Package ‘spherepc’

October 14, 2022

Type Package

Title Spherical Principal Curves

Version 0.1.7

Author Jongmin Lee [aut, cre],
Jang-Hyun Kim [ctb],
Hee-Seok Oh [aut]

Maintainer Jongmin Lee <jongminlee9218@gmail.com>

Description Fitting dimension reduction methods to data lying on two-dimensional sphere. This package provides principal geodesic analysis, principal circle, principal curves proposed by Hauberg, and spherical principal curves. Moreover, it offers the method of locally defined principal geodesics which is underway. The detailed procedures are described in Lee, J., Kim, J.-H. and Oh, H.-S. (2021) <doi:10.1109/TPAMI.2020.3025327>. Also see Kim, J.-H., Lee, J. and Oh, H.-S. (2020) <arXiv:2003.02578>.

Depends R (>= 3.5.0)

License GPL (>= 3)

Encoding UTF-8

Imports geosphere, rgl, sphereplot, stats

SystemRequirements XQuartz (on MacOS)

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2021-10-07 06:50:02 UTC

R topics documented:

`Cal.recon` *Calculating reconstruction error*

Description

This function calculates reconstruction error.

Usage

```
Cal.recon(data, line)
```

Arguments

`data`	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude.
`line`	longitude and latitude of a line as a matrix or data frame with two columns.

Details

This function calculates reconstruction error from the data to the line. Longitude should range from -180 to 180 and latitude from -90 to 90. This function requires to load 'geosphere' R package.

Value

summation of squared distance from the data to the line on the unit sphere.

Author(s)

Jongmin Lee

Examples

```
library(geosphere)           # This function needs to load 'geosphere' R packages.
data <- rbind(c(0, 0), c(50, -10), c(100, -70))
line <- rbind(c(30, 30), c(-20, 50), c(50, 80))
Cal.recon(data, line)
```

`Crossprod` *Crossproduct of vectors*

Description

This function performs the cross product of two three-dimensional vectors.

Usage

```
Crossprod(vec1, vec2)
```

Arguments

vec1	three-dimensional vector.
vec2	three-dimensional vector.

Details

This function performs the cross product of two three-dimensional vectors.

Value

three-dimensional vector.

Author(s)

Jongmin Lee

Examples

```
Crossprod(c(1, 1, 1), c(5,6,10))
```

`Dist.pt` *The number of distinct points.*

Description

This function calculates the number of distinct point in the given data.

Usage

```
Dist.pt(data)
```

Arguments

`data`	matrix or dataframe consisting of spatial locations with two columns. Each row represents longitude and latitude.

Details

This function calculates the number of distinct points in the given data.

Value

a numeric.

Author(s)

Jongmin Lee

Examples

```
Dist.pt(rbind(c(0, 0), c(0, 1), c(1, 0), c(1, 1), c(0, 0)))
```

`Earthquake` *Earthquake*

Description

It is an earthquake data from the U.S Geological Survey that collect significant earthquakes (8+ Mb magnitude) around the Pacific Ocean since the year 1900. The data are available from (https://www.usgs.gov). Additionally, note that distribution of the data has the following features: 1) scattered, 2) curvilinear one-dimensional structure on the sphere.

Usage

```
data(Earthquake)
```

Format

A data frame consisting of time, latitude, longitude, depth, magnitude, etc.

Examples

```
data(Earthquake)
names(Earthquake)
# collect spatial locations (longitude/latitude) of data.
earthquake <- cbind(Earthquake$longitude, Earthquake$latitude)
library(rgl)
library(sphereplot)
library(geosphere)
#### example 1: principal geodesic analysis (PGA)
PGA(earthquake)

#### example 2: principal circle
circle <- PrincipalCircle(earthquake)         # get center and radius of principal circle
PC <- GenerateCircle(circle[1:2], circle[3])   # generate Principal circle
# plot
sphereplot::rgl.sphgrid()
sphereplot::rgl.sphpoints(earthquake, radius = 1, col = "blue", size = 12)
sphereplot::rgl.sphpoints(PC, radius = 1, col = "red", size = 9)

#### example 3: spherical principal curves (SPC, SPC.Hauberg)
SPC(earthquake)         # spherical principal curves.
SPC.Hauberg(earthquake) # principal curves by Hauberg on sphere.

#### example 4: local principal geodesics (LPG)
LPG(earthquake, scale = 0.5, nu = 0.2, maxpt = 20)
LPG(earthquake, scale = 0.4, nu = 0.3)
```

Expmap *Exponential map*

Description

This function performs the exponential map at (0, 0, 1) on the unit 2-sphere.

Usage

```
Expmap(vec)
```

Arguments

vec	element of two-dimensional Euclidean vector space.

Details

This function performs exponential map at (0, 0, 1) on the unit sphere. `vec` is an element of the tangent plane of the unit sphere at (0, 0, 1), and the result is an element of the unit sphere in three-dimensional Euclidean space.

Value

three-dimensional vector.

Author(s)

Jongmin Lee

References

Fletcher, P. T., Lu, C., Pizer, S. M. and Joshi, S. (2004). Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Transactions on Medical Imaging, 23, 995-1005.

