Package ‘spinifex’

September 28, 2021

Title Manual Tours, Manual Control of Dynamic Projections of Numeric Multivariate Data

Version 0.3.1

Description Data visualization tours animates linear projection of multivariate data as its basis (ie. orientation) changes. The 'spinifex' packages generates paths for manual tours by manipulating the contribution of a single variable at a time Cook & Buja (1997) <doi:10.1080/10618600.1997.10474754>. Other types of tours, such as grand (random walk) and guided (optimizing some objective function) are available in the 'tourn' package Wickham et al. <doi:10.18637/jss.v040.i02>, 'spinifex' builds on 'tourn' and can render tours with 'gganimate' and 'plotly' graphics, and allows for exporting as an .html widget and as an .gif, respectively. This work is fully discussed in Spyrison & Cook (2020) <doi:10.32614/RJ-2020-027>.

Depends R (>= 3.5.0), tourr

License CC BY-NC-SA 4.0

URL https://github.com/nspyrison/spinifex/

BugReports https://github.com/nspyrison/spinifex/issues

Imports ggplot2, gganimate, plotly, shiny, Rdimtools, transformr, magrittr

Suggests MASS, hexbin, htmlwidgets, gifski, png, dplyr, GGally, rmarkdown, knitr, testthat, lifecycle, covr, spelling

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Language en-US

NeedsCompilation no

Author Nicholas Spyrison [aut, cre], Dianne Cook [aut, ths]
Maintainer Nicholas Spyrison <spyrison@gmail.com>
Repository CRAN
Date/Publication 2021-09-28 12:30:02 UTC

**R topics documented:**

- .bind_elements2df .................................................. 3
- .init4proto .......................................................... 3
- .lapply_rep_len ...................................................... 4
- animate_gganimate .................................................. 5
- animate_plotly ....................................................... 6
- array2df .............................................................. 7
- as_history_array ..................................................... 8
- basis_guided .......................................................... 9
- basis_half_circle .................................................. 10
- basis_odp ............................................................. 10
- basis_oda ............................................................. 11
- basis_onpp .......................................................... 12
- basis_pca ............................................................ 13
- BreastCancer ............................................................ 14
- create_manip_space .................................................. 15
- filmstrip ............................................................. 16
- ggtour ............................................................... 17
- interpolate_manual_tour ........................................... 18
- is_orthonormal ........................................................ 19
- manip_var_of ........................................................ 20
- manual_tour .......................................................... 20
- map_absolute .......................................................... 22
- map_relative .......................................................... 23
- penguins ............................................................... 24
- PimaIndiansDiabetes_long ........................................... 25
- PimaIndiansDiabetes_wide .......................................... 27
- play_manual_tour ...................................................... 28
- play_tour_path ........................................................ 30
- proto_basis ............................................................. 31
- proto_default .......................................................... 33
- proto_density .......................................................... 34
- proto_hex .............................................................. 35
- proto_highlight ....................................................... 36
- proto_origin ............................................................ 37
- proto_point ............................................................ 39
- proto_text ............................................................. 40
- render ................................................................. 41
- render_gganimate ..................................................... 42
- render_plotly .......................................................... 44
- rotate_manip_space .................................................. 45
- run_app ................................................................. 46
.bind_elements2df

Binds replicated elements of a list as columns of a data frame.

Description

Internal function. To be applied to aes_args replicates elements to length data

Usage

.bind_elements2df(list, df)

Arguments

list A list of arguments such as those passed in aes_args and identity_args.
df A data.frame to column bind the elements of list to.

See Also

Other Internal utility: .init4proto, .lapply_rep_len(), interpolate_manual_tour()

Examples

## This function is not meant for external use

.init4proto

Initialize common obj from .global ggtour() objects & test their existence

Description

Internal expression. Creates local .objects to be commonly consumed by spinifex proto_* functions.

Usage

.init4proto
Description

Internal function. To be applied to `aes_args` and `identity_args`, replicates vectors of length data to length of data*frames for animation.

Usage

```
lapply_rep_len(list, to_length, expected_length)
```

Arguments

- `list` A list of arguments such as those passed in `aes_args` and `identity_args`.
- `to_length` Scalar number, length of the output vector; the number of rows in the data frames to replicate to.
- `expected_length` Scalar number, the expected length of the each element of `list`.

See Also

Other Internal utility: `.bind_elements2df()`, `.lapply_rep_len()`, `interpolate_manual_tour()`

Examples

```r
## This function is not meant for external use
```
animate_gganimate

Animate a ggtour as a .gif via \{gganimate\}

Description

Animates the ggplot return of ggtour() and added proto_*() functions as a .gif without interaction, through use of \{gganimate\}.

Usage

animate_gganimate(
  ggtour,
  fps = 8,
  rewind = FALSE,
  start_pause = 1,
  end_pause = 1,
  ...
)

Arguments

  ggtour A grammar of graphics tour with appended protos added. A return from ggtour() + proto_*().
  fps Scalar number of Frames Per Second, the speed the animation should play at.
  rewind Whether or not the animation should play backwards, in reverse order once reaching the end. Defaults to FALSE.
  start_pause The duration in seconds to wait before starting the animation. Defaults to 1 second.
  end_pause The duration in seconds to wait after ending the animation, before it restarts from the first frame. Defaults to 1 second.
  ... Other arguments passed to gganimate::animate.

See Also

  gganimate::animate
  Other ggtour animator: animate_plotly()

Examples

  dat <- scale_sd(tourr::flea[, 1:6])
  clas <- tourr::flea$species
  bas <- basis_pca(dat)
  mv <- manip_var_of(bas)
  mt_path <- manual_tour(bas, manip_var = mv)
  ggt <- ggtour(mt_path, dat, angle = .1) +
proto_basis() +
proto_origin() +
proto_point(aes_args = list(color = clas, shape = clas),
    identity_args = list(size = 1.5, alpha = .7))

## Not run:
animate_gganimate(ggt)

## Example saving gganime to a .gif, may require additional setup.
if(F){
  anim <- animate_gganimate(ggt, fps = 10, rewind = TRUE,
    start_pause = 1, end_pause = 2)
  gganimate::anim_save("my_tour.gif",
    animation = anim,
    path = "./figures")
}
## End(Not run)

animate_plotly  

**Animate a ggtour as and HTML widget via \{plotly\}**

**Description**

Animates the static ggtour() and added proto_*() functions as a \{plotly\} animation, an .html widget with slider and hover tooltip showing the row number.

**Usage**

```r
animate_plotly(ggtour, fps = 8, ...)
```

**Arguments**

- `ggtour`  
  A grammar of graphics tour with appended protos added. A return from gg-
tour() + proto_*().

- `fps`  
  Scalar number of Frames Per Second, the speed the animation should play at.

- `...`  
  Other arguments passed to `plotly::layout`.

**See Also**

- `plotly::ggplotly`
- Other ggtour animator: `animate_gganimate()`

**Examples**

```r
dat <- scale_sd(tourr::flea[, 1:6])
clas <- tourr::flea$species
bas <- basis_pca(dat)
mv <- manip_var_of(bas)
mt_path <- manual_tour(bas, manip_var = mv)
```
array2df <- ggtour(mt_path, dat, angle = .1) +
  proto_origin() +
  proto_basis() +
  proto_point(aes_args = list(color = clas, shape = clas),
              identity_args = list(size = 1.5, alpha = .7))
## Not run:
animate_plotly(ggtour)

## Example saving plotly to a .html widget, may require additional setup.
if(F){
  anim <- animate_plotly(ggtour, fps = 10)
  htmlwidgets::saveWidget(widget = anim, file = "/figures/my_tour.html",
                          selfcontained = TRUE)}
## End(Not run)

array2df

Turns a tour path array into a long data frame.

Description

Internal function, many end users will not need this. Takes the result of manual_tour() or tourr::save_history(). Restructures the array of interpolated bases into a long data frame for use in ggplots.

Usage

array2df(basis_array, data = NULL, basis_label = if (is.null(data) == FALSE) abbreviate(colnames(data), 3) else 1:nrow(basis_array), data_label = if (is.null(data) == FALSE) abbreviate(colnames(data), 3) else paste0("v", 1:ncol(basis_array)) )

Arguments

basis_array A full (p, d, n_frames) interpolated basis array of a tour, the output of manual_tour or save_history(*_tour()).
data Optional, (n, p) dataset to project, consisting of numeric variables.
basis_label Optional, labels for the reference frame, a character vector of the number of variables. Defaults to the 3 character abbreviation of the original variables names.
data_label Optional, labels for plotly tooltip and return object. Defaults to the rownames of the data, if available, then the row number.
Examples

```r
## This function is not meant for external use!!
dat_std <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)

## Radial tour array to long df, as used in play_manual_tour()
mt_array <- manual_tour(basis = bas, manip_var = mv)
ls_df_frames <- array2df(basis_array = mt_array, data = dat_std,
    basis_label = paste0("MyLabs", 1:nrow(bas)))
str(ls_df_frames)

## tourr::save_history tour array to long df, as used in play_tour_path()
gt_array <- tourr::save_history(data = dat_std, max_bases = 10)
ls_df_frames2 <- array2df(basis_array = gt_array, data = dat_std)
str(ls_df_frames2)
```

---

### as_history_array

Changes an array of bases into a "history_array" class for use in `tourr::interpolate()`.

