Package: frechet (via r-universe)

August 13, 2024

Type Package

Title Statistical Analysis for Random Objects and Non-Euclidean Data

URL https://github.com/functionaldata/tFrechet

BugReports https://github.com/functionaldata/tFrechet/issues

Version 0.2.0

Encoding UTF-8

Date 2020-12-15

Maintainer Yaqing Chen <yaqchen@ucdavis.edu>

Description Provides implementation of statistical methods for random objects lying in various metric spaces, which are not necessarily linear spaces. The core of this package is Fréchet regression for random objects with Euclidean predictors, which allows one to perform regression analysis for non-Euclidean responses under some mild conditions. Examples include distributions in L^2-Wasserstein space, covariance matrices endowed with power metric (with Frobenius metric as a special case), Cholesky and log-Cholesky metrics. References: Petersen, A., & Müller, H.-G. (2019) <doi:10.1214/17-AOS1624>.

License BSD_3_clause + file LICENSE

LazyData false

Imports corrplot, fdadensity, fdapace (>= 0.5.5), Matrix, methods, pracma, osqp, trust

Suggests Rcpp (>= 0.11.5), testthat

RoxygenNote 7.2.3

Repository https://functionaldata.r-universe.dev

RemoteUrl https://github.com/functionaldata/tfrechet

RemoteRef HEAD

RemoteSha f94e092ded81c0b95896c5e4cca0e593a8b1f463

Contents

`AddDensReg` *Additive functional regression for densities as responses*

Description

Smooth backfitting procedure for estimating nonparametric functional additive models for density responses

Usage

```
AddDensReg(Ly, X, x = NULL, hu = NULL, hx = NULL, dSup = NULL)
```

Arguments

`Ly`	A list of n vectors holding random samples independently drawn from each density response
`X`	An n-by-d design matrix whose row vectors consist of regressors for component functions
`x`	A grid matrix whose row vectors consist of evaluation points for component functions. Defaults to the observed design matrix.

hu	A scalar bandwidth for kernel smoothing marginal mean function of the functional additive model
dSup	A 2-dimensional vector that specify the lower and upper limits of the common support for density responses. Defaults to the range of the observed random samples.
huA	d-dimensional vector bandwidth for kernel smoothing each component function

Details

AddDensReg fits additive functional regression models for density responses, where the conditional mean function of a transformed density response is given by the summation of nonparametric univariate functions associated with d covariates, respectively. Instead of having functional responses as infinite-dimensional random objects, AddDensReg has inputs with random samples independently drawn from each random density, and density responses are reconstructed by kernel density estimation. The current version only supports the log-quantile density (LQD) transformation proposed by Petersen and Mueller (2016).

Value

A list holding the following fields:

lqdGrid	A grid where the LQD transformed density responses are evaluated. Defaults to an equally space grid over dSup.
lqdSbfMean	The marginal mean function estimates evaluated on lqdGrid
LlqdSbfComp	A list of d matrices holding the estimates of each component function evaluated on lqdGrid
densGrid	A grid where density responses are evaluated. Defaults to an equally space grid over the range of the observed random samples.
densSbfMean	The density inversion of lqdSbfMean evaluated on densGrid
LdensSbfComp	A list of d matrices holding the density inversion of each component function together with lqdSbfMean evaluated on lqdGrid

References

Han, K., Mueller, H.-G., and Park, B. U. (2020), "Additive functional regression for densities as responses", Journal of the American Statistical Association , 115 (530), pp.997-1010.

Peterson, A. and Mueller, H.-G. (2016), "Functional data analysis for density functions by transformation to a Hilbert space", The Annals of Statistics, 44(1), pp.183-218

