
animate_faces(flea_s[1:2, 1:6], grand_tour(5))
```
**display_groupxy**

Display 2D tour projections displayed separately by groups

**Description**

This function is designed to allow comparisons across multiple groups, especially for examining things like two (or more) different models on the same data. The primary display is a scatterplot, with lines or contours overlaid.

**Usage**

```r
display_groupxy(
  centr = TRUE,
  axes = "center",
  half_range = NULL,
  col = "black",
  pch = 20,
  cex = 1,
  edges = NULL,
  edges.col = "black",
  edges.width = 1,
  group_by = NULL,
  plot_xgp = TRUE,
  palette = "Zissou 1",
  ...
)
```

```r
animate_groupxy(data, tour_path = grand_tour(), ...)
```

**Arguments**

- `centr` 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 use for points, can be a vector or hexcolors or a factor. Defaults to "black".
- `pch` shape of the point to be plotted. Defaults to 20.
- `cex` size of the point to be plotted. Defaults to 1.
- `edges` A two column integer matrix giving indices of ends of lines.
- `edges.col` colour of edges to be plotted. Defaults to "black".
- `edges.width` line width for edges, default 1
- `group_by` variable to group by. Must have less than 25 unique values.
plot_xgp | if TRUE, plots points from other groups in light grey
palette | name of color palette for point colour, used by `hcl.colors`, default "Zissou 1"
... | other arguments passed on to `animate` and `display_groupxy`
data | matrix, or data frame containing numeric columns
tour_path | tour path generator, defaults to 2d grand tour

Examples

```r
animate_groupxy(flea[, 1:6], col = flea$species, pch = flea$species, group_by = flea$species)
animate_groupxy(flea[, 1:6], col = flea$species, pch = flea$species, group_by = flea$species, plot_xgp = FALSE)
```

```r
# Edges example
x <- data.frame(x1=runif(10, -1, 1), x2=runif(10, -1, 1), x3=runif(10, -1, 1))
x$cl <- factor(c(rep("A", 3), rep("B", 3), rep("C", 4)))
x.edges <- cbind(from=c(1,2, 4,5, 7,8,9), to=c(2,3, 5,6, 8,9,10))
x.edges.col <- factor(c(rep("A", 2), rep("B", 2), rep("C", 3)))
animate_groupxy(x[,1:3], col=x$cl, group_by=x$cl, edges=x.edges, edges.col=x.edges.col)
```

### display_idx

Display a 1D linear aggregation index

**Description**

Animate a 1D tour path for data where individuals are ranked by a multivariate index. Allows one to examine the sensitivity of the ranking on the linear combination. Variables should be scaled to be between 0-1. This is only designed to work with a local tour, or a radial tour.

**Usage**

```r
display_idx(
  center = FALSE,
  half_range = NULL,
  abb_vars = TRUE,
  col = "red",
  cex = 3,
  panel_height_ratio = c(3, 2),
  label_x_pos = 0.7,
  label = NULL,
  label_cex = 1,
  label_col = "grey80",
  add_ref_line = TRUE,
  axis_bar_col = "#000000",
  axis_bar_lwd = 3,
  axis_label_cex_upper = 1,
  axis_label_cex_lower = 1,
)```
animate_idx(data, tour_path = grand_tour(1), ...)

Arguments

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.

abb_vars logical, whether to abbreviate the variable name, if long
col the color used for points, can be a vector or hexcolors or a factor, default to
"red".
cex the size used for points, default to 0.5
panel_height_ratio input to the height argument in [graphics::layout()] for the height of data and
axis panel.

label_x_pos the x position of text label, currently labels are positioned at a fixed x value for
each observation
label the text label, a vector
label_cex the size for text labels
label_col the color for text labels
add_ref_line whether to add a horizontal reference line for each observation, logical default
to TRUE
axis_bar_col the color of the axis bar
axis_bar_lwd the width of the axis bar
axis_label_cex_upper the size of the axis label in the upper panel
axis_label_cex_lower the size of the axis label in the lower panel
axis_bar_label_cex the size of the axis label
axis_bar_label_col the color of the axis label
axis_var_cex the size of the variable name to the right of the axis panel
axis_var_col the color of the variable name to the right of the axis panel
palette name of color palette for point colour, used by hcl.colors, default "Zissou 1"
... ignored
data matrix, or data frame containing numeric columns
tour_path tour path generator, defaults to 2d grand tour

Examples

```r
data(places)
places_01 <- apply(places[1:10,1:9], 2, function(x) (x-min(x))/(max(x)-min(x)))
b <- matrix(rep(1/sqrt(9), 9), ncol=1)
places_init <- cbind(places_01, idx = as.vector(as.matrix(places_01) %*% b))
places_sorted <- places_init[order(places_init[,10]), 1:9]
animate_idx(places_sorted, tour_path = local_tour(b, angle=pi/8),
  label=as.character(places$stnum[1:9]),
  label_x_pos = 0)
```

---

**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**

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

```r
str(ozone)
animate_image(ozone)
```
display_pca

Display tour path with principal component scores with original axes

Description

Animate a 2D tour path on data that has been transformed into principal components, and also show the original variable axes.

