

# Package ‘npbr’

May 9, 2026

**Type** Package

**Title** Nonparametric Boundary Regression

**Version** 1.8

**Date** 2023-03-21

**Author** Abdelaati Daouia <Abdelaati.Daouia@tse-fr.eu>, Thibault Laurent <thibault.laurent@univ-tlse1.fr>, Hohsuk Noh <word5810@gmail.com>

**Maintainer** Thibault Laurent <thibault.laurent@univ-tlse1.fr>

**Depends** R (>= 4.0.0), graphics, stats, utils

**Imports** Benchmarking, np, quadprog, Rglpk (>= 0.6-2), splines

**Description** A variety of functions for the best known and most innovative approaches to nonparametric boundary estimation. The selected methods are concerned with empirical, smoothed, unrestricted as well as constrained fits under both separate and multiple shape constraints. They cover robust approaches to outliers as well as data envelopment techniques based on piecewise polynomials, splines, local linear fitting, extreme values and kernel smoothing. The package also seamlessly allows for Monte Carlo comparisons among these different estimation methods. Its use is illustrated via a number of empirical applications and simulated examples.

**License** GPL (>= 2)

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2023-03-22 09:00:05 UTC

## Contents

npbr-package . . . . .	2
air . . . . .	5
cub_spline_est . . . . .	6
cub_spline_kn . . . . .	8
dea_est . . . . .	10
dfs_momt . . . . .	12
dfs_pick . . . . .	14
dfs_pwm . . . . .	17

green . . . . .	19
kern_smooth . . . . .	20
kern_smooth_bw . . . . .	22
kopt_momt_pick . . . . .	24
loc_est . . . . .	26
loc_est_bw . . . . .	28
loc_max . . . . .	30
mopt_pwm . . . . .	31
nuclear . . . . .	33
pick_est . . . . .	34
poly_degree . . . . .	35
poly_est . . . . .	37
post . . . . .	38
quad_spline_est . . . . .	39
quad_spline_kn . . . . .	41
records . . . . .	43
rho_momt_pick . . . . .	44
rho_pwm . . . . .	46
<b>Index</b>	<b>48</b>

---

 npbr-package

*Nonparametric boundary regression*


---

## Description

The package npbr (Daouia et al., 2017) is the first free specialized software for data edge and frontier analysis in the statistical literature. It provides a variety of functions for the best known and most innovative approaches to nonparametric boundary estimation. The selected methods are concerned with empirical, smoothed, unrestricted as well as constrained fits under both separate and multiple shape constraints. They also cover data envelopment techniques as well as robust approaches to outliers. The routines included in **npbr** are user friendly and afford a large degree of flexibility in the estimation specifications. They provide smoothing parameter selection for the modern local linear and polynomial spline methods and for some promising extreme value techniques. Also, they seamlessly allow for Monte Carlo comparisons among the implemented estimation procedures. This package will be very useful for statisticians and applied researchers interested in employing nonparametric boundary regression models. Its use is illustrated with a number of empirical applications and simulated examples.

## Details

Suppose that we have  $n$  pairs of observations  $(x_i, y_i)$ ,  $i = 1, \dots, n$ , from a bivariate distribution with a density  $f(x, y)$  in  $R^2$ . The support  $\Psi$  of  $f$  is assumed to be of the form

$$\Psi = \{(x, y) | y \leq \varphi(x)\} \supseteq \{(x, y) | f(x, y) > 0\}$$

$$\{(x, y) | y > \varphi(x)\} \subseteq \{(x, y) | f(x, y) = 0\},$$

where the graph of  $\varphi$  corresponds to the locus of the curve above which the density  $f$  is zero. We consider the estimation of the frontier function  $\varphi$  based on the sample  $\{(x_i, y_i), i = 1, \dots, n\}$  in the general setting where the density  $f$  may have sudden jumps at the frontier, decay to zero or rise up to infinity as it approaches its support boundary.

The overall objective of the present package is to provide a large variety of functions for the best known approaches to nonparametric boundary regression, including the vast class of methods employed in both Monte Carlo comparisons of Daouia et al. (2016) and Noh (2014) as well as other promising nonparametric devices, namely the extreme-value techniques of Gijbels and Peng (2000), Daouia et al. (2010) and Daouia et al. (2012). The various functions in the **npbr** package are summarized in the table below. We are not aware of any other existing set of statistical routines more adapted to data envelope fitting and robust frontier estimation. Only the classical nonsmooth FDH and DEA methods can be found in some available packages dedicated to the economic literature on measurements of the production performance of enterprises, such as the programs **Benchmarking** by Bogetoft and Otto (2011) and **FEAR** by Wilson (2008). Other contributions to the econometric literature on frontier analysis by Parmeter and Racine (2013) can be found at <https://socialsciences.mcmaster.ca/racinej/Gallery/Home.html>. The package **npbr** is actually the first free specialized software for the statistical literature on nonparametric frontier analysis. The routines included in **npbr** are user friendly and highly flexible in terms of estimation specifications. They allow the user to locate the boundary from data by making use of both empirical and smooth fits as well as (un)constrained estimates under single and multiple shape constraints. They also integrate smoothing parameter selection for the innovative methods based on local linear techniques, polynomial splines, extreme values and kernel smoothing, though the proposed selection procedures can be computationally demanding.

In addition, the package will be extremely useful for researchers and practitioners interested in employing nonparametric boundary regression methods. On one hand, such methods are very appealing because they rely on very few assumptions and benefit from their modeling flexibility, function approximation power and ability to detect the boundary structure of data without recourse to any *a priori* parametric restrictions on the shape of the frontier and/or the distribution of noise. On the other hand, the package offers **R** users and statisticians in this active area of research simple functions to compute the empirical mean integrated squared error, the empirical integrated squared bias and the empirical integrated variance of the implemented frontier estimators. This seamlessly allows the interested researcher to reproduce the Monte Carlo estimates obtained in the original articles and, perhaps most importantly, to easily compare the quality of any new proposal with the competitive existing methods.

Function	Description	Reference
<code>dea_est</code>	DEA, FDH and linearized FDH	Farrell (1957) Deprins et al. (1984), Hall and Park (2002) Jeong and Simar (2006)
<code>loc_est</code>	Local linear fitting	Hall et al. (1998), Hall and Park (2004)
<code>loc_est_bw</code>	Bandwidth choice for local linear fitting	Hall and Park (2004)
<code>poly_est</code>	Polynomial estimation	Hall et al. (1998)
<code>poly_degree</code>	Optimal polynomial degree selection	Daouia et al. (2015)
<code>dfs_momt</code>	Moment type estimation	Daouia et al. (2010),

<code>dfs_pick</code>	Pickands type estimation	Dekkers et al. (1989) Daouia et al. (2010), Dekkers and de Haan (1989)
<code>rho_momt_pick</code>	Conditional tail index estimation	Daouia et al. (2010), Dekkers et al. (1989), Dekkers and de Haan (1989)
<code>kopt_momt_pick</code>	Threshold selection for moment/Pickands frontiers	Daouia et al. (2010)
<code>dfs_pwm</code>	Nonparametric frontier regularization	Daouia et al. (2012)
<code>loc_max</code>	Local constant estimation	Gijbels and Peng (2000)
<code>pick_est</code>	Local extreme-value estimation	Gijbels and Peng (2000)
<code>quad_spline_est</code>	Quadratic spline fitting	Daouia et al. (2015)
<code>quad_spline_kn</code>	Knot selection for quadratic spline fitting	Daouia et al. (2015)
<code>cub_spline_est</code>	Cubic spline fitting	Daouia et al. (2015)
<code>cub_spline_kn</code>	Knot selection for cubic spline fitting	Daouia et al. (2015)
<code>kern_smooth</code>	Nonparametric kernel boundary regression	Parmeter and Racine (2013), Noh (2014)
<code>kern_smooth_bw</code>	Bandwidth choice for kernel boundary regression	Parmeter and Racine (2013), Noh (2014)

### Author(s)

Abdelaati Daouia <Abdelaati.Daouia@tse-fr.eu>, Thibault Laurent <thibault.laurent@univ-tlse1.fr>, Hohsuk Noh <word5810@gmail.com>

Maintainer: Thibault Laurent <thibault.laurent@univ-tlse1.fr>

### References

- Daouia, A., Florens, J.-P. and Simar, L. (2010). Frontier estimation and extreme value theory. *Bernoulli*, **16**, 1039-1063.
- Daouia, A., Florens, J.-P. and Simar, L. (2012). Regularization of Nonparametric Frontier Estimators. *Journal of Econometrics*, **168**, 285-299.
- Daouia, A., Laurent, T. and Noh, H. (2017). npbr: A Package for Nonparametric Boundary Regression in R. *Journal of Statistical Software*, **79**(9), 1-43. doi:10.18637/jss.v079.i09.
- Daouia, A., Noh, H. and Park, B.U. (2016). Data Envelope fitting with constrained polynomial splines. *Journal of the Royal Statistical Society: Series B*, **78**(1), 3-30. doi:10.1111/rssb.12098.
- Dekkers, A.L.M. and L. de Haan (1989). On the estimation of extreme-value index and large quantiles estimation. *Annals of Statistics*, **17**, 1795-1832.
- Dekkers, A.L.M., Einmahl, J.H.J. and L. de Haan (1989). A moment estimator for the index of an extreme-value distribution. *Annals of Statistics*, **17**, 1833-1855.
- Deprins, D., Simar, L. and Tulkens H. (1984). Measuring labor efficiency in post offices, in: M. Marchand, P. Pestieau and H. Tulkens (Eds), *The performance of Public Enterprises: Concepts and Measurements*. North-Holland, Amsterdam, 243-267.

- Farrell, M.J. (1957). The measurement of productive efficiency. *Journal of the Royal Statistical Society, Series A*, **120**, 253-281.
- Gijbels, I. and Peng, L. (2000). Estimation of a support curve via order statistics. *Extremes*, **3**, 251-277.
- Hall, P., Park, B.U. and Stern, S.E. (1998). On polynomial estimators of frontiers and boundaries. *Journal of Multivariate Analysis*, **66**, 71-98.
- Hall, P. and Park, B.U. (2004). Bandwidth choice for local polynomial estimation of smooth boundaries. *Journal of Multivariate Analysis*, **91**, 240-261.
- Jeong, S.-O. and Simar, L. (2006). Linearly interpolated FDH efficiency score for nonconvex frontiers. *Journal of Multivariate Analysis*, **97**, 2141-2161.
- Noh, H. (2014). Frontier Estimation using Kernel Smoothing with Data Transformation. *Journal of the Korean Statistical Society*, **43**, 503-512.
- Parmeter, C. and Racine, J.S. (2013). Smooth Constrained Frontier Analysis. In *Recent Advances and Future Directions in Causality, Prediction, and Specification Analysis: Essays in Honor of Halbert L. White, Jr.*, Springer Verlag, (X. Chen and N.R. Swanson Eds), 463-488.

---

air

*European air controllers*

---

## Description

The dataset is concerned with the assessment of the efficiency of 37 European Air Controllers. The performance of each controller can be measured by its “distance” from the upper support boundary, or equivalently, the set of the most efficient controllers. This dataset is taken from Mouchart and Simar (2002). Here, the activity of the controllers is described by one input (an aggregate factor of different kind of labor) and one output (an aggregate factor of the activity produced, based on the number of controlled air movements, the number of controlled flight hours, etc.). See also Daouia, Florens and Simar (2008).

## Usage

```
data(air)
```

## Format

A data frame with 37 observations on the following 2 variables.

xtab an input.

ytab an output.

## References

- Daouia, A., Florens, J.-P. and Simar, L. (2008). Functional Convergence of Quantile-type Frontiers with Application to Parametric Approximations. *Journal of Statistical Planning and Inference*, **138**, 708-725.
- Mouchart, M. and L. Simar (2002). Efficiency analysis of Air Controllers: first insights, Consulting report 0202, Institut de Statistique, UCL, Belgium.

**Examples**

```
data("air")
```

---

cub_spline_est	<i>Cubic spline fitting</i>
----------------	-----------------------------

---

**Description**

The function `cub_spline_est` is an implementation of the (un)constrained cubic spline estimates proposed by Daouia, Noh and Park (2016).

**Usage**

```
cub_spline_est(xtab, ytab, x, kn = ceiling((length(xtab))^(1/4)), method= "u",
               all.dea=FALSE, control = list("tm_limit" = 700))
```

**Arguments**

<code>xtab</code>	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
<code>ytab</code>	a numeric vector of the same length as <code>xtab</code> containing the observed outputs $y_1, \dots, y_n$ .
<code>x</code>	a numeric vector of evaluation points in which the estimator is to be computed.
<code>kn</code>	an integer specifying the number of inter-knot segments used in the computation of the spline estimate.
<code>method</code>	a character equal to "u" (unconstrained estimator), "m" (under the monotonicity constraint) or "mc" (under simultaneous monotonicity and concavity constraints).
<code>all.dea</code>	a boolean.
<code>control</code>	a list of parameters to the GLPK solver. See *Details* of <code>help(Rglpk_solve_LP)</code> .

