Package 'spinifex'

April 18, 2021

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

Version 0.2.8

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:10.2307/1390747>]. Other types of tours, such as grand (random walk) and guided (optimizing some objective function) are available in the 'tourr' package ['Wickham' 'et' 'al.' (2011) <doi:10.18637/jss.v040.i02>]. 'spinifex' builds on 'tourr' 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 at ['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

Suggests dplyr, GGally, rmarkdown, htmlwidgets, knitr, testthat, 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 2021-04-18 14:50:02 UTC

R topics documented:

`array2df` *Turns a tour path array into a long data frame.*

Description

Typically called by a wrapper function, `play_manual_tour` or `play_tour_path`. Takes the result of `tourr::save_history()` or `manual_tour()` and restructures the data from an array to a long data frame for use in ggplots.

Usage

```
array2df(array, data = NULL, label = NULL)
```

Arguments

array	A (p, d, n_frames) array of a tour, the output of manual_tour().
data	Optional, (n, p) dataset to project, consisting of numeric variables.
label	Optional, labels for the reference frame of length 1 or the number of variables used. Defaults to an abbreviation of the variables.

Value

A list containing an array of basis frames (p, d, n_frames) and an array of data frames (n, d, n_frames) if data is present.

Examples

```
## Setup
dat_std <- tourr::rescale(wine[, 2:14])
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()
tour_array <- manual_tour(basis = bas, manip_var = mv)
str(
  array2df(array = tour_array, data = dat_std,
           label = paste0("MyLabs", 1:nrow(bas)))
)

## tourr::save_history tour array to long df, as used in play_tour_path()
hist_array <- tourr::save_history(data = dat_std, max_bases = 10)
str(
  array2df(array = hist_array, data = dat_std,
           label = paste0("MyLabs", 1:nrow(bas)))
)
```

---

as_history_array *Changes an array of bases into a "history_array" for use in tourr::interpolate*

Description

Attaches data to an array and assigns the custom class "history_array" as used in tourr. Typically called by other spinifex functions.

Usage

```
as_history_array(basis_array, data)
```

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.

Examples

```
dat_std <- scale_sd(wine[, 2:14])
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)
mt_array <- manual_tour(basis = bas, manip_var = mv)
as_history_array(mt_array, dat_std)
```

---

basis_guided *Solve for the last basis of a guided tour.*

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

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

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

```
dat_std <- scale_sd(wine[, 2:14])
basis_guided(data = dat_std, index_f = tourr::holes())

basis_guided(data = dat_std, index_f = tourr::cmass(),
             alpha = .4, cooling = .9, max.tries = 30)
```

---

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[, 2:14])
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 = 2L, 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`.

References

Li B, Wang C, Huang D (2009). "Supervised feature extraction based on orthogonal discriminant projection." Neurocomputing, 73(1-3), 191-196.

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[, 2:14])
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 = 2L)
```

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

Ye J (2005). "Characterization of a Family of Algorithms for Generalized Discriminant Analysis on Undersampled Problems." J. Mach. Learn. Res., 6, 483-502. ISSN 1532-4435.

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[, 2:14])
clas <- wine$Type
basis_olda(data = dat_std, class = clas)
```

basis_onpp *The basis of Orthogonal Locality Preserving Projection (OLPP)*

Description

Orthogonal Locality Preserving Projection (OLPP) is the orthogonal variant of LPP, a linear approximation to Laplacian Eigenmaps. It finds a linear approximation to the eigenfunctions of the Laplace-Beltrami operator on the graph-approximated data manifold. For the more details on `type` see `Rdimtools::aux.graphnbd()`.

Usage

```
basis_onpp(data, d = 2L, 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

He X (2005). Locality Preserving Projections. PhD Thesis, University of Chicago, Chicago, IL, USA.

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[, 2:14])
basis_onpp(data = dat_std)
```

basis_pca *The basis of Principal Component Analysis (PCA)*

Description

The basis of Principal Component Analysis (PCA)

Usage

```
basis_pca(data, d = 2L)
```

Arguments

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

Value

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

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[, 2:14])
basis_pca(data = dat_std)
```

BreastCancer *Wisconsin Breast Cancer Database*

Description

Formatted subset of mlbench's `BreastCancer` (not explicitly exported). See `help(BreastCancer, package = "mlbench")` for the original documentation.

Usage

```
BreastCancer
```

Format

Data frame with 675 observations of 11 variables: factor Id, 9 integer variables, and target factor Class.

Details

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.

Data frame with 675 observations of 11 variables: factor Id, 9 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 Epthelial cell size
- Bare.nuclei, Bare nuclei
- Bl.cromatin, Bland chromatin
- Normal.nucleoli, Normal Nucleoli
- Mitoses, Mitoses
- Class, Class of cancer, either "benign" or "malignant"

Reproducing this dataset:

```
require("mlbench")
data(BreastCancer)

d <- BreastCancer
d <- d[!duplicated(d), ] ## Remove 8 duplicate rows
d <- d[complete.cases(d), ] ## Remove 16 row-wise incomplete rows
mat <- as.matrix(d[ , 2:10])
mat <- sapply(mat, as.integer)
BreastCancer <-
  data.frame(Id = as.factor(d$Id), mat, Class = as.factor(d$Class))
## save(BreastCancer, file = "./data/BreastCancer.rda")
```

Examples

```
str(BreastCancer)
## Not run:
dat <- scale_sd(BreastCancer[, 2:10])
clas <- BreastCancer$Class
bas <- prcomp(dat)$rotation[, 1:2]
mvar <- which(abs(bas[, 1]) == max(abs(bas[, 1])))

play_manual_tour(basis = bas, data = dat, manip_var = mvar,
```

```
                render_type = render_gganimate, color = clas, shape = clas)

## End(Not run)
```

`create_manip_space` *Create a manipulation space to rotate the manip 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)
```

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.

Value

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

Examples

```
## Setup
dat_std <- scale_sd(wine[, 2:14])
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)

create_manip_space(basis = bas, manip_var = mv)
```

`is_orthonormal` *Test if a numeric matrix is orthonormal.*

Description

Handles more cases than tourr::is_orthonormal().

Usage

```
is_orthonormal(x, tol = 0.001)
```

Arguments

x	Numeric matrix to test the orthonormality of.
tol	Max tolerance of floating point differences. Elecment-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 in the first column of the supplied 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 = 1L)
```

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

```
dat_std <- scale_sd(wine[, 2:14])
bas <- basis_pca(dat_std)
manip_var_of(basis = bas)
```

`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

```
manual_tour(
  basis,
  manip_var,
  theta = NULL,
  phi_min = 0L,
  phi_max = 0.5 * pi,
  angle = 0.05,
  ...
)
```

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.
`angle`	Target distance (in radians) between steps. Defaults to .05.
`...`	Handles unused arguments that are being also being passed from `play_manual_tour()` to `render_()`.

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

```
## Setup
dat_std <- scale_sd(wine[, 2:14])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)

## Required arguments
manual_tour(basis = bas, manip_var = mv)

## Full arguments
manual_tour(basis = bas, manip_var = mv,
            theta = pi / 2, phi_min = pi / 16, phi_max = pi, angle = .8)
```

---

pan_zoom *Pan (offset) and zoom (scale) a 2 column matrix or dataframe.*

Description

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

Usage

```
pan_zoom(pan = c(0L, 0L), zoom = c(1L, 1L), x = NULL)
```

Arguments

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

Value

Scaled and offset `x`.

See Also

`scale_axes` for preset choices.

Examples

```
rb <- tourr::basis_random(6, 2)
pan_zoom(pan = c(-1, 0), zoom = c(2/3, 2/3), x = rb)
```

PimaIndiansDiabetes_long

Pima Indians Diabetes Database

Description

Formatted subset of mlbench's PimaIndiansDiabetes2 (not explicitly exported). See help(PimaIndiansDiabetes2,packa = "mlbench") for the original documentation.

Usage

```
PimaIndiansDiabetes_long
```

Format

Data frame with 724 observations of 7 variables: 6 numeric variables, and target factor diabetes.

Details

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).

Data frame with 724 observations of 7 variables: 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"

Reproducing this dataset:

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

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

Examples

```
str(PimaIndiansDiabetes_long)
## Not run:
dat <- scale_sd(PimaIndiansDiabetes_long[, 1:6])
clas <- PimaIndiansDiabetes_long$diabetes
bas <- prcomp(dat)$rotation[, 1:2]
mvar <- which(abs(bas[, 1]) == max(abs(bas[, 1])))

play_manual_tour(basis = bas, data = dat, manip_var = mvar,
                 render_type = render_gganimate, color = clas, shape = clas)

## End(Not run)
```

PimaIndiansDiabetes_wide

Pima Indians Diabetes Database

Description

Formatted subset of mlbench's `PimaIndiansDiabetes2` (not explicitly exported). See `help(PimaIndiansDiabetes2,packa = "mlbench")` for the original documentation.

Usage

```
PimaIndiansDiabetes_wide
```

Format

Data frame with 392 observations of 9 variables: 8 numeric variables, and target factor diabetes.

Details

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).

Data frame with 392 observations of 9 variables: 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"

Reproducing this dataset:

```
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

```
str(PimaIndiansDiabetes_wide)
## Not run:
dat <- scale_sd(PimaIndiansDiabetes_wide[, 1:8])
clas <- PimaIndiansDiabetes_wide$diabetes
bas <- prcomp(dat)$rotation[, 1:2]
mvar <- which(abs(bas[, 1]) == max(abs(bas[, 1])))

play_manual_tour(basis = bas, data = dat, manip_var = mvar),
                 render_type = render_gganimate, color = clas, shape = clas)

## End(Not run)
```

`play_manual_tour` *Animate a manual tour*

Description

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

```
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[, 2:14])
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

Takes the result of `tourr::save_history()` or `manual_tour()`, interpolates over the path and renders into a specified `render_type`.

Usage

```
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

```
dat_std <- scale_sd(wine[, 2:14])
clas <- wine$Type
bas <- basis_pca(dat_std)

## Not run:
```

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

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 =
                 list(ggplot2::theme_void(), ggplot2::ggtitle("My title")),
               render_type = render_gganimate)

## 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)
```

---

render_ *Prepare the ggplot object before passing to either animation package.*

Description

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

Usage

```
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 pan_zoom() 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 5.
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

```
## Setup
dat_std <- scale_sd(wine[, 2:14])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)

manual_array <- manual_tour(basis = bas, manip_var = mv)
manual_df <- array2df(array = manual_array, data = dat_std)

## Required arguments
render_(frames = manual_df)

## Full arguments
require("ggplot2")
render_(frames = manual_df, 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

Takes the result of `array2df()` and renders them into a *gganimate* animation.

Usage

```
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.
`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[, 2:14])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)
manual_array <- manual_tour(basis = bas, manip_var = mv)
manual_df <- array2df(array = manual_array, data = dat_std)

## Not run:
render_gganimate(frames = manual_df)

require("ggplot2")
render_gganimate(frames = manual_df, 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 = manual_df, axes = "bottomleft",
                   gif_filename = "myRadialTour.gif", gif_path = "./output")
}

## End(Not run)
```

`render_plotly` *Animation the frames as a HTML widget.*

Description

Takes the result of `array2df()` and animations them via `{plotly}` into a self-contained HTML widget.

Usage

```
render_plotly(
  fps = 8L,
  tooltip = "none",
  html_filename = NULL,
  save_widget_args = list(),
  ...
)
```

Arguments

fps	Frames animated per second. Defaults to 8.
tooltip	Character vector of aesthetic mappings to show in the hover-over tooltip (passed to plotly::ggplot()). Defaults to "none". "all" shows all the aesthetic mappings. The order of text controls the order they appear. For example, tooltip = c("id", "frame", "x", "y", "category", "color").
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 .gif output.

---

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

manip_space	A (p, d+1) dim matrix (manipulation space) to be rotated.
theta	Angle (radians) of "in-projection-plane" rotation (ie. on xy- of the projection). Typically set by the manip_type argument in proj_data().
phi	Angle (radians) of "out-of-projection-plane" rotation (ie. into the z-direction of the manipulation space. Effectively changes the norm of the manip_var in the projection plane.

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

```
## Setup
dat_std <- scale_sd(wine[, 2:14])
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))
```

---

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

```
run_app(app_nm = "intro", ...)
```

Arguments

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

Value

Runs a locally hosted shiny app.

Examples

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

## End(Not run)
```

scale_axes *Returns the axis scale and position.*

Description

Typically called by other functions to scale the position of the axes relative to the data.

Usage

```
scale_axes(
  x,
  position = c("center", "left", "right", "bottomleft", "topright", "off",
    "pan_zoom() call;", pan_zoom(c(-1L, 0L), c(0.7, 0.7))),
  to = data.frame(x = c(-1L, 1L), y = c(-1L, 1L))
)
```

Arguments

x	Numeric table, first 2 columns and scaled and offset relative to the to argument.
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.

Value

Transformed values of x, dimension and class unchanged.

See Also

pan_zoom for more manual control.

Examples

```
rb <- tourr::basis_random(4, 2)

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

scale_sd *Preprocessing variable transformation*

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[, 2:14])
scale_01(data = wine[, 2:14])
```

spinifex *spinifex*

Description

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 tour, perhaps via a guided tour.

Details

Its main functions are:

- `run_app()`, running `run_app("intro")` will open an introductory `shiny` app demonstrating radial tours.
- `play_manual_tour()`, performs a manual tour, returning a `plotly` animate by default.
- `play_tour_path()`, turns a tour path into an animation, returning a `plotly` object by default.
- `view_frame()`, plot a basis set on a reference axis.
- `view_manip_space()`, plot a manipulation space highlighting the manip var.

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

Author(s)

Maintainer: Nicholas Spyrison <spyrison@gmail.com>

Authors:

- Dianne Cook [thesis advisor]

See Also

tourr (package)

`theme_spinifex` *A ggplot2 theme suggested for linear projections with spinifex. The default value for ggproto arguments in spinifex functions.*

Description

A ggplot2 theme suggested for linear projections with spinifex. The default value for ggproto arguments in spinifex functions.

