Package ‘spinifex’

January 8, 2025

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

Version 0.3.8

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:10.1080/10618600.1997.10474754>. Other types of tours, such as grand (random walk) and guided (optimizing some objective function) are available in the 'tourr' package Wickham et al. <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 in Spyrison & Cook (2020) <doi:10.32614/RJ-2020-027>.

Depends R (>= 3.5.0), tourr

License MIT + file LICENSE

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

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

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

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

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Language en-US

Config/testthat/edition 3

NeedsCompilation no

Author Nicholas Spyrison [aut, cre] (<`https://orcid.org/0000-0002-8417-0212`>), Dianne Cook [aut, ths] (<`https://orcid.org/0000-0002-3813-7155`>)

Maintainer Nicholas Spyrison <`spyrison@gmail.com`>

Repository CRAN

Date/Publication 2025-01-08 22:10:02 UTC

Contents

`.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 the length of the data and bind as a column.

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()`

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
```

Format

An object of class `expression` of length 1.

See Also

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

Examples

```
## This expression. is not meant for external use.
```

`.lapply_rep_len` *Replicate all vector elements of a list*

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()`, `.init4proto`

Examples

```
## 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`	Number of Frames Per Second, the speed resulting animation.
`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()`, `filmstrip()`

Examples

```
library(spinifex)
dat     <- scale_sd(penguins_na.rm[, 1:4])
clas    <- penguins_na.rm$species
bas     <- basis_pca(dat)
mv      <- manip_var_of(bas)
mt_path <- manual_tour(bas, manip_var = mv)

ggt <- ggtour(mt_path, dat, angle = .3) +
  proto_default(aes_args = list(color = clas, shape = clas),
                identity_args = list(size = 1.5, alpha = .7))
## Not run:
## Default .gif rendering
animate_gganimate(ggt)

if(FALSE){ ## Don't accidentally save file
  ## Option arguments, rendering to default .gif
  anim <- animate_gganimate(
    ggt, fps = 10, rewind = TRUE,
    start_pause = 1, end_pause = 2,
    height = 10, width = 15, units = "cm", ## "px", "in", "cm", or "mm."
    res = 200 ## resolution, pixels per dimension unit I think
  )
  ## Save rendered animation
  gganimate::anim_save("my_tour.gif",
                       animation = anim,
                       path = "./figures")

  ## Alternative renderer saving directly to .mp4
  animate_gganimate(ggt, fps = 5,
    height = 4, width = 6, units = "in", ## "px", "in", "cm", or "mm."
    res = 200, ## resolution, pixels per dimension unit I think
    renderer = gganimate::av_renderer("./my_tour.mp4"))
}

## 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

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

Arguments

`ggtour`	A grammar of graphics tour with appended protos added. A return from `ggtour() + proto_*()`.
`fps`	Number of Frames Per Second, the speed resulting animation.
`...`	Other arguments passed to `plotly::ggplotly`.

See Also

Other ggtour animator: `animate_gganimate()`, `filmstrip()`

Examples

```
library(spinifex)
dat     <- scale_sd(penguins_na.rm[, 1:4])
clas    <- penguins_na.rm$species
bas     <- basis_pca(dat)
mv      <- manip_var_of(bas)
mt_path <- manual_tour(bas, manip_var = mv)

ggt <- ggtour(mt_path, dat, angle = .3) +
  proto_origin() +
  proto_basis() +
  proto_point(aes_args = list(color = clas, shape = clas),
              identity_args = list(size = 1.5, alpha = .7))

animate_plotly(ggt, width = 700, height = 450) ## pixels only, no resolution argument

## Example saving to a .html widget, may require additional setup.
if(FALSE){
  anim <- animate_plotly(ggt, fps = 10,
                         width = 700, height = 450) ## in pixels

  htmlwidgets::saveWidget(widget = anim,
                          file = "./figures/my_tour.html",
                          selfcontained = TRUE)}
```

`append_fixed_y` *Append a fixed vertical height*

Description

Adds/overwrites the y of the projected data. Usefully for 1D projections and appending information related to, but independent from the projection; model predictions or residuals for instance. Wants to be called early so that the following proto calls adopt the changes.

Usage

```
append_fixed_y(fixed_y)
```

Arguments

`fixed_y`	Vector of length of the data, values to fix vertical height. Typically related to but not an explanatory variable, for instance, predicted Y, or residuals.

See Also

Other ggtour proto functions: `facet_wrap_tour()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_density2d()`, `proto_hex()`, `proto_highlight()`, `proto_hline0()`, `proto_origin()`, `proto_point()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat     <- scale_sd(penguins_na.rm[, 1:4])
clas    <- penguins_na.rm$species
bas     <- basis_pca(dat)
mv      <- manip_var_of(bas)
mt_path <- manual_tour(bas, manip_var = mv)

# Fixed y height with related information, independent of a 1D tour
# _eg_ predictions or residuals.
message("don't forget to scale your fixed_y.")
dummy_y <- scale_sd(as.integer(clas) + rnorm(nrow(dat), 0, .5))
gt_path <- save_history(dat, grand_tour(d = 1), max_bases = 5)

message("append_fixed_y wants to be called early so other proto's adopt the fixed_y.")
ggt <- ggtour(gt_path, dat, angle = .3) +
  append_fixed_y(fixed_y = dummy_y) + ## insert/overwrites vertical values.
  proto_point(list(fill = clas, color = clas)) +
  proto_basis1d() +
  proto_origin()

animate_plotly(ggt)
```

`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 = NULL,
```

```
  data_label = rownames(data),
  do_center_frame = TRUE
)
```

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`	Labels for basis display, a character vector with length equal to the number of variables. Defaults to NULL; 3 character abbreviation from colnames of data or rownames of basis.
`data_label`	Labels for `plotly` tooltip display. Defaults to the rownames of data. If null, initializes to 1:nrow(data).
`do_center_frame`	Whether or not to center the mean within each animation frame. Defaults to TRUE.

Examples

```
## !!This function is not meant for external use!!
dat  <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas  <- basis_pca(dat)
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,
                         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, max_bases = 10)
ls_df_frames2 <- array2df(basis_array = gt_array, data = dat)
str(ls_df_frames2)
```

`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 = 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 producing functions: `basis_half_circle()`, `basis_odp()`, `basis_olda()`, `basis_onpp()`, `basis_pca()`

Examples

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

basis_guided(data = dat, 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 producing functions: `basis_guided()`, `basis_odp()`, `basis_olda()`, `basis_onpp()`, `basis_pca()`

Examples

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

`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`.

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 producing functions: `basis_guided()`, `basis_half_circle()`, `basis_olda()`, `basis_onpp()`, `basis_pca()`

Examples

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

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 producing functions: `basis_guided()`, `basis_half_circle()`, `basis_odp()`, `basis_onpp()`, `basis_pca()`

Examples

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

`basis_onpp` *The basis of Orthogonal Neighborhood Preserving Projection (ONPP)*

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

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 producing functions: `basis_guided()`, `basis_half_circle()`, `basis_odp()`, `basis_olda()`, `basis_pca()`

Examples

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

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

Description

The orthogonal linear components of the variables in the next largest direction of variance.

Usage

