Package 'qdfauction'

November 10, 2026

Type Package

Title Nonparametric Quantile Density Estimation and Structural Analysis of First-Price Auctions

Version 0.3.0

Description Nonparametric estimators of the quantile density function (the derivative of the quantile function): kernel, boundary-corrected kernel, Poisson-weight and Bernstein-polynomial smoothers, indirect estimators (Jones (1992) <https://www.ism.ac.jp/editsec/aism/pdf/044_4_0721.pdf>; Soni, Dewan and Jain (2012) <doi:10.1016/j.csda.2012.04.014>; Chaubey, Dewan and Li (2024) <doi:10.1080/03610926.2023.2222922>), and linear, hard-thresholded and block-thresholded wavelet estimators (Chesneau, Dewan and Doosti (2016) <doi:10.1016/j.csda.2015.08.006>; Shirazi and Doosti (2022) <doi:10.1080/03610918.2019.1656250>), with smoothing parameters chosen by biased, robust likelihood (Wu (2019) <doi:10.1080/07350015.2018.1424633>) or least-squares cross-validation. The estimators are applied to the structural analysis of first-price sealed-bid auctions: recovery of private values for independent or affiliated values and risk-neutral or risk-averse bidders (Doosti, Dewan and Talebian (2025) <doi:10.1016/j.econlet.2025.112670>; Doosti (2026) <doi:10.1016/j.econlet.2026.112837>), estimation of the private-value density with kernel benchmarks (Hickman and Hubbard (2015) <doi:10.1002/jae.2385>; Marmer and Shneyerov (2012) <doi:10.1016/j.jeconom.2011.09.020>; Zincenko (2024) <doi:10.1016/j.jeconom.2024.105734>) and confidence bands (Ma, Marmer and Shneyerov (2019) <doi:10.1016/j.jeconom.2019.02.006>), and estimation of the seller's expected revenue and optimal reserve price, with tools for simulating equilibrium bids and comparing estimators.

License GPL (>= 3)

URL https://github.com/DoostiH/qdfauction

BugReports https://github.com/DoostiH/qdfauction/issues

Encoding UTF-8

Language en-GB

Depends R (>= 4.1)

Imports stats, splines, utils, graphics, grDevices

Suggests testthat (>= 3.1.7), knitr, rmarkdown, np

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

NeedsCompilation no

RoxygenNote 7.3.1

Author Hassan Doosti [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-0290-8122>, affiliation: School of Mathematical and Physical Sciences, Macquarie University),
Yogendra P. Chaubey [ctb, cph] (original R code of the quantile density estimators and cross-validation criteria, revised and vectorised in R/estimators.R and R/bandwidth.R),
Isha Dewan [ctb, cph] (original R code of the quantile density estimators and cross-validation criteria, revised and vectorised in R/estimators.R and R/bandwidth.R),
Jun Li [ctb, cph] (original R code of the quantile density estimators and cross-validation criteria, revised and vectorised in R/estimators.R and R/bandwidth.R),
Brent R. Hickman [ctb, cph] (original MATLAB implementation of the boundary-corrected kernel density estimator, translated to R in kde_hh()),
Timothy P. Hubbard [ctb, cph] (original MATLAB implementation of the boundary-corrected kernel density estimator, translated to R in kde_hh()),
Brani Vidakovic [ctb, cph] (original MATLAB routines Phijk, dwtr, Idwtr and loc_lin, translated to R in scaling_function(), phi_jk(), dwt_periodic(), idwt_periodic() and loclin_smooth()),
Vadim Marmer [ctb, cph] (original MATLAB implementation of the quantile-based estimator, translated to R in density_marmer_shneyerov(); reference implementation used to validate fpa_density_ci()),
Jun Ma [ctb] (reference implementation of inference for the GPV estimator, used to validate fpa_density_ci()),
Artyom Shneyerov [ctb, cph] (original MATLAB implementation of the quantile-based estimator, translated to R in density_marmer_shneyerov()),
Federico Zincenko [ctb, cph] (original implementation of the kernel estimator of private values with boundary extrapolation, adapted in fpa_values(method = 'zincenko'))

Maintainer Hassan Doosti <Hassan.doosti@mq.edu.au>

Repository CRAN

Date/Publication 2026-11-10 10:00:02 UTC

Contents

bcv_criterion . . . 4
bid_icc . . . 6
block_length_auto . . . 6
copula_fit . . . 7
copula_psi . . . 7
copula_psi_np . . . 8
copula_spec_test . . . 9
copula_tau_to_theta . . . 10
density_marmer_shneyerov . . . 11
dwt_periodic . . . 12
fpa_bid_function . . . 13
fpa_block_boot . . . 14
fpa_corner_test . . . 15
fpa_density . . . 17
fpa_density_boot . . . 19
fpa_density_ci . . . 21
fpa_density_cv . . . 23
fpa_evaluate_values . . . 24
fpa_montecarlo . . . 25
fpa_optimal_reserve . . . 26
fpa_revenue . . . 27
fpa_revenue_boot . . . 28
fpa_revenue_model . . . 29
fpa_revenue_qd . . . 31
fpa_simulate . . . 33
fpa_values . . . 34
fpa_virtual_value_model . . . 36
kde_hh . . . 38
loclin_smooth . . . 39
make_kernel . . . 39
phi_jk . . . 40
plot.fpa_density . . . 41
plot.fpa_density_ci . . . 41
plot.fpa_revenue_boot . . . 42
plot.fpa_values . . . 42
plot.qdf . . . 43
predict.qdf . . . 43
proc_check_winning_bid . . . 44
proc_cost . . . 44
proc_cost_boot . . . 45
proc_M . . . 46
proc_optimal_ceiling . . . 46
proc_simulate . . . 47
proc_values . . . 47
psi_jk . . . 48
qdf . . . 49

qdf_jones 51
qdf_kernel 53
qdf_poisson 54
qdf_wavelet 55
qdf_wavelet_nl 57
rcopula_archimedean 59
resample_blocks 60
select_bandwidth 60
select_fpa_density 62
select_wavelet 63
threshold_coefficients 64
timber 65
wavelet_filter 67

Index **68**

`bcv_criterion` *Cross-validation criteria for quantile density estimators*

Description

Three data-driven criteria for the smoothing parameter of a quantile density estimator, written as functions of $m = 1/h$ so that they can be passed to [stats::optimize()]. Each is evaluated on the grid $u_i = i/n$ with spacings $\Delta_i = X_{(i)} - X_{(i-1)}$, $X_{(0)} = 0$, so that $\sum_i \Delta_i \, \phi(\hat{q}(u_i))$ approximates $\int \phi\{1/\hat{g}(x)\}\, dx$.

