Package 'NPFD'

June 7, 2026

Title N-Power Fourier Deconvolution

Version 1.0.1

Description Provides tools for non-parametric Fourier deconvolution using the N-Power Fourier Deconvolution (NPFD) method. This package includes methods for density estimation (densprf()) and sample generation (createSample()), enabling users to perform statistical analyses on mixed or replicated data sets.

Depends R (>= 3.5.0)

Imports siggenes, KernSmooth, splines, stats, graphics, VGAM

License GPL-3

Encoding UTF-8

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation no

Author Akin Anarat [aut, cre]

Maintainer Akin Anarat <akin.anarat@hhu.de>

Repository CRAN

Date/Publication 2026-06-07 11:30:02 UTC

Contents

createSample *Create a Sample from a Centered Distribution*

Description

This function creates a sample from a centered distribution based on replicates of mixed data.

Usage

```
createSample(z1, z2)
```

Arguments

z1	A numeric vector where $z_1 = x_1 + y$.
z2	A numeric vector of the same length as z_1 where $z_2 = x_2 + y$.

Value

A numeric vector representing a sample from the centered distribution.

Examples

```
# Set seed for reproducibility
set.seed(123)

# Generate random data
x1 <- rnorm(1000)
x2 <- rnorm(1000)
y <- rgamma(1000, 10, 2)
z1 <- x1 + y
z2 <- x2 + y

# Use createSample to generate a sample
x <- createSample(z1, z2)

# Perform density estimation
f.x <- stats::density(x, adjust = 1.5)
x.x <- f.x$x
f <- dnorm(x.x)

# Plot the results
plot(NULL, xlim = range(f.x$x), ylim = c(0, max(f, f.x$y)), xlab = "x", ylab = "Density")
lines(x.x, f, col = "blue", lwd = 2)
lines(f.x, col = "orange", lwd = 2)
legend("topright", legend = c(expression(f), expression(f[x])), col = c("blue", "orange"), lwd = 2)
```

deconvolve *N-Power Fourier Deconvolution*

Description

Estimates the density f_y, given vectors x and z, where f_z results from the convolution of f_x and f_y.

Usage

```
deconvolve(
  x = NULL,
  z,
  mode = c("empirical", "denspr"),
  dfx = 5,
  dfz = 5,
  Lx = 100,
  Lz = 100,
  Ly = 100,
  ymin = NULL,
  ymax = NULL,
  N = 1:100,
  FT.grid = seq(0, 100, 0.1),
  lambda = 1,
  eps = 10^-3,
  delta = 10^-2,
  error = c("unknown", "normal", "laplacian"),
  sigma = NULL,
  calc.error = FALSE,
  plot = FALSE,
  legend = TRUE,
  positive = FALSE
)
```

Arguments

x	Vector of observations for x.
z	Vector of observations for z.
mode	Deconvolution mode (empirical or denspr). If empirical, the Fourier transforms of x and z are estimated using the empirical form. If denspr, they are calculated based on the density estimations using densprf (see the package **siggenes**).
dfx	Degrees of freedom for the estimation of f_x if mode is set to denspr.
dfz	Degrees of freedom for the estimation of f_z if mode is set to denspr.
Lx	Number of points for f_x-grid if mode is set to denspr.
Lz	Number of points for f_z-grid if mode is set to denspr.

Ly	Number of points for f_y-grid.
ymin	Minimum for f_y-grid.
ymax	Maximum for f_y-grid.
N	Possible power values.
FT.grid	Vector of grid for Fourier transformation of f_x and f_z.
lambda	Smoothing parameter.
eps	Tolerance for convergence.
delta	Small margin value.
error	Error model (unknown, normal, laplacian). If unknown, the Fourier transform of x is calculated based on the mode. If normal, the exact form of the Fourier transform of a centered normal distribution with standard deviation sigma is used for x. If laplacian, the exact form of the Fourier transform of a centered Laplace distribution with standard deviation sigma is used for x.
sigma	Standard deviation for normal or Laplacian error.
calc.error	Logical indicating whether to calculate error (10 x ISE between f_z and $f_x * f_y$).
plot	Logical indicating whether to plot f_z vs. $f_x * f_y$ if calc.error is TRUE.
legend	Logical indicating whether to include a legend in the plot if calc.error is TRUE.
positive	Logical indicating whether to enforce non-negative density estimation.