Usage

display_pca(
  center = TRUE,
  axes = "center",
  half_range = NULL,
  col = "black",
  pch = 20,
  cex = 1,
  pc_coefs = NULL,
  edges = NULL,
  edges.col = "black",
  palette = "Zissou 1",
  ...
)

animate_pca(data, tour_path = grand_tour(), rescale = FALSE, ...)

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 use for points, can be a vector or hexcolors or a factor. Defaults to "black".
pch shape of the point to be plotted. Defaults to 20.
cex size of the point to be plotted. Defaults to 1.
pc_coefs coefficients relating the original variables to principal components. This is required.
edges A two column integer matrix giving indices of ends of lines.
edges.col colour of edges to be plotted, Defaults to "black.
palette name of color palette for point colour, used by hcl.colors, default "Zissou 1"
... other arguments passed on to animate and display_slice
data matrix, or data frame containing numeric columns
**display_pcp**

  tour_path  
  rescale  

**Examples**

```r
flea_std <- apply(flea[, 1:6], 2, function(x) (x-mean(x))/sd(x))
flea_pca <- prcomp(flea_std, center = FALSE, )
flea_coefs <- flea_pca$rotation[, 1:3]
flea_scores <- flea_pca$x[, 1:3]
animate_pca(flea_scores, pc_coefs = flea_coefs)
```

---

**Description**

Animate a nD tour path with a parallel coordinates plot.

**Usage**

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

**Arguments**

- `...`  
- `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**

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

Display tour path with a sage scatterplot

Description

Animate a 2D tour path with a sage scatterplot that uses a radial transformation on the projected points to re-allocate the volume projected across the 2D plane.

Usage

display_sage(
  axes = "center",
  half_range = NULL,
  col = "black",
  pch = 20,
  gam = 1,
  R = NULL,
  palette = "Zissou 1",
  ...
)

animate_sage(data, tour_path = grand_tour(), ...)

Arguments

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 use for points, can be a vector or hexcolors or a factor. Defaults to "black".
pch marker for points. Defaults to 20.
gam scaling of the effective dimensionality for rescaling. Defaults to 1.
R scale for the radial transformation. If not set, defaults to maximum distance from origin to each row of data.
palette name of color palette for point colour, used by hcl.colors, default "Zissou 1"
... other arguments passed on to animate and display_sage
data matrix, or data frame containing numeric columns
tour_path tour path generator, defaults to 2d grand tour

Examples

# Generate uniform samples in a 10d sphere using the geozoo package
sphere10 <- geozoo::sphere.solid.random(10)$points
# Columns need to be named before launching the tour
colnames(sphere10) <- paste0("x", 1:10)
# Standard grand tour display, points cluster near center
animate_xy(sphere10)

# Sage display, points are uniformly distributed across the disk
animate_sage(sphere10)

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_slice

Display tour path with a sliced scatterplot

Description

Animate a 2D tour path with a sliced scatterplot.

Usage

display_slice(
  center = TRUE,
  axes = "center",
  half_range = NULL,
  col = "black",
  pch_slice = 20,
  pch_other = 46,
  cex_slice = 2,
  cex_other = 1,
  v_rel = NULL,
  anchor = NULL,
  anchor_nav = "off",
  edges = NULL,
  edges.col = "black",
  palette = "Zissou 1",
  ...
)

animate_slice(data, tour_path = grand_tour(), rescale = FALSE, ...)

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 use for points, can be a vector or hexcolors or a factor. Defaults to "black".

pch_slice marker for plotting points inside the slice. Defaults to 20.

pch_other marker for plotting points outside the slice. Defaults to 46.

cex_slice size of the points inside the slice. Defaults to 2.

cex_other size if the points outside the slice. Defaults to 1.

v_rel relative volume of the slice. If not set, suggested value is calculated and printed to the screen.
display_stars

anchor A vector specifying the reference point to anchor the slice. If NULL (default) the slice will be anchored at the data center.
anchor_nav position of the anchor: center, topright or off
edges A two column integer matrix giving indices of ends of lines.
edges.col colour of edges to be plotted, Defaults to "black.
palette name of color palette for point colour, used by hcl.colors, default "Zissou 1"
... other arguments passed on to animate and display_slice
data matrix, or data frame containing numeric columns
tour_path tour path generator, defaults to 2d grand tour
rescale Default FALSE. If TRUE, rescale all variables to range [0,1].

Examples

# Generate samples on a 3d and 5d hollow sphere using the geozoo package
sphere3 <- geozoo::sphere.hollow(3)$points
sphere5 <- geozoo::sphere.hollow(5)$points

# Columns need to be named before launching the tour
colnames(sphere3) <- c("x1", "x2", "x3")
colnames(sphere5) <- c("x1", "x2", "x3", "x4", "x5")

# Animate with the slice display using the default parameters
animate_slice(sphere3)
animate_slice(sphere5)

# Animate with off-center anchoring
anchor3 <- matrix(rep(0.7, 3), ncol=3)
anchor5 <- matrix(rep(0.3, 5), ncol=5)
animate_slice(sphere3, anchor = anchor3)
animate_slice(sphere5, anchor = anchor5, v_rel = 0.02)

---

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 Anaglyph 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, cex = 1, ...)

