Package 'rbbnp'

April 30, 2025

Type Package

Title A Bias Bound Approach to Non-Parametric Inference

Version 0.3.0

Maintainer Xinyu DAI <xinyu_dai@brown.edu>

Description A novel bias-bound approach for non-parametric inference is introduced, focusing on both density and conditional expectation estimation. It constructs valid confidence intervals that account for the presence of a non-negligible bias and thus make it possible to perform inference with optimal mean squared error minimizing bandwidths. This package is based on Schennach (2020) <doi:10.1093/restud/rdz065>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

URL https://doi.org/10.1093/restud/rdz065

Imports purrr, pracma, tidyr, dplyr, ggplot2, gridExtra

RoxygenNote 7.3.2

Depends R (>= 3.5)

NeedsCompilation no

Author Xinyu DAI [aut, cre], Susanne M Schennach [aut]

Repository CRAN

Date/Publication 2025-04-30 19:20:01 UTC

Contents

`biasBound_condExpectation` *Bias bound approach for conditional expectation estimation*

Description

Estimates the density at a given point or across a range, and provides visualization options for density, bias, and confidence intervals.

Usage

```
biasBound_condExpectation(
  Y,
  X,
  x = NULL,
  h = NULL,
  h_method = "cv",
  alpha = 0.05,
  est_Ar = NULL,
  resol = 100,
  xi_lb = NULL,
  xi_ub = NULL,
  methods_get_xi = "Schennach",
  if_plot_ft = FALSE,
  ora_Ar = NULL,
  if_plot_conditional_mean = TRUE,
  kernel.fun = "Schennach2004",
  if_approx_kernel = TRUE,
  kernel.resol = 1000
)
```

Arguments

`Y`	A numerical vector of sample data.
`X`	A numerical vector of sample data.
`x`	Optional. A scalar or range of points where the density is estimated. If NULL, a range is automatically generated.
`h`	A scalar bandwidth parameter. If NULL, the bandwidth is automatically selected using the method specified in 'h_method'.
`h_method`	Method for automatic bandwidth selection when h is NULL. Options are "cv" (cross-validation) and "silverman" (Silverman's rule of thumb). Default is "cv".
`alpha`	Confidence level for intervals. Default is 0.05.
`est_Ar`	Optional list of estimates for A and r. If NULL, they are computed using `get_est_Ar()`.
`resol`	Resolution for the estimation range. Default is 100.
`xi_lb`	Optional. Lower bound for the interval of Fourier Transform frequency xi. Used for determining the range over which A and r is estimated. If NULL, it is automatically determined based on the methods_get_xi.
`xi_ub`	Optional. Upper bound for the interval of Fourier Transform frequency xi. Similar to xi_lb, it defines the upper range for A and r estimation. If NULL, the upper bound is determined based on the methods_get_xi.
`methods_get_xi`	A string specifying the method to automatically determine the xi interval if xi_lb and xi_ub are NULL. Options are "Schennach" and "Schennach_loose". If "Schennach" the range is selected based on the Theorem 2 in Schennach2020, if "Schennach_loose", it is defined by the initial interval given in Theorem 2 without selecting the xi_n.

`if_plot_ft` Logical. If TRUE, plots the Fourier transform.

`ora_Ar` Optional list of oracle values for A and r.

`if_plot_conditional_mean` Logical. If TRUE, plots the conditional mean estimation.

`kernel.fun` A string specifying the kernel function to be used. Options are "Schennach2004", "sinc", "normal", "epanechnikov".

`if_approx_kernel` Logical. If TRUE, uses approximations for the kernel function.

`kernel.resol` The resolution for kernel function approximation. See `fun_approx`.

Value

A list containing various outputs including estimated values, plots, and intervals.