See Also

SBFitting in the fdapace package

Examples

```
library(MASS)

# additive component functions
g1 <- function (u, x1) sin(2*pi*u) * (2*x1 - 1)
g2 <- function (u, x2) sin(2*pi*u) * sin(2*pi*x2)

g <- function (u, x) g1(u, x[1]) + g2(u, x[2])

# generating random samples from conditional quantile functions
GenLqdNoise <- function (u, e) e[1]*sin(pi*u) + e[2]*sin(2*pi*u)
GenQdensResp <- function (u, x, e) exp(g(u, x) + GenLqdNoise(u, e))

GenQuantileResp <- function (u, x, e) {

  tmp1 <- integrate(GenQdensResp, lower = 0, upper = u, x = x, e = e)$value
  tmp2 <- integrate(GenQdensResp, lower = 0, upper = 1, x = x, e = e)$value

  return (tmp1 / tmp2)
}

set.seed(999)
n <- 150
N <- 250

Sigma <- matrix(c(1, 0.5, 0.5, 1), nrow = 2, ncol = 2)
X <- pnorm(mvrnorm(n, rep(0, 2), Sigma))

Ly <- list()
for (i in 1:n) {
  U_i <- runif(N)
  E_i <- c(rnorm(1, 0, 0.1), rnorm(1, 0, 0.05))
  Ly[[i]] <- sapply(1:N, function (l) GenQuantileResp(U_i[l], X[i,], E_i))
}

M <- 51
x1 <- x2 <- seq(0, 1, length.out = M)
x <- cbind(x1, x2)

hu <- 0.05
hx <- c(0.075, 0.075)
dSup <- c(0, 1)

# estimating the functional additive model
estAddDensReg <- AddDensReg(Ly = Ly, X = X, x = x, hu = hu, hx = hx, dSup = dSup)

# true LQD component functions
g1Eval <- g2Eval <- matrix(nrow = length(estAddDensReg$lqdGrid), ncol = M)
for (l in seq(estAddDensReg$lqdGrid)) {
  for (m in seq(M)) {
    g1Eval[l,m] <- g1(estAddDensReg$lqdGrid[l], x1[m])
    g2Eval[l,m] <- g2(estAddDensReg$lqdGrid[l], x2[m])
```

```
  }
}

# LQD component function estimates
g0Sbf <- estAddDensReg$lqdSbfMean
gjSbf <- estAddDensReg$LlqdSbfComp

# true density component functions
dens1Eval <- dens2Eval <- matrix(nrow = length(estAddDensReg$densGrid), ncol = M)
for (m in seq(M)) {
  dens1Eval[,m] <- fdadensity::lqd2dens(lqd = g1Eval[,m],
                                        lqdSup = estAddDensReg$lqdGrid,
                                        dSup = estAddDensReg$densGrid)
  dens2Eval[,m] <- fdadensity::lqd2dens(lqd = g2Eval[,m],
                                        lqdSup = estAddDensReg$lqdGrid,
                                        dSup = estAddDensReg$densGrid)
}

# density component function estimates
dens0Sbf <- estAddDensReg$densSbfMean
densjSbf <- estAddDensReg$LdensSbfComp

# graphical illustration of LQD component function estimates
par(mfrow = c(2,2))
par(mar=rep(0.5, 4)+0.1)
persp(estAddDensReg$lqdGrid, x1, g1Eval,
      theta = 35, phi = 35,
      xlab = '\n u', ylab = '\n x1', zlab = '\n LQD component (g1)',
      border = NA, shade = 0.5,
      ticktype = 'detailed')

persp(estAddDensReg$lqdGrid, x2, g2Eval,
      theta = 35, phi = 35,
      xlab = '\n u', ylab = '\n x1', zlab = '\n LQD component  (g2)',
      border = NA, shade = 0.5,
      ticktype = 'detailed')

persp(estAddDensReg$lqdGrid, x1, gjSbf[[1]],
      theta = 35, phi = 35,
      xlab = '\n u', ylab = '\n x1', zlab = '\n SBF estimate (g1)',
      border = NA, shade = 0.5,
      ticktype = 'detailed')

persp(estAddDensReg$lqdGrid, x2, gjSbf[[2]],
      theta = 35, phi = 35,
      xlab = '\n u', ylab = '\n x2', zlab = '\n SBF estimate (g2)',
      border = NA, shade = 0.5,
      ticktype = 'detailed')

# graphical illustration of density component function estimates
par(mfrow = c(2,2))
par(mar=rep(0.5, 4)+0.1)
persp(estAddDensReg$densGrid, x1, dens1Eval,
```

```
      theta = 35, phi = 35,
     xlab = '\n y', ylab = '\n x1', zlab = '\n\n Component density \n (Inversion of g0 + g1)',
      border = NA, shade = 0.5,
      ticktype = 'detailed')

persp(estAddDensReg$densGrid, x2, dens2Eval,
      theta = 35, phi = 35,
     xlab = '\n y', ylab = '\n x1', zlab = '\n\n Component density \n (Inversion of g0 + g2)',
      border = NA, shade = 0.5,
      ticktype = 'detailed')

persp(estAddDensReg$densGrid, x1, densjSbf[[1]],
      theta = 35, phi = 35,
      xlab = '\n y', ylab = '\n x1', zlab = '\n\n SBF estimate \n (Inversion of g0 + g1)',
      border = NA, shade = 0.5,
      ticktype = 'detailed')

persp(estAddDensReg$densGrid, x2, densjSbf[[2]],
      theta = 35, phi = 35,
      xlab = '\n y', ylab = '\n x1', zlab = '\n\n SBF estimate \n (Inversion of g0 + g2)',
      border = NA, shade = 0.5,
      ticktype = 'detailed')

# fitted density responses
fitAddDensReg <- AddDensReg(Ly = Ly, X = X, hu = hu, hx = hx, dSup = dSup)

# fitted LQD component functions
g0SBFit <- fitAddDensReg$lqdSbfMean
gjSBFit <- fitAddDensReg$LlqdSbfComp

# fitted density responses
densSBFit <- lapply(1:n,
                    function (i) {
                      gSBFit <- g0SBFit + gjSBFit[[1]][,i] + gjSBFit[[2]][,i]
                      densSBFit_i <- fdadensity::lqd2dens(lqd = gSBFit,
                                                          lqdSup = fitAddDensReg$lqdGrid,
                                                           dSup = fitAddDensReg$densGrid)
                      return (densSBFit_i)
                    }
)

# graphical illustration of fitted density responses
set.seed(999)
ind <- sample(1:n, 12)
par(mfrow = c(3, 4))
par(mar=c(4, 4, 4, 1)+0.1)
for (i in ind) {
  hist_i <- hist(Ly[[i]], plot = FALSE)
  hist(Ly[[i]], probability = TRUE,
       ylim = range(c(hist_i$density, densSBFit[[i]])),
       xlab = 'Y',
     main = paste(i, '-th random sample \n with X = (', round(X[i,1],2), ', ', round(X[i,2],2), ')', sep = ''))
  lines(fitAddDensReg$densGrid, densSBFit[[i]], col = 2, lwd = 2)
```

```
}
```

---

color.bar *Generate color bar/scale.*

Description

Generate color bar/scale.

Usage

```
color.bar(
  colVal = NULL,
  colBreaks = NULL,
  min = NULL,
  max = NULL,
  lut = NULL,
  nticks = 5,
  ticks = NULL,
  title = NULL
)
```

Arguments

colVal	A numeric vector giving the variable values to which each color is corresponding. It overrides `min` (and `max`) if `min` > `min(colVal)` (`max` < `max(colVal)`).
colBreaks	A numeric vector giving the breaks dividing the range of variable into different colors. It overrides `min` and `max`.
min	A scalar giving the minimum value of the variable represented by colors.
max	A scalar giving the maximum value of the variable represented by colors.
lut	Color vector. Default is `colorRampPalette(colors = c("pink","royalblue"))(length(colBreaks)-1)`.
nticks	An integer giving the number of ticks used in the axis of color bar.
ticks	A numeric vector giving the locations of ticks used in the axis of color bar; it overrides `nticks`.
title	A character giving the label of the variable according to which the color bar is generated.

Value

No return value.

CovFMean *Fréchet mean of covariance matrices*

Description

Fréchet mean computation for covariance matrices.

Usage

```
CovFMean(M = NULL, optns = list())
```

Arguments

M	A q by q by n array (resp. a list of q by q matrices) where `M[,,i]` (resp. `M[[i]]`) contains the i-th covariance matrix of dimension q by q.
optns	A list of options control parameters specified by `list(name=value)`. See 'Details'.

Details

Available control options are

metric Metric type choice, `"frobenius"`, `"power"`, `"log_cholesky"`, `"cholesky"` - default: `"frobenius"` which corresponds to the power metric with `alpha` equal to 1.

alpha The power parameter for the power metric, which can be any non-negative number. Default is 1 which corresponds to Frobenius metric.

weights A vector of weights to compute the weighted barycenter. The length of `weights` is equal to the sample size n. Default is equal weights.

Value

A list containing the following fields:

Mout	A list containing the Fréchet mean of the covariance matrices in `M`.
optns	A list containing the `optns` parameters utilized.