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

Arguments

- **blue**: blue colour (for right eye)
- **red**: red colour (for left eye)
- **cex**: size of the point to be plotted. Defaults to 1.
- **...**: 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

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

display_trails  Display tour path with trails

Description

Animate a 2D tour path with a point trails

Usage

```r
display_trails(
  center = TRUE,
  axes = "center",
  half_range = NULL,
  col = "black",
  pch = 20,
  cex = 1,
  past = 3,
  ...
)
```

```r
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**: shape of the point to be plotted. Defaults to 20.
- **cex**: magnification of plotting text relative to default. Defaults to 1.
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

Examples

animate_trails(flea[,1:6], col=flea$species)

---

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,
  cex = 1,
  edges = NULL,
  edges.col = "black",
  edges.width = 1,
  obs_labels = NULL,
  palette = "Zissou 1",
  ...
)

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 use for points, can be a vector or hexcolors or a factor. Defaults to "black".


**display_xy**

- **pch** shape of the point to be plotted, can be a factor or integer. Defaults to 20.
- **cex** size of the point to be plotted. Defaults to 1.
- **edges** A two column integer matrix giving indices of ends of lines.
- **edges.col** colour of edges to be plotted, Defaults to "black"
- **edges.width** line width for edges, default 1
- **obs_labels** vector of text labels to display
- **palette** name of color palette for point colour, used by `hcl.colors`, default "Zissou 1"
- **...** 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**

```r
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(),
   scale = TRUE
)
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"
)

animate_xy(flea[, -7], col = flea$species)
animate_xy(flea[, -7], col = flea$species,
   pch = flea$species)
animate_xy(flea[, -7], col = flea$species,
   obs_labels=as.character(1:nrow(flea)), axes="off")

# 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)
)
```
**draw_tour_axes**

*Draw tour axes on the projected data with base graphics*

**Description**

Draw tour axes on the projected data with base graphics

**Usage**

draw_tour_axes(proj, labels, limits = 1, position = "center", ...)

**Arguments**

- **proj**: matrix of projection coefficients
- **labels**: variable names for the axes, of length the same as the number of rows of proj
- **limits**: value setting the lower and upper limits of projected data, default 1
- **position**: position of the axes: center (default), bottomleft or off
- ... other arguments passed

**Examples**

data(flea)
flea_std <- apply(flea[,1:6], 2, function(x) (x-mean(x))/sd(x))
prj <- basis_random(ncol(flea[,1:6]), 2)
flea_prj <- as.data.frame(as.matrix(flea_std) %*% prj)
par(pty = "s", mar = rep(0.1, 4))
plot(flea_prj$V1, flea_prj$V2, 
    xlim = c(-3, 3), ylim = c(-3, 3),
    xlab="P1", ylab="P2")
draw_tour_axes(prj, colnames(flea)[1:6], limits=3)

plot(flea_prj$V1, flea_prj$V2, 
    xlim = c(-3, 3), ylim = c(-3, 3),
    xlab="P1", ylab="P2")
draw_tour_axes(prj, colnames(flea)[1:6], limits=3, position="bottomleft")

---

**estimate_eps**

*Estimate cutoff eps for section pursuit.*

**Description**

Estimate cutoff eps for section pursuit.

**Usage**

estimate_eps(N, p, res, K, K_theta, r_breaks)
Arguments

- \( N \) total number of points in the input data.
- \( p \) number of dimensions of the input data.
- \( \text{res} \) resolution, \((\text{slice radius})/(\text{data radius})\)
- \( K \) total number of bins
- \( K_{\text{theta}} \) number of angular bins
- \( r_{\text{breaks}} \) boundaries of the radial bins

---

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

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

Examples

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

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

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

```r
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

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

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[1:2, ] <- 1 / 4
animate_xy(flea[, 1:4], frozen_tour(2, frozen))

## Not run:
# This freezes one entire direction which causes a problem,
# and is caught by error handling.
# If you want to do this it would be best with a dependence
# tour, with one variable set one axis, eg 3rd variable to
# x axis would be indicated from the code below
frozen <- matrix(NA, nrow = 4, ncol = 2)
frozen[3, ] <- c(0, 1)
animate_xy(flea[, 1:4], frozen_tour(2, frozen))

## End(Not run)

# Two frozen variables in five.
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
- `...` : arguments sent to the generator

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_section_tour  

A guided section tour path.

Description

The guided section tour is a variation of the guided tour that is using a section pursuit index for the selection of target planes.

Usage

```
guided_section_tour(
  index_f,
  d = 2,
  alpha = 0.5,
)  
```
cooling = 0.99,
max.tries = 25,
max.i = Inf,
v_rel = NULL,
anchor = NULL,
search_f = search_geodesic,
...
)

Arguments

index_f: the section pursuit index function to optimise. The function needs to take three arguments, the projected data, the vector of distances from the current projection plane, and the slice thickness h.
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
max.i: the maximum index value, stop search if a larger value is found
v_rel: relative volume of the slice. If not set, suggested value is calculated and printed to the screen.
anchor: A vector specifying the reference point to anchor the slice. If NULL (default) the slice will be anchored at the data center.
search_f: the search strategy to use
...: arguments sent to the search_f

Details

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

See Also

slice_index for an example of an index functions. search_geodesic, search_better, search_better_random for different search strategies

