Package ‘spinifex’

April 9, 2019

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

Version  0.1.0

Description  Generates the path for manual tours
The grand tour is an algorithm that shows all possible projections given sufficient time.
Guided uses projection pursuit to steer the tour towards interesting projections.
The 'spinifex' package implements manual control, where the contribution of a selected variable can be adjusted between -1 to 1, to examine the sensitivity of structure in the data to that variable.
The result is an animation where the variable is toured into and out of the projection completely, which can be rendered using the ‘gganimate’ and ‘plotly’ packages.

Depends  R (>= 3.4.0), tourr
License  GPL-3
Encoding  UTF-8
LazyData  true
RoxygenNote  6.1.1

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

BugReports  https://github.com/nspyrison/spinifex/issues
Imports  dplyr, GGally, ggplot2, plotly, tibble, webshot
Suggests  gganimate, gifski, knitr, png, rmarkdown, RColorBrewer
VignetteBuilder  knitr
NeedsCompilation  no

Author  Nicholas Spyrison [aut, cre],
        Dianne Cook [aut, ths]

Maintainer  Nicholas Spyrison <spyrison@gmail.com>

Repository  CRAN

Date/Publication  2019-04-09 12:00:04 UTC
array2df

**Description**

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

**Usage**

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

**Arguments**

- `array` A (p, d, n_slides) array of a tour, the output of `manual_tour()`.
- `data` Optional, (n, p) dataset to project, consisting of numeric variables.

**Value**

A list containing the (p, d, n_slides) basis slides array, and the (n, d, n_slides) data slides array.
Examples

```r
flea_std <- tourr::rescale(tourr::flea[, 1:6])
rb <- basis_random(n = ncol(flea_std))
mtour <- manual_tour(basis = rb, manip_var = 4)
array2df(array = mtour, data = flea_std)
```

---

**breastcancer**

**Wisconsin Breast Cancer Database**

**Description**

Formatted subset of mlbench::BreastCancer. See mlbench for original data more context.

**Usage**

`breastcancer`

**Format**

Data frame (tibble) with 675 observations on 10 variables: a factor Id, 9 numeric variables, and target class.

**Details**

The objective is to identify each of a number of benign or malignant classes. Samples arrive periodically as Dr. Wolberg reports his clinical cases. The database therefore reflects this chronological grouping of the data. This grouping information appears immediately below, having been removed from the data itself. Each variable except for the first was converted into 11 primitive numerical attributes with values ranging from 0 through 10. There are 16 missing attribute values.

Data frame (tibble) with 675 observations on 10 variables: a factor Id, 9 numeric variables, and target class:

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

Reproducing this dataset:
library("mlbench")

d <- mlbench::BreastCancer
d <- d[!duplicated(d), ]
d <- d[complete.cases(d), ]
mat <- as.matrix(d[, 2:9])
mat <- apply(mat, 2, as.numeric)
breastbancer <- dplyr::as_tibble(data.frame(Id = d$Id, mat, Class = d$Class))

Examples

str(breastcancer)
## Not run:
play_manual_tour(data = breastcancer[, 2:9], manip_var = 3, init_rescale_data = TRUE)

## End(Not run)

---

col_of

Return col values for a given categorical variable

description

Returns string col values for a passed categorical variable.

Usage

`col_of(cat, pallet_name = "Dark2")`

Arguments

cat The categorical variable to return the col values for.
pallet_name The name of the RColorBrewer pallet to get the colors from. Defaults to "Dark2"

Value

The integer col values for a passed categorical variable.

Examples

`col_of(tourr::flea$species)`
create_manip_space

Create a manipulation space

Description

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

Usage

create_manip_space(basis, manip_var)

Arguments

- **basis**: A (p, d) orthonormal matrix.
- **manip_var**: Number of the column/dimension to rotate.

Value

A (p, d+1) orthonormal matrix, the manipulation space.

Examples

```r
flea_std <- tourr::rescale(tourr::flea[,1:6])
rb <- basis_random(n = ncol(flea_std))
create_manip_space(basis = rb, manip_var = 4)
```

---

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 manual tour of the manip_var, which is rotated from phi’s starting position to phi_max, to phi_min, and back to the start position.

Usage

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

- **basis**: A (p, d) dim orthonormal matrix. Required, no default.
- **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 manual 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 bases.

Value

A (p, d, 4) history_array of the manual tour. The bases set for phi_start, phi_min, phi_max, and back to phi_start. To be called by `tourr::interpolate()`.

Examples

```r
flea_std <- tourr::rescale(tourr::flea[,1:6])
rb <- basis_random(n = ncol(flea_std))
manual_tour(basis = rb, manip_var = 4)
```

---

**pch_of**

*Return pch values for a given categorical variable*

Description

Returns integer pch values for a passed categorical variable.