```
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 producing functions: `basis_guided()`, `basis_half_circle()`, `basis_odp()`, `basis_olda()`, `basis_onpp()`

Examples

```
dat <- scale_sd(wine[, 2:6])
basis_pca(data = dat)
```

`BreastCancer_na.rm` *Wisconsin Breast Cancer Database*

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_na.rm
```

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 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"

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

Source

J.W. Smith., el al. 1988. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261–265). IEEE Computer Society Press.

mlbench, R package. F. Leisch & E. Dimitriadou, 2021. mlbench: Machine Learning Benchmark Problems `https://CRAN.R-project.org/package=mlbench`

Examples

```
library(spinifex)
str(BreastCancer_na.rm)
dat  <- scale_sd(BreastCancer_na.rm[, 1:8])
clas <- BreastCancer_na.rm$Class

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

ggt <- ggtour(mt, dat, angle = .2) +
  proto_default(aes_args = list(color = clas, shape = clas))

animate_plotly(ggt)
```

`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.

See Also

Other manual tour adjacent functions: `interpolate_manual_tour()`, `manip_var_of()`, `manual_tour()`, `rotate_manip_space()`

Examples

```
library(spinifex)
dat <- scale_sd(wine[, 2:6])
bas <- basis_pca(dat)
mv  <- manip_var_of(bas)
create_manip_space(basis = bas, manip_var = mv)

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

`devMessage` *Development message*

Description

Send a message if the 4th chunk of the package version is 9000.

Usage

```
devMessage(text)
```

Arguments

`text`	A character string to message() if package version is _9000.

`draw_basis` *Draw a basis on a static ggplot*

Description

Additively draws a basis on a static ggplot. Not a `geom` or `proto`. Expects

Usage

```
draw_basis(
  basis,
  map_to = data.frame(x = c(0, 1), y = c(0, 1)),
  position = c("left", "center", "right", "bottomleft", "topright", "off"),
  manip_col = "blue",
  line_size = 0.6,
  text_size = 4,
  basis_label = abbreviate(gsub("[^[:alnum:]=]", "", rownames(basis), 3L))
)
```

Arguments

`basis`	A (p*d) basis to draw. Draws the first two components. If facet is used cbind the facet variable to a specific facet level (2nd example), otherwise the basis prints on all facet levels.
`map_to`	A data.frame to scale the basis to. Defaults to a unitbox; data.frame(x = c(0,1), y = c(0,1)).
`position`	The position, to place the basis axes relative to the centered data. `_basis` Expects one of c("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 0.6.
`text_size`	Size of the text label of the variables. Defaults to 4.
`basis_label`	The text labels of the data variables. Defaults to the 3 character abbreviation of the rownames of the basis.

Examples

```
library(spinifex)
library(ggplot2)
dat  <- scale_sd(penguins_na.rm[, 1:4])
clas <- penguins_na.rm$species
bas  <- basis_pca(dat)
proj <- as.data.frame(dat %*% bas)

ggplot() +
  geom_point(aes(PC1, PC2), proj) +
  draw_basis(bas, proj, "left") +
  coord_fixed()

## Aesthetics and basis on specific facet levels
proj <- cbind(proj, clas = penguins_na.rm$species)
bas <- cbind(as.data.frame(bas), clas = levels(clas)[2])
ggplot() +
  facet_wrap(vars(clas)) +
  geom_point(aes(PC1, PC2, color = clas, shape = clas), proj) +
  draw_basis(bas, proj, "left") +
  theme_spinifex()
## To repeat basis in all facet levels don't cbind a facet variable.
```

`facet_wrap_tour` *Wrap a 1d ribbon of panels into 2d for animation*

Description

Create and wrap a 1d ribbon of panels in 2d. Because of the side effects of `ggtour` and `facet_wrap_tour` this wants to be applied after `ggtour` and before any `proto_*` functions. `plotly` may not display well with with faceting.

Usage

```
facet_wrap_tour(facet_var, nrow = NULL, ncol = NULL, dir = "h")
```

Arguments

`facet_var`	Expects a single variable to facet the levels of. Should be a vector, not a formula (`~cyl`) or `ggplot2::vars()` call.
`nrow`	Number of rows. Defaults to NULL; set by display dim.
`ncol`	Number of columns. Defaults to NULL; set by display dim.
`dir`	Direction of wrapping: either "h" horizontal by rows, or "v", for vertical by columns. Defaults to "h".

See Also

Other ggtour proto functions: `append_fixed_y()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_density2d()`, `proto_hex()`, `proto_highlight()`, `proto_hline0()`, `proto_origin()`, `proto_point()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat     <- scale_sd(penguins_na.rm[, 1:4])
clas    <- penguins_na.rm$species
bas     <- basis_pca(dat)
mv      <- manip_var_of(bas)
mt_path <- manual_tour(bas, manip_var = mv)

## d = 2 case
message("facet_wrap_tour wants be called early, so that other proto's adopt the facet_var.")
ggt <- ggtour(mt_path, dat, angle = .3) +
  facet_wrap_tour(facet_var = clas, ncol = 2, nrow = 2) +
  proto_default(aes_args = list(color = clas, shape = clas),
                identity_args = list(size = 1.5))
## Not run:
animate_gganimate(ggt) ## May not always play well with plotly

## End(Not run)
```

`filmstrip` *Create a "filmstrip" of the frames of a ggtour.*

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. Is very demanding on the plots pane, works better with ggsave().

Usage

```
filmstrip(ggtour, ...)
```

Arguments

`ggtour`	A grammar of graphics tour with appended protos added. A return from `ggtour() + proto_*()`
`...`	optionally pass arguments to ggplot2::facet_wrap, such as nrow = 3, ncol = 2, scales = "free".

See Also

Other ggtour animator: `animate_gganimate()`, `animate_plotly()`

Examples

```
library(spinifex)
dat  <- scale_sd(penguins_na.rm[, 1:4])
clas <- penguins_na.rm$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 = .3) +
  proto_point(list(color = clas, shape = clas),
              list(size = 1.5)) +
  proto_basis()
filmstrip(ggt)