See Also

`Logmap`.

Examples

```
Expmap(c(1, 2))
```

`ExtrinsicMean` *Finding Extrinsic Mean*

Description

This function identifies the extrinsic mean of data on the unit 2-sphere.

Usage

```
ExtrinsicMean(data, weights = rep(1, nrow(data)))
```

Arguments

`data`	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude (denoted by degrees).
`weights`	vector of weights.

Details

This function identifies the extrinsic mean of data.

Value

two-dimensional vector.

Note

In the case of spheres, if data set is not contained in a hemisphere, then it is possible that the extrinsic mean of the data set does not exists; for example, a great circle.

Author(s)

Jongmin Lee

References

Jongmin Lee, Jang-Hyun Kim and Hee-Seok Oh. (2021). Spherical Principal Curves. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43, 2165-2171. <doi.org/10.1109/TPAMI.2020.3025327>.
Jang-Hyun Kim, Jongmin Lee and Hee-Seok Oh. (2020). Spherical Principal Curves <arXiv:2003.02578>.

See Also

IntrinsicMean.

Examples

```
#### comparison of Intrinsic mean and extrinsic mean.
#### example: noisy circular data set.
library(rgl)
library(sphereplot)
library(geosphere)
n <- 500                  # the number of samples.
x <- 360 * runif(n) - 180
sigma <- 5
y <- 60 + sigma * rnorm(n)
simul.circle <- cbind(x, y)
data <- simul.circle
In.mean <- IntrinsicMean(data)
Ex.mean <- ExtrinsicMean(data)
## plot (color of data is "blue"; that of intrinsic mean is "red" and
## that of extrinsic mean is "green".)
sphereplot::rgl.sphgrid()
sphereplot::rgl.sphpoints(data, radius = 1, col = "blue", size = 12)
sphereplot::rgl.sphpoints(In.mean[1], In.mean[2], radius = 1, col = "red", size = 12)
sphereplot::rgl.sphpoints(Ex.mean[1], Ex.mean[2], radius = 1, col = "green", size = 12)
```

GenerateCircle *Generating circle on sphere*

Description

This function makes a circle on the unit 2-sphere.

Usage

```
GenerateCircle(center, radius, T = 1000)
```

Arguments

`center`	center of circle with spatial locations (longitude and latitude denoted by degrees).
`radius`	radius of circle. It should be in [0, pi].
`T`	the number of points that make up circle. The points in circle are equally spaced. The default is 1000.

Details

This function makes a circle on the unit 2-sphere.

Value

matrix consisting of spatial locations with two columns.

Author(s)

Jongmin Lee

See Also

PrincipalCircle.

Examples

```
library(rgl)
library(sphereplot)
library(geosphere)
circle <- GenerateCircle(c(0, 0), 1)
# plot (It requires to load 'rgl', 'sphereplot', and 'geosphere' R package.)
sphereplot::rgl.sphgrid()
sphereplot::rgl.sphpoints(circle[, 1], circle[, 2], radius = 1, col = "blue", size = 12)
```

`IntrinsicMean` *Finding Intrinsic Mean*

Description

This function calculates the intrinsic mean of data on sphere.

Usage

```
IntrinsicMean(data, weights = rep(1, nrow(data)), thres = 1e-5)
```

Arguments

`data`	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude (denoted by degrees).
`weights`	vector of weights.
`thres`	threshold of the stopping conditions.

Details

This function calculates the intrinsic mean of data. The intrinsic mean is found by the gradient descent algorithm, which works well if the data is well-localized. In the case of spheres, if data is contained in a hemisphere, then the algorithm converges.

Value

two-dimensional vector.

Author(s)

Jongmin Lee

References

Fletcher, P. T., Lu, C., Pizer, S. M. and Joshi, S. (2004). Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Transactions on Medical Imaging, 23, 995-1005. Jongmin Lee, Jang-Hyun Kim and Hee-Seok Oh. (2021). Spherical Principal Curves. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43. 2165-2171. <doi.org/10.1109/TPAMI.2020.3025327>.

See Also

`ExtrinsicMean`.

Examples

```
#### comparison of Intrinsic mean and extrinsic mean.
#### example: circular data set.
library(rgl)
library(sphereplot)
library(geosphere)
n <- 500
x <- 360 * runif(n) - 180
sigma <- 5
y <- 60 + sigma * rnorm(n)
simul.circle <- cbind(x, y)
data <- simul.circle
In.mean <- IntrinsicMean(data)
Ex.mean <- ExtrinsicMean(data)
## plot (color of data is "blue"; that of intrinsic mean is "red" and
## that of extrinsic mean is "green".)
sphereplot::rgl.sphgrid()
sphereplot::rgl.sphpoints(data, radius = 1, col = "blue", size = 12)
sphereplot::rgl.sphpoints(In.mean[1], In.mean[2], radius = 1, col = "red", size = 12)
sphereplot::rgl.sphpoints(Ex.mean[1], Ex.mean[2], radius = 1, col = "green", size = 12)
```

`Kernel.Gaussian` *Gaussian kernel function*

Description

This function returns the value of a Gaussian kernel function.

Usage

```
Kernel.Gaussian(vec)
```

Arguments

`vec`	any length of vector.

