Package 'spherepc'

March 30, 2020

Type Package

Title Spherical Principal Curves

Version 0.1.4

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

Maintainer Jongmin Lee <jongminlee9218@gmail.com>

Description Fitting a principal curve to data lying in the spherical surface.
This package provides principal circle, principal geodesic analysis, Hauberg's
principal curves, and spherical principal curves. Moreover, it offers locally
defined principal geodesics which are currently under study. The detailed procedures
are described in Jang-Hyun Kim, Jongmin Lee and Hee-Seok Oh (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 2020-03-30 13:20:02 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`	A three-dimensional vector.
`vec2`	A 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 point 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.
sphereplot::rgl.sphgrid()                      # plot
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 sphere.

Usage

```
Expmap(vec)
```

Arguments

vec	an element of two-dimensional Euclidean 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 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.
`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, such as a great circle.

Author(s)

Jongmin Lee

References

Jang-Hyun Kim, Jongmin Lee, 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 sphere.

Usage

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

Arguments

center	center of circle.
radius	radius of circle. It should be in [0, pi].
T	the number of points in circle.

Details

This function makes a circle on a 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 the 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.
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.

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	an element of the unit sphere in three-dimensional Euclidean 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,
col = c("blue", "green", "red"), size = c(12, 4, 6))
```

Arguments

data	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude.
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 PGA.
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). 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.
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.
col	three-dimensional vector which represents colors of data, points in the resulting curves, and the connecting line between points in the resulting curves, respectively.
size	three-dimensional vector which represents sizes of data, points in the resulting curves, and the connecting line between points in the resulting curves, respectively.

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 of points in the resulting curves.
`line`	connecting line between points of `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
n <- 500                          # the number of samples.
sigma <- 0.2                      # noise level.
r <- runif(n) * 42
theta <- -pi/10 * r + 2 + sigma * rnorm(n)
x <- r * cos(theta)
y <- r * sin(theta)
simul.spiral <- cbind(x - 90, y)
LPG(simul.spiral, scale = 0.07, nu = 0.1)

#### example 2: zigzag data set
n <- 50                           # the number of samples is 6 * n = 300.
sigma <- 2                        # noise level.
x1 <- runif(n) * 20 - 20
y1 <- x1 + 20 + sigma * rnorm(n)
x2 <- runif(n) * 20 - 20
y2 <- - x2 + 20 + sigma * rnorm(n)
x3 <- runif(n) * 20 - 20
y3 <- x3 + 60 + sigma * rnorm(n)
x4 <- runif(n) * 20 - 20
y4 <- - x4 - 20 + sigma * rnorm(n)
x5 <- runif(n) * 20 - 20
y5 <- x5 - 20 + sigma * rnorm(n)
x6 <- runif(n) * 20 - 20
y6 <- - x6 - 60 + sigma * rnorm(n)
x <- c(x1, x2, x3, x4, x5, x6)
y <- c(y1, y2, y3, y4, y5, y6)
simul.zigzag <- cbind(x, y)
LPG(simul.zigzag, scale = 0.1, nu = 0.1)
```

```