**Details**

Let  $a$  and  $b$  be, respectively, the minimum and maximum of the design points  $x_1, \dots, x_n$ . Denote a partition of  $[a, b]$  by  $a = t_0 < t_1 < \dots < t_{k_n} = b$  (see below the selection process). Let  $N = k_n + 3$  and  $\pi(x) = (\pi_0(x), \dots, \pi_{N-1}(x))^T$  be the vector of normalized B-splines of order 4 based on the knot mesh  $\{t_j\}$  (see Daouia et al. (2015)). The unconstrained (option `method="u"`) cubic spline estimate of the frontier  $\varphi(x)$  is then defined by  $\tilde{\varphi}_n(x) = \pi(x)^T \tilde{\alpha}$ , where  $\tilde{\alpha}$  minimizes

$$\int_0^1 \pi(x)^T \alpha \, dx = \sum_{j=1}^N \alpha_j \int_0^1 \pi_j(x) \, dx$$

over  $\alpha \in \mathbf{R}^N$  subject to the envelopment constraints  $\pi(x_i)^T \alpha \geq y_i$ ,  $i = 1, \dots, n$ . A simple way of choosing the knot mesh in this unconstrained setting is by considering the  $j/k_n$ th quantiles  $t_j = x_{[jn/k_n]}$  of the distinct values of  $x_i$  for  $j = 1, \dots, k_n - 1$ . The number of inter-knot segments  $k_n$  is obtained by calling the function `cub_spline_kn` (option `method="u"`).

In what concerns the monotonicity constraint, we use the following sufficient condition for the monotonicity:

$$\alpha_0 \leq \alpha_1 \leq \dots \leq \alpha_{N-1}$$

. This condition was previously used in Lu et al. (2007) and Pya and Wood (2015). Note that since the condition corresponds to linear constraints on  $\alpha$ , the estimator satisfying the monotonicity can be obtained via linear programming.

When the estimate is required to be both monotone and concave, we use the function `cub_spline_est` with the option `method="mc"`. Such estimate is obtained as the cubic spline function which minimizes the same linear objective function as the unconstrained estimate subject to the same linear envelopment constraints, the monotonicity constraint above and the additional linear concavity constraints  $\pi''(t_j)^T \alpha \leq, j = 0, 1, \dots, k_n$ , where the second derivative  $\pi''$  is a linear spline. Regarding the choice of knots, we just apply the same scheme as for the unconstrained cubic spline estimate.

### Value

Returns a numeric vector with the same length as `x`. Returns a vector of NA if no solution has been found by the solver (GLPK).

### Author(s)

Hohsuk Noh

### References

- Daouia, A., Noh, H. and Park, B.U. (2016). Data Envelope fitting with constrained polynomial splines. *Journal of the Royal Statistical Society: Series B*, **78**(1), 3-30. doi:10.1111/rssb.12098.
- Lu, M., Zhang, Y. and Huang, J. (2007). Estimation of the mean function with panel count data using monotone polynomial splines. *Biometrika*, **94**, 705-718.
- Pya, N. and Wood, S. N. (2015). Shape constrained additive models. *Statistical Computing*, to appear.
- Schumaker, L.L. (2007). *Spline Functions: Basic Theory*, 3rd edition, Cambridge University Press.

### See Also

[cub\\_spline\\_kn](#)

### Examples

```
data("air")
x.air <- seq(min(air$xtab), max(air$xtab), length.out=101)

# 1. Unconstrained cubic spline fits
# Optimal number of inter-knot segments via the BIC criterion
(kn.bic.air.u <- cub_spline_kn(air$xtab, air$ytab,
  method="u", type="BIC"))
# Unconstrained spline estimate
y.cub.air.u <- cub_spline_est(air$xtab, air$ytab,
  x.air, kn=kn.bic.air.u, method="u")
```

```

# 2. Monotonicity constraint
# Optimal number of inter-knot segments via the BIC criterion
(kn.bic.air.m<-cub_spline_kn(air$xtab, air$ytab,
  method="m", type="BIC"))
# Monotonic splines estimate
y.cub.air.m<-cub_spline_est(air$xtab, air$ytab,
  x.air, kn=kn.bic.air.m, method="m")

# 3. Monotonicity and Concavity constraints
# Optimal number of inter-knot segments via the BIC criterion
(kn.bic.air.mc<-cub_spline_kn(air$xtab, air$ytab,
  method="mc", type="BIC"))
# Monotonic/Concave splines estimate
y.cub.air.mc<-cub_spline_est(air$xtab, air$ytab,
  x.air, kn=kn.bic.air.mc, method="mc", all.dea=TRUE)

# Representation
plot(x.air, y.cub.air.u, lty=1, lwd=4, col="green",
  type="l", xlab="log(COST)", ylab="log(OUTPUT)")
lines(x.air, y.cub.air.m, lty=2, lwd=4, col="cyan")
lines(x.air, y.cub.air.mc, lty=3, lwd=4, col="magenta")
points(ytab~xtab, data=air)
legend("topleft", col=c("green","cyan","magenta"),
  lty=c(1,2,3), legend=c("unconstrained", "monotone",
  "monotone + concave"), lwd=4, cex=0.8)

```

---

cub_spline_kn	<i>AIC and BIC criteria for choosing the number of inter-knot segments in cubic spline fits</i>
---------------	---

---

## Description

Computes the optimal number of inter-knot segments for the (un)constrained cubic spline fit proposed by Daouia, Noh and Park (2016).

## Usage

```

cub_spline_kn(xtab, ytab, method, krange = 1:20, type = "AIC",
  control = list("tm_limit" = 700))

```

## Arguments

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
method	a character equal to "u" (unconstrained estimator), "m" (under the monotonicity constraint) or "mc" (under simultaneous monotonicity and concavity constraints).

krange	a vector of integers specifying the range in which the optimal number of inter-knot segments is to be selected.
type	a character equal to "AIC" or "BIC".
control	a list of parameters to the GLPK solver. See *Details* of help(Rglpk_solve_LP).

### Details

The implementation of the unconstrained cubic spline smoother  $\tilde{\varphi}_n$  (see [cub\\_spline\\_est](#)) is based on the knot mesh  $\{t_j\}$ , with  $t_j = x_{[jn/k_n]}$  being the  $j/k_n$ th quantile of the distinct values of  $x_i$  for  $j = 1, \dots, k_n - 1$ . Because the number of knots  $k_n$  determines the complexity of the spline approximation, its choice may then be viewed as model selection through the minimization of the following two information criteria:

$$AIC(k) = \log \left( \sum_{i=1}^n (\tilde{\varphi}_n(x_i) - y_i) \right) + (k + 2)/n,$$

$$BIC(k) = \log \left( \sum_{i=1}^n (\tilde{\varphi}_n(x_i) - y_i) \right) + \log n \cdot (k + 2)/2n.$$

The first one (option type = "AIC") is similar to the famous Akaike information criterion (Akaike, 1973) and the second one (option type = "BIC") to the Bayesian information criterion (Schwartz, 1978). For the implementation of the concave cubic spline estimator, just apply the same scheme as for the unconstrained version.

### Value

Returns an integer.

### Author(s)

Hohsuk Noh.

### References

- Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle, in *Second International Symposium of Information Theory*, eds. B. N. Petrov and F. Csaki, Budapest: Akademia Kiado, 267–281.
- Daouia, A., Noh, H. and Park, B.U. (2016). Data Envelope fitting with constrained polynomial splines. *Journal of the Royal Statistical Society: Series B*, **78**(1), 3-30. doi:10.1111/rssb.12098.
- Schwartz, G. (1978). Estimating the dimension of a model, *Annals of Statistics*, 6, 461–464.

### See Also

[cub\\_spline\\_est](#).

## Examples

```
data("air")
# a. Unconstrained cubic spline fits
(kn.bic.air.u<-cub_spline_kn(air$xtab, air$ytab,
  method="u", type="BIC"))
# b. Monotone cubic spline smoother
(kn.bic.air.m<-cub_spline_kn(air$xtab, air$ytab,
  method="m", type="BIC"))
# c. Monotone and Concave cubic spline smoother
(kn.bic.air.mc<-cub_spline_kn(air$xtab, air$ytab,
  method="mc", type="BIC"))
```

---

 dea\_est

 DEA, FDH and linearized FDH estimators.
 

---

## Description

The function implements the empirical FDH (free disposal hull), LFDH (linearized FDH) and DEA (data envelopment analysis) frontier estimators.

## Usage

```
dea_est(xtab, ytab, x, type = "dea")
```

## Arguments

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
type	a character equal to "dea", "fdh" or "lfdh".

## Details

There are mainly two usual frontier estimation methods for preserving monotonicity: the free disposal hull (FDH) introduced by Deprins *et al.* (1984) and the data envelopment analysis (DEA) initiated by Farrell (1957). The FDH boundary is the lowest “stair-case” monotone curve covering all the data points

$$\varphi_n(x) := \max\{y_i, i : x_i \leq x\}.$$

An improved version of this estimator, referred to as the linearized FDH (LFDH), is obtained by drawing the polygonal line smoothing the staircase FDH curve. It has been considered in Hall and Park (2002) and Jeong and Simar (2006). When the joint support of data is in addition convex, the DEA estimator is defined as the least concave majorant of the FDH frontier (see also Gijbels *et al.* (1999)). We employ the function DEA from the package **Benchmarking** to implement the function dea\_est.

**Value**

Returns a numeric vector with the same length as `x`.

**Author(s)**

Hohsuk Noh.

**References**

Bogetoft, P. and Otto, L. (2011), *Benchmarking with DEA, SFA and R*, Springer-Verlag.

Deprins, D., Simar, L. and H. Tulkens (1984). Measuring labor efficiency in post offices, in *The performance of Public Enterprises: Concepts and Measurements* (M. Marchand, P. Pestieau and H. Tulkens Eds), North-Holland, Amsterdam, 243–267.

Farrell, M.J. (1957). The measurement of productive efficiency. *Journal of the Royal Statistical Society: Series A*, 120, 253–281.

Gijbels, I., Mammen, E., Park, B.U. and Simar, L. (1999). On estimation of monotone and concave frontier functions, *Journal of American Statistical Association*, 94, 220–228.

Hall, P. and Park, B.U. (2002). New methods for bias correction at endpoints and boundaries, *Annals of Statistics*, 30, 1460-1479.

Jeong, S.-O. and Simar, L. (2006). Linearly interpolated FDH efficiency score for nonconvex frontiers, *Journal of Multivariate Analysis*, 97, 2141–2161.

**See Also**

[quad\\_spline\\_est](#), [cub\\_spline\\_est](#).

**Examples**

```
data("green")
plot(OUTPUT~COST, data=green)
x <- seq(min(green$COST), max(green$COST), length.out=1001)
# We compute and represent the DEA, FDH and LFDH estimates
lines(x, dea_est(green$COST, green$OUTPUT, x, type="dea"),
      lty=4, lwd=4, col="cyan")
lines(x, dea_est(green$COST, green$OUTPUT, x, type="fdh"),
      lty=1, lwd=4, col="green")
lines(x, dea_est(green$COST, green$OUTPUT, x, type="lfdh"),
      lty=2, lwd=4, col="magenta")
legend("topleft", legend=c("dea", "fdh", "lfdh"),
      col=c("cyan", "green", "magenta"), lty=c(4,1,2), lwd=4)
```

dfs\_momt

*Moment frontier estimator***Description**

This function is an implementation of the moment-type estimator developed by Daouia, Florens and Simar (2010).

**Usage**

```
dfs_momt(xtab, ytab, x, rho, k, ci=TRUE)
```

**Arguments**

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
rho	a numeric vector of the same length as x or a scalar, which determines the values of rho.
k	a numeric vector of the same length as x or a scalar, which determines the thresholds at which the moment estimator will be computed.
ci	a boolean, TRUE for computing the confidence interval.

