Package 'fastlpr'

May 8, 2026

Title Fast Local Polynomial Regression and Kernel Density Estimation

Version 1.0.1

Description Non-Uniform Fast Fourier Transform ('NUFFT')-accelerated local polynomial regression and kernel density estimation for large, scattered, or complex-valued datasets. Provides automatic bandwidth selection via Generalized Cross-Validation (GCV) for regression and Likelihood Cross-Validation (LCV) for density estimation. This is the 'R' port of the 'fastLPR' 'MATLAB'/'Python' toolbox, achieving O(N + M log M) computational complexity through custom 'NUFFT' implementation with Gaussian gridding. Supports 1D/2D/3D data, complex-valued responses, heteroscedastic variance estimation, and confidence interval computation. Performance optimized with vectorized 'R' code and compiled helpers via 'Rcpp'/'RcppArmadillo'. Extends the 'FKreg' toolbox of Wang et al. (2022) <doi:10.48550/arXiv.2204.07716> with 'Python' and 'R' ports. Applied in Li et al. (2022) <doi:10.1016/j.neuroimage.2022.119190>. Uses 'NUFFT' methods based on Greengard and Lee (2004) <doi:10.1137/S003614450343200X>, binning-accelerated kernel estimation of Wand (1994) <doi:10.1080/10618600.1994.10474656>, and local polynomial regression framework of Fan and Gijbels (1996, ISBN:978-0412983214).

License GPL-3

Encoding UTF-8

ByteCompile false

Depends R (>= 4.2.0)

Imports stats, utils, grDevices, graphics, compiler, Rcpp (>= 1.0.0)

Suggests testthat (>= 3.0.0), akima, rgl, R.matlab

LinkingTo Rcpp, RcppArmadillo

SystemRequirements GNU make

URL https://github.com/rigelfalcon/fastLPR

BugReports https://github.com/rigelfalcon/fastLPR/issues

RoxygenNote 7.3.1

NeedsCompilation yes

Author Ying Wang [aut, cre],
Min Li [aut]

Maintainer Ying Wang <yingwangrigel@gmail.com>

Repository CRAN

Date/Publication 2026-04-21 08:20:02 UTC

Contents

`cv_fastkde` *Fast Kernel Density Estimation with automatic bandwidth selection*

Description

Performs kernel density estimation using NUFFT acceleration with automatic bandwidth selection via Likelihood Cross-Validation (LCV).

Usage

```
cv_fastkde(x, h = NULL, opt = NULL)
```

Arguments

`x`	N x d matrix of data points (N samples, d dimensions).
`h`	Bandwidth parameter(s) or grid for LCV selection.
`opt`	Options list with fields similar to cv_fastlpr.

Value

KDE results list containing estimated density, evaluation grid, and LCV results.

See Also

cv_fastlpr, get_hlist

Examples

```
x <- matrix(rnorm(200), ncol = 1)
kde <- cv_fastkde(x)
```

---

cv_fastlpr *Fast Local Polynomial Regression with automatic bandwidth selection*

Description

This is the main function for fastLPR toolbox. It performs nonparametric regression using kernel-weighted local polynomial methods with NUFFT (Non-Uniform Fast Fourier Transform) acceleration. The function automatically selects the optimal bandwidth via Generalized Cross-Validation (GCV) when multiple bandwidth candidates are provided.