Examples

# Generate samples on a 3d hollow sphere using the geozoo package
set.seed(12345)
sphere3 <- geozoo::sphere.hollow(3)$points
# Columns need to be named before launching the tour
colnames(sphere3) <- c("x1", "x2", "x3")
# Off-center anchoring
anchor3 <- matrix(rep(0.75, 3), ncol=3)
# Index setup
r_breaks <- linear_breaks(5, 0, 1)
a_breaks <- angular_breaks(10)
eps <- estimate_eps(nrow(sphere3), ncol(sphere3), 0.1 / 1, 5 * 10, 10, r_breaks)
idx <- slice_index(r_breaks, a_breaks, eps, bintype = "polar", power = 1, reweight = TRUE, p = 3)
# Running the guided section tour select sections showing a big hole in the center
animate_slice(sphere3, guided_section_tour(idx, v_rel = 0.1, anchor = anchor3, max.tries = 5),
             v_rel = 0.1, anchor = anchor3)

---

**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**

```r
guided_tour(
  index_f,
  d = 2,
  alpha = 0.5,
  cooling = 0.99,
  max.tries = 25,
  max.i = Inf,
  search_f = search_geodesic,
  n_sample = 100,
  ...
)
```

**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
- `max.i`: the maximum index value, stop search if a larger value is found
- `search_f`: the search strategy to use: `search_geodesic`, `search_better`, `search_better_random`, `search_polish`. Default is `search_geodesic`.
- `n_sample`: number of samples to generate if `search_f` is `search_polish`
- `...`: arguments sent to the `search_f`
holes

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, preferably between 0 and 1. search_geodesic, search_better, search_better_random for different search strategies

Examples

flea_std <- apply(flea[,1:6], 2, function(x) (x-mean(x))/sd(x))
animate_xy(flea_std, guided_tour(holes()), sphere = TRUE)

animate_xy(flea_std, guided_tour(holes(), search_f = search_better_random), sphere = TRUE)
animate_dist(flea_std, guided_tour(holes(), 1), sphere = TRUE)
animate_xy(flea_std, guided_tour(lda_pp(flea$species)), sphere = TRUE, col = flea$species)

# 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_std[, 1:3]
tries <- replicate(5, save_history(f, guided_tour(holes())), simplify = FALSE)

holes

Holes index.

Description


Usage

holes()
Laser measurements

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, cycle = FALSE)

Arguments

- basis_set: input basis set
- angle: target distance (in radians) between bases
- cycle: For planned_tour cycle through continuously (TRUE) or stop after first pass (FALSE)

Examples

t1 <- save_history(flea[, 1:6], grand_tour(1), max = 3)
dim(t1)
dim(interpolate(t1, 0.01))
dim(interpolate(t1, 0.05))
dim(interpolate(t1, 0.1))
t2 <- save_history(flea[, 1:6], grand_tour(2), max = 2)
dim(interpolate(t2, 0.05))

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**

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

---

**Description**


**Usage**

```r
lda_pp(cl)
```

**Arguments**

- `cl` : class to be used. Such as "color"

**linear_breaks**

*Returns n equidistant bins between a and b*

**Description**

Returns n equidistant bins between a and b

**Usage**

```r
linear_breaks(n, a, b)
```
Arguments

n  number of bins
a  lower bound
b  upper bound

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)
manual_slice

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

```r
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))
```

manual_slice

*Manually slice along a variable axis.*

Description

The manual slice tour takes the current projection, with `display_slice`, and changes the slice center.

Usage

```r
manual_slice(
  data,
  proj,
  var = 1,
  nsteps = 20,
  v_rel = 0.01,
  rescale = FALSE,
  sphere = FALSE,
  col = "black",
  half_range = NULL,
  anchor_nav = "topright",
  palette = "Zissou 1",
  ...
)
```

Arguments

- **data**: numeric matrix, with n rows and p columns
- **proj**: projection from which slices are constructed
- **var**: variable axis to run the center along: 1, ..., p
- **nsteps**: number of changes in center to make
v_rel  relative volume of the slice. If not set, suggested value is calculated and printed to the screen.
rescale Default FALSE. If TRUE, rescale all variables to range [0,1]?
sphere  if true, sphere all variables
col  color to use for points, can be a vector or hexcolors or a factor. Defaults to "black".
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.
anchor_nav  position of the anchor: center, topright or off
palette  name of color palette for point colour, used by hcl.colors, default "Zissou 1"
...  other options passed to output device

Examples

# Note that you might need to use the quartz()
# on OSX to see the animation
sphere5 <- data.frame(geozoo::sphere.hollow(5)$points)
proj <- basis_random(5, 2)
manual_slice(sphere5, proj, var=3, nsteps=10, rescale=TRUE, half_range=1.5)

mapColors

Map vector of factors to color

Description

Map vector of factors to color

Usage

mapColors(x, palette)

Arguments

x  vector

palette  name of color palette for point colour, used by hcl.colors, default "Zissou 1"
mapShapes

Map vector of factors to pch

Description
Map vector of factors to pch

Usage
mapShapes(x)

Arguments
x vector

norm_bin

Normality index.

Description
Compares the similarity between the projected distribution and a normal distribution.

