Package 'lionfish'

March 13, 2025

Title Interactive 'tourr' Using 'python'

Version 1.0.27

Description Extends the functionality of the 'tourr' package by an interactive
graphical user interface. The interactivity allows users to
effortlessly refine their 'tourr' results by manual intervention,
which allows for integration of expert knowledge and aids the
interpretation of results. For more information on 'tourr' see
Wickham et. al (2011) <doi:10.18637/jss.v040.i02> or
<https://github.com/ggobi/tourr>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Imports reticulate, tourr (>= 1.2.4), data.table, utils

Suggests ggplot2, viridis, randomForest, knitr, rmarkdown, markdown,
colorspace, ggbeeswarm, patchwork, mvtnorm, tidyverse, dplyr,
gridExtra, flexclust

VignetteBuilder knitr

URL https://mmedl94.github.io/lionfish/

Config/Needs/website knitr

Depends R (>= 3.5.0)

LazyData true

NeedsCompilation no

Author Matthias Medl [aut, cre] (<https://orcid.org/0000-0002-3354-4579>)

Maintainer Matthias Medl <matthias.medl@chello.at>

Repository CRAN

Date/Publication 2025-03-13 20:50:02 UTC

Contents

---

ausActiv   *Australian Vacation Activities Dataset*

Description

Australian Vacation Activities Dataset

Usage

```
ausActiv
```

Format

A 1003 x 44 array of binary responses

Details

The Australian Vacation Activities dataset includes responses from 1,003 adult Australians who were surveyed on their vacation activities through a permission-based internet panel in 2007.The responses are coded in binary with 0 indicating that the tourist didn't partake in the activity and 1 indicating they did.

Source

https://statistik.boku.ac.at/nachlass_leisch/MSA/

Examples

```
data(ausActiv)
head(ausActiv)
```

---

check_conda_env *Check Whether 'anaconda' Environment Exists*

Description

Checks if 'anaconda' environment of the given name is installed and returns TRUE if so.

Usage

```
check_conda_env(env_name = "r-lionfish")
```

Arguments

env_name   a string that defines the name of the 'anaconda' environment reticulate uses.

Value

boolean

Examples

```
check_conda_env(env_name="r-lionfish")
```

---

check_venv *Check Whether Virtual 'python' Environment Exists*

Description

Checks whether virtual 'python' environment of a given name exists an returns TRUE if it does.

Usage

```
check_venv(env_name = "r-lionfish")
```

Arguments

env_name   a string that defines the name of the 'python' environment reticulate uses.

Value

boolean

Examples

```
check_venv(env_name="r-lionfish")
```

ChemicalManufacturingProcess

Chemical Manufacturing Process Dataset

Description

Chemical Manufacturing Process Dataset

Usage

ChemicalManufacturingProcess

Format

A 176 x 58 array of continuous variables

Details

This data set contains information about a chemical manufacturing process, in which the goal is to understand the relationship between the process and the resulting final product yield and can be found in the in the AppliedPredictiveModeling R package. The data has been copied from http://appliedpredictivemodeling.com/data in agreement with their license.

Source

http://appliedpredictivemodeling.com/data

Examples

```
data(ChemicalManufacturingProcess)
head(ChemicalManufacturingProcess)
```

---

get_guided_holes_better_history

Get Guided Tour-Holes-Better History

Description

Returns a guided tour with the holes index and the 'search_better' argument to the python' backend. This guided tour is generated with the 'tourr' functions 'save_history' and 'guided_tour'.

Usage

```
get_guided_holes_better_history(data, dimension)
```

Arguments

data	the dataset to calculate the projections with.
dimension	1 for a 1d tour or 2 for a 2d tour

Value

history object containing the projections of the requested tour

Examples

```
data("flea", package = "tourr")
flea <- flea[-7]
get_guided_holes_better_history(flea, 2)
```

```
get_guided_holes_history
```
Get Guided Tour-Holes History

Description

Returns a guided tour with the holes index to the python' backend. This guided tour is generated with the 'tourr' functions 'save_history' and 'guided_tour'.

Usage

```
get_guided_holes_history(data, dimension)
```

Arguments

data	the dataset to calculate the projections with.
dimension	1 for a 1d tour or 2 for a 2d tour

Value

history object containing the projections of the requested tour

Examples

```
data("flea", package = "tourr")
flea <- flea[-7]
get_guided_holes_history(flea, 2)
```

---

get_guided_lda_history

Get Guided Tour-LDA History

Description

Returns a guided tour with the LDA index to the python' backend. This guided tour is generated with the 'tourr' functions 'save_history' and 'guided_tour'.

Usage

```
get_guided_lda_history(data, clusters, dimension)
```

Arguments

data	the dataset to calculate the projections with
clusters	the clusters for the lda to be performed on
dimension	1 for a 1d tour or 2 for a 2d tour

Value

history object containing the projections of the requested tour

Examples

```
data("flea", package = "tourr")
clusters <- as.numeric(factor(flea[[7]]))
get_guided_lda_history(flea[-7], clusters, 2)
```

---

get_local_history   *Get Local Tour History*

Description

Returns a local tour based on currently displayed projection(s) to the 'python' backend. This local tour is generated with the 'tourr' functions 'save_history' and 'local_tour'.

