Package ‘spinebil’

July 11, 2025

Type Package

Title Investigating New Projection Pursuit Index Functions

Version 0.1.6

Description Projection pursuit is used to find interesting low-dimensional projections of high-dimensional data by optimizing an index over all possible projections. The 'spinebil' package contains methods to evaluate the performance of projection pursuit index functions using tour methods. A paper describing the methods can be found at <doi:10.1007/s00180-020-00954-8>.

License GPL-3

Encoding UTF-8

Depends R (>= 4.1.0)

Imports tourr, ggplot2, tibble, stats, dplyr, tidyr, tictoc, cassowaryr

Suggests minerva, testthat, purrr

RoxygenNote 7.3.2

NeedsCompilation no

Author Ursula Laa [aut] (ORCID: <`https://orcid.org/0000-0002-0249-6439`>), Dianne Cook [aut] (ORCID: <`https://orcid.org/0000-0002-3813-7155`>), Tina Rashid Jafari [aut, cre] (ORCID: <`https://orcid.org/0009-0008-3605-5341`>)

Maintainer Tina Rashid Jafari <`tina.rashidjafari@gmail.com`>

Repository CRAN

Date/Publication 2025-07-11 08:20:02 UTC

Contents

basisMatrix 2
basisVector 3
compareSmoothing 3
distanceDist 4
distanceToSp 5

getIndexMean 5
getTrace 6
jitterAngle 7
jitterPoints 7
pipeData 8
plotRotation 8
plotSmoothingComparison 9
plotTrace 9
profileRotation 10
scagIndex 10
sinData 11
spiralData 12
squintAngleEstimate 13
timeSequence 14

Index **15**

`basisMatrix` *Generate 2-d basis in directions i, j in n dimensions (i,j <= n)*

Description

Generate 2-d basis in directions i, j in n dimensions (i,j <= n)

Usage

```
basisMatrix(i, j, n)
```

Arguments

`i`	first basis direction
`j`	second basis direction
`n`	number of dimensions

Value

basis matrix

`basisVector` *Generate basis vector in direction i in n dimensions (i <= n)*

Description

Generate basis vector in direction i in n dimensions (i <= n)

Usage

```
basisVector(i, n)
```

Arguments

`i`	selected direction
`n`	number of dimensions

Value

basis vector

`compareSmoothing` *Compare traces with different smoothing options.*

Description

Compare traces with different smoothing options.

Usage

```
compareSmoothing(d, tPath, idx, alphaV = c(0.01, 0.05, 0.1), n = 10)
```

Arguments

`d`	Data matrix
`tPath`	Interpolated tour path (as list of projections)
`idx`	Index function
`alphaV`	Jitter amounts to compare (for jittering angle or points)
`n`	Number of evaluations entering mean value calculation

Value

Table of mean index values

Examples

```
d <- spiralData(3, 30)
tPath <- tourr::save_history(d, max_bases=2)
tPath <- as.list(tourr::interpolate(tPath, 0.3))
idx <- scagIndex("stringy")
compS <- compareSmoothing(d, tPath, idx, alphaV = c(0.01, 0.05), n=2)
plotSmoothingComparison(compS)
```

`distanceDist` *Collecting all pairwise distances between input planes.*

Description

The distribution of all pairwise distances is useful to understand the optimisation in a guided tour, to compare e.g. different optimisation methods or different number of noise dimensions.

Usage

```
distanceDist(planes, nn = FALSE)
```

Arguments

`planes`	Input planes (e.g. result of guided tour)
`nn`	Set true to only consider nearest neighbour distances (dummy, not yet implemented)

Value

numeric vector containing all distances

Examples

```
planes1 <- purrr::rerun(10, tourr::basis_random(5))
planes2 <- purrr::rerun(10, tourr::basis_random(10))
d1 <- distanceDist(planes1)
d2 <- distanceDist(planes2)
d <- tibble::tibble(dist=c(d1, d2), dim=c(rep(5,length(d1)),rep(10,length(d2))))
ggplot2::ggplot(d) + ggplot2::geom_boxplot(ggplot2::aes(factor(dim), dist))
```

`distanceToSp` *Collecting distances between input planes and input special plane.*

Description

If the optimal view is known, we can use the distance between a given plane and the optimal one as a proxy to diagnose the performance of the guided tour.

Usage

```
distanceToSp(planes, specialPlane)
```

Arguments

`planes`	Input planes (e.g. result of guided tour)
`specialPlane`	Plane defining the optimal view

Value

numeric vector containing all distances

Examples

```
planes <- purrr::rerun(10, tourr::basis_random(5))
specialPlane <- basisMatrix(1,2,5)
d <- distanceToSp(planes, specialPlane)
plot(d)
```

`getIndexMean` *Evaluate mean index value over n jittered views.*

Description

Evaluate mean index value over n jittered views.