Usage

`pch_of(cat)`

Arguments

- **cat**: The categorical variable to return the pch values for.

Value

The integer pch values for a passed categorical variable.

Examples

```r
pch_of(tourr::flea$species)
```
play_manual_tour

---

**Description**

Performs the sepicify manual tour and returns an animation of render_type. For use with tourr::save_history(). tour paths see play_tour_path(). A wrapper function for manual_tour(), array2df(), and render_.

**Usage**

```r
play_manual_tour(data, basis = NULL, manip_var, theta = NULL, phi_min = 0, phi_max = 0.5 * pi, angle = 0.05, manip_col = "blue", render_type = render_plotly, col = "black", pch = 20, axes = "center", fps = 3, init_rescale_data = FALSE, ...)
```

**Arguments**

- `data` (n, p) dataset to project, consisting of numeric variables.
- `basis` A (p, d) dim orthonormal numeric matrix. If it’s left null, random basis will be used.
- `manip_var` Number of the column/dimension to rotate.
- `theta` Angle in radians of "in-plane" rotation, on the XY plane of the reference frame. If left NULL, will initialize the radial angle of the manip_var.
- `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 bases.
- `manip_col` String of the color to highlight the manip_var.
- `render_type` Which graphics to render to. Defaults to render_plotly, alternative use render_gganimate.
- `col` Color of the projected points. Defaults to "black".
- `pch` Point character of the projected points. Defaults to 20.
- `axes` Position of the axes: "center", "bottomleft" or "off". Defaults to "center".
- `fps` Frames/slides shown per second. Defaults to 3.
- `init_rescale_data` When TRUE will apply tourr::rescale() on the data. Defaults to FALSE.
- `...` Optionally pass additional arguments to the render_type for plotting options.

**Value**

An animation of a manual tour.
Examples

```r
## Not run:
flea_std <- tourr::rescale(tourr::flea[,1:6])
rbb <- tourr::basis_random(n = ncol(flea_std))

play_manual_tour(data = flea_std, basis = rb, manip_var = 4)

play_manual_tour(data = flea_std, basis = rb, manip_var = 6,
                 render_type = render_gganimate, col = col_of(flea$species), axes = "bottomleft")

## End(Not run)
```

---

**play_tour_path**  
*Render display of a provided tour path*

Description

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

Usage

```r
play_tour_path(tour_path, data = NULL, angle = 0.05,
               render_type = render_plotly, col = "black", pch = 20,
               axes = "center", fps = 3, ...)
```

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 bases.
- `render_type`: Which graphics to render to. Defaults to `render_plotly`, alternative use `render_gganimate`.
- `col`: Color of the projected points. Defaults to "black".
- `pch`: Point character of the projected points. Defaults to 20.
- `axes`: Position of the axes: "center", "bottomleft" or "off". Defaults to "center".
- `fps`: Frames/slides shown per second. Defaults to 3.
- `...`: Optionally pass additional arguments to `render_type`. 
Examples

```r
## Not run:
flea_std <- rescale(tourr::flea[,1:6])
tpath <- save_history(flea_std, max = 3)

play_tour_path(tour_path = tpath, data = flea_std, angle = .15)
play_tour_path(tour_path = tpath, data = flea_std, angle = .15, fps = 4,
  render_type = render_gganimate, col = col_of(flea$species), axes = "bottomleft")

## End(Not run)
```

---

### render_.

Render the ggplot before the animation package

---

Description

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

Usage

```r
render_(slides, manip_col = "blue", col = "black", pch = 20,
  axes = "center")
```

Arguments

- `slides` The result of `array2df()`, a long df of the projected frames.
- `manip_col` String of the color to highlight the `manip_var`. Defaults to "blue".
- `col` Color of the projected points. Defaults to "black".
- `pch` Point character of the projected points. Defaults to 20.
- `axes` Position of the axes: "center", "bottomleft" or "off". Defaults to "center".

Value

A ggplot2 object ready to be called by `render_plotly()` or `render_gganimate()`.
render_gganimate

Render the slides as a gganimate animation

Description

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

Usage

```r
render_gganimate(slides, manip_col = "blue", col = "black", pch = 20,
                   axes = "center", fps = 3, ...)
```

Arguments

- `slides`: The result of `array2df()`, a long df of the projected frames.
- `manip_col`: String of the color to highlight the `manip_var`. Defaults to "blue".
- `col`: Color of the projected points. Defaults to "black".
- `pch`: Point character of the projected points. Defaults to 20.
- `axes`: Position of the axes: "center", "bottomleft" or "off". Defaults to "center".
- `fps`: Frames/slides shown per second. Defaults to 3.
- `...`: Optional, pass addition arguments to `gganimate::transition_states()`.

