Package 'frechet'

December 9, 2023

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 0.3.0

Encoding UTF-8

Date 2023-12-07

Language en-US

Maintainer Yaqing Chen <yqchen@stat.rutgers.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 2-Wasserstein space, covariance matrices endowed with power metric (with Frobenius metric as a special case), Cholesky and log-Cholesky metrics, spherical data. References: Petersen, A., & Müller, H.-G. (2019) <doi:10.1214/17-AOS1624>.

License BSD_3_clause + file LICENSE

LazyData false

Imports boot, corrplot, e1071, fdadensity, fdapace (>= 0.5.5), Matrix, methods, pracma, quadprog, osqp, stats, trust, utils

Suggests Rcpp (>= 0.11.5), testthat, igraph, mpoly, truncnorm

RoxygenNote 7.1.0

NeedsCompilation no

Author Yaqing Chen [aut, cre],
Yidong Zhou [aut],
Han Chen [aut],
Alvaro Gajardo [aut],
Jianing Fan [aut],

Qixian Zhong [aut],
Paromita Dubey [aut],
Kyunghee Han [aut],
Satarupa Bhattacharjee [aut],
Changbo Zhu [ctb],
Su I Iao [ctb],
Poorbita Kundu [ctb],
Petersen Alexander [aut],
Hans-Georg Müller [cph, ths, aut]

Repository CRAN

Date/Publication 2023-12-09 15:50:08 UTC

R topics documented:

`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.

CovFIntegral *Generalized Fréchet integrals of covariance matrix*

Description

Calculating generalized Fréchet integrals of covariance (equipped with Frobenius norm)

Usage

```
CovFIntegral(phi, t_out, X)
```

Arguments

phi	An eigenfunction along which we want to project the network
t_out	Support of phi
X	A three dimensional array of dimension length(t_out) x m x m, where X[i,,] is an m x m covariance matrix.

Value

A list of the following:

f	An adjacency matrix which corresponds to the Fréchet integral of X along phi

References

Dubey, P., & Müller, H. G. (2020). Functional models for time-varying random objects. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 82(2), 275-327.

Examples

```
set.seed(5)
library(mpoly)
n <- 100
N <- 50
t_out <- seq(0,1,length.out = N)

p2 <- as.function(mpoly::jacobi(2,4,3),silent=TRUE)
p4 <- as.function(mpoly::jacobi(4,4,3),silent=TRUE)
p6 <- as.function(mpoly::jacobi(6,4,3),silent=TRUE)

# first three eigenfunctions
phi1 <- function(t){
 p2(2*t-1)*t^(1.5)*(1-t)^2/(integrate(function(x) p2(2*x-1)^2*x^(3)*(1-x)^4,0,1))$value^(1/2)
}
phi2 <- function(t){
 p4(2*t-1)*t^(1.5)*(1-t)^2/(integrate(function(x) p4(2*x-1)^2*x^(3)*(1-x)^4,0,1))$value^(1/2)
}
```

```
phi3 <- function(t){
 p6(2*t-1)*t^(1.5)*(1-t)^2/(integrate(function(x) p6(2*x-1)^2*x^(3)*(1-x)^4,0,1))$value^(1/2)
}

# random component of covariance matrices
P12 <- 0 ## elements between communities
Score <- matrix(runif(n*4), nrow = n)
# first community
P1_cov <- 0.5 + 0.4*Score[,1] %*% t(phi1(t_out)) + 0.1*Score[,2] %*% t(phi3(t_out))
# second community
P2_cov <- 0.5 + 0.3*Score[,3] %*% t(phi2(t_out)) + 0.1*Score[,4] %*% t(phi3(t_out))
P1_diag <- 2 #diagonal elements of the first community
P2_diag <- 3 #diagonal elements of the second community

# create Network edge matrix
N_net1 <- 5 # first community number
N_net2 <- 5 # second community number

# I: four dimension array of n x n matrix of squared distances between the time point u
# of the ith process and process and the time point v of the jth object process,
# e.g.: I[i,j,u,v] <- d_F^2(X_i(u) X_j(v)).
I <- array(0, dim = c(n,n,N,N))
for(u in 1:N){
  for(v in 1:N){
    #frobenius norm between two adjcent matrix
   I[,,u,v] <- outer(P1_cov[,u], P1_cov[,v], function(a1, a2) (a1-a2)^2*(N_net1^2-N_net1)) +
      outer(P2_cov[,u], P2_cov[,v], function(a1, a2) (a1-a2)^2*(N_net2^2-N_net2))
  }
}

# check ObjCov work
Cov_result <- ObjCov(t_out, I, 3, smooth=FALSE)
Cov_result$lambda  # 0.266 0.15 0.04

# e.g. subj 2
subj <- 2
# X_mat is the network for varying times with X[i,,]
# is the adjacency matrices for the ith time point
X_mat <- array(0, c(N,(N_net1+N_net2), (N_net1+N_net2)))
for(i in 1:N){
  # edge between communities is P12
  Mat <- matrix(P12, nrow = (N_net1+N_net2), ncol = (N_net1+N_net2))
  # edge within the first communitiy is P1
  Mat[1:N_net1, 1:N_net1] <- P1_cov[subj, i]
  # edge within the second community is P2
  Mat[(N_net1+1):(N_net1+N_net2), (N_net1+1):(N_net1+N_net2)] <- P2_cov[subj, i]
  diag(Mat) <- c(rep(P1_diag,N_net1),rep(P2_diag, N_net2)) #diagonal element is 0
  X_mat[i,,] <- Mat
}
# output the functional principal network(adjacency matrice) of the second eigenfunction
CovFIntegral(Cov_result$phi[,2], t_out, X_mat)
```