Examples

```
# Example 1: point estimation of conditional expectation of Y on X
biasBound_condExpectation(
 Y = sample_data$Y,
 X = sample_data$X,
 x = 1,
 h = 0.09,
 kernel.fun = "Schennach2004"
)

# Example 2: conditional expectation with automatic bandwidth selection using cross-validation
# biasBound_condExpectation(
# Y = sample_data$Y,
#  X = sample_data$X,
#  h = NULL,
#  h_method = "cv",
#  xi_lb = 1,
#  xi_ub = 12,
#  kernel.fun = "Schennach2004"
# )

# Example 3: conditional expectation with automatic bandwidth selection using Silverman's rule
# biasBound_condExpectation(
# Y = sample_data$Y,
#  X = sample_data$X,
#  h = NULL,
#  h_method = "silverman",
#  methods_get_xi = "Schennach",
#  if_plot_ft = TRUE,
#  kernel.fun = "Schennach2004"
# )
```

biasBound_density *Bias bound approach for density estimation*

Description

Estimates the density at a given point or across a range, and provides visualization options for density, bias, and confidence intervals.

Usage

```
biasBound_density(
  X,
  x = NULL,
  h = NULL,
  h_method = "cv",
  alpha = 0.05,
  resol = 100,
  xi_lb = NULL,
  xi_ub = NULL,
  methods_get_xi = "Schennach",
  if_plot_density = TRUE,
  if_plot_ft = FALSE,
  ora_Ar = NULL,
  kernel.fun = "Schennach2004",
  if_approx_kernel = TRUE,
  kernel.resol = 1000
)
```

Arguments

`X`	A numerical vector of sample data.
`x`	Optional. A scalar or range of points where the density is estimated. If NULL, a range is automatically generated.
`h`	A scalar bandwidth parameter. If NULL, the bandwidth is automatically selected using the method specified in 'h_method'.
`h_method`	Method for automatic bandwidth selection when h is NULL. Options are "cv" (cross-validation) and "silverman" (Silverman's rule of thumb). Default is "cv".
`alpha`	Confidence level for intervals. Default is 0.05.
`resol`	Resolution for the estimation range. Default is 100.
`xi_lb`	Optional. Lower bound for the interval of Fourier Transform frequency xi. Used for determining the range over which A and r is estimated. If NULL, it is automatically determined based on the methods_get_xi.
`xi_ub`	Optional. Upper bound for the interval of Fourier Transform frequency xi. Similar to xi_lb, it defines the upper range for A and r estimation. If NULL, the upper bound is determined based on the methods_get_xi.

methods_get_xi — A string specifying the method to automatically determine the xi interval if xi_lb and xi_ub are NULL. Options are "Schennach" and "Schennach_loose". If "Schennach" the range is selected based on the Theorem 2 in Schennach2020, if "Schennach_loose", it is defined by the initial interval given in Theorem 2 without selecting the xi_n.

if_plot_density — Logical. If TRUE, plots the density estimation.

if_plot_ft — Logical. If TRUE, plots the Fourier transform.

ora_Ar — Optional list of oracle values for A and r.

kernel.fun — A string specifying the kernel function to be used. Options are "Schennach2004", "sinc", "normal", "epanechnikov".

if_approx_kernel — Logical. If TRUE, uses approximations for the kernel function.

kernel.resol — The resolution for kernel function approximation. See fun_approx.

Value

A list containing various outputs including estimated values, plots, and intervals.

Examples

```
# Example 1: Specifying x for point estimation with manually selected xi range
# from a fixed bandwidth
biasBound_density(
  X = sample_data$X,
  x = 1,
  h = 0.09,
  xi_lb = 1,
  xi_ub = 12,
  if_plot_ft = TRUE,
  kernel.fun = "Schennach2004"
)

# Example 2: Density estimation with automatic bandwidth selection using cross-validation
# biasBound_density(
#   X = sample_data$X,
#   h = NULL,
#   h_method = "cv",
#   xi_lb = 1,
#   xi_ub = 12,
#   if_plot_ft = FALSE,
#   kernel.fun = "Schennach2004"
# )

# Example 3: Density estimation with automatic bandwidth selection using Silverman's rule
#  biasBound_density(
#   X = sample_data$X,
#   h = NULL,
#   h_method = "silverman",
#   methods_get_xi = "Schennach",
```

```
#   if_plot_ft = TRUE,
#   kernel.fun = "Schennach2004"
# )
```