References

- *Petersen, A. and Müller, H.-G. (2019). Fréchet regression for random objects with Euclidean predictors. The Annals of Statistics, 47(2), 691–719.*
- *Petersen, A., Deoni, S. and Müller, H.-G. (2019). Fréchet estimation of time-varying covariance matrices from sparse data, with application to the regional co-evolution of myelination in the developing brain. The Annals of Applied Statistics, 13(1), 393–419.*
- *Lin, Z. (2019). Riemannian geometry of symmetric positive definite matrices via Cholesky decomposition. Siam. J. Matrix. Anal, A. 40, 1353–1370.*

Examples

```
#Example M input
n=10 #sample size
m=5 # dimension of covariance matrices
M <- array(0,c(m,m,n))
for (i in 1:n){
 y0=rnorm(m)
 aux<-diag(m)+y0%*%t(y0)
 M[,,i]<-aux
}
Fmean=CovFMean(M=M,optns=list(metric="frobenius"))
```

`CreateCovRegPlot` *Plots for Fréchet regression for covariance matrices.*

Description

Plots for Fréchet regression for covariance matrices.

Usage

```
CreateCovRegPlot(x, optns = list())
```

Arguments

`x`	A covReg object obtained from `CovFMean`, `GloCovReg` or `LocCovReg`.
`optns`	A list of control options specified by `list(name=value)`. See 'Details'.

Details

Available control options are

ind.xout A vector holding the indices of elements in `x$Mout` at which the plots will be made. Default is
- `1:length(x$Mout)` when `x$Mout` is of length no more than 3;
- `c(1,round(length(x$Mout)/2),length(x$Mout))` when `x$Mout` is of length greater than 3.

nrow An integer — default: 1; subsequent figures will be drawn in an `optns$nrow`-by-`ceiling(length(ind.xout)/optns$nrow)` array.

plot.type Character with two choices, "continuous" and "categorical". The former plots the correlations in a continuous scale of colors by magnitude while the latter categorizes the positive and negative entries into two different colors. Default is "continuous"

plot.clust Character, the ordering method of the correlation matrix. `"original"` for original order (default); `"AOE"` for the angular order of the eigenvectors; `"FPC"` for the first principal component order; `"hclust"` for the hierarchical clustering order, drawing 4 rectangles on the graph according to the hierarchical cluster; `"alphabet"` for alphabetical order.

plot.method Character, the visualization method of correlation matrix to be used. Currently, it supports seven methods, named "circle" (default), "square", "ellipse", "number", "pie", "shade" and "color".

CorrOut Logical, indicating if output is shown as correlation or covariance matrix. Default is `FALSE` and corresponds to a covariance matrix.

plot.display Character, "full" (default), "upper" or "lower", display full matrix, lower triangular or upper triangular matrix.

Value

No return value.

Examples

```
#Example y input
n=20            # sample size
t=seq(0,1,length.out=100)       # length of data
x = matrix(runif(n),n)
theta1 = theta2 = array(0,n)
for(i in 1:n){
 theta1[i] = rnorm(1,x[i],x[i]^2)
 theta2[i] = rnorm(1,x[i]/2,(1-x[i])^2)
}
y = matrix(0,n,length(t))
phi1 = sqrt(3)*t
phi2 = sqrt(6/5)*(1-t/2)
y = theta1%*%t(phi1) + theta2 %*% t(phi2)
xout = matrix(c(0.25,0.5,0.75),3)
Cov_est=GloCovReg(x=x,y=y,xout=xout,optns=list(corrOut = FALSE, metric="power",alpha=3))
CreateCovRegPlot(Cov_est, optns = list(ind.xout = 2, plot.method = "shade"))

CreateCovRegPlot(Cov_est, optns = list(plot.method = "color"))
```

`CreateDensity` *Create density functions from raw data, histogram objects or frequency tables with bins*

Description

Create kernel density estimate along the support of the raw data using the HADES method.

Usage

```
CreateDensity(
  y = NULL,
  histogram = NULL,
  freq = NULL,
```

```
  bin = NULL,
  optns = list()
)
```

Arguments

y	A vector of raw readings.
histogram	A histogram object in R. Use this option when histogram object is only available, but not the raw data y. The default is NULL.
freq	A frequency vector. Use this option when frequency table is only available, but not the raw sample or the histogram object. The corresponding bin should be provided together. The default is NULL.
bin	A bin vector having its length with length(freq)+1. Use this option when frequency table is only available, but not the raw sample or the histogram object. The corresponding freq should be provided together.The default is NULL.
optns	A list of options control parameters specified by list(name=value). See 'Details'.

Details

Available control options are

userBwMu The bandwidth value for the smoothed mean function; positive numeric - default: determine automatically based on the data-driven bandwidth selector proposed by Sheather and Jones (1991)

nRegGrid The number of support points the KDE; numeric - default: 101.

delta The size of the bin to be used; numeric - default: diff(range(y))/1000. It only works when the raw sample is available.

kernel smoothing kernel choice, "rect", "gauss", "epan", "gausvar", "quar" - default: "gauss".

infSupport logical if we expect the distribution to have infinite support or not; logical - default: FALSE.

outputGrid User defined output grid for the support of the KDE, it overrides nRegGrid; numeric - default: NULL.

Value

A list containing the following fields:

bw	The bandwidth used for smoothing.
x	A vector of length nRegGrid with the values of the KDE's support points.
y	A vector of length nRegGrid with the values of the KDE at the support points.

References

- *H.-G. Müller, J.L. Wang and W.B. Capra (1997). "From lifetables to hazard rates: The transformation approach." Biometrika 84, 881–892.*
- *S.J. Sheather and M.C. Jones (1991). "A reliable data-based bandwidth selection method for kernel density estimation." JRSS-B 53, 683–690.*
- *H.-G. Müller, U. Stadtmüller, and T. Schmitt. (1987) "Bandwidth choice and confidence intervals for derivatives of noisy data." Biometrika 74, 743–749.*