**Details**

Combining ideas from Dekkers, Einmahl and de Haan (1989) with the dimensionless transformation  $\{z_i^x := y_i \mathbf{1}_{\{x_i \leq x\}}, i = 1, \dots, n\}$  of the observed sample  $\{(x_i, y_i), i = 1, \dots, n\}$ , the authors estimate the conditional endpoint  $\varphi(x)$  by

$$\tilde{\varphi}_{momt}(x) = z_{(n-k)}^x + z_{(n-k)}^x M_n^{(1)} \{1 + \rho_x\}$$

where  $M_n^{(1)} = (1/k) \sum_{i=0}^{k-1} (\log z_{(n-i)}^x - \log z_{(n-k)}^x)$ ,  $z_{(1)}^x \leq \dots \leq z_{(n)}^x$  are the ascending order statistics corresponding to the transformed sample  $\{z_i^x, i = 1, \dots, n\}$  and  $\rho_x > 0$  is referred to as the extreme-value index and has the following interpretation: when  $\rho_x > 2$ , the joint density of data decays smoothly to zero at a speed of power  $\rho_x - 2$  of the distance from the frontier; when  $\rho_x = 2$ , the density has sudden jumps at the frontier; when  $\rho_x < 2$ , the density increases toward infinity at a speed of power  $\rho_x - 2$  of the distance from the frontier. Most of the contributions to econometric literature on frontier analysis assume that the joint density is strictly positive at its support boundary, or equivalently,  $\rho_x = 2$  for all  $x$ . When  $\rho_x$  is unknown, Daouia et al. (2010) suggest to use the following two-step estimator: First, estimate  $\rho_x$  by the moment estimator  $\tilde{\rho}_x$  implemented in the function `rho_momt_pick` by utilizing the option `method="moment"`, or by the Pickands estimator  $\hat{\rho}_x$  by using the option `method="pickands"`. Second, use the estimator  $\tilde{\varphi}_{momt}(x)$ , as if  $\rho_x$  were known, by substituting the estimated value  $\tilde{\rho}_x$  or  $\hat{\rho}_x$  in place of  $\rho_x$ . The 95% confidence interval of  $\varphi(x)$  derived from the asymptotic normality of  $\tilde{\varphi}_{momt}(x)$  is given by

$$[\tilde{\varphi}_{momt}(x) \pm 1.96 \sqrt{V(\rho_x)/k z_{(n-k)}^x} M_n^{(1)} (1 + 1/\rho_x)]$$

where  $V(\rho_x) = \rho_x^2(1 + 2/\rho_x)^{-1}$ . The sample fraction  $k = k_n(x)$  plays here the role of the smoothing parameter and varies between 1 and  $N_x - 1$ , with  $N_x = \sum_{i=1}^n \mathbf{1}_{\{x_i \leq x\}}$  being the number of observations  $(x_i, y_i)$  with  $x_i \leq x$ . See [kopt\\_momt\\_pick](#) for an automatic data-driven rule for selecting  $k$ .

### Value

Returns a numeric vector with the same length as  $x$ .

### Note

As it is common in extreme-value theory, good results require a large sample size  $N_x$  at each evaluation point  $x$ . See also the note in [kopt\\_momt\\_pick](#).

### Author(s)

Abdelaati Daouia and Thibault Laurent (converted from Leopold Simar's Matlab code).

### References

Daouia, A., Florens, J.P. and Simar, L. (2010). Frontier Estimation and Extreme Value Theory, *Bernoulli*, 16, 1039-1063.

Dekkers, A.L.M., Einmahl, J.H.J. and L. de Haan (1989), A moment estimator for the index of an extreme-value distribution, *nnals of Statistics*, 17, 1833-1855.

### See Also

[dfs\\_pick](#), [kopt\\_momt\\_pick](#).

### Examples

```
data("post")
x.post <- seq(post$input[100], max(post$input),
  length.out = 100)
# 1. When rho[x] is known and equal to 2, we set:
rho <- 2
# To determine the sample fraction k=k[n](x)
# in tilde(varphi[momt])(x).
best_kn.1 <- kopt_momt_pick(post$input, post$yprod,
  x.post, rho = rho)
# To compute the frontier estimates and confidence intervals:
res.momt.1 <- dfs_momt(post$input, post$yprod, x.post,
  rho = rho, k = best_kn.1)
# Representation
plot(yprod~input, data = post, xlab = "Quantity of labor",
  ylab = "Volume of delivered mail")
lines(x.post, res.momt.1[,1], lty = 1, col = "cyan")
lines(x.post, res.momt.1[,2], lty = 3, col = "magenta")
lines(x.post, res.momt.1[,3], lty = 3, col = "magenta")

## Not run:
```

```

# 2. rho[x] is unknown and estimated by
# the Pickands estimator tilde(rho[x])
rho_momt <- rho_momt_pick(post$xinput, post$yprod,
  x.post)
best_kn.2 <- kopt_momt_pick(post$xinput, post$yprod,
  x.post, rho = rho_momt)
res.momt.2 <- dfs_momt(post$xinput, post$yprod, x.post,
  rho = rho_momt, k = best_kn.2)
# 3. rho[x] is unknown independent of x and estimated
# by the (trimmed) mean of tilde(rho[x])
rho_trimmean <- mean(rho_momt, trim=0.00)
best_kn.3 <- kopt_momt_pick(post$xinput, post$yprod,
  x.post, rho = rho_trimmean)
res.momt.3 <- dfs_momt(post$xinput, post$yprod, x.post,
  rho = rho_trimmean, k = best_kn.3)

# Representation
plot(yprod~xinput, data = post, col = "grey",
  xlab = "Quantity of labor", ylab = "Volume of delivered mail")
lines(x.post, res.momt.2[,1], lty = 1, lwd = 2, col = "cyan")
lines(x.post, res.momt.2[,2], lty = 3, lwd = 4, col = "magenta")
lines(x.post, res.momt.2[,3], lty = 3, lwd = 4, col = "magenta")
plot(yprod~xinput, data = post, col = "grey",
  xlab = "Quantity of labor", ylab = "Volume of delivered mail")
lines(x.post, res.momt.3[,1], lty = 1, lwd = 2, col = "cyan")
lines(x.post, res.momt.3[,2], lty = 3, lwd = 4, col = "magenta")
lines(x.post, res.momt.3[,3], lty = 3, lwd = 4, col = "magenta")

## End(Not run)

```

---

dfs\_pick

*Pickands frontier estimator*


---

## Description

This function is an implementation of the Pickands type estimator developed by Daouia, Florens and Simar (2010).

## Usage

```
dfs_pick(xtab, ytab, x, k, rho, ci=TRUE)
```

## Arguments

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.

k	a numeric vector of the same length as x or a scalar, which determines the thresholds at which the Pickands estimator will be computed.
rho	a numeric vector of the same length as x or a scalar, which determines the values of rho.
ci	a boolean, TRUE for computing the confidence interval.

### Details

Built on the ideas of Dekkers and de Haan (1989), Daouia et al. (2010) propose to estimate the frontier point  $\varphi(x)$  by

$$\hat{\varphi}_{pick}(x) = \frac{z_{(n-k+1)}^x - z_{(n-2k+1)}^x}{2^{1/\rho_x} - 1} + z_{(n-k+1)}^x$$

from the transformed data  $\{z_i^x, i = 1, \dots, n\}$  described in [dfs\\_momt](#), where  $\rho_x > 0$  is the same tail-index as in [dfs\\_momt](#). If  $\rho_x$  is known (typically equal to 2 if the joint density of data is believed to have sudden jumps at the frontier), then one can use the estimator  $\hat{\varphi}_{pick}(x)$  in conjunction with the function [kopt\\_momt\\_pick](#) which implements an automatic data-driven method for selecting the threshold  $k$ . In contrast, if  $\rho_x$  is unknown, one could consider using the following two-step estimator: First, estimate  $\rho_x$  by the Pickands estimator  $\hat{\rho}_x$  implemented in the function [rho\\_momt\\_pick](#) by using the option `method="pickands"`, or by the moment estimator  $\tilde{\rho}_x$  by utilizing the option `method="moment"`. Second, use the estimator  $\hat{\varphi}_{pick}(x)$ , as if  $\rho_x$  were known, by substituting the estimated value  $\hat{\rho}_x$  or  $\tilde{\rho}_x$  in place of  $\rho_x$ . The pointwise 95% confidence interval of the frontier function obtained from the asymptotic normality of  $\hat{\varphi}_{pick}(x)$  is given by

$$[\hat{\varphi}_{pick}(x) \pm 1.96\sqrt{v(\rho_x)/(2k)}(z_{(n-k+1)}^x - z_{(n-2k+1)}^x)]$$

where  $v(\rho_x) = \rho_x^{-2}2^{-2/\rho_x}/(2^{-1/\rho_x} - 1)^4$ . Finally, to select the threshold  $k = k_n(x)$ , one could use the automatic data-driven method of Daouia et al. (2010) implemented in the function [kopt\\_momt\\_pick](#) (option `method="pickands"`).

### Value

Returns a numeric vector with the same length as x.

### Note

As it is common in extreme-value theory, good results require a large sample size  $N_x$  at each evaluation point  $x$ . See also the note in [kopt\\_momt\\_pick](#).

### Author(s)

Abdelaati Daouia and Thibault Laurent (converted from Leopold Simar's Matlab code).

### References

- Daouia, A., Florens, J.P. and Simar, L. (2010). Frontier Estimation and Extreme Value Theory, *Bernoulli*, 16, 1039-1063.
- Dekkers, A.L.M., Einmahl, J.H.J. and L. de Haan (1989), A moment estimator for the index of an extreme-value distribution, *Annals of Statistics*, 17, 1833-1855.

**See Also**

[dfs\\_momt](#), [kopt\\_momt\\_pick](#).

**Examples**

```

data("post")
x.post<- seq(post$xinput[100],max(post$xinput),
  length.out=100)
# 1. When rho[x] is known and equal to 2, we set:
rho<-2
# To determine the sample fraction k=k[n](x)
# in hat(varphi[pick])(x).
best_kn.1<-kopt_momt_pick(post$xinput, post$yprod,
  x.post, method="pickands", rho=rho)
# To compute the frontier estimates and confidence intervals:
res.pick.1<-dfs_pick(post$xinput, post$yprod, x.post,
  rho=rho, k=best_kn.1)
# Representation
plot(yprod~xinput, data=post, xlab="Quantity of labor",
  ylab="Volume of delivered mail")
lines(x.post, res.pick.1[,1], lty=1, col="cyan")
lines(x.post, res.pick.1[,2], lty=3, col="magenta")
lines(x.post, res.pick.1[,3], lty=3, col="magenta")

## Not run:
# 2. rho[x] is unknown and estimated by
# the Pickands estimator hat(rho[x])
rho_pick<-rho_momt_pick(post$xinput, post$yprod,
  x.post, method="pickands")
best_kn.2<-kopt_momt_pick(post$xinput, post$yprod,
  x.post, method="pickands", rho=rho_pick)
res.pick.2<-dfs_pick(post$xinput, post$yprod, x.post,
  rho=rho_pick, k=best_kn.2)
# 3. rho[x] is unknown independent of x and estimated
# by the (trimmed) mean of hat(rho[x])
rho_trimmean<-mean(rho_pick, trim=0.00)
best_kn.3<-kopt_momt_pick(post$xinput, post$yprod,
  x.post, rho=rho_trimmean, method="pickands")
res.pick.3<-dfs_pick(post$xinput, post$yprod, x.post,
  rho=rho_trimmean, k=best_kn.3)

# Representation
plot(yprod~xinput, data=post, col="grey", xlab="Quantity of labor",
  ylab="Volume of delivered mail")
lines(x.post, res.pick.2[,1], lty=1, lwd=2, col="cyan")
lines(x.post, res.pick.2[,2], lty=3, lwd=4, col="magenta")
lines(x.post, res.pick.2[,3], lty=3, lwd=4, col="magenta")
plot(yprod~xinput, data=post, col="grey", xlab="Quantity of labor",
  ylab="Volume of delivered mail")
lines(x.post, res.pick.3[,1], lty=1, lwd=2, col="cyan")
lines(x.post, res.pick.3[,2], lty=3, lwd=4, col="magenta")
lines(x.post, res.pick.3[,3], lty=3, lwd=4, col="magenta")

```

```
## End(Not run)
```

---

```
dfs_pwm
```

*Probability-weighted moment frontier estimator*

---

## Description

This function is an implementation of the probability-weighted moment frontier estimator developed by Daouia, Florens and Simar (2012).

## Usage

```
dfs_pwm(xtab, ytab, x, coefm, a=2, rho, ci=TRUE)
```

## Arguments

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
coefm	a tuning parameter (integer) larger than or equal to 1.
a	a smoothing parameter (integer) larger than or equal to 2.
rho	a numeric vector of the same length as x or a scalar, which determines the values of rho.
ci	a boolean, TRUE for computing the confidence interval.

## Details

The regularized frontier estimator introduced by Daouia et al. (2012) is based on the unregularized probability-weighted moment estimator

$$\hat{\varphi}_m(x) = \varphi_{fdh}(x) - \int_0^{\varphi_{fdh}(x)} \hat{F}^m(y|x) dy$$

where the trimming order  $m \geq 1$  is an integer such that  $m = m_n \rightarrow \infty$  as  $n \rightarrow \infty$ , and  $\hat{F}(y|x) = \sum_{i=1}^n 1_{(x_i \leq x, y_i \leq y)} / \sum_{i=1}^n 1_{(x_i \leq x)}$ . The implemented estimator of  $\varphi(x)$  is then defined as

$$\tilde{\varphi}_m(x) = \hat{\varphi}_m(x) + \Gamma(1 + 1/\bar{\rho}_x) \left(1/m \hat{\ell}_x\right)^{1/\bar{\rho}_x}$$

where

$$\bar{\rho}_x = \log(a) \left\{ \log \left( \frac{\hat{\varphi}_m(x) - \hat{\varphi}_{am}(x)}{\hat{\varphi}_{am}(x) - \hat{\varphi}_{a^2m}(x)} \right) \right\}^{-1}, \quad \hat{\ell}_x = \frac{1}{m} \left[ \frac{\Gamma(1 + 1/\bar{\rho}_x)(1 - a^{-1/\bar{\rho}_x})}{\hat{\varphi}_m(x) - \hat{\varphi}_{am}(x)} \right]^{\bar{\rho}_x},$$

with  $a \geq 2$  being a fixed integer. If the true tail-index  $\rho_x = \beta_x + 2$  is known, we set  $\bar{\rho}_x = \rho_x$  in the expressions above. The two smoothing parameters  $m$  and  $a$  have to be fixed by the user in the 4th and 5th arguments of the function.

