Package 'MonotonicityTest'

April 24, 2025

Type Package

Author Dylan Huynh [aut, cre]

Maintainer Dylan Huynh <dylanhuynh@utexas.edu>

Title Nonparametric Bootstrap Test for Regression Monotonicity

Version 1.2

Description Implements nonparametric bootstrap tests for detecting monotonicity
in regression functions from Hall, P. and Heckman, N. (2000) <doi:10.1214/aos/1016120363>
Includes tools for visualizing results using Nadaraya-Watson kernel regression and supports
efficient computation with 'C++'.

License GPL

Encoding UTF-8

RoxygenNote 7.3.2

LinkingTo Rcpp, RcppEigen

Imports Rcpp (>= 1.0.13-1), parallel, stats, graphics, ggplot2 (>=
3.0.0), rlang

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2025-04-24 19:20:02 UTC

Contents

---

create_kernel_plot   *Generate Kernel Plot*

Description

Creates a scatter plot of the input vectors X and Y, and overlays a Nadaraya-Watson kernel regression curve using the specified bandwidth.

Usage

```
create_kernel_plot(X, Y, bandwidth = bw.nrd(X) * (length(X)^-0.1), nrows = 4)
```

Arguments

X	Vector of x values.
Y	Vector of y values.
bandwidth	Kernel bandwidth used for the Nadaraya-Watson estimator. Can be a single numeric value or a vector of bandwidths. Default is calculated as bw.nrd(X) * (length(X) ^ -0.1).
nrows	Number of rows in the facet grid if multiple bandwidths are provided. Does not do anything if only a single bandwidth value is provided. Default is 4.

Value

A ggplot object containing the scatter plot(s) with the kernel regression curve(s). If a vector of bandwidths is supplied, the plots are put into a grid using faceting.

References

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability and Its Applications*, **9**(1), 141–142.

Watson, G. S. (1964). Smooth estimates of regression functions. *Sankhyā: The Indian Journal of Statistics, Series A*, 359-372.

Examples

```
# Example 1: Basic plot on quadratic function
seed <- 42
set.seed(seed)
X <- runif(500)
Y <- X ^ 2 + rnorm(500, sd = 0.1)
plot <- create_kernel_plot(X, Y, bandwidth = bw.nrd(X) * (length(X) ^ -0.1))
```

---

diabetes *A Simulated Diabetes Dataset*

Description

This dataset contains simulated medical measurements for Diabetes and is emulated after data from the Diabetes Prevention Program. Each column represents change in a key metabolic indicators after two years for the placebo group receiving no treatment.

Usage

```
data("diabetes", package="MonotonicityTest")
```

Format

A data frame with 1000 rows and 4 variables:

CLDL Change in low-density lipoprotein (LDL) cholesterol (mg/dL).

GLUCOSE Change in fasting plasma glucose levels (mg/dL).

TRIG Change in triglyceride levels (mg/dL).

HBA1C Change in hemoglobin A1c levels (%).

Examples

```
data("diabetes", package="MonotonicityTest")
names(diabetes)
```

---

monotonicity_test *Perform Monotonicity Test*

Description

Performs a monotonicity test between the vectors X and Y as described in Hall and Heckman (2000). This function uses a bootstrap approach to test for monotonicity in a nonparametric regression setting.

Usage

```
monotonicity_test(
  X,
  Y,
  bandwidth = bw.nrd(X) * (length(X)^-0.1),
  boot_num = 200,
  m = floor(0.05 * length(X)),
  ncores = 1,
  negative = FALSE,
  seed = NULL
)
```

Arguments

X	Numeric vector of predictor variable values. Must not contain missing or infinite values.
Y	Numeric vector of response variable values. Must not contain missing or infinite values.
bandwidth	Numeric value for the kernel bandwidth used in the Nadaraya-Watson estimator. Default is calculated as bw.nrd(X) * (length(X) ^ -0.1).
boot_num	Integer specifying the number of bootstrap samples. Default is 200.
m	Integer parameter used in the calculation of the test statistic. Corresponds to the minimum window size to calculate the test statistic over or a "smoothing" parameter. Lower values increase the sensitivity of the test to local deviations from monotonicity. Default is floor(0.05 * length(X)).
ncores	Integer specifying the number of cores to use for parallel processing. Default is 1.
negative	Logical value indicating whether to test for a monotonic decreasing (negative) relationship. Default is FALSE.
seed	Optional integer for setting the random seed. If NULL (default), the global random state is used.

Details

The test evaluates the following hypotheses:

H_0: The regression function is monotonic

- *Non-decreasing* if negative = FALSE
- *Non-increasing* if negative = TRUE

H_A: The regression function is not monotonic

Value

A list with the following components:

p The p-value of the test. A small p-value (e.g., < 0.05) suggests evidence against the null hypothesis of monotonicity.

dist The distribution of test statistic under the null from bootstrap samples. The length of dist is equal to boot_num.

stat The test statistic T_m calculated from the original data.

plot A ggplot object with a scatter plot where the points of the "critical interval" are highlighted. This critical interval is the interval where T_m is greatest.

interval Numeric vector containing the indices of the "critical interval". The first index indicates where the interval starts, and the second indicates where it ends in the sorted X vector.

Note

For large datasets (e.g., $n \geq 6500$) this function may require significant computation time due to having to compute the statistic for every possible interval. Consider reducing boot_num, using a subset of the data, or using parallel processing with ncores to improve performance.

In addition to this, a minimum of 300 observations is recommended for kernel estimates to be reliable.

References

Hall, P., & Heckman, N. E. (2000). Testing for monotonicity of a regression mean by calibrating for linear functions. *The Annals of Statistics*, **28**(1), 20–39.

Examples

```
# Example 1: Usage on monotonic increasing function
# Generate sample data
seed <- 42
set.seed(seed)

X <- runif(500)
Y <- 4 * X + rnorm(500, sd = 1)
result <- monotonicity_test(X, Y, boot_num = 25, seed = seed)

print(result)

# Example 2: Usage on non-monotonic function
seed <- 42
set.seed(seed)

X <- runif(500)
Y <- (X - 0.5) ^ 2 + rnorm(500, sd = 0.5)
result <- monotonicity_test(X, Y, boot_num = 25, seed = seed)

print(result)
```

Index