* 'bcv_criterion()' is the biased (least-squares) cross-validation used by Chaubey, Dewan and Li (2024) and Doosti, Dewan and Talebian (2025):

$$\mathrm{BCV}(h) = \sum_i \Delta_i \, \hat{q}(u_i; h)^{-2} - \frac{2}{n} \sum_i \hat{q}_{-i}(u_i; h)^{-1},$$

i.e. the classical $\int \hat{g}^2 - 2\,\bar{\hat{g}}_{-i}$ written in terms of $\hat{q} = 1/\hat{g}$. (Equation (11) of Doosti, Dewan and Talebian (2025) prints $\hat{q}^2$ in the first term; the simulation code, reproduced here, uses $\hat{q}^{-2}$.) * 'rlcv_criterion()' is the robust likelihood cross-validation of Wu (2019), $-\{\frac{1}{n}\sum_i \ln^*(\hat{q}_{-i}(u_i)^{-1}) - b^*(h)\}$, with $\ln^*$ linearised below the threshold a_n. It is written so that it is *minimised*. * 'wbcv_criterion()' is the weighted criterion of Doosti (2026b), which targets $u\,q(u)$ (the quantity entering the first-price auction inversion) rather than $q(u)$:

$$\mathrm{WBCV}(h) = \sum_i \Delta_i \{u_i \hat{q}(u_i)\}^2 - \frac{2}{n} \sum_i u_i / \hat{q}_{-i}(u_i).$$

Not recommended; kept only for backward compatibility. The criterion is not scale-invariant: its two terms have different units (the first scales like the cube of the data units, the second like their inverse), so the selected bandwidth depends on the units of the data; on data of order one or larger the first term dominates and the criterion tends to select the smoothest fit in 'm_range'. The criterion is being revised (Doosti 2026b, revised version); 'wbcv_criterion()' and 'bandwidth = "wbcv"' issue a warning and are kept only so that existing code runs. Use BCV, RLCV or a numeric bandwidth instead.

Usage

```
bcv_criterion(m, x, fun, loo = TRUE, u_scheme = c("i/n", "i/(n+1)"))

rlcv_criterion(m, x, fun, loo = TRUE, u_scheme = c("i/n", "i/(n+1)"))

wbcv_criterion(m, x, fun, loo = TRUE, u_scheme = c("i/(n+1)", "i/n"))
```

Arguments

m	inverse bandwidth, $m = 1/h$.
x	numeric sample (sorted internally).
fun	an estimator with signature ‘fun(u, x, h)‘, e.g. [qdf_kernel()]. Extra arguments (such as ‘H‘ for [qdf_soni()]) should be bound with a closure before calling.
loo	logical; if ‘TRUE‘ (default) leave-one-out fits are computed exactly (by refitting without observation i, or by the index shift of [qdf_bernstein_loo()] for the Bernstein estimator). If ‘FALSE‘ the scaling approximation $\hat{q}_{-i} \approx \hat{q}\,(n-1)/n$ is used. The approximation is not valid for estimators built from order statistics: removing $X_{(i)}$ shifts every higher quantile position, and with it least-squares criteria select the roughest fit in ‘m_range‘. It is kept only for backward compatibility.
u_scheme	grid on which the criterion is evaluated: ‘"i/n"‘ (the published convention) or ‘"i/(n+1)"‘.

Value

a single numeric value; smaller is better.

References

Wu, X. (2019). Robust likelihood cross validation for kernel density estimation. *Journal of Business & Economic Statistics*, 37, 761–770.

See Also

[select_bandwidth()], [qdf()]

Examples

```
x <- rgamma(60, 5, 1)
bcv_criterion(20, x, qdf_kernel)
rlcv_criterion(20, x, qdf_kernel)
optimize(bcv_criterion, c(10, 50), x = x, fun = qdf_bernstein)$minimum
```

bid_icc *Between-auction share of residual bid variance*

Description

One-way ANOVA intraclass correlation of (homogenised, log) bids by auction. Large values indicate a component common to the bidders in an auction, i.e. within-auction dependence.

Usage

```
bid_icc(x, auction)
```

Arguments

x	Numeric vector, e.g. residuals of a log-bid regression.
auction	Auction identifier, same length as x.

Value

The ICC estimate.

block_length_auto *Automatic block length for the auction bootstrap*

Description

Automatic block length for the auction bootstrap

Usage

```
block_length_auto(B, mult = 2, warn_below = 15, winner = c("max", "min"))
```

Arguments

B	numeric matrix of bids, auctions in rows in time order, bidders in columns.
mult	multiplier on the Politis-White length; 2 is the value validated in the paper, results are insensitive over 1.5 to 3.
warn_below	warn when T / L_pw is below this value: the sample is then too short for the dependence range it contains and no block length recovers the full sampling variance (the paper's Table 6).
winner	"max" for a selling auction (highest bid wins), "min" for a procurement auction (lowest bid wins); the winning-bid series is built accordingly. Pass procurement bids in their original orientation with winner = "min"; the fitting itself is done by proc_values(), which reflects them.

Value

integer block length with attributes "L_pw" and "T_over_Lpw".

`copula_fit` *Pseudo-maximum-likelihood estimation of an Archimedean copula from bids*

Description

Transforms bids to pooled Hazen scores $\hat{u} = \mathrm{rank}/(N+1)$, forms all within-auction pairs, and maximises the bivariate copula log-likelihood over θ. Only pairs within the same auction are used (pairs straddling auctions are independent and would bias $\hat{\theta}$ toward independence).

Usage

```
copula_fit(bids, family = c("clayton", "gumbel", "frank"), bounds = NULL)
```

Arguments

`bids`	numeric matrix, one row per auction and one column per bidder.
`family`	‘"clayton"‘, ‘"gumbel"‘ or ‘"frank"‘.
`bounds`	optional search interval for θ.

Value

a list with ‘theta‘, ‘tau‘ (implied Kendall’s tau), ‘family‘, ‘loglik‘ and ‘n_pairs‘.

Examples

```
set.seed(1)
U <- rcopula_archimedean(100, 3, "clayton", theta = 2)
copula_fit(U, "clayton")
```

`copula_psi` *Copula multiplier and diagonal functions of exchangeable Archimedean copulas*

Description

For an exchangeable n-dimensional Archimedean copula C: * ‘copula_psi()‘ is the copula multiplier of Doosti (2026a), $\psi(u) = C_2(u,\dots,u)/c(u,\dots,u)$, which enters the inverse bid function $v = b + \psi(u)\,q(u)$. Under independence $\psi(u) = u/(n-1)$ and the GPV formula is recovered. * ‘copula_delta()‘ is the diagonal $\delta(u) = C(u,\dots,u)$, the distribution function of the maximum of the n uniform scores. * ‘copula_delta_prime()‘ is its derivative.