Usage

```
get_local_history(data, starting_projection)
```

Arguments

data	the dataset to calculate the projections with. In practice only the two first rows of the dataset are provided as the actual data is not needed.
starting_projection	
	the initial projection one wants to initiate the local tour from

Value

history object containing the projections of the requested tour

Examples

```
library(tourr)
data("flea", package = "tourr")
flea <- flea[-7]
prj <- tourr::basis_random(ncol(flea), 2)
get_local_history(flea, prj)
```

---

init_env *Initialize Environment for 'python' Backend*

Description

Initializes the 'python' backend required for the functionality of lionfish. At first it searches whether a 'python' environment with the provided name exists or not. If it does, it will be loaded and the 'python' function 'check_backend' is run to check if it works properly. If no 'python' environment with the provided name exists, it will be installed and then loaded. This can either be done with or without Anaconda as package manager. Anaconda can be more robust, but the GUI will appear dated. Thus, trying 'init_env' with virtual_env = "virtual_env" out first is recommended. For 'Windows' users, the path to the tk and tcl libraries will be set, otherwise tkinter cannot run.

Usage

```
init_env(env_name = "r-lionfish", virtual_env = "virtual_env", local = FALSE)
```

Arguments

env_name	a string that defines the name of the python environment reticulate uses. This can be useful if one wants to use a preinstalled python environment.
virtual_env	either "virtual_env" or "anaconda". "virtual_env" creates a virtual environment, which has the advantage that the GUI looks much nicer and no previous python installation is required,but the setup of the environment can be more error prone. "anaconda" installs the python environment via Anaconda, which can be more stable, but the GUI looks more dated.
local	logical

Value

initializes python environment

Examples

```
if (check_venv()){
init_env(env_name = "r-lionfish", virtual_env = "virtual_env")
} else if (check_conda_env()){
init_env(env_name = "r-lionfish", virtual_env = "anaconda")
}
```

---

interactive_tour   *R Wrapper for 'interactive_tour' Function Written in 'python'*

Description

Launches the lionfish GUI and at minimum requires the data do be loaded and the plot_objects. The other parameters are optional. For technical reasons the parameters half_range, n_plot_cols, n_subsets, color_scale, label_size and display_size cannot be adjusted from within the GUI. The GUI has to be closed and relaunched (possibly with load_interactive_tour) if you want to change them. Please visit https://mmedl94.github.io/lionfish/index.html for a detailed description of the GUI and its features.

Usage

```
interactive_tour(
  data,
  plot_objects,
  feature_names = NULL,
  half_range = NULL,
  n_plot_cols = 2,
  preselection = FALSE,
  preselection_names = FALSE,
  n_subsets = 3,
  display_size = 5,
  hover_cutoff = 10,
  label_size = 15,
  color_scale = "default",
  color_scale_heatmap = "default",
  axes_blendout_threshhold = 1
)
```

Arguments

data	the dataset you want to investigate
plot_objects	a named list of objects you want to be displayed. Each entry requires a definition of the type of display and a specification of what should be plotted.

feature_names   names of the features of the dataset

half_range   factor that influences the scaling of the displayed tour plots. Small values lead
to more spread out datapoints (that might not fit the plotting area), while large
values lead to the data being more compact. If not provided a good estimate will
be calculated and used.

n_plot_cols   specifies the number of columns of the grid of the final display.

preselection   a vector that specifies in which subset each datapoint should be put initially.

preselection_names
a vector that specifies the names of the preselection subsets

n_subsets   the total number of available subsets.

display_size   rough size of each subplot in inches

hover_cutoff   number of features at which the switch from intransparent to transparent labels
that can be hovered over to make them intransparent occurs

label_size   size of the labels of the feature names of 1d and 2d tours

color_scale   a viridis/matplotlib colormap to define the color scheme of the subgroups

color_scale_heatmap
a viridis/matplotlib colormap to define the color scheme of the heatmap

axes_blendout_threshhold
initial value of the threshold for blending out projection axes with a smaller
length

Value

opens the interactive GUI

Examples

```
library(tourr)
data("flea", package = "tourr")
data <- flea[1:6]
clusters <- as.numeric(flea$species)
flea_subspecies <- unique(flea$species)
feature_names <- colnames(data)

guided_tour_history <- tourr::save_history(data,
  tour_path = tourr::guided_tour(holes())
)
grand_tour_history_1d <- tourr::save_history(data,
  tour_path = tourr::grand_tour(d = 1)
)

half_range <- max(sqrt(rowSums(data^2)))

obj1 <- list(type = "2d_tour", obj = guided_tour_history)
obj2 <- list(type = "1d_tour", obj = grand_tour_history_1d)
obj3 <- list(type = "scatter", obj = c("tars1", "tars2"))
obj4 <- list(type = "hist", obj = "head")
```

```
if (check_venv()){
init_env(env_name = "r-lionfish", virtual_env = "virtual_env")
} else if (check_conda_env()){
init_env(env_name = "r-lionfish", virtual_env = "anaconda")
}

if (interactive()){
interactive_tour(
  data = data,
  plot_objects = list(obj1, obj2, obj3, obj4),
  feature_names = feature_names,
  half_range = half_range,
  n_plot_cols = 2,
  preselection = clusters,
  preselection_names = flea_subspecies,
  n_subsets = 5,
  display_size = 5
)
}
```