Description

Internal function, many end users will not need this. Attaches data to an array and assigns the custom class "history_array" as used in `tourr`. Typically called by basis arrays from `spinifex` functions.

Usage

```r
as_history_array(basis_array, data = NULL)
```

Arguments

- `basis_array` An array of bases.
- `data` The data matrix to be projected through the basis. This is `tourr::save_history` objects, but not consumed downstream in `spinifex`.

Value

An array of numeric bases with custom class "history_array" for consumption by `tourr::interpolate`.

See Also

- `tourr::save_history` for preset choices.
basis_guided

Examples

```r
## !!This function is not meant for external use!!
dat_std <- scale_sd(wine[, 2:6])
basis_guided(data = dat_std, index_f = tourr::holes())
basis_guided(data = dat_std, index_f = tourr::cmass(), alpha = .4, cooling = .9, max.tries = 10, n_sample = 4)
```

Description

Performs simulated annealing on the index function, solving for it’s local extrema. Returns only the last identified basis of the optimization. A truncated, muted extension of tourr::save_history(guided_tour()).

Usage

```r
basis_guided(data, index_f = tourr::holes(), d = 2, ...)
```

Arguments

- **data**: Numeric matrix or data.frame of the observations.
- **index_f**: The index function to optimize. tourr exports holes(), cmass(), and lda_pp(class).
- **d**: Number of dimensions in the projection space.
- **...**: Optional, other arguments to pass to tourr::guided_tour

Value

Numeric matrix of the last basis of a guided tour.

See Also

tourr::guided_tour for annealing arguments.

Other basis identifiers: basis_half_circle(), basis_odp(), basis_olda(), basis_onpp(), basis_pca()

Examples

```r
dat_std <- scale_sd(wine[, 2:6])
basis_guided(data = dat_std, index_f = tourr::holes())

basis_guided(data = dat_std, index_f = tourr::cmass(),
alpha = .4, cooling = .9, max.tries = 10, n_sample = 4)
```
basis_half_circle

Create a basis that gives uniform contribution in a circle

Description

Orthonormalizes uniform variable contributions on a unit circle. This serves as a NULL basis, one that is variable agnostic while spacing the variables to have minimize variable dependence.

Usage

basis_half_circle(data)

Arguments

data The data to create a basis for.

See Also

Other basis identifiers: basis_guided(), basis_odp(), basis_olda(), basis_onpp(), basis_pca()

Examples

dat_std <- scale_sd(wine[, 2:6])
bas <- basis_half_circle(dat_std)

basis_odp

The basis of Orthogonal Discriminant Projection (ODP)

Description

Orthogonal Discriminant Projection (ODP) is a linear dimension reduction method with class supervision. It maximizes weighted difference between local and non-local scatter while local information is also preserved by constructing a neighborhood graph.

Usage

basis_odp(data, class, d = 2, type = c("proportion", 0.1), ...)

Arguments

data Numeric matrix or data.frame of the observations, coerced to matrix.
class The class for each observation, coerced to a factor.
d Number of dimensions in the projection space. of class.
type A vector specifying the neighborhood graph construction. Expects: c("knn",k), c("enn",radius), or c("proportion",ratio). Defaults to c("knn", sqrt(nrow(data))), nearest neighbors equal to the square root of observations.
... Optional, other arguments to pass to Rdimtools::do.odp.
basis_olda

References

See Also
Rdimtools::do.odp for locality preservation arguments.
Rdimtools::aux.graphnbd for details on type.
Other basis identifiers: basis_guided(), basis_half_circle(), basis_olda(), basis_onpp(), basis_pca()

Examples

dat_std <- scale_sd(wine[, 2:6])
clas <- wine$Type
basis_odp(data = dat_std, class = clas)

basis_olda The basis of Orthogonal Linear Discriminant Analysis (OLDA)

Description
Orthogonal LDA (OLDA) is an extension of classical LDA where the discriminant vectors are orthogonal to each other.

Usage
basis_olda(data, class, d = 2)

Arguments
data Numeric matrix or data.frame of the observations, coerced to matrix.
class The class for each observation, coerced to a factor.
d Number of dimensions in the projection space.

Value
A numeric matrix, an orthogonal basis that best distinguishes the group means of class.

References
See Also

Rdimtools::do.olda

Other basis identifiers: basis_guided(), basis_half_circle(), basis_odp(), basis_onpp(), basis_pca()

Examples

dat_std <- scale_sd(wine[, 2:6])
clas <- wine$Type
basis_olda(data = dat_std, class = clas)

basis_onpp

The basis of Orthogonal Neighborhood Preserving Projection (OLPP)

Description

Orthogonal Neighborhood Preserving Projection (ONPP) is an unsupervised linear dimension reduction method. It constructs a weighted data graph from LLE method. Also, it develops LPP method by preserving the structure of local neighborhoods. For the more details on type see Rdimtools::aux.graphnbd().

Usage

basis_onpp(data, d = 2, type = c("knn", sqrt(nrow(data))))

Arguments

data Numeric matrix or data.frame of the observations, coerced to matrix.
d Number of dimensions in the projection space.
type A vector specifying the neighborhood graph construction. Expects; c("knn", k), c("enn", radius), or c("proportion", ratio). Defaults to c("knn", sqrt(nrow(data))), nearest neighbors equal to the square root of observations.

Value

Orthogonal matrix basis that distinguishes the levels of class based on local and non-local variation as weighted against the neighborhood graph.

References

See Also

Rdimtools::do.onpp
Rdimtools::aux.graphnbd for details on type.

Other basis identifiers: basis_guided(), basis_half_circle(), basis_odp(), basis_olda(), basis_pca()

Examples

dat_std <- scale_sd(wine[, 2:6])
basis_onpp(data = dat_std)

declare

basis_pca(data, d = 2)

Arguments

data Numeric matrix or data.frame of the observations.
d Number of dimensions in the projection space.

See Also

Rdimtools::do.pca

Other basis identifiers: basis_guided(), basis_half_circle(), basis_odp(), basis_olda(), basis_onpp()

Examples

dat_std <- scale_sd(wine[, 2:6])
basis_pca(data = dat_std)
Description
The objective is to identify each of a number of benign or malignant classes. Samples arrive periodically as Dr. Wolberg reports his clinical cases. The database therefore reflects this chronological grouping of the data. This grouping information appears immediately below, having been removed from the data itself. Each variable except for the first was converted into 11 primitive numerical attributes with values ranging from 0 through 10. Rows with missing attribute values and duplicate rows removed.

Usage
BreastCancer

Format
A data frame with 675 observations of 8 numeric variables and target factor Class.

- Id, Sample code number
- Cl.thickness, Clump thickness
- Cell.size, Uniformity of cell size
- Cell.shape, Uniformity of cell shape
- Marg.adhesion, Marginal adhesion
- Epith.c.size, Single Epithelial cell size
- Bare.nuclei, Bare nuclei
- Bl.cromatin, Bland chromatin
- Normal.nucleoli, Normal Nucleoli
- Mitoses, Mitoses
- Class, Class of cancer, either "benign" or "malignant"

Details
This is a cleaned subset of mlbench's BreastCancer. See help(BreastCancer, package = "mlbench") for the original.

Replicating this dataset:

require("mlbench")
data(BreastCancer)

raw <- BreastCancer
## rownumber index of 8 duplicate 16 incomplete rows
idx <- !duplicated(raw) & complete.cases(raw)
create_manip_space

```
d <- raw[idx, 3:10]
d <- apply(d, 2L, as.integer)
d <- data.frame(d, Class = as.factor(raw$Class[idx]))
BreastCancer <- d
## save(BreastCancer, file = "/data/BreastCancer.rda")
```

Source


Examples

```
library("spinifex")
str(spinifex::BreastCancer)
dat <- scale_sd(spinifex::BreastCancer[, 1:8])
clas <- spinifex::BreastCancer$Class

bas <- basis_pca(dat)
mv <- manip_var_of(bas)
mt <- manual_tour(bas, mv)

ggt <- ggtour(mt, dat, angle = .2) +
  proto_default(list(color = clas, shape = clas))
## Not run:
animate_plotly(ggt)
## End(Not run)
```

create_manip_space

Create a manipulation space to rotate the manipulation variable in.

Description

Typically called by manual_tour(). Creates a (p, d) orthonormal matrix, the manipulation space from the given basis right concatenated with a zero vector, with manip_var set to 1.

Usage

```
create_manip_space(basis, manip_var = manip_var_of(basis))
```

Arguments

- **basis**: A (p, d) orthonormal numeric matrix, the linear combination the original variables contribute to projection frame. Required, no default.
- **manip_var**: The number of the variable/column to rotate. Defaults to `manip_var_of(basis)`, the variable with the largest contribution in the basis.
Value

A $(p, d + 1)$ orthonormal matrix, the manipulation space to manipulate the projection in.

Examples

```r
## Setup
dat_std <- scale_sd(wine[, 2:6])
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)
create_manip_space(basis = bas, manip_var = mv)

## d = 1 case
bas1d <- basis_pca(dat_std, d = 1)
mv <- manip_var_of(bas1d)
create_manip_space(bas1d, mv)
```

Description

Appends `facet_wrap(vars(frame_number))` & minor themes to the ggtour. If the number of frames is more than desired, try increasing the `angle` argument on the tour.

Usage

```r
filmstrip(ggtour)
```

Arguments

- **ggtour**: A grammar of graphics tour with appended protos added. A return from `ggtour() + proto_*()`.