Examples

```
### compact support case

# input: raw sample
set.seed(100)
n <- 100
x0 <-seq(0,1,length.out=51)
Y <- rbeta(n,3,2)
f1 <- CreateDensity(y=Y,optns = list(outputGrid=x0))

# input: histogram
histY <- hist(Y)
f2 <- CreateDensity(histogram=histY,optns = list(outputGrid=x0))

# input: frequency table with unequally spaced (random) bins
binY <- c(0,sort(runif(9)),1)
freqY <- c()
for (i in 1:(length(binY)-1)) {
  freqY[i] <- length(which(Y>binY[i] & Y<=binY[i+1]))
}
f3 <- CreateDensity(freq=freqY, bin=binY,optns = list(outputGrid=x0))

# plot
plot(f1$x,f1$y,type='l',col=2,lty=2,lwd=2,
     xlim=c(0,1),ylim=c(0,2),xlab='domain',ylab='density')
points(f2$x,f2$y,type='l',col=3,lty=3,lwd=2)
points(f3$x,f3$y,type='l',col=4,lty=4,lwd=2)
points(x0,dbeta(x0,3,2),type='l',lwd=2)
legend('topleft',
       c('true','raw sample','histogram','frequency table (unequal bin)'),
       col=1:4,lty=1:4,lwd=3,bty='n')

### infinite support case

# input: raw sample
set.seed(100)
n <- 200
x0 <-seq(-3,3,length.out=101)
Y <- rnorm(n)
f1 <- CreateDensity(y=Y,optns = list(outputGrid=x0))

# input: histogram
```

```
histY <- hist(Y)
f2 <- CreateDensity(histogram=histY,optns = list(outputGrid=x0))

# input: frequency table with unequally spaced (random) bins
binY <- c(-3,sort(runif(9,-3,3)),3)
freqY <- c()
for (i in 1:(length(binY)-1)) {
  freqY[i] <- length(which(Y>binY[i] & Y<=binY[i+1]))
}
f3 <- CreateDensity(freq=freqY, bin=binY,optns = list(outputGrid=x0))

# plot
plot(f1$x,f1$y,type='l',col=2,lty=2,lwd=2,
     xlim=c(-3,3),ylim=c(0,0.5),xlab='domain',ylab='density')
points(f2$x,f2$y,type='l',col=3,lty=3,lwd=2)
points(f3$x,f3$y,type='l',col=4,lty=4,lwd=2)
points(x0,dnorm(x0),type='l',lwd=2)
legend('topright',
       c('true','raw sample','histogram','frequency table (unequal bin)'),
       col=1:4,lty=1:4,lwd=3,bty='n')
```

DenFMean *Fréchet means of densities.*

Description

Obtain Fréchet means of densities with respect to L^2-Wasserstein distance.

Usage

```
DenFMean(yin = NULL, hin = NULL, qin = NULL, optns = list())
```

Arguments

`yin`	A matrix or list holding the sample of measurements for the observed distributions. If `yin` is a matrix, each row holds the measurements for one distribution.
`hin`	A list holding the histograms of an observed distribution.
`qin`	A matrix or list holding the quantile functions of the response. If `qin` is a matrix, each row holds the quantile function of an observed distribution taking values on `optns$qSup`. Note that only one of the three `yin`, `hin`, and `qin` needs to be input. If more than one of them are specified, `yin` overwrites `hin`, and `hin` overwrites `qin`.
`optns`	A list of options control parameters specified by `list(name=value)`.

Details

Available control options are qSup, nqSup, bwDen, ndSup, dSup, delta, kernelDen, infSupport, and denLowerThreshold. See LocDenReg for details.

weights A vector of weights to compute the weighted barycenter. The length of weights is equal to the sample size. Default is equal weights.

Value

A list containing the following components:

dout	A numeric vector holding the density of the Fréchet mean.
dSup	A numeric vector giving the domain grid of dout when it is a matrix.
qout	A numeric vector holding the quantile function of the Fréchet mean.
qSup	A numeric vector giving the domain grid of qout.
optns	A list of control options used.

Examples

```
xin = seq(0,1,0.05)
yin = lapply(xin, function(x) {
  rnorm(100, rnorm(1,x + x^2,0.005), 0.05)
})
res <- DenFMean(yin=yin)
plot(res)
```

---

dist4cov *Distance between covariance matrices*

Description

Distance computation between two covariance matrices

Usage

```
dist4cov(A = NULL, B = NULL, optns = list())
```

Arguments

A	an p by p matrix
B	an p by p matrix
optns	A list of options control parameters specified by list(name=value). See 'Details'.

Details

Available control options are

metric Metric type choice, "frobenius", "power", "log_cholesky" and "cholesky" - default: "frobenius", which corresponds to the power metric with alpha equal to 1.

alpha The power parameter for the power metric, which can be any non-negative number. Default is 1 which corresponds to Frobenius metric.

Value

A list containing the following fields:

dist	the distance between covariance matrices A and B.
optns	A list containing the optns parameters utilized.

References

- *Petersen, A. and Müller, H.-G. (2016). Fréchet integration and adaptive metric selection for interpretable covariances of multivariate functional data. Biometrika, 103, 103–120.*
- *Petersen, A. and Müller, H.-G. (2019). Fréchet regression for random objects with Euclidean predictors. The Annals of Statistics, 47(2), 691–719.*
- *Petersen, A., Deoni, S. and Müller, H.-G. (2019). Fréchet estimation of time-varying covariance matrices from sparse data, with application to the regional co-evolution of myelination in the developing brain. The Annals of Applied Statistics, 13(1), 393–419.*

Examples

```
# M input as array
m <- 5 # dimension of covariance matrices
M <- array(0,c(m,m,2))
for (i in 1:2) {
 y0 <- rnorm(m)
 aux <- diag(m) + y0 %*% t(y0)
 M[,,i] <- aux
}
A <- M[,,1]
B <- M[,,2]
frobDist <- dist4cov(A=A, B=B, optns=list(metric="frobenius"))
```

---

dist4den *Lˆ2 Wasserstein distance between two distributions.*

Description

L^2 Wasserstein distance between two distributions.

Usage

```
dist4den(d1 = NULL, d2 = NULL, fctn_type = NULL, optns = list())
```

Arguments

d1, d2	Lists holding the density functions or quantile functions of the two distributions. Each list consists of two numeric vectors x and y of the same length, where x holds the support grid and y holds the values of the function. Note that the type of functions representing the distributions in d1 and d2 should be the same—either both are density functions, or both are quantile functions. If both are quantile functions, all elements in d1$x and d2$x must be between 0 and 1. d1$x and d2$x may have different lengths.
fctn_type	Character vector of length 1 holding the function type in d1 and d2 representing the distributions: "density" (default), "quantile".
optns	A list of control parameters specified by list(name=value).

Details

Available control options are:

nqSup A scalar giving the length of the support grid of quantile functions based on which the L^2 Wasserstein distance (i.e., the L^2 distance between the quantile functions) is computed. Default is 201.

Value

A scalar holding the L^2 Wasserstein distance between d1 and d2.

Examples

```
d1 <- list(x = seq(-6,6,0.01))
d1$y <- dnorm(d1$x)
d2 <- list(x = d1$x + 1)
d2$y <- dnorm(d2$x, mean = 1)
dist <- dist4den(d1 = d1,d2 = d2)
```

---

expSphere — *Compute an exponential map for a unit hypersphere.*

Description

Compute an exponential map for a unit hypersphere.