## d = 1 case & specify facet dim
bas1d     <- basis_pca(dat, d = 1)
mt_path1d <- manual_tour(basis = bas1d, manip_var = mv)
ggt1d <- ggtour(mt_path1d, dat, angle = 99) +
  proto_default1d(aes_args = list(fill = clas, color = clas))
filmstrip(ggt1d, nrow = 12, ncol = 3)
```

`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,
  basis_label = NULL,
  data_label = NULL,
  do_center_frame = TRUE
)
```

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 (radians) for interpolation frames between frames of the `basis_array`. Defaults to .05. To opt out of interpolation set to NA or 0.
`basis_label`	`plotly` tooltip labels for the basis. Defaults to NULL; 3 character abbreviation of the rownames of basis.
`data_label`	`plotly` tooltip labels for the data. Defaults to the NULL, rownames and/or numbers of data.
`do_center_frame`	Whether or not to center the mean within each animation frame. Defaults to TRUE.

See Also

Other ggtour proto functions: `append_fixed_y()`, `facet_wrap_tour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_density2d()`, `proto_hex()`, `proto_highlight()`, `proto_hline0()`, `proto_origin()`, `proto_point()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat     <- scale_sd(penguins_na.rm[, 1:4])
clas    <- penguins_na.rm$species
bas     <- basis_pca(dat)
mv      <- manip_var_of(bas)
mt_path <- manual_tour(bas, manip_var = mv)

## d = 2 case
ggt <- ggtour(basis_array = mt_path, data = dat, angle = .3) +
  proto_default(aes_args = list(color = clas, shape = clas),
                identity_args = list(size = 1.5, alpha = .8))

animate_plotly(ggt)


## Finer control calling individual proto_* functions
ggt <- ggtour(basis_array = mt_path, data = dat, angle = .3) +
  proto_point(aes_args = list(color = clas, shape = clas),
              identity_args = list(size = 1.5, alpha = .8),
              row_index = which(clas == levels(clas)[1])) +
  proto_basis(position = "right",
              manip_col = "red",
              text_size = 7L) +
  proto_origin()

animate_plotly(ggt)


## d = 1 case
bas1d <- basis_pca(dat, d = 1)
mt_path1d <- manual_tour(basis = bas1d, manip_var = mv)

ggt1d <- ggtour(basis_array = mt_path1d, data = dat, angle = .3) +
  proto_default1d(aes_args = list(fill= clas, color = clas))

animate_plotly(ggt1d)


## Single basis
ggt <- ggtour(basis_array = bas, data = dat) +
  proto_default(aes_args = list(fill= clas, color = clas))
## ggtour() returns a static ggplot2 plot

ggt
### or as html widget with tooltips
animate_plotly(ggt)
```

`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`.

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 manual tour adjacent functions: `create_manip_space()`, `manip_var_of()`, `manual_tour()`, `rotate_manip_space()`

Examples

```
## This function is not meant for external use
dat  <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas  <- basis_pca(dat)
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_any_layer_class` *Check ggplot layers for use of a specific geom*

Description

Checks if any of the layers of a ggplot contain a specific class.

Usage

```
is_any_layer_class(ggplot, class_nm = "GeomDensity")
```

Arguments

`ggplot`	Check the layers of this ggplot object
`class_nm`	The class name to check, note this differs slightly from the name of the geom function. Defaults to "GeomDensity", checking to see if geom_density was used in any of the layers.

See Also

`ggplot2::theme` for all theme options.

Examples

```
library(ggplot2)
library(spinifex)

g <- ggplot(mtcars, aes(disp, color = factor(cyl))) +
  geom_density() + geom_histogram()
is_any_layer_class(g, "GeomDensity")
is_any_layer_class(g, "GeomPoint")
```

`is_orthonormal` *Orthonormality of a matrix*

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 of the element-wise distance of t(x) %*% x from the identity matrix.

Value

Single logical, whether or not the matrix is orthonormal.

Examples

```
spinifex::is_orthonormal(tourr::basis_random(n = 6))
spinifex::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.

See Also

Other manual tour adjacent functions: `create_manip_space()`, `interpolate_manual_tour()`, `manual_tour()`, `rotate_manip_space()`

Examples

```
dat <- scale_sd(wine[, 2:6])
bas <- basis_pca(dat)

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

```
manual_tour(
  basis,
  manip_var,
  theta = NULL,
  phi_min = 0,
  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.

See Also

Other manual tour adjacent functions: `create_manip_space()`, `interpolate_manual_tour()`, `manip_var_of()`, `rotate_manip_space()`

Examples

```
library(spinifex)
dat  <- scale_sd(wine[, 2:6])
clas <- wine$Type
bas  <- basis_pca(dat)
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)
```

```
## Animating with ggtour() & proto_* (d = 2 case)
mt <- manual_tour(basis = bas, manip_var = mv)
ggt <- ggtour(mt, dat, angle = .2) +
  proto_origin() +
  proto_point(list(color = clas, shape = clas)) +
  proto_basis()

animate_plotly(ggt)


## d = 1 case
## basis could be 1- or 2D; protos_* only use 1st column
mv  <- manip_var_of(bas)
mt  <- manual_tour(basis = bas, manip_var = mv)
ggt <- ggtour(mt, dat, angle = .3) +
  proto_density(aes_args = list(color = clas, fill = clas)) +
  proto_basis1d() +
  proto_origin1d()

animate_plotly(ggt)


## Bring your own basis
bas <- matrix(rnorm(2 * ncol(dat)), ncol = 2)
bas <- orthonormalise(bas) ## manual_tour warns if basis isn't orthonormal
mt  <- manual_tour(basis = bas, manip_var = 1)
ggt <- ggtour(mt, dat, angle = .2) +
  proto_default(aes_args = list(color = clas, shape = clas))

animate_plotly(ggt)
```

`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

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

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 functions: `map_relative()`

Examples

```
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", "right", "bottomleft", "topright", "off", "top1d",
   "floor1d", "bottom1d", "full", "facetleft", "facetright", "facettop", "facetbottom"),
  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: c("center", "left", "right", "bottomleft", "topright", "off", "full", "top1d", "floor1d", "bottom1d", "full", "facetleft", "facetright", "facettop", "facetbottom").
`to`	Data.frame to scale to. Based on the min/max of the first 2 columns. If left NULL defaults to data.frame(x = c(0, 1), y = c(0, 1).

Value

Transformed values of `x`, dimension and class unchanged.

See Also

`map_absolute` for more manual control.