---

create_biasBound_config
Create a configuration object for bias bound estimations

Description

Create a configuration object for bias bound estimations

Usage

```
create_biasBound_config(
  X,
  Y = NULL,
  h = NULL,
  h_method = "cv",
  use_fft = TRUE,
  alpha = 0.05,
  resol = 100,
  xi_lb = NULL,
  xi_ub = NULL,
  methods_get_xi = "Schennach",
  kernel.fun = "Schennach2004",
  if_approx_kernel = TRUE,
  kernel.resol = 1000
)
```

Arguments

X	A numerical vector of sample data.
Y	Optional. A numerical vector of sample data for conditional expectation.
h	A scalar bandwidth parameter. If NULL, the bandwidth is automatically selected using the method specified in 'h_method'.
h_method	Method for automatic bandwidth selection when h is NULL. Options are "cv" (cross-validation) and "silverman" (Silverman's rule of thumb). Default is "cv".
use_fft	Ignored. Maintained for backward compatibility.
alpha	Confidence level for intervals.
resol	Resolution for the estimation range.
xi_lb	Lower bound for the interval of Fourier Transform frequency.
xi_ub	Upper bound for the interval of Fourier Transform frequency.
methods_get_xi	Method to determine xi interval.

`kernel.fun` Kernel function to be used. Options include "normal", "epanechnikov", "Schennach2004", and "sinc".

`if_approx_kernel` Use approximations for the kernel function.

`kernel.resol` Resolution for kernel approximation.

Value

A configuration object (list) with all parameters

---

create_kernel_functions

Create kernel functions based on configuration

Description

Create kernel functions based on configuration

Usage

```
create_kernel_functions(
  kernel.fun = "Schennach2004",
  if_approx_kernel = TRUE,
  kernel.resol = 1000
)
```

Arguments

`kernel.fun` A string specifying the kernel function to be used.

`if_approx_kernel` Logical. If TRUE, uses approximations for the kernel function.

`kernel.resol` The resolution for kernel function approximation.

Value

A list containing kernel function, its Fourier transform, and the kernel type cv_bandwidth *Cross-Validation for Bandwidth Selection*

Description

Implements least-squares cross-validation for bandwidth selection with any kernel function. Uses the self-convolution approach for accurate estimation of the integral term.

Usage

```
cv_bandwidth(
  X,
  h_grid = NULL,
  kernel_func,
  kernel_type = "normal",
  grid_size = 512
)
```

Arguments

X	A numerical vector of sample data.
h_grid	A numerical vector of bandwidth values to evaluate. If NULL (default), a grid is automatically generated based on the range and distribution of the data.
kernel_func	The kernel function to use for cross-validation.
kernel_type	A string identifying the kernel type, used only for reference bandwidth.
grid_size	Number of grid points for evaluation. Default is 512.

Value

A scalar representing the optimal bandwidth that minimizes the cross-validation score.

Examples

```
# Generate sample data
X <- rnorm(100)
# Get optimal bandwidth using cross-validation with a normal kernel
kernel_functions <- create_kernel_functions("normal")
h_opt <- cv_bandwidth(X, kernel_func = kernel_functions$kernel,
                      kernel_type = kernel_functions$kernel_type)
```

DATA_PATH *The Path to the Data Folder*

Description

This variable provides the path to the `data` folder within the package.

Value

The path to the package's internal data folder as a character string.

epanechnikov_kernel *Epanechnikov Kernel*

Description

Epanechnikov Kernel

Usage

```
epanechnikov_kernel(u)
```

Arguments

u	A numerical value or vector representing the input to the kernel function.

Value

Returns the value of the Epanechnikov kernel function at the given input.

epanechnikov_kernel_ft *Fourier Transform Epanechnikov Kernel*

Description

Fourier Transform Epanechnikov Kernel

Usage

```
epanechnikov_kernel_ft(xi)
```

Arguments

xi	A numerical value or vector representing the frequency domain.

