Package: fdadensity (via r-universe)

September 1, 2024

Type Package

Title Functional Data Analysis for Density Functions by Transformation to a Hilbert Space

Version 0.1.0

Date 2017-09-13

Author A. Petersen, P. Z. Hadjipantelis and H.G. Mueller

Maintainer Alexander Petersen <petersen@pstat.ucsb.edu>

Description Functional Data Analysis for Density Functions by Transformation to a Hilbert space. A ``better'' domain idea.

Depends R (>= 3.1.1)

License BSD_3_clause + file LICENSE

LazyData false

Imports fdapace (>= 0.2.0)

Suggests testthat

RoxygenNote 6.0.1

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

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

RemoteRef HEAD

RemoteSha 03a6ef0156fa908880b068383c16869065ee7c42

Contents

`BacteriaPI` *pH distribution of 813 bacterial organisms*

Description

The approximate kernel density estimates of the 813 bacterial organisms' isoelectric point (pI) protein distributions.

Format

A matrix with 813 rows and 768 columns:

rowname General organism identifier

colspace pH in [0,14]

References

The authors would like to thank Dr. Chris Knight for providing the original data.

`CreateDensity` *Create density from raw data*

Description

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

Usage

```
CreateDensity(y, optns = list())
```

Arguments

`y`	A vector of raw readings.
`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 (using 'CV' or 'GCV'); positive numeric - default: determine automatically based on CV

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

delta The size of the bin to be used; numeric - default: determine automatically as "max(c(diff(range(y))/1000, min(diff(sort(unique(y))))))"

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: TRUE

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`	Variance for measure error.The bandwidth used by smoothing.
`x`	A vector of length *nGridReg* with the values of the KDE's support points.
`y`	A vector of length *nGridReg* with the values of the KDE at the support points.

References

HG Mueller, JL Wang and WB Capra (1997). "From lifetables to hazard rates: The transformation approach." Biometrika 84, 881-892.

Examples

```
par(mfrow=c(1,2))
makeComparisonPlotE <- function(N, mySeed = 123){
  set.seed(mySeed)
  asdf2  = (rexp(N, rate = 1.5))
  plot(density(asdf2, bw = "SJ"), main= "Exponential (rate=1.5)",
    xlab = paste0(collapse = '', c( "N = ", as.character(N))) , ylim = c(0, 1.45))
  lines(density(asdf2), col='red')
  Unormal = CreateDensity(y = asdf2, optns = list(infSupport = FALSE))
  lines(col='green', x = Unormal$x, y = Unormal$y)
  lines(col='magenta' , Unormal$x, dexp(Unormal$x, rate = 1.5))
  abline(v = min(asdf2))
  abline(v = max(asdf2))
  legend(legend = c("SJ", "R-default", "HADES-like", "True PDF"),
    lwd= 2, col=c("black", "red", "green", "magenta"), bty = 'n', 'topright')
}

makeComparisonPlotE(100)
makeComparisonPlotE(2000)
```

CreateModeOfVarPlotLQ2D *Functional Principal Component Analysis mode of variation plot*

Description

Create the k-th mode of variation plot around the mean. The red-line is the functional mean, the grey shaded areas show the range of variations around the mean: $\pm Q\sqrt{\lambda_k}\phi_k$ for the dark grey area Q = 1, and for the light grey are Q = 2.

Usage

```
CreateModeOfVarPlotLQ2D(fpcaObj, dSup = NULL, k = 1, domain = "Q",
  alpha = 0, numOfModes = 7, dSupPlot = NULL, ...)
```

Arguments

`fpcaObj`	An FPCA class object returned by FPCA().
`dSup`	The support of the original density used by LQD(relevant only for density domain)
`k`	The k-th mode of variation to plot (default k = 1)
`domain`	character defining if we should plot on the Quantile ('Q') or the Density ('D') domain (default: 'Q')
`alpha`	regularisation parameter
`numOfModes`	scalar number of principal modes to plot (relevant only for density domain, needs to be an odd number >1.) (default: 7)
`dSupPlot`	The support of the original density used for plotting (relevant only for density domain)
`...`	Additional arguments for the 'plot' function.