Other linear mapping functions: `map_absolute()`

Examples

```
## !!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_na.rm` *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_na.rm
```

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:

```
require(palmerpenguins)
d <- palmerpenguins::penguins
d <- d[complete.cases(d), ] ## Remove missing, 2 obs of numeric and several in sex
d <- d[, c(3:6, 1, 7, 2)]   ## Numeric to front, group factors, remove year
penguins_na.rm <- as.data.frame(d) ## Remove {tibble} dependency
## save(penguins_na.rm, file = "./data/penguins_na.rm.rda")
```

Source

palmerpenguins R package. A. Horst, 2020. Palmer Archipelago (Antarctica) Penguin Data. https://CRAN.R-project.org/package=palmerpenguins

Adelie penguins: Palmer Station Antarctica LTER and K. Gorman. 2020. Structural size measurements and isotopic signatures of foraging among adult male and female Adelie penguins (Pygoscelis adeliae) nesting along the Palmer Archipelago near Palmer Station, 2007-2009 ver 5. Environmental Data Initiative doi:10.6073/pasta/98b16d7d563f265cb52372c8ca99e60f

Gentoo penguins: Palmer Station Antarctica LTER and K. Gorman. 2020. Structural size measurements and isotopic signatures of foraging among adult male and female Gentoo penguin (Pygoscelis papua) nesting along the Palmer Archipelago near Palmer Station, 2007-2009 ver 5. Environmental Data Initiative doi:10.6073/pasta/7fca67fb28d56ee2ffa3d9370ebda689

Chinstrap penguins: Palmer Station Antarctica LTER and K. Gorman. 2020. Structural size measurements and isotopic signatures of foraging among adult male and female Chinstrap penguin (Pygoscelis antarcticus) nesting along the Palmer Archipelago near Palmer Station, 2007-2009 ver 6. Environmental Data Initiative doi:10.6073/pasta/c14dfcfada8ea13a17536e73eb6fbe9e

Originally published in: Gorman KB, Williams TD, Fraser WR (2014) Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus Pygoscelis). PLoS ONE 9(3): e90081. doi:10.1371/journal.pone.0090081

Examples

```
library(spinifex)
str(penguins_na.rm)
dat   <- scale_sd(penguins_na.rm[, 1:4])
clas1 <- penguins_na.rm$species
clas2 <- penguins_na.rm$sex

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

ggt <- ggtour(mt, dat, angle = .2) +
  proto_default(aes_args = list(color = clas1, shape = clas2))

animate_plotly(ggt)
```

`PimaIndiansDiabetes_long`

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, squared))
- 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:

```
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")
```

Source

J.W. Smith., el al. 1988. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261–265). IEEE Computer Society Press.

mlbench, R package. F. Leisch & E. Dimitriadou, 2021. mlbench: Machine Learning Benchmark Problems https://CRAN.R-project.org/package=mlbench

Examples

```
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)
```

```
mt  <- manual_tour(bas, mv)

ggt <- ggtour(mt, dat, angle = .2) +
  proto_default(aes_args = list(color = clas, shape = clas))

animate_plotly(ggt)
```

`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, squared))
- 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:

```
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(aes_args = list(color = clas, shape = clas))

animate_plotly(ggt)
```

`play_manual_tour` *Animate a manual tour. superseded*

Description

[Superseded], 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

```
play_manual_tour(
  basis = NULL,
  data,
  manip_var,
  theta = NULL,
  phi_min = 0,
  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

```
library(spinifex)
message("It's suggested to switch to the proto api, see `?ggtour` to get started.")

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

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

suppressWarnings(
  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(FALSE){ ## Don't accidentally save file
  ## 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

```
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

```
library(spinifex)
message("It's suggested to switch to the proto api, see `?ggtour` to get started.")

dat_std <- scale_sd(wine[, 2:6])
clas    <- wine$Type
bas     <- basis_pca(dat_std)
gt_path <- save_history(dat_std, tour_path = tourr::grand_tour(), max = 5)

## Not run:
suppressWarnings(
  play_tour_path(tour_path = gt_path, data = dat_std)
)

suppressWarnings(
  play_tour_path(tour_path = gt_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(FALSE){ ## Don't accidentally save file
  ## Export plotly .html widget
  play_tour_path(tour_path = gt_path, data = dat_std,
                 render_type = render_plotly,
                 html_filename = "myRadialTour.html")

  ## Export gganimate .gif
  play_tour_path(tour_path = gt_path, data = dat_std,
                 render_type = render_gganimate,
                 gif_path = "myOutput", gif_filename = "myRadialTour.gif")
}

## End(Not run)
```

`plot_pca` *Plot 2 components of Principal Component Analysis*

Description

Performs PCA on the data and used `proto_default` to plot with percent variation labels.

Usage

```
plot_pca(data, components = c(1, 2), ...)
```

Arguments

`data`	Numeric matrix or data.frame of the observations.
`components`	The 2 numbers of the principal components to use.
`...`	Optionally pass arguments to `proto_default`

See Also

`proto_default()`

Examples

```
dat <- scale_sd(wine[, 2:6])
plot_pca(data = dat)

## Different components, class coloring
clas <- as.factor(wine$Type)
plot_pca(data = dat, components = c(1, 3), position = "center",
         aes_args = list(color = clas, shape = clas))
```

`plot_pca_scree` *Plot 2 components of Principal Component Analysis*

Description

Performs PCA on the data and used `proto_default` to plot with percent variation labels. Also appends a screeplot of the component variances.

Usage

```
plot_pca_scree(data, components = c(1, 2), ...)
```

Arguments

`data`	Numeric matrix or data.frame of the observations.
`components`	The 2 numbers of the principal components to use.
`...`	Optionally pass arguments to `proto_default`

Examples

```
dat <- scale_sd(wine[, 2:6])
plot_pca_scree(data = dat)

## Different components, class coloring
clas <- as.factor(wine$Type)
plot_pca_scree(data = dat, components = c(1, 3), position = "center",
               aes_args = list(color = clas, shape = clas))
```

`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