Examples

```r
## Not run:
flea_std <- tourr::rescale(tourr::flea[, 1:6])

rb <- basis_random(n = ncol(flea_std))
mtour <- manual_tour(basis = rb, manip_var = 4)
sshow <- array2df(array = mtour, data = flea_std)
render_gganimate(slides = sshow)

render_gganimate(slides = sshow, col = col_of(flea$species),
                  axes = "bottomleft", fps = 2)

## End(Not run)
```
render_plotly

render_plotly  Render the slides as a plotly animation

Description

Takes the result of array2df() and renders them into a plotly animation.

Usage

render_plotly(slides, manip_col = "blue", col = "black", pch = 20, 
axes = "center", fps = 3, ...)

Arguments

slides  The result of array2df(), a long df of the projected frames.
manip_col  String of the color to highlight the manip_var. Defaults to "blue".
col  Color of the projected points. Defaults to "black".
pch  Point character of the projected points. Defaults to 20.
axes  Position of the axes: "center", "bottomleft" or "off". Defaults to "center".
fps  Frames/slides shown per second. Defaults to 3.
...  Optional, pass addition arguments to plotly::animation_opts() and plotly::layout().

Examples

```r
## Not run:
flea_std <- tourr::rescale(tourr::flea[, 1:6])

rb <- basis_random(n = ncol(flea_std))
mtour <- manual_tour(basis = rb, manip_var = 4)
sshow <- array2df(array = mtour, data = flea_std)
render_plotly(slides = sshow)

render_plotly(slides = sshow, col = col_of(flea$species),
               axes = "bottomleft", fps = 2)
## End(Not run)
```
rotate_manip_space  
*Rotate and return the manipulation space*

**Description**

Typically called by `manual_tour()`. Rotates a \((p, d+1)\) manipulation space matrix by \((d+1, d+1)\) rotation matrix, returning \((p, d+1)\) matrix rotation space. The first 2 variables of which are the linear combination of the variables for a 2d projection.

**Usage**

```r
rotate_manip_space(manip_space, theta, phi)
```

**Arguments**

- `manip_space`: A \((p, d+1)\) dim manipulation space to be rotated.
- `theta`: Angle in radians of rotation "in-plane", on the XY plane of the reference frame. Typically set from `manip_type` in `proj_data()`.
- `phi`: Angle in radians of rotation "out-of-plane", the z axis of the reference frame. Effectively changes the norm of XY contributions of the `manip_var`.

**Value**

A \((p, d+1)\) orthonormal matrix of the rotated (manipulation) space.

**Examples**

```r
flea_std <- tourr::rescale(tourr::flea[,1:6])

rb <- basis_random(n = ncol(flea_std))
msp <- create_manip_space(basis = rb, manip_var = 4)
rotate_manip_space(msp, theta = runif(1, max = 2 * pi), phi = runif(1, max = 2 * pi))
```

---

set_axes_position  
*Returns the axis scale and position*

**Description**

Typically called, by other functions to scale axes.

**Usage**

```r
set_axes_position(x, axes)
```
spinifex

Arguments

x numeric data object to scale and offset
axes Position of the axes: "center", "bottomleft" or "off". Defaults to "center".

Value

axis_scale and axis_scale

Examples

rb <- basis_random(4, 2)
set_axes_position(x = rb, axes = "bottomleft")

Description

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

Details

Its main functions are:

- `play_manual_tour()`, performs a manual tour, returning a plotly animate by default.
- `play_tour_path()`, turns a tour path into an animation, returning a plotly object by default.
- `view_basis()`, plot a basis set on a reference axis.
- `view_manip_space()`, plot a manipulation space highlighting the manip var.

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

Author(s)

Maintainer: Nicholas Spyrison <spyrison@gmail.com>
Authors:

- Dianne Cook [thesis advisor]

See Also
tourr (package)
view_basis

Plot projection frame and return the axes table.

Description

Uses base graphics to plot the circle with axes representing the projection frame. Returns the corresponding table.

Usage

```r
view_basis(basis, labels = paste0("V", 1:nrow(basis)), data = NULL, axes = "center")
```

Arguments

- `basis`: A (p, d) basis, XY linear combination of each dimension (numeric variable).
- `labels`: Optional character vector of p length, add name to the axes in the reference frame, typically the variable names.
- `data`: Optional (n, p) data, plots xy scatterplot on the frame.
- `axes`: Position of the axes: "center", "bottomleft" or "off". Defaults to "center".

Value

ggplot object of the basis.