Value

Returns the value of the Fourier transform of the Epanechnikov kernel at the given frequency/frequencies.

`EXT_DATA_PATH` *The Path to the External Data Folder for Non-R Data Files*

Description

This variable provides the path to the `extdata` folder within the package, where non-standard R data files are stored.

Value

The path to the package's external data folder (for non-standard R data files) as a character string.

`fun_approx` *Approximation Function for Intensive Calculations*

Description

This function provides a lookup-based approximation for calculations that are computationally intensive. Once computed, it stores the results in an environment and uses linear interpolation for new data points to speed up subsequent computations.

Usage

```
fun_approx(u, u_lb = -100, u_ub = 100, resol = 10000, fun = W_kernel)
```

Arguments

`u`	A vector of values where the function should be evaluated.
`u_lb`	Lower bound for the precomputed range. Defaults to -10.
`u_ub`	Upper bound for the precomputed range. Defaults to 10.
`resol`	The resolution or number of sample points in the precomputed range. Defaults to 1000.
`fun`	A function for which the approximation is computed. Defaults to the `W` function.

Details

The fun_approx function works by initially creating a lookup table of function values based on the range specified by u_lb and u_ub and the resolution resol. This precomputation only happens once for a given set of parameters (u_lb, u_ub, resol, and fun). Subsequent calls to fun_approx with the same parameters use the lookup table to find the closest precomputed points to the requested u values and then return an interpolated result.

Linear interpolation is used between the two closest precomputed points in the lookup table. This ensures a smooth approximation for values in between sample points.

This function is especially useful for computationally intensive functions where recalculating function values is expensive or time-consuming. By using a combination of precomputation and interpolation, fun_approx provides a balance between accuracy and speed.

Value

A vector of approximated function values corresponding to u.

---

gen_sample_data *Generate Sample Data*

Description

This function used for generate some sample data for experiment

Usage

```
gen_sample_data(size, dgp, seed = NULL)
```

Arguments

size	control the sample size.
dgp	data generating process, have options "normal", "chisq", "mixed", "poly", "2_fold_uniform".
seed	random seed number.

Value

A numeric vector of length size. The elements of the vector are generated according to the specified dgp:

normal Normally distributed values with mean 0 and standard deviation 2.

chisq Chi-squared distributed values with df = 10.

mixed Half normally distributed (mean 0, sd = 2) and half chi-squared distributed (df = 10) values.

poly Values from a polynomial cumulative distribution function on [0,1].

2_fold_uniform Sum of two uniformly distributed random numbers.

get_avg_f1x *Kernel point estimation*

Description

Computes the point estimate using the specified kernel function.

Usage

```
get_avg_f1x(X, x, h, inf_k)
```

Arguments

X	A numerical vector of sample data.
x	A scalar representing the point where the density is estimated.
h	A scalar bandwidth parameter.
inf_k	Kernel function used for the computation.

Value

A scalar representing the kernel density estimate at point x.

get_avg_fyx *Kernel point estimation*

Description

Computes the point estimate using the specified kernel function.

Usage

```
get_avg_fyx(Y, X, x, h, inf_k)
```

Arguments

Y	A numerical vector representing the sample data of variable Y.
X	A numerical vector representing the sample data of variable X.
x	A scalar representing the point where the density is estimated.
h	A scalar bandwidth parameter.
inf_k	Kernel function used for the computation.

Value

A scalar representing the kernel density estimate at point x.

get_avg_phi — *Compute Sample Average of Fourier Transform Magnitude*

Description

Compute Sample Average of Fourier Transform Magnitude

Usage

```
get_avg_phi(Y = 1, X, xi)
```

Arguments

Y	A numerical vector representing the sample data of variable Y.
X	A numerical vector representing the sample data of variable X.
xi	A single numerical value representing the frequency at which the Fourier transform is computed.