```
proto_basis(
  position = c("left", "center", "right", "bottomleft", "topright", "full", "off"),
  manip_col = "blue",
  line_size = 0.6,
  text_size = 4
)

proto_basis1d(
  position = c("bottom1d", "floor1d", "top1d", "full", "off"),
  manip_col = "blue",
  segment_size = 2,
  text_size = 4,
  text_offset = -1.15
)
```

Arguments

`position`	The position, to place the basis axes relative to the data. `proto_basis` expects one of c("left", "center", "right", "bottomleft", "topright", "off"), defaults to "left". `proto_basis1d` expects one of c("bottom1d", "floor1d", "top1d", "off"). Defaults to "bottom1d".
`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 .6.
`text_size`	Size of the text label of the variables. Defaults to 4.
`segment_size`	(1D bases only) the width thickness of the rectangle bar showing variable magnitude on the axes. Defaults to 2.
`text_offset`	The horizontal offset of the text labels relative to the variable contributions in the basis between (-1, 1). Defaults to -1.15.

See Also

Other ggtour proto functions: `append_fixed_y()`, `facet_wrap_tour()`, `ggtour()`, `proto_default()`, `proto_density()`, `proto_density2d()`, `proto_hex()`, `proto_highlight()`, `proto_hline0()`, `proto_origin()`, `proto_point()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat  <- scale_sd(penguins_na.rm[, 1:4])
clas <- penguins_na.rm$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 = .3) +
  proto_point() +
  proto_basis()

animate_plotly(ggt)


## Customize basis
ggt2 <- ggtour(mt_path, dat) +
  proto_basis(position = "right", manip_col = "green",
              line_size = .8, text_size = 8)

animate_plotly(ggt2)


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

ggt1d <- ggtour(mt_path1d, dat, angle = .3) +
  proto_density() +
  proto_basis1d()

animate_plotly(ggt1d)


## Customized basis1d
ggt1d <- ggtour(mt_path1d, dat, angle = .3) +
  proto_density() +
  proto_basis1d(position     = "bottom",
                manip_col    = "pink",
                segment_size = 3,
                text_size    = 6,
                text_offset  = 1.2)

animate_plotly(ggt1d)
```

`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

```
proto_default(
  position = c("left", "center", "right", "bottomleft", "topright", "off"),
  ...
)

proto_default1d(position = c("bottom1d", "floor1d", "top1d", "off"), ...)
```

Arguments

`position`	The position, to place the basis axes relative to the data. `proto_basis` expects one of c("left", "center", "right", "bottomleft", "topright", "off"), defaults to "left". `proto_basis1d` expects one of c("bottom1d", "floor1d", "top1d", "off"). Defaults to "bottom1d".
`...`	Optionally pass additional arguments to `proto_point` or `proto_density`.

See Also

Other ggtour proto functions: `append_fixed_y()`, `facet_wrap_tour()`, `ggtour()`, `proto_basis()`, `proto_density()`, `proto_density2d()`, `proto_hex()`, `proto_highlight()`, `proto_hline0()`, `proto_origin()`, `proto_point()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat  <- scale_sd(penguins_na.rm[, 1:4])
clas <- penguins_na.rm$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(aes_args = list(color = clas, shape = clas))

animate_plotly(ggt)

library(spinifex)

## 1D case:
gt_path <- save_history(dat, grand_tour(d = 1), max_bases = 3)

ggt <- ggtour(gt_path, dat) +
  proto_default1d(aes_args = list(fill = clas, color = clas))
```

```
animate_plotly(ggt)
```

`proto_density` *Tour proto for data, 1D density, with rug marks*

Description

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

Usage

```
proto_density(
  aes_args = list(),
  identity_args = list(alpha = 0.7),
  row_index = NULL,
  density_position = c("identity", "stack", "fill"),
  rug_shape = c(3, 142, 124, NULL)
)
```

Arguments

`aes_args`	A list of arguments to call inside of aes(). aesthetic mapping of the primary geom. For example, `geom_point(aes(color = my_fct, shape = my_fct))` becomes `aes_args = list(color = my_fct, shape = my_fct)`.
`identity_args`	A list of static, identity arguments passed into the primary geom. For instance, `geom_point(size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`. Also passes more foundational arguments such as stat and position, though these been tested less.
`row_index`	A numeric or logical index of rows to subset to. Defaults to NULL, all observations.
`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.
`rug_shape`	Numeric, the number of the shape to make rug marks. Expects either 3 142, 124 or NULL, ’+’, ’|’ (plotly), ’|’ (ggplot2) respectively. Defaults to 3.

See Also

Other ggtour proto functions: `append_fixed_y()`, `facet_wrap_tour()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density2d()`, `proto_hex()`, `proto_highlight()`, `proto_hline0()`, `proto_origin()`, `proto_point()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat  <- scale_sd(penguins_na.rm[, 1:4])
clas <- penguins_na.rm$species

## Manual tour
bas <- basis_olda(dat, clas)
mt  <- manual_tour(bas, manip_var = 2)
ggt <- ggtour(mt, dat, angle = .3) +
  proto_density(aes_args = list(color = clas, fill = clas)) +
  proto_basis1d() +
  proto_origin1d()

animate_plotly(ggt)


## Grand tour
gt_path <- save_history(dat, grand_tour(), max = 3)
ggt <- ggtour(gt_path, dat, angle = .3) +
  proto_density(aes_args = list(color = clas, fill = clas)) +
  proto_basis1d() +
  proto_origin1d()

animate_plotly(ggt)
```

`proto_density2d` *Tour proto for data, 1D density, with rug marks*

Description

Adds `geom_density_2d()` of the projected data.

Usage

```
proto_density2d(
  aes_args = list(),
  identity_args = list(bins = 4),
  row_index = NULL
)
```

Arguments

`aes_args`	A list of arguments to call inside of aes(). aesthetic mapping of the primary geom. For example, `geom_point(aes(color = my_fct, shape = my_fct))` becomes `aes_args = list(color = my_fct, shape = my_fct)`.

`identity_args`	A list of static, identity arguments passed into the primary geom. For instance, `geom_point(size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`. Also passes more foundational arguments such as stat and position, though these have been tested less.
`row_index`	A numeric or logical index of rows to subset to. Defaults to NULL, all observations.