The pointwise 95% confidence interval of  $\varphi(x)$  derived from the asymptotic normality of  $\tilde{\varphi}_m(x)$  is given by  $[\tilde{\varphi}_m(x) \pm 1.96 \hat{\sigma}(m, x)/\sqrt{n}]$  where

$$\hat{\sigma}^2(m, x) = \frac{2m^2}{\hat{F}_X(x)} \int_0^{\varphi_{fah}(x)} \int_0^{\varphi_{fah}(x)} \hat{F}^m(y|x) \hat{F}^{m-1}(u|x) (1 - \hat{F}(u|x)) 1_{(y \leq u)} dy du,$$

with  $\hat{F}_X(x) = (1/n) \sum_{i=1}^n 1_{(x_i \leq x)}$ . Note that the standard deviation  $\sigma(m, x)/\sqrt{n}$  of the bias-corrected estimator  $\tilde{\varphi}_m(x)$  is adjusted by a bootstrap estimator in the numerical illustrations of Daouia et al. (2012), whereas the exact estimate  $\hat{\sigma}(m, x)/\sqrt{n}$  is utilized in the implemented function. A practical choice of  $m$  that Daouia et al. (2012) have employed is the simple rule of thumb  $m = \text{coefm} \times N_x^{1/3}$ , where  $N_x = \sum_{i=1}^n 1_{\{x_i \leq x\}}$ , and the integer  $\text{coefm}$  as well as the second smoothing parameter  $a$  are to be tuned by the user to avoid numerical instabilities in the pointwise estimates of the tail-index  $\rho_x$  and the frontier function  $\varphi(x)$ . The user may start with the values  $\text{coefm}=5$  and  $a=2$  [respectively,  $\text{coefm}=10$  and  $a=20$ ] for computing the estimator  $\tilde{\varphi}_m(x)$  [respectively,  $\bar{\rho}_x$ ]. Note that tail-index estimation and frontier estimation are conducted separately.

### Value

Returns a numeric vector with the same length as  $x$ .

### Note

The computational burden here is demanding, so be forewarned.

### Author(s)

Abdelaati Daouia and Thibault Laurent (converted from Abdelaati Daouia's Matlab code).

### References

Daouia, A., Florens, J.-P. and Simar, L. (2012). Regularization of Nonparametric Frontier Estimators. *Journal of Econometrics*, 168, 285-299.

### See Also

[rho\\_pwm](#), [mopt\\_pwm](#).

### Examples

```
data("post")
x.post<- seq(post$input[100],max(post$input),
  length.out=100)
## Not run:
# 1. When rho[x] is known and equal to 2, we set:
rho<-2
res.pwm.1<- dfs_pwm(post$input, post$yprod, x.post, coefm=5,
  a=2, rho, ci=TRUE)
```

```

# 2. When rho[x] is unknown and dependent of x,
# its estimate hat(rho[x]) is obtained via:
rho_pwm <- rho_pwm(post$xinput, post$yprod, x.post, coefm=10, a=20)
# and the corresponding frontier estimator via:
res_pwm.2<- dfs_pwm(post$xinput, post$yprod, x.post, coefm=5,
  a=2, rho_pwm, ci=TRUE)
# 3. When rho[x] is unknown but independent of x,
# a robust estimation strategy is by using the (trimmed) mean
# over the estimates hat(rho[x]):
rho_trimmean<-mean(rho_pwm, trim=0.00)
res_pwm.3<- dfs_pwm(post$xinput, post$yprod, x.post, coefm=5,
  a=2, rho_trimmean, ci=TRUE)

## End(Not run)

```

---

green

*American electric utility companies*


---

## Description

The dataset consists of 123 American electric utility companies. As in the set-up of Gijbels *et al.* (1999), we used the measurements of the variables  $y_i = \log(q_i)$  and  $x_i = \log(c_i)$ , where  $q_i$  is the production output of the company  $i$  and  $c_i$  is the total cost involved in the production. For a detailed description and analysis of these data see, *e.g.*, Christensen and Greene (1976) and Greene (1990).

## Usage

```
data(green)
```

## Format

A data frame with 123 observations on the following 2 variables.

COST a numeric vector.

OUTPUT a numeric vector.

## Source

Gijbels *et al.* (1999).

## References

Christensen, L.R. and Greene, W.H. (1976). Economies of Scale in U.S. Electric Power Generation, *Journal of Political Economy*, University of Chicago Press, 84, 655-76.

Gijbels, I., Mammen, E., Park, B.U. and Simar, L. (1999). On estimation of monotone and concave frontier functions. *Journal of American Statistical Association*, 94, 220-228.

Greene, W.H. (1990). A Gamma-distributed stochastic frontier model, *Journal of Econometrics*, 46, 141-163.

**Examples**

```
data("green")
```

---

kern_smooth	<i>Frontier estimation via kernel smoothing</i>
-------------	---

---

**Description**

The function kern\_smooth implements two frontier estimators based on kernel smoothing techniques. One is from Noh (2014) and the other is from Parmeter and Racine (2013).

**Usage**

```
kern_smooth(xtab, ytab, x, h, method="u", technique="noh",
  control = list("tm_limit" = 700))
```

**Arguments**

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
h	determines the bandwidth at which the smoothed kernel estimate will be computed.
method	a character equal to "u" (unconstrained estimator), "m" (under the monotonicity constraint) or "mc" (under simultaneous monotonicity and concavity constraints).
technique	which estimation method to use. "Noh" specifies the use of the method in Noh (2014) and "pr" is for the method in Parmeter and Racine (2013).
control	a list of parameters to the GLPK solver. See *Details* of help(Rglpk_solve_LP).

**Details**

To estimate the frontier function, Parmeter and Racine (2013) considered the following generalization of linear regression smoothers

$$\hat{\varphi}(x|p) = \sum_{i=1}^n p_i A_i(x) y_i,$$

where  $A_i(x)$  is the kernel weight function of  $x$  for the  $i$ th data depending on  $x_i$ 's and the sort of linear smoothers. For example, the Nadaraya-Watson kernel weights are  $A_i(x) = K_i(x) / (\sum_{j=1}^n K_j(x))$ , where  $K_i(x) = h^{-1} K\{(x - x_i)/h\}$ , with the kernel function  $K$  being a bounded and symmetric probability density, and  $h$  is a bandwidth. Then, the weight vector  $p = (p_1, \dots, p_n)^T$  is chosen to minimize the distance  $D(p) = (p - p_u)^T (p - p_u)$  subject to the envelopment constraints and the

choice of the shape constraints, where  $p_u$  is an  $n$ -dimensional vector with all elements being one. The envelopement and shape constraints are

$$\begin{aligned}\hat{\varphi}(x_i|p) - y_i &= \sum_{i=1}^n p_i A_i(x_i) y_i - y_i \geq 0, \quad i = 1, \dots, n; \quad (\text{envelopment constraints}) \\ \hat{\varphi}^{(1)}(x|p) &= \sum_{i=1}^n p_i A_i^{(1)}(x) y_i \geq 0, \quad x \in \mathcal{M}; \quad (\text{monotonicity constraints}) \\ \hat{\varphi}^{(2)}(x|p) &= \sum_{i=1}^n p_i A_i^{(2)}(x) y_i \leq 0, \quad x \in \mathcal{C}, \quad (\text{concavity constraints})\end{aligned}$$

where  $\hat{\varphi}^{(s)}(x|p) = \sum_{i=1}^n p_i A_i^{(s)}(x) y_i$  is the  $s$ th derivative of  $\hat{\varphi}(x|p)$ , with  $\mathcal{M}$  and  $\mathcal{C}$  being the collections of points where monotonicity and concavity are imposed, respectively. In our implementation of the estimator, we simply take the entire dataset  $\{(x_i, y_i), i = 1, \dots, n\}$  to be  $\mathcal{M}$  and  $\mathcal{C}$  and, in case of small samples, we augment the sample points by an equispaced grid of length 201 over the observed support  $[\min_i x_i, \max_i x_i]$  of  $X$ . For the weight  $A_i(x)$ , we use the Nadaraya-Watson weights.

Noh (2014) considered the same generalization of linear smoothers  $\hat{\varphi}(x|p)$  for frontier estimation, but with a difference choice of the weight  $p$ . Using the same envelopement and shape constraints as Parmeter and Racine (2013), the weight vector  $p$  is chosen to minimize the area under the fitted curve  $\hat{\varphi}(x|p)$ , that is  $A(p) = \int_a^b \hat{\varphi}(x|p) dx = \sum_{i=1}^n p_i y_i \left( \int_a^b A_i(x) dx \right)$ , where  $[a, b]$  is the true support of  $X$ . In practice, we integrate over the observed support  $[\min_i x_i, \max_i x_i]$  since the theoretic one is unknown. In what concerns the kernel weights  $A_i(x)$ , we use the Priestley-Chao weights

$$A_i(x) = \begin{cases} 0 & , i = 1 \\ (x_i - x_{i-1})K_i(x) & , i \neq 1 \end{cases} ,$$

where it is assumed that the pairs  $(x_i, y_i)$  have been ordered so that  $x_1 \leq \dots \leq x_n$ . The choice of such weights is motivated by their convenience for the evaluation of the integral  $\int A_i(x) dx$ .

## Value

Returns a numeric vector with the same length as  $x$ . Returns a vector of NA if no solution has been found by the solver (GLPK).

## Author(s)

Hohsuk Noh

## References

Noh, H. (2014). Frontier estimation using kernel smoothing estimators with data transformation. *Journal of the Korean Statistical Society*, 43, 503-512.

Parmeter, C.F. and Racine, J.S. (2013). Smooth constrained frontier analysis in *Recent Advances and Future Directions in Causality, Prediction, and Specification Analysis*, Springer-Verlag, New York, 463-488.

## See Also

[kern\\_smooth\\_bw](#).

## Examples

```
## Not run:
data("green")
x.green <- seq(min(log(green$COST)), max(log(green$COST)),
  length.out = 101)
options(np.tree=TRUE, crs.messages=FALSE, np.messages=FALSE)
# 1. Unconstrained
(h.bic.green.u <- kern_smooth_bw(log(green$COST),
  log(green$OUTPUT), method = "u", technique = "noh",
  bw_method = "bic"))
y.ks.green.u <- kern_smooth(log(green$COST),
  log(green$OUTPUT), x.green, h = h.bic.green.u,
  method = "u", technique = "noh")

# 2. Monotonicity constraint
(h.bic.green.m <- kern_smooth_bw(log(green$COST),
  log(green$OUTPUT), method = "m", technique = "noh",
  bw_method = "bic"))
y.ks.green.m <- kern_smooth(log(green$COST),
  log(green$OUTPUT), x.green, h = h.bic.green.m,
  method = "m", technique = "noh")

# 3. Monotonicity and Concavity constraints
(h.bic.green.mc <- kern_smooth_bw(log(green$COST), log(green$OUTPUT),
  method="mc", technique="noh", bw_method="bic"))
y.ks.green.mc <- kern_smooth(log(green$COST),
  log(green$OUTPUT), x.green, h=h.bic.green.mc, method="mc",
  technique="noh")

# Representation
plot(log(OUTPUT)~log(COST), data=green, xlab="log(COST)",
  ylab="log(OUTPUT)")
lines(x.green, y.ks.green.u, lty=1, lwd=4, col="green")
lines(x.green, y.ks.green.m, lty=2, lwd=4, col="cyan")
lines(x.green, y.ks.green.mc, lty=3, lwd=4, col="magenta")
legend("topleft", col=c("green", "cyan", "magenta"),
  lty=c(1,2,3), legend=c("unconstrained", "monotone",
  "monotone + concave"), lwd=4, cex=0.8)

## End(Not run)
```

---

kern\_smooth\_bw

*Bandwidth selection for kernel smoothing frontier estimators*


---

## Description

The function `kern_smooth_bw` provides two bandwidth selection methods. One is the least squares cross-validation developed by Parmeter and Racine (2013). The other is the BIC developed in Noh (2014).

**Usage**

```
kern_smooth_bw(xtab, ytab, method="u", technique="noh", bw_method="bic",
  control = list("tm_limit" = 700))
```

**Arguments**

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
method	a character equal to "u" (unconstrained estimator), "m" (under the monotonicity constraint) or "mc" (under simultaneous monotonicity and concavity constraints).
technique	which estimation technique to use: "Noh" specifies the use of the method in Noh (2014), while "pr" is for the method in Parmeter and Racine (2013).
bw_method	which bandwidth selection method to use: "cv" returns the bandwidth that minimizes the least squares cross-validation criterion, and "bic" returns the bandwidth minimizing the BIC.
control	a list of parameters to the GLPK solver. See *Details* of help(Rglpk_solve_LP).