Value

Returns the sample estimation of expected Fourier transform at frequency `xi`.

get_avg_phi_log — *Compute log sample average of fourier transform and get mod*

Description

Compute log sample average of fourier transform and get mod

Usage

```
get_avg_phi_log(Y = 1, X, ln_xi)
```

Arguments

Y	A numerical vector representing the sample data of variable Y.
X	A numerical vector representing the sample data of variable X.
ln_xi	A single numerical value representing the log frequency at which the Fourier transform is computed.

Value

Returns the log sample estimation of expected Fourier transform at frequency `xi`.

`get_conditional_var` *get the conditional variance of Y on X for given x*

Description

get the conditional variance of Y on X for given x

Usage

```
get_conditional_var(X, Y, x, h, kernel_func)
```

Arguments

`X`	A numerical vector representing the sample data of variable X.
`Y`	A numerical vector representing the sample data of variable Y.
`x`	The specific point at which the conditional variance is to be calculated.
`h`	A bandwidth parameter used in the kernel function for smoothing.
`kernel_func`	A kernel function used to weigh observations in the neighborhood of point x.

Value

Returns a non-negative scalar representing the estimated conditional variance of Y given X at the point x. Returns 0 if the computed variance is negative.

`get_est_Ar` *get the estimation of A and r*

Description

This function estimates the parameters A and r by optimizing an objective function over a specified range of frequency values and r values.

Usage

```
get_est_Ar(Y = 1, X, xi_interval, r_stepsize = 150)
```

Arguments

`Y`	A numerical vector representing the sample data of variable Y.
`X`	A numerical vector representing the sample data of variable X.
`xi_interval`	A list with elements `xi_lb` and `xi_ub` representing the lower and upper bounds of the frequency interval.
`r_stepsize`	An integer value representing the number of steps in the r range. This controls the granularity of the estimation. Higher values lead to finer granularity but increase computation time.

Details

The function internally defines a range for the natural logarithm of frequency values (`ln_xi_range`) and a range for the parameter `r` (`r_range`). It then defines an optimization function `optim_ln_A` to minimize the integral of a given function over the `ln_xi_range`. The actual estimation is done by finding the `r` and `A` value that minimizes the the area of the line $\ln A - r \ln \xi$ under the constraint that the line should not go below the Fourier transform curve.

Value

A named vector with elements `est_A` and `est_r` representing the estimated values of A and r, respectively.

`get_est_B` *get the estimation of B*

Description

get the estimation of B

Usage

```
get_est_B(Y)
```

Arguments

`Y`	A numerical vector representing the sample data of variable Y.

Value

The mean of the absolute values of the elements in Y, representing the estimated value of B.

`get_est_b1x` *Estimation of bias b1x*

Description

Computes the bias estimate for given parameters.

Usage

```
get_est_b1x(X, h, est_Ar, inf_k_ft, ...)
```

Arguments

`X`	A numerical vector representing the sample data of variable X.
`h`	A scalar bandwidth parameter.
`est_Ar`	A vector containing the estimated A and r parameters.
`inf_k_ft`	A kernel Fourier transform function.
`...`	Additional arguments passed to the quadgk integration function.

Value

A scalar representing the bias b1x estimate.

`get_est_byx` *Estimation of bias byx*

Description

Estimation of bias byx

Usage

```
get_est_byx(Y, X, ...)
```

Arguments

`Y`	A numerical vector representing the sample data of variable Y.
`X`	A numerical vector representing the sample data of variable X.
`...`	Additional arguments passed to other methods.

Value

A scalar representing the bias byx estimate.

`get_est_vy` *get the estimation of Vy*

Description

get the estimation of Vy

Usage

```
get_est_vy(Y)
```

Arguments

`Y`	A numerical vector representing the sample data of variable Y.

`get_sigma` *Estimation of sigma*

Description

Computes the sigma estimate for given parameters.

Usage

```
get_sigma(X, x, h, inf_k)
```

Arguments

`X`	A numerical vector of sample data.
`x`	A scalar representing the point where the density is estimated.
`h`	A scalar bandwidth parameter.
`inf_k`	Kernel function used for the computation.