Usage

```
getIndexMean(proj, d, alpha, idx, method = "jitterAngle", n = 10)
```

Arguments

`proj`	Original projection plane
`d`	Data matrix
`alpha`	Jitter amount (for jittering angle or points)
`idx`	Index function
`method`	Select between "jitterAngle" (default) and "jitterPoints" (otherwise we return original index value)
`n`	Number of evaluations entering mean value calculation

Value

Mean index value

`getTrace` *Tracing the index over an interpolated planned tour path.*

Description

Tracing is used to test if the index value varies smoothly over an interpolated tour path. The index value is calculated for the data d in each projection in the interpolated sequence. Note that all index functions must take the data in 2-d matrix format and return the index value.

Usage

```
getTrace(d, m, indexList, indexLabels)
```

Arguments

`d`	data
`m`	list of projection matrices for the planned tour
`indexList`	list of index functions to calculate for each entry
`indexLabels`	labels used in the output

Value

index values for each interpolation step

Examples

```
d <- spiralData(4, 100)
m <- list(basisMatrix(1,2,4), basisMatrix(3,4,4))
indexList <- list(tourr::holes(), tourr::cmass())
indexLabels <- c("holes", "cmass")
trace <- getTrace(d, m, indexList, indexLabels)
plotTrace(trace)
```

`jitterAngle` *Re-evaluate index after jittering the projection by an angle alpha.*

Description

Re-evaluate index after jittering the projection by an angle alpha.

Usage

```
jitterAngle(proj, d, alpha, idx)
```

Arguments

`proj`	Original projection plane
`d`	Data matrix
`alpha`	Jitter angle
`idx`	Index function

Value

New index value

`jitterPoints` *Re-evaluate index after jittering all points by an amount alpha.*

Description

Re-evaluate index after jittering all points by an amount alpha.

Usage

```
jitterPoints(projData, alpha, idx)
```

Arguments

`projData`	Original projected data points
`alpha`	Jitter amount (passed into the jitter() function)
`idx`	Index function

Value

New index value

`pipeData` *Generating a sample of points on a pipe*

Description

Points are drawn from a uniform distribution between -1 and 1, the pipe structure is generated by rejecting points if they are not on a circle with radius 1 and thickness t in the last two parameters.

Usage

```
pipeData(n, p, t = 0.1)
```

Arguments

`n`	sample dimensionality
`p`	number of sample points to generate
`t`	thickness of circle, default=0.1

Value

sample points in tibble format

Examples

```
pipeData(4, 100)
pipeData(2, 100, 0.5)
```

`plotRotation` *Plot rotation traces of indexes obtained with profileRotation.*

Description

Plot rotation traces of indexes obtained with profileRotation.

Usage

```
plotRotation(resMat)
```

Arguments

`resMat`	data (result of profileRotation)

Value

ggplot visualisation of the tracing data

`plotSmoothingComparison` *Plot the comparison of smoothing methods.*

Description

Plotting method for the results of compareSmoothing. The results are mapped by facetting over values of alpha and mapping the method (jitterAngle, jitterPoints, noSmoothing) to linestyle and color (black dashed, black dotted, red solid). By default legend drawing is turned off, but can be turned on via the lPos argument, e.g. setting to "bottom" for legend below the plot.

Usage

```
plotSmoothingComparison(smMat, lPos = "none")
```

Arguments

`smMat`	Result from compareSmoothing
`lPos`	Legend position passed to ggplot2 (default is none for no legend shown)

Value

ggplot visualisation of the comparison

`plotTrace` *Plot traces of indexes obtained with `getTrace`.*

Description

Plot traces of indexes obtained with `getTrace`.

Usage

```
plotTrace(resMat, rescY = TRUE)
```

Arguments

`resMat`	data (result of getTrace)
`rescY`	bool to fix y axis range to [0,1]

Value

ggplot visualisation of the tracing data

`profileRotation` *Test rotation invariance of index functions for selected 2-d data set.*

Description

Ideally a projection pursuit index should be roation invariant, we test this explicitly by profiling the index while rotating a distribution.