Usage

```
copula_psi(u, family = "independence", theta = NULL, n)

copula_delta(u, family = "independence", theta = NULL, n)

copula_delta_prime(u, family = "independence", theta = NULL, n)
```

Arguments

u	numeric vector in $(0, 1)$.
family	‘"independence"‘, ‘"clayton"‘, ‘"gumbel"‘ or ‘"frank"‘.
theta	copula parameter (ignored for independence).
n	number of bidders.

Value

numeric vector.

References

Doosti, H. (2026a). A copula-quantile density approach to affiliated private value auctions. *Economics Letters*, 261, 112837. doi:10.1016/j.econlet.2026.112837

Examples

```
u <- c(0.2, 0.5, 0.8)
copula_psi(u, "clayton", theta = 2, n = 3)
copula_psi(u, "independence", n = 3)      # u / (n - 1)
copula_delta(u, "gumbel", theta = 1.5, n = 3)
```

---

copula_psi_np *Nonparametric estimator of the copula multiplier*

Description

Estimates the copula multiplier $\psi(u)$ of the inverse bid function $v = b + \psi(u)\, q(u)$ without a parametric copula. In the rank scale, with U a bidder's rank and M the highest rank among its rivals in the same auction,

$$\psi(u) = \frac{P(M \le u \mid U = u)}{f_{M|U}(u \mid u)},$$

the ratio of the conditional distribution and density of the highest rival bid at the bidder's own bid, expressed in ranks (the quantity identified by Li, Perrigne and Vuong 2002). Both are estimated with beta kernels on $[0, 1]$ (Chen 1999; Chen and Fan 2006):

$$\hat{\psi}(u) = \frac{\sum_i K_b(\hat{U}_i; u)\, 1\{\hat{M}_i \le u\}}{\sum_i K_b(\hat{U}_i; u)\, K_b(\hat{M}_i; u)}, \qquad K_b(x; u) = \mathrm{Beta}(x;\, u/b + 1,\, (1-u)/b + 1).$$

Under independence $\psi(u) = u/(n-1)$. This is the flexible copula multiplier of the copula–quantile density estimator (Doosti 2026a); it is used by ‘fpa_values(family = "nonparametric")‘ and by [copula_spec_test()].

Usage

```
copula_psi_np(bids, u, b = NULL)
```

Arguments

bids	an ‘L‘ by ‘n‘ matrix or data frame of bids (one row per auction).
u	points in $(0, 1)$ at which to evaluate $\hat{\psi}$.
b	beta-kernel bandwidth; default $0.5\, N^{-1/3}$, N the number of bids.

Value

numeric vector $\hat{\psi}(u)$.

References

Chen, S. X. (1999). Beta kernel estimators for density functions. *Computational Statistics & Data Analysis*, 31, 131–145.

Chen, X. and Fan, Y. (2006). Estimation of copula-based semiparametric time series models. *Journal of Econometrics*, 130, 307–335.

Doosti, H. (2026a). A copula-quantile density approach to affiliated private value auctions. *Economics Letters*, 261, 112837. doi:10.1016/j.econlet.2026.112837

Examples

```
set.seed(1)
U <- rcopula_archimedean(300, 3, "clayton", theta = 2)
u <- c(0.2, 0.5, 0.8)
rbind(nonparametric = copula_psi_np(U, u),
      clayton       = copula_psi(u, "clayton", 2, 3),
      independence  = copula_psi(u, "independence", n = 3))
```

---

copula_spec_test *Specification test for a parametric copula of the private values*

Description

Tests whether an exchangeable Archimedean copula family is compatible with the bids, using $T = \sup_u |\hat{\psi}_{NP}(u) - \psi(u; \hat{\theta})|$ over a grid of interior quantiles (Doosti 2026a), where $\hat{\psi}_{NP}$ is [copula_psi_np()] and $\hat{\theta}$ is the pseudo-maximum-likelihood estimate of [copula_fit()]. Critical values come from a parametric bootstrap under the null: rank matrices are simulated from the fitted copula, the parameter is re-estimated and the statistic recomputed. Because both multipliers depend on the bids only through their ranks, the test does not involve the marginal distribution.

Usage

```
copula_spec_test(
  bids,
  family = c("clayton", "gumbel", "frank", "independence"),
  B = 199,
  grid = seq(0.1, 0.9, by = 0.02),
  b = NULL
)
```

Arguments

bids	an ‘L‘ by ‘n‘ matrix or data frame of bids.
family	‘"clayton"‘, ‘"gumbel"‘, ‘"frank"‘ or ‘"independence"‘.
B	number of bootstrap replications.
grid	interior quantiles over which the supremum is taken.
b	beta-kernel bandwidth, see [copula_psi_np()].

Value

an object of class ‘"htest"‘ with the statistic, the bootstrap p-value and the fitted parameter.

Reproducibility

Call [set.seed()] beforehand to make the bootstrap reproducible.

Examples

```
set.seed(1)
U <- rcopula_archimedean(150, 3, "gumbel", theta = 2)
copula_spec_test(U, "gumbel", B = 99)
copula_spec_test(U, "independence", B = 99)
```

---

copula_tau_to_theta *Convert between Kendall's tau and the copula parameter*

Description

Convert between Kendall's tau and the copula parameter

Usage

```
copula_tau_to_theta(tau, family = c("clayton", "gumbel", "frank"))

copula_theta_to_tau(theta, family = c("clayton", "gumbel", "frank"))
```

Arguments

tau, theta	scalar values.
family	'"clayton"', '"gumbel"' or '"frank"'.

Value

numeric scalar.

Examples

```
copula_tau_to_theta(0.5, "clayton")   # 2
copula_theta_to_tau(2, "gumbel")      # 0.5
```

---

density_marmer_shneyerov

Quantile-based density estimator of Marmer and Shneyerov (2012)

Description

Estimates the private-value density in a symmetric IPV first-price auction directly from bids: the bid quantile function and density are estimated with a triweight kernel, the preliminary value quantile $Q(\tau)+\tau/\{(n-1)\hat{g}(q(\tau))\}$ is monotonised around $\tau_0 = 0.5$, inverted to give $\hat{F}(v)$, and the density is

$$\hat{f}(v) = \left[\tfrac{n}{n-1}\hat{g}^{-1} - \tfrac{1}{n-1}\hat{F}(v)\,\hat{g}'/\hat{g}^{3}\right]^{-1}$$

evaluated at $q(\hat{F}(v))$. Bandwidths follow the authors' normal-reference rules ($1.06\,\hat{\sigma}\,n^{-1/5}$ for g, $1.06\,\hat{\sigma}\,n^{-1/7}$ for g'). This function is an R translation of the authors' MATLAB implementation, which is available from the first author's website (see 'inst/COPYRIGHTS' for the exact location).