Value

A scalar representing the sigma estimate at point x. Returns 0 if the density estimate is negative.

`get_sigma_yx` *Estimation of sigma_yx*

Description

Estimation of sigma_yx

Usage

```
get_sigma_yx(Y, X, x, h, inf_k)
```

Arguments

`Y`	A numerical vector representing the sample data of variable Y.
`X`	A numerical vector representing the sample data of variable X.
`x`	The specific point at which sigma_yx is to be estimated.
`h`	A bandwidth parameter used in the kernel function for smoothing.
`inf_k`	A kernel function used to weigh observations in the neighborhood of point x.

Value

Returns a scalar representing the estimated value of sigma_yx at the point x. Returns 0 if either fyx or conditional variance is negative.

get_xi_interval — *get xi interval*

Description

get xi interval

Usage

```
get_xi_interval(Y = 1, X, methods = "Schennach")
```

Arguments

Y	A numerical vector representing the sample data of variable Y.
X	A numerical vector representing the sample data of variable X.
methods	A character string indicating the method to use for calculating the xi interval. Supported methods are "Schennach" and "Schennach_loose". Defaults to "Schennach".

Details

The "Schennach" method computes the xi interval by performing a test based on the Schennach's theorem, adjusting the upper bound `xi_ub` if the test condition is met. The "Schennach_loose" method provides a looser calculation of the xi interval without performing the Schennach's test.

Value

A list containing the lower (`xi_lb`) and upper (`xi_ub`) bounds of the xi interval.

kernel_reg — *Kernel Regression function*

Description

Kernel Regression function

Usage

```
kernel_reg(X, Y, x, h, kernel_func)
```

Arguments

`X`	A numerical vector representing the sample data of variable X.
`Y`	A numerical vector representing the sample data of variable Y.
`x`	The point at which the regression function is to be estimated.
`h`	A bandwidth parameter that determines the weight assigned to each observation in X.
`kernel_func`	A function that computes the weight of each observation based on its distance to x.

Value

Returns a scalar representing the estimated value of the regression function at the point x.

`normal_kernel` *Normal Kernel Function*

Description

Normal Kernel Function

Usage

```
normal_kernel(u)
```

Arguments

`u`	A numerical value or vector representing the input to the kernel function.

Value

Returns the value of the Normal kernel function at the given input.

`normal_kernel_ft` *Fourier Transform of Normal Kernel*

Description

Fourier Transform of Normal Kernel

Usage

```
normal_kernel_ft(xi)
```

Arguments

`xi`	A numerical value or vector representing the frequency domain.

Value

Returns the value of the Fourier transform of the Normal kernel at the given frequency/frequencies.

`plot_ft` *Plot the Fourier Transform*

Description

Plot the Fourier Transform of the

Usage

```
plot_ft(X, xi_interval, ft_plot.resol = 500)
```

Arguments

`X`	A numerical vector of sample data.
`xi_interval`	A list containing the lower (`xi_lb`) and upper (`xi_ub`) bounds of the xi interval.
`ft_plot.resol`	An integer representing the resolution of the plot, specifically the number of points used to represent the Fourier transform. Defaults to 500.

Details

C = 1, the parameter in $O(1/n^{0.25})$, see more details in in Schennach (2020).

Value

A ggplot object representing the plot of the Fourier transform.

Examples

```
plot_ft(
  sample_data$X,
  xi_interval = list(xi_lb = 1, xi_ub = 50),
  ft_plot.resol = 1000
)
```

rpoly01 — *Generate n samples from the distribution*

Description

Generate n samples from the distribution

Usage

```
rpoly01(n, k = 5)
```

Arguments

n	The number of samples to generate.
k	The exponent in the distribution function, defaults to 5.

Value

A vector of n samples from the specified polynomial distribution.

CDF: f(x) = (x-1)^k + 1

sample_data — *Sample Data*

Description

Sample Data

Usage

```
sample_data
```

Format

A data frame with 1000 rows and 2 variables:

X Numeric vector, generated from 2 fold uniform distribution.