Details

This function returns the value of a Gaussian kernel function. The value of kernel represents the similarity from origin. The function returns a vector whose length is same as `vec`.

Value

vector.

Author(s)

Jongmin Lee

See Also

`Kernel.indicator`, `Kernel.quartic`.

Examples

```
Kernel.Gaussian(c(0, 1/2, 1, 2))
```

`Kernel.indicator` *Indicator kernel function*

Description

This function returns the value of an indicator kernel function.

Usage

```
Kernel.indicator(vec)
```

Arguments

`vec`	any length of vector.

Details

This function returns the value of an indicator kernel function. The value of kernel represents similarity from origin. The function returns a vector whose length is same as `vec`.

Value

vector.

Author(s)

Jongmin Lee

See Also

`Kernel.Gaussian`, `Kernel.quartic`.

Examples

```
Kernel.indicator(c(0, 1/2, 1, 2))
```

`Kernel.quartic` *Quartic kernel function*

Description

This function returns the value of a quartic kernel function.

Usage

```
Kernel.quartic(vec)
```

Arguments

`vec`	any length of vector.

Details

This function returns the value of quartic kernel function. The value of kernel represents similarity from origin. The function returns a vector whose length is same as `vec`.

Value

vector.

Author(s)

Jongmin Lee

See Also

`Kernel.Gaussian`, `Kernel.indicator`.

Examples

```
Kernel.quartic(c(0, 1/2, 1, 2))
```

`Logmap` *Logarithm map*

Description

This function performs the logarithm map at (0, 0, 1) on the unit sphere.

Usage

```
Logmap(vec)
```

Arguments

`vec`	element of the unit sphere in three-dimensional Euclidean vector space.

Details

This function performs the logarithm map at (0, 0, 1) on the unit sphere. Note that, `vec` is normalized to be contained in the unit sphere.

Value

two-dimensional vector.

Author(s)

Jongmin Lee

References

Fletcher, P. T., Lu, C., Pizer, S. M. and Joshi, S. (2004). Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Transactions on Medical Imaging, 23, 995-1005.

See Also

`Expmap`.

Examples

```
Logmap(c(1/sqrt(2), 1/sqrt(2), 0))
```

LPG *Local principal geodesics*

Description

Locally definded principal geodesic analysis.

Usage

```
LPG(data, scale = 0.04, tau = scale/3, nu = 0, maxpt = 500,
seed = NULL, kernel = "indicator", thres = 1e-4,
col1 = "blue", col2 = "green", col3 = "red")
```

Arguments

`data`	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude (denoted by degrees).
`scale`	scale parameter for this function. The argument is the degree to which `LPG` expresses data locally; thus, as `scale` grows as the result of `LPG` become similar to that of the `PGA` function. The default is 0.4.
`tau`	forwarding or backwarding distance of each step. It is empirically recommended to choose a third of `scale`, which is the default of this argument.
`nu`	parameter to alleviate the bias of resulting curves. `nu` represents the viscosity of the given data and it should be selected in [0, 1). The default is zero. When the `nu` is close to 1, the curves usually swirl around, similar to the motion of a large viscous fluid. The swirling can be controlled by the argument `maxpt`.
`maxpt`	maximum number of points that each curve has. The default is 500.
`seed`	random seed number.
`kernel`	kind of kernel function. The default is the indicator kernel and alternatives are quartic or Gaussian.
`thres`	threshold of the stopping condition for the `IntrinsicMean` contained in the `LPG` function. The default is 1e-4.
`col1`	color of data. The default is blue.
`col2`	color of points in the resulting principal curves. The default is green.
`col3`	color of the resulting curves. The default is red.

Details

Locally definded principal geodesic analysis. The result is sensitive to `scale` and `nu`, especially `scale` should be carefully chosen according to the structure of the given data.

Value

plot and a list consisting of

`prin.curves`	spatial locations (represented by degrees) of points in the resulting curves.
`line`	connecting lines between points in `prin.curves`.
`num.curves`	the number of the resulting curves.

Author(s)

Jongmin Lee

See Also

PGA, SPC, SPC.Hauberg.