Usage

```
density_marmer_shneyerov(x, bids, n, tau0 = 0.5, cap = 1000)
```

Arguments

x	evaluation points on the value scale.
bids	pooled bids.
n	number of bidders per auction.
tau0	reference quantile for the monotonisation.
cap	upper cap on the estimate (numerical safeguard).

Value

numeric vector of density estimates at 'x'.

References

Marmer, V. and Shneyerov, A. (2012). Quantile-based nonparametric inference for first-price auctions. *Journal of Econometrics*, 167(2), 345–357. doi:10.1016/j.jeconom.2011.09.020

`dwt_periodic` *Periodic discrete wavelet transform*

Description

Forward (‘dwt_periodic()‘) and inverse (‘idwt_periodic()‘) periodised orthonormal DWT with ‘L‘ levels, as in Vidakovic (1999). ‘length(x)‘ must be divisible by ‘2^L‘. Coefficients are returned as ‘c(smooth, detail_coarsest, ..., detail_finest)‘. Translated from the MATLAB routines ‘dwtr‘ and ‘Idwtr‘ of B. Vidakovic.

Usage

```
dwt_periodic(x, L, filter = "coif1")

idwt_periodic(w, L, filter = "coif1")
```

Arguments

`x`	numeric vector.
`L`	number of levels.
`filter`	see [wavelet_filter()].
`w`	wavelet coefficients from [dwt_periodic()].

Value

numeric vector of the same length as ‘x‘.

Examples

```
x <- rnorm(32)
all.equal(idwt_periodic(dwt_periodic(x, 3, "coif1"), 3, "coif1"), x)
```

`fpa_bid_function` *Equilibrium bid function of a symmetric first-price auction*

Description

Computes the symmetric equilibrium bid function in quantile space, $\beta(u) = Q(u) - m(u)$ with the markup solving $m'(u) = q(u) - m(u)/\psi(u)$, $m(u_0) = 0$ (Milgrom and Weber 1982; Doosti 2026a). Under independence $\psi(u) = u/(n-1)$ and this is the Guerre–Perrigne–Vuong bid function $\beta(v) = v - \int_0^v F(t)^{n-1}dt/F(v)^{n-1}$. A reserve price at quantile 'p0' sets $m(p_0) = 0$. For CRRA bidders with utility $U(t) = t^{1-\eta}$ the first-order condition is $v = b + (1-\eta)\psi(u)q(u)$ (Guerre, Perrigne and Vuong 2009), so the markup solves the same ODE with ψ replaced by $(1-\eta)\psi$; under independence this gives Zincenko's (2024) risk-averse bid function $\beta(v) = e_1 \int_0^v xF(x)^{e_2} f(x)\,dx/F(v)^{e_1}$, $e_1 = (n-1)/(1-\eta)$, $e_2 = (n+\eta-2)/(1-\eta)$.

Usage

```
fpa_bid_function(
  Q,
  q = NULL,
  n,
  family = "independence",
  theta = NULL,
  p0 = 0,
  eta = 0,
  U = fpa_ugrid()
)
```

Arguments

`Q`	quantile function of private values (a function of 'u').
`q`	quantile density of private values (a function of 'u'); if 'NULL' it is obtained by numerical differentiation of 'Q'.
`n`	number of bidders.
`family`	'"independence"', '"clayton"', '"gumbel"' or '"frank"'.
`theta`	copula parameter (ignored for independence).
`p0`	screening quantile of the reserve price (default 0, no reserve).
`eta`	CRRA risk-aversion parameter in $[0, 1)$ (0: risk neutral).
`U`	quantile grid; see [fpa_ugrid()].

Value

a list with 'U', 'bid' ($\beta(U)$), 'value' ($Q(U)$), 'markup' and a function 'beta(v)' interpolating the bid at value 'v'.

Examples

```
bf <- fpa_bid_function(Q = function(u) qgamma(u, 2, 10),
                       q = function(u) 1 / dgamma(qgamma(u, 2, 10), 2, 10), n = 5)
bf$beta(c(0.1, 0.2, 0.5))
# compare with the direct integral form for IPV
v <- 0.2; v - integrate(function(t) pgamma(t, 2, 10)^4, 0, v)$value / pgamma(v, 2, 10)^4
```

`fpa_block_boot` *Block bootstrap of a statistic of the recovered values*

Description

Block bootstrap of a statistic of the recovered values

Usage

```
fpa_block_boot(
  B,
  stat,
  L = "auto",
  B_boot = 199L,
  level = 0.9,
  method = "bernstein",
  bandwidth = "bcv",
  monotone = "spline",
  winner = "max",
  ...
)

fpa_revenue_boot_blocks(B, r, v_s = 0, L = "auto", ...)

fpa_quantile_boot_blocks(B, u = c(0.25, 0.5, 0.75), L = "auto", ...)
```

Arguments

`B`	bid matrix, auctions in rows in time order.
`stat`	function of an fpa_values() fit returning a numeric vector.
`L`	"auto" for block_length_auto(B), or an integer; 1 gives the i.i.d. auction bootstrap.
`B_boot`	number of bootstrap resamples.
`level`	nominal level of the percentile interval.
`method`, `bandwidth`, `monotone`	passed to fpa_values(); the bandwidth is selected once on the full sample and held fixed in every refit.
`winner`	which bid identifies the auction for the block-length rule: "max" for a sale auction, "min" for procurement.

...	further arguments to fpa_values().
r	reserve price(s); @param v_s seller valuation.
v_s	seller's valuation, passed on to the statistic where it applies.
u	probabilities at which the recovered value quantile is evaluated.

Value

list with point, boot_sd, lower, upper, L, L_pw, T_over_Lpw, boot (the B_boot x length(stat) matrix of bootstrap statistics).

`fpa_corner_test` *Test for a corner solution of the optimal reserve under affiliation*

Description

Decides whether any binding reserve price is optimal. The point rule classifies the sample as a corner (no binding reserve optimal) when

$$T = \min_{u \in [p_{lo},\, p_{hi}]} \hat{\omega}(u) - v_s \geq 0,$$

with $\hat{\omega}$ the estimated affiliated virtual value of [fpa_virtual_value()]. The window starts at ‘p_lo = 0.10‘ rather than at the boundary: with v_s equal to the lower end of the support, $\omega(u) \to v_s$ as $u \to 0$ for every copula, so the region near zero carries no information about the corner, while the quantile density of the pseudo-values is least accurately estimated there.

The decision is the point rule on the full sample (‘decision = "point"‘, the default and the rule used in Doosti 2026c): a corner is reported when $T \geq 0$. The null tested is "no reserve above the ‘p_lo‘ quantile beats no reserve". In applications the copula family should be chosen with [copula_spec_test()].