- norm_bin compares the count in 100 histogram bins
- norm_kol compares the cdf based on the Kolmogorov–Smirnov test (KS test)

Usage
norm_bin(nr)
norm_kol(nr)

Arguments
nr The number of rows in the target matrix

Examples
# manually compute the norm_kol index
# create the index function
set.seed(123)
index <- norm_kol(nrow(flea[, 1:3]))
# create the projection
proj <- matrix(c(1, 0, 0), nrow = 3)
# pre-process the example data
flea_s <- sphere_data(flea[, 1:3])
# produce the index value
index(flea_s %*% proj)
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 Analysis. 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

```r
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. 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

Examples

```r
eexample(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**

```r
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**

```r
path1d <- save_history(flea[, 1:6], grand_tour(1), 3)
path2d <- save_history(flea[, 1:6], grand_tour(2), 3)

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)
  ggplot(data = df, aes(x = step, y = value, group = obs:var, colour = var)) +
    geom_line() +
    facet_wrap(~obs)

  library(tidyverse)
  ggplot(
    data = pivot_wider(df,
      id_cols = c(obs, step),
      names_from = var, names_prefix = "Var",
      values_from = value
    ),
    aes(x = Var1, y = Var2)
  ) +
    geom_point() +
    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**

```r
# This code is to be used as an example but you should increase
# the max from 2 to 50, say, to check tour coverage.
flea_std <- apply(flea[,1:6], 2, function(x) (x-mean(x))/sd(x))
grand <- interpolate(save_history(flea_std, max = 2), 0.2)

# The grand tour -----------------------------
# Look at the tour path in a tour, how well does it cover a sphere
# Using MDS to summarise the high-d space of projections
# Last basis is a duplicate, needs removing
d <- path_dist(grand[,,-dim(grand)[[3]]])
ord <- as.data.frame(MASS::isoMDS(d)$points)
require(ggplot2)
 ggplot(data = ord, aes(x=V1, y=V2)) +
   geom_path() +
   coord_equal() +
   labs(x = NULL, y = NULL)

# Compare five guided tours -----------------------------
holes1d <- guided_tour(holes(), 1)
tour_reps <- replicate(5, save_history(flea_std, holes1d, max = 2),
  simplify = FALSE)
tour_reps2 <- lapply(tour_reps, interpolate, 0.2)
bases <- unlist(lapply(tour_reps2, as.list), recursive = FALSE)
class(bases) <- "history_list"
index_values <- paths_index(tour_reps2, holes())
index_values$step <- index_values$step.1
d <- path_dist(bases)
ord <- as.data.frame(cmdscale(d, 2))
info <- cbind(ord, index_values)
 ggplot(data = info, aes(x = step, y = value, group = try)) +
   geom_line()
```
### path_index

Compute index values for a tour history.

#### Description

Compute index values for a tour history.

#### Usage

```r
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

```r
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 distributed 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, fluoridation 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, academic 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

```r
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

```r
planned_tour(basis_set, cycle = FALSE)
planned2_tour(basis_set)
```

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

---

radial_tour

*A radial tour path.*

Description

The radial tour rotates a chosen variable axis out of the current projection.

Usage

radial_tour(start, mvar = 1, ...)

---
Rat CNS

Arguments

- **start**: initial projection matrix
- **mvar**: variable(s) chosen to rotate out
- ... additional arguments for drawing

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

```r
animate_xy(flea[, 1:6], radial_tour(basis_random(6, 2), mvar = 4))
animate_xy(flea[, 1:6], radial_tour(basis_random(6, 2), mvar = c(3,4)))
animate_dist(flea[, 1:6], radial_tour(basis_random(6, 1), mvar = 4))
animate_scatmat(flea[, 1:6], radial_tour(basis_random(6, 3), mvar = 4))
```

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 embryonic timepoint from the original data with the number corresponding to the day
- e13, an embryonic timepoint from the original data with the number corresponding to the day
- e15, an embryonic timepoint from the original data with the number corresponding to the day
- e18, an embryonic timepoint from the original data with the number corresponding to the day
- e21, an embryonic timepoint from the original data with the number corresponding to the day
- p0, a postnatal timepoint from the original data with the number corresponding to the day
- p7, a postnatal timepoint from the original data with the number corresponding to the day
- p14, a postnatal timepoint from the original data with the number corresponding to the day
- a, a postnatal timepoint 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.


References

https://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 = FALSE,
  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)
render_anim

... other options passed to output device
apf angle (in radians) per frame
frames number of frames in output
rescale default FALSE. If TRUE, rescale all variables to range [0,1]
sphere if true, sphere all variables
start starting projection. If NULL, uses path default.

Examples

```r
tmp_path <- tempdir()
render(flea[, 1:6], grand_tour(), display_xy(), "pdf",
  frames = 3,
  file.path(tmp_path, "test.pdf")
)
render(flea[, 1:6], grand_tour(), display_xy(), "png",
  frames = 3,
  file.path(tmp_path, "test-%03d.png")
)
```

render_anim Render a set of animation frames

Description

This function takes a set of frames as produced by save_history(), and creates the projected data and axes in for format needed to create the animation using plotly. It will be useful for showing a tour where mouseover can be used to identify points. Note that for now this only works for 2D projections.

Usage