#### example 3: 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 4: tree data
## stem
set.seed(7)
n <- 300                  # the number of samples in stem.
sigma <- 0.1              # noise level of stem.
lat <- 70 * runif(n) - 10
lon <- 0 + sigma * rnorm(n)
stem <- cbind(lon, lat)
## branch
n2 <- 200                 # the number of samples of each branch.
sigma2 <- 0.05            # noise level of branch.
lon <- -20 * runif(n2)
lat <-  -lon + 10 + sigma2 * rnorm(n2)
branch1 <- cbind(lon, lat)
lon <- 20 * runif(n2)
lat <- lon + sigma2 * rnorm(n2)
branch3 <- cbind(lon, lat)
lon <- 20 * runif(n2)
lat <- lon + 20 + sigma2 * rnorm(n2)
branch2 <- cbind(lon, lat)
lon <- 20 * runif(n2)
lat <- lon + 40 + sigma2 * rnorm(n2)
branch5 <- cbind(lon, lat)
lon <- -20 * runif(n2)
lat <- -lon + 30 + sigma2 * rnorm(n2)
branch4 <- cbind(lon, lat)
branch <- rbind(branch1, branch2, branch3, branch4, branch5)
## sub-branches
n3 <- 20                  # the number of samples of each sub-branch.
sigma3 <- 0.01            # noise level of sub-branch.
l <- 1    # length of sub-branches
lon <- l * runif(n3) - 10
lat <- lon + 30 + sigma3 * rnorm(n3)
branches1 <- cbind(lon, lat)
lon <- -l * runif(n3) + 10
lat <- -lon + 20 + sigma3 * rnorm(n3)
branches2 <- cbind(lon, lat)
lon <- -l * runif(n3) + 10
lat <- -lon + 40 + sigma3 * rnorm(n3)
branches17 <- cbind(lon, lat)
lon <- l * runif(n3) - 14
lat <- lon + 38 + sigma3 * rnorm(n3)
branches3 <- cbind(lon, lat)
lon <- -l * runif(n3) + 14
lat <- -lon + 28 + sigma3 * rnorm(n3)
branches4 <- cbind(lon, lat)
```

```
lon <- -l * runif(n3) + 14
lat <- -lon + 48 + sigma3 * rnorm(n3)
branches18 <- cbind(lon, lat)
lon <- -l * runif(n3) - 12
lat <- lon + 34 + sigma3 * rnorm(n3)
branches5 <- cbind(lon, lat)
lon <- l * runif(n3) + 12
lat <- -lon + 24 + sigma3 * rnorm(n3)
branches6 <- cbind(lon, lat)
lon <- l * runif(n3) + 12
lat <- -lon + 44 + sigma3 * rnorm(n3)
branches20 <- cbind(lon, lat)
lon <- -l * runif(n3) - 16
lat <- lon + 42 + sigma3 * rnorm(n3)
branches7 <- cbind(lon, lat)
lon <- l * runif(n3) + 16
lat <- -lon + 32 + sigma3 * rnorm(n3)
branches8 <- cbind(lon, lat)
lon <- l * runif(n3) + 16
lat <- -lon + 52 + sigma3 * rnorm(n3)
branches19 <- cbind(lon, lat)
lon <- l * runif(n3) + 10
lat <- -lon + 60 + sigma3 * rnorm(n3)
branches9 <- cbind(lon, lat)
lon <- -l * runif(n3) - 10
lat <- lon + 50 + sigma3 * rnorm(n3)
branches10 <- cbind(lon, lat)
lon <- -l * runif(n3) + 12
lat <- -lon + 64 + sigma3 * rnorm(n3)
branches11 <- cbind(lon, lat)
lon <- l * runif(n3) - 12
lat <- lon + 54 + sigma3 * rnorm(n3)
branches12 <- cbind(lon, lat)
lon <-  l * runif(n3) + 14
lat <- -lon + 68 + sigma3 * rnorm(n3)
branches13 <- cbind(lon, lat)
lon <- -l * runif(n3) - 14
lat <- lon + 58 + sigma3 * rnorm(n3)
branches14 <- cbind(lon, lat)
lon <- -l * runif(n3) + 16
lat <- -lon + 72 + sigma3 * rnorm(n3)
branches15 <- cbind(lon, lat)
lon <- l * runif(n3) - 16
lat <- lon + 62 + sigma3 * rnorm(n3)
branches16 <- cbind(lon, lat)
sub.branches <- rbind(branches1, branches2, branches3,
branches4,branches5, branches6, branches7, branches8,
branches9, branches10, branches11, branches12,
branches13, branches14, branches15, branches16,
branches17, branches18, branches19, branches20)
tree <- rbind(stem, branch, sub.branches)
LPG(tree, scale = 0.03, nu = 0.1, seed = 7)
```

PGA *Principal geodesic analysis*

Description

This function performs principal geodesic analysis.

Usage

```
PGA(data, col = c("blue", "red"), size = c(12, 6))
```

Arguments

`data`	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude.
`col`	two-dimensional vector which represents colors of data and the principal geodesic line.
`size`	two-dimensional vector which represents sizes of data and the principal geodesic line.

Details

This function performs principal geodesic analysis.