Usage

```
cv_fastlpr(x, y, h = NULL, opt = NULL)
```

Arguments

x	N x d matrix of predictors (N samples, d dimensions). Can be real or complex-valued. Each row is one observation. For complex-valued predictors, use x = real + 1i*imag.
y	N x 1 vector of responses. Can be real or complex-valued. Must have same number of rows as x.
h	Bandwidth parameter(s) (optional, default: automatic selection). Scalar: same bandwidth for all dimensions. 1 x d vector: different bandwidth per dimension. k x d matrix: grid of k bandwidth combinations for GCV selection. Use get_hlist() to generate bandwidth candidates.
opt	Options list (optional) with fields:

order Polynomial order (default: 0) 0 = Nadaraya-Watson (local constant) 1 = Local linear regression 2 = Local quadratic regression

kernel_type Kernel function (default: 'gaussian') 'gaussian' or 'epanechnikov'

dstd Number of DOF samples for variance estimation (default: 0) Set to 5-20 for heteroscedastic variance estimation

y_type_out Output type (default: 'mean') 'mean' = estimate conditional mean E[Y|X] 'variance' = estimate conditional variance Var[Y|X]

N Grid resolution for evaluation (default: auto)

xrange Evaluation range (default: data range)

seed Random seed for Monte Carlo DOF estimation (default: NULL, no seed set). Set to an integer for reproducible bandwidth selection.

verbose Logical. Print bandwidth selection details (default: FALSE).

Value

Regression results list containing:

yhat Fitted values at evaluation grid points (N_grid x 1)

fpp_yhat Interpolant object for prediction at any point. Use: y_pred = regs$fpp_yhat(x_new)

gcv_yhat GCV results list (if multiple bandwidths provided)

- **h1se** Selected bandwidth (1-SE rule)
- **hmin** Bandwidth with minimum GCV
- **gcv** GCV values for all bandwidths

xq Evaluation grid points (list for multi-dimensional)

xlist Grid vectors (list)

opt Options used for regression

xraw Original predictor data

yraw Original response data

Author(s)

Ying Wang, Min Li

References

Wang, Y., & Li, M. (2024). Fast and Exact Kernel-Weighted Regression for Large-Scale Scattered and Complex-Valued Data. Journal of Statistical Software (under review).

See Also

cv_fastkde, get_hlist, fastlpr_interval, fastlpr_plot

Examples

```
# Example 1: 1D regression with automatic bandwidth selection
x <- matrix(runif(500) * 20, ncol = 1)
y <- sin(x) + 0.2 * rnorm(500)
hlist <- get_hlist(20, c(0.01, 1), "logspace")
opt <- list(order = 1)  # Local linear
regs <- cv_fastlpr(x, y, hlist, opt)

# Plot results
plot(x, y, pch = ".", col = "black")
# Note: fpp_yhat is an interpolation function
x_pred <- seq(min(x), max(x), length.out = 100)
```

```
y_pred <- regs$fpp_yhat(x_pred)
lines(x_pred, y_pred, col = "blue", lwd = 2)
```

`fastkde_plot` *Plot KDE results*

Description

Visualizes kernel density estimation results.

Usage

```
fastkde_plot(fpp, x_range = NULL, n_points = 1000, ...)
```

Arguments

`fpp`	KDE interpolator from cv_fastkde.
`x_range`	Optional range for plotting.
`n_points`	Number of points for plotting (default: 1000).
`...`	Additional plotting parameters.

Value

Called for its side effect (plotting). Returns `NULL` invisibly.

See Also

`cv_fastkde`

Examples

```
x <- matrix(rnorm(200), ncol = 1)
kde <- cv_fastkde(x)
fastkde_plot(kde$fpp)
```

fastkde_plot_bandwidth *Plot Bandwidth Selection Diagnostics for KDE*

Description

Visualizes the bandwidth selection process for kernel density estimation, showing the LCV score as a function of bandwidth with the selected optimum marked.

Usage

```
fastkde_plot_bandwidth(kde, main = "Bandwidth Selection (LCV)",
                       xlab = NULL, ylab = NULL, ...)
```

Arguments

kde	A fastkde_result object returned by cv_fastkde.
main	Title for the plot.
xlab	Label for the x-axis (default: auto-generated).
ylab	Label for the y-axis (default: auto-generated).
...	Additional arguments passed to plot.

Value

Called for its side effect (plotting). Returns NULL invisibly.