Y Numeric vector, `Y = -X^2 + 3*X + rnorm(1000)*X`.

select_bandwidth *Select Optimal Bandwidth*

Description

Selects an optimal bandwidth using the specified method.

Usage

```
select_bandwidth(
  X,
  Y = NULL,
  method = "cv",
  kernel.fun = "normal",
  if_approx_kernel = TRUE,
  kernel.resol = 1000
)
```

Arguments

X	A numerical vector of sample data.
Y	Optional. A numerical vector of sample data for conditional expectation estimation.
method	A string specifying the bandwidth selection method. Options are "cv" for cross-validation and "silverman" for Silverman's rule of thumb. Defaults to "cv".
kernel.fun	A string specifying the kernel type. Options include "normal", "epanechnikov", "Schennach2004", and "sinc".
if_approx_kernel	Logical. If TRUE, uses approximations for the kernel function.
kernel.resol	The resolution for kernel function approximation.

Value

A scalar representing the optimal bandwidth.

Examples

```
# Generate sample data
X <- rnorm(100)
# Get optimal bandwidth using cross-validation with normal kernel
h_opt <- select_bandwidth(X, method = "cv", kernel.fun = "normal")
# Get optimal bandwidth using Silverman's rule with Schennach kernel
h_opt <- select_bandwidth(X, method = "silverman", kernel.fun = "Schennach2004")
```

silverman_bandwidth *Silverman's Rule of Thumb for Bandwidth Selection*

Description

Implements Silverman's rule of thumb for selecting an optimal bandwidth in kernel density estimation.

Usage

```
silverman_bandwidth(X, kernel_type = "normal")
```

Arguments

X	A numerical vector of sample data.
kernel_type	A string identifying the kernel type.

Value

A scalar representing the optimal bandwidth.

Examples

```
# Generate sample data
X <- rnorm(100)
# Get optimal bandwidth using Silverman's rule
h_opt <- silverman_bandwidth(X, kernel_type = "normal")
```

sinc *Infinite Kernel Function*

Description

Infinite Kernel Function

Usage

```
sinc(u)
```

Arguments

u	A numerical value or vector where the sinc function is evaluated.

Value

The value of the sinc function at each point in u.

sinc_ft — *Define the closed form FT of the infinite order kernel sin(x)/(pi*x)*

Description

Define the closed form FT of the infinite order kernel sin(x)/(pi*x)

Usage

```
sinc_ft(x)
```

Arguments

x	A numerical value or vector where the Fourier Transform is evaluated.

Value

The value of the Fourier Transform of the sinc function at each point in x.

true_density_2fold — *True density of 2-fold uniform distribution*

Description

True density of 2-fold uniform distribution

Usage

```
true_density_2fold(x)
```

Arguments

x	A numerical value or vector where the true density function is evaluated.

Value

The value of the true density of the 2-fold uniform distribution at each point in x.

`W_kernel` *Define the inverse Fourier transform function of W*

Description

Define the inverse Fourier transform function of W

Usage

```
W_kernel(u, L = 10)
```

Arguments

u	A numerical value or vector representing the time or space domain.
L	The limit for numerical integration, defines the range of integration as $[-L, L]$. Defaults to 10.

Value

A numerical value or vector representing the inverse Fourier transform of the infinite order kernel at the given time or space point(s).

`W_kernel_ft` *Define the Fourier transform of a infinite kernel proposed in Schennach 2004*

Description

Define the Fourier transform of a infinite kernel proposed in Schennach 2004

Usage

```
W_kernel_ft(xi, xi_lb = 0.5, xi_ub = 1.5)
```

Arguments

xi	A numerical value or vector representing the frequency domain.
xi_lb	The lower bound for the frequency domain. Defaults to 0.5.
xi_ub	The upper bound for the frequency domain. Defaults to 1.5.

Value

A numerical value or vector representing the Fourier transform of the infinite order kernel at the given frequency/frequencies.

Index