```r
render_anim(
  data,
  vars = NULL,
  frames,
  edges = NULL,
  axis_labels = NULL,
  obs_labels = NULL,
  limits = 1,
  position = "center"
)
```
Arguments

data matrix, or data frame containing numeric columns, should be standardised to have mean 0, sd 1
vars numeric columns of data to be projected, as a vector, eg 1:4
frames array of projection matrices, should be interpolated already
edges to and from of row id’s to connect with an line
axis_labels labels of the axes to be displayed
obs_labels labels of the observations to be available for interactive mouseover
limits value setting the lower and upper limits of projected data, default 1
position position of the axes: center (default), left of data or off

Value

list containing indexed projected data, edges, circle and segments for axes

Examples

data(flea)
flea_std <- apply(flea[,1:6], 2, function(x) (x-mean(x))/sd(x))
t1 <- save_history(flea_std, max=2)
t1i <- tourr::interpolate(t1, 0.1)
p <- render_anim(data=flea_std, frames=t1i)
if (require(ggplot2)) {
  pg <- ggplot() +
    geom_path(data=p$circle, aes(x=c1, y=c2, frame=frame)) +
    geom_segment(data=p$axes, aes(x=x1, y=y1, xend=x2, yend=y2, frame=frame)) +
    geom_text(data=p$axes, aes(x=x2, y=y2, frame=frame, label=axis_labels)) +
    geom_point(data=p$frames, aes(x=P1, y=P2, frame=frame, label=obs_labels)) +
    coord_equal() +
    theme_bw() +
    theme(axis.text=element_blank(),
          axis.title=element_blank(),
          axis.ticks=element_blank(),
          panel.grid=element_blank())
  if (interactive()) {
    require(plotly)
    ggplotly(pg, width=500, height=500) |> 
      animation_button(label="Go") |> 
      animation_slider(len=0.8, x=0.5, xanchor="center") |> 
      animation_opts(easing="linear", transition=0, redraw=FALSE)
  }
}
render_gif

Description

Render frames of animation to a gif file

Usage

```
render_gif(
  data,
  tour_path,
  display,
  gif_file = "animation.gif",
  ...
  apf = 1/10,
  frames = 50,
  rescale = FALSE,
  sphere = FALSE,
  start = NULL,
  loop = TRUE
)
```

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`
- **gif_file**: Name of gif file (default = "animation.gif")
- **...**: other options passed to `png`
- **apf**: angle (in radians) per frame
- **frames**: number of frames in output
- **rescale**: default FALSE. If TRUE, rescale all variables to range [0,1]
- **sphere**: if true, sphere all variables
- **start**: starting projection. If NULL, uses path default.
- **loop**: Logical for gifski to loop or not, default=TRUE

Examples

```r
## Not run:
# gifski needs to be installed to render a gif
if (requireNamespace("gifski", quietly = TRUE)) {
  gif_file <- file.path(tempdir(), "test.gif")
  render_gif(flea[, 1:6], grand_tour(), display_xy(), gif_file)
```

render_proj

Render plotly animation frame

Description
This function takes a projection matrix as produced by save_history(), and draws it on the projected data like a biplot. This will produce the data objects needed in order for the user to plot with base or ggplot2. Note that for now this only works for 2D projections.

Usage
render_proj(
  data,
  prj,
  axis_labels = NULL,
  obs_labels = NULL,
  limits = 1,
  position = "center"
)

Arguments
data matrix, or data frame containing numeric columns, should be standardised to have mean 0, sd 1
prj projection matrix
axis_labels of the axes to be displayed
obs_labels labels of the observations to be available for interactive mouseover
limits value setting the lower and upper limits of projected data, default 1
position position of the axes: center (default), bottomleft or off

Value
list containing projected data, circle and segments for axes

Examples
data(flea)
flea_std <- apply(flea[,1:6], 2, function(x) (x-mean(x))/sd(x))
prj <- basis_random(ncol(flea[,1:6]), 2)
p <- render_proj(flea_std, prj)
if (require("ggplot2")) {
  # End(Not run)
ggplot() +
  geom_path(data=p$circle, aes(x=c1, y=c2)) +
  geom_segment(data=p$axes, aes(x=x1, y=y1, xend=x2, yend=y2)) +
  geom_text(data=p$axes, aes(x=x2, y=y2, label=rownames(p$axes))) +
  geom_point(data=p$data_prj, aes(x=P1, y=P2)) +
  xlim(-1,1) + ylim(-1, 1) +
  theme_bw() +
  theme(aspect.ratio=1,
    axis.text=element_blank(),
    axis.title=element_blank(),
    axis.ticks=element_blank(),
    panel.grid=element_blank())

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 = FALSE,
  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**: Default FALSE. 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.
- **...**: additional arguments passed to tour path

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 <- sphere_data(testdata)
t2 <- save_history(testdata, guided_tour(holes(), max.tries = 10),
                   max = 5)
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()))

search_better

Search for a better projection near the current projection.

Description

Search for a better projection near the current projection.
Usage

```r
search_better(
  current,
  alpha = 0.5,
  index,
  tries,
  max.tries = Inf,
  ..., 
  method = "linear",
  cur_index = NA
)
```

Arguments

- `current`: starting projection
- `alpha`: the angle used to search the target basis from the current basis
- `index`: index function
- `tries`: the counter of the outer loop of the optimizer
- `max.tries`: maximum number of iteration before giving up
- `...`: other arguments being passed into the `search_better()` method
- `method`: whether the nearby bases are found by a linear/ geodesic formulation
- `cur_index`: the index value of the current basis

Examples