See Also

Other ggtour proto: `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_hex()`, `proto_highlight()`, `proto_origin()`, `proto_point()`, `proto_text()`

Examples

```r
dat <- scale_sd(tourr::flea[, 1:6])
clas <- tourr::flea$species
bas <- basis_pca(dat)
mv <- manip_var_of(bas)

## d = 2 case
mt_path <- manual_tour(bas, manip_var = mv)
ggt <- ggtour(mt_path, dat, angle = .25) +
    proto_basis() +
```

filmstrip

Create a "filmstrip" of the frames of a ggtour.
proto_point(list(color = clas, shape = clas),
    list(size = 1.5))
filmstrip(ggt)

## d = 1 case
bas1d <- basis_pca(dat, d = 1)
mt_path1d <- manual_tour(basis = bas1d, manip_var = mv)
ggt1d <- ggtour(mt_path1d, dat, angle = .3) +
    proto_default1d(list(fill = clas))
ggt1d <- ggtour(mt_path1d, dat) +
    proto_default1d(list(fill = clas))

## Not run:
filmstrip(ggt1d)
## End(Not run)

ggtour

Prepare a new grammar of graphics tour

Description

ggtour() initializes a ggplot object for a tour. proto_* functions are added to the tour, analogous to ggplot() + geom_* . The final tour object is then animated with animate_plotly() or animate_ggtour(), or passed to filmstrip() for static plot faceting on frames.

Usage

ggtour(basis_array, data = NULL, angle = 0.05, facet_by = NULL)

Arguments

basis_array An array of projection bases for the tour, as produced with manual_tour() or tour::save_history(), or a single basis.
data Numeric data to project. If left NULL, will check if it data is stored as an attribute of the the basis_array.
angle Target angle (in radians) for interpolation for tour::save_history() generated basis_array. Defaults to .05.
facet_by Optionally, add a vector to facet the ggtour on. Similar to adding facet_grid(rows = facet_by) to the tour.

See Also

Other ggtour proto: filmstrip(), proto_basis(), proto_default(), proto_density(), proto_hex(), proto_highlight(), proto_origin(), proto_point(), proto_text()
Examples

dat <- scale_sd(tourr::flea[, 1:6])
clas <- tourr::flea$species
bas <- basis_pca(dat)
mv <- manip_var_of(bas)
mt_path <- manual_tour(bas, manip_var = mv)

## Returns headless ggplot(), but required for proto_* functions.
ggtour(mt_path, dat, angle = .15)

## d = 2 case
ggt <- ggtour(mt_path, dat) +
    proto_basis() +
    proto_point(list(color = clas, shape = clas),
                list(size = 1.5))
## Not run:
animate_plotly(ggt)
## End(Not run)

## d = 1 case
bas1d <- basis_pca(dat, d = 1)
mt_path1d <- manual_tour(basis = bas1d, manip_var = mv)
ggt1d <- ggtour(mt_path1d, dat, angle = .2) +
    proto_default1d(list(fill = clas))
## Not run:
animate_plotly(ggt1d)
## End(Not run)

## d = 2, with facet
ggt <- ggtour(mt_path, dat, facet_by = clas) +
    proto_default(list(color = clas, shape = clas), list(size = 1.5))
## Not run:
animate_plotly(ggt)
## End(Not run)

## d = 1, with facet
ggt1d <- ggtour(mt_path1d, dat, facet_by = clas) +
    proto_default1d(list(color = clas, shape = clas, fill = clas))
## faceted 1d doesn’t work the best with plotly; esp rug, and basis segments.
## Not run:
animate_gganimate(ggt1d)
## End(Not run)

---

interpolate_manual_tour

Interpolates a manual tour

Description

Internal function. Interpolates a manual tour over the stored theta, and phi specifications. Returns an interpolated basis_array to be consumed by array2df.
is_orthonormal

Usage

interpolate_manual_tour(basis_array, angle = 0.05)

Arguments

basis_array array, of the target bases, the extrema of the walk/segments.
angle The step size between interpolated frames, in radians.

See Also

Other Internal utility: .bind_elements2df(), .init4proto, .lapply_rep_len()

Examples

## This function is not meant for external use
dat_std <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)
mt <- manual_tour(bas, mv)

interp <- spinifex:::interpolate_manual_tour(basis_array = mt, angle = .1)
dim(interp)
str(interp)

is_orthonormal

Test if a numeric matrix is orthonormal, that is, each column is orthogonal, at a right angle with the others, and each column has a norm length of 1. This must be true for a projection to be linear.

Description

Test if a numeric matrix is orthonormal, that is, each column is orthogonal, at a right angle with the others, and each column has a norm length of 1. This must be true for a projection to be linear.

Usage

is_orthonormal(x, tol = 0.001)

Arguments

x Numeric matrix to test the orthonormality of.
tol Max tolerance of floating point differences. Element-wise distance of t(x) %**% x from the identity matrix.

Value

Single logical, whether or not the matrix is orthonormal.
Examples

```
is_orthonormal(tourr::basis_random(n = 6))
is_orthonormal(matrix(1:12, ncol = 2), tol = 0.01)
```

---

**manip_var_of**  
Suggest a manipulation variable.

---

**Description**

Find the column number of the variable with the rank-ith largest contribution of the basis. Useful for identifying a variable to change the contribution of in a manual tour, it’s `manip_var` argument.

**Usage**

```
manip_var_of(basis, rank = 1)
```

**Arguments**

- **basis**  
  Numeric matrix (p x d), orthogonal liner combinations of the variables.

- **rank**  
  The number, specifying the variable with the rank-th largest contribution. Defaults to 1.

**Value**

Numeric scalar, the column number of a variable.

**Examples**

```
## Setup
dat_std <- scale_sd(wine[, 2:6])
bas <- basis_pca(dat_std)

manip_var_of(basis = bas) ## Variable with the largest contribution
manip_var_of(basis = bas, rank = 5) ## Variable with 5th-largest contribution
```

---

**manual_tour**  
Produce the series of projection bases to rotate a variable into and out of a projection.

---

**Description**

Typically called by `array2af()`. An array of projections, the radial tour of the `manip_var`, which is rotated from phi’s starting position to phi_max, to phi_min, and back to the start position.
Usage

```r
manual_tour(
  basis,
  manip_var,
  theta = NULL,
  phi_min = 0L,
  phi_max = pi/2,
  data = NULL
)
```

Arguments

- **basis**: A (p, d) orthonormal numeric matrix. The linear combination the original variables contribute to projection space. Defaults to NULL, generating a random basis.
- **manip_var**: Integer column number or string exact column name of the variable to manipulate. Required, no default.
- **theta**: Angle in radians of "in-plane" rotation, on the xy plane of the reference frame. Defaults to theta of the basis for a radial tour.
- **phi_min**: Minimum value phi should move to. Phi is angle in radians of the "out-of-plane" rotation, the z-axis of the reference frame. Required, defaults to 0.
- **phi_max**: Maximum value phi should move to. Phi is angle in radians of the "out-of-plane" rotation, the z-axis of the reference frame. Required, defaults to pi/2.
- **data**: Optionally attach data to the basis path.

Value

A (p, d, 4) history_array of the radial tour. The bases set for phi_start, phi_min, phi_max, and back to phi_start.

Examples

```r
## Setup
dat_std <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)
manual_tour(basis = bas, manip_var = mv)

## All arguments
manual_tour(basis = bas, manip_var = mv,
            theta = pi / 2, phi_min = pi / 16, phi_max = pi)

## d = 1 case
bas1d <- basis_pca(dat_std, d = 1)
mv <- manip_var_of(bas1d)
manual_tour(basis = bas1d, manip_var = mv)
```
# Animating with ggtour() & proto_*

```r
mt <- manual_tour(basis = bas, manip_var = mv)
ggt <- ggtour(mt, dat_std, angle = .2) +
    proto_origin() +
    proto_point(list(color = clas, shape = clas)) +
    proto_basis()
```

## Not run:

```r
animate_plotly(ggt)
```

## End(Not run)

---

**map_absolute**

Manually offset and scale the first 2 columns of a matrix or data.frame.

## Description

A manual variant of `map_relative()`. Can be used as the axes argument to manually set the size and locations of the axes.

## Usage

```r
map_absolute(x, offset = c(0L, 0L), scale = c(1L, 1L))
```

## Arguments

- **x**: Numeric data object with 2 columns to scale and offset. Defaults to NULL, passing arguments to `scale_axes` for use internally.
- **offset**: 2 Numeric values to offset/pan the first 2 dimensions of x.
- **scale**: 2 Numeric values to scale/zoom to the first 2 dimensions of x.

## Value

Scaled and offset x.

## See Also

- `scale_axes` for preset choices.

Other Linear mapping: `map_relative()`

## Examples

```r
bas <- tourr::basis_random(4, 2)
map_absolute(bas, offset = c(-2, 0), scale = c(2/3, 2/3))
```
map_relative

Returns the axis scale and position.

Description

Internal function. Typically called by other functions to scale the position of the axes data.frame or another data.frame to plot relative to the data.