Examples

```
library(rgl)
library(sphereplot)
library(geosphere)

#### example 1: spiral data
## longitude and latitude are expressed in degrees
set.seed(40)
n <- 900                                    # the number of samples
sigma1 <- 1; sigma2 <- 2.5;                 # noise levels
radius <- 73; slope <- pi/16                # radius and slope of spiral
## polar coordinate of (longitude, latitude)
r <- runif(n)^(2/3) * radius; theta <- -slope * r + 3
## transform to (longitude, latitude)
correction <- (0.5 * r/radius + 0.3)        # correction of noise level
lon1 <- r * cos(theta) + correction * sigma1 * rnorm(n)
lat1 <- r * sin(theta) + correction * sigma1 * rnorm(n)
lon2 <- r * cos(theta) + correction * sigma2 * rnorm(n)
lat2 <- r * sin(theta) + correction * sigma2 * rnorm(n)
spiral1 <- cbind(lon1, lat1); spiral2 <- cbind(lon2, lat2)
## plot spiral data
rgl.sphgrid(col.lat = 'black', col.long = 'black')
rgl.sphpoints(spiral1, radius = 1, col = 'blue', size = 12)
## implement the LPG to (noisy) spiral data
LPG(spiral1, scale = 0.06, nu = 0.1, seed = 100)
LPG(spiral2, scale = 0.12, nu = 0.1, seed = 100)
#### example 2: zigzag data set
set.seed(10)
n <- 50                                # the number of samples is 6 * n = 300
sigma1 <- 2; sigma2 <- 5               # noise levels
x1 <- x2 <- x3 <- x4 <- x5 <- x6 <- runif(n) * 20 - 20
y1 <- x1 + 20 + sigma1 * rnorm(n); y2 <- -x2 + 20 + sigma1 * rnorm(n)
y3 <- x3 + 60 + sigma1 * rnorm(n); y4 <- -x4 - 20 + sigma1 * rnorm(n)
y5 <- x5 - 20 + sigma1 * rnorm(n); y6 <- -x6 - 60 + sigma1 * rnorm(n)
x <- c(x1, x2, x3, x4, x5, x6); y <- c(y1, y2, y3, y4, y5, y6)
```

```
simul.zigzag1 <- cbind(x, y)
## plot zigzag data
sphereplot::rgl.sphgrid(col.lat = 'black', col.long = 'black')
sphereplot::rgl.sphpoints(simul.zigzag1, radius = 1, col = 'blue', size = 12)
## implement the LPG to zigzag data
LPG(simul.zigzag1, scale = 0.1, nu = 0.1, maxpt = 45, seed = 50)

## noisy zigzag data
set.seed(10)
z1 <- z2 <- z3 <- z4 <- z5 <- z6 <- runif(n) * 20 - 20
w1 <- z1 + 20 + sigma2 * rnorm(n); w2 <- -z2 + 20 + sigma2 * rnorm(n)
w3 <- z3 + 60 + sigma2 * rnorm(n); w4 <- -z4 - 20 + sigma2 * rnorm(n)
w5 <- z5 - 20 + sigma2 * rnorm(n); w6 <- -z6 - 60 + sigma2 * rnorm(n)
z <- c(z1, z2, z3, z4, z5, z6); w <- c(w1, w2, w3, w4, w5, w6)
simul.zigzag2 <- cbind(z, w)
## implement the LPG to noisy zigzag data
LPG(simul.zigzag2, scale = 0.2, nu = 0.1, maxpt = 18, seed = 20)


#### example 3: Doubly circular data set
set.seed(30)
n <- 200
sigma <- 1
x1 <- 40 * runif(n) - 20
y1 <- (-x1^2 + 400)^(1/2) + 30 + sigma * rnorm(n)
x2 <- 40 * runif(n) - 20
y2 <- -(-x2^2 + 400)^(1/2) + 30 + sigma * rnorm(n)
y3 <- 40 * runif(n) + 10
x3 <- -(-y3^2 + 60 * y3 - 500)^(1/2) + sigma * rnorm(n)
y4 <- 40 * runif(n) + 10
x4 <- (-y4^2 + 60 * y4 - 500)^(1/2) + sigma * rnorm(n)
Dc1 <- cbind(c(x1, x2, x3, x4), c(y1, y2, y3, y4))
z1 <- 40 * runif(n) - 20
w1 <- (400 - z1^2)^(1/2) - 20 + sigma * rnorm(n)
z2 <- 40 * runif(n) - 20
w2 <- -(400 - z2^2)^(1/2) - 20 + sigma * rnorm(n)
w3 <- -40 * runif(n)
z3 <- (-w3^2 - 40 * w3)^(1/2) + sigma * rnorm(n)
w4 <- -40 * runif(n)
z4 <- -(-w4^2 - 40 * w4)^(1/2) + sigma * rnorm(n)
Dc2 <- cbind(c(z1, z2, z3, z4), c(w1, w2, w3, w4))
Dc <- rbind(Dc1, Dc2)
LPG(Dc, scale = 0.15, nu = 0.1, maxpt = 22,)


#### example 4: real earthquake data
data(Earthquake)
names(Earthquake)
earthquake <- cbind(Earthquake$longitude, Earthquake$latitude)
LPG(earthquake, scale = 0.5, nu = 0.2, maxpt = 20)
LPG(earthquake, scale = 0.4, nu = 0.3)
```

```
#### example 5: tree data
## tree consists of stem, branches and subbranches

## generate stem
set.seed(10)
n1 <- 200; n2 <- 100; n3 <- 15 # the number of samples in stem, a branch, and a subbranch
sigma1 <- 0.1; sigma2 <- 0.05; sigma3 <- 0.01 # noise levels
noise1 <- sigma1 * rnorm(n1); noise2 <- sigma2 * rnorm(n2); noise3 <- sigma3 * rnorm(n3)
l1 <- 70; l2 <- 20; l3 <- 1            # length of stem, branches, and subbranches
rep1 <- l1 * runif(n1)                 # repeated part of stem
stem <- cbind(0 + noise1, rep1 - 10)

## generate branch
rep2 <- l2 * runif(n2)                 # repeated part of branch
branch1 <- cbind(-rep2, rep2 + 10 + noise2); branch2 <- cbind(rep2, rep2 + noise2)
branch3 <- cbind(rep2, rep2 + 20 + noise2); branch4 <- cbind(rep2, rep2 + 40 + noise2)
branch5 <- cbind(-rep2, rep2 + 30 + noise2)
branch <- rbind(branch1, branch2, branch3, branch4, branch5)

## generate subbranches
rep3 <- l3 * runif(n3)                 # repeated part in subbranches
branches1 <- cbind(rep3 - 10, rep3 + 20 + noise3)
branches2 <- cbind(-rep3 + 10, rep3 + 10 + noise3)
branches3 <- cbind(rep3 - 14, rep3 + 24 + noise3)
branches4 <- cbind(-rep3 + 14, rep3 + 14 + noise3)
branches5 <- cbind(-rep3 - 12, -rep3 + 22 + noise3)
branches6 <- cbind(rep3 + 12, -rep3 + 12 + noise3)
branches7 <- cbind(-rep3 - 16, -rep3 + 26 + noise3)
branches8 <- cbind(rep3 + 16, -rep3 + 16 + noise3)
branches9 <- cbind(rep3 + 10, -rep3 + 50 + noise3)
branches10 <- cbind(-rep3 - 10, -rep3 + 40 + noise3)
branches11 <- cbind(-rep3 + 12, rep3 + 52 + noise3)
branches12 <- cbind(rep3 - 12, rep3 + 42 + noise3)
branches13 <- cbind(rep3 + 14, -rep3 + 54 + noise3)
branches14 <- cbind(-rep3 - 14, -rep3 + 44 + noise3)
branches15 <- cbind(-rep3 + 16, rep3 + 56 + noise3)
branches16 <- cbind(rep3 - 16, rep3 + 46 + noise3)
branches17 <- cbind(-rep3 + 10, rep3 + 30 + noise3)
branches18 <- cbind(-rep3 + 14, rep3 + 34 + noise3)
branches19 <- cbind(rep3 + 16, -rep3 + 36 + noise3)
branches20 <- cbind(rep3 + 12, -rep3 + 32 + noise3)
sub.branches <- rbind(branches1, branches2, branches3, branches4, branches5, branches6,
+    branches7, branches8, branches9, branches10, branches11, branches12, branches13,
+    branches14, branches15, branches16, branches17, branches18, branches19, branches20)

## tree consists of stem, branch and subbranches
tree <- rbind(stem, branch, sub.branches)

## plot tree data
sphereplot::rgl.sphgrid(col.lat = 'black', col.long = 'black')
sphereplot::rgl.sphpoints(tree, radius = 1, col = 'blue', size = 12)

## implement the LPG function to tree data
```

```
LPG(tree, scale = 0.03, nu = 0.2, seed = 10)
```

