Package ‘fdadensity’

March 29, 2025

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

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

Type Package

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

Version 0.1.4

Date 2025-03-28

Maintainer Alexander Petersen <petersen@stat.byu.edu>

Description An implementation of the methodology described in Petersen and Mueller (2016) <doi:10.1214/15-AOS1363> for the functional data analysis of samples of density functions. Densities are first transformed to their corresponding log quantile densities, followed by ordinary Functional Principal Components Analysis (FPCA). Transformation modes of variation yield improved interpretation of the variability in the data as compared to FPCA on the densities themselves. The standard fraction of variance explained (FVE) criterion commonly used for functional data is adapted to the transformation setting, also allowing for an alternative quantification of variability for density data through the Wasserstein metric of optimal transport.

Depends R (>= 3.3.0)

License BSD_3_clause + file LICENSE

LazyData false

Imports Rcpp (>= 0.11.5), fdapace (>= 0.3.0)

LinkingTo Rcpp

NeedsCompilation yes

Suggests testthat

RoxygenNote 7.3.2

Encoding UTF-8

Author A. Petersen [aut],
P. Z. Hadjipantelis [aut],
H.G. Mueller [aut],
Alexander Petersen [cre]

Repository CRAN

Date/Publication 2025-03-29 00:30:09 UTC

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

---

CreateModeOfVarPlotLQ2D
Transformation Mode of Variation Plot

Description

Create the k-th transformation mode of variation plot.

Usage

```
CreateModeOfVarPlotLQ2D(
  fpcaObj,
  domain = "D",
  k = 1,
  dSup = NULL,
  Qvec = -2:2,
  alpha = 0,
  useAlpha = FALSE,
  ...
)
```

Arguments

`fpcaObj`	An FPCA class object returned by FPCA() on the log quantile density functions.
`domain`	should the mode be plotted in LQD ('Q') or density space ('D', the default).
`k`	The k-th mode of variation to plot (default k = 1)
`dSup`	The common support of the original densities. Only relevant for `domain = 'D'`
`Qvec`	Vector of values Q to be plotted. If 0 is not included, it will be added (default is -2:2). Only relevant for `domain = 'D'`
`alpha`	(De)regularisation parameter (default is 0). See details.
`useAlpha`	logical - should deregularisation be performed? Default:FALSE
`...`	Additional arguments for the 'plot' function.

Details

If `domain = 'D'` (the default), the a transformation mode of variation is plotted. The red-line is $\psi^{-1}(\nu)$, where ν is the mean in LQD space and ψ is the LQD transformation. Other lines correspond to perturbations by adding multiples of the LQD eigenfunctions ρ_k (with eigenvalues τ_k) : $\psi^{-1}(\nu + Q\sqrt{\tau_k}\rho_k)$ for the values Q in `Qvec`. If `alpha` is positive, will attempt to deregularise (see `DeregulariseByAlpha`). This will throw an error if alpha is too large.

If `domain = 'Q'`, ordinary modes of variation are plotted in LQD space (see documentation for `CreateModeOfVarPlot` in `fdapace`).

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2016

See Also

`DeregulariseByAlpha`

Examples

```
## Densities for Top 50 Male Baby Names
data(Top50BabyNames)
x = Top50BabyNames$x

# Perform Transformation FPCA for male baby name densities
X = FPCAdens(dmatrix = t(Top50BabyNames$dens$male), dSup = Top50BabyNames$x, useAlpha = TRUE,
                  optns = list(dataType = 'Dense', error = FALSE, methodSelectK = 2))

# Plot Modes

Qvec = quantile(X$xiEst[,1], probs = c(0.1, 0.25, 0.75, 0.9))/sqrt(X$lambda[1])
CreateModeOfVarPlotLQ2D(X, k = 1, dSup = x, Qvec = Qvec, main = 'First Mode, Density Space')
CreateModeOfVarPlotLQ2D(X, domain = 'Q', k = 1, dSup = x, Qvec = Qvec,
                            main = 'First Mode, LQD space')

Qvec = quantile(X$xiEst[,2], probs = c(0.1, 0.25, 0.75, 0.9))/sqrt(X$lambda[2])
CreateModeOfVarPlotLQ2D(X, k = 2, dSup = x, Qvec = Qvec, main = 'Second Mode, Density Space')
CreateModeOfVarPlotLQ2D(X, domain = 'Q', k = 2, dSup = x, Qvec = Qvec,
                            main = 'Second Mode, LQD space')
```