Usage

map_relative(
  x,
  position = c("center", "left", "top1d", "floor1d", "right", "bottomleft", "topright", "off"),
  to = NULL
)

Arguments

- **x**: Numeric matrix or data.frame, first 2 columns and scaled and offset the to object.
- **position**: Text specifying the position the axes should go to. Defaults to "center" expects one of: "center", "left", "right", "bottomleft", "topright", or "off".
- **to**: Table to appropriately set the size and position of the axes to. Based on the min/max of the first 2 columns. If left NULL defaults to data.frame(x = c(-1L, 1L), y = c(-1L, 1L)).

Value

Transformed values of x, dimension and class unchanged.

See Also

- map_absolute for more manual control.

Other Linear mapping: map_absolute()

Examples

```r
## !!This function is not meant for external use!!
rb <- tourr::basis_random(4, 2)

map_relative(x = rb, position = "bottomleft")
map_relative(x = rb, position = "right", to = wine[, 2:3])
```
penguins

Size measurements for adult foraging penguins near Palmer Station, Antarctica

Description

Includes measurements for penguin species, island in Palmer Archipelago, size (flipper length, body mass, bill dimensions), and sex.

Usage

penguins

Format

A data frame with 333 rows and 4 numeric variables and 3 factor variables

- `bill_length_mm` a number denoting bill length (millimeters)
- `bill_depth_mm` a number denoting bill depth (millimeters)
- `flipper_length_mm` an integer denoting flipper length (millimeters)
- `body_mass_g` an integer denoting body mass (grams)
- `species` a factor denoting penguin species (Adelie, Chinstrap and Gentoo)
- `sex` a factor denoting penguin sex (female, male)
- `island` a factor denoting island in Palmer Archipelago, Antarctica (Biscoe, Dream or Torgersen)

Details

This is a cleaned subset of `palmerpenguins::penguins`.

Replicating this dataset:

```r
require("palmerpenguins")
d <- palmerpenguins::penguins
d <- d[!is.na(d$sex), ] ## Remove missing
d <- d[, c(3:6, 1, 7, 2)] ## Numeric to front, group factors, remove year
penguins <- as.data.frame(d) ## Remove {tibble} dependency
## save(penguins, file = "/data/penguins.rda")
```

Source

Pima Indians Diabetes Dataset, long

Description

The data set PimaIndiansDiabetes2 contains a corrected version of the original data set. While the UCI repository index claims that there are no missing values, closer inspection of the data shows several physical impossibilities, e.g., blood pressure or body mass index of 0. In PimaIndiansDiabetes2, all zero values of glucose, pressure, triceps, insulin and mass have been set to NA, see also Wahba et al (1995) and Ripley (1996).

Usage

PimaIndiansDiabetes_long
Format
A data frame with 724 observations of 6 numeric variables, and target factor diabetes.

- pregnant, Number of times pregnant
- glucose, Plasma glucose concentration (glucose tolerance test)
- pressure, Diastolic blood pressure (mm Hg)
- mass, Body mass index (weight in kg/(height in m)^2)
- pedigree, Diabetes pedigree function
- age, Age (years)
- diabetes, Class variable (test for diabetes), either "pos" or "neg"

Details
This is a cleaned subset of mlbench’s PimaIndiansDiabetes2. See help(PimaIndiansDiabetes2, package = "mlbench").

Replicating this dataset:

```r
require("mlbench")
data(PimaIndiansDiabetes2)

d <- PimaIndiansDiabetes2
d <- d[, c(1:3, 6:9)]  ## Remove 2 columns with the most NAs
# Remove ~44 row-wise incomplete rows
PimaIndiansDiabetes_long <- d
## save(PimaIndiansDiabetes_long, file = "../data/PimaIndiansDiabetes_long.rda")
```

Source


Examples
```r
library("spinifex")
str(PimaIndiansDiabetes_long)
dat <- scale_sd(PimaIndiansDiabetes_long[, 1:6])
clas <- PimaIndiansDiabetes_long$diabetes

bas <- basis_pca(dat)
mv <- manip_var_of(bas)
m <- manual_tour(bas, mv)

tgt <- ggtour(m, dat, angle = .2) +
proto_default(list(color = clas, shape = clas))
```
PimaIndiansDiabetes_wide

## Not run:
animate_plotly(ggt)
## End(Not run)

PimaIndiansDiabetes_wide

*Pima Indians Diabetes Dataset, wide*

### Description

The data set PimaIndiansDiabetes2 contains a corrected version of the original data set. While the UCI repository index claims that there are no missing values, closer inspection of the data shows several physical impossibilities, e.g., blood pressure or body mass index of 0. In PimaIndiansDiabetes2, all zero values of glucose, pressure, triceps, insulin and mass have been set to NA, see also Wahba et al (1995) and Ripley (1996).

### Usage

PimaIndiansDiabetes_wide

### Format

A data frame with 392 observations of 8 numeric variables, and target factor diabetes.

- `pregnant`, Number of times pregnant
- `glucose`, Plasma glucose concentration (glucose tolerance test)
- `pressure`, Diastolic blood pressure (mm Hg)
- `triceps`, Triceps skin fold thickness (mm)
- `insulin`, 2-Hour serum insulin (mu U/ml)
- `mass`, Body mass index (weight in kg/(height in m)^2)
- `pedigree`, Diabetes pedigree function
- `age`, Age (years)
- `diabetes`, Class variable (test for diabetes), either "pos" or "neg"

### Details

This is a cleaned subset of mlbench's PimaIndiansDiabetes2. See `help(PimaIndiansDiabetes2, package = "mlbench")`.

Replicating this dataset:

```r
require("mlbench")
data(PimaIndiansDiabetes2)

d <- PimaIndiansDiabetes2
d <- d[complete.cases(d), ]  ## Remove ~350 row-wise incomplete rows
PimaIndiansDiabetes_wide <- d
## save(PimaIndiansDiabetes_wide, file = "./data/PimaIndiansDiabetes_wide.rda")
```
Examples

library("spinifex")
str(PimaIndiansDiabetes_wide)
dat <- scale_sd(PimaIndiansDiabetes_wide[, 1:8])
clas <- PimaIndiansDiabetes_wide$diabetes

bas <- basis_pca(dat)
mv <- manip_var_of(bas)
mt <- manual_tour(bas, mv)

ggt <- ggtour(mt, dat, angle = .2) + proto_default(list(color = clas, shape = clas))
## Not run:
animate_plotly(ggt)
## End(Not run)

---

**play_manual_tour**  
*Animate a manual tour*

**Description**

[Supserseded], see **ggtour**. Performs the a manual tour and returns an animation of render_type. For use with **tourr::save_history()** tour paths see **play_tour_path()**.

**Usage**

```r
play_manual_tour(
  basis = NULL,
  data,
  manip_var,
  theta = NULL,
  phi_min = 0L,
  phi_max = 0.5 * pi,
  angle = 0.05,
  render_type = render_plotly,
  ...
)
```

**Arguments**

- **basis**: A (p, d) orthonormal numeric matrix. The linear combination the original variables contribute to projection space. Defaults to NULL, generating a random basis.
- **data**: (n, p) dataset to project, consisting of numeric variables.
- **manip_var**: Integer column number or string exact column name of the. variable to manipulate. Required, no default.
theta  Angle in radians of "in-plane" rotation, on the xy plane of the reference frame. Defaults to theta of the basis for a radial tour.

phi_min Minimum value phi should move to. Phi is angle in radians of the "out-of-plane" rotation, the z-axis of the reference frame. Required, defaults to 0.

phi_max Maximum value phi should move to. Phi is angle in radians of the "out-of-plane" rotation, the z-axis of the reference frame. Required, defaults to pi/2.

angle Target distance (in radians) between steps. Defaults to .05.

render_type Which graphics to render to. Defaults to render_plotly.

... Optionally pass additional arguments to render_ and the function used in render_type.

Value
An animation of a radial tour.

See Also
render_ For arguments to pass into ....

Examples

dat_std <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)

## Not run:
play_manual_tour(basis = bas, data = dat_std, manip_var = mv)

play_manual_tour(basis = bas, data = dat_std, manip_var = mv, theta = .5 * pi, axes = "right", fps = 5, angle = .08, phi_min = 0, phi_max = 2 * pi, aes_args = list(color = clas, shape = clas), identity_args = list(size = 1.5, alpha = .7), ggproto = list(ggplot2::theme_void(), ggplot2::ggtitle("My title")), render_type = render_gganimate)

## Saving output may require additional setup
if(F){ ## Don't run by mistake

## Export plotly .html widget
play_manual_tour(basis = bas, data = dat_std, manip_var = 6, render_type = render_plotly, html_filename = "myRadialTour.html")

## Export gganimate .gif
play_manual_tour(basis = bas, data = dat_std, manip_var = 1, render_type = render_gganimate, gif_filename = "myRadialTour.gif", gif_path = "/output")

## End(Not run)
play_tour_path  Animates the provided tour path.

Description

[Superseded], see ggtour. Takes the result of tourr::save_history() or manual_tour(), interpolates over the path and renders into a specified render_type.

Usage

```r
play_tour_path(
  tour_path,
  data = NULL,
  angle = 0.05,
  render_type = render_plotly,
  ...
)
```

Arguments

tour_path  The result of tourr::save_history() or manual_tour().
data  Optional, number of columns must match that of tour_path.
angle  Target distance (in radians) between steps. Defaults to .05.
render_type  Graphics to render to. Defaults to render_plotly, alternative use render_gganimate.
...  Optionally pass additional arguments to render_ and the function used in render_type.

See Also

render_  For arguments to pass into ....