`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')
```

`DenANOVA` *Fréchet ANOVA for Densities*

Description

Fréchet analysis of variance for densities with respect to L^2-Wasserstein distance.

Usage

```
DenANOVA(
  yin = NULL,
  hin = NULL,
  din = NULL,
  qin = NULL,
  supin = NULL,
  group = NULL,
  optns = list()
)
```

Arguments

`yin`	A matrix or data frame or list holding the sample of measurements for the observed distributions. If `yin` is a matrix or data frame, each row holds the measurements for one distribution.
`hin`	A list holding the histograms for the observed distributions.

din	A matrix or data frame or list holding the density functions. If din is a matrix or data frame, each row of din holds the density function for one distribution.
qin	A matrix or data frame or list holding the quantile functions. If qin is a matrix or data frame, each row of qin holds the quantile function for one distribution. Note that the input can be only one of the four yin, hin, din, and qin. If more than one of them are specified, yin overwrites hin, hin overwrites din, and din overwrites qin.
supin	A matrix or data frame or list holding the support grids of the density functions in din or the quantile functions in qin. If supin is a matrix or data frame, each row of supin holds the support grid of the corresponding density function or quantile function. Ignored if the input is yin or hin. It can also be a vector if all density functions in din or all quantile functions in qin have the same support grid.
group	A vector containing the group memberships of the corresponding observed distributions in yin or hin or din or qin.
optns	A list of control parameters specified by list(name = value). See 'Details'.

Details

Available control options are

boot Logical, also compute bootstrap p-value if TRUE. Default is FALSE.

R The number of bootstrap replicates. Only used when boot is TRUE. Default is 1000.

nqSup A scalar giving the number of the support points for 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.

qSup A numeric vector holding the support grid on [0, 1] based on which the L^2 Wasserstein distance (i.e., the L^2 distance between the quantile functions) is computed. It overrides nqSup.

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 A scalar giving the number of support points the kernel density estimation used in CreateDensity(); numeric - default: 101.

dSup User defined output grid for the support of kernel density estimation used in CreateDensity(), it overrides ndSup.

delta A scalar giving 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 DenANOVA object — a list containing the following fields:

pvalAsy	a scalar holding the asymptotic p-value.
pvalBoot	a scalar holding the bootstrap p-value. Returned if optns$boot is TRUE.
optns	the control options used.

References

- *Dubey, P. and Müller, H.G., 2019. Fréchet analysis of variance for random objects. Biometrika, 106(4), pp.803-821.*

Examples

```
set.seed(1)
n1 <- 100
n2 <- 100
delta <- 1
qSup <- seq(0.01, 0.99, (0.99 - 0.01) / 50)
mu1 <- rnorm(n1, mean = 0, sd = 0.5)
mu2 <- rnorm(n2, mean = delta, sd = 0.5)
Y1 <- lapply(1:n1, function(i) {
  qnorm(qSup, mu1[i], sd = 1)
})
Y2 <- lapply(1:n2, function(i) {
  qnorm(qSup, mu2[i], sd = 1)
})
Ly <- c(Y1, Y2)
Lx <- qSup
group <- c(rep(1, n1), rep(2, n2))
res <- DenANOVA(qin = Ly, supin = Lx, group = group, optns = list(boot = TRUE))
res$pvalAsy # returns asymptotic pvalue
res$pvalBoot # returns bootstrap pvalue
```

DenCPD *Fréchet Change Point Detection for Densities*

Description

Fréchet change point detection for densities with respect to L^2-Wasserstein distance.

Usage

```
DenCPD(
  yin = NULL,
  hin = NULL,
  din = NULL,
  qin = NULL,
  supin = NULL,
  optns = list()
)
```

Arguments

yin	A matrix or data frame or list holding the sample of measurements for the observed distributions. If yin is a matrix or data frame, each row holds the measurements for one distribution.
hin	A list holding the histograms for the observed distributions.
din	A matrix or data frame or list holding the density functions. If din is a matrix or data frame, each row of din holds the density function for one distribution.
qin	A matrix or data frame or list holding the quantile functions. If qin is a matrix or data frame, each row of qin holds the quantile function for one distribution. Note that the input can be only one of the four yin, hin, din, and qin. If more than one of them are specified, yin overwrites hin, hin overwrites din, and din overwrites qin.
supin	A matrix or data frame or list holding the support grids of the density functions in din or the quantile functions in qin. If supin is a matrix or data frame, each row of supin holds the support grid of the corresponding density function or quantile function. Ignored if the input is yin or hin. It can also be a vector if all density functions in din or all quantile functions in qin have the same support grid.
optns	A list of control parameters specified by list(name = value). See 'Details'.

Details

Available control options are

cutOff A scalar between 0 and 1 indicating the interval, i.e., [cutOff, 1 - cutOff], in which candidate change points lie.

Q A scalar representing the number of Monte Carlo simulations to run while approximating the critical value (stardized Brownian bridge). Default is 1000.

boot Logical, also compute bootstrap p-value if TRUE. Default is FALSE.

R The number of bootstrap replicates. Only used when boot is TRUE. Default is 1000.

nqSup A scalar giving the number of the support points for 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.

qSup A numeric vector holding the support grid on [0, 1] based on which the L^2 Wasserstein distance (i.e., the L^2 distance between the quantile functions) is computed. It overrides nqSup.

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 A scalar giving the number of support points the kernel density estimation used in `CreateDensity()`; numeric - default: 101.

dSup User defined output grid for the support of kernel density estimation used in `CreateDensity()`, it overrides `ndSup`.

delta A scalar giving 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 `DenCPD` object — a list containing the following fields:

`tau`	a scalar holding the estimated change point.
`pvalAsy`	a scalar holding the asymptotic p-value.
`pvalBoot`	a scalar holding the bootstrap p-value. Returned if `optns$boot` is TRUE.
`optns`	the control options used.

References

- *Dubey, P. and Müller, H.G., 2020. Fréchet change-point detection. The Annals of Statistics, 48(6), pp.3312-3335.*

Examples