`dens2lqd` *Function for converting densities to log quantile density functions*

Description

Function for converting densities to log quantile density functions

Usage

```
dens2lqd(dens, dSup, N = length(dSup), lqdSup = 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 lqd function; default length(dSup)
`lqdSup`	support for lqd domain - must begin at 0 and end at 1; default [0,1] with N-equidistant support points

Value

lqd log quantile density on lqdSup

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2016

See Also

`normaliseDensities`

Examples

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

`dens2qd` *Function for converting Densities to Quantile Densities*

Description

Function for converting Densities to Quantile Densities

Usage

```
dens2qd(
  dens,
  dSup = seq(0, 1, length.out = length(dens)),
  qdSup = seq(0, 1, length.out = length(dens)),
  useSplines = TRUE
)
```

Arguments

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

Value

qd quantile density values on qdSup

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2016

See Also

`normaliseDensities`

Examples

```
x <- seq(0,2,length.out =512)
y <- rep(0.5,length.out =512)
y.qd <- dens2qd(dens=y, dSup = x) # should equate # 2
```

`dens2quantile` *Function for converting Densities to Quantile Functions*

Description

Function for converting Densities to Quantile Functions

Usage

```
dens2quantile(
  dens,
  dSup = seq(0, 1, length.out = length(dens)),
  qSup = seq(0, 1, length.out = length(dens)),
  useSplines = TRUE
)
```

Arguments

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

Value

Q quantile function on qSup

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2016

See Also

`normaliseDensities`

Examples

```
x <- seq(0,2,length.out =512)
y <- rep(0.5,length.out =512)
y.quantile <- dens2quantile(dens=y, dSup = x) # should equate # 2*seq(0, 1, length.out = 512)
```

`DeregulariseByAlpha` *Function to deregularise densities to have (smaller) minimum value*

Description

If possible, deregularises the input density `y` to have minimum density value is `alpha`. See details.

Usage

```
DeregulariseByAlpha(x, y, alpha = 0)
```

Arguments

`x`	support of the density
`y`	values of the density
`alpha`	scalar to deregularise with (default = 0) - this will be the minimum value of the deregularised density, unless `min(y)` < `alpha`, in which case no deregularisation will be performed

Details

If `min(y)` <= `alpha`, or `y` is the uniform distribution, no deregularisation is performed and `y` is returned. If `min(y)*diff(range(x))` > 1, the deregularisation is not possible and an error is thrown. Otherwise, the deregularised density in an inverse manner to `RegulariseByAlpha`.

Value

dens density values on x

See Also

`RegulariseByAlpha`

Examples

```
 x = seq(0,1,length.out=122)
 y = seq(0.1,1.9,length.out=122)
 z = DeregulariseByAlpha(x=x, y=y, alpha = 0)
```

`FPCAdens` *FPCA for densities by log quantile density transformation*

Description

Perform FPCA on LQD-transformed densities

Usage

```
FPCAdens(
  dmatrix,
  dSup,
  lqdSup = seq(0, 1, length.out = length(dSup)),
  useAlpha = FALSE,
  alpha = 0.01,
  optns = list(dataType = "Dense", error = FALSE)
)
```

Arguments

`dmatrix`	Matrix holding the density values on dSup - all rows must be strictly positive and integrate to 1
`dSup`	Support (grid) for Density domain
`lqdSup`	Support grid for lqd domain (default = seq(0, 1, length.out = length(dSup)))
`useAlpha`	should regularisation be performed (default=FALSE)
`alpha`	Scalar to regularise the supports with (default=0.01)
`optns`	A list of options for FPCA. See documentation for `fdapace::FPCA`.

Details

Densities are transformed to log-quantile densities, followed by standard FPCA. If `useAlpha = TRUE`, densities are regularized before transformation

Value

X - a list containing the output of `fdapace::FPCA` in log-quantile density space. See documentation of `fdapace::FPCA` for descriptions of the variables in X

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2016

See Also

`RegulariseByAlpha`, `lqd2dens`, `MakeLQDsample`, `FPCA`

Examples

```
## Densities for Top 50 Female Baby Names
data(Top50BabyNames)

# Perform Transformation FPCA for male baby name densities
X = FPCAdens(dmatrix = t(Top50BabyNames$dens$female), dSup = Top50BabyNames$x, useAlpha = TRUE,
                  optns = list(dataType = 'Dense', error = FALSE, methodSelectK = 2))
x = Top50BabyNames$x