Examples

```r
dat_std <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas <- basis_pca(dat_std)

## Tour history from tourr::save_history
g_path <- tourr::save_history(dat_std, tour_path = tourr::grand_tour(), max = 5)

## Not run:
play_tour_path(tour_path = g_path, data = dat_std)
play_tour_path(tour_path = g_path, data = dat_std, axes = "bottomleft", angle = .08, fps = 8, aes_args = list(color = clas, shape = clas), identity_args = list(size = 1.5, alpha = .7), ggproto =
```
proto_basis

## Saving a .gif (may require additional setup)
if(F) {
  ## Don't run by mistake
  ## Export plotly .html widget
  play_tour_path(tour_path = tpath, data = dat_std,
                 render_type = render_plotly,
                 html_filename = "myRadialTour.html")

  ## Export gganimate .gif
  play_tour_path(tour_path = tpath, data = dat_std,
                 render_type = render_gganimate,
                 gif_path = "myOutput", gif_filename = "myRadialTour.gif")
}
## End (Not run)

---

**proto_basis**  
*Tour proto for a 2D and 1D basis axes respectively*

### Description

Adds basis axes to the animation, the direction and magnitude of contributions of the variables to the projection space inscribed in a unit circle for 2D or rectangle of unit width for 1D.

### Usage

```r
proto_basis(
  position = c("left", "center", "right", "bottomleft", "topright", "off"),
  manip_col = "blue",
  line_size = 1,
  text_size = 5
)
```

```r
proto_basis1d(
  position = c("top1d", "floor1d", "off"),
  manip_col = "blue",
  segment_size = 2,
  text_size = 5
)
```

### Arguments

- **position**: The position, to place the basis axes relative to the centered data. Expects one of ("left", "center", "right", "bottomleft", "topright", "off"), defaults to "left".
- **manip_col**: The color to highlight the manipulation variable with. Not applied if the tour isn’t a manual tour. Defaults to "blue".
line_size (2D bases only) the thickness of the lines used to make the axes and unit circle. Defaults to 1.

text_size Size of the text label of the variables.

segment_size (1D bases only) the width thickness of the rectangle bar showing variable magnitude on the axes. Defaults to 2.

See Also

Other ggtour proto: `filmstrip()`, `ggtour()`, `proto_default()`, `proto_density()`, `proto_hex()`, `proto_highlight()`, `proto_origin()`, `proto_point()`, `proto_text()`

Other ggtour proto: `filmstrip()`, `ggtour()`, `proto_default()`, `proto_density()`, `proto_hex()`, `proto_highlight()`, `proto_origin()`, `proto_point()`, `proto_text()`

Examples

dat <- scale_sd(tourr::flea[, 1:6])
clas <- tourr::flea$species
bas <- basis_pca(dat)
mv <- manip_var_of(bas)

## 2D case:
mt_path <- manual_tour(bas, manip_var = mv)

ggt <- ggtour(mt_path, dat, angle = .2) +
    proto_basis()
## Not run:
animate_plotly(ggt)
## End(Not run)

## Customize basis

## 1D case:
bas1d <- basis_pca(dat, d = 1)
mv <- manip_var_of(bas, 3)
mt_path1d <- manual_tour(bas1d, manip_var = mv)

ggt1d <- ggtour(mt_path1d, dat, angle = .2) +
    proto_basis1d()
## Not run:
animate_plotly(ggt1d)
## End(Not run)
**proto_default**  
*Wrapper function for default 2D/1D tours respectively.*

**Description**

An easier way to get to default 2D tour settings. Returns a list of `proto_origin()`, `proto_point(...)`, `proto_basis()` for 2D. Returns a list of `proto_origin1d()`, `proto_density(...)`, `proto_basis1d()` for 1D.

**Usage**

```r
proto_default(aes_args = list(), identity_args = list(alpha = 0.9))
```

```r
proto_default1d(aes_args = list(), identity_args = list(alpha = 0.7))
```

**Arguments**

- **aes_args**: A list of aesthetic arguments to passed to `geom_point(aes(X))`. Any mapping of the data to an aesthetic, for example, `geom_point(aes(color = myCol, shape = myCol))` becomes `aes_args = list(color = myCol, shape = myCol)`.
- **identity_args**: A list of static, identity arguments passed into `geom_point()`, but outside of `aes()`, for instance `geom_point(aes(...), size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`.

**See Also**

Other ggtour proto: `filmstrip()`, `ggtour()`, `proto_basis()`, `proto_density()`, `proto_hex()`, `proto_highlight()`, `proto_origin()`, `proto_point()`, `proto_text()`

Other ggtour proto: `filmstrip()`, `ggtour()`, `proto_basis()`, `proto_density()`, `proto_hex()`, `proto_highlight()`, `proto_origin()`, `proto_point()`, `proto_text()`

**Examples**

```r
dat <- scale_sd(tourr::flea[, 1:6])
clas <- tourr::flea$species

## 2D case:
bas <- basis_pca(dat)
mv <- manip_var_of(bas)
mt_path <- manual_tour(bas, mv)

ggt <- ggtour(mt_path, dat) +
  proto_default(list(color = clas, shape = clas))

## Not run:
animate_plotly(ggt)
## End(Not run)
## 1D case:

gt_path <- tourr::save_history(dat, grand_tour(d = 1), max_bases = 3)
```
proto_density

Tour proto for data, 1D density, with rug marks

Description

Adds `geom_density()` and `geom_rug()` of the projected data. Density position = "stack" does not work with `animate_plotly()`, GH issue is open.

Usage

```r
proto_density(
  aes_args = list(),
  identity_args = list(alpha = 0.7),
  density_position = c("identity", "stack", "fill"),
  do_add_rug = TRUE
)
```

Arguments

- **aes_args**: A list of aesthetic arguments to passed to `geom_point(aes(X))`. Any mapping of the data to an aesthetic, for example, `geom_point(aes(color = myCol, shape = myCol))` becomes `aes_args = list(color = myCol, shape = myCol)`.
- **identity_args**: A list of static, identity arguments passed into `geom_point()`, but outside of `aes()`, for instance `geom_point(aes(...), size = 2, alpha = 0.7)` becomes `identity_args = list(size = 2, alpha = 0.7)`.
- **density_position**: The `ggplot2` position of `geom_density()`. Either c("identity", "stack"), defaults to "identity". Warning: "stack" does not work with `animate_plotly()` at the moment.
- **do_add_rug**: Logical, weather or not to add the rug marks below the density curves.

See Also

Other ggtour proto: `filmstrip()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_hex()`, `proto_highlight()`, `proto_origin()`, `proto_point()`, `proto_text()`
Examples

```r
dat <- scale_sd(tourr::flea[, 1:6])
clas <- tourr::flea$species
gt_path <- save_history(dat, grand_tour(), max = 3)

ggt <- ggtour(gt_path, dat) +
    proto_density(aes_args = list(color = clas, fill = clas)) +
    proto_basis1d()

## Not run:
animate_plotly(ggt)
## End(Not run)
```

proto_hex

Tour proto for data, hexagonal heatmap

Description

Adds `geom_hex()` of the projected data. Does not display hexagons in plotly animations; will not work with `animate_plotly()`.

Usage

```r
proto_hex(aes_args = list(), identity_args = list(), bins = 30)
```

Arguments

- `aes_args` A list of aesthetic arguments to passed to `geom_point(aes(X)). Any mapping of the data to an aesthetic, for example, `geom_point(aes(color = myCol, shape = myCol))` becomes `aes_args = list(color = myCol, shape = myCol)`.

- `identity_args` A list of static, identity arguments passed into `geom_point()`, but outside of `aes()`, for instance `geom_point(aes(...), size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`.

- `bins` Numeric vector giving number of bins in both vertical and horizontal directions. Defaults to 30.

See Also

Other ggtour proto: `filmstrip()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_highlight()`, `proto_origin()`, `proto_point()`, `proto_text()`

Examples

```r
raw <- ggplot2::diamonds
dat <- scale_sd(raw[1:10000, c(1, 5:6, 8:10)])
gt_path <- save_history(dat, grand_tour(), max = 3)

## 10000 rows is quite heavy to animate.
## Decrease the points drawn in each frame by aggregating many points into
### a hexagon heatmap, using geom_hex!

ggp <- ggtour(gt_path, dat) +
    proto_basis() +
    proto_hex(bins = 20)

## Hexagons don’t show up in plotly animation.
## Not run:
animate_gganimate(ggp)
## End(Not run)

---

**proto_highlight**

#### Tour proto highlighting specified points

**Description**

A `geom_point` or `geom_segment(*1d)` call to draw attention to a subset of points. Subset the projected data frames to the specified `rownum_index` of the original data frame with specified highlighting aesthetics. The order you apply highlighting is important when using with other `proto_*` functions.

**Usage**

```r
proto_highlight(
    rownum_index,
    aes_args = list(),
    identity_args = list(color = "red", size = 5, shape = 8),
    mark_initial = if (length(rownum_index) == 1) TRUE else FALSE
)
```

```r
proto_highlight1d(
    rownum_index,
    aes_args = list(),
    identity_args = list(color = "red", linetype = 2, alpha = 0.9),
    mark_initial = if (length(rownum_index) == 1) TRUE else FALSE
)
```

**Arguments**

- `rownum_index` One or more integers, the row numbers of the to highlight. Should be within `1:n`, the rows of the original data.
- `aes_args` A list of aesthetic arguments to be passed to `geom_point(aes(X))`. Any mapping of the data to an aesthetic, for example, `geom_point(aes(color = myCol, shape = myCol))` becomes `aes_args = list(color = myCol, shape = myCol)`.
- `identity_args` A list of static, identity arguments passed into `geom_point()`, but outside of `aes()`. For example, `geom_point(aes(...), size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`. Typically a single numeric for point size, alpha, or similar.
- `mark_initial` Logical, whether or not to leave a fainter mark at the subset’s initial position. By default: TRUE if single row number given else FALSE.
See Also

Other ggtour proto: `filmstrip()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_hex()`, `proto_origin()`, `proto_point()`, `proto_text()`

Other ggtour proto: `filmstrip()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_hex()` , `proto_origin()`, `proto_point()`, `proto_text()`

Examples

dat <- scale_sd(tourr::flea[, 1:6])
clas <- tourr::flea$species  
gt_path <- tourr::save_history(dat, grand_tour(), max_bases = 5)

## d = 2 case, Highlighting 1 obs defaults mark_initial to TRUE.