With ‘B > 0‘, auctions are resampled with replacement, with the bandwidths of both stages selected once on the full sample and held fixed in every replication, and the frequency with which the point rule reports a corner is returned. Two alternative uses of the bootstrap are available but are not the default: ‘decision = "bootstrap"‘ reports a corner when that frequency is at least ‘level‘; and ‘corner_not_rejected‘ records whether the upper $1 - \alpha$ bootstrap percentile of T is non-negative (a conservative rule, since T, a minimum of a noisy curve, is biased downward).

Usage

```
fpa_corner_test(
  bids,
  v_s = 0,
  family,
  method = "poisson",
  bandwidth = "rlcv",
  qdf_method = "poisson",
  p_lo = 0.1,
  p_hi = 0.98,
```

```
  B = 0,
  decision = c("point", "bootstrap"),
  level = 0.5,
  alpha = 0.1,
  ...
)

## S3 method for class 'fpa_corner_test'
print(x, ...)
```

Arguments

`bids`	an ‘L‘ by ‘n‘ matrix of bids, one row per auction.
`v_s`	seller’s valuation.
`family`	copula family used to recover the values (see [fpa_values()]); required, because the corner result depends on it.
`method, bandwidth`	first-stage quantile density estimator and bandwidth rule for [fpa_values()]; ‘bandwidth‘ is also used for the quantile density of the pseudo-values.
`qdf_method`	quantile density estimator for the pseudo-values.
`p_lo, p_hi`	window of quantiles over which the minimum is taken.
`B`	number of bootstrap replications (0: point rule only).
`decision`	‘"point"‘ (default): the point rule on the full sample; ‘"bootstrap"‘: a corner when the bootstrap frequency is at least ‘level‘ (requires ‘B > 0‘).
`level`	bootstrap corner frequency above which a corner is reported when ‘decision = "bootstrap"‘.
`alpha`	level for the upper bootstrap percentile of the statistic.
`...`	further arguments to [fpa_values()].
`x`	an ‘"fpa_corner_test"‘ object.

Value

An object of class ‘"fpa_corner_test"‘: a list with the ‘statistic‘ T, the decision ‘corner‘, the estimated virtual value ‘omega‘ on the grid ‘U‘, ‘v_s‘, ‘p_range‘, the ‘bandwidth‘s used in the two stages and the ‘decision‘ rule; with ‘B > 0‘ also ‘boot_stat‘, ‘boot_corner_freq‘, ‘boot_upper‘ and ‘corner_not_rejected‘. The ‘print()‘ method is called for its side effect of printing the decision and returns ‘x‘ invisibly.

Reproducibility

Call ‘set.seed()‘ beforehand to make the bootstrap reproducible.

References

Doosti, H. (2026c). Optimal reserve prices in affiliated private value auctions: a copula–quantile density approach. SSRN preprint. doi:10.2139/ssrn.7273869

Examples

```
set.seed(1)
Q <- function(u) (1 - 8 * u / 9)^(-1/2)
sim <- fpa_simulate(60, 3, Q, family = "clayton",
                    theta = copula_tau_to_theta(0.5, "clayton"))
fpa_corner_test(sim$bids, v_s = 1, family = "clayton")
```

`fpa_density` *Nonparametric estimation of the private-value density in first-price auctions*

Description

Two-step estimation of the density f of independent private values. The first step recovers pseudo-values $\hat{v}_i$ from the bids (by default with the Hickman–Hubbard boundary-corrected GPV inversion, or with any [fpa_values()] fit); the second step estimates the density of the pseudo-values by * ‘"wavelet_ll"‘: linear wavelet estimator followed by local-linear smoothing (the best performer in Doosti, Dewan and Sbai 2026); * ‘"wavelet_linear"‘, ‘"wavelet_hard"‘, ‘"wavelet_block"‘: linear, hard-thresholded and block-thresholded wavelet estimators; * ‘"wavelet_theory"‘: the hard-thresholding estimator of Theorem 2 of Doosti, Dewan and Sbai (2026) on a periodised basis, with coefficients $\hat{\beta}_{j,k} = N^{-1}\sum_i \psi_{j,k}(\Pi\hat{v}_i)$, coarsest level $\tau = 3$, finest level $2^{j_1} \approx \sigma_N^{-1}$ and level-dependent thresholds $\eta_j = \kappa\, 2^{j/2}\sigma_N$, $\sigma_N = N^{-2/5}(\log N)^{1/2}$; by default κ is set so that the threshold at the finest level is the universal threshold $\hat{s}\sqrt{2\log N}$ ($\hat{s}$: median absolute finest-level coefficient / 0.6745); pass ‘kappa‘ to override; * ‘"kde_hh"‘: the boundary-corrected kernel estimator of Hickman and Hubbard (2015), i.e. GPV with boundary correction at both stages; * ‘"kde"‘: Gaussian kernel with Sheather–Jones bandwidth; * ‘"empirical_quantile"‘: finite-difference density from the empirical quantile function of the pseudo-values; * ‘"marmer_shneyerov"‘: the quantile-based estimator of Marmer and Shneyerov (2012), computed directly from the bids (no first stage).

Because the wavelet basis lives on $[0,1]$, bids and pseudo-values are mapped to the unit interval by a monotone map (‘scale‘): with ‘"auto"‘ the identity when the bids lie in $[0,1]$ (pseudo-values outside are projected onto $[0,1]$ for the wavelet coefficients), otherwise $x \mapsto (\log x - a)/(b-a)$ (‘"log"‘, which handles the heavy right tail of the GPV ratio) or a linear map (‘"minmax"‘). Densities are reported on the original scale through the Jacobian.

Usage

```
fpa_density(
  bids,
  n = NULL,
  method = c("wavelet_ll", "wavelet_linear", "wavelet_hard", "wavelet_block",
   "wavelet_theory", "kde_hh", "kde", "empirical_quantile", "marmer_shneyerov"),
  first_stage = "gpv_hh",
  x = NULL,
  j0 = 5L,
  h = 0.15,
  scale = c("auto", "log", "minmax", "none"),
```

```
  filter = "db3",
  dwt_filter = "coif1",
  levels = 3L,
  kernel = "triweight",
  side = c("lower", "both"),
  ...
)
```

Arguments

`bids`	pooled bids (vector) or an ‘L‘ by ‘n‘ matrix.
`n`	bidders per auction (from ‘ncol(bids)‘ for a matrix).
`method`	second-stage estimator; see Description.
`first_stage`	‘"gpv_hh"‘ (default), ‘"gpv"‘, an [fpa_values()] object, or the name of a [qdf()] method to use through [fpa_values()].
`x`	evaluation grid on the value scale. Defaults to 200 points spanning the pseudo-values (excluding the extreme 0.5 percent at each end).
`j0`	resolution level, or ‘"cv"‘ to choose ‘j0‘ (and ‘h‘) by cross-validated held-out log-likelihood ([select_fpa_density()]).
`h`	local-linear bandwidth for ‘"wavelet_ll"‘ (ignored if ‘j0 = "cv"‘).
`scale`	‘"auto"‘, ‘"log"‘, ‘"minmax"‘ or ‘"none"‘.
`filter`, `dwt_filter`, `levels`	wavelet settings; see [qdf_wavelet()].
`kernel`, `side`	kernel and correction side for [kde_hh()] (used in the ‘"gpv_hh"‘ first stage and the ‘"kde_hh"‘ second stage).
`...`	for ‘first_stage‘ given as a [qdf()] method name, further arguments to [fpa_values()]; for ‘"wavelet_block"‘, arguments to [threshold_coefficients()]; for ‘j0 = "cv"‘, ‘j0_grid‘, ‘h_grid‘, ‘folds‘, ‘criterion‘.