# Plot Modes

Qvec = quantile(X$xiEst[,1], probs = c(0.1, 0.25, 0.75, 0.9))/sqrt(X$lambda[1])
CreateModeOfVarPlotLQ2D(X, k = 1, dSup = x, Qvec = Qvec, main = 'First Mode, Density space')
CreateModeOfVarPlotLQ2D(X, domain = 'Q', k = 1, dSup = x, Qvec = Qvec,
                            main = 'First Mode, LQD space')

Qvec = quantile(X$xiEst[,2], probs = c(0.1, 0.25, 0.75, 0.9))/sqrt(X$lambda[2])
CreateModeOfVarPlotLQ2D(X, k = 2, dSup = x, Qvec = Qvec, main = 'Second Mode, Density Space')
CreateModeOfVarPlotLQ2D(X, domain = 'Q', k = 2, dSup = x, Qvec = Qvec,
                            main = 'Second Mode, LQD space')
```

`GetFVE` *Compute Metric-based Fraction of Variance Explained*

Description

When FPCA is performed on the log quantile density functions, the fraction of variance explained by the first K components is computed based on the density reconstruction and chosen metric.

Usage

```
GetFVE(fpcaObj, dmatrix, dSup, metric = "L2", useAlpha = FALSE, alpha = 0.01)
```

Arguments

`fpcaObj`	PACE output (FPCA on LQDs)
`dmatrix`	matrix of original densities measures on grid dSup, rows correspond to individual densities
`dSup`	support for Density domain - max and min mark the boundary of the support.
`metric`	metric for measuring variance - 'L2' for Euclidean or 'W' for Wasserstein
`useAlpha`	should regularisation be performed to densities in dmatrix? This should be set to TRUE if densities were regularised prior to FPCA (default = FALSE)
`alpha`	scalar to regularise before computing FVE. If useAlpha = TRUE, this should match the value used to regularise prior to FPCA (default = 0.01)

Details

The fraction of variance explained (FVE) by the first K principal components corresponding to the LQD functions is computed by taking the K-dimensional LQD representations, transforming back to densities, and comparing the reconstruction to the original densities using the chosen metric. If densities were regularised prior to transformation and FPCA, the same regularisation parameters should be used here.

Value

FVEvector

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2016

See Also

`RegulariseByAlpha`,`lqd2quantile`

Examples

```
data(Top50BabyNames)

# Perform Transformation FPCA for male baby name densities
dSup = Top50BabyNames$x
X = FPCAdens(dmatrix = t(Top50BabyNames$dens$male), dSup = dSup, useAlpha = TRUE,
                   optns = list(dataType = 'Dense', error = FALSE, methodSelectK = 8))

# Compute FVE - must compare to regularized densities

fveL2 = GetFVE(fpcaObj = X, dmatrix = t(Top50BabyNames$dens$male), dSup = dSup, useAlpha = TRUE)
fveW = GetFVE(fpcaObj = X, dmatrix = t(Top50BabyNames$dens$male), dSup = dSup,
                   metric = 'W', useAlpha = TRUE)
```

`getWFmean` *Wasserstein Frechet Mean Computation*

Description

Function for computing the Wasserstein Frechet mean through quantile density averaging

Usage

```
getWFmean(
  dmatrix,
  dSup,
  N = length(dSup),
  qdSup = seq(0, 1, length.out = N),
  useAlpha = FALSE,
  alpha = 0.01
)
```

Arguments

`dmatrix`	matrix of density values on dSup - must be strictly positive and each row must integrate to 1
`dSup`	support (grid) for Density domain
`N`	desired number of points on a [0,1] grid for quantile density functions; default length(dSup)
`qdSup`	support for LQ domain - must begin at 0 and end at 1; default [0,1] with N-equidistant support points
`useAlpha`	should regularisation be performed (default=FALSE)
`alpha`	Scalar to regularise the supports with (default=0.01)

Value

wfmean the Wasserstein-Frechet mean density

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2016

Examples

```
x <- seq(0,1,length.out = 101)
# linear densities on (0, 1)
y <- t(sapply(seq(0.5, 1.5, length.out = 10), function(b) b + 2*(1 - b)*x))
wfmean = getWFmean(y, x)