Examples

```
x <- matrix(rnorm(500), ncol = 1)
hlist <- get_hlist(20, c(0.05, 2))
kde <- cv_fastkde(x, hlist)
fastkde_plot_bandwidth(kde)
```

fastlpr_interval *Compute Confidence or Prediction Intervals*

Description

Computes pointwise confidence intervals for the conditional mean or prediction intervals for new observations, using the mean and variance estimates from local polynomial regression.

Usage

```
fastlpr_interval(mu, sigma, alpha = 0.05, type = "confidence")
```

Arguments

mu	Mean regression result from cv_fastlpr (a list with field fpp_yhat).
sigma	Variance regression result from cv_fastlpr (a list with field fpp_yhat), or a numeric vector/matrix of standard deviations.
alpha	Significance level (default: 0.05 for 95% intervals).
type	Type of interval: "confidence" for the conditional mean or "prediction" for new observations (default: "confidence").

Value

List with lower and upper interval bounds.

See Also

cv_fastlpr, fastlpr_plot_interval

Examples

```
x <- matrix(runif(200), ncol = 1)
y <- sin(2 * pi * x) + rnorm(200, sd = 0.2)
mu <- cv_fastlpr(x, y)
sigma <- cv_fastlpr(x, (y - mu$yhat)^2)
ci <- fastlpr_interval(mu, sigma)
```

---

fastlpr_plot *Plot regression results*

Description

Visualizes fitted regression results.

Usage

```
fastlpr_plot(fpp_yhat, x_range = NULL, n_points = 1000, ...)
```

Arguments

fpp_yhat	Interpolant function or result object.
x_range	Optional range for plotting.
n_points	Number of points for plotting (default: 1000).
...	Additional plotting parameters.

Value

Called for its side effect (plotting). Returns NULL invisibly.

See Also

cv_fastlpr

Examples

```
x <- matrix(runif(200), ncol = 1)
y <- sin(2 * pi * x) + rnorm(200, sd = 0.2)
fit <- cv_fastlpr(x, y)
fastlpr_plot(fit$fpp_yhat)
```

---

fastlpr_plot_interval *Plot Confidence or Prediction Interval Bands*

Description

Visualizes confidence or prediction interval bands for local polynomial regression estimates. Adds shaded interval regions to an existing plot.

Usage

```
fastlpr_plot_interval(ci, col = "green", alpha = 0.2, add = TRUE, ...)
```

Arguments

ci	An interval structure returned by fastlpr_interval.
col	Color for the interval band (default: "green").
alpha	Transparency level for the band (default: 0.2).
add	Logical; if TRUE (default), add to existing plot.
...	Additional arguments passed to plotting functions.

Value

Called for its side effect (plotting). Returns NULL invisibly.

See Also

fastlpr_interval, cv_fastlpr

Examples

```
set.seed(42)
x <- sort(runif(200))
y <- sin(2 * pi * x) + rnorm(200, sd = 0.3)
hlist <- get_hlist(20, c(0.01, 0.5))
regs <- cv_fastlpr(x, y, hlist)
```

fastlpr_predict *Predict at new points*

Description

Evaluates fitted regression at new predictor values.

Usage

```
fastlpr_predict(regs, x_new)
```

Arguments

regs	Regression results from cv_fastlpr.
x_new	New predictor values (matrix).

Value

Predicted response values.

See Also

cv_fastlpr

Examples

```
x <- matrix(runif(200), ncol = 1)
y <- sin(2 * pi * x) + rnorm(200, sd = 0.2)
fit <- cv_fastlpr(x, y)
x_new <- matrix(seq(0, 1, length.out = 50), ncol = 1)
yhat <- fastlpr_predict(fit, x_new)
```

get_hlist *Generate bandwidth candidates for cross-validation*

Description

Creates a grid of bandwidth candidates for cross-validation. Port from MATLAB's get_hlist.m (unified API v2.0).