Value

An object of class ‘"fpa_density"‘: a list with ‘x‘, ‘f‘ (density on the value scale), ‘values‘ (pseudo-values), ‘values_u‘ (on the unit scale), ‘map‘, ‘method‘, ‘j0‘, ‘h‘, ‘first_stage‘, ‘n‘, and ‘fun‘, a function returning the density at arbitrary values.

References

Doosti, H., Dewan, I. and Sbai, E. (2026). Adaptive wavelet estimation of private-value densities in first-price auctions. SSRN preprint. doi:10.2139/ssrn.7618223

Hickman, B. R. and Hubbard, T. P. (2015). *Journal of Applied Econometrics*, 30, 739–762.

Marmer, V. and Shneyerov, A. (2012). *Journal of Econometrics*, 167, 345–357.

Examples

```
set.seed(1)
sim <- fpa_simulate(40, 5, Q = function(u) qbeta(u, 2, 2))
fd <- fpa_density(sim$bids, method = "wavelet_ll", j0 = 5, h = 0.1)
plot(fd, true = function(v) dbeta(v, 2, 2))
# timber auctions, three bidders, log scale
data(timber)
fd3 <- fpa_density(timber$n3, method = "wavelet_ll", j0 = 4, h = 0.15)
plot(fd3)
```

`fpa_density_boot` *Bootstrap confidence intervals and bands for any private-value density estimator*

Description

Pointwise percentile intervals and a uniform confidence band for the private-value density estimated by any [fpa_density()] method, for example the quantile-based estimator of Marmer and Shneyerov (2012) or the wavelet estimators. The whole two-step procedure (first stage, unit map, second stage) is repeated on 'B' bootstrap samples. The uniform band is $\hat{f}(x) \pm c^* \, \hat{\sigma}^*(x)$, with $\hat{\sigma}^*$ the bootstrap standard deviation and c^* the $1-\alpha$ quantile of $\sup_x |\hat{f}^*(x) - \hat{f}(x)| / \hat{\sigma}^*(x)$. For the GPV estimator, [fpa_density_ci()] gives the analytic studentisation of Ma, Marmer and Shneyerov (2019).

The percentile bootstrap reproduces the variability of the estimator but not its smoothing bias, so at the usual bandwidths the intervals can under-cover in moderate samples (in a small check with the Marmer–Shneyerov estimator, 240 bids and nominal level 0.95, pointwise coverage was about 0.86). Undersmoothing, i.e. a smaller bandwidth than is optimal for estimation, is the usual remedy.

Usage

```
fpa_density_boot(
  bids,
  n = NULL,
  method = "marmer_shneyerov",
  x = NULL,
  B = 199,
  level = 0.95,
  resample = c("bids", "auctions", "blocks"),
  L = "auto",
  ...
)

## S3 method for class 'fpa_density_boot'
plot(x, true = NULL, ...)
```

Arguments

bids	pooled bids (vector) or an ‘L‘ by ‘n‘ matrix.
n	bidders per auction (from ‘ncol(bids)‘ for a matrix).
method	second-stage estimator; see Description.
x	evaluation grid on the value scale. Defaults to 200 points spanning the pseudo-values (excluding the extreme 0.5 percent at each end).
B	number of bootstrap replications.
level	confidence level.
resample	‘"bids"‘ (pooled bids with replacement) or ‘"auctions"‘ (whole auctions; requires a bid matrix).
L	block length when ‘resample = "blocks"‘: ‘"auto"‘ for the rule of [block_length_auto()], or an integer. Ignored by the other schemes.
...	further arguments to [fpa_density()] (e.g. ‘j0‘, ‘h‘, ‘first_stage‘).
true	optional true density function (simulated data).

Details

If the auctions have a time order and the same bidders recur, resampling auctions or bids independently understates the sampling uncertainty, in proportion to the dependence; use ‘resample = "blocks"‘ in that case.

Value

an object of class ‘"fpa_density_boot"‘ with ‘x‘, ‘f‘, ‘se‘, ‘lower‘, ‘upper‘ (pointwise), ‘band_lower‘, ‘band_upper‘, ‘crit‘, ‘level‘, ‘method‘, ‘B‘ and the matrix ‘boot‘ of bootstrap estimates.

Reproducibility

Call [set.seed()] beforehand to make the bootstrap reproducible.

Examples

```
set.seed(1)
sim <- fpa_simulate(60, 4, Q = function(u) qbeta(u, 2, 2))
bb <- fpa_density_boot(sim$bids, method = "marmer_shneyerov",
                       x = seq(0.15, 0.85, by = 0.05), B = 49)
bb
plot(bb, true = function(v) dbeta(v, 2, 2))
```

`fpa_density_ci` *Confidence intervals and uniform confidence bands for the private-value density*

Description

Inference for the two-step estimator of Guerre, Perrigne and Vuong (2000) following Ma, Marmer and Shneyerov (2019). The first stage uses the empirical distribution function of the bids and a fourth-order triweight kernel density estimate with bandwidth $h_g = 3.72\,\hat{\sigma}_b\,n^{-1/5}$; bids within h_g of the sample extremes are trimmed. The second stage is a triweight kernel density estimate of the pseudo-values with bandwidth $h_f = 3.15\,\hat{\sigma}_v\,n_T^{-1/5}$, where $\hat{\sigma}_v$ and n_T are the standard deviation and number of the trimmed pseudo-values (Section 8 of the paper). The trimmed pseudo-values are those of bids at least h_g from the sample extremes. As scale estimate we use Silverman's robust $\min(\hat{\sigma}, \mathrm{IQR}/1.349)$ rather than the standard deviation alone; the two coincide in the paper's designs, but the robust version protects the bandwidths against the few very large pseudo-values that arise where the estimated bid density is small.