# Plot WF mean with Euclidean Mean
matplot(x, t(y), ylab = 'Density', type = 'l', lty = 1, col = 'black')
lines(x, wfmean, lwd = 2, col = 'red')
lines(x, colMeans(y), lwd = 2, col = 'blue')
legend('topright', col = c('black', 'red', 'blue'), lwd = c(1, 2, 2),
          legend = c('Densities', 'WF Mean', 'Euclidean Mean'))
```

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

Description

Function for converting log quantile densities to densities

Usage

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

Arguments

`lqd`	log quantile density on lqdSup
`lqdSup`	support forlqd domain - must begin at 0 and end at 1
`dSup`	support for Density domain - max and min values 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, 2016

Examples

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

`lqd2quantile` *Function for converting log quantile densities to quantile functions*

Description

Function for converting log quantile densities to quantile functions

Usage

```
lqd2quantile(
  lqd,
  lqdSup = seq(0, 1, length.out = length(lqd)),
  lb = 0,
  useSplines = TRUE
)
```

Arguments

`lqd`	log quantile density on lqdSup
`lqdSup`	support for lqd domain - must begin at 0 and end at 1
`lb`	lower bound of support for Density domain - default is 0.
`useSplines`	fit spline to the lqd when doing the numerical integration (default: TRUE)

Value

quantile values on lqdSup

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2016

Examples

```
x <- seq(1,3,length.out =512)
y.lqd <- rep(log(2), times = 512)
y <- lqd2quantile(lqd = y.lqd, lb = 1) # should equate # seq(1, 3, length.out = 512)
```

`MakeDENsample` *Convenience function for converting log quantile densities to densities*

Description

See 'lqd2dens' and 'DeregulariseByAlpha' for more details. This function transforms the log quantile densities in 'qmatrix' to density functions, optionally followed by deregularisation.

Usage

```
MakeDENsample(
  qmatrix,
  lqdSup = seq(0, 1, length.out = ncol(qmatrix)),
  dSup = seq(0, 1, length.out = ncol(qmatrix)),
  useAlpha = FALSE,
  alpha = 0
)
```

Arguments

`qmatrix`	Matrix holding the log quantile density values on [0,1]
`lqdSup`	Support grid for input log quantile densities (default = seq(0, 1, length.out = ncol(qmatrix)))
`dSup`	Support grid for output densities (default = seq(0, 1, length.out = ncol(qmatrix)))
`useAlpha`	Logical indicator to deregularise the densities (default = FALSE)
`alpha`	Scalar to deregularise the density - where possible, this will be the minimum value for the deregularised densities (default=0)

Value

list with the 'DEN' transformed data, and 'dSup' that matches the input argument.

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2016

See Also

`DeregulariseByAlpha`, `lqd2dens`

Examples

```
x <- seq(0,1,length.out = 101)
# linear densities on (0, 1)
y <- t(sapply(seq(0.5, 1.5, length.out = 10), function(b) b + 2*(1 - b)*x))

# Get LQDs

y.lqd = MakeLQDsample(dmatrix = y, dSup = x)
matplot(y.lqd$lqdSup, t(y.lqd$LQD), ylab = 'LQD', type = 'l', lty = 1, col = 'black')

# Get Densities Back

y.dens = MakeDENsample(y.lqd$LQD, lqdSup = x, dSup = x) # should equate to y above
# These should look the same
matplot(y.dens$dSup, t(y.dens$DEN), ylab = 'Density', type = 'l', lty = 1, col = 'blue')
matplot(x, t(y), ylab = 'Original Density', type = 'l', lty = 1, col = 'red')
```

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

Description

See 'dens2lqd' and 'RegulariseByAlpha' for more details. This function first (transforms the densities in 'dmatrix' to log quantile density functions, optionally followed by regularisation.

Usage

```
MakeLQDsample(
  dmatrix,
  dSup,
  lqdSup = seq(0, 1, length.out = length(dSup)),
  useAlpha = FALSE,
  alpha = 0.01
)
```

Arguments

`dmatrix`	Matrix holding the density values on dSup - all rows must be strictly positive and integrate to 1
`dSup`	Support (grid) for Density domain
`lqdSup`	Support grid for lqd domain (default = seq(0, 1, length.out = length(dSup)))
`useAlpha`	should regularisation be performed (default=FALSE)
`alpha`	Scalar to regularise the supports with (default=0.01)

Value

list with 'LQD', a matrix of log quantile density functions, and 'lqdSup' that matches the input argument

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2016

See Also

`RegulariseByAlpha`,`dens2lqd`

Examples