PGA *Principal geodesic analysis*

Description

This function performs principal geodesic analysis.

Usage

```
PGA(data, col1 = "blue", col2 = "red")
```

Arguments

`data`	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude (denoted by degrees.
`col1`	color of data. The default is blue.
`col2`	color of the principal geodesic line. The default is red

Details

This function performs principal geodesic analysis.

Value

plot and a list consisting of

`line`	spatial locations (longitude and latitude by degrees) of points in the principal geodesic line.

Note

This function requires to load 'sphereplot', 'geosphere' and 'rgl' R package.

Author(s)

Jongmin Lee

References

Fletcher, P. T., Lu, C., Pizer, S. M. and Joshi, S. (2004). Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Transactions on Medical Imaging, 23, 995-1005.

See Also

`LPG`.

Examples

```
library(rgl)
library(sphereplot)
library(geosphere)
#### example 1: noisy half-great circle data
circle <- GenerateCircle(c(150, 60), radius = pi/2)
half.circle <- circle[circle[, 1] < 0, , drop = FALSE]
sigma <- 2
half.circle <- half.circle + sigma * rnorm(nrow(half.circle))
PGA(half.circle)

#### example 2: noisy S-shaped data
#### The data consists of two parts: x ~ Uniform[0, 20], y = sqrt(20 * x - x^2) + N(0, sigma^2),
#### x ~ Uniform[-20, 0], y = -sqrt(-20 * x - x^2) + N(0, sigma^2).
n <- 500
x <- 60 * runif(n)
sigma <- 2
y <- (60 * x - x^2)^(1/2) + sigma * rnorm(n)
simul.S1 <- cbind(x, y)
z <- -60 * runif(n)
w <- -(-60 * z - z^2)^(1/2)+ sigma * rnorm(n)
simul.S2 <- cbind(z, w)
simul.S <- rbind(simul.S1, simul.S2)
PGA(simul.S)
```

`PrincipalCircle` *Principal circle on a sphere*

Description

This function fits a principal circle on sphere via gradient descent algorithm.

Usage

```
PrincipalCircle(data, step.size = 1e-3, thres = 1e-5, maxit = 10000)
```

Arguments

`data`	matrix or data frame consisting of spatial locations (longitude and latitude denoted by degrees) with two columns.
`step.size`	step size of gradient descent algorithm. For convergence of the algorithm, `step.size` is recommended to be below 0.01. The default is 1e-3.
`thres`	threshold of the stopping condition. The default is 1e-5.
`maxit`	maximum number of iterations. The default is 10000.