```r
## Not run:
gt <- ggtour(gt_path, dat) +
      proto_highlight(rownum_index = 5) +
      proto_point()
## End(Not run)
```

## Custom aesthetics. Highlighting multiple points defaults mark_initial to FALSE

```r
## Not run:
gt2 <- ggtour(gt_path, dat) +
      proto_highlight(rownum_index = c(2, 6, 19),
                      identity_args = list(color = "blue", size = 4, shape = 2)) +
      proto_point(list(color = clas, shape = clas),
                   list(size = 2, alpha = .7))
## End(Not run)
```

## 1D case:

```r
## Not run:
gt <- ggtour(gt_path, dat) +
      proto_default1d(list(fill = clas, color = clas)) +
      proto_highlight1d(rownum_index = 7)
## End(Not run)
```

```r
gt2 <- ggtour(gt_path, dat) +
      proto_default1d(list(fill = clas, color = clas)) +
      proto_highlight1d(rownum_index = c(2, 6, 7))
## End(Not run)
```
proto_origin

Description

Adds a zero mark showing the location of the origin for the central data area.

Usage

```
proto_origin(
    identity_args = list(color = "grey60", size = 0.5, alpha = 0.9),
    tail_size = 0.05
)
```

```
proto_origin1d(identity_args = list(color = "grey60", size = 0.5, alpha = 0.9))
```

Arguments

- **identity_args**: A list of static, identity arguments passed into `geom_point()`, but outside of `aes()`, for instance `geom_point(aes(...), size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`.
- **tail_size**: How long the origin mark should extended relative to the observations. Defaults to .05, 5% of the projection space.

See Also

Other `ggtour proto`: `filmstrip()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_hex()`, `proto_highlight()`, `proto_point()`, `proto_text()`

Examples

```
dat <- scale_sd(tourr::flea[, 1:6])
clas <- tourr::flea$species

## 2D case:
gt_path <- tourr::save_history(dat, grand_tour(), max_bases = 5)

ggt <- ggtour(gt_path, dat, angle = .1) +
    proto_origin() +
    proto_point()

## Not run:
animate_plotly(ggt)

## End(Not run)

## 1D case:
gt_path1d <- tourr::save_history(dat, grand_tour(d = 1), max_bases = 5)

ggt <- ggtour(gt_path1d, dat) +
    proto_origin1d() +
    proto_density(list(fill = clas, color = clas))

## Not run:
animate_plotly(ggt)
```
proto_point

## End(Not run)

### proto_point

#### Tour proto for data point

**Description**

Adds `geom_point()` of the projected data.

**Usage**

```r
proto_point(aes_args = list(), identity_args = list())
```

**Arguments**

- `aes_args`: A list of aesthetic arguments to passed to `geom_point(aes(X))`. Any mapping of the data to an aesthetic, for example, `geom_point(aes(color = myCol, shape = myCol))` becomes `aes_args = list(color = myCol, shape = myCol)`.
- `identity_args`: A list of static, identity arguments passed into `geom_point()`, but outside of `aes()`, for instance `geom_point(aes(...), size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`.

**See Also**

Other `ggtour proto`: `filmstrip()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_hex()`, `proto_highlight()`, `proto_origin()`, `proto_text()`

**Examples**

```r
dat <- scale_sd(tourr::flea[, 1:6])
clas <- tourr::flea$species
gt_path <- tourr::save_history(dat, grand_tour(), max_bases = 5)

ggt <- ggtour(gt_path, dat, angle = .1) +
  proto_point()
## Not run:
animate_plotly(ggt)
## End(Not run)

ggt2 <- ggtour(gt_path, dat) +
  proto_point(list(color = clas, shape = clas),
              list(size = 2, alpha = .7))
## Not run:
animate_plotly(ggt2)
## End(Not run)
```
**proto_text**  
*Tour proto for data, text labels*

**Description**
Adds `geom_text()` of the projected data.

**Usage**

```r
proto_text(
  aes_args = list(),
  identity_args = list(nudge_x = 0.05),
  rownum_index = NULL
)
```

**Arguments**

- **aes_args**  
  A list of aesthetic arguments to passed to `geom_point(aes(X))`. Any mapping of the data to an aesthetic, for example,  
  `geom_point(aes(color = myCol,shape = myCol))` becomes `aes_args = list(color = myCol,shape = myCol)`.

- **identity_args**  
  A list of static, identity arguments passed into `geom_point()`, but outside of `aes()`, for instance  
  `geom_point(aes(...),size = 2,alpha = .7)` becomes `identity_args = list(size = 2,alpha = .7)`.

- **rownum_index**  
  One or more integers, the row numbers of the to subset to. Should be within 1:n,  
  the rows of the original data. Defaults to NULL, labeling all rows.

**See Also**
Other `ggtour proto`: `filmstrip()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`,  
`proto_hex()`, `proto_highlight()`, `proto_origin()`, `proto_point()`

**Examples**

```r
dat <- scale_sd(tourr::flea[, 1:6])
clas <- tourr::flea$species
bas <- basis_pca(dat)
mv <- manip_var_of(bas)
gt_path <- save_history(dat, tourr::grand_tour(), max_bases = 5)
ggt <- ggtour(gt_path, dat, angle = .2) +
  proto_text(list(color = clas))
## Not run:
animate_plotly(ggt)
## End(Not run)

## Custom labels, subset of points

ggt2 <- ggtour(gt_path, dat) +
  proto_text(list(color = clas, size = as.integer(clas)),
```

```r
```
render_

```r
list(alpha = .7),
rownames = 1:15)
```

## Not run:
animate_plotly(ggt2)
## End(Not run)

Prepare the ggplot object before passing to either animation package.

**Description**

[Superseded], see `ggtour`. Typically called by `render_plotly()` or `render_gganimate()`. Takes the result of `array2df()`, and renders them into a ggplot2 object.

**Usage**

```r
render_(
  frames,
  axes = "center",
  manip_col = "blue",
  line_size = 1L,
  text_size = 5L,
  aes_args = list(),
  identity_args = list(),
  ggproto = list(theme_spinifex())
)
```

**Arguments**

- `frames` The result of `array2df()`, a long df of the projected frames.
- `axes` Position of the axes, expects one of: "center", "left", "right", "bottomleft", "topright", "off", or a map_absolute() call. Defaults to "center".
- `manip_col` String of the color to highlight the `manip_var`, if used. Defaults to "blue".
- `line_size` The size of the lines of the unit circle and variable contributions of the basis. Defaults to 1.
- `text_size` The size of the text labels of the variable contributions of the basis. Defaults to 1.
- `aes_args` A list of aesthetic arguments to passed to `geom_point(aes(X)). Any mapping of the data to an aesthetic, for example, `geom_point(aes(color = myCol, shape = myCol))` becomes `aes_args = list(color = myCol, shape = myCol)`.
- `identity_args` A list of static, identity arguments passed into `geom_point()`, but outside of `aes()`: `geom_point(aes(),X)`. Typically a single numeric for point size, alpha, or similar. For example, `geom_point(aes(),size = 2,alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`.
- `ggproto` A list of ggplot2 function calls. Anything that would be "added" to ggplot(); in the case of applying a theme, `ggplot() + theme_bw()` becomes `ggproto = list(theme_bw())`. Intended for aesthetic ggplot2 functions (not geom_* family).
Examples

```r
## Setup
dat_std <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)

mt_array <- manual_tour(basis = bas, manip_var = mv)
mt_df_ls <- array2df(basis_array = mt_array, data = dat_std)

## Required arguments
render_(frames = mt_df_ls)

## Full arguments
require("ggplot2")
render_(frames = mt_df_ls, axes = "left", manip_col = "purple",
aes_args = list(color = clas, shape = clas),
identity_args = list(size = 1.5, alpha = .7),
ggproto = list(theme_minimal(),
  ggtitle("My title"),
  scale_color_brewer(palette = "Set2")))
```

render_gganimate  **Render the frames as a gganimate animation.**

Description

[Superseded], see ggtour. Takes the result of array2df() and renders them into a gganimate animation.

Usage