---

load_interactive_tour   *R Wrapper for 'load_interactive_tour' Function Written in 'python'*

Description

Loads a previously saved snapshot created by pressing the "Save projections and subsets" within the GUI. The data that was loaded when saving a snapshot has to be provided to this function when loading that snapshot. Additionally, this function allows to adjust some parameters of the GUI when loading a snapshot, such as display_size or label_size.

Usage

```
load_interactive_tour(
  data,
  directory_to_save,
  half_range = NULL,
  n_plot_cols = 2,
  preselection = FALSE,
  preselection_names = FALSE,
  n_subsets = FALSE,
  display_size = 5,
  hover_cutoff = 10,
  label_size = 15,
  color_scale = "default",
  color_scale_heatmap = "default",
  axes_blendout_threshhold = 1
)
```

Arguments

`data`	the dataset you want to investigate. Must be the same as the dataset that was loaded when the save was created!
`directory_to_save`	
	path to the location of the save folder
`half_range`	factor that influences the scaling of the displayed tour plots. Small values lead to more spread out datapoints (that might not fit the plotting area), while large values lead to the data being more compact. If not provided a good estimate will be calculated and used.
`n_plot_cols`	specifies the number of columns of the grid of the final display.
`preselection`	a vector that specifies in which subset each datapoint should be put initially.
`preselection_names`	
	a vector that specifies the names of the preselection subsets
`n_subsets`	the total number of available subsets (up to 10).
`display_size`	rough size of each subplot in inches
`hover_cutoff`	number of features at which the switch from intransparent to transparent labels that can be hovered over to make them intransparent occurs
`label_size`	size of the labels of the feature names of 1d and 2d tours
`color_scale`	a viridis/matplotlib colormap to define the color scheme of the subgroups
`color_scale_heatmap`	
	a viridis/matplotlib colormap to define the color scheme of the heatmap
`axes_blendout_threshhold`	
	initial value of the threshold for blending out projection axes with a smaller length

Value

opens the interactive GUI

Examples

```
data("flea", package = "tourr")
data <- flea[1:6]

if (check_venv()){
init_env(env_name = "r-lionfish", virtual_env = "virtual_env")
} else if (check_conda_env()){
init_env(env_name = "r-lionfish", virtual_env = "anaconda")
}

pytourr_dir <- find.package("lionfish", lib.loc = NULL, quiet = TRUE)
pytourr_dir <- paste(pytourr_dir, "/inst/test_snapshot", sep = "")
if (interactive()){
load_interactive_tour(data, pytourr_dir)
}
```

---

render_proj_inter	*Modification of the 'render_proj' Function of 'tourr'*

Description

Modification of the render_proj() function of tourr so that the half_range is calculated with max(sqrt(rowSums(data^2))) or can be provided as argument.

Usage

```
render_proj_inter(
  data,
  prj,
  half_range = NULL,
  axis_labels = NULL,
  obs_labels = NULL,
  limits = 1,
  position = "center"
)
```

Arguments

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

Value

list containing projected data, circle and segments for axes

Examples

```
library(tourr)
data("flea", package = "tourr")
flea_std <- apply(flea[,1:6], 2, function(x) (x-mean(x))/sd(x))
prj <- tourr::basis_random(ncol(flea[,1:6]), 2)
p <- render_proj_inter(flea_std, prj)
```

---

risk *Risk Dataset*

Description

Risk Dataset

Usage

```
risk
```

Format

A 563 x 6 array of Likert scale data

Details

Adult Australian residents that have undertaken at least one holiday in the last year, which involved staying away from home for at least four nights, were asked what risks they have taken in the past. The questions were about recreational, health, career, financial, safety and social risk. The response options were on the following: never (1), rarely (2), quite often (3), often (4) or very often (5)

Source

https://statistik.boku.ac.at/nachlass_leisch/MSA/

Examples

```
data(risk)
head(risk)
```

---

winterActiv *Austrian Vacation Activities Dataset*

Description

Austrian Vacation Activities Dataset

Usage

```
winterActiv
```

Format

A 2961 x 27 array of binary responses

Details

The Austrian Vacation Activities dataset comprises responses from 2,961 adult tourists who spent their holiday in Austria during the 1997/98 season. The responses are coded in binary with 0 indicating that the tourist didn't partake in the activity and 1 indicating they did.

Source

https://statistik.boku.ac.at/nachlass_leisch/MSA/

Examples

```
data(winterActiv)
head(winterActiv)
```

Index