Details

This function fits a principal circle on sphere via gradient descent algorithm. The function returns three-dimensional vectors whose components represent longitude and latitude of the center and the radius of the circle in regular order.

Value

three-dimensional vector.

Author(s)

Jongmin Lee

References

Jongmin Lee, Jang-Hyun Kim and Hee-Seok Oh. (2021). Spherical principal curves. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43, 2165-2171. <doi.org/10.1109/TPAMI.2020.3025327>.

Jang-Hyun Kim, Jongmin Lee and Hee-Seok Oh (2020), Spherical principal curves <arXiv:2003.02578>.

See Also

GenerateCircle

Examples

```
library(rgl)
library(sphereplot)
library(geosphere)
#### example 1: half-great circle data
circle <- GenerateCircle(c(150, 60), radius = pi/2)
half.great.circle <- circle[circle[, 1] < 0, , drop = FALSE]
sigma <- 2
half.great.circle <- half.great.circle + sigma * rnorm(nrow(half.great.circle))
## find a principal circle

PC <- PrincipalCircle(half.great.circle)
result <- GenerateCircle(PC[1:2], PC[3])
## plot
rgl.sphgrid()
rgl.sphpoints(half.great.circle, radius = 1, col = "blue", size = 12)
rgl.sphpoints(result, radius = 1, col = "red", size = 6)

#### example 2: circular data
n <- 700
x <- seq(-180, 180, length.out = n)
sigma <- 5
y <- 45 + sigma * rnorm(n)
simul.circle <- cbind(x, y)
## find a principal circle
PC <- PrincipalCircle(simul.circle)
result <- GenerateCircle(PC[1:2], PC[3])
```

```
## plot
sphereplot::rgl.sphgrid()
sphereplot::rgl.sphpoints(simul.circle, radius = 1, col = "blue", size = 12)
sphereplot::rgl.sphpoints(result, radius = 1, col = "red", size = 6)

#### example 3: earthquake data
data(Earthquake)
names(Earthquake)
earthquake <- cbind(Earthquake$longitude, Earthquake$latitude)
PC <- PrincipalCircle(earthquake)
result <- GenerateCircle(PC[1:2], PC[3])
## plot
sphereplot::rgl.sphgrid(col.long = "black", col.lat = "black")
sphereplot::rgl.sphpoints(earthquake, radius = 1, col = "blue", size = 12)
sphereplot::rgl.sphpoints(result, radius = 1, col = "red", size = 6)
```

Proj.Hauberg *Projecting the nearest point*

Description

This function performs the approximated projection for each data.

Usage

```
Proj.Hauberg(data, line)
```

Arguments

data	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude.
line	longitude and latitude of line as a matrix or data frame with two columns.

Details

This function returns the nearest points in `line` for each point in the data. The function requires to load the 'geosphere' R package.

Value

matrix consisting of spatial locations with two columns.

Author(s)

Jongmin Lee

References

Hauberg, S. (2016). Principal curves on Riemannian manifolds. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38, 1915-1921.

Jang-Hyun Kim, Jongmin Lee and Hee-Seok Oh. (2020). Spherical Principal Curves <arXiv:2003.02578>.

Jongmin Lee, Jang-Hyun Kim and Hee-Seok Oh. (2021). Spherical principal curves. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43, 2165-2171. <doi.org/10.1109/TPAMI.2020.3025327>.

See Also

SPC.Hauberg

Examples

```
library(geosphere)
Proj.Hauberg(rbind(c(0, 0), c(10, -20)), rbind(c(50, 10), c(40, 20), c(30, 30)))
```

Rotate *Rotating point on a sphere*

Description

Rotate a point on the unit 2-sphere.

Usage

```
Rotate(pt1, pt2)
```

Arguments

pt1	spatial location.
pt2	spatial location.

Details

This function rotates `pt2` to the extent that `pt1` to spherical coordinate (0, 90). The function returns a point as a form of three-dimensional Euclidean coordinate.

Value

three-dimensional vector.

Author(s)

Jongmin Lee

References

https://en.wikipedia.org/wiki/Rodrigues_rotation_formula

See Also

Rotate.inv.

Examples

```
## If "pt1" is north pole (= (0, 90)), Rotate() function returns Euclidean coordinate of "pt2".
Rotate(c(0, 90), c(10, 10))  # It returns Euclidean coordinate of spatial location (10, 10).
# The Trans.Euclid() function converts spatial coordinate (10, 10) to Euclidean coordinate.
Trans.Euclid(c(10, 10))
```

`Rotate.inv` *Rotating point on a sphere*

Description

Rotate a point on the unit 2-sphere.

Usage

```
Rotate.inv(pt1, pt2)
```

Arguments

`pt1`	spatial location.
`pt2`	spatial location.

Details

This function rotates `pt2` to the extent that the spherical coordinate (0, 90) is rotated to `pt1`. The function is the inverse of the `Rotate` function, and returns a point as a form of three-dimensional Euclidean coordinate.

Value

three-dimensional vector.