```r
render_gganimate(
  fps = 8L,
  rewind = FALSE,
  start_pause = 0.5,
  end_pause = 1L,
  gif_filename = NULL,
  gif_path = NULL,
  gganimate_args = list(),
  ...
)
```

Arguments

- `fps`  Frames animated per second. Defaults to 8.
- `rewind`  Logical, should the animation play backwards after reaching the end? Default to FALSE.
render_gganimate

start_pause Number of seconds to pause on the first frame for. Defaults to .5.
end_pause Number of seconds to pause on the last frame for. Defaults to 1.
gif_filename Optional, saves the animation as a GIF to this string (without the directory path). Defaults to NULL (no GIF saved). For more output control, call gganimate::anim_save() on a return object of render_gganimate().
gif_path Optional, A string of the directory path (without the filename) to save a GIF to. Defaults to NULL (current work directory).
gganimate_args A list of arguments assigned to a vector passe outside of an aes() call. Anything that would be put in geom_point(aes(), X). Typically a single numeric for point size, alpha, or similar For example, geom_point(aes(), size = 2, alpha = .7) becomes identity_args = list(size = 2, alpha = .7).
... Passes arguments to render_(...).

See Also

render_ for ... arguments.

gganimate::anim_save for more control of .gif output.

Examples

dat_std <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)
mt <- manual_tour(basis = bas, manip_var = mv)
mt_df_ls <- array2df(basis_array = mt, data = dat_std)

## Not run:
render_gganimate(frames = mt_df_ls)

require("ggplot2")
render_gganimate(
  frames = mt_df_ls, axes = "bottomleft",
  fps = 10, rewind = TRUE, start_pause = 1, end_pause = 1.5,
  aes_args = list(color = clas, shape = clas),
  identity_args = list(size = 2, alpha = .7),
  ggproto = list(theme_void(),
    ggtitle("My title"),
    scale_color_brewer(palette = "Set2")))

## Saving a .gif(may require additional setup)
if(F) ## Don't run by mistake
  render_gganimate(frames = mt_df_ls, axes = "bottomleft",
    gif_filename = "myRadialTour.gif", gif_path = "./output")

## End(Not run)
render_plotly  
*Animation the frames as a HTML widget.*

**Description**

[Superseded], see [ggtour](#). Takes the result of `array2df()` and animations them via `{plotly}` into a self-contained HTML widget.

**Usage**

```r
render_plotly(fps = 8L, html_filename = NULL, save_widget_args = list(), ...)
```

**Arguments**

- `fps` Frames animated per second. Defaults to 8.
- `html_filename` Optional, saves the plotly object as an HTML widget to this string (without the directory path). Defaults to NULL (not saved). For more output control use `save_widget_args` or call `htmlwidgets::saveWidget()` on a return object of `render_plotly()`.
- `save_widget_args` A list of arguments to be called in `htmlwidgets::saveWidget()` when used with a `html_filename`.
- `...` Passes arguments to `render_(...)`.

**See Also**

- `render_` for ... arguments.
- `ggplotly` for source documentation of tooltip.
- `saveWidget` for more control of .html output.

**Examples**

```r
dat_std <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)
mt_array <- manual_tour(basis = bas, manip_var = mv)
mt_df_ls <- array2df(basis_array = mt_array, data = dat_std)

## Not run:
render_plotly(frames = mt_df_ls)
```

```r
require("ggplot2")
render_plotly(
  frames = mt_df_ls, axes = "bottomleft", fps = 10,
  aes_args = list(color = clas, shape = clas),
  identity_args = list(size = 1.5, alpha = .7),
)```
rotate_manip_space

Performs a rotation on the manipulation space of the given manip var.

Description
A specific R3 rotation of the manipulation space for a 2D tour. Typically called by manual_tour(). The first 2 columns are x and y in the projection plane. The 3rd column extends "in the z-direction" orthogonal to the projection plane.

Usage
rotate_manip_space(manip_space, theta, phi)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>manip_space</td>
<td>A (p, d+1) dim matrix (manipulation space) to be rotated.</td>
</tr>
<tr>
<td>theta</td>
<td>Angle (radians) of &quot;in-projection-plane&quot; rotation (ie. on xy- of the projection). Typically set by the manip_type argument in proj_data().</td>
</tr>
<tr>
<td>phi</td>
<td>Angle (radians) of &quot;out-of-projection-plane&quot; rotation (ie. into the z-direction of the manipulation space. Effectively changes the norm of the manip_var in the projection plane.</td>
</tr>
</tbody>
</table>

Value
A (p, d+1) orthonormal matrix of the rotated (manipulation) space. The first 2 columns are x and y in the projection plane. The 3rd column extends "in the z-direction" orthogonal to the projection plane.

Examples

```r
## Setup
dat_std <- scale_sd(wine[, 2:6])
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)
msp <- create_manip_space(basis = bas, manip_var = mv)
rotate_manip_space(msp, theta = runif(1, max = 2 * pi), phi = runif(1, max = 2 * pi))

## d = 1 case
bas1d <- basis_pca(dat_std, d = 1)
```
mv <- manip_var_of(bas1d)
msp <- create_manip_space(bas1d, mv)
rotate_manip_space(msp, theta = 0, phi = runif(1, max = 2 * pi))

---

**run_app**

Runs a shiny app demonstrating manual tours

**Description**

Runs a local shiny app that demonstrates manual tour and comparable traditional techniques for static projections of multivariate data sets.

**Usage**

```r
run_app(app_nm = "radial_tour", ...)
```

**Arguments**

- `app_nm` name of the shiny app to run. Expects "manual_tour".
- `...` Other arguments passed into `shiny::runApp()`. Such as `display.mode = "showcase"`.

**Value**

Runs a locally hosted shiny app.

**Examples**

```r
## Not run:
run_app("radial_tour")
run_app(app_nm = "radial_tour", display.mode = "showcase")
## End(Not run)
```

---

**save_history**

A wrapper muting the text byproduct of `tourr::save_history`

**Description**

A wrapper muting the text byproduct of `tourr::save_history`

**Usage**

```r
save_history(..., verbose = FALSE)
```
scale_sd

Arguments

... additional arguments passed to tour path
verbose Whether or not to suppress the text output byproduct from tourr::save_history(). Defaults to FALSE.

See Also
tourr::save_history

Examples
tour_path <- save_history(data = wine[, 2:6], grand_tour(), max_bases = 10)
dim(tour_path)

scale_sd

Preprocess numeric variables

Description

Centers and scales each column by standard deviation (sd) or to the interval (0, 1).

Usage

scale_sd(data)
scale_01(data)

Arguments

data Numeric matrix or data.frame of the observations.

Examples

scale_sd(data = wine[, 2:6])
scale_01(data = wine[, 2:6])
spinifex is a package that extends the package tourr. It builds the functionality for manual tours and allows other tours to be rendered by plotly or gganimate. Tours are a class of dynamic linear (orthogonal) projections of numeric multivariate data from $p$ down to $d$ dimensions that are viewed as an animation as $p$-space is rotated. Manual tours manipulate a selected variable, exploring how they contribute to the sensitivity of the structure in the projection. This is particularly useful after finding an interesting basis, perhaps via a guided tour optimizing the projection for some objective function.

Details

GitHub: https://github.com/nspyris/spinifex

See Also

manual_tour() ggtour() proto_default()
Examples

```r
theme_spinifex()

require("ggplot2")

ggplot(mtcars, aes(wt, mpg, color = as.factor(cyl))) +
  geom_point() + theme_spinifex()
```

Description

[Superseded], see `ggtour`. Projects the specified rotation as a 2D ggplot object. One static frame of manual tour. Useful for providing user-guided interaction.

Usage

```r
view_frame(
  basis = NULL,
  data = NULL,
  manip_var = NULL,
  theta = 0L,
  phi = 0L,
  basis_label = abbreviate(row.names(basis), 3L),
  rescale_data = FALSE,
  ...
)
```

Arguments

- **basis**: A (p, d) dim orthonormal numeric matrix. Defaults to NULL, giving a random basis.
- **data**: A (n, p) dataset to project, consisting of numeric variables.
- **manip_var**: Optional, number of the variable to rotate. If NULL, theta and phi must be 0 as is no manip space to rotate.
- **theta**: Angle in radians of "in-projection plane" rotation, on the xy plane of the reference frame. Defaults to 0, no rotation.
- **phi**: Angle in radians of the "out-of-projection plane" rotation, into the z-direction of the axes. Defaults to 0, no rotation.
- **basis_label**: Optional, character vector of p length, add name to the axes in the frame, defaults to 3 letter abbreviation of the original variable names.
- **rescale_data**: When TRUE scales the data to between 0 and 1. Defaults to FALSE.
- **...**: Optionally pass additional arguments to the proto_default for projection point aesthetics;
**view_manip_space**

Plot 2D projection frame and return the axes table.

### Description

Uses base graphics to plot the circle with axes representing the projection frame. Returns the corresponding table. Only works for 2d manual tours.