Standard errors use the estimator (4.2) of the asymptotic variance, which accounts for the estimation error of the first stage. Pointwise intervals are percentile bootstrap intervals (Theorem 4.2; the default) or normal intervals $\hat{f} \pm z_{1-\alpha/2}\,\mathrm{se}$. The analytic standard error contains only the leading, first-stage term of the variance; in moderate samples, and where $F(v)$ is small, the second-stage kernel variance is not negligible and normal intervals can under-cover, whereas the percentile intervals and the uniform band, whose critical value is bootstrapped, are not affected. The uniform band over 'grid' is $\hat{f} \pm \zeta^*\,\mathrm{se}$, where ζ^* is the $1-\alpha$ quantile of $\sup_v |\hat{f}^*(v) - \hat{f}(v)|/\mathrm{se}(v)$ over bootstrap samples of the pooled bids (Theorem 5.2 and Corollary 5.1 of the paper).

The theory requires 'grid' to lie in the interior of the support of the valuations. By default it spans the 5th to 95th percentiles of the pseudo-values of the bids between the 10th and 90th bid percentiles, which keeps it away from the long right tail of pseudo-values that the first stage produces where the bid density is small; with real data, choose the range with care, since the band is only valid where the density of values is bounded away from zero. Observations whose estimated bid density is not positive (possible with a fourth-order kernel near the boundary) are dropped.

Usage

```
fpa_density_ci(
  bids,
  n = NULL,
  grid = NULL,
  level = 0.95,
  B = 499,
  pointwise = c("percentile", "normal"),
  h_g = NULL,
  h_f = NULL,
  trim_estimator = FALSE,
  constants = c(3.72, 3.15),
  sd_v = c("trimmed", "all"),
  se = c("augmented", "asymptotic")
)
```

Arguments

bids	pooled bids (vector) or an ‘L‘ by ‘n‘ matrix.
n	number of bidders per auction (from ‘ncol(bids)‘ for a matrix).
grid	evaluation points; default 201 points between the 5th and 95th percentiles of the pseudo-values of the bids between the 10th and 90th bid percentiles (see Details).
level	confidence level.
B	number of bootstrap replications.
pointwise	‘"normal"‘ or ‘"percentile"‘.
h_g, h_f	optional bandwidths overriding the rules above.
trim_estimator	logical. The trimming of bids within h_g of the sample extremes is always used to compute h_f. With the default ‘FALSE‘ all pseudo-values then enter the density estimate and its variance, as in the authors’ Monte Carlo implementation; with ‘TRUE‘ the trimmed bids are also excluded from the estimate (the trimming factor T_{il} of the paper), which with rule-of-thumb bandwidths removes a substantial part of the support in moderate samples.
constants	rule-of-thumb constants for h_g and h_f (default ‘c(3.72, 3.15)‘, the Silverman constants for the fourth- and second-order triweight kernels, as in Section 8 of the paper).
sd_v	standard deviation used in h_f: of the ‘"trimmed"‘ pseudo-values with n_T (default, as in the paper) or of ‘"all"‘ pseudo-values with n.
se	‘"augmented"‘ (default): the asymptotic variance (4.2) plus the second-stage kernel variance $\hat{f}(v)\int K^2/(nh_f)$, a term of smaller order that is asymptotically negligible but keeps the standard error positive and improves it in moderate samples, where the U-statistic (4.2) can be close to or below zero; ‘"asymptotic"‘: the estimator (4.2) alone, as in the paper. Both give asymptotically valid uniform bands.

Value

An object of class ‘"fpa_density_ci"‘: a list with ‘v‘ (grid), ‘f‘ (estimate), ‘se‘, ‘lower‘, ‘upper‘ (pointwise interval), ‘band_lower‘, ‘band_upper‘ (uniform band), ‘crit‘ (ζ^*), ‘level‘, ‘h_g‘, ‘h_f‘, ‘values‘ (trimmed pseudo-values) and ‘B‘.

Reproducibility

Fold assignment (or bootstrap resampling) uses R’s random number generator; call [set.seed()] beforehand to make results exactly reproducible.

References

Ma, J., Marmer, V. and Shneyerov, A. (2019). Inference for first-price auctions with Guerre, Perrigne, and Vuong’s estimator. *Journal of Econometrics*, 211, 507–538. doi:10.1016/j.jeconom.2019.02.006

Guerre, E., Perrigne, I. and Vuong, Q. (2000). Optimal nonparametric estimation of first-price auctions. *Econometrica*, 68, 525–574.

Examples

```
set.seed(1)
sim <- fpa_simulate(100, 5, Q = function(u) u)       # uniform values
ci  <- fpa_density_ci(sim$bids, grid = seq(0.2, 0.8, by = 0.01), B = 99)
ci
plot(ci, true = function(v) dunif(v))
```

`fpa_density_cv` *Compare density estimators by held-out log-likelihood*

Description

Outer ‘folds‘-fold cross-validation of several second-stage estimators against a common target (the held-out pseudo-values from the first stage fitted on the training fold), as in Table 3 of Doosti, Dewan and Sbai (2026). Wavelet tuning parameters are fixed at ‘j0‘, ‘h‘.

Usage

```
fpa_density_cv(
  bids,
  n = NULL,
  methods = c("wavelet_ll", "kde_hh", "marmer_shneyerov"),
  folds = 5L,
  j0 = 5L,
  h = 0.15,
  first_stage = "gpv_hh",
  scale = "auto",
  kernel = "triweight",
  side = "lower",
  ...
)
```

Arguments

`bids`	pooled bids (vector) or an ‘L‘ by ‘n‘ matrix.
`n`	bidders per auction (from ‘ncol(bids)‘ for a matrix).
`methods`	character vector of second-stage methods.
`folds`	number of outer folds.
`j0`	resolution level, or ‘"cv"‘ to choose ‘j0‘ (and ‘h‘) by cross-validated held-out log-likelihood ([select_fpa_density()]).
`h`	local-linear bandwidth for ‘"wavelet_ll"‘ (ignored if ‘j0 = "cv"‘).
`first_stage`	‘"gpv_hh"‘ (default), ‘"gpv"‘, an [fpa_values()] object, or the name of a [qdf()] method to use through [fpa_values()].
`scale`	‘"auto"‘, ‘"log"‘, ‘"minmax"‘ or ‘"none"‘.

`kernel`, `side`	kernel and correction side for [kde_hh()] (used in the ‘"gpv_hh"‘ first stage and the ‘"kde_hh"‘ second stage).
`...`	for ‘first_stage‘ given as a [qdf()] method name, further arguments to [fpa_values()]; for ‘"wavelet_block"‘, arguments to [threshold_coefficients()]; for ‘j0 = "cv"‘, ‘j0_grid‘, ‘h_grid‘, ‘folds‘, ‘criterion‘.