Author(s)

Jongmin Lee

References

https://en.wikipedia.org/wiki/Rodrigues_rotation_formula

See Also

Rotate.

Examples

```
## If "pt1" is north pole (= (0, 90)), Rotate.inv() returns Euclidean coordinate of "pt2".
# It returns Euclidean coordinate of spatial location (-100, 80).
Rotate.inv(c(0, 90), c(-100, 80))
# It converts spatial coordinate (-100, 80) to Euclidean coordinate.
Trans.Euclid(c(-100, 80))
```

`SPC` *Spherical principal curves*

Description

This function fits a spherical principal curve.

Usage

```
SPC(data, q = 0.1, T = nrow(data), step.size = 1e-3, maxit = 10,
type = "Intrinsic", thres = 0.1, deletePoints = FALSE, plot.proj = FALSE,
kernel = "quartic", col1 = "blue", col2 = "green", col3 = "red")
```

Arguments

`data`	matrix or data frame consisting of spatial locations with two columns. Each row represents a longitude and latitude (denoted by degrees).
`q`	numeric value of the smoothing parameter. The parameter plays the same role, as the bandwidth does in kernel regression, in the `SPC` function. The value should be a numeric value between 0.01 and 0.5. The default is 0.1.
`T`	the number of points making up the resulting curve. The default is 1000.
`step.size`	step size of the `PrincipalCircle` function. The default is 0.001. The resulting principal circle is used by an initial estimate of the `SPC`.
`maxit`	maximum number of iterations. The default is 30.
`type`	type of mean on the sphere. The default is "Intrinsic" and the other choice is "Extrinsic".
`thres`	threshold of the stopping condition. The default is 0.1
`deletePoints`	logical value. The argument is an option of whether to delete points or not. If `deletePoints` is FALSE, this function leaves the points in curves which do not have adjacent data for each expectation step. As a result, the function usually returns a closed curve, i.e. a curve without endpoints. If `deletePoints` is TRUE, this function deletes the points in curves which do not have adjacent data for each expectation step. As a result, The `SPC` function usually returns an open curve, i.e. a curve with endpoints. The default is FALSE.
`plot.proj`	logical value. If the argument is TRUE, the projection line for each data is plotted. The default is FALSE.
`kernel`	kind of kernel function. The default is quartic kernel and alternatives are indicator or Gaussian.

col1	color of data. The default is blue.
col2	color of points in the principal curves. The default is green.
col3	color of resulting principal curves. The default is red.

Details

This function fits a spherical principal curves, and requires to load the 'rgl', 'sphereplot', and 'geosphere' R packages.

Value

plot and a list consisting of

prin.curves	spatial locations (denoted by degrees) of points in the resulting principal curves.
line	connecting line bewteen points of prin.curves.
converged	whether or not the algorithm converged.
iteration	the number of iterations of the algorithm.
recon.error	sum of squared distances from the data to their projections.
num.dist.pt	the number of distinct projections.

Note

This function requires to load 'rgl', 'sphereplot', and 'geosphere' R packages.

Author(s)

Jongmin Lee

References

Jang-Hyun Kim, Jongmin Lee and Hee-Seok Oh. (2020). Spherical Principal Curves <arXiv:2003.02578>.

Jongmin Lee, Jang-Hyun Kim and Hee-Seok Oh. (2021). Spherical principal curves. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43, 2165-2171. <doi.org/10.1109/TPAMI.2020.3025327>.

See Also

SPC.Hauberg.

Examples

```
library(rgl)
library(sphereplot)
library(geosphere)

#### example 2: waveform data
n <- 200
alpha <- 1/3; freq <- 4                       # amplitude and frequency of wave
sigma1 <- 2; sigma2 <- 10                     # noise levels
lon <- seq(-180, 180, length.out = n)         # uniformly sampled longitude
```

```
lat <- alpha * 180/pi * sin(freq * lon * pi/180) + 10.       # latitude vector
## add Gaussian noises on latitude vector
lat1 <- lat + sigma1 * rnorm(length(lon)); lat2 <- lat + sigma2 * rnorm(length(lon))
wave1 <- cbind(lon, lat1); wave2 <- cbind(lon, lat2)
## implement SPC to the (noisy) waveform data
SPC(wave1, q = 0.05)
SPC(wave2, q = 0.05)

#### example 1: earthquake data
data(Earthquake)
names(Earthquake)
earthquake <- cbind(Earthquake$longitude, Earthquake$latitude)
SPC(earthquake, q = 0.1)
## options 1: plot the projection lines (use option of plot.proj = TRUE)
SPC(earthquake, q = 0.1, plot.proj = TRUE)
## option 2: open principal curves (use option of deletePoints = TRUE)
SPC(earthquake, q = 0.04, deletePoints = TRUE)
```

SPC.Hauberg *principal curves by Hauberg on a sphere*

Description

This function fits a principal curve by Hauberg on the unit 2-sphere.