Examples

```r
rb <- basis_random(4, 2)
view_basis(basis = rb)

flea_std <- tourr::rescale(tourr::flea[, 1:4])
view_basis(basis = rb, data = flea_std)
view_basis(basis = rb, data = flea_std, axes = "bottomleft")
```

view_manip_space

Plot projection frame and return the axes table.

Description

Uses base graphics to plot the circle with axes representing the projection frame. Returns the corresponding table.

Usage

```r
view_manip_space(basis, manip_var, manip_col = "blue", theta = pi * 5/12, z_col = "red", labels = paste0("V", 1:nrow(basis)))
```
**Arguments**

- **basis**: A (p, d) basis, XY linear combination of each dimension (numeric variable).
- **manip_var**: Number of the column/dimension to rotate.
- **manip_col**: String of the color to highlight the `manip_var`.
- **theta**: Angle in radians to rotate the manip space. Defaults to $\pi * 5/12$.
- **z_col**: Color to illustrate the z direction or out of the projection plane.
- **labels**: Optional, character vector of `p` length, add name to the axes in the reference frame, typically the variable names.

**Value**

- `ggplot` object of the basis.

**Examples**

```r
flea_std <- tourr::rescale(tourr::flea[, 1:6])
rb <- basis_random(ncol(flea_std), 2)

view_manip_space(basis = rb, manip_var = 4)
```

---

**Description**

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

**Usage**

- `weather`

**Format**

Data frame (tibble) of 366 observations of 20 variables, one year of daily observations of weather variables at Canberra airport in Australia starting November 2007.

**Details**

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

The data has been processed to provide a target variable `RainTomorrow` (whether there is rain on the following day - No/Yes) and a risk variable 'RISK_MM' (how much rain recorded in millimeters). Various transformations were performed on the source data. The dataset is quite small and is useful only for repeatable demonstration of various data science operations.
The source dataset is Copyright by the Australian Commonwealth Bureau of Meteorology and is provided as part of the rattle package with permission.

Data frame (tibble) of 366 observations of 20 variables, one year of daily observations of weather variables at Canberra airport in Australia starting November 2007:

- Date, The date of observation (a Date object).
- 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 so-called Class A pan evaporation (mm) in the 24 hours to 9am.
- Sunshine, The number of hours of bright sunshine in the day.
- WindGustSpeed, The speed (km/h) of the strongest wind gust in the 24 hours to midnight.
- WindSpeed9am, Wind speed (km/hr) averaged over 10 minutes prior to 9am.
- WindSpeed3pm, Wind speed (km/hr) averaged over 10 minutes prior to 3pm.
- Humid9am, Relative humidity (percent) at 9am.
- Humid3pm, Relative humidity (percent) at 3pm.
- Pressure9am, Atmospheric pressure (hpa) reduced to mean sea level at 9am.
- Pressure3pm, Atmospheric pressure (hpa) reduced to mean sea level at 3pm.
- Cloud9am, Fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eighths. It records how many eighths of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.
- Cloud3pm, Fraction of sky obscured by cloud (in "oktas": eighths) at 3pm. See Cloud9am for a description of the values.
- Temp9am, Temperature (degrees C) at 9am.
- Temp3pm, Temperature (degrees C) at 3pm.
- RainToday, Integer: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0.
- RISK_MM, The amount of rain. A kind of measure of the "risk".
- RainTomorrow, The target variable. Did it rain tomorrow?

Reproducing this dataset:

```r
library("rattle.data")
weather <- dplyr::as.tibble(weather[, c(1,3:7,9,12:24)])
```

**Source**


**References**

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 categoric variable.

Usage

wine

Format

data frame (tibble) of 178 observations of 13 variables, target class Type and 12 numeric variables.

Details

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

Data frame (tibble) of 178 observations of 13 variables, target class Type and 12 numeric variables:

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

Reproducing this dataset:

library("rattle.data")
wine <- dplyr::as.tibble(wine)

Examples

str(wine)
## Not run:
play_manual_tour(data = wine[, 2:14], manip_var = 1, init_rescale_data = TRUE)

## End(Not run)
Index

*Topic datasets
  breastcancer, 3
  weather, 15
  wine, 17

array2df, 2
breastcancer, 3
col_of, 4
create_manip_space, 5
manual_tour, 5
pch_of, 6
play_manual_tour, 7
play_manual_tour(), 13
play_tour_path, 8
play_tour_path(), 13
render_, 9
render_gganimate, 10
render_plotly, 11
rotate_manip_space, 12
set_axes_position, 12
spinifex, 13
spinifex-package (spinifex), 13
view_basis, 14
view_basis(), 13
view_manip_space, 14
view_manip_space(), 13
weather, 15
wine, 17