```
set.seed(1)
n1 <- 100
n2 <- 200
delta <- 0.75
qSup <- seq(0.01, 0.99, (0.99 - 0.01) / 50)
mu1 <- rnorm(n1, mean = delta, sd = 0.5)
mu2 <- rnorm(n2, mean = 0, sd = 0.5)
Y1 <- lapply(1:n1, function(i) {
  qnorm(qSup, mu1[i], sd = 1)
})
Y2 <- lapply(1:n2, function(i) {
  qnorm(qSup, mu2[i], sd = 1)
})
Ly <- c(Y1, Y2)
Lx <- qSup
```

```
res <- DenCPD(qin = Ly, supin = Lx, optns = list(boot = TRUE))
res$tau # returns the estimated change point
res$pvalAsy # returns asymptotic pvalue
res$pvalBoot # returns bootstrap pvalue
```

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)
```

DenFVar *Fréchet Variance for Densities*

Description

Obtain Fréchet variance for densities with respect to L^2-Wasserstein distance.

Usage

```
DenFVar(
  yin = NULL,
  hin = NULL,
  din = NULL,
  qin = NULL,
  supin = NULL,
  optns = list()
)
```

Arguments

yin	A matrix or data frame or list holding the sample of measurements for the observed distributions. If yin is a matrix or data frame, each row holds the measurements for one distribution.
hin	A list holding the histograms for the observed distributions.
din	A matrix or data frame or list holding the density functions. If din is a matrix or data frame, each row of din holds the density function for one distribution.
qin	A matrix or data frame or list holding the quantile functions. If qin is a matrix or data frame, each row of qin holds the quantile function for one distribution. Note that the input can be only one of the four yin, hin, din, and qin. If more than one of them are specified, yin overwrites hin, hin overwrites din, and din overwrites qin.
supin	A matrix or data frame or list holding the support grids of the density functions in din or the quantile functions in qin. If supin is a matrix or data frame, each row of supin holds the support grid of the corresponding density function or quantile function. Ignored if the input is yin or hin. It can also be a vector if all density functions in din or all quantile functions in qin have the same support grid.
optns	A list of control parameters specified by list(name = value). See 'Details'.

Details

Available control options are

nqSup A scalar giving the number of the support points for 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.

qSup A numeric vector holding the support grid on [0, 1] based on which the L^2 Wasserstein distance (i.e., the L^2 distance between the quantile functions) is computed. It overrides `nqSup`.

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 A scalar giving the number of support points the kernel density estimation used in `CreateDensity()`; numeric - default: 101.

dSup User defined output grid for the support of kernel density estimation used in `CreateDensity()`, it overrides `ndSup`.

delta A scalar giving 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 fields:

`DenFVar`	A scalar holding the Fréchet variance.
`optns`	A list of control options used.

Examples

```
set.seed(1)
n <- 100
mu <- rnorm(n, mean = 0, sd = 0.5)
qSup <- seq(0.01, 0.99, (0.99 - 0.01) / 50)
Ly <- lapply(1:n, function(i) qnorm(qSup, mu[i], sd = 1))
Lx <- qSup
res <- DenFVar(qin = Ly, supin = Lx)
res$DenFVar
```

`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` *L^2 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 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>.

`GloCorReg` *Global Fréchet regression for correlation matrices*

Description

Global Fréchet regression for correlation matrices with Euclidean predictors.

Usage

```
GloCorReg(x, M, xOut = NULL, optns = list())
```

Arguments

`x`	an n by p matrix or data frame of predictors.
`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 correlation matrix of dimension q by q.
`xOut`	an m by p matrix or data frame of output predictor levels. It can be a vector of length p if m = 1.
`optns`	A list of options control parameters specified by `list(name=value)`. See 'Details'.

Details

Available control options are

metric choice of metric. `'frobenius'` and `'power'` are supported, which corresponds to Frobenius metric and Euclidean power metric, respectively. Default is Frobenius metric.

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

digits the integer indicating the number of decimal places (round) to be kept in the output. Default is NULL, which means no round operation.

Value

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

`fit`	a list of estimated correlation matrices at `x`.
`predict`	a list of estimated correlation matrices at `xOut`. Included if `xOut` is not `NULL`.
`RSquare`	Fréchet coefficient of determination.
`AdjRSquare`	adjusted Fréchet coefficient of determination.
`residuals`	Frobenius distance between the true and fitted correlation matrices.
`xOut`	the output predictor level used.
`optns`	the control options used.

References

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

Examples

```
# Generate simulation data
n <- 100
q <- 10
d <- q * (q - 1) / 2
xOut <- seq(0.1, 0.9, length.out = 9)
x <- runif(n, min = 0, max = 1)
y <- list()
for (i in 1:n) {
  yVec <- rbeta(d, shape1 = x[i], shape2 = 1 - x[i])
  y[[i]] <- matrix(0, nrow = q, ncol = q)
  y[[i]][lower.tri(y[[i]])] <- yVec
  y[[i]] <- y[[i]] + t(y[[i]])
  diag(y[[i]]) <- 1
}
# Frobenius metric
fit1 <- GloCorReg(x, y, xOut,
  optns = list(metric = "frobenius", digits = 5)
)
# Euclidean power metric
fit2 <- GloCorReg(x, y, xOut,
  optns = list(metric = "power", alpha = .5)
)
```

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)
```

GloPointPrReg *Global Cox point process regression.*

Description

Global Fréchet regression for replicated Cox point processes with respect to L^2-Wasserstein distance on shape space and Euclidean 2-norm on intensity factor space.

Usage