Usage

```
expSphere(base, tg)
```

Arguments

`base`	A unit vector of length m holding the base point of the tangent space.
`tg`	A vector of length m of which the exponential map is taken.

Value

A unit vector of length m.

`frameSphere` *Generate a "natural" frame (orthonormal basis)*

Description

Generate a "natural" frame (orthonormal basis) for the tangent space at `x` on the unit sphere.

Usage

```
frameSphere(x)
```

Arguments

`x`	A unit vector of length d.

Details

The first $(i+1)$ elements of the ith basis vector are given by $\sin\theta_i \prod_{j=1}^{i-1}\cos\theta_j$, $\sin\theta_i \sin\theta_1 \prod_{j=2}^{i-1}\cos\theta_j$, $\sin\theta_i \sin\theta_2 \prod_{j=3}^{i-1}\cos\theta_j$, $\ldots$, $\sin\theta_i \sin\theta_{i-1}$, $-\cos\theta_i$, respectively. The rest elements (if any) of the ith basis vector are all zero.

Value

A d-by-$(d-1)$ matrix where columns hold the orthonormal basis of the tangent space at `x` on the unit sphere.

Examples

```
frameSphere(c(1,0,0,0))
```

`frechet` *frechet: Statistical Analysis for Random Objects and Non-Euclidean Data*

Description

Provides implementation of statistical methods for random objects lying in various metric spaces, which are not necessarily linear spaces. The core of this package is Fréchet regression for random objects with Euclidean predictors, which allows one to perform regression analysis for non-Euclidean responses under some mild conditions. Examples include distributions in L^2-Wasserstein space, covariance matrices endowed with power metric (with Frobenius metric as a special case), Cholesky and log-Cholesky metrics. References: Petersen, A., & Müller, H.-G. (2019) <doi:10.1214/17-AOS1624>.

`GloCovReg` *Global Fréchet regression of covariance matrices*

Description

Global Fréchet regression of covariance matrices with Euclidean predictors.

Usage

```
GloCovReg(x, y = NULL, M = NULL, xout, optns = list())
```

Arguments

`x`	An n by p matrix of predictors.
`y`	An n by l matrix, each row corresponds to an observation, l is the length of time points where the responses are observed. See 'metric' option in 'Details' for more details.
`M`	A q by q by n array (resp. a list of q by q matrices) where `M[,,i]` (resp. `M[[i]]`) contains the i-th covariance matrix of dimension q by q. See 'metric' option in 'Details' for more details.
`xout`	An m by p matrix of output predictor levels.
`optns`	A list of options control parameters specified by `list(name=value)`. See 'Details'.

Details

Available control options are

corrOut Boolean indicating if output is shown as correlation or covariance matrix. Default is `FALSE` and corresponds to a covariance matrix.

metric Metric type choice, `"frobenius"`, `"power"`, `"log_cholesky"`, `"cholesky"` - default: `"frobenius"` which corresponds to the power metric with `alpha` equal to 1. For power (and Frobenius) metrics, either `y` or `M` must be input; `y` would override `M`. For Cholesky and log-Cholesky metrics, `M` must be input and `y` does not apply.

alpha The power parameter for the power metric. Default is 1 which corresponds to Frobenius metric.

Value

A `covReg` object — a list containing the following fields:

`xout`	An m by p matrix of output predictor levels.
`Mout`	A list of estimated conditional covariance or correlation matrices at `xout`.
`optns`	A list containing the `optns` parameters utilized.

References

- *Petersen, A. and Müller, H.-G. (2019). Fréchet regression for random objects with Euclidean predictors. The Annals of Statistics, 47(2), 691–719.*
- *Petersen, A., Deoni, S. and Müller, H.-G. (2019). Fréchet estimation of time-varying covariance matrices from sparse data, with application to the regional co-evolution of myelination in the developing brain. The Annals of Applied Statistics, 13(1), 393–419.*
- *Lin, Z. (2019). Riemannian geometry of symmetric positive definite matrices via Cholesky decomposition. Siam. J. Matrix. Anal, A. 40, 1353–1370.*

Examples

```
#Example y input
n=50             # sample size
t=seq(0,1,length.out=100)       # length of data
x = matrix(runif(n),n)
theta1 = theta2 = array(0,n)
for(i in 1:n){
 theta1[i] = rnorm(1,x[i],x[i]^2)
 theta2[i] = rnorm(1,x[i]/2,(1-x[i])^2)
}
y = matrix(0,n,length(t))
phi1 = sqrt(3)*t
phi2 = sqrt(6/5)*(1-t/2)
y = theta1%*%t(phi1) + theta2 %*% t(phi2)
xout = matrix(c(0.25,0.5,0.75),3)
Cov_est=GloCovReg(x=x,y=y,xout=xout,optns=list(corrOut=FALSE,metric="power",alpha=3))
#Example M input
n=10 #sample size
```

```
m=5 # dimension of covariance matrices
M <- array(0,c(m,m,n))
for (i in 1:n){
 y0=rnorm(m)
 aux<-diag(m)+y0%*%t(y0)
 M[,,i]<-aux
}
x=cbind(matrix(rnorm(n),n),matrix(rnorm(n),n)) #vector of predictor values
xout=cbind(runif(3),runif(3)) #output predictor levels
Cov_est=GloCovReg(x=x,M=M,xout=xout,optns=list(corrOut=FALSE,metric="power",alpha=3))
```

GloDenReg *Global density regression.*

Description

Global Fréchet regression for densities with respect to L^2-Wasserstein distance.

Usage

```
GloDenReg(
  xin = NULL,
  yin = NULL,
  hin = NULL,
  qin = NULL,
  xout = NULL,
  optns = list()
)
```

Arguments

xin	An n by p matrix or a vector of length n (if p=1) with input measurements of the predictors.
yin	A matrix or list holding the sample of observations of the response. If `yin` is a matrix, each row holds the observations of the response corresponding to a row in `xin`.
hin	A list holding the histograms of the response corresponding to each row in `xin`.
qin	A matrix or list holding the quantile functions of the response. If `qin` is a matrix, each row holds the quantile function of the response taking values on `optns$qSup` corresponding to a row in `xin`. Note that only one of the three `yin`, `hin`, and `qin` needs to be input. If more than one of them are specified, `yin` overwrites `hin`, and `hin` overwrites `qin`.
xout	A k by p matrix or a vector of length k (if p=1) with output measurements of the predictors. Default is `xin`.
optns	A list of control parameters specified by `list(name=value)`.

Details

Available control options are qSup, nqSup, lower, upper, Rsquared, bwDen, ndSup, dSup, delta, kernelDen, infSupport, and denLowerThreshold. Rsquared is explained as follows and see LocDenReg for the other options.