See Also

Other ggtour proto functions: `append_fixed_y()`, `facet_wrap_tour()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_hex()`, `proto_highlight()`, `proto_hline0()`, `proto_origin()`, `proto_point()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat     <- scale_sd(penguins_na.rm[, 1:4])
clas    <- penguins_na.rm$species
gt_path <- save_history(dat, grand_tour(), max = 3)

## geom_density_2d args can be passed in identity_args (bins, binwidth, breaks)
ggt <- ggtour(gt_path, dat, angle = .3) +
  proto_density2d(aes_args = list(color = clas, fill = clas),
                  identity_args = list(binwidth = .3)) +
  proto_point(aes_args = list(color = clas, shape = clas),
              identity_args = list(alpha = .2)) +
  proto_basis()

animate_plotly(ggt)
```

`proto_frame_cor2` *Tour proto for frames square correlation*

Description

Adds text to the animation, the frame and its specified correlation.

Usage

```
proto_frame_cor2(
  text_size = 4,
  row_index = NULL,
  xy_position = c(0.7, -0.1),
  ...
)
```

Arguments

`text_size`	Size of the text. defaults to 4.
`row_index`	A numeric or logical index of rows to subset to. Defaults to NULL, all observations.
`xy_position`	Vector of the x and y position, the fraction of the range of the data in each direction. The projection data is contained in (0, 1) in each direction. Defaults to c(.7, -.1), in the bottom right.
`...`	Optionally, pass additional arguments to `stats::cor`, specifying the type of within frame correlation.

See Also

`stats::cor`

Examples

```
library(spinifex)
dat     <- scale_sd(penguins_na.rm[, 1:4])
clas    <- penguins_na.rm$species
gt_path <- save_history(dat, grand_tour(), max_bases = 5)

ggt <- ggtour(gt_path, dat, angle = .3) +
  proto_default(aes_args = list(color = clas, shape = clas)) +
  proto_frame_cor2(xy_position = c(.5, 1.1))

animate_plotly(ggt)
```

`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

```
proto_hex(
  aes_args = list(),
  identity_args = list(),
  row_index = NULL,
  bins = 30
)
```

Arguments

`aes_args`	A list of arguments to call inside of aes(). aesthetic mapping of the primary geom. For example, `geom_point(aes(color = my_fct, shape = my_fct))` becomes `aes_args = list(color = my_fct, shape = my_fct)`.
`identity_args`	A list of static, identity arguments passed into the primary geom. For instance, `geom_point(size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`. Also passes more foundational arguments such as stat and position, though these have been tested less.
`row_index`	A numeric or logical index of rows to subset to. Defaults to NULL, all observations.
`bins`	Numeric vector giving number of bins in both vertical and horizontal directions. Defaults to 30.

See Also

Other ggtour proto functions: `append_fixed_y()`, `facet_wrap_tour()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_density2d()`, `proto_highlight()`, `proto_hline0()`, `proto_origin()`, `proto_point()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
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.
## Increase performance by aggregating many points into few hexagons
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 highlighing specified points*

Description

A `geom_point` or `geom_segment`(1d case) call to draw attention to a subset of points. This is mostly redundant `proto_point` with the implementation of the `row_index` argument on data protos, still helpful in the 1d case and for `mark_initial`, does not use bkg_row_color

Usage

```
proto_highlight(
  aes_args = list(),
  identity_args = list(color = "red", size = 5, shape = 8),
  row_index = 1,
  mark_initial = FALSE
)

proto_highlight1d(
  aes_args = list(),
  identity_args = list(color = "red", linetype = 2, alpha = 0.9),
  row_index = 1,
  mark_initial = FALSE
)
```

Arguments

`aes_args`	A list of arguments to call inside of aes(). aesthetic mapping of the primary geom. For example, `geom_point(aes(color = my_fct, shape = my_fct))` becomes `aes_args = list(color = my_fct, shape = my_fct)`.
`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)`. #' Typically a single numeric for point size, alpha, or similar.
`row_index`	A numeric or logical index of rows to subset to. Defaults to 1, highlighting the first row.
`mark_initial`	Logical, whether or not to leave a fainter mark at the subset's initial position. Defaults to FALSE.

See Also

Other ggtour proto functions: `append_fixed_y()`, `facet_wrap_tour()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_density2d()`, `proto_hex()`, `proto_hline0()`, `proto_origin()`, `proto_point()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat     <- scale_sd(penguins_na.rm[, 1:4])
clas    <- penguins_na.rm$species
gt_path <- save_history(dat, grand_tour(), max_bases = 5)

## d = 2 case
ggt <- ggtour(gt_path, dat, angle = .3) +
  proto_default(aes_args = list(color = clas, shape = clas)) +
  proto_highlight(row_index = 5)

animate_plotly(ggt)
```

```
## Highlight multiple observations
ggt2 <- ggtour(gt_path, dat, angle = .3) +
  proto_default(aes_args = list(color = clas, shape = clas)) +
  proto_highlight(row_index = c( 2, 6, 19),
                  identity_args = list(color = "blue", size = 4, shape = 4))

animate_plotly(ggt2)

## 1D case:
gt_path1d <- save_history(dat, grand_tour(d = 1), max_bases = 3)

ggt <- ggtour(gt_path1d, dat, angle = .3) +
  proto_default1d(aes_args = list(fill = clas, color = clas)) +
  proto_highlight1d(row_index = 7)

animate_plotly(ggt)


## Highlight multiple observations, mark_initial defaults to off
ggt2 <- ggtour(gt_path1d, dat, angle = .3) +
  proto_default1d(aes_args = list(fill = clas, color = clas)) +
  proto_highlight1d(row_index = c(2, 6, 7),
                    identity_args = list(color = "green", linetype = 1))

animate_plotly(ggt2)
```

`proto_hline0` *Tour proto adding a vertical/horizontal line*

Description

Adds a vertical/horizontal line with an intercept of 0, scaled to the data frame.

Usage

```
proto_hline0(identity_args = list(color = "grey80", size = 0.5, alpha = 0.9))

proto_vline0(identity_args = list(color = "grey80", size = 0.5, alpha = 0.9))
```

Arguments

`identity_args`	A list of static, identity arguments passed into the primary geom. For instance, `geom_point(size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`. Also passes more foundational arguments such as stat and position, though these have been tested less.

See Also

Other ggtour proto functions: `append_fixed_y()`, `facet_wrap_tour()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_density2d()`, `proto_hex()`, `proto_highlight()`, `proto_origin()`, `proto_point()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat  <- scale_sd(penguins_na.rm[, 1:4])
clas <- penguins_na.rm$species

## 2D case:
gt_path <- save_history(dat, grand_tour(), max_bases = 5)
ggt <- ggtour(gt_path, dat, angle = .1) +
  proto_point(list(color = clas, shape = clas)) +
  proto_hline0() + ## horizonatal line at 0
  proto_vline0()   ## vertical line at 0

animate_plotly(ggt)
```

`proto_origin` *Tour proto for data origin zero mark*

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 the primary geom. For instance, `geom_point(size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`. Also passes more foundational arguments such as stat and position, though these have been tested less.
`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 functions: `append_fixed_y()`, `facet_wrap_tour()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_density2d()`, `proto_hex()`, `proto_highlight()`, `proto_hline0()`, `proto_point()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat  <- scale_sd(penguins_na.rm[, 1:4])
clas <- penguins_na.rm$species

## 2D case:
gt_path <- save_history(dat, grand_tour(), max_bases = 5)
ggt <- ggtour(gt_path, dat, angle = .1) +
  proto_point(list(color = clas, shape = clas)) +
  proto_origin() ## `+` in center

animate_plotly(ggt)


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

ggt <- ggtour(gt_path1d, dat) +
  proto_density(list(fill = clas, color = clas)) +
  proto_origin1d() ## Adds line at 0.

animate_plotly(ggt)
```

`proto_point` *Tour proto for data point*

Description

Adds `geom_point()` of the projected data.