```
GloPointPrReg(xin = NULL, tin = NULL, T0 = NULL, xout = NULL, optns = list())
```

Arguments

xin	An n by p matrix with input measurements of the predictors.
tin	A list holding the sample of event times of each replicated point process, where the ith element of the list tin holds the event times of the point process corresponding to the ith row of xin.

T0	A positive scalar that defines the time window [0,T0] where the replicated Cox point processes are observed.
xout	A k by p matrix 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 bwDen (see LocDenReg for this option description) and

L Upper Lipschitz constant for quantile space; numeric -default: 1e10.

M Lower Lipschitz constant for quantile space; numeric -default: 1e-10.

dSup User defined output grid for the support of kernel density estimation used in CreateDensity() for mapping from quantile space to shape space. This grid must be in [0,T0]. Default is an equidistant grid with nqSup+2 points.

nqSup A scalar with the number of equidistant points in (0,1) used to obtain the empirical quantile function from each point process. Default: 500.

Value

A list containing the following components:

xout	Input xout.
dSup	Support of each estimated (up to a constant) conditional intensity regression function in the columns of intensityReg.
intensityReg	A matrix of dimension length(dSup) by nrow(xout) holding the estimated intensity regression functions up to a constant over the support grid dSup, where each column corresponds to a predictor level in the corresponding row of xout.
xin	Input xin.
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.

Gajardo, Á. and Müller, H.-G. (2022). "Cox Point Process Regression." IEEE Transactions on Information Theory, 68(2), 1133-1156.

Examples

```
n=100
alpha_n=sqrt(n)
alpha1=2.0
beta1=1.0
gridQ=seq(0,1,length.out=500+2)[2:(500+1)]
X=runif(n,0,1)#p=1
tau=matrix(0,nrow=n,ncol=1)
for(i in 1:n){
```

```
  tau[i]=alpha1+beta1*X[i]+truncnorm::rtruncnorm(1, a=-0.3, b=0.3, mean = 0, sd = 1.0)
}
Ni_n=matrix(0,nrow=n,ncol=1)
u0=0.4
u1=0.5
u2=0.05
u3=-0.01
tin=list()
for(i in 1:n){
  Ni_n[i]=rpois(1,alpha_n*tau[i])
  mu_x=u0+u1*X[i]+truncnorm::rtruncnorm(1,a=-0.1,b=0.1,mean=0,sd=1)
  sd_x=u2+u3*X[i]+truncnorm::rtruncnorm(1,a=-0.02,b=0.02,mean=0,sd=0.5)
  if(Ni_n[i]==0){
    tin[[i]]=c()
  }else{
   tin[[i]]=truncnorm::rtruncnorm(Ni_n[i],a=0,b=1,mean=mu_x,sd=sd_x) #Sample from truncated normal
  }
}
res=GloPointPrReg(
  xin=matrix(X,ncol=1),tin=tin,
  T0=1,xout=matrix(seq(0,1,length.out=10),ncol=1),
  optns=list(bwDen=0.1)
)
```

`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)
```

LocCorReg *Local Fréchet regression for correlation matrices*

Description

Local Fréchet regression for correlation matrices with Euclidean predictors.

Usage

```
LocCorReg(x, M, xOut = NULL, optns = list())
```

Arguments

x	an n by p matrix or data frame of predictors.
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 correlation matrix of dimension q by q.
xOut	an m by p matrix or data frame of output predictor levels. It can be a vector of length p if m = 1.
optns	A list of options control parameters specified by list(name=value). See 'Details'.

Details

Available control options are

metric choice of metric. `'frobenius'` and `'power'` are supported, which corresponds to Frobenius metric and Euclidean power metric, respectively. Default is Frobenius metric.

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

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

bw bandwidth for local Fréchet regression, if not entered it would be chosen from cross validation.

digits the integer indicating the number of decimal places (round) to be kept in the output. Default is NULL, which means no round operation.

Value

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

`fit`	a list of estimated correlation matrices at `x`.
`predict`	a list of estimated correlation matrices at `xOut`. Included if `xOut` is not `NULL`.
`residuals`	Frobenius distance between the true and fitted correlation matrices.
`xOut`	the output predictor level used.
`optns`	the control options used.

References

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

Examples

```
# Generate simulation data

n <- 100
q <- 10
d <- q * (q - 1) / 2
xOut <- seq(0.1, 0.9, length.out = 9)
x <- runif(n, min = 0, max = 1)
y <- list()
for (i in 1:n) {
  yVec <- rbeta(d, shape1 = sin(pi * x[i]), shape2 = 1 - sin(pi * x[i]))
  y[[i]] <- matrix(0, nrow = q, ncol = q)
  y[[i]][lower.tri(y[[i]])] <- yVec
  y[[i]] <- y[[i]] + t(y[[i]])
  diag(y[[i]]) <- 1
}
# Frobenius metric
fit1 <- LocCorReg(x, y, xOut,
  optns = list(metric = "frobenius", digits = 2)
)
```

```
# Euclidean power metric
fit2 <- LocCorReg(x, y, xOut,
  optns = list(
    metric = "power", alpha = .5,
    kernel = "epan", bw = 0.08
  )
)
```

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)
```

LocPointPrReg *Local Cox point process regression.*

Description

Local Fréchet regression for replicated Cox point processes with respect to L^2-Wasserstein distance on shape space and Euclidean 2-norm on intensity factor space.

Usage