Usage

```
theme_spinifex()
```

Examples

```
theme_spinifex()

require("ggplot2")
ggplot(mtcars, aes(wt, mpg, color = as.factor(cyl))) +
  geom_point() + theme_spinifex()
```

`view_frame` *Plot a single frame of a manual tour*

Description

Projects the specified rotation as a 2D ggplot object. One static frame of manual tour. Useful for providing user-guided interaction.

Usage

```
view_frame(
  basis = NULL,
  data = NULL,
  manip_var = NULL,
  theta = 0L,
  phi = 0L,
  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.
label	Optionally, provide a character vector of length p (or 1) to label the variable contributions to the axes, Default NULL, results in a 3 character abbreviation of the variable names.
rescale_data	When TRUE scales the data to between 0 and 1. Defaults to FALSE.
...	Optionally pass additional arguments to the render_type for projection point aesthetics;

Value

A ggplot object of the rotated projection.

Examples

```
dat_std <- scale_sd(wine[, 2:14])
clas <- wine$Type
bas <- basis_pca(dat_std)
mv <- manip_var_of(bas)

view_frame(basis = bas)

view_frame(basis = bas, data = dat_std, manip_var = mv)

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, label = paste0("MyNm", 1:ncol(dat_std)),
           aes_args = list(color = clas, shape = clas),
           identity_args = list(size = 1.5, alpha = .7),
           ggproto = list(ggplot2::theme_void(), ggplot2::ggtitle("My title")))
```

`view_manip_space` *Plot 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.

Usage

```
view_manip_space(
  basis,
  manip_var,
  tilt = 0.1 * pi,
  label = paste0("x", 1L:nrow(basis)),
  manip_col = "blue",
  manip_sp_col = "red",
  line_size = 1L,
  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.
`label`	Optional, character vector of p length, add name to the axes in the reference frame, typically the 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[, 2:14])
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, label = paste0("MyNm", 1:ncol(dat_std)),
                 manip_col = "purple", manip_sp_col = "orange",
                 ggproto = list(ggplot2::theme_void(), ggplot2::ggtitle("My title")))
```

---

weather *Sample dataset of daily weather observations from Canberra airport in Australia.*

Description

A subset from `rattle.data::weather`, instructions to reproduce below.

Usage

```
weather
```

Format

Data frame of 366 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.

Details

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.

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.

The source dataset is Copyright by the Australian Commonwealth Bureau of Meteorology and is provided as part of the rattle package with permission.

Data frame of 366 observations of 20 variables. One year of daily observations of weather variables at Canberra airport in Australia starting November 2007:

- 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.
- Sunshine, The number of hours of bright sunshine in the day.
- WindGustSpeed, The speed (km/h) of the strongest wind gust in the 24 hours to midnight.
- WindSpeed9am, Wind speed (km/hr) averaged over 10 minutes prior 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 eighths. 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.

Reproducing this dataset:

```
require("rattle.data")
weather <- weather[, c(1, 3:7, 9, 12:21, 23, 22, 24)]
## save(weather, file = "./data/weather.rda")
```

Source

The daily observations are available from http://www.bom.gov.au/climate/data. Copyright Commonwealth of Australia 2010, Bureau of Meteorology.

Definitions adapted from http://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml

References

Data source: http://www.bom.gov.au/climate/dwo/ and http://www.bom.gov.au/climate/data.

Examples

```
str(weather)
## Not run:
dat <- scale_sd(weather[, 2:18])
clas_x <- weather$RainToday
clas_y <- weather$RainTomorrow
bas <- prcomp(dat)$rotation[, 1:2]
mvar <- which(abs(bas[, 1]) == max(abs(bas[, 1])))

play_manual_tour(basis = bas, data = dat, manip_var = mvar,
                 render_type = render_gganimate, color = clas_y, shape = clas_x)

## End(Not run)
```

`wine` *The wine dataset from the UCI Machine Learning Repository.*

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

Data frame of 178 observations of 13 variables, target class `Type` and 12 numeric variables.

Details

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

Data frame of 178 observations of 13 variables, 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

Reproducing this dataset:

```
requireNamespace("rattle.data")
wine
```

Examples

```
str(wine)
## Not run:
dat <- scale_sd(wine[, 2:14])
clas <- wine$Type
bas <- prcomp(dat)$rotation[, 1:2]
mvar <- which(abs(bas[, 1]) == max(abs(bas[, 1])))

play_manual_tour(basis = bas, data = dat, manip_var = mvar,
                 render_type = render_gganimate, color = clas, shape = clas)

## End(Not run)
```

Index