Usage

```
proto_point(
  aes_args = list(),
  identity_args = list(alpha = 0.9),
  row_index = NULL,
  bkg_color = "grey80"
)
```

Arguments

`aes_args`	A list of arguments to call inside of aes(). aesthetic mapping of the primary geom. For example, `geom_point(aes(color = my_fct, shape = my_fct))` becomes `aes_args = list(color = my_fct, shape = my_fct)`.
`identity_args`	A list of static, identity arguments passed into the primary geom. For instance, `geom_point(size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`. Also passes more foundational arguments such as stat and position, though these have been tested less.
`row_index`	A numeric or logical index of rows to subset to. Defaults to NULL, all observations.
`bkg_color`	The character color by name or hexadecimal to display background observations, those not in the `row_index`. Defaults to "grey80". Use FALSE or NULL to skip rendering background points. Other aesthetic values such as shape and alpha are set adopted from `aes_args` and `identity_args`.

See Also

Other ggtour proto functions: `append_fixed_y()`, `facet_wrap_tour()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_density2d()`, `proto_hex()`, `proto_highlight()`, `proto_hline0()`, `proto_origin()`, `proto_text()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat     <- scale_sd(penguins_na.rm[, 1:4])
clas    <- penguins_na.rm$species
gt_path <- save_history(dat, grand_tour(), max_bases = 5)

ggt <- ggtour(gt_path, dat, angle = .3) +
  proto_point(aes_args = list(color = clas, shape = clas),
              identity_args = list(size = 2, alpha = .7))

animate_plotly(ggt)


## Select/highlight observations with `row_index`
ggt <- ggtour(gt_path, dat, angle = .3) +
  proto_point(aes_args = list(color = clas, shape = clas),
              identity_args = list(size = 2, alpha = .7),
              row_index = which(clas == levels(clas)[1]),
              bkg_color = "grey80") ## FALSE or NULL to skip plotting background

animate_plotly(ggt)
```

`proto_text` *Tour proto for data, text labels*

Description

Adds `geom_text()` of the projected data.

Usage

```
proto_text(
  aes_args = list(vjust = "outward", hjust = "outward"),
  identity_args = list(nudge_x = 0.05),
  row_index = NULL
)
```

Arguments

`aes_args`	A list of arguments to call inside of aes(). aesthetic mapping of the primary geom. For example, `geom_point(aes(color = my_fct, shape = my_fct))` becomes `aes_args = list(color = my_fct, shape = my_fct)`.
`identity_args`	A list of static, identity arguments passed into the primary geom. For instance, `geom_point(size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`. Also passes more foundational arguments such as stat and position, though these have been tested less.
`row_index`	A numeric or logical index of rows to subset to. Defaults to NULL, all observations.

See Also

Other ggtour proto functions: `append_fixed_y()`, `facet_wrap_tour()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_density2d()`, `proto_hex()`, `proto_highlight()`, `proto_hline0()`, `proto_origin()`, `proto_point()`, `proto_text_repel()`

Examples

```
library(spinifex)
dat     <- scale_sd(penguins_na.rm[, 1:4])
clas    <- penguins_na.rm$species
bas     <- basis_pca(dat)
mv      <- manip_var_of(bas)
gt_path <- save_history(dat, grand_tour(), max_bases = 5)

ggt <- ggtour(gt_path, dat, angle = .2) +
  proto_text(list(color = clas))

animate_plotly(ggt)
```

```
## Custom labels, subset of points
dat     <- mtcars[c("mpg", "disp", "hp", "drat", "wt")]
clas    <- as.factor(mtcars$cyl)
bas     <- basis_olda(dat, clas)
lab     <- abbreviate(rownames(mtcars))
gt_path <- save_history(dat, grand_tour(), max_bases = 5)

ggt2 <- ggtour(gt_path, dat) +
  proto_text(list(color = clas, label = lab),
             list(alpha = .7),
             row_index = 1:15)

animate_plotly(ggt2)
```

`proto_text_repel` *Tour proto for data, text labels that repel*

Description

Adds `ggrepel::geom_text_repel()` of the projected data.

Usage

```
proto_text_repel(
  aes_args = list(vjust = "outward", hjust = "outward"),
  identity_args = list(nudge_x = 0.05),
  row_index = NULL
)
```

Arguments

`aes_args`	A list of arguments to call inside of aes(). aesthetic mapping of the primary geom. For example, `geom_point(aes(color = my_fct, shape = my_fct))` becomes `aes_args = list(color = my_fct, shape = my_fct)`.
`identity_args`	A list of static, identity arguments passed into the primary geom. For instance, `geom_point(size = 2, alpha = .7)` becomes `identity_args = list(size = 2, alpha = .7)`. Also passes more foundational arguments such as stat and position, though these have been tested less.
`row_index`	A numeric or logical index of rows to subset to. Defaults to NULL, all observations.

See Also

Other ggtour proto functions: `append_fixed_y()`, `facet_wrap_tour()`, `ggtour()`, `proto_basis()`, `proto_default()`, `proto_density()`, `proto_density2d()`, `proto_hex()`, `proto_highlight()`, `proto_hline0()`, `proto_origin()`, `proto_point()`, `proto_text()`

Examples

```
library(spinifex)
dat     <- scale_sd(penguins_na.rm[, 1:4])
clas    <- penguins_na.rm$species
bas     <- basis_pca(dat)
mv      <- manip_var_of(bas)
gt_path <- save_history(dat, grand_tour(), max_bases = 5)

ggt <- ggtour(gt_path, dat, angle = .2) +
  proto_text_repel(list(color = clas)) +
  proto_point(list(color = clas, shape = clas),
              list(size = 1))

animate_gganimate(ggt)


## Custom labels, subset of points
dat     <- mtcars[c("mpg", "disp", "hp", "drat", "wt")]
clas    <- as.factor(mtcars$cyl)
bas     <- basis_olda(dat, clas)
lab     <- abbreviate(rownames(mtcars))
gt_path <- save_history(dat, grand_tour(), max_bases = 3)

ggt2 <- ggtour(gt_path, dat) +
  proto_text_repel(list(color = clas, label = lab),
                   list(size = 1),
                   row_index = 1:30) +
  proto_point(list(color = clas, shape = clas),
              list(size = 1),
              row_index = 1:30)

animate_gganimate(ggt2)
```

`render_` *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