### Usage

```r
view_manip_space(
  basis,
  manip_var,
  tilt = 0.1 * pi,
  basis_label = abbreviate(row.names(basis), 3L),
  manip_col = "blue",
  manip_sp_col = "red",
  line_size = 1L,
)```

### Value

A ggplot object of the rotated projection.

### See Also

`proto_default` For arguments to pass into ....

### Examples

```r
## Setup
dat_std <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)

## Minimal example
## Not run:
view_frame(basis = bas)

## Typical example
view_frame(basis = bas, data = dat_std, manip_var = mv, axes = "left")

## Full example
rtheta <- runif(1, 0, 2 * pi)
rphi <- runif(1, 0, 2 * pi)
view_frame(basis = bas, data = dat_std, manip_var = mv,
           theta = rtheta, phi = rphi, basis_label = paste0("MyNm", 1:ncol(dat_std)),
           aes_args = list(color = clas, shape = clas),
           identity_args = list(size = 1.5, alpha = .7))
## End(Not run)
```
```r

view_manip_space

  text_size = 5L,
  ggproto = list(theme_spinifex())
)

Arguments

  basis       A (p, d) orthonormal numeric matrix. The linear combination the original variables contribute to projection space. Required, no default.
  manip_var   Number of the column/dimension to rotate.
  tilt        angle in radians to rotate the projection plane. Defaults to .1 * pi.
  basis_label Optional, character vector of p length, add name to the axes in the frame, defaults to 3 letter abbreviation of the original variable names.
  manip_col   String of the color to highlight the manip_var.
  manip_sp_col Color to illustrate the z direction, orthogonal to the projection plane.
  line_size   The size of the lines of the unit circle and variable contributions of the basis. Defaults to 1.
  text_size   The size of the text labels of the variable contributions of the basis. Defaults to 5.
  ggproto     A list of ggplot2 function calls. Anything that would be "added" to ggplot(); in the case of applying a theme, ggplot() + theme_bw() becomes ggproto = list(theme_bw()). Intended for aesthetic ggplot2 functions (not geom_* family).

Value

  ggplot object of the basis.

Examples

  dat_std <- scale_sd(wine[, 2:6])
  bas <- basis_pca(dat_std)
  mv <- manip_var_of(bas)

  view_manip_space(basis = bas, manip_var = mv)
  view_manip_space(basis = bas, manip_var = mv, 
                   tilt = 2/12 * pi, basis_label = paste0("MyNm", 1:ncol(dat_std)), 
                   manip_col = "purple", manip_sp_col = "orange", 
                   ggproto = list(ggplot2::theme_void(), ggplot2::ggtitle("My title")))
```
Sample dataset of daily weather observations from Canberra airport in Australia.

Description

One year of daily weather observations collected from the Canberra airport in Australia was obtained from the Australian Commonwealth Bureau of Meteorology and processed to create this sample dataset for illustrating data mining using R and Rattle.

Usage

weather

Format

A data frame of 354 observations of 20 variables. One year of daily observations of weather variables at Canberra airport in Australia between November 1, 2007 and October 31, 2008.

- Date, The date of observation (Date class).
- MinTemp, The minimum temperature in degrees Celsius.
- MaxTemp, The maximum temperature in degrees Celsius.
- Rainfall, The amount of rainfall recorded for the day in mm.
- Evaporation, The "Class A pan evaporation" (mm) in the 24 hours to 9am.
- WindSpeed3pm, Wind speed (km/hr) averaged over 10 minutes prior to 3pm.
- Humid9am, Relative humidity (percent) at 9am.
- Humid3pm, Relative humidity (percent) at 3pm.
- Pressure9am, Atmospheric pressure (hpa) reduced to mean sea level at 9am.
- Pressure3pm, Atmospheric pressure (hpa) reduced to mean sea level at 3pm.
- Cloud9am, Fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eightths. It records how many eighths of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.
- Cloud3pm, Fraction of sky obscured by cloud (in "oktas": eighths) at 3pm. See Cloud9am for a description of the values.
- Temp9am, Temperature (degrees C) at 9am.
- Temp3pm, Temperature (degrees C) at 3pm.
- RISK_MM, The amount of rain. A kind of measure of the "risk".
- RainToday, Factor: "yes" if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0.
- RainTomorrow, Factor: "yes" if it rained the following day, the target variable.

Details

The data has been processed to provide a target variable `RainTomorrow` (whether there is rain on the following day - No/Yes) and a risk variable `RISK_MM` (how much rain recorded in millimeters). Various transformations were performed on the source data. The dataset is quite small and is useful only for repeatable demonstration of various data science operations.

This is a cleaned subset of `rattle::weather`.

Replicating this dataset:

```r
require("rattle")
d <- rattle::weather[, c(1, 3:7, 9, 12:21, 23, 22, 24)]
d <- d[complete.cases(d), ] ## Remove ~12 row-wise incomplete rows
d <- as.data.frame(d) ## Remove tibble dependency
weather <- d
## save(weather, file = ".\data\weather.rda")
```

Source


rattle, R package. G. Williams, 2020. rattle: Graphical User Interface for Data Science in R [https://CRAN.R-project.org/package=rattle](https://CRAN.R-project.org/package=rattle)

Examples

```r
library("spinifex")
str(spinifex::weather)
dat <- scale_sd(spinifex::weather[, 2:18])
clas <- spinifex::weather$RainTomorrow

bas <- basis_pca(dat)
mv <- manip_var_of(bas)
mt <- manual_tour(bas, mv)

ggt <- ggtour(mt, dat, angle = .2) +
  proto_default(list(color = clas, shape = clas))
## Not run:
  animate_plotly(ggt)
## End(Not run)
```

Description

The wine dataset contains the results of a chemical analysis of wines grown in a specific area of Italy. Three types of wine are represented in the 178 samples, with the results of 13 chemical analyses recorded for each sample. The Type variable has been transformed into a categorical variable.
Usage

wine

Format

A data frame of 178 observations of target class Type and 12 numeric variables:

- Type, The type of wine, the target factor, 1 (59 obs), 2 (71 obs), and 3 (48 obs).
- Alcohol, Alcohol
- Malic, Malic acid
- Ash, Ash
- Alcalinity, Alcalinity of ash
- Magnesium, Magnesium
- Phenols, Total phenols
- Flavanoids, Flavanoids
- Nonflavanoids, Nonflavanoid phenols
- Proanthocyanins, Proanthocyanins
- Color, Color intensity
- Hue, Hue
- Dilution, D280/OD315 of diluted wines
- Proline, Proline

Details

The data contains no missing values and consist of only numeric data, with a three class target variable (Type) for classification.

Replicating this dataset:

```r
require("rattle")
str(rattle::wine)
## save(wine, file = "./data/wine.rda")
```

Source

rattle, R package. G. Williams, 2020. rattle: Graphical User Interface for Data Science in R
https://CRAN.R-project.org/package=rattle

Examples

library("spinifex")
str(wine)
dat <- scale_sd(wine[, 2:6])
clas <- wine$Type

bas <- basis_pca(dat)
mv <- manip_var_of(bas)
mt <- manual_tour(bas, mv)

ggt <- ggtour(mt, dat, angle = .2) +
  proto_default(list(color = clas, shape = clas))
## Not run:
animate_plotly(ggt)
## End(Not run)
Index

* Internal utility
  .bind_elements2df, 3
  .init4proto, 3
  .lapply_rep_len, 4
  interpolate_manual_tour, 18

* Linear mapping
  map_absolute, 22
  map_relative, 23

* basis identifiers
  basis_guided, 9
  basis_half_circle, 10
  basis_odp, 10
  basis_olda, 11
  basis_onpp, 12
  basis_pca, 13

* datasets
  .init4proto, 3
  BreastCancer, 14
  penguins, 24
  PimaIndiansDiabetes_long, 25
  PimaIndiansDiabetes_wide, 27
  weather, 52
  wine, 53

* ggtour animator
  animate_gganimate, 5
  animate_plotly, 6

* ggtour proto
  filmstrip, 16
  ggtour, 17
  proto_basis, 31
  proto_default, 33
  proto_density, 34
  proto_hex, 35
  proto_highlight, 36
  proto_origin, 37
  proto_point, 39
  proto_text, 40

* manual tour
  manip_var_of, 20
INDEX

play_tour_path, 30
plotly::ggplotly, 6
plotly::layout, 6
proto_basis, 16, 17, 31, 33–35, 37–40
proto_basis1d (proto_basis), 31
proto_def, (proto_default), 33
proto_def1d (proto_default), 33
proto_def2d (proto_default), 33
proto_default, 16, 17, 32, 33, 34, 35, 37–40, 50
proto_default(), 48
proto_default1d (proto_default), 33
proto_default2d, (proto_default), 33
proto_density, 16, 17, 32, 33, 34, 35, 37–40
proto_density1d (proto_density), 34
proto_hex, 16, 17, 32–34, 35, 37–40
proto_highlight, 16, 17, 32–35, 36, 38–40
proto_highlight1d (proto_highlight), 36
proto_highlight_2d (proto_highlight), 36
proto_origin, 16, 17, 32–35, 37, 39, 40
proto_origin1d (proto_origin), 37
proto_origin2d (proto_origin), 37
proto_point, 16, 17, 32–35, 37, 38, 39, 40
proto_points (proto_point), 39
proto_text, 16, 17, 32–35, 37–39, 40

Rdimtools::aux.graphnbd, 11, 13
Rdimtools::aux.graphnbd(), 12
Rdimtools::do.odp, 10, 11
Rdimtools::do.olda, 12
Rdimtools::do.onpp, 13
Rdimtools::do.pca, 13
render_, 29, 30, 41, 43, 44
render_gganimate, 42
render_plotly, 44
rotate_manip_space, 45
run_app, 46

save_history, 46
saveWidget, 44
scale_01 (scale_sd), 47
scale_axes, 22
scale_sd, 47
spinifex, 48

theme_spinifex, 48
tourr::guided_tour, 9
tourr::save_history, 8, 46, 47
view_frame, 49

view_manip_space, 50

weather, 52
wine, 53