Usage

```
profileRotation(d, indexList, indexLabels, n = 200)
```

Arguments

`d`	data (2 column matrix containing distribution to be rotated)
`indexList`	list of index functions to calculate for each entry
`indexLabels`	labels used in the output
`n`	number of steps in the rotation (default = 200)

Value

index values for each rotation step

Examples

```
d <- as.matrix(sinData(2, 30))
indexList <- list(tourr::holes(), scagIndex("stringy"), mineIndexE("MIC"))
indexLabels <- c("holes", "skinny", "mic")
pRot <- profileRotation(d, indexList, indexLabels, n = 50)
plotRotation(pRot)
```

`scagIndex` *Matching index functions to the required format.*

Description

These are convenicence functions that format scagnostics and mine index functions for direct use with the guided tour or other functionalities in this package.

Usage

```
scagIndex(indexName)

mineIndex(indexName)

mineIndexE(indexName)

holesR()

cmassR()
```

Arguments

`indexName`	Index name to select from group of indexes.

Value

function taking 2-d data matrix and returning the index value

Functions

- `scagIndex()`: Scagnostics index from binostics package
- `mineIndex()`: MINE index from minerva package
- `mineIndexE()`: MINE index from minerva package (updated estimator)
- `holesR()`: rescaling the tourr holes index
- `cmassR()`: rescaling the tourr cmass index

`sinData` *Generating sine wave sample*

Description

n-1 points are drawn from a normal distribution with mean=0, sd=1, the points in the final direction are calculated as the sine of the values of direction n-1 with additional jittering controled by the jitter factor f.

Usage

```
sinData(n, p, f = 1)
```

Arguments

`n`	sample dimensionality
`p`	number of sample points to generate
`f`	jitter factor, default=1

Value

sample points in tibble format

Examples

```
sinData(4, 100)
sinData(2, 100, 200)
```

`spiralData` *Generating spiral sample*

Description

n-2 points are drawn from a normal distribution with mean=0, sd=1, the points in the final two direction are sampled along a spiral by samping the angle from a normal distribution with mean=0, sd=2*pi (absolute values are used to fix the orientation of the spiral).

Usage

```
spiralData(n, p)
```

Arguments

`n`	sample dimensionality
`p`	number of sample points to generate

Value

sample points in matrix format

Examples

```
spiralData(4, 100)
```

`squintAngleEstimate` *Estimating squint angle of 2-d structure in high-d dataset under selected index.*

Description

We estimate the squint angle by interpolating from a random starting plane towards the optimal view until the index value of the selected index function is above the selected cutoff. Since this depends on the direction, this is repeated with n randomly selected planes giving a distribution representative of the squint angle.

Usage

```
squintAngleEstimate(
  data,
  indexF,
  cutoff,
  structurePlane,
  n = 100,
  stepSize = 0.01
)
```

Arguments

`data`	Input data
`indexF`	Index function
`cutoff`	Threshold index value above which we assume the structure to be visible
`structurePlane`	Plane defining the optimal view
`n`	Number of random starting planes (default = 100)
`stepSize`	Interpolation step size fixing the accuracy (default = 0.01)

Value

numeric vector containing all squint angle estimates

Examples

```
data <- spiralData(4, 50)
indexF <- scagIndex("stringy")
cutoff <- 0.7
structurePlane <- basisMatrix(3,4,4)
squintAngleEstimate(data, indexF, cutoff, structurePlane, n=1)
```

`timeSequence` *Time each index evaluation for projections in the tour path.*

Description

Index evaluation timing may depend on the data distribution, we evaluate the computing time for a set of different projections to get an overview of the distribution of computing times.

Usage

```
timeSequence(d, t, idx, pmax)
```

Arguments

`d`	Input data in matrix format
`t`	List of projection matrices (e.g. interpolated tour path)
`idx`	Index function
`pmax`	Maximum number of projections to evaluate (cut t if longer than pmax)

Value

numeric vector containing all distances

Examples

```
d <- spiralData(4, 500)
t <- purrr::map(1:10, ~ tourr::basis_random(4))
idx <- scagIndex("stringy")
timeSequence(d, t, idx, 10)
```

Index

`basisMatrix`, 2
`basisVector`, 3

`cmassR` *(*`scagIndex`*)*, 10
`compareSmoothing`, 3

`distanceDist`, 4
`distanceToSp`, 5

`getIndexMean`, 5
`getTrace`, 6, *9*

`holesR` *(*`scagIndex`*)*, 10

`jitterAngle`, 7
`jitterPoints`, 7

`mineIndex` *(*`scagIndex`*)*, 10
`mineIndexE` *(*`scagIndex`*)*, 10

`pipeData`, 8
`plotRotation`, 8
`plotSmoothingComparison`, 9
`plotTrace`, 9
`profileRotation`, 10

`scagIndex`, 10
`sinData`, 11
`spiralData`, 12
`squintAngleEstimate`, 13

`timeSequence`, 14