```
render_(
  frames,
  axes = "center",
  manip_col = "blue",
  line_size = 0.6,
```

```
    text_size = 4,
    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 .6.
`text_size`	The size of the text labels of the variable contributions of the basis. Defaults to 4.
`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

```
library(spinifex)
message("It's suggested to switch to the proto api, see `?ggtour` to get started.")

## 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

```
render_gganimate(
  fps = 8,
  rewind = FALSE,
  start_pause = 0.5,
  end_pause = 1,
  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

```
library(spinifex)
message("It's suggested to switch to the proto api, see `?ggtour` to get started.")

## Setup
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(FALSE) ## 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

```
render_plotly(fps = 8, 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

```
library(spinifex)
message("It's suggested to switch to the proto api, see `?ggtour` to get started.")

## 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)


render_plotly(frames = mt_df_ls)

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),
  ggproto = list(theme_bw(), scale_color_brewer(palette = "Set2")))

## Saving a .gif, may require additional setup
if(FALSE) ## Don't accidentally save file
  render_plotly(frames = mt_df_ls, axes = "bottomleft", fps = 10,
                html_filename = "myRadialTour.html")
```

`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.

See Also

Other manual tour adjacent functions: `create_manip_space()`, `interpolate_manual_tour()`, `manip_var_of()`, `manual_tour()`

Examples

```
library(spinifex)
dat <- scale_sd(wine[, 2:6])
bas <- basis_pca(dat)
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, 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

```
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

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

`save_history` *Save a tour basis array.*

Description

Save a tour path so it can later be displayed in many different ways. A wrapper function can mute the noisy text side effects of `tourr::save_history`. Changes a few argument defaults differ: doesn't scale data columns to (0, 1), max_bases = 10, appends the start basis if tour_path is grand, it isn't already there, and has correct dim.

Usage

```
save_history(
  data,
  tour_path = tourr::grand_tour(),
  max_bases = 10,
  start = NULL,
  rescale = FALSE,
  sphere = FALSE,
  step_size = Inf,
  verbose = getOption("verbose"),
  ...
)
```

Arguments

`data`	Matrix, or data frame containing complete numeric columns
`tour_path`	Tour path generator. Defaults to `tourr::grand_tour`.
`max_bases`	The maximum number of new bases to generate. Some tour paths (like the guided tour) may generate less than the maximum. Defaults to 10.
`start`	Optional basis to start at.
`rescale`	Whether or not to rescale all variables to range (0,1). Defaults to FALSE.
`sphere`	Whether or not to sphere (whiten) covariance matrix to the identity matrix. Defaults to FALSE.
`step_size`	Distance (in radians) between target frames (not interpolated frames). Defaults to Inf, forcing new basis generation at each step.
`verbose`	Whether or not to suppress the text output side effects from `tourr::save_history`. Defaults to FALSE.
`...`	Additional arguments passed to `tourr::new_tour`.

See Also

`tourr::save_history` `tourr::new_tour` `tourr::grand_tour`

Examples

```
library(spinifex)
dat <- scale_sd(penguins_na.rm[, 1:4])
## A grand tour path
gt_path <- save_history(data = dat, tour_path = grand_tour(), max_bases = 10)
dim(gt_path)

## A 1d grand tour path
gt1d_path <- save_history(dat, grand_tour(d = 1), 10)
dim(gt1d_path)

## A holes guided tour path
holes_path <- save_history(dat, guided_tour(holes(), max.tries = 10))
```

```
dim(holes_path)

## These are basis_arrays to be used in ?spinifex::ggtour()
```

`scale_colour_discrete` *Set default color & fill for discrete variables*

Description

Masks ggplot2's default color/fill color palette for discrete variables.

Usage

```
scale_colour_discrete(...)

scale_fill_discrete(...)
```

Arguments

`...`	Passes arguments to ggplot2::scale_colour/fill_brewer.

`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` *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 basis, perhaps via a guided tour optimizing the projection for some objective function.

Details

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

Author(s)

Maintainer: Nicholas Spyrison `<spyrison@gmail.com>` (ORCID)

Authors:

- Dianne Cook (ORCID) [thesis advisor]

See Also

`manual_tour()` `ggtour()` `proto_default()`

`theme_spinifex` *Theme spinifex*

Description

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

Usage

```
theme_spinifex(...)
```

Arguments

`...`	Optionally pass arguments to `ggplot2::theme()`.

See Also

`ggplot2::theme` for all theme options.

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

[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

```
view_frame(
  basis = NULL,
  data = NULL,
  manip_var = NULL,
  theta = 0,
  phi = 0,
  basis_label = abbreviate(row.names(basis), 3),
  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;

Value

A ggplot object of the rotated projection.

See Also

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

Examples

```
library(spinifex)
message("It's suggested to switch to the proto api, see `?ggtour` to get started.")

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

## Minimal example

suppressWarnings(
  view_frame(basis = bas)
)

## Typical example
suppressWarnings(
  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)
suppressWarnings(
  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))
)
```

`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

```
view_manip_space(
  basis,
  manip_var,
  tilt = 0.1 * pi,
  basis_label = abbreviate(row.names(basis), 3),
  manip_col = "blue",
  manip_sp_col = "red",
  line_size = 0.6,
  text_size = 4
)
```

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 abbriviation of the orginal 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.

Value

ggplot object of the basis.

Examples

```
library(spinifex)

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")
```

`weather` *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_na.rm
```

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 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.

Copyright Commonwealth of Australia 2010, Bureau of Meteorology. Definitions adapted from http://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml

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:

```
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_na.rm <- d
## save(weather_na.rm, file = "./data/weather_na.rm.rda")
```

Source

Bureau of Meteorology, Commonwealth of Australia `https://reg.bom.gov.au/climate/data-services/`

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(weather_na.rm)
dat  <- scale_sd(weather_na.rm[, 2:18])
clas <- weather_na.rm$RainTomorrow

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

ggt <- ggtour(mt, dat, angle = .2) +
  proto_default(aes_args = list(color = clas, shape = clas))

animate_plotly(ggt)
```

`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

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:

```
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`

PARVUS. M. Forina. et al. 1988. Elsevier, Amsterdam, PARVUS: An extendable package of programs for data exploration, classification and correlation. ISBN 0-44-430121z

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(aes_args = list(color = clas, shape = clas))

animate_plotly(ggt)
```

Index