```
LocPointPrReg(xin = NULL, tin = NULL, T0 = NULL, xout = NULL, optns = list())
```

Arguments

xin	An n by p matrix with input measurements of the predictors, where p is at most 2.
tin	A list holding the sample of event times of each replicated point process, where the ith element of the list tin holds the event times of the point process corresponding to the ith row of xin.
T0	A positive scalar that defines the time window [0,T0] where the replicated Cox point processes are observed.

xout	A k by p matrix 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 bwDen, kernelReg (see LocDenReg for these option descriptions) and

L Upper Lipschitz constant for quantile space; numeric -default: 1e10.

M Lower Lipschitz constant for quantile space; numeric -default: 1e-10.

dSup User defined output grid for the support of kernel density estimation used in CreateDensity() for mapping from quantile space to shape space. This grid must be in [0,T0]. Default is an equidistant with nqSup+2 points.

nqSup A scalar with the number of equidistant points in (0,1) used to obtain the empirical quantile function from each point process. Default: 500.

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 leave-one-out cross-validation.

Value

A list containing the following components:

xout	Input xout.
dSup	Support of each estimated (up to a constant) conditional intensity regression function in the columns of intensityReg.
intensityReg	A matrix of dimension length(dSup) by nrow(xout) holding the estimated intensity regression functions up to a constant over the support grid dSup, where each column corresponds to a predictor level in the corresponding row of xout.
xin	Input xin.
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.

Gajardo, Á. and Müller, H.-G. (2022). "Cox Point Process Regression." IEEE Transactions on Information Theory, 68(2), 1133-1156.

Examples

```
n=100
alpha_n=sqrt(n)
alpha1=2.0
beta1=1.0
gridQ=seq(0,1,length.out=500+2)[2:(500+1)]
X=runif(n,0,1)#p=1
tau=matrix(0,nrow=n,ncol=1)
for(i in 1:n){
```

```
  tau[i]=alpha1+beta1*X[i]+truncnorm::rtruncnorm(1, a=-0.3, b=0.3, mean = 0, sd = 1.0)
}
Ni_n=matrix(0,nrow=n,ncol=1)
u0=0.4
u1=0.5
u2=0.05
u3=-0.01
tin=list()
for(i in 1:n){
  Ni_n[i]=rpois(1,alpha_n*tau[i])
  mu_x=u0+u1*X[i]+truncnorm::rtruncnorm(1,a=-0.1,b=0.1,mean=0,sd=1)
  sd_x=u2+u3*X[i]+truncnorm::rtruncnorm(1,a=-0.02,b=0.02,mean=0,sd=0.5)
  if(Ni_n[i]==0){
    tin[[i]]=c()
  }else{
   tin[[i]]=truncnorm::rtruncnorm(Ni_n[i],a=0,b=1,mean=mu_x,sd=sd_x) #Sample from truncated normal
  }
}
res=LocPointPrReg(
  xin=matrix(X,ncol=1),
  tin=tin,T0=1,xout=matrix(seq(0,1,length.out=10),ncol=1),
  optns=list(bwDen=0.1,bwReg=0.1)
)
```

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.

`NetANOVA` *Fréchet ANOVA for Networks*

Description

Fréchet analysis of variance for graph Laplacian matrices, covariance matrices, or correlation matrices with respect to the Frobenius distance.

Usage

```
NetANOVA(Ly = NULL, group = NULL, optns = list())
```

Arguments

Ly	A list (length n) of m by m matrices or a m by m by n array where Ly[, , i] contains an m by m matrix, which can be either graph Laplacian matrices or covariance matrices or correlation matrices.
group	A vector containing the group memberships of the corresponding matrices in Ly.
optns	A list of control parameters specified by list(name = value). See 'Details'.

Details

Available control options are:

boot Logical, also compute bootstrap p-value if TRUE. Default is FALSE.

R The number of bootstrap replicates. Only used when boot is TRUE. Default is 1000.

Value

A NetANOVA object — a list containing the following fields:

pvalAsy	A scalar holding the asymptotic p-value.
pvalBoot	A scalar holding the bootstrap p-value. Returned if optns$boot is TRUE.
optns	The control options used.

References

- *Dubey, P. and Müller, H.G., 2019. Fréchet analysis of variance for random objects. Biometrika, 106(4), pp.803-821.*

Examples

```
set.seed(1)
n1 <- 100
n2 <- 100
gamma1 <- 2
gamma2 <- 3
Y1 <- lapply(1:n1, function(i) {
  igraph::laplacian_matrix(igraph::sample_pa(n = 10, power = gamma1,
                                             directed = FALSE),
                           sparse = FALSE)
})
Y2 <- lapply(1:n2, function(i) {
  igraph::laplacian_matrix(igraph::sample_pa(n = 10, power = gamma2,
                                             directed = FALSE),
                           sparse = FALSE)
})
Ly <- c(Y1, Y2)
group <- c(rep(1, n1), rep(2, n2))
res <- NetANOVA(Ly, group, optns = list(boot = TRUE))
res$pvalAsy # returns asymptotic pvalue
res$pvalBoot # returns bootstrap pvalue
```

NetCPD *Fréchet Change Point Detection for Networks*

Description

Fréchet change point detection for graph Laplacian matrices, covariance matrices, or correlation matrices with respect to the Frobenius distance.

Usage

```
NetCPD(Ly = NULL, optns = list())
```

Arguments

Ly	A list (length n) of m by m matrices or a m by m by n array where Ly[, , i] contains an m by m matrix, which can be either graph Laplacian matrices or covariance matrices or correlation matrices.
optns	A list of control parameters specified by list(name = value). See 'Details'.

Details

Available control options are:

cutOff A scalar between 0 and 1 indicating the interval, i.e., [cutOff, 1 - cutOff], in which candidate change points lie.

Q A scalar representing the number of Monte Carlo simulations to run while approximating the critical value (stardized Brownian bridge). Default is 1000.

boot Logical, also compute bootstrap p-value if TRUE. Default is FALSE.

R The number of bootstrap replicates. Only used when boot is TRUE. Default is 1000.

Value

A NetCPD object — a list containing the following fields:

tau	a scalar holding the estimated change point.
pvalAsy	A scalar holding the asymptotic p-value.
pvalBoot	A scalar holding the bootstrap p-value. Returned if optns$boot is TRUE.
optns	The control options used.

References

- *Dubey, P. and Müller, H.G., 2020. Fréchet change-point detection. The Annals of Statistics, 48(6), pp.3312-3335.*

Examples