**Details**

As with any smoothed techniques, the bandwidth selection is critical to the quality of the frontier estimator. Parmeter and Racine (2013)'s recommendation is to use the least squares cross-validation method implemented with `bw_method="cv"` in the function `kern_smooth_bw`. Instead, Noh (2014) proposed to select the bandwidth which minimizes the following criterion:

$$BIC(h) = \log \left( \sum_{i=1}^n (\hat{\varphi}(x_i | \hat{p}(h)) - y_i) \right) + \frac{\log n \cdot \text{tr}(S(h))}{2n},$$

where  $\hat{p}(h)$  is the chosen weight vector associated to the bandwidth  $h$ , and  $\text{tr}(S(h))$  is the trace of the smoothing matrix

$$S(h) = \begin{pmatrix} A_1(x_1) & \cdots & A_n(x_1) \\ \vdots & \ddots & \vdots \\ A_1(x_n) & \cdots & A_n(x_n) \end{pmatrix}.$$

The function `kern_smooth_bw` computes the optimal bandwidth from this criterion with option `bw_method="bic"`.

**Value**

Returns an optimal bandwidth depending on the specified selection method.

**Author(s)**

Hohsuk Noh

## References

- Noh, H. (2014). Frontier estimation using kernel smoothing estimators with data transformation. *Journal of the Korean Statistical Society*, 43, 503-512.
- Parmeter, C.F. and Racine, J.S. (2013). Smooth constrained frontier analysis in *Recent Advances and Future Directions in Causality, Prediction, and Specification Analysis*, Springer-Verlag, New York, 463-488.

## See Also

[kern\\_smooth](#).

## Examples

```
## Not run:
data("green")
x.green <- seq(min(log(green$COST)), max(log(green$COST)),length.out=101)
options(np.tree=TRUE,crs.messages=FALSE,np.messages=FALSE)
h.pr.green.m<-kern_smooth_bw(log(green$COST),log(green$OUTPUT), method="m",
  technique="pr", bw_method="cv")
h.noh.green.m<-kern_smooth_bw(log(green$COST),log(green$OUTPUT), method="m",
  technique="noh", bw_method="bic")
y.pr.green.m<-kern_smooth(log(green$COST),log(green$OUTPUT), x.green,
  h=h.pr.green.m, method="m", technique="pr")
y.noh.green.m<-kern_smooth(log(green$COST),log(green$OUTPUT), x.green,
  h=h.noh.green.m, method="m", technique="noh")
plot(log(OUTPUT)~log(COST), data=green, xlab="log(COST)",ylab="log(OUTPUT)")
lines(x.green, y.pr.green.m, lwd=4, lty=3, col="red")
lines(x.green, y.noh.green.m, lwd=4, lty=3, col="blue")
legend("topleft", col=c("blue","red"),lty=3, legend=c("noh","pr"),
  lwd=4, cex=0.8)

## End(Not run)
```

---

kopt\_momt\_pick

*Optimal k in moment and Pickands frontier estimators*

---

## Description

This function gives the optimal sample fraction  $k$  in the moment and Pickands type of estimators introduced by Daouia, Florens and Simar (2010).

## Usage

```
kopt_momt_pick(xtab, ytab, x, rho, method="moment", wind.coef=0.1)
```

**Arguments**

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
rho	a numeric vector of the same length as x or a scalar, which determines the values of rho.
method	a character equal to "moment" or "pickands".
wind.coef	a scalar coefficient to be selected in the interval (0,1].

**Details**

This function is an implementation of an experimental method by Daouia et al. (2010) for the automated threshold selection (choice of  $k = k_n(x)$ ) for the moment frontier estimator  $\tilde{\varphi}_{momt}(x)$  [see [dfs\\_momt](#)] in case method="moment" and for the Pickands frontier estimator  $\hat{\varphi}_{pick}(x)$  [see [dfs\\_pick](#)] in case method="pickands". The idea is to select first (for each  $x$ ) a grid of values for the sample fraction  $k_n(x)$  given by  $k = 1, \dots, [\sqrt{N_x}]$ , where  $[\sqrt{N_x}]$  stands for the integer part of  $\sqrt{N_x}$  with  $N_x = \sum_{i=1}^n 1_{\{x_i \leq x\}}$ , and then select the  $k$  where the variation of the results is the smallest. To achieve this here, Daouia et al. (2010) compute the standard deviations of  $\tilde{\varphi}_{momt}(x)$  [option method="moment"] or  $\hat{\varphi}_{pick}(x)$  [option method="pickands"] over a "window" of size  $\max(3, [wind.coef \times \sqrt{N_x}/2])$ , where the coefficient `wind.coef` should be selected in the interval (0, 1] in such a way to avoid numerical instabilities. The default option `wind.coef=0.1` corresponds to having a window large enough to cover around 10% of the possible values of  $k$  in the selected range of values for  $k_n(x)$ . The value of  $k$  where the standard deviation is minimal defines the desired sample fraction  $k_n(x)$ .

**Value**

Returns a numeric vector with the same length as x.

**Note**

In order to choose a reasonable estimate  $\tilde{\varphi}_{momt}(x) = \tilde{\varphi}_{momt}(x, k)$  [see [dfs\\_momt](#)] and  $\hat{\varphi}_{pick}(x) = \hat{\varphi}_{pick}(x, k)$  [see [dfs\\_pick](#)] of the frontier function  $\varphi(x)$ , for each fixed  $x$ , one can construct the plot of the estimator of interest, consisting of the points  $\{(k, \tilde{\varphi}_{momt}(x, k))\}_k$  or  $\{(k, \hat{\varphi}_{pick}(x, k))\}_k$ , and select a value of the estimate at which the obtained graph looks stable. This is this kind of idea which guides the proposed automatic data-driven rule for a chosen grid of values of  $x$ . The main difficulty with such a method is that the plots of  $\tilde{\varphi}_{momt}(x, k)$  or  $\hat{\varphi}_{pick}(x, k)$  as functions of  $k$ , for each  $x$ , may be so unstable that reasonable values of  $k$  [which would correspond to the true value of  $\varphi(x)$ ] may be hidden in the graphs. In result, the obtained frontier estimator may exhibit considerable volatility as a function of  $x$ . One way to avoid such instabilities is by tuning the choice of the parameter `wind.coef` in the interval (0,1]. Note that the default value is `wind.coef=0.1`. The user can also improve appreciably the estimation of  $\varphi(x)$  by refining the estimation of the extreme-value index  $\rho_x$  (see [rho\\_momt\\_pick](#) for details).

**Author(s)**

Abdelaati Daouia and Thibault Laurent (converted from Leopold Simar's Matlab code).

## References

Daouia, A., Florens, J.P. and Simar, L. (2010). Frontier Estimation and Extreme Value Theory, *Bernoulli*, 16, 1039-1063.

Dekkers, A.L.M., Einmahl, J.H.J. and L. de Haan (1989), A moment estimator for the index of an extreme-value distribution, *Annals of Statistics*, 17, 1833-1855.

## See Also

[dfs\\_momt](#), [dfs\\_pick](#).

## Examples

```
data("post")
x.post<- seq(post$xinput[100],max(post$xinput),
  length.out=100)
# When rho[x] is known and equal to 2, we set:
rho<-2
# a. Optimal k in Pickands frontier estimators
best_kn.pick<-kopt_momt_pick(post$xinput, post$yprod,
  x.post, method="pickands", rho=rho)
# b. Optimal k in moment frontier estimators
## Not run:
best_kn.momt<-kopt_momt_pick(post$xinput, post$yprod,
  x.post, rho=rho)

## End(Not run)
```

---

loc\_est

*Local linear frontier estimator*

---

## Description

Computes the local linear smoothing frontier estimator of Hall, Park and Stern (1998) and Hall and Park (2004).

## Usage

```
loc_est(xtab, ytab, x, h, method="u", control = list("tm_limit" = 700))
```

## Arguments

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
h	determines the bandwidth at which the local linear estimate will be computed.
method	a character equal to "u" (unconstrained estimator) or "m" (improved version of the unconstrained estimator).
control	a list of parameters to the GLPK solver. See *Details* of help(Rglpk_solve_LP).

## Details

In the unconstrained case (option method="u"), the implemented estimator of  $\varphi(x)$  is defined by

$$\hat{\varphi}_{n,LL}(x) = \min \left\{ z : \text{there exists } \theta \text{ such that } y_i \leq z + \theta(x_i - x) \right. \\ \left. \text{for all } i \text{ such that } x_i \in (x - h, x + h) \right\},$$

where the bandwidth  $h$  has to be fixed by the user in the 4th argument of the function. This estimator may lack of smoothness in case of small samples and has no guarantee of being monotone even if the true frontier is so. Following the curvature of the monotone frontier  $\varphi$ , the unconstrained estimator  $\hat{\varphi}_{n,LL}$  is likely to exhibit substantial bias, especially at the sample boundaries (see Daouia et al (2016) for numerical illustrations). A simple way to remedy to this drawback is by imposing the extra condition  $\theta \geq 0$  in the definition of  $\hat{\varphi}_{n,LL}(x)$  to get

$$\tilde{\varphi}_{n,LL}(x) = \min \left\{ z : \text{there exists } \theta \geq 0 \text{ such that } y_i \leq z + \theta(x_i - x) \right. \\ \left. \text{for all } i \text{ such that } x_i \in (x - h, x + h) \right\}.$$

As shown in Daouia et al (2016), this version only reduces the vexing bias and border defects of the original estimator when the true frontier is monotone. The option method="m" indicates that the improved fit  $\tilde{\varphi}_{n,LL}(x)$  should be utilized in place of  $\hat{\varphi}_{n,LL}(x)$ . Hall and Park (2004) proposed a bootstrap procedure for selecting the optimal bandwidth  $h$  in  $\hat{\varphi}_{n,LL}(x)$  and  $\tilde{\varphi}_{n,LL}(x)$  (see the function [loc\\_est\\_bw](#)).

## Value

Returns a numeric vector with the same length as  $x$ . Returns a vector of NA if no solution has been found by the solver (GLPK).

## Author(s)

Hohsuk Noh.

## References

- Daouia, A., Noh, H. and Park, B.U. (2016). Data Envelope fitting with constrained polynomial splines. *Journal of the Royal Statistical Society: Series B*, **78**(1), 3-30. doi:10.1111/rssb.12098.
- Hall, P. and Park, B.U. (2004). Bandwidth choice for local polynomial estimation of smooth boundaries. *Journal of Multivariate Analysis*, **91**, 240-261.
- Hall, P., Park, B.U. and Stern, S.E. (1998). On polynomial estimators of frontiers and boundaries. *Journal of Multivariate Analysis*, **66**, 71-98.

## See Also

[loc\\_est\\_bw](#).

**Examples**

```

data("nuclear")
x.nucl <- seq(min(nuclear$xtab), max(nuclear$xtab),
  length.out=101)
# 1. Unconstrained estimator
# Optimal bandwidths over 100 bootstrap replications
## Not run:
h.nucl.u <- loc_est_bw(nuclear$xtab, nuclear$ytab,
  x.nucl, h=40, B=100, method="u")

## End(Not run)
(h.nucl.u<-79.11877)
y.nucl.u<-loc_est(nuclear$xtab, nuclear$ytab, x.nucl,
  h=h.nucl.u, method="u")

# 2. improved version of the estimator
# Optimal bandwidths over 100 bootstrap replications
## Not run:
h.nucl.m <- loc_est_bw(nuclear$xtab, nuclear$ytab,
  x.nucl, h=40, B=100, method="m")

## End(Not run)
(h.nucl.m<-79.12)
y.nucl.m<-loc_est(nuclear$xtab, nuclear$ytab, x.nucl,
  h=h.nucl.m, method="m")

# 3. Representation
plot(x.nucl, y.nucl.u, lty=1, lwd=4, col="magenta", type="l")
lines(x.nucl, y.nucl.m, lty=2, lwd=4, col="cyan")
points(ytab~xtab, data=nuclear)
legend("topleft", legend=c("unconstrained", "improved"),
  col=c("magenta", "cyan"), lwd=4, lty=c(1,2))

```

---

loc\_est\_bw

*Bandwidth selection for the local linear frontier estimator*


---

**Description**

Computes the optimal bootstrap bandwidth proposed by Hall and Park (2004) for the local linear frontier estimator.

**Usage**

```

loc_est_bw(xtab, ytab, x, hini, B = 5, method = "u",
  fix.seed = FALSE, control = list("tm_limit" = 700))

```

**Arguments**

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
hini	the initial bandwidth at which the local linear estimate will be computed.
B	number of bootstrap replications.
method	a character equal to "u" (unconstrained estimator) or "m" (improved version of the unconstrained estimator).
fix.seed	a boolean equal to TRUE for fixing the seed (bootstrap sampling).
control	a list of parameters to the GLPK solver. See *Details* of help(Rglpk_solve_LP).

**Details**

For a detailed description of the bootstrap procedure, see Hall and Park (2004).

**Value**

Returns the optimal bootstrap bandwidth.

**Note**

The computational burden here is very demanding, so be forewarned.

**Author(s)**

Hohsuk Noh.

**References**

Hall, P. and Park, B.U. (2004). Bandwidth choice for local polynomial estimation of smooth boundaries. *Journal of Multivariate Analysis*, 91, 240-261.