```
x <- seq(0,1,length.out = 101)
# some log quantile densities on (0, 1)
y <- t(sapply(seq(0.5, 1.5, length.out = 10), function(b) -log(b^2 + 4*(1-b)*x)/2))

# Get densities

y.dens = MakeDENsample(qmatrix = y, lqdSup = x, dSup = x)$DEN
matplot(x, t(y.dens), ylab = 'Density', type = 'l', lty = 1, col = 'black')

# Get LQDs Back

y.lqd = MakeLQDsample(y.dens, lqdSup = x, dSup = x)
# These should match
matplot(y.lqd$lqdSup, t(y.lqd$LQD), ylab = 'LQD', type = 'l', lty = 1, col = 'blue')
matplot(x, t(y), ylab = 'LQD', type = 'l', lty = 1, col = 'red')
```

`normaliseDensities` *Normalise Densities*

Description

Preprocessing function to ensure densities integrate to 1

Usage

```
normaliseDensities(dmatrix, dSup = 1:ncol(dmatrix))
```

Arguments

`dmatrix`	Matrix with rows representing distinct densities on dSup - all entries must be nonnegative
`dSup`	Support (grid) for Density domain

Details

Uses trapezoidal integration to normalise the densities to have integral 1

Value

matrix 'dmatrix' consisting of rows of input of the same name that have been normalised to have integral 1

References

Functional Data Analysis for Density Functions by Transformation to a Hilbert space, Alexander Petersen and Hans-Georg Mueller, 2016

Examples

```
## Normalise collection of truncated normal densities

mu <- seq(-2, 2, by = 0.5)
dSup = seq(-3, 3, length.out = 101)
y <- t(sapply(mu, function(m) dnorm(x = dSup, mean = m)))

# Should return warnings about densities not integrating to 1
lqd = MakeLQDsample(dmatrix = y, dSup = dSup)

# Normalise and rerun without warning
dens <- normaliseDensities(dmatrix = y, dSup = dSup)
lqd = MakeLQDsample(dmatrix = dens, dSup = dSup)
```

`qd2dens` *Function for converting Quantile Densities to Densities*

Description

Function for converting Quantile Densities to Densities

Usage

```
qd2dens(
  qd,
  qdSup = seq(0, 1, length.out = length(qd)),
  dSup,
  useSplines = TRUE
)
```

Arguments

`qd`	quantile density on qdSup
`qdSup`	support for quantile domain - must begin at 0 and end at 1 (default = seq(0, 1, length.out = length(qd)))
`dSup`	support for Density domain - max and min values mark the boundary of the support.
`useSplines`	fit spline to the qd 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, 2016

Examples

```
x <- seq(0,1,length.out =512)
y <- rep(2,length.out =512)
y.dens <- qd2dens(qd=y, qdSup = x, dSup = seq(0, 2, length.out = 512)) # should equate # 1/2
```

`RegulariseByAlpha` *Function to regularise densities to have (larger) minimum value*

Description

If possible, regularises the input density `y` to have minimum density value is `alpha`. See details.

Usage

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

Arguments

`x`	support of the density
`y`	values of the density
`alpha`	scalar to regularise with (default = 0.01) - this will be the minimum value of the regularised density, unless `min(y)` > `alpha`, in which case no regularisation will be performed

Details

If `min(y)` >= `alpha` or `y` is the uniform distribution, no regularisation is performed and `y` is returned. If `alpha*diff(range(x))` > `1`, the regularisation is not possible and an error is thrown. Otherwise, the regularised density is computed by adding an appropriate constant `gam` `y`, followed by renormalisation to have integral 1.

Value

dens density values on x

See Also

`DeregulariseByAlpha`,`normaliseDensities`

Examples

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

`Top50BabyNames` *Baby name popularity densities for 50 male and 50 female names in the USA*

Description

Baby name popularity densities, obtained by smoothing year-to-year popularity indices from 1950 to 2016, after normalization to have integral equal to 1. The top 50 names, in absolute popularity, are included for each gender.

Format

A list with two variables

x grid of years between 1950 and 2016, of length 67.

dens list of length two, corresponding to male (`dens$male`) and female(`dens$female`) names. Each is a 67-by-50 matrix of density estimates, where each column corresponds to a unique baby name given by the corresponding column name.

References

Data from the `R` package `babynames`, originally from the US Social Security Administration

Index