```
set.seed(1)
n1 <- 100
n2 <- 100
gamma1 <- 2
gamma2 <- 3
Y1 <- lapply(1:n1, function(i) {
  igraph::laplacian_matrix(igraph::sample_pa(n = 10, power = gamma1,
                                             directed = FALSE),
                           sparse = FALSE)
})
Y2 <- lapply(1:n2, function(i) {
  igraph::laplacian_matrix(igraph::sample_pa(n = 10, power = gamma2,
                                             directed = FALSE),
                           sparse = FALSE)
})
Ly <- c(Y1, Y2)
res <- NetCPD(Ly, optns = list(boot = TRUE))
res$tau # returns the estimated change point
res$pvalAsy # returns asymptotic pvalue
res$pvalBoot # returns bootstrap pvalue
```

NetFIntegral *Generalized Fréchet integrals of network*

Description

Calculating generalized Fréchet integrals of networks (equipped with Frobenius norm of adjacency matrices with zero diagonal elements and non negative off diagonal elements.)

Usage

```
NetFIntegral(phi, t_out, X, U)
```

Arguments

phi	An eigenfunction along which we want to project the network
t_out	Support of phi
X	A three dimensional array of dimension length(t_out) x m x m, where X[i,,] is an m x m network adjacency matrix. The diagonal elements of adjacency matrices are zero and the off diagonal entries lie between zero and U.
U	Upper bound of off-diagonal entries

Value

A list of the following:

f	An adjacency matrix which corresponds to the Fréchet integral of X along phi

References

Dubey, P., & Müller, H. G. (2020). Functional models for time-varying random objects. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 82(2), 275-327.

Examples

```
set.seed(5)
n <- 100
N <- 50
t_out <- seq(0,1,length.out = N)
library(mpoly)
p2 <- as.function(mpoly::jacobi(2,4,3),silent=TRUE)
p4 <- as.function(mpoly::jacobi(4,4,3),silent=TRUE)
p6 <- as.function(mpoly::jacobi(6,4,3),silent=TRUE)

# first three eigenfunctions
phi1 <- function(t){
p2(2*t-1)*t^(1.5)*(1-t)^2 / (integrate(function(x) p2(2*x-1)^2*x^(3)*(1-x)^4,0,1))$value^(1/2)
}
phi2 <- function(t){
p4(2*t-1)*t^(1.5)*(1-t)^2 / (integrate(function(x) p4(2*x-1)^2*x^(3)*(1-x)^4,0,1))$value^(1/2)
}
phi3 <- function(t){
p6(2*t-1)*t^(1.5)*(1-t)^2 / (integrate(function(x) p6(2*x-1)^2*x^(3)*(1-x)^4,0,1))$value^(1/2)
}

# random component of adjacency matrices
P12 <- 0.1 ## edge between compunities
Score <- matrix(runif(n*4), nrow = n)
# edge within first community
P1_vec <- 0.5 + 0.4*Score[,1] %*% t(phi1(t_out)) + 0.1*Score[,2] %*% t(phi3(t_out))
# edge within second community
P2_vec <- 0.5 + 0.3*Score[,3] %*% t(phi2(t_out)) + 0.1*Score[,4] %*% t(phi3(t_out))

# create Network edge matrix
N_net1 <- 5 # first community number
N_net2 <- 5 # second community number

# I: four dimension array of n x n matrix of squared distances between the time point u
# of the ith process and process and the time point v of the jth object process,
# e.g.: I[i,j,u,v] <- d_F^2(X_i(u) X_j(v)).
I <- array(0, dim = c(n,n,N,N))
for(u in 1:N){
  for(v in 1:N){
   #frobenius norm between two adjcent matrix
   I[,,u,v] <- outer(P1_vec[,u], P1_vec[,v], function(a1, a2) (a1-a2)^2*(N_net1^2-N_net1)) +
      outer(P2_vec[,u], P2_vec[,v], function(a1, a2) (a1-a2)^2*(N_net2^2-N_net2))
  }
}
```

```
# check ObjCov work
Cov_result <- ObjCov(t_out, I, 3, smooth=FALSE)
Cov_result$lambda  # 0.266 0.15 0.04

# sum((Cov_result$phi[,1] - phi1(t_out))^2) / sum(phi1(t_out)^2)
# sum((Cov_result$phi[,2] - phi2(t_out))^2) / sum(phi2(t_out)^2)
# sum((Cov_result$phi[,3] - phi3(t_out))^2) / sum(phi3(t_out)^2)

# e.g. subj 2
subj <- 2
# X_mat is the network for varying times with X[i,,] is the adjacency matrices
# for the ith time point
X_mat <- array(0, c(N,(N_net1+N_net2), (N_net1+N_net2)))
for(i in 1:N){
  # edge between communities is P12
  Mat <- matrix(P12, nrow = (N_net1+N_net2), ncol = (N_net1+N_net2))
  # edge within the first communitiy is P1
  Mat[1:N_net1, 1:N_net1] <- P1_vec[subj, i]
  # edge within the second community is P2
  Mat[(N_net1+1):(N_net1+N_net2), (N_net1+1):(N_net1+N_net2)] <- P2_vec[subj, i]
  diag(Mat) <- 0 #diagonal element is 0
  X_mat[i,,] <- Mat
}
# output the functional principal network(adjacency matrice) of the second eigenfunction
NetFIntegral(Cov_result$phi[,2], t_out, X_mat, 2)
```

`NetFVar` *Fréchet Variance for Networks*

Description

Obtain Fréchet variance for graph Laplacian matrices, covariance matrices, or correlation matrices with respect to the Frobenius distance.

Usage