**See Also**

[loc\\_est](#).

**Examples**

```
## Not run:
data("nuclear")
x.nucl <- seq(min(nuclear$xtab), max(nuclear$xtab),
  length.out = 101)
# 1. Unconstrained case
# Optimal bandwidths over 100 bootstrap replications
system.time(
h.nucl.u <- loc_est_bw(nuclear$xtab, nuclear$ytab,
  x.nucl, hini = 40, B = 1, method = "u")
```

```

)
# result is 79.11877

# 2. Monotonicity constraint
# Optimal bandwidths over 100 bootstrap replications
h.nucl.m <- loc_est_bw(nuclear$xtab, nuclear$ytab,
  x.nucl, hini = 40, B = 100, method = "m")
# result is 79.12

## End(Not run)

```

---

loc\_max

*Local maximum frontier estimators*


---

### Description

Computes the local constant and local DEA boundary estimates proposed by Gijbels and Peng (2000).

### Usage

```
loc_max(xtab, ytab, x, h, type="one-stage")
```

### Arguments

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
h	determines the bandwidth at which the estimate will be computed.
type	a character equal to "one-stage" or "two-stage".

### Details

When estimating  $\varphi(x)$ , for a given point  $x \in \mathbb{R}$ , the methodology of Gijbels and Peng consists of considering a strip around  $x$  of width  $2h$ , where  $h = h_n \rightarrow 0$  with  $nh_n \rightarrow \infty$  as  $n \rightarrow \infty$ , and focusing then on the  $y_i$  observations falling into this strip. More precisely, they consider the transformed variables  $z_i^{xh} = y_i \mathbf{1}_{(|x_i - x| \leq h)}$ ,  $i = 1, \dots, n$ , and the corresponding order statistics  $z_{(1)}^{xh} \leq \dots \leq z_{(n)}^{xh}$ .

The simple maximum  $z_{(n)}^{xh} = \max_{i=1, \dots, n} z_i^{xh}$  defines then the local constant estimator of the frontier point  $\varphi(x)$  [option type="one-stage"]. This opens a way to a two-stage estimation procedure as follows. In a first stage, Gijbels and Peng calculate the maximum  $z_{(n)}^{xh}$ . Then, they suggest to replace each observation  $y_i$  in the strip of width  $2h$  around  $x$  by this maximum, leaving all observations outside the strip unchanged. More precisely, they define  $\tilde{y}_i = y_i$  if  $|x_i - x| > h$  and  $\tilde{y}_i = z_{(n)}^{xh}$  if  $|x_i - x| \leq h$  either. Then, they apply the DEA estimator (see the function `dea_est`) to these transformed data  $(x_i, \tilde{y}_i)$ , giving the local DEA estimator (option type="two-stage"). An *ad hoc* way of selecting  $h$  is by using for instance the function `npcdistbw` from the **np** package (see Daouia et al. (2016) for details).

**Value**

Returns a numeric vector with the same length as *x*.

**Author(s)**

Abdelaati Daouia and Thibault Laurent.

**References**

Daouia, A., Laurent, T. and Noh, H. (2017). npbr: A Package for Nonparametric Boundary Regression in R. *Journal of Statistical Software*, **79**(9), 1-43. doi:10.18637/jss.v079.i09.

Gijbels, I. and Peng, L. (2000). Estimation of a support curve via order statistics, *Extremes*, **3**, 251–277.

**See Also**

[dea\\_est](#)

**Examples**

```
data("green")
x.green <- seq(min(log(green$COST)), max(log(green$COST)),
  length.out=101)
# Local maximum frontier estimates
# a. Local constant estimator
loc_max_1stage<-loc_max(log(green$COST), log(green$OUTPUT),
  x.green, h=0.5, type="one-stage")
# b. Local DEA estimator
loc_max_2stage<-loc_max(log(green$COST), log(green$OUTPUT),
  x.green, h=0.5, type="two-stage")
# Representation
plot(log(OUTPUT)~log(COST), data=green)
lines(x.green, loc_max_1stage, lty=1, col="magenta")
lines(x.green, loc_max_2stage, lty=2, col="cyan")
legend("topleft", legend=c("one-stage", "two-stage"),
  col=c("magenta", "cyan"), lty=c(1,2))
```

---

mopt\_pwm

*Threshold selection for the PWM frontier estimator*


---

**Description**

This function implements the optimal smoothing parameter *coefm* involved in the probability-weighted moment frontier estimator of Daouia, Florens and Simar (2012).

**Usage**

```
mopt_pwm(xtab, ytab, x, a=2, rho, wind.coef=0.1)
```

**Arguments**

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
a	a smoothing parameter (integer) larger than or equal to 2 (2 by default).
rho	a numeric vector of the same length as x or a scalar, which determines the values of rho.
wind.coef	a scalar coefficient to be selected in the interval (0,1].

**Details**

This is an implementation of an automated selection of the parameter `coefm` involved in the probability-weighted moment (PWM) estimator  $\hat{\varphi}_{pwm}(x)$  [see `dfs_pwm`]. It is an adaptation of the experimental method `kopt_momt_pick` by Daouia et al. (2010). The idea is to select first (for each  $x$ ) a grid of values for the parameter `coefm` given by  $c = 1, \dots, \min(10, \lfloor \sqrt{N_x} \rfloor)$ , where  $N_x = \sum_{i=1}^n 1_{\{x_i \leq x\}}$ , and then select the  $c$  where the variation of the results is the smallest. To achieve this, we compute the standard deviations of  $\hat{\varphi}_{pwm}(x)$  over a “window” of size  $wind.coef \times \min(10, \lfloor \sqrt{N_x} \rfloor)$ , where the coefficient `wind.coef` should be selected in the interval  $(0, 1]$  in such a way to avoid numerical instabilities. The default option `wind.coef=0.1` corresponds to having a window large enough to cover around 10% of the possible values of  $c$  in the selected range of values for `coefm`. The value of  $c$  where the standard deviation is minimal defines the desired `coefm`.

**Value**

Returns a numeric vector with the same length as `x`.

**Author(s)**

Abdelaati Daouia and Thibault Laurent.

**References**

Daouia, A., Florens, J.-P. and Simar, L. (2010). Frontier estimation and extreme value theory. *Bernoulli*, 16, 1039-1063.

**See Also**

`dfs_pwm`, `kopt_momt_pick`.

**Examples**

```
data("post")
x.post<- seq(post$input[100],max(post$input),
  length.out=100)
## Not run:
# When rho[x] is known and equal to 2:
best_cm.1<- mopt_pwm(post$input, post$yprod,
```

```
x.post, a=2, rho=2)
## End(Not run)
```

---

nuclear

*Reliability programs of nuclear reactors*

---

### Description

The dataset from the US Electric Power Research Institute (EPRI) consists of 254 toughness results obtained from non-irradiated representative steels. For each steel  $i$ , fracture toughness  $y_i$  and temperature  $x_i$  were measured.

### Usage

```
data(nuclear)
```

### Format

A data frame with 254 observations on the following 2 variables.

xtab Temperature.

ytab Fracture toughness of each material.

### Source

US Electric Power Research Institute (EPRI).

### References

Daouia, A., Girard, S. and Guillou, A. (2014). A Gamma-moment approach to monotonic boundary estimation. *Journal of Econometrics*, 78, 727-740.

### Examples

```
data("nuclear")
```

pick\_est

*Local Pickands' frontier estimator***Description**

Computes the Pickands type of estimator introduced by Gijbels and Peng (2000).

**Usage**

```
pick_est(xtab, ytab, x, h, k, type="one-stage")
```

**Arguments**

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
h	determines the bandwidth at which the estimate will be computed.
k	a numeric vector of the same length as x, which determines the thresholds at which the Pickands' estimator will be computed.
type	a character equal to "one-stage" or "two-stage".

**Details**

The local Pickands' frontier estimator (option `type="one-stage"`), obtained by applying the well-known approach of Dekkers and de Haan (1989) in conjunction with the transformed sample of  $z_i^{xh}$ 's described in the function `loc_max`, is defined as

$$z_{(n-k)}^{xh} + \left( z_{(n-k)}^{xh} - z_{(n-2k)}^{xh} \right) \left\{ 2^{-\log \frac{z_{(n-k)}^{xh} - z_{(n-2k)}^{xh}}{z_{(n-2k)}^{xh} - z_{(n-4k)}^{xh}} / \log 2} - 1 \right\}^{-1}.$$

It is based on three upper order statistics  $z_{(n-k)}^{xh}$ ,  $z_{(n-2k)}^{xh}$ ,  $z_{(n-4k)}^{xh}$ , and depends on  $h$  (see `loc_max`) as well as an intermediate sequence  $k = k(x, n) \rightarrow \infty$  with  $k/n \rightarrow 0$  as  $n \rightarrow \infty$ . The two smoothing parameters  $h$  and  $k$  have to be fixed in the 4th and 5th arguments of the function.

Also, the user can replace each observation  $y_i$  in the strip of width  $2h$  around  $x$  by the resulting local Pickands', leaving all observations outside the strip unchanged. Then, one may apply the DEA estimator (see the function `dea_est`) to the obtained transformed data, giving the local DEA estimator (option `type="two-stage"`).

**Value**

Returns a numeric vector with the same length as  $x$ .

**Author(s)**

Abdelaati Daouia and Thibault Laurent.

## References

Dekkers, A.L.M. and L. de Haan (1989). On the estimation of extreme-value index and large quantiles estimation, *Annals of Statistics*, 17, 1795-1832.

Gijbels, I. and Peng, L. (2000). Estimation of a support curve via order statistics, *Extremes*, 3, 251-277.

## See Also

[dea\\_est](#)

## Examples

```
## Not run:
data("green")
plot(log(OUTPUT)~log(COST), data=green)
x <- seq(min(log(green$COST)), max(log(green$COST)), length.out=101)
h=0.5
nx<-unlist(lapply(x,function(y) length(which(abs(log(green$COST)-y)<=h))))
k<-trunc(nx^0.1)
lines(x, pick_est(log(green$COST), log(green$OUTPUT), x, h=h, k=k), lty=1, col="red")

## End(Not run)
```

---

poly_degree	<i>AIC and BIC criteria for choosing the optimal degree of the polynomial frontier estimator</i>
-------------	--

---

## Description

Computes the optimal degree of the unconstrained polynomial frontier estimator proposed by Hall, Park and Stern (1998).

## Usage

```
poly_degree(xtab, ytab, prange=0:20, type="AIC",
  control = list("tm_limit" = 700))
```

## Arguments

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
prange	a vector of integers specifying the range in which the optimal degree of the polynomial frontier estimator is to be selected.
type	a character equal to "AIC" or "BIC".
control	a list of parameters to the GLPK solver. See *Details* of help(Rglpk_solve_LP).

## Details

As the degree  $p$  of the polynomial estimator  $\hat{\varphi}_{n,p}$  (see [poly\\_est](#)) determines the dimensionality of the approximating function, we may view the problem of choosing  $p$  as model selection. By analogy to the information criteria proposed by Daouia et al. (2016) in the boundary regression context, we obtain the optimal polynomial degree by minimizing

$$AIC(p) = \log \left( \sum_{i=1}^n (\hat{\varphi}_{n,p}(x_i) - y_i) \right) + (p+1)/n,$$

$$BIC(p) = \log \left( \sum_{i=1}^n (\hat{\varphi}_{n,p}(x_i) - y_i) \right) + \log n(p+1)/(2n).$$

The first one (option type = "AIC") is similar to the famous Akaike information criterion Akaike (1973) and the second one (option type = "BIC") to the Bayesian information criterion Schwartz (1978).

## Value

Returns an integer.

## Author(s)

Hohsuk Noh.

## References

- Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle, in *Second International Symposium of Information Theory*, eds. B. N. Petrov and F. Csaki, Budapest: Akademia Kiado, 267–281.
- Daouia, A., Noh, H. and Park, B.U. (2016). Data Envelope fitting with constrained polynomial splines. *Journal of the Royal Statistical Society: Series B*, **78**(1), 3-30. doi:10.1111/rssb.12098.
- Hall, P., Park, B.U. and Stern, S.E. (1998). On polynomial estimators of frontiers and boundaries. *Journal of Multivariate Analysis*, **66**, 71-98.
- Schwartz, G. (1978). Estimating the dimension of a model, *Annals of Statistics*, **6**, 461–464.

## See Also

[poly\\_est](#)

## Examples

```
data("air")
x.air <- seq(min(air$xtab), max(air$xtab),
  length.out = 101)
# Optimal polynomial degrees via the AIC criterion
(p.aic.air <- poly_degree(air$xtab, air$ytab,
  type = "AIC"))
# Optimal polynomial degrees via the BIC criterion
(p.bic.air <- poly_degree(air$xtab, air$ytab,
  type = "BIC"))
```

---

poly\_est                      *Polynomial frontier estimators*

---

**Description**

Computes the polynomial-type estimators of frontiers and boundaries proposed by Hall, Park and Stern (1998).

**Usage**

```
poly_est(xtab, ytab, x, deg, control = list("tm_limit" = 700))
```

**Arguments**

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
deg	an integer (polynomial degree).
control	a list of parameters to the GLPK solver. See *Details* of help(Rglpk_solve_LP).