Usage

```
SPC.Hauberg(data, q = 0.1, T = nrow(data), step.size = 1e-3, maxit = 10,
type = "Intrinsic", thres = 1e-2, deletePoints = FALSE, plot.proj = FALSE,
kernel = "quartic", col1 = "blue", col2 = "green", col3 = "red")
```

Arguments

data	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude (denoted by degrees).
q	numeric value of the smoothing parameter. The parameter plays the same role, as the bandwidth does in kernel regression, in the SPC function. The value should be a numeric value between 0.01 and 0.5. The default is 0.1.
T	the number of points making up the resulting curve. The default is 1000.
step.size	step size of the PrincipalCircle function. The default is 0.001. The resulting principal circle is used by an initialization of the SPC.
maxit	maximum number of iterations. The default is 30.
type	type of mean on sphere. The default is "Intrinsic" and the alternative is "extrinsic".
thres	threshold of the stopping condition. The default is 0.01.

`deletePoints`	logical value. The argument is an option of whether to delete points or not. If `deletePoints` is FALSE, this function leaves the points in curves which do not have adjacent data for each expectation step. As a result, the function usually returns a closed curve, i.e. a curve without endpoints. If `deletePoints` is TRUE, this function deletes the points in curves which do not have adjacent data for each expectation step. As a result, The `SPC` function usually returns an open curve, i.e. a curve with endpoints. The default is FALSE.
`plot.proj`	logical value. If the argument is TRUE, the projection line for each data is plotted. The default is FALSE.
`kernel`	kind of kernel function. The default is quartic kernel and alternatives are indicator or Gaussian.
`col1`	color of data. The default is blue.
`col2`	color of points in the principal curves. The default is green.
`col3`	color of resulting principal curves. The default is red.

Details

This function fits a principal curve proposed by Hauberg on the sphere, and requires to load the 'rgl', 'sphereplot', and 'geosphere' R packages.

Value

plot and a list consisting of

`prin.curves`	spatial locations (denoted by degrees) of points in the resulting principal curves.
`line`	connecting line bewteen points of `prin.curves`.
`converged`	whether or not the algorithm converged.
`iteration`	the number of iterations of the algorithm.
`recon.error`	sum of squared distances from the data to their projections.
`num.dist.pt`	the number of distinct projections.
`plot`	plotting of the data and principal curves.

Note

This function requires to load 'rgl', 'sphereplot', and 'geosphere' R packages.

Author(s)

Jongmin Lee

References

Hauberg, S. (2016). Principal curves on Riemannian manifolds. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38, 1915-1921.

Jang-Hyun Kim, Jongmin Lee and Hee-Seok Oh. (2020). Spherical Principal Curves <arXiv:2003.02578>.

Jongmin Lee, Jang-Hyun Kim and Hee-Seok Oh. (2021). Spherical Principal Curves. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43, 2165-2171. <doi.org/10.1109/TPAMI.2020.3025327>.

See Also

SPC, Proj.Hauberg.

Examples

```
library(rgl)
library(sphereplot)
library(geosphere)

#### example 1: earthquake data
data(Earthquake)
names(Earthquake)
earthquake <- cbind(Earthquake$longitude, Earthquake$latitude)
SPC.Hauberg(earthquake, q = 0.1)

#### example 2: waveform data
## longitude and latitude are expressed in degrees
n <- 200
alpha <- 1/3; freq <- 4                         # amplitude and frequency of wave
sigma <- 2                                      # noise level
lon <- seq(-180, 180, length.out = n)           # uniformly sampled longitude
lat <- alpha * 180/pi * sin(freq * lon * pi/180) + 10.       # latitude vector
## add Gaussian noises on latitude vector
lat1 <- lat + sigma * rnorm(length(lon))
wave <- cbind(lon, lat1)
## implement principal curves by Hauberg to the waveform data
SPC.Hauberg(wave, q = 0.05)
```

`Trans.Euclid` *Transforming into Euclidean coordinate*

Description

This function converts a spherical coordinate to a Euclidean coordinate.

Usage

```
Trans.Euclid(vec)
```

Arguments

vec	two-dimensional spherical coordinate.

Details

This function converts a two-dimensional spherical coordinate to a three-dimensional Euclidean coordinate. Longitude should be range from -180 to 180 and latitude from -90 to 90.

Value

three-dimensional vector.

Author(s)

Jongmin Lee

See Also

`Trans.sph`.

Examples

```
Trans.Euclid(c(0, 0))
Trans.Euclid(c(0, 90))
Trans.Euclid(c(90, 0))
Trans.Euclid(c(180, 0))
Trans.Euclid(c(-90, 0))
```

`Trans.sph` *Transforming into spherical coordinate*

Description

This function converts a Euclidean coordinate to a spherical coordinate.

Usage

```
Trans.sph(vec)
```

Arguments

`vec`	three-dimensional Euclidean coordinate.

Details

This function converts a three-dimensional Euclidean coordinate to a two-dimensional spherical coordinate. If `vec` is not in the unit sphere, it is divided by its magnitude so that the result lies on the unit sphere.

Value

two-dimensional vector.

Author(s)

Jongmin Lee

See Also

Trans.Euclid.

Examples

```
Trans.sph(c(1, 0, 0))
Trans.sph(c(0, 1, 0))
Trans.sph(c(0, 0, 1))
Trans.sph(c(-1, 0 , 0))
Trans.sph(c(0, -1, 0))
```

Index