```r
animate_xy(flea[, 1:6], guided_tour(holes(), search_f = search_better))
```

**search_better_random**  
*Search for a better projection using simulated annealing*

Description

Given an initial $t_0$, the cooling scheme updates temperature at

$$ T = t_0 / \log(i + 1) $$

The candidate basis is sampled via

$$ B_j = (1 - \alpha) \ast B_i + \alpha \ast B $$

where alpha defines the neighbourhood, $B_i$ is the current basis, B is a randomly generated basis.

The acceptance probability is calculated as

$$ prob = \exp - abs(I(B_i) - I(B_j))/T $$

For more information, see [https://projecteuclid.org/download/pdf_1/euclid.ss/1177011077](https://projecteuclid.org/download/pdf_1/euclid.ss/1177011077)
Usage

```r
search_better_random(
  current,
  alpha = 0.5,
  index,
  tries,
  max.tries = Inf,
  method = "linear",
  cur_index = NA,
  t0 = 0.01,
  ...
)
```

Arguments

- `current`: starting projection
- `alpha`: the angle used to search the target basis from the current basis
- `index`: index function
- `tries`: the counter of the outer loop of the optimiser
- `max.tries`: maximum number of iteration before giving up
- `method`: whether the nearby bases are found by a linear/geodesic formulation
- `cur_index`: the index value of the current basis
- `t0`: initial decrease in temperature
- `...`: other arguments being passed into the `search_better_random()`

Examples

```r
animate_xy(flea[, 1:6], guided_tour(holes(), search_f = search_better_random))
```

---

**search_geodesic**

*A pseudo-derivative, line search algorithm.*

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), which is effectively a gradient search. Then it performs a linear search along the geodesic in that direction, traveling up to half way around the sphere.
Usage

```r
search_geodesic(
    current,
    alpha = 1,
    index,
    tries,
    max.tries = 5,
    ..., 
    n = 5,
    delta = 0.01,
    cur_index = NA
)
```

Arguments

- `current`: starting projection
- `alpha`: maximum distance to travel (currently ignored)
- `index`: interestingness index function
- `tries`: the counter of the outer loop of the optimizer
- `max.tries`: maximum number of failed attempts before giving up
- `...`: other arguments being passed into the `search_geodesic()`
- `n`: number of random steps to take to find best direction
- `delta`: step size for evaluation of best direction
- `cur_index`: index value for starting projection, set NA if it needs to be calculated

Details

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

Examples

```r
animate_xy(flea[, 1:6], guided_tour(holes(), search_f = search_geodesic))
```

search_polish

Search very locally to find slightly better projections to polish a broader search.

Description

Search very locally to find slightly better projections to polish a broader search.
Usage

search_polish(
  current,
  alpha = 0.5,
  index,
  tries,
  polish_max_tries = 30,
  cur_index = NA,
  n_sample = 100,
  polish_cooling = 1,
  ...
)

Arguments

current the current projection basis
alpha the angle used to search the target basis from the current basis
index index function
tries the counter of the outer loop of the optimiser
polish_max_tries maximum number of iteration before giving up
cur_index the index value of the current basis
n_sample number of samples to generate
polish_cooling percentage of reduction in polish_alpha when no better basis is found
... other arguments being passed into the search_polish()

Examples

data(t1)
best_proj <- t1[, , dim(t1)[3]]
attr(best_proj, "data") <- NULL
best_proj <- unclass(drop(best_proj))
animate_xy(
  flea[, 1:6],
guided_tour(holes()),
  search_f = search_polish(
    polish_max_tries = 5),
  start = best_proj
)
search_posse

Search for a better projection based on Poss, 1995

Description

Search for a better projection based on Poss, 1995

Usage

```r
search_posse(
  current,
  alpha = 0.5,
  index,
  tries,
  max.tries = 300,
  cur_index = NA,
  ...
)
```

Arguments

current      starting projection
alpha        the angle used to search the target basis from the current basis
index        index function
tries        the counter of the outer loop of the opotimiser
max.tries    maximum number of iteration before giving up
cur_index    the index value of the current basis
...          other arguments being passed into the search_better()

slice_index

Section pursuit index.

Description

Calculates a section pursuit index that compares the distribution inside and outside a slice.
**Usage**

```r
slice_index(
  breaks_x,
  breaks_y,
  eps,
  bintype = "polar",
  power = 1,
  flip = 1,
  reweight = FALSE,
  p = 4
)
```

**Arguments**

- `breaks_x`: binning on the first variable (x or radius).
- `breaks_y`: binning on the second variable (y or angle).
- `eps`: cutoff values to suppress summing up small differences. Vector with one entry for each bin, can be estimated using `estimate_eps`.
- `bintype`: select polar (default) or square binning.
- `power`: exponent q used in the index computation.
- `flip`: sign of the index computation, select +1 when searching for low densities and -1 when searching for high densities.
- `reweight`: if TRUE will reweight according to the expected distribution in a uniform hypersphere (default is FALSE).
- `p`: number of variables in the data (needed for accurate reweighting, default is 4).

---

**sphere_data**

*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**