Rsquared A logical variable indicating whether R squared would be returned. Default is FALSE.

Value

A list containing the following components:

xout	Input xout.
dout	A matrix or list holding the output densities corresponding to xout. If dout is a matrix, each row gives a density and the domain grid is given in dSup. If dout is a list, each element is a list of two components, x and y, giving the domain grid and density function values, respectively.
dSup	A numeric vector giving the domain grid of dout when it is a matrix.
qout	A matrix holding the quantile functions of the output densities. Each row corresponds to a value in xout.
qSup	A numeric vector giving the domain grid of qout.
xin	Input xin.
din	Densities corresponding to the input yin, hin or qin.
qin	Quantile functions corresponding to the input yin, hin or qin.
Rsq	A scalar giving the R squared value if optns$Rsquared = TRUE.
optns	A list of control options used.

References

Petersen, A., & Müller, H.-G. (2019). "Fréchet regression for random objects with Euclidean predictors." The Annals of Statistics, 47(2), 691–719.

Examples

```
xin = seq(0,1,0.05)
yin = lapply(xin, function(x) {
  rnorm(100, rnorm(1,x,0.005), 0.05)
})
qSup = seq(0,1,0.02)
xout = seq(0,1,0.25)
res1 <- GloDenReg(xin=xin, yin=yin, xout=xout, optns = list(qSup = qSup))
plot(res1)

hin = lapply(yin, function(y) hist(y, breaks = 50, plot=FALSE))
res2 <- GloDenReg(xin=xin, hin=hin, xout=xout, optns = list(qSup = qSup))
plot(res2)
```

GloSpheReg — *Global Fréchet Regression for Spherical Data*

Description

Global Fréchet regression for spherical data with respect to the geodesic distance.

Usage

```
GloSpheReg(xin = NULL, yin = NULL, xout = NULL)
```

Arguments

xin	A vector of length n or an n-by-p matrix with input measurement points.
yin	An n-by-m matrix holding the spherical data, of which the sum of squares of elements within each row is 1.
xout	A vector of length k or an k-by-p with output measurement points; Default: the same grid as given in `xin`.

Value

A list containing the following components:

xout	Input `xout`.
yout	A k-by-m matrix holding the fitted responses, of which each row is a spherical vector, corresponding to each element in `xout`.
xin	Input `xin`.
yin	Input `yin`.

References

Petersen, A., & Müller, H.-G. (2019). "Fréchet regression for random objects with Euclidean predictors." The Annals of Statistics, 47(2), 691–719.

Examples

```
n <- 101
xin <- seq(-1,1,length.out = n)
theta_true <- rep(pi/2,n)
phi_true <- (xin + 1) * pi / 4
ytrue <- apply( cbind( 1, phi_true, theta_true ), 1, pol2car )
yin <- t( ytrue )
xout <- xin
res <- GloSpheReg(xin=xin, yin=yin, xout=xout)
```

LocCovReg *Local Fréchet regression of covariance matrices*

Description

Local Fréchet regression of covariance matrices with Euclidean predictors.

Usage

```
LocCovReg(x, y = NULL, M = NULL, xout, optns = list())
```

Arguments

x	An n by p matrix of predictors.
y	An n by l matrix, each row corresponds to an observation, l is the length of time points where the responses are observed. See 'metric' option in 'Details' for more details.
M	A q by q by n array (resp. a list of q by q matrices) where `M[,,i]` (resp. `M[[i]]`) contains the i-th covariance matrix of dimension q by q. See 'metric' option in 'Details' for more details.
xout	An m by p matrix of output predictor levels.
optns	A list of options control parameters specified by `list(name=value)`. See 'Details'.

Details

Available control options are

corrOut Boolean indicating if output is shown as correlation or covariance matrix. Default is `FALSE` and corresponds to a covariance matrix.

metric Metric type choice, `"frobenius"`, `"power"`, `"log_cholesky"`, `"cholesky"` - default: `"frobenius"` which corresponds to the power metric with `alpha` equal to 1. For power (and Frobenius) metrics, either `y` or `M` must be input; `y` would override `M`. For Cholesky and log-Cholesky metrics, `M` must be input and `y` does not apply.

alpha The power parameter for the power metric. Default is 1 which corresponds to Frobenius metric.

bwMean A vector of length p holding the bandwidths for conditional mean estimation if `y` is provided. If `bwMean` is not provided, it is chosen by cross validation.

bwCov A vector of length p holding the bandwidths for conditional covariance estimation. If `bwCov` is not provided, it is chosen by cross validation.

kernel Name of the kernel function to be chosen from `"rect"`, `"gauss"`, `"epan"`, `"gausvar"`, `"quar"`. Default is `"gauss"`.

Value

A covReg object — a list containing the following fields:

xout An m by p matrix of output predictor levels.

Mout A list of estimated conditional covariance or correlation matrices at xout.

optns A list containing the optns parameters utilized.

References

- *Petersen, A. and Müller, H.-G. (2019). Fréchet regression for random objects with Euclidean predictors. The Annals of Statistics, 47(2), 691–719.*
- *Petersen, A., Deoni, S. and Müller, H.-G. (2019). Fréchet estimation of time-varying covariance matrices from sparse data, with application to the regional co-evolution of myelination in the developing brain. The Annals of Applied Statistics, 13(1), 393–419.*
- *Lin, Z. (2019). Riemannian geometry of symmetric positive definite matrices via Cholesky decomposition. Siam. J. Matrix. Anal, A. 40, 1353–1370.*

Examples

```
#Example y input
n=30             # sample size
t=seq(0,1,length.out=100)       # length of data
x = matrix(runif(n),n)
theta1 = theta2 = array(0,n)
for(i in 1:n){
 theta1[i] = rnorm(1,x[i],x[i]^2)
 theta2[i] = rnorm(1,x[i]/2,(1-x[i])^2)
}
y = matrix(0,n,length(t))
phi1 = sqrt(3)*t
phi2 = sqrt(6/5)*(1-t/2)
y = theta1%*%t(phi1) + theta2 %*% t(phi2)
xout = matrix(c(0.25,0.5,0.75),3)
Cov_est=LocCovReg(x=x,y=y,xout=xout,optns=list(corrOut=FALSE,metric="power",alpha=3))

#Example M input
n=30 #sample size
m=30 #dimension of covariance matrices
M <- array(0,c(m,m,n))
for (i in 1:n){
 y0=rnorm(m)
 aux<-15*diag(m)+y0%*%t(y0)
 M[,,i]<-aux
}
x=matrix(rnorm(n),n)
xout = matrix(c(0.25,0.5,0.75),3) #output predictor levels
Cov_est=LocCovReg(x=x,M=M,xout=xout,optns=list(corrOut=FALSE,metric="power",alpha=0))
```

`LocDenReg` *Local density regression.*

Description

Local Fréchet regression for densities with respect to L^2-Wasserstein distance.