```
NetFVar(Ly = NULL)
```

Arguments

`Ly`	A list (length n) of m by m matrices or a m by m by n array where `Ly[, , i]` contains an m by m matrix, which can be either graph Laplacian matrices or covariance matrices or correlation matrices.

Value

A list containing the following fields:

`NetFVar`	A scalar holding the Fréchet variance.
`NetFMean`	A matrix holding the Fréchet mean.

Examples

```
set.seed(1)
n <- 100
U <- pracma::randortho(10)
Ly <- lapply(1:n, function(i) {
  U %*% diag(rexp(10, (1:10)/2)) %*% t(U)
})
res <- NetFVar(Ly)
res$NetFVar
```

`ObjCov` *Object Covariance*

Description

Calculating covariance for time varying object data

Usage

```
ObjCov(tgrid, I, K, smooth = TRUE)
```

Arguments

`tgrid`	Time grid for the time varying object data and covariance function
`I`	A four dimension array of `n` x `n` matrix of squared distances between the time point u of the ith process and process and the time point v of the jth object process, e.g.: $I[i, j, u, v] = d^2(X_i(u)X_j(v))$
`K`	Numbers of principal components
`smooth`	Logical indicating if the smoothing is enabled when calculating the eigenvalues and eigenfunctions

Value

A list of the following:

`C`	Estimated object covariance (non-smooth) on the 2D grid of dimension `length(tgrid)` X `length(tgrid)`
`sC`	Estimated object covariance (smooth) on the 2D grid of dimension `length(tgrid)` X `length(tgrid)`
`tgrid`	Time grid for the time varying object data and covariance function
`K`	Numbers of principal components
`phi`	Matrix of smooth eigenfunctions (dimension: `length(tgrid)` X `K`)
`lambda`	Vector of eigenvalues of dimension `K`

References

Dubey, P., & Müller, H. G. (2020). Functional models for time-varying random objects. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 82(2), 275-327.

Examples

```
### functional covariate
phi1 <- function(x) -cos(pi*x/10)/sqrt(5)
phi2 <- function(x)  sin(pi*x/10)/sqrt(5)

lambdaX <- c(4,2)
# training set
n <- 100
N <- 50
tgrid <- seq(0,10,length.out = N)

Xi <- matrix(rnorm(2*n),nrow=n,ncol=2)
CovX <- lambdaX[1] * phi1(tgrid) %*% t(phi1(tgrid)) + lambdaX[2] * phi2(tgrid) %*% t(phi2(tgrid))
comp1 = lambdaX[1]^(1/2) * Xi[,1] %*% t(phi1(tgrid))
comp2 = lambdaX[2]^(1/2) * Xi[,2] %*% t(phi2(tgrid))
SampleX <- comp1 + comp2

I <- array(0, c(n,n,N,N))
for (u in 1:N){
  for (v in 1:N){
    temp1 <- SampleX[,u]
   temp2 <- SampleX[,v]
    I[,,u,v] <- outer(temp1, temp2, function(v1,v2){
     (v1 - v2)^2
   })
 }
}

result_cov <- ObjCov(tgrid, I, 2)
result_cov$lambda #4 2

sC <- result_cov$sC
sum((sC-CovX)^2) / sum(sC^2)
sum((phi1(tgrid)-result_cov$phi[,1])^2)/sum(phi1(tgrid)^2)
```

`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 gradient 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.

VarObj — *Fréchet Variance Trajectory for densities*

Description

Modeling time varying density objects with respect to L^2-Wasserstein distance by Fréchet variance trajectory

Usage

```
VarObj(tgrid, yin = NULL, hin = NULL, din = NULL, qin = NULL, optns = list())
```

Arguments

tgrid	Time grid vector for the time varying object data.
yin	An array or list of lists holding the samples of observations. If yin is an array, it has size n x length(tgrid) x numbers of samples holding the observation, such that yin[i,j,] holds the observations to the ith sample at the jth time grid. If yin is a list of lists, yin[[i]][[j]] holds the observations to the ith sample at the jth time grid.
hin	A list of lists holding the histogram for each subject. hin[[i]][[j]] holds the histogram to the ith sample at the jth time grid.

din	A three dimension array of size n x length(tgrid) x length(optns$dSup) holding the observed densities, such that din[i,j,] holds the observed density function taking values on optns$dSup corresponding to the ith sample at the jth time grid.
qin	A three dimension array of size n x length(tgrid) x length(optns$qSup) holding the observed quantiles, such that din[i,j,] holds the observed density function taking values on optns$qSup corresponding to the ith sample at the jth time grid. Note that only one of yin, hin, din and qin needs to be input. If more than one of them are specified, yin overwrites hin, hin overwrites din and din overwrites qin. where each row holds the observations for one subject on the common grid tGrid.
optns	a list of options control parameters specified by list(name=value).

Details

Available control options are qSup, nqSup, dSup and other options in FPCA of fdapace.

Value

A list of the following:

tgridout	Time grid vector for the output time varying object data.
K	Numbers of principal components.
nu	A vector of dimension length(tgridout) giving the mean function support on tgridout of the Fréchet variance function.
lambda	A vector of dimension K containing eigenvalues.
phi	A length(tgridout) X K matrix containing eigenfunctions support on tgridout of the Fréchet variance function.
xiEst	A n X K matrix containing the FPC estimates.
cumFVE	A vector of dimension K with the fraction of the cumulative total variance explained with each additional FPC.
FPCAObj	FPCA Object of Fréchet variance function.
tgridin	Input tgrid.
qSup	A vector of dimension length(tgridin) giving the domain grid of quantile functions qout.
qout	A three dimension array of dimension n x length(tgridin) x length(qSup) holding the observed quantiles, such that qout[i,j,] holds the observed density function taking values on qSup corresponding to the ith sample at the jth time grid.
qmean	A length(tgridin) X length(qSup) matrix containing the time varying Fréchet mean function.
VarTraj	A n X length(tgridin) matrix containing the variance trajectory.

References

Dubey, P., & Müller, H. G. (2021). Modeling Time-Varying Random Objects and Dynamic Networks. Journal of the American Statistical Association, 1-33.

Examples