**Details**

The data edge is modeled by a single polynomial  $\varphi_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_p x^p$  of known degree  $p$  that envelopes the full data and minimizes the area under its graph for  $x \in [a, b]$ , with  $a$  and  $b$  being respectively the lower and upper endpoints of the design points  $x_1, \dots, x_n$ . The implemented function is the estimate  $\hat{\varphi}_{n,p}(x) = \hat{\theta}_0 + \hat{\theta}_1 x + \dots + \hat{\theta}_p x^p$  of  $\varphi(x)$ , where  $\hat{\theta} = (\hat{\theta}_0, \hat{\theta}_1, \dots, \hat{\theta}_p)^T$  minimizes  $\int_a^b \varphi_\theta(x) dx$  over  $\theta \in \mathbb{R}^{p+1}$  subject to the envelopment constraints  $\varphi_\theta(x_i) \geq y_i$ ,  $i = 1, \dots, n$ .

**Value**

Returns a numeric vector with the same length as  $x$ . Returns a vector of NA if no solution has been found by the solver (GLPK).

**Author(s)**

Hohsuk Noh.

**References**

Hall, P., Park, B.U. and Stern, S.E. (1998). On polynomial estimators of frontiers and boundaries. *Journal of Multivariate Analysis*, 66, 71-98.

**See Also**

[loc\\_est](#)

**Examples**

```

data("air")
x.air <- seq(min(air$xtab), max(air$xtab),
  length.out = 101)
# Optimal polynomial degrees via the AIC criterion
(p.aic.air <- poly_degree(air$xtab, air$ytab,
  type = "AIC"))
# Polynomial boundaries estimate
y.poly.air <- poly_est(air$xtab, air$ytab, x.air,
  deg = p.aic.air)
# Representation
plot(x.air, y.poly.air, lty = 1, lwd = 4,
  col = "magenta", type = "l")
points(ytab~xtab, data = air)
legend("topleft", legend = paste("degree =", p.aic.air),
  col = "magenta", lwd = 4, lty = 1)

```

---

 post

*European postal services*


---

**Description**

The dataset post about the cost of the delivery activity of the postal services in Europe was first analyzed by Cazals, Florens and Simar (2002). There are 4,000 post offices observed in 1994. For each post office  $i$ , the input  $x_i$  is the labor cost measured by the quantity of labor, which represents more than 80% of the total cost of the delivery activity. The output  $y_i$  is defined as the volume of delivered mail (in number of objects). It should be noted that noise has been added to the original data.

**Usage**

```
data(post)
```

**Format**

A data frame with 4000 observations on the following 2 variables.

xinput a numeric vector.

yprod a numeric vector.

**References**

Cazals, C., Florens, J.-P., Simar, L. (2002), Nonparametric frontier estimation: a robust approach, *Journal of Econometrics*, 106, 1-25.

**Examples**

```
data("post")
```

---

quad\_spline\_est      *Quadratic spline frontiers*


---

### Description

This function is an implementation of the (un)constrained quadratic spline smoother proposed by Daouia, Noh and Park (2016).

### Usage

```
quad_spline_est(xtab, ytab, x, kn = ceiling((length(xtab))^(1/4)), method= "u",
  all.dea = FALSE, control = list("tm_limit" = 700))
```

### Arguments

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
kn	an integer specifying the number of inter-knot segments used in the computation of the spline estimate.
method	a character equal to "u" (unconstrained estimator), "m" (under the monotonicity constraint) or "mc" (under simultaneous monotonicity and concavity constraints).
all.dea	a boolean.
control	a list of parameters to the GLPK solver. See *Details* of help(Rglpk_solve_LP).

### Details

Let  $a$  and  $b$  be, respectively, the minimum and maximum of the design points  $x_1, \dots, x_n$ . Denote a partition of  $[a, b]$  by  $a = t_0 < t_1 < \dots < t_{k_n} = b$  (see below the selection process). Let  $N = k_n + 1$  and  $\pi(x) = (\pi_1(x), \dots, \pi_N(x))^T$  be the vector of normalized B-splines of order 3 based on the knot mesh  $\{t_j\}$  (see, e.g., Schumaker (2007)). When the true frontier  $\varphi(x)$  is known or required to be monotone nondecreasing (option `cv=0`), its constrained quadratic spline estimate is defined by  $\hat{\varphi}_n(x) = \pi(x)^T \hat{\alpha}$ , where  $\hat{\alpha}$  minimizes

$$\int_0^1 \pi(x)^T \alpha \, dx = \sum_{j=1}^N \alpha_j \int_0^1 \pi_j(x) \, dx$$

over  $\alpha \in \mathbb{R}^N$  subject to the envelopment and monotonicity constraints  $\pi(x_i)^T \alpha \geq y_i, i = 1, \dots, n$ , and  $\pi'(t_j)^T \alpha \geq 0, j = 0, 1, \dots, k_n$ , with  $\pi'$  being the derivative of  $\pi$ .

Considering the special connection of the spline smoother  $\hat{\varphi}_n$  with the traditional FDH frontier  $\varphi_n$  (see the function `dea_est`), Daouia et al. (2015) propose an easy way of choosing the knot mesh. Let  $(\mathcal{X}_1, \mathcal{Y}_1), \dots, (\mathcal{X}_N, \mathcal{Y}_N)$  be the observations  $(x_i, y_i)$  lying on the FDH boundary (i.e.  $y_i = \varphi_n(x_i)$ ). The basic idea is to pick out a set of knots equally spaced in percentile ranks among

the  $\mathcal{N}$  FDH points  $(\mathcal{X}_\ell, \mathcal{Y}_\ell)$  by taking  $t_j = \mathcal{X}_{[j\mathcal{N}/k_n]}$ , the  $j/k_n$ th quantile of the values of  $\mathcal{X}_\ell$  for  $j = 1, \dots, k_n - 1$ . The choice of the number of internal knots is then viewed as model selection through the minimization of the AIC and BIC information criteria (see the function `quad_spline_kn`).

When the monotone boundary  $\varphi(x)$  is also believed to be concave (option `cv=1`), its constrained fit is defined as  $\hat{\varphi}_n^*(x) = \pi(x)^T \hat{\alpha}^*$ , where  $\hat{\alpha}^* \in \mathbb{R}^N$  minimizes the same objective function as  $\hat{\alpha}$  subject to the same envelopment and monotonicity constraints and the additional concavity constraints  $\pi''(t_j^*)^T \alpha \leq 0$ ,  $j = 1, \dots, k_n$ , where  $\pi''$  is the constant second derivative of  $\pi$  on each inter-knot interval and  $t_j^*$  is the midpoint of  $(t_{j-1}, t_j]$ .

Regarding the choice of knots, the same scheme as for  $\hat{\varphi}_n$  can be applied by replacing the FDH points  $(\mathcal{X}_1, \mathcal{Y}_1), \dots, (\mathcal{X}_\mathcal{N}, \mathcal{Y}_\mathcal{N})$  with the DEA points  $(\mathcal{X}_1^*, \mathcal{Y}_1^*), \dots, (\mathcal{X}_\mathcal{M}^*, \mathcal{Y}_\mathcal{M}^*)$ , that is, the observations  $(x_i, y_i)$  lying on the piecewise linear DEA frontier (see the function `dea_est`). Alternatively, the strategy of just using all the DEA points as knots is also working quite well for datasets of modest size as shown in Daouia et al. (2016). In this case, the user has to choose the option `all.dea=TRUE`.

### Value

Returns a numeric vector with the same length as `x`. Returns a vector of NA if no solution has been found by the solver (GLPK).

### Author(s)

Hohsuk Noh.

### References

- Daouia, A., Noh, H. and Park, B.U. (2016). Data Envelope fitting with constrained polynomial splines. *Journal of the Royal Statistical Society: Series B*, **78**(1), 3-30. doi:10.1111/rssb.12098.
- Schumaker, L.L. (2007). *Spline Functions: Basic Theory*, 3rd edition, Cambridge University Press.

### See Also

[quad\\_spline\\_kn](#)

### Examples

```
## Not run:
data("green")
x.green <- seq(min(log(green$COST)), max(log(green$COST)), length.out=101)
# 1. Unconstrained quadratic spline fits
# Optimal number of inter-knot segments via the BIC criterion
(kn.bic.green.u<-quad_spline_kn(log(green$COST),
  log(green$OUTPUT), method="u", type="BIC"))
# Unconstrained spline estimate
y.quad.green.u<-quad_spline_est(log(green$COST),
  log(green$OUTPUT), x.green, kn=kn.bic.green.u, method="u")

# 2. Monotonicity constraint
# Optimal number of inter-knot segments via the BIC criterion
(kn.bic.green.m<-quad_spline_kn(log(green$COST),
```

```

log(green$OUTPUT), method="m", type="BIC"))
# Monotonic splines estimate
y.quad.green.m<-quad_spline_est(log(green$COST),
log(green$OUTPUT), x.green, kn=kn.bic.green.m, method="m")

# 3. Monotonicity and Concavity constraints
# Optimal number of inter-knot segments via the BIC criterion
(kn.bic.green.mc<-quad_spline_kn(log(green$COST),
log(green$OUTPUT), method="mc", type="BIC"))
# Monotonic/Concave splines estimate
y.quad.green.mc<-quad_spline_est(log(green$COST),
log(green$OUTPUT), x.green, kn=kn.bic.green.mc,
method="mc", all.dea=TRUE)

# Representation
plot(x.green, y.quad.green.u, lty=1, lwd=4, col="green",
type="l", xlab="log(COST)", ylab="log(OUTPUT)")
lines(x.green, y.quad.green.m, lty=2, lwd=4, col="cyan")
lines(x.green, y.quad.green.mc, lwd=4, lty=3, col="magenta")
points(log(OUTPUT)~log(COST), data=green)
legend("topleft", col=c("green", "cyan", "magenta"),
lty=c(1,2,3), legend=c("unconstrained", "monotone",
"monotone + concave"), lwd=4, cex=0.8)

## End(Not run)

```

---

quad_spline_kn	<i>AIC and BIC criteria for choosing the optimal number of inter-knot segments in quadratic spline fits</i>
----------------	---

---

## Description

Computes the optimal number  $k_n$  of inter-knot segments in the quadratic spline fits proposed by Daouia, Noh and Park (2016).

## Usage

```
quad_spline_kn(xtab, ytab, method, krange = 1:20, type = "AIC",
control = list("tm_limit" = 700))
```

## Arguments

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
method	a character equal to "u" (unconstrained estimator), "m" (under the monotonicity constraint) or "mc" (under simultaneous monotonicity and concavity constraints).

krange	a vector of integers specifying the range in which the optimal number of inter-knot segments is to be selected.
type	a character equal to "AIC" or "BIC".
control	a list of parameters to the GLPK solver. See *Details* of help(Rglpk_solve_LP).

### Details

For the implementation of the unconstrained quadratic spline smoother  $\tilde{\varphi}_n$  (see [quad\\_spline\\_est](#)), based on the knot mesh  $\{t_j = x_{[jn/k_n]} : j = 1, \dots, k_n - 1\}$ , the user has to employ the option `method="u"`. Since the number  $k_n$  determines the complexity of the spline approximation, its choice may be viewed as model selection via the minimization of the following Akaike (option `type="AIC"`) or Bayesian (option `type="BIC"`) information criteria:

$$A\tilde{I}C(k) = \log \left( \sum_{i=1}^n (\tilde{\varphi}_n(x_i) - y_i) \right) + (k + 2)/n,$$

$$B\tilde{I}C(k) = \log \left( \sum_{i=1}^n (\tilde{\varphi}_n(x_i) - y_i) \right) + \log n \cdot (k + 2)/2n.$$

For the implementation of the monotone (option `method="m"`) quadratic spline smoother  $\hat{\varphi}_n$  (see [quad\\_spline\\_est](#)), the authors first suggest using the set of knots  $\{t_j = \mathcal{X}_{[j\mathcal{N}/k_n]}, j = 1, \dots, k_n - 1\}$  among the FDH points  $(\mathcal{X}_\ell, \mathcal{Y}_\ell)$ ,  $\ell = 1, \dots, \mathcal{N}$  (function [quad\\_spline\\_est](#)). Then, they propose to choose  $k_n$  by minimizing the following AIC (option `type="AIC"`) or BIC (option `type="BIC"`) information criteria:

$$A\hat{I}C(k) = \log \left( \sum_{i=1}^n (\hat{\varphi}_n(x_i) - y_i) \right) + (k + 2)/n,$$

$$B\hat{I}C(k) = \log \left( \sum_{i=1}^n (\hat{\varphi}_n(x_i) - y_i) \right) + \log n \cdot (k + 2)/2n.$$

A small number of knots is typically needed as elucidated by the asymptotic theory.

For the implementation of the monotone and concave (option `method="mc"`) spline estimator  $\hat{\varphi}_n^*$ , just apply the same scheme as above by replacing the FDH points  $(\mathcal{X}_\ell, \mathcal{Y}_\ell)$  with the DEA points  $(\mathcal{X}_\ell^*, \mathcal{Y}_\ell^*)$  (see [dea\\_est](#)).