```r
sphere_data(df)
```

**Arguments**

- `df`: data frame or matrix
**splines2d**

*Spline based index.*

---

**Description**

Compares the variance in residuals of a fitted spline model to the overall variance to find functional dependence in 2D projections of the data.

**Usage**

```r
splines2d()
```

---

**t1**

*Saved history of guided tour with holes*

---

**Description**

This data was generated from the following code: `set.seed(2020) t1 <- save_history(flea[, 1:6], guided_tour(holes()), max = 100) attr(t1, "class") <- NULL` And used as an example for `search_polish()` to start optimising from the best projection from `search_geodesic`. t1 is a 3D array or 2D projections.

---

**Tropical Atmosphere Ocean**

*Tropical Atmosphere Ocean data*

---

**Description**

This is a subset of data taken from the NOAA web site https://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**

https://www.pmel.noaa.gov/tao/
Index

* algebra
  proj_dist, 48
* datasets
  Flea measurements, 29
  Laser measurements, 36
  Olive oil measurements, 42
  Ozone measurements, 42
  Places Ratings, 46
  Rat CNS, 49
  t1, 63
  Tropical Atmosphere Ocean, 63
* dynamic
  planned_tour, 47
* hplot
  cmass, 7
  dcor2d, 7
  display_andrews, 8
  display_depth, 10
  display_dist, 11
  display_faces, 13
  display_image, 17
  display_pcp, 19
  display_scatmat, 21
  display_stars, 23
  display_stereo, 24
  holes, 35
  interpolate, 36
  lda_pp, 37
  norm_bin, 41
  path_index, 45
  pda_pp, 46
  planned_tour, 47
  render, 50
  splines2d, 63
* manip
  rescale, 55
  sphere_data, 62
* optimize
  search_better, 56
  search_better_random, 57
  search_geodesic, 58
  search_polish, 59
  search_posse, 61
  anchored_orthogonal_distance, 3
  andrews, 4, 8
  angular_breaks, 4
  animate, 5, 8, 9, 11–13, 15, 17–21, 23–27, 31, 32, 35, 38, 39, 48, 49
  animate_andrews (display_andrews), 8
  animate_density2d (display_density2d), 9
  animate_depth (display_depth), 10
  animate_dist (display_dist), 11
  animate_faces (display_faces), 13
  animate_groupxy (display_groupxy), 14
  animate_idx (display_idx), 15
  animate_image (display_image), 17
  animate_pca (display_pca), 18
  animate_pcp (display_pcp), 19
  animate_sage (display_sage), 20
  animate_scatmat (display_scatmat), 21
  animate_slice, 33
  animate_slice (display_slice), 22
  animate_stars (display_stars), 23
  animate_stereo (display_stereo), 24
  animate_trails (display_trails), 25
  animate_xy (display_xy), 26
  areColors, 6
  center, 6
  cmass, 7, 30, 35
  dcor2d, 7
  dependence_tour, 7
  display_andrews, 8
  display_density2d, 9, 9
  display_depth, 10
  display_dist, 11
  display_faces, 13
INDEX

display_groupxy, 14, 15
display_idx, 15
display_image, 17
display_pca, 18
display_pcp, 19, 50, 53
display_sage, 20, 20
display_scmat, 21
display_slice, 18, 22, 23
display_stars, 23
display_scytata, 24
display_trails, 25
display_xy, 26, 26, 27, 50, 53
draw_tour_axes, 28
estimate_eps, 28, 62
faces2, 13
flea (Flea measurements), 29
Flea measurements, 29
freeze, 30, 31
frozen_guided_tour, 30, 30
frozen_tour, 30
grand_tour, 31
guided_section_tour, 32
guided_tour, 34, 39
hcl.colors, 8, 9, 12, 15, 16, 18, 20, 23, 27, 40
holes, 30, 35, 35
interpolate, 36
laser (Laser measurements), 36
Laser measurements, 36
lda_pp, 30, 35, 37
linear_breaks, 37
little_tour, 38, 48
local_tour, 38
manual_slice, 39
mapColors, 40
mapShapes, 41
norm_bin, 41
norm_kol (norm_bin), 41
olive (Olive oil measurements), 42
Olive oil measurements, 42
ozone (Ozone measurements), 42
Ozone measurements, 42
path_curves, 43
path_dist, 44
path_index, 45
pda_pp, 46
pdf, 50
places (Places Ratings), 46
Places Ratings, 46
planned2_tour (planned_tour), 47
planned_tour, 36, 47
png, 50, 53
proj_dist, 48
radial_tour, 48
Rat CNS, 49
ratcns (Rat CNS), 49
render, 5, 8, 31–33, 35, 38, 39, 48, 49, 50
render_anima, 51
render_gif, 53
render_proj, 54
rescale, 55
save_history, 8, 31–33, 35, 38, 39, 43, 45, 48, 49, 55
search_better, 33–35, 56
search_better_random, 33–35, 57
search_geodesic, 33–35, 58
search_polish, 34, 59
search_posse, 61
slice_index, 33, 61
sphere_data, 62
splines2d, 63
stars, 24
t1, 63
tao (Tropical Atmosphere Ocean), 63
Tropical Atmosphere Ocean, 63