```
set.seed(1)
#use yin
tgrid = seq(1, 50, length.out = 50)
dSup = seq(-10, 60, length.out = 100)
yin = array(dim=c(30, 50, 100))
for(i in 1:30){
  yin[i,,] = t(sapply(tgrid, function(t){
    rnorm(100, mean = rnorm(1, mean = 1, sd = 1/t))
  }))
}
result1 = VarObj(tgrid, yin = yin)
plot(result1$phi[,1])
plot(result1$phi[,2])
yin2 = replicate(30, vector("list", 50), simplify = FALSE)
for(i in 1:30){
  for(j in 1:50){
    yin2[[i]][[j]] = yin[i,j,]
  }}
result1 = VarObj(tgrid, yin = yin2)

# use hin
tgrid = seq(1, 50, length.out = 50)
dSup = seq(-10, 60, length.out = 100)
hin =  replicate(30, vector("list", 50), simplify = FALSE)
for(i in 1:30){
  for (j in 1:50){
    hin[[i]][[j]] = hist(yin[i,j,])
  }
}
result2 = VarObj(tgrid, hin = hin)

# use din
tgrid = seq(1, 50, length.out = 50)
dSup = seq(-10, 60, length.out = 100)
din = array(dim=c(30, 50, 100))
for(i in 1:30){
  din[i,,] = t(sapply(tgrid, function(t){
    dnorm(dSup, mean = rnorm(1, mean = t, sd = 1/t))
  }))
}
result3 = VarObj(tgrid, din = din, optns=list(dSup = dSup))

# use qin
tgrid = seq(1, 50, length.out = 50)
qSup = seq(0.00001,1-0.00001,length.out = 100)
qin = array(dim=c(30, 50, 100))
for(i in 1:30){
  qin[i,,] = t(sapply(tgrid, function(t){
    qnorm(qSup, mean = rnorm(1, mean = t, sd = 1/t))
  }))
```

```
}
result4 = VarObj(tgrid, qin = qin, optns=list(qSup = round(qSup, 4)))
```

`WassFIntegral` *Generalized Fréchet integrals of 1D distribution*

Description

Calculating generalized Fréchet integrals of 1D distribution (equipped with Wasserstein distance)

Usage

```
WassFIntegral(phi, t_out, Q, Qout)
```

Arguments

`phi`	An eigenfunction along which we want to project the distribution
`t_out`	Support of `phi`
`Q`	A `length(t_out)` X `length(Qout)` matrix whose jth row corresponds to the quantile function on grid `Qout` for the jth time point.
`Qout`	Support of the quantile valued process

Value

A list of the following:

`f`	Quantile function corresponding to the frechet integral of `Q` along `phi`

References

Dubey, P., & Müller, H. G. (2020). Functional models for time-varying random objects. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 82(2), 275-327.

Examples

```
#simulation as in the paper Dubey, P., & Müller, H. G. (2020).
# Functional models for time-varying random objects.
# JRSSB, 82(2), 275-327.

n <- 100
N <- 50
t_out <- seq(0,1,length.out = N)

phi1 <- function(t){
  (t^2-0.5)/0.3416
}
phi2 <- function(t){
```

```
  sqrt(3)*t
}
phi3 <- function(t){
  (t^3 - 0.3571*t^2 - 0.6*t + 0.1786)/0.0895
}

Z <- cbind(rnorm(n)*sqrt(12), rnorm(n), runif(n)*sqrt(72), runif(n)*sqrt(9))
mu_vec <- 1 + Z[,1] %*% t(phi1(t_out)) + Z[,2] %*% t(phi3(t_out))
sigma_vec <- 3 + Z[,3] %*% t(phi2(t_out)) + Z[,4] %*% t(phi3(t_out))

# grids of quantile function
Nq <- 40
eps <- 0.00001
Qout <- seq(0+eps,1-eps,length.out=Nq)

# I: four dimension array of n x n matrix of squared distances
# between the time point u of the ith process and
# process and the time point v of the jth object process,
# e.g.: I[i,j,u,v] <- d_w^2(X_i(u) X_j(v)).
I <- array(0, dim = c(n,n,N,N))
for(i in 1:n){
  for(j in 1:n){
    for(u in 1:N){
      for(v in 1:N){
        #wasserstein distance between distribution X_i(u) and X_j(v)
       I[i,j,u,v] <- (mu_vec[i,u] - mu_vec[j,v])^2 + (sigma_vec[i,u] - sigma_vec[j,v])^2
      }
    }
  }
}

# check ObjCov work
Cov_result <- ObjCov(t_out, I, 3)
#Cov_result$lambda #12 6 1.75

# calculate Q
i <- 6 # for the ith subject
Q <- t(sapply(1:N, function(t){
  qnorm(Qout, mean = mu_vec[i,t], sd = sigma_vec[i,t])
}))

score_result <- WassFIntegral(Cov_result$phi[,1], t_out, Q, Qout)
score_result$f
```

Index