Package 'langevitour'

February 24, 2024

Title Langevin Tour

Version 0.7

Description An HTML widget that randomly tours 2D projections of numerical data. A random walk through projections of the data is shown. The user can manipulate the plot to use specified axes, or turn on Guided Tour mode to find an informative projection of the data. Groups within the data can be hidden or shown, as can particular axes. Points can be brushed, and the selection can be linked to other widgets using crosstalk. The underlying method to produce the random walk and projection pursuit uses Langevin dynamics. The widget can be used from within R, or included in a self-contained R Markdown or Quarto document or presentation, or used in a Shiny app.

URL `https://logarithmic.net/langevitour/`

BugReports `https://github.com/pfh/langevitour/issues/`

Imports htmlwidgets, crosstalk, RANN, assertthat

Suggests shiny, palmerpenguins, knitr, rmarkdown

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

VignetteBuilder knitr

Depends R (>= 2.10)

NeedsCompilation yes

Author Paul Harrison [aut, cre] (<`https://orcid.org/0000-0002-3980-268X`>)

Maintainer Paul Harrison `<pfh@logarithmic.net>`

Repository CRAN

Date/Publication 2024-02-24 04:30:02 UTC

R topics documented:

`knnDenoise` *k-nearest neighbor denoising of a set of points*

Description

Reduce noise in a high-dimensional dataset by averaging each point with its nearby neighbors.

Usage

```
knnDenoise(X, block = rep(1, nrow(X)), k = 30, steps = 2)
```

Arguments

`X`	A matrix of numeric data, or something that can be cast to a matrix. Each row represents a point.
`block`	Optional. A block for each row in X. A factor, or something that can be cast to a factor. Denoising will be performed independently within each block.
`k`	Number of nearest neighbors to find around each point (including itself).
`steps`	Number of steps to take along the directed k-nearest neighbor graph. `steps=1` uses the k-nearest neighbors, `steps=2` uses the k-nearest neighbors and their k-nearest neighbors, etc.

Details

`knnDenoise` first finds the `k`-nearest neighbors to each point (including the point itself). Then, for each point, the average is found of the points reachable in `steps` steps along the directed k-nearest neighbor graph.

Examples

```
library(palmerpenguins)

completePenguins <- na.omit(penguins[,c(1,3,4,5,6)])

# Dimensions need to be on comparable scales to apply knnDenoise
scaled <- scale(completePenguins[,-1])

denoised <- knnDenoise(scaled)

langevitour(denoised, completePenguins$species, pointSize=2)
```

`langevitour` *Langevin Tour*

Description

Make a Langevin Tour HTML widget, which can be used to explore high-dimensional numerical datasets.

Usage

```
langevitour(
  X,
  group = NULL,
  name = NULL,
  center = NULL,
  scale = NULL,
  extraAxes = NULL,
  lineFrom = NULL,
  lineTo = NULL,
  lineColors = NULL,
  axisColors = NULL,
  levelColors = NULL,
  colorVariation = 0.1,
  pointSize = 1,
  subsample = NULL,
  state = NULL,
  width = NULL,
  height = NULL,
  elementId = NULL,
  link = NULL,
  link_filter = TRUE
)
```

Arguments

`X`	The data to plot. A matrix of numeric data, or something that can be cast to a matrix. Rows will be shown as points in the widget. Columns are the variables of your data.
`group`	A group for each row in X, will be used to color points. A factor, or something that can be cast to a factor.
`name`	A name for each row in X.
`center`	Center for each variable. If omitted, the column means will be used.
`scale`	Scale for each variable. Scale +/- center will be the range of guaranteed visible data. If omitted, a reasonable default will be chosen, equal for all variables. (The default is the largest singular value of the centered X times 2.5.)