Value

A list with the following components:

x	A vector of x-values of the resulting density estimation.
y	A vector of y-values of the resulting density estimation.
N	The power used in the deconvolution process.
error	The calculated error if calc.error = TRUE.

Author(s)

Akin Anarat <akin.anarat@hhu.de>

References

Anarat A., Krutmann, J., and Schwender, H. (2024). A nonparametric statistical method for deconvolving densities in the analysis of proteomic data. Submitted.

Examples

```
# Deconvolution when mixed data and data from an independent experiment are provided:
set.seed(123)
x <- rnorm(1000)
y <- rgamma(1000, 10, 2)
z <- x + y
```

```
f <- function(x) dgamma(x, 10, 2)

independent.x <- rnorm(100)

fy.NPFD <- deconvolve(independent.x, z, calc.error = TRUE, plot = TRUE)
x.x <- fy.NPFD$x
fy <- f(x.x)

# Check power and error values
fy.NPFD$N
fy.NPFD$error

# Plot density functions
plot(NULL, xlim = range(y), ylim = c(0, max(fy, fy.NPFD$y)), xlab = "x", ylab = "Density")
lines(x.x, fy, col = "blue", lwd = 2)
lines(fy.NPFD, col = "orange", lwd = 2)
legend("topright", legend = c(expression(f[y]), expression(f[y]^{NPFD})),
       col = c("blue", "orange"), lwd = c(2, 2))

# For replicated mixed data:
set.seed(123)
x1 <- VGAM::rlaplace(1000, 0, 1/sqrt(2))
x2 <- VGAM::rlaplace(1000, 0, 1/sqrt(2))
y <- rgamma(1000, 10, 2)
z1 <- z <- x1 + y
z2 <- x2 + y

x <- createSample(z1, z2)

fy.NPFD <- deconvolve(x, z, mode = "denspr", calc.error = TRUE, plot = TRUE)
x.x <- fy.NPFD$x
fy <- f(x.x)

# Check power and error values
fy.NPFD$N
fy.NPFD$error

# Plot density functions
plot(NULL, xlim = range(y), ylim = c(0, max(fy, fy.NPFD$y)), xlab = "x", ylab = "Density")
lines(x.x, fy, col = "blue", lwd = 2)
lines(fy.NPFD, col = "orange", lwd = 2)
legend("topright", legend = c(expression(f[y]), expression(f[y]^{NPFD})),
       col = c("blue", "orange"), lwd = c(2, 2))

# When the distribution of x is asymmetric and the sample size is very small:
set.seed(123)
x <- rgamma(5, 4, 2)
y <- rgamma(1000, 10, 2)
z <- x + y

fy.NPFD <- deconvolve(x, z, mode = "empirical", lambda = 2)
x.x <- fy.NPFD$x
fy <- f(x.x)
```

```
# Check power value
fy.NPFD$N

# Plot density functions
plot(NULL, xlim = range(y), ylim = c(0, max(fy, fy.NPFD$y)), xlab = "x", ylab = "Density")
lines(x.x, fy, col = "blue", lwd = 2)
lines(fy.NPFD, col = "orange", lwd = 2)
legend("topright", legend = c(expression(f[y]), expression(f[y]^{NPFD})),
       col = c("blue", "orange"), lwd = c(2, 2))
```

---

densprf *Density Estimation Function*

Description

This function estimates the density using a Poisson GLM with natural splines.

Usage

```
densprf(
  x,
  n.interval = NULL,
  df = 5,
  knots.mode = TRUE,
  type.nclass = c("wand", "scott", "FD"),
  addx = FALSE
)
```

Arguments

x	Input data vector.
n.interval	Number of intervals (optional).
df	Degrees of freedom for the splines.
knots.mode	Boolean to determine if quantiles should be used for knots.
type.nclass	Method for determining number of classes.
addx	Add x values (optional).

Details

densprf is a modification of the denspr function from the **siggenes** package.

For more details, see the documentation in the **siggenes** package.

Value

The function densprf(x) returns a function that, for a given input z, computes the estimated density evaluated at the position values of z as a result.

Examples

```
# Set seed for reproducibility
set.seed(123)

# Generate random data
z <- rnorm(1000)

# Apply densprf function
f <- densprf(z)

# Define sequences for evaluation
x1 <- seq(-4, 4, 0.5)
x2 <- seq(-5, 5, 0.1)

# Evaluate the density function at specified points
f1 <- f(x1)
f2 <- f(x2)
```

Index