### Value

Returns an integer.

### Author(s)

Hohsuk Noh.

## References

- Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle, in *Second International Symposium of Information Theory*, eds. B. N. Petrov and F. Csaki, Budapest: Akademia Kiado, 267–281.
- Daouia, A., Noh, H. and Park, B.U. (2016). Data Envelope fitting with constrained polynomial splines. *Journal of the Royal Statistical Society: Series B*, **78**(1), 3-30. doi:10.1111/rssb.12098.
- Schwartz, G. (1978). Estimating the dimension of a model, *Annals of Statistics*, 6, 461–464.

## See Also

[quad\\_spline\\_est](#)

## Examples

```
data("green")
## Not run:
# BIC criteria for choosing the optimal number of
# inter-knot segments in:
# a. Unconstrained quadratic spline fits
(kn.bic.green.u <- quad_spline_kn(log(green$COST),
  log(green$OUTPUT), method = "u", type = "BIC"))
# b. Monotone quadratic spline smoother
(kn.bic.green.m <- quad_spline_kn(log(green$COST),
  log(green$OUTPUT), method = "m", type = "BIC"))
# c. Monotone and concave quadratic spline smoother
(kn.bic.green.mc<-quad_spline_kn(log(green$COST),
  log(green$OUTPUT), method = "mc", type = "BIC"))

## End(Not run)
```

---

records

*Annual sport records*

---

## Description

The dataset records is concerned with the yearly best men’s outdoor 1500m times starting from 1966. Following Jirak, Meister and Reiss (2014), the lower boundary can be interpreted as the best possible time for a given year. This boundary is not believed to be shape constrained and can be estimated by any unconstrained shape nonparametric method.

## Usage

```
data(records)
```

## Format

A data frame with 46 observations on the following 2 variables.

year year.

result 1500m record in seconds.

## References

Jirak, M., Meister, A. and M. Reiss (2014), Optimal adaptive estimation in nonparametric regression with one-sided errors. *Annals of Statistics*, 42, 1970–2002.

## Examples

```
data("records")
```

---

rho_momt_pick	<i>Optimal rho for moment and Pickands frontier estimator</i>
---------------	---

---

## Description

This function gives the optimal rho involved in the moment and Pickands estimators of Daouia, Florens and Simar (2010).

## Usage

```
rho_momt_pick(xtab, ytab, x, method="moment", lrho=1, urho=Inf)
```

## Arguments

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
method	a character equal to "moment" or "pickands".
lrho	a scalar, minimum rho threshold value.
urho	a scalar, maximum rho threshold value.

## Details

This function computes the moment and Pickands estimates of the extreme-value index  $\rho_x$  involved in the frontier estimators  $\tilde{\varphi}_{momt}(x)$  [see [dfs\\_momt](#)] and  $\hat{\varphi}_{pick}(x)$  [see [dfs\\_pick](#)]. In case method="moment", the estimator of  $\rho_x$  defined as

$$\tilde{\rho}_x = - \left( M_n^{(1)} + 1 - \frac{1}{2} \left[ 1 - (M_n^{(1)})^2 / M_n^{(2)} \right]^{-1} \right)^{-1}$$

is based on the moments  $M_n^{(j)} = (1/k) \sum_{i=0}^{k-1} \left( \log z_{(n-i)}^x - \log z_{(n-k)}^x \right)^j$  for  $j = 1, 2$ , with  $z_{(1)}^x \leq \dots \leq z_{(n)}^x$  are the ascending order statistics corresponding to the transformed sample  $\{z_i^x := y_i \mathbf{1}_{\{x_i \leq x\}}, i = 1, \dots, n\}$ . In case method="pickands", the estimator of  $\rho_x$  is given by

$$\hat{\rho}_x = - \log 2 / \log \left\{ (z_{(n-k+1)}^x - z_{(n-2k+1)}^x) / (z_{(n-2k+1)}^x - z_{(n-4k+1)}^x) \right\}.$$

To select the threshold  $k = k_n(x)$  in  $\tilde{\rho}_x$  and  $\hat{\rho}_x$ , Daouia et al. (2010) have suggested to use the following data driven method for each  $x$ : They first select a grid of values for  $k = k_n(x)$ . For the Pickands estimator  $\hat{\rho}_x$ , they choose  $k_n(x) = \lfloor N_x/4 \rfloor - k + 1$ , where  $k$  is an integer varying between 1 and the integer part  $\lfloor N_x/4 \rfloor$  of  $N_x/4$ , with  $N_x = \sum_{i=1}^n 1_{\{x_i \leq x\}}$ . For the moment estimator  $\tilde{\rho}_x$ , they choose  $k_n(x) = N_x - k$ , where  $k$  is an integer varying between 1 and  $N_x - 1$ . Then, they evaluate the estimator  $\hat{\rho}_x(k)$  (respectively,  $\tilde{\rho}_x(k)$ ) and select the  $k$  where the variation of the results is the smallest. They achieve this by computing the standard deviation of  $\hat{\rho}_x(k)$  (respectively,  $\tilde{\rho}_x(k)$ ) over a “window” of  $\max(\lfloor \sqrt{N_x/4} \rfloor, 3)$  (respectively,  $\max(\lfloor \sqrt{N_x - 1} \rfloor, 3)$ ) successive values of  $k$ . The value of  $k$  where this standard deviation is minimal defines the value of  $k_n(x)$ . The user can also appreciably improve the estimation of  $\rho_x$  and  $\varphi(x)$  itself by tuning the choice of the lower limit (default option `lrho=1`) and upper limit (default option `urho=Inf`).

### Value

Returns a numeric vector with the same length as  $x$ .

### Note

In order to choose a reasonable estimate  $\tilde{\rho}_x = \tilde{\rho}_x(k)$  and  $\hat{\rho}_x = \hat{\rho}_x(k)$  of the extreme-value index  $\rho_x$ , for each fixed  $x$ , one can construct the plot of the estimator of interest, consisting of the points  $\{(k, \tilde{\rho}_x(k))\}_k$  or  $\{(k, \hat{\rho}_x(k))\}_k$ , and select a value of the estimate at which the obtained graph looks stable. This is this kind of idea which guides the proposed automatic data-driven rule for a chosen grid of values of  $x$ . The main difficulty with such a method is that the plots of  $\tilde{\rho}_x(k)$  or  $\hat{\rho}_x(k)$  as functions of  $k$ , for each  $x$ , may be so unstable that reasonable values of  $k$  [which would correspond to the true value of  $\rho_x$ ] may be hidden in the graphs. In results, the obtained extreme-value index estimator and the frontier estimator itself may exhibit considerable volatility as functions of  $x$ . The user can appreciably improve the estimation of  $\rho_x$  and  $\varphi(x)$  by tuning the choice of the lower limit (default option `lrho=1`) and upper limit (default option `urho=Inf`).

### Author(s)

Abdelaati Daouia and Thibault Laurent (codes converted from Matlab’s Leopold Simar code).

### References

- Daouia, A., Florens, J.P. and Simar, L. (2010). Frontier Estimation and Extreme Value Theory, *Bernoulli*, 16, 1039-1063.
- Dekkers, A.L.M., Einmahl, J.H.J. and L. de Haan (1989), A moment estimator for the index of an extreme-value distribution, *The Annals of Statistics*, 17(4), 1833-1855.

### See Also

[dfs\\_momt](#), [dfs\\_pick](#)

### Examples

```
data("post")
x.post<- seq(post$xinput[100],max(post$xinput),
  length.out=100)
## Not run:
```

```
# a. Optimal rho for Pickands frontier estimator
rho_pick<-rho_momt_pick(post$input, post$yprod,
  x.post, method="pickands")
# b. Optimal rho for moment frontier estimator
rho_momt<-rho_momt_pick(post$input, post$yprod,
  x.post, method="moment")

## End(Not run)
```

rho\_pwm

*Probability-weighted moment frontier estimator***Description**

This function is an implementation of the Probability-weighted moment frontier estimator developed by Daouia, Florens and Simar (2012).

**Usage**

```
rho_pwm(xtab, ytab, x, a=2, lrho=1, urho=Inf)
```

**Arguments**

xtab	a numeric vector containing the observed inputs $x_1, \dots, x_n$ .
ytab	a numeric vector of the same length as xtab containing the observed outputs $y_1, \dots, y_n$ .
x	a numeric vector of evaluation points in which the estimator is to be computed.
a	a smoothing parameter (integer) larger than or equal to 2.
lrho	a scalar, minimum rho threshold value.
urho	a scalar, maximum rho threshold value.

**Details**

The function computes the probability-weighted moment (PWM) estimator  $\bar{\rho}_x$  utilized in the frontier estimate  $\hat{\varphi}_{pwm}(x)$  [see [dfs\\_pwm](#)]. This estimator depends on the smoothing parameters  $a$  and  $m$ . A simple selection rule of thumb that Daouia et al. (2012) have employed is  $a = 2$  [default option in the 4th argument of the function] and  $m = coefm \times N_x^{1/3}$ , where  $N_x = \sum_{i=1}^n 1_{\{x_i \leq x\}}$  and the integer  $coefm$  is to be tuned by the user. To choose this parameter in an optimal way for each  $x$ , we adapt the automated threshold selection method of Daouia et al. (2010) as follows: We first evaluate the estimator  $\bar{\rho}_x$  over a grid of values of  $coefm$  given by  $c = 1, \dots, 150$ . Then, we select the  $c$  where the variation of the results is the smallest. This is achieved by computing the standard deviation of the estimates  $\bar{\rho}_x$  over a “window” of  $\max([\sqrt{150}], 3)$  successive values of  $c$ . The value of  $c$  where this standard deviation is minimal defines the value of  $coefm$ . The user can also appreciably improve the estimation of the extreme-value index  $\rho_x$  and the frontier function  $\varphi_x$  itself by tuning the choice of the lower limit (default option  $lrho=1$ ) and upper limit (default option  $urho=Inf$ ).

**Value**

Returns a numeric vector with the same length as  $x$ .

**Note**

The computational burden here is demanding, so be forewarned.

**Author(s)**

Abdelaati Daouia and Thibault Laurent.

**References**

Daouia, A., Florens, J.-P. and Simar, L. (2010). Frontier estimation and extreme value theory. *Bernoulli*, 16, 1039-1063.

Daouia, A., Florens, J.-P. and Simar, L. (2012). Regularization of Nonparametric Frontier Estimators. *Journal of Econometrics*, 168, 285-299.

**See Also**

[dfs\\_pwm](#), [mopt\\_pwm](#).

**Examples**

```
data("post")
x.post<- seq(post$xinput[100],max(post$xinput),
  length.out=100)
## Not run:
# When rho[x] is unknown and dependent of x,
# its estimate hat(rho[x]) is obtained via:
rho_pwm <- rho_pwm(post$xinput, post$yprod, x.post, a=20)

## End(Not run)
```

# Index

## \* datasets

air, 5  
green, 19  
nuclear, 33  
post, 38  
records, 43

## \* nonparametric

cub\_spline\_est, 6  
cub\_spline\_kn, 8  
dea\_est, 10  
dfs\_momt, 12  
dfs\_pick, 14  
dfs\_pwm, 17  
kopt\_momt\_pick, 24  
loc\_max, 30  
mopt\_pwm, 31  
npbr-package, 2  
pick\_est, 34  
poly\_degree, 35  
quad\_spline\_est, 39  
quad\_spline\_kn, 41  
rho\_momt\_pick, 44  
rho\_pwm, 46

## \* optimize

cub\_spline\_est, 6  
cub\_spline\_kn, 8  
dea\_est, 10  
kern\_smooth, 20  
kern\_smooth\_bw, 22  
loc\_est, 26  
loc\_est\_bw, 28  
npbr-package, 2  
poly\_degree, 35  
poly\_est, 37  
quad\_spline\_est, 39  
quad\_spline\_kn, 41

air, 5

cub\_spline\_est, 4, 6, 9, 11

cub\_spline\_kn, 4, 6, 7, 8

dea\_est, 3, 10, 30, 31, 34, 35, 39, 40, 42  
dfs\_momt, 3, 12, 15, 16, 25, 26, 44, 45  
dfs\_pick, 4, 13, 14, 25, 26, 44, 45  
dfs\_pwm, 4, 17, 32, 46, 47

green, 19

kern\_smooth, 4, 20, 24  
kern\_smooth\_bw, 4, 21, 22  
kopt\_momt\_pick, 4, 13, 15, 16, 24, 32

loc\_est, 3, 26, 29, 37  
loc\_est\_bw, 3, 27, 28  
loc\_max, 4, 30, 34

mopt\_pwm, 18, 31, 47

npbr (npbr-package), 2  
npbr-package, 2  
nuclear, 33

pick\_est, 4, 34  
poly\_degree, 3, 35  
poly\_est, 3, 36, 37  
post, 38

quad\_spline\_est, 4, 11, 39, 42, 43  
quad\_spline\_kn, 4, 40, 41

records, 43  
rho\_momt\_pick, 4, 12, 15, 25, 44  
rho\_pwm, 18, 46