Package 'sillysplines'

July 24, 2026

Title Linear Splines to Generate Decision Boundaries

Version 1.0.1

Description Create two dimensional datasets with decision boundaries set by linear splines. An HTML widget enables users to draw the splines on a web page and generate a JSON file that can be used to generate datasets.

License MIT + file LICENSE

Encoding UTF-8

Imports htmlwidgets, jsonlite, withr

Suggests rlang, shiny, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://janithwanni.github.io/sillysplines/, https://github.com/janithwanni/sillysplines

Config/roxygen2/version 8.0.0

BugReports https://github.com/janithwanni/sillysplines/issues

NeedsCompilation no

Author Janith Wanniarachchi [aut, cre, cph]

Maintainer Janith Wanniarachchi <janithcwanni@gmail.com>

Repository CRAN

Date/Publication 2026-07-24 10:10:01 UTC

Contents

`add_noise_vars` *Add noise variables to create high-dimensional data*

Description

This function generates samples from a uniform distribution to be used as noise variables. This can be used to assess the effect of high-dimensions on a model fit, and explainers.

Usage

```
add_noise_vars(data, n_vars = 2)
```

Arguments

`data`	A matrix or data frame with three columns (x, y, class).
`n_vars`	Integer; number of additional variables to generate.

Value

A data frame with three+n_vars columns, like:

x1 Random uniform coordinate

x2 Random uniform coordinate

x3 From the original data x

x4 From the original data y

class Binary class label ("Above" or "Below")

Examples

```
coords <- matrix(c(0.2, 0.3,
                   0.4, 0.5,
                   0.6, 0.7), ncol = 2, byrow = TRUE)

df <- create_data(coord = coords, n_samples = 500, seed = 717)

df <- add_noise_vars(df, 2)
head(df)
```

`classify_boundary` *Classify Points Relative to a Boundary Line*

Description

Classifies data points as being above or below a boundary defined by a set of coordinates. The function uses piecewise linear interpolation to determine the boundary's y-value at each x-coordinate in the data, then classifies points based on whether they lie above or below this boundary.

Usage

```
classify_boundary(data, boundary_coord, classes = c("Above", "Below"))
```

Arguments

`data`	A data frame containing at least two columns named 'x' and 'y' representing the coordinates of points to be classified.
`boundary_coord`	A two-column matrix or data frame where the first column contains x-coordinates and the second column contains y-coordinates defining the boundary line. Points should be ordered by x-coordinate for proper interpolation.
`classes`	A character vector of length 2 specifying the class labels. Default is `c("Above", "Below")`. The first element is assigned to points above the boundary, the second to points below.

Details

The function uses `stats::approxfun()` with `rule = 2` for piecewise linear interpolation. This means:

- Between boundary points, linear interpolation is used
- For x-values outside the range of boundary coordinates, the boundary is extrapolated linearly using the slope at the nearest endpoint

Points exactly on the boundary (y == boundary_y) are classified as "Below" (the second class).

Value

A character vector of the same length as the number of rows in `data`, containing class labels for each point.

See Also

`approxfun` for details on interpolation

Examples

```
# Create sample data
data <- data.frame(
  x = c(1, 2, 3, 4, 5),
  y = c(2, 4, 3, 5, 1)
)

# Define a simple linear boundary
boundary <- data.frame(
  x = c(1, 5),
  y = c(2, 4)
)

# Classify points
classify_boundary(data, boundary)

# Use custom class labels
classify_boundary(data, boundary, classes = c("High", "Low"))

# Complex boundary with multiple segments
boundary_complex <- data.frame(
  x = c(1, 2, 3, 4, 5),
  y = c(1, 3, 2, 4, 3)
)
classify_boundary(data, boundary_complex)
```

`create_data` *Create a 2D Synthetic Dataset With Class Labels*

Description

Generates a synthetic two-dimensional dataset using random uniform values. Each point is assigned a class label based on whether it lies above all coordinate thresholds provided. Coordinates can be supplied directly as a matrix/data frame or indirectly via a JSON file containing an array of coordinate pairs.

Usage

```
create_data(
  coord_filepath = NULL,
  coord = NULL,
  n_samples = 10000,
  type = "uniform",
  seed = 1835,
  gap = 0,
  r = 0
)
```

Arguments

`coord_filepath`	A character string specifying a file path to a JSON file containing a list/array of coordinate pairs. Optional if `coord` is supplied.
`coord`	A matrix or data frame with two columns (x and y) specifying threshold coordinates. Optional if `coord_filepath` is supplied.
`n_samples`	Integer; number of synthetic 2D points to generate.
`type`	String; How should the points be generated. Can be either 'uniform' (default), 'normal', or 'grid'
`seed`	Integer; Random seed to be used for generating dataset.
`gap`	Numeric; Value close to 0 defining gap at the boundary, less than 0.2
`r`	Numeric; Correlation for generating normal sample, between (-0.9, 0.9), default 0.

Details

A point is labelled `class = 1` if it is above **all** coordinate thresholds such that:

$$x > coord[, 1] \text{ and } y > coord[, 2]$$

Otherwise, the point is labelled `0`.

Value

A data frame with three columns:

x Random uniform x-coordinate

y Random uniform y-coordinate

class Binary class label ("Above" or "Below")

Examples

```
coords <- matrix(c(0.2, 0.3,
                   0.4, 0.5,
                   0.6, 0.7), ncol = 2, byrow = TRUE)

df <- create_data(coord = coords, n_samples = 500, seed = 717)
head(df)
```

sillysplines *Create a box to draw silly spline*

Description

This creates a simple htmlwidget to be used

Usage

```
sillysplines(width = 640, height = 640, elementId = "app")
```

Arguments

width	width of container in pixels
height	height of container in pixels
elementId	The id to be used for the container element

Value

Returns an `htmlWidget` to be used within an HTML document

Examples

```
sillysplines()
```

sillysplinesOutput *Shiny bindings for sillysplines*

Description

Output and render functions for using sillysplines within Shiny applications and interactive Rmd documents.

Usage

```
sillysplinesOutput(outputId, width = "100%", height = "400px")

renderSillysplines(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates an HTML widget (or a promise of an HTML widget).
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.
	@return Returns a htmlwidgets::shinyRenderWidget to be used in the server section of a shiny app to connect the sillysplinesOutput defined above.

Value

Returns an htmlwidgets::shinyWidgetOutput to be used within the UI of a shiny app.

Examples

```
library(shiny)
app <- shinyApp(
  ui = fluidPage(sillysplinesOutput('splines')),
  server = function(input, output) {
    splines_widget <- sillysplines()
    output$splines= renderSillysplines(splines_widget)
  }
)

if (interactive()) app
```

---

trim_shape *Trim data to be have correlated predictors*

Description

This function removes points outside of a elliptical region

Usage

```
trim_shape(data, r = 0.6)
```

Arguments

data	A matrix or data frame with three columns (x, y, class).
r	Numeric; Correlation defining the shape, between -1, 1; default 0.6

Value

A data frame with three+n_vars columns, like:

x Random uniform coordinate

y Random uniform coordinate

class Binary class label ("Above" or "Below")

Examples

```
coords <- matrix(c(0.2, 0.3,
                   0.4, 0.5,
                   0.6, 0.7), ncol = 2, byrow = TRUE)

df <- create_data(coord = coords, n_samples = 500, seed = 717)

df <- trim_shape(df)
plot(df$x, df$y, col=ifelse(df$class == "Above", "red", "yellow"))
```

Index