Usage

```
get_hlist(n, range, spacing = "logspace")
```

Arguments

n	Number of bandwidth candidates per dimension. Scalar: same number of points for all dimensions. Vector: specify number of points per dimension. Typical values: 20 for 1D, c(15, 15) for 2D.
range	Range of bandwidths (min, max) for each dimension. 1D: c(h_min, h_max) or matrix with 1 row. Multi-D: matrix with one row per dimension. Rule of thumb: h_min = 0.1 * sd(x), h_max = 1.0 * sd(x).
spacing	Spacing function type (default: "logspace"). "logspace": logarithmic spacing (better for exploring bandwidth scales). "linear": linear spacing.

Value

Matrix of bandwidth candidates (n_total x d). Each row is one bandwidth combination.

See Also

cv_fastlpr, cv_fastkde

Examples

```
# 1D: 20 bandwidths from 0.01 to 1 (log scale, default)
hlist <- get_hlist(20, c(0.01, 1))

# 2D: 10x10 grid of bandwidths
hlist <- get_hlist(10, rbind(c(0.01, 1), c(0.01, 1)))

# Linear spacing
hlist <- get_hlist(10, c(0.5, 0.6), "linear")
```

---

get_rcpp_info *Get Rcpp Information*

Description

Returns information about the Rcpp backend including version and available acceleration features.

Usage

```
get_rcpp_info()
```

Value

A list with Rcpp version and capabilities if available, NULL otherwise.

Examples

```
get_rcpp_info()
```

is_fastkde_result *Check if Object is a fastkde_result*

Description

Tests whether an object is of class `fastkde_result`.

Usage

```
is_fastkde_result(x)
```

Arguments

`x`	An R object to test.

Value

`TRUE` if `x` inherits from class `"fastkde_result"`, `FALSE` otherwise.

Examples

```
is_fastkde_result(list())  # FALSE
```

is_fastlpr_result *Check if Object is a fastlpr_result*

Description

Tests whether an object is of class `fastlpr_result`.

Usage

```
is_fastlpr_result(x)
```

Arguments

`x`	An R object to test.

Value

`TRUE` if `x` inherits from class `"fastlpr_result"`, `FALSE` otherwise.

Examples

```
is_fastlpr_result(list())  # FALSE
```

multispace *Generate multi-dimensional grid*

Description

Utility function to generate multi-dimensional grids.

Usage

```
multispace(x_min, x_max, N, space = "linear", type = "array", transpose = FALSE)
```

Arguments

x_min	Minimum values per dimension.
x_max	Maximum values per dimension.
N	Grid size per dimension.
space	Grid spacing: "linear" (default) or "logspace".
type	Output type: "array" (default) or "cell".
transpose	Whether to transpose output (default: FALSE).

Value

List of grid vectors.

Examples

```
grid <- multispace(0, 1, 50)
```

rcpp_available *Check if Rcpp Acceleration is Available*

Description

Tests whether the compiled Rcpp code is loaded and functional. This is useful for checking if C++ acceleration will be used.

Usage

```
rcpp_available()
```

Value

Logical. TRUE if Rcpp functions are available, FALSE otherwise.

Examples

```
rcpp_available()
```

set_defaults *Set default options*

Description

Utility function to set a single default option field.

Usage

```
set_defaults(opt, field, default)
```

Arguments

opt	User-provided options list.
field	Field name to set.
default	Default value for the field.

Value

Merged options list.

Examples

```
opt <- list(order = 1)
opt <- set_defaults(opt, "kernel", "gaussian")
```

zscore *Z-score normalization*

Description

Standardizes data to zero mean and unit variance.

Usage

```
zscore(x)
```

Arguments

x	Numeric vector or matrix.

Value

Standardized data with attributes for mean and standard deviation.

Examples

```
result <- zscore(rnorm(100, mean = 5, sd = 2))
mean(result$z)  # approximately 0
```

Index