Usage

```
LocDenReg(
  xin = NULL,
  yin = NULL,
  hin = NULL,
  qin = NULL,
  xout = NULL,
  optns = list()
)
```

Arguments

`xin`	An n by p matrix or a vector of length n if p=1 holding the n observations of the predictor.
`yin`	A matrix or list holding the sample of observations of the response. If `yin` is a matrix, each row holds the observations of the response corresponding to a predictor value in the corresponding row of `xin`.
`hin`	A list holding the histograms of the response corresponding to each predictor value in the corresponding row of `xin`.
`qin`	A matrix or list holding the quantile functions of the response. If `qin` is a matrix, the support of the quantile functions should be the same (i.e., `optns$qSup`), and each row of `qin` holds the quantile function corresponding to a predictor value in the corresponding row of `xin`. If the quantile functions are evaluated on different grids, then `qin` should be a list, each element consisting of two components `x` and `y` holding the support grid and the corresponding values of the quantile functions, respectively. Note that only one of the three `yin`, `hin`, and `qin` needs to be input. If more than one of them are specified, `yin` overwrites `hin`, and `hin` overwrites `qin`.
`xout`	An m by p matrix or a vector of length m if p=1 holding the m output predictor values. Default is `xin`.
`optns`	A list of control parameters specified by `list(name=value)`. See 'Details'.

Details

Available control options are

bwReg A vector of length p used as the bandwidth for the Fréchet regression or `"CV"` (default), i.e., a data-adaptive selection done by cross-validation.

kernelReg A character holding the type of kernel functions for local Fréchet regression for densities; "rect", "gauss", "epan", "gausvar", "quar" - default: "gauss".

qSup A numeric vector holding the grid on [0,1] quantile functions take value on. Default is an equidistant grid.

nqSup A scalar giving the length of qSup. Default is 201.

lower A scalar with the lower bound of the support of the distribution. Default is NULL.

upper A scalar with the upper bound of the support of the distribution. Default is NULL.

bwRange A 2 by p matrix whose columns contain the bandwidth selection range for each corresponding dimension of the predictor xin for the case when bwReg equals "CV". Default is NULL and is automatically chosen by a data-adaptive method.

bwDen The bandwidth value used in CreateDensity() for density estimation; positive numeric - default: determine automatically based on the data-driven bandwidth selector proposed by Sheather and Jones (1991).

ndSup The number of support points the kernel density estimation uses in CreateDensity(); numeric - default: 101.

dSup User defined output grid for the support of kernel density estimation used in CreateDensity(), it overrides nRegGrid; numeric - default: NULL

delta The size of the bin to be used used in CreateDensity(); numeric - default: diff(range(y))/1000. It only works when the raw sample is available.

kernelDen A character holding the type of kernel functions used in CreateDensity() for density estimation; "rect", "gauss", "epan", "gausvar", "quar" - default: "gauss".

infSupport logical if we expect the distribution to have infinite support or not, used in CreateDensity() for density estimation; logical - default: FALSE

denLowerThreshold FALSE or a positive value giving the lower threshold of the densities used in CreateDensity(); default: 0.001 * mean(qin[,ncol(qin)] - qin[,1]).

Value

A list containing the following components:

xout	Input xout.
dout	A matrix or list holding the output densities corresponding to xout. If dout is a matrix, each row gives a density and the domain grid is given in dSup. If dout is a list, each element is a list of two components, x and y, giving the domain grid and density function values, respectively.
dSup	A numeric vector giving the domain grid of dout when it is a matrix.
qout	A matrix holding the quantile functions of the output densities. Each row corresponds to a value in xout.
qSup	A numeric vector giving the domain grid of qout.
xin	Input xin.
din	Densities corresponding to the input yin, hin or qin.
qin	Quantile functions corresponding to the input yin, hin or qin.
optns	A list of control options used.

References

Petersen, A., & Müller, H.-G. (2019). "Fréchet regression for random objects with Euclidean predictors." The Annals of Statistics, 47(2), 691–719.

Examples

```
xin = seq(0,1,0.05)
yin = lapply(xin, function(x) {
  rnorm(100, rnorm(1,x + x^2,0.005), 0.05)
})
qSup = seq(0,1,0.02)
xout = seq(0,1,0.1)
res1 <- LocDenReg(xin=xin, yin=yin, xout=xout, optns = list(bwReg = 0.12, qSup = qSup))
plot(res1)

xout <- xin
hin = lapply(yin, function(y) hist(y, breaks = 50))
res2 <- LocDenReg(xin=xin, hin=hin, xout=xout, optns = list(qSup = qSup))
plot(res2)
```

LocSpheReg — *Local Fréchet Regression for Spherical Data*

Description

Local Fréchet regression for spherical data with respect to the geodesic distance.

Usage

```
LocSpheReg(xin = NULL, yin = NULL, xout = NULL, optns = list())
```

Arguments

`xin`	A vector of length n with input measurement points.
`yin`	An n by m matrix holding the spherical data, of which the sum of squares of elements within each row is 1.
`xout`	A vector of length k with output measurement points; Default: `xout = xin`.
`optns`	A list of options control parameters specified by `list(name=value)`. See 'Details'.

Details

Available control options are

bw A scalar used as the bandwidth or `"CV"` (default).

kernel A character holding the type of kernel functions for local Fréchet regression for densities; `"rect"`, `"gauss"`, `"epan"`, `"gausvar"`, `"quar"` - default: `"gauss"`.

Value

A list containing the following components:

`xout`	Input `xout`.
`yout`	A k by m matrix holding the fitted responses, of which each row is a spherical vector, corresponding to each element in `xout`.
`xin`	Input `xin`.
`yin`	Input `yin`.
`optns`	A list of control options used.

References

Petersen, A., & Müller, H.-G. (2019). "Fréchet regression for random objects with Euclidean predictors." The Annals of Statistics, 47(2), 691–719.