Examples

```
library(fdapace)
set.seed(1)
n <- 20
pts <- seq(0, 1, by=0.05)
sampWiener <- Wiener(n, pts)
sampWiener <- Sparsify(sampWiener, pts, 10)
res <- FPCA(sampWiener$Ly, sampWiener$Lt)
CreateModeOfVarPlotLQ2D(res)
```

dens2lqd — *Function for converting densities to quantile functions*

Description

Function for converting densities to quantile functions

Usage

```
dens2lqd(dens, dSup, N = length(dSup), lqSup = NULL)
```

Arguments

`dens`	density values on dSup - must be strictly positive and integrate to 1
`dSup`	support (grid) for Density domain
`N`	desired number of points on a [0,1] grid for quantile function; default length(dSup)
`lqSup`	support for LQ domain - must begin at 0 and end at 1; default [0,1] with N-equidistant support points

Value

lqd log quantile density on lqSup

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2015

Examples

```
x <- seq(0,2,length.out =512)
y <- rep(0.5,length.out =512)
yOnLQ <- dens2lqd( dens=y, dSup = x) # should equate # -log(1/2)
```

fastTrapz — *Calculate integral via trapezoid integration*

Description

Calculate the value of an integra via trapezoid integration

Usage

```
fastTrapz(x, y)
```

Arguments

`x`	A vector of time-points 'x'; has to be sorted.
`y`	A vector of values at time f(x)

Value

A scalar with the associated value

Examples

```
n <- 1001
x <- seq(0, 3*pi, len = n)
y <- sin(x)
fastTrapz(x, y) # 1.999985
```

`GetFVE` *Function for converting densities to quantile functions*

Description

Function for converting densities to quantile functions

Usage

```
GetFVE(fpcaObj, dmatrix, dSup, alpha = 0.01)
```

Arguments

`fpcaObj`	PACE output (FPCA on LQDs)
`dmatrix`	matrix of densities measures on grid dout, rows correspond to individual densities
`dSup`	support for Density domain - max and min values are assumed to mark the boundary of the support.
`alpha`	scalar to regularise with (default = 0.01)

Value

FVEvector

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2015

`lqd2dens` *Function for converting log quantile densities to densities*

Description

Function for converting log quantile densities to densities

Usage

```
lqd2dens(lqd, lqSup = seq(0, 1, length.out = length(lqd)), dSup,
  useSplines = TRUE)
```

Arguments

`lqd`	log quantile density on lqSup
`lqSup`	support for LQ domain - must begin at 0 and end at 1
`dSup`	support for Density domain - max and min values are assumed to mark the boundary of the support.
`useSplines`	fit spline to the lqd when doing the numerical integration (default: TRUE)

Value

dens density values on dSup

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2015

Examples

```
x <- seq(0,2,length.out =512)
y <- rep(0.5,length.out =512)
yOnLQ <- dens2lqd(dens=y, dSup = x) # should equate # -log(1/2)
yOnDens <- lqd2dens(dSup=x, lqd = yOnLQ)
```

MakeDENsample *Convenience function for converting log-quantile functions to densities*

Description

See 'lqd2dens' and 'RegulariseByAlpha' for more details. This function will automatically calculate the:sout support points that the 99.5-th percentile (calculated using all available samples) is not positive. It will then normalise the new densities to integraate to one, regularise by 'alpha' and calculate the LQ projection.

Usage

```
MakeDENsample(qmatrix, alpha = 0, dSupUsedInLQ, dSup, useAlpha = FALSE)
```

Arguments

`qmatrix`	Matrix holding the log-quantiles values on [0,1] and assumed to lay strictly on dSupUsedInLQ
`alpha`	Scalar to deregularise the supports with (default=0)
`dSupUsedInLQ`	Support (grid) for Density domain that was used when calculating the LQ projections (maybe different from the original)
`dSup`	Original support (grid) for Density domain of the data
`useAlpha`	Logical indicator to regularise the support so the smallest value of each density is zero

Value

list with the 'DEN' projected data, and 'dSup' that was originally used for the data.

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2015

MakeLQDsample *Convenience function for converting densities to log-quantile functions*

Description

See 'dens2lqd' and 'RegulariseByAlpha' for more details. This function will automatically trim out support points that the 99.5-th percentile (calculated using all available samples) is not positive. It will then normalise the new densities to integraate to one, regularise by 'alpha' and calculate the LQ projection.

Usage

```
MakeLQDsample(dmatrix, alpha = 0, dInput)
```

Arguments

`dmatrix`	Matrix holding the density values on dInput - assumed to be strictly positive and integrate to 1
`alpha`	Scalar to regularise the supports with (default=0)
`dInput`	Support (grid) for Density domain

Value

list with the 'LQD' projected data, the 'alpha' used and the 'newSupport' that was defined after trimming the data.

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2015

`RegulariseByAlpha` *Function to regularise non-strictly positive densities by a scalar alpha*

Description

Completely uniform distributions cannot be regularised by a level alpha.

Usage

```
RegulariseByAlpha(x, y, alpha = 0.01, deregularise = FALSE)
```

Arguments

`x`	support of the density
`y`	values of the density
`alpha`	scalar to regularise with (default = 0.01)
`deregularise`	logical to deregularise by alpha (instead of regularising) (default: FALSE)

Value

dens density values on dSup

Examples

```
x = seq(0,1,length.out =122)
y = seq(0,2,length.out =122)
z = RegulariseByAlpha(x=x, y=y, alpha = 0.1)
```

Index