`extraAxes`	A matrix with each column defining a projection of interest. The columns of `X %*% extraAxes` will be presented as extra "variables".
`lineFrom`	A vector of row numbers. Draw lines starting at these rows.
`lineTo`	A vector of row numbers. Draw lines ending at these rows.
`lineColors`	Character vector. A CSS color for each line.
`axisColors`	Character vector. CSS colors for each variable and then each extra axis.
`levelColors`	Character vector. CSS colors for each level of `group`.
`colorVariation`	Number between 0 and 1. Individual points are given slightly different brightnesses. How strong should this effect be?
`pointSize`	Point radius in pixels. A single number, or a number for each row in X.
`subsample`	For speed, randomly subsample down to this many rows.
`state`	A JSON string, or an object that htmlwidgets will convert to the correct JSON. Initial widget state settings. The state of a widget can be obtained from its "further controls and information" pane. I am not going to guarantee that states will be compatible between versions of langevitour. Hint: Since JSON uses double quotes, surround the string in single quotes.
`width`	Width of widget in CSS units, for example "700px" or "100%".
`height`	Height of widget in CSS units, for example "600px" or "75vh".
`elementId`	An element ID for the widget, see `htmlwidgets::createWidget`.
`link`	A SharedData object from the crosstalk package to share selections and filters with other htmlwidgets. The data in this object is not used, just the keys and group name. The rows of `link$origData()` should correspond to the rows of X.
`link_filter`	TRUE or FALSE. If using crosstalk, should hiding groups in langevitour also cause them to be filtered in linked widgets?

Details

The only required argument is `X`, the high-dimensional collection of points. The `group` argument is also commonly used so that groups of points can be distinguished by color. Further arguments adjust the appearance or provide advanced features.

langevitour will by default not scale variables individually. If you want variables to be individually scaled, use something like `scale=apply(X,2,sd)*4`. Using the `scale` argument rather than modifying `X` directly ensures the plot axes within the widgets retain the original units.

In Javascript, the langevitour object can be obtained using `document.getElementById(elementId).langevitour`. For example you could have a button that sets the state of a widget using `document.getElementById(elementId).langevi`

Value

An htmlwidget object.

Examples

```
library(palmerpenguins)

completePenguins <- na.omit(penguins[,c(1,3,4,5,6)])
scale <- apply(completePenguins[,-1], 2, sd)*4

langevitour(
    completePenguins[,-1],
    completePenguins$species,
    scale=scale, pointSize=2)


# An example setting the widget's initial state

langevitour(
    completePenguins[,-1],
    completePenguins$species,
    scale=scale, pointSize=2,
    state='{"guideType":"pca","labelInactive":["bill_length_mm"]}')
```

`langevitour-shiny` *Shiny bindings for langevitour*

Description

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

Usage

```
langevitourOutput(outputId, width = "100%", height = "600px")

renderLangevitour(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 a langevitour, usually a block of code ending with a call to `langevitour()`
`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.

Examples

```
library(shiny)
library(palmerpenguins)

completePenguins <- na.omit(penguins[,c(1,3,4,5,6)])
scale <- apply(completePenguins[,-1], 2, sd)*4

ui <- fluidPage(
    sliderInput('zoom', 'Zoom', 0, min=-1, max=1, step=0.1),
    langevitourOutput('widget')
)

server <- function(input,output) {
    output$widget <- renderLangevitour({
        langevitour(
            completePenguins[,-1],
            completePenguins$species,
            scale=scale * 10^input$zoom, pointSize=2)
    })
}

app <- shinyApp(ui, server)

# Use runApp(app) or runGadget(app) to run app.
```

`zeiselPC` *Principal components of scRNA-Seq of mouse brain cells*

Description

Single-cell RNA-Seq gene expression of 2,816 mouse brain cells (Zeisel, 2015). The top 10 principal components were produced using the steps in the Bioconductor OSCA workflow.

Usage

```
data(zeiselPC)
```

Format

A data frame with 2,816 rows representing brain cells and 11 columns:

type Cell type.

PC1 Principal component score.

PC2 Principal component score.

PC3 Principal component score.

PC4 Principal component score.

PC5 Principal component score.

PC6 Principal component score.

PC7 Principal component score.

PC8 Principal component score.

PC9 Principal component score.

PC10 Principal component score.

References

Zeisel, A., Muñoz-Manchado, A. B., Codeluppi, S., Lönnerberg, P., La Manno, G., Juréus, A., Marques, S., Munguba, H., He, L., Betsholtz, C., Rolny, C., Castelo-Branco, G., Hjerling-Leffler, J., & Linnarsson, S. (2015). Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*, 347(6226), 1138–1142. doi:10.1126/science.aaa1934

Examples

```
data(zeiselPC)
langevitour(zeiselPC[,-1], zeiselPC$type)
```

Index