Examples

```
set.seed(1)
n <- 200
# simulate the data according to the simulation in Petersen & Müller (2019)
xin <- runif(n)
err_sd <- 0.2
xout <- seq(0,1,length.out = 51)

phi_true <- acos(xin)
theta_true <- pi * xin
ytrue <- cbind(
  sin(phi_true) * cos(theta_true),
  sin(phi_true) * sin(theta_true),
  cos(phi_true)
)
basis <- list(
  b1 = cbind(
    cos(phi_true) * cos(theta_true),
    cos(phi_true) * sin(theta_true),
    -sin(phi_true)
  ),
  b2 = cbind(
    sin(theta_true),
    -cos(theta_true),
    0
  )
)
yin_tg <- basis$b1 * rnorm(n, mean = 0, sd = err_sd) +
  basis$b2 * rnorm(n, mean = 0, sd = err_sd)
yin <- t(sapply(seq_len(n), function(i) {
  tgNorm <- sqrt(sum(yin_tg[i,]^2))
  if (tgNorm < 1e-10) {
    return(ytrue[i,])
  } else {
```

```
    return(sin(tgNorm) * yin_tg[i,] / tgNorm + cos(tgNorm) * ytrue[i,])
  }
}))

res <- LocSpheReg(xin=xin, yin=yin, xout=xout, optns = list(bw = 0.15, kernel = "epan"))
```

`logSphere` *Compute a log map for a unit hypersphere.*

Description

Compute a log map for a unit hypersphere.

Usage

```
logSphere(base, x)
```

Arguments

`base`	A unit vector of length m holding the base point of the tangent space.
`x`	A unit vector of length m which the log map is taken.

Value

A tangent vector of length m.

`plot.denReg` *Plots for Fréchet regression for univariate densities.*

Description

Plots for Fréchet regression for univariate densities.

Usage

```
## S3 method for class 'denReg'
plot(
  x,
  obj = NULL,
  prob = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  ylim = NULL,
  xlim = NULL,
  col.bar = TRUE,
```

```
    widrt = 4,
    col.lab = NULL,
    nticks = 5,
    ticks = NULL,
    add = FALSE,
    pos.prob = 0.9,
    colPalette = NULL,
    ...
)
```

Arguments

x	A denReg object, result of DenFMean, GloDenReg or LocDenReg.
obj	An integer indicating which output to be plotted; 1, 2, 3, 4, and 5 for dout, qout, din, qin, and reference chart for qout, respectively - default: 1.
prob	A vector specifying the probability levels for reference chart if obj is set to 5. Default: c(0.05,0.25,0.5,0.75,0.95).
xlab	Character holding the label for x-axis; default: "Probability" when obj is 2 or 4, "" when obj is 1 or 3, "x" when obj is 5.
ylab	Character holding the label for y-axis; default: "Quantile" when obj is 2, 4, or 5, and "Density" when obj is 1 or 3.
main	Character holding the plot title; default: NULL.
ylim	A numeric vector of length 2 holding the range of the y-axis to be drawn; default: automatically determined by the input x.
xlim	A numeric vector of length 2 holding the range of the x-axis to be drawn; default: automatically determined by the input x.
col.bar	A logical variable indicating whether a color bar is presented on the right of the plot - default: TRUE.
widrt	A scalar giving the width ratio between the main plot and the color bar - default: 4.
col.lab	A character giving the color bar label.
nticks	An integer giving the number of ticks used in the axis of color bar.
ticks	A numeric vector giving the locations of ticks used in the axis of color bar; it overrides nticks.
add	Logical; only works when obj is 5. If TRUE add to an already existing plot. Taken as FALSE (with a warning if a different value is supplied) if no graphics device is open.
pos.prob	FALSE or a scalar less than 0 or larger than 1. FALSE: no probability levels will be labeled on the quantile curves; a scalar between 0 and 1: indicating where to put the probability levels along the curves on growth charts: 0 and 1 correspond to left and right ends, respectively. Default: 0.9.
colPalette	A function that takes an integer argument (the required number of colors) and returns a character vector of colors interpolating the given sequence (e.g., heat.colors, terrain.colors and functions created by colorRampPalette). Default is colorRampPalette(colors = c("pink","royalblue")) for more than one curves and "black" otherwise.

... Can set up lty, lwd, etc.

Value

No return value.

Note

see DenFMean, GloDenReg and LocDenReg for example code.

---

pol2car *Transform polar to Cartesian coordinates*

Description

Transform polar to Cartesian coordinates

Usage

```
pol2car(p)
```

Arguments

p	A vector of length d ($d \geq 2$) with the first element being the radius and the others being the angles, where p[2] takes values in $[0, 2\pi]$ and p[i] takes values in $[-\pi/2, \pi/2]$, for all $i > 2$ if any.

Value

A vector of length d holding the corresponding Cartesian coordinates

$$\left(r \prod_{i=1}^{d-1} \cos\theta_i, r \sin\theta_1 \prod_{i=2}^{d-1} \cos\theta_i, r \sin\theta_2 \prod_{i=3}^{d-1} \cos\theta_i, \ldots, r \sin\theta_{d-2} \cos\theta_{d-1}, r \sin\theta_{d-1} \right),$$

where r is given by p[1] and θ_i is given by p[i+1] for $i = 1, \ldots, d-1$.

Examples

```
pol2car(c(1, 0, pi/4)) # should equal c(1,0,1)/sqrt(2)
pol2car(c(1, pi, 0)) # should equal c(-1,0,0)
```

`SpheGeoDist` *Geodesic distance on spheres.*

Description

Geodesic distance on spheres.

Usage

```
SpheGeoDist(y1, y2)
```

Arguments

y1, y2	Two unit vectors, i.e., with L^2 norm equal to 1, of the same length.

Value

A scalar holding the geodesic distance between y1 and y2.

Examples

```
d <- 3
y1 <- rnorm(d)
y1 <- y1 / sqrt(sum(y1^2))
y2 <- rnorm(d)
y2 <- y2 / sqrt(sum(y2^2))
dist <- SpheGeoDist(y1,y2)
```

`SpheGeoGrad` *Compute gradient w.r.t. y of the geodesic distance* $\arccos(x\hat{}\top y)$ *on a unit hypersphere*

Description

Compute gradient w.r.t. y of the geodesic distance $\arccos(x^\top y)$ on a unit hypersphere

Usage

```
SpheGeoGrad(x, y)
```

Arguments

x, y	Two unit vectors.

Value

A vector holding radient w.r.t. y of the geodesic distance between x and y.

`SpheGeoHess` *Hessian $\partial\hat{}2/\partial y\partial y\hat{}\top$ of the geodesic distance $\arccos(x\hat{}\top y)$ on a unit hypersphere*

Description

Hessian $\partial^2/\partial y\partial y^\top$ of the geodesic distance $\arccos(x^\top y)$ on a unit hypersphere

Usage

```
SpheGeoHess(x, y)
```

Arguments

`x, y`	Two unit vectors.

Value

A Hessian matrix.

Index