Value

plot and a list consisting of

`line`	spatial locations 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 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 with two columns. Each row represents longitude and latitude.
`step.size`	step size of gradient descent algorithm. For convergence of the algorithm, `step.size` is recommended to be below 0.01.
`thres`	threshold of the stopping condition.
`maxit`	maximum number of iterations.

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

Jang-Hyun Kim, Jongmin Lee, 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.

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 sphere*

Description

Rotate a point on the unit sphere.

Usage

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

Arguments

pt1	a spatial location.
pt2	a 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 coornate 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 sphere*

Description

Rotate a point on the unit sphere.

Usage

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

Arguments

`pt1`	a spatial location.
`pt2`	a 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 coornate 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 = 30,
type = "Intrinsic", thres = 1e-2, deletePoints = FALSE, plot.proj = FALSE,
kernel = "quartic", col = c("blue", "green", "red", "black"), size = c(12, 6, 6))
```

Arguments

data	matrix or data frame consisting of spatial locations with two columns. Each row represents a longitude and latitude.
q	numeric value of the smoothing parameter. Intuitively speaking, the role of this argument is similar to the that of bandwidth for kernel regression. The value should be a numeric value between 0.01 and 0.5. The default is 0.1.
T	the number of points in the resulting curve.
step.size	step size of the PrincipalCircle function. The resulting principal circle is used by an initialization of the SPC.
maxit	maximum number of iterations.
type	type of mean on the sphere. The default is "Intrinsic" and the other choice is "Extrinsic".
thres	threshold of the stopping condition.
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.

col	four-dimensional vector which represents colors of data, points in the resulting curves, the connecting line between points in the resulting curves, and projection lines, respectively.
size	three-dimensional vector which represents sizes of data, points in the principal curves, and the connecting line between points in the curves, respectively.

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 points of 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, Hee-Seok Oh. (2020). Spherical Principal Curves <arXiv:2003.02578>.

See Also

SPC.Hauberg.

Examples

```
library(rgl)
library(sphereplot)
library(geosphere)
#### example 1: earthquake data
data(Earthquake)
names(Earthquake)
earthquake <- cbind(Earthquake$longitude, Earthquake$latitude)
SPC(earthquake, q = 0.1)
#### example 2: waveform data
```

```
n <- 200
alpha <- 1/3    # amplitude
freq <- 4       # frequency
sigma <- 2
lon <- seq(-180, 180, length.out = n)
lat <- alpha * 180/pi * sin(lon * pi/180 * freq) + 10 + sigma * rnorm(length(lon))
wave <- cbind(lon, lat)
SPC(wave, q = 0.05)
```

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

Description

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

Usage

```
SPC.Hauberg(data, q = 0.1, T = nrow(data), step.size = 1e-3, maxit = 30,
type = "Intrinsic", thres = 1e-2, deletePoints = FALSE, plot.proj = FALSE,
kernel = "quartic", col = c("blue", "green", "red", "black"), size = c(12, 6, 6))
```

Arguments

data	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude.
q	numeric value of the smoothing parameter. Intuitively speaking, the role of this argument is similar to the that of bandwidth for kernel regression. The value should be a numeric value between 0.01 and 0.5. The default is 0.1.
T	the number of points in the resulting curve.
step.size	step size of the PrincipalCircle. The resulting principal circle is used by an initialization of the SPC.
maxit	maximum number of iterations.
type	type of mean on sphere. The default is "Intrinsic" and the alternative is "extrinsic".
thres	threshold of the stopping condition.
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.
`col`	four-dimensional vector which represents colors of data, points in the resulting curves, the connecting line between points in the resulting curves, and projection lines, respectively.
`size`	three-dimensional vector which represents sizes of data, points in the principal curves, and the connecting line between points in the curves, respectively.

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 points of 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>.

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
n <- 200
alpha <- 1/3    # amplitude
freq <- 4       # frequency
sigma <- 2
lon <- seq(-180, 180, length.out = n)
lat <- alpha * 180/pi * sin(lon * pi/180 * freq) + 10 + sigma * rnorm(length(lon))
wave <- cbind(lon, lat)
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