Value

a named numeric vector of summed held-out log-likelihoods.

`fpa_evaluate_values` *Evaluate recovered values or an estimated density against the truth*

Description

Performance criteria used in the papers behind this package.

‘fpa_evaluate_values()‘ scores recovered private values (Doosti, Dewan and Talebian 2025): ‘mise‘, the mean integrated squared error between a Gaussian kernel density (Sheather–Jones bandwidth) of the values and the true density; ‘loglik‘, $\sum_i \log f(\hat{v}_i)$; ‘ks‘, the Kolmogorov–Smirnov distance between the empirical distribution of the values and F; and ‘msep‘, the mean squared error against the true values when they are supplied (Doosti 2026a), with ‘msep_trim‘ excluding a share ‘trim‘ at each end of the true-value distribution.

‘fpa_evaluate_density()‘ scores an [fpa_density()] object on its grid: ‘mise‘ over the whole grid and over the upper ‘tail‘ share of the support, and the expected log-likelihood $\int f \log \hat{f}$ (Doosti, Dewan and Sbai 2026).

Usage

```
fpa_evaluate_values(values, pdf, cdf = NULL, true = NULL, trim = 0.05)

fpa_evaluate_density(density, pdf, tail = 0.1)
```

Arguments

`values`	numeric vector of recovered values, or an [fpa_values()] object.
`pdf`, `cdf`	true density and distribution functions.
`true`	optional true values, aligned with ‘values‘ (for an [fpa_values()] object, in the layout of the original bids).
`trim`	trimming share for ‘msep_trim‘.
`density`	an [fpa_density()] object.
`tail`	share of the upper support for the tail MISE.

Value

a named numeric vector.

Examples

```
set.seed(1)
sim <- fpa_simulate(20, 5, Q = function(u) qgamma(u, 10, 2))
fv  <- fpa_values(sim$bids, method = "bernstein", bandwidth = "bcv", monotone = "spline")
fpa_evaluate_values(fv, pdf = function(x) dgamma(x, 10, 2),
                    cdf = function(x) pgamma(x, 10, 2), true = sim$values)
```

`fpa_montecarlo` *Monte Carlo comparison of estimators*

Description

Repeats: simulate auctions with [fpa_simulate()], apply each estimator in ‘estimators‘ to the bids, and score it with ‘score‘. Designed to regenerate the simulation tables of the papers behind this package.

Usage

```
fpa_montecarlo(
  R,
  L,
  n,
  Q,
  q = NULL,
  family = "independence",
  theta = NULL,
  estimators,
  score,
  seed = NULL,
  verbose = FALSE,
  p0 = 0,
  eta = 0
)
```

Arguments

`R`	number of replications.
`L`, `n`	auctions and bidders per replication.
`Q`, `q`, `family`, `theta`	private-value model, as in [fpa_simulate()].
`estimators`	a named list of functions ‘function(bids, values)‘ returning an object accepted by ‘score‘ (typically [fpa_values()] or [fpa_density()] fits, or a numeric vector).
`score`	a function ‘function(fit, bids, values)‘ returning a named numeric vector, e.g. a wrapper around [fpa_evaluate_values()].
`seed`	optional seed.

verbose	print progress every 10 replications.
p0, eta	screening quantile of the reserve price and CRRA parameter, passed to [fpa_bid_function()] and [fpa_simulate()].

Value

a data frame with columns ‘rep‘, ‘estimator‘ and one column per criterion; ‘summary()‘ on the result (via ‘aggregate‘) gives the table.

Examples

```
set.seed(1)
pdf <- function(x) dgamma(x, 10, 2); cdf <- function(x) pgamma(x, 10, 2)
mc <- fpa_montecarlo(R = 5, L = 20, n = 5, Q = function(u) qgamma(u, 10, 2),
  estimators = list(
    B_B  = function(b, v) fpa_values(b, method = "bernstein", bandwidth = "bcv"),
    GPV  = function(b, v) fpa_values(b, method = "gpv")),
  score = function(fit, b, v) fpa_evaluate_values(fit, pdf, cdf, v))
aggregate(cbind(mise, loglik, ks, msep) ~ estimator, mc, mean)
```

---

fpa_optimal_reserve *Optimal reserve price from recovered private values*

Description

Maximises the plug-in revenue [fpa_revenue()] over a grid of reserve prices.

Usage

```
fpa_optimal_reserve(values, eta = NULL, v_s = 0, grid = NULL)
```

Arguments

values	an [fpa_values()] object fitted to a bid *matrix*, or an ‘L‘ by ‘n‘ matrix of private values.
eta	CRRA parameter; taken from ‘values‘ when it is an [fpa_values()] object.
v_s	seller’s valuation of the object (revenue when unsold).
grid	candidate reserve prices; defaults to 200 points between the 2nd and 98th percentiles of the values.

Value

a list with ‘r_star‘, ‘revenue‘ and the evaluated ‘grid‘ and ‘R‘.

`fpa_revenue` *Plug-in estimator of the seller's expected revenue*

Description

Estimates the seller's expected revenue $R(r)$ in a symmetric first-price auction with reserve price r, from recovered private values (Zincenko 2024, eq. 6; Doosti 2026b, eq. 11). With L auctions of n bidders, pooled pseudo-values $\hat{v}_1 \le \ldots \le \hat{v}_N$, their empirical distribution $\hat{F}$, and the winner's value $\check{V}_\ell = \max_i \hat{v}_{i\ell}$ in auction ℓ,

$$\hat{R}(r) = v_s \hat{F}(r)^n + \frac{1}{L}\sum_{\ell=1}^{L} 1\{\check{V}_\ell > r\}\,\hat{\beta}(\check{V}_\ell; r),$$

$$\hat{\beta}(v; r) = \frac{r\,\hat{F}(r)^{e_1} + e_1 N^{-1}\sum_{r \le \hat{v}_j \le v} \hat{v}_j \hat{F}(\hat{v}_j)^{e_2}}{\hat{F}(v)^{e_1}},$$

with $e_1 = (n-1)/(1-\eta)$, $e_2 = (n+\eta-2)/(1-\eta)$ for CRRA bidders with parameter η ($\eta = 0$: risk neutral), and v_s the seller's own valuation.

Usage

```
fpa_revenue(values, r, eta = NULL, v_s = 0)
```

Arguments

`values`	an [fpa_values()] object fitted to a bid *matrix*, or an 'L' by 'n' matrix of private values.
`r`	reserve prices at which to evaluate the revenue.
`eta`	CRRA parameter; taken from 'values' when it is an [fpa_values()] object.
`v_s`	seller's valuation of the object (revenue when unsold).

Value

numeric vector $\hat{R}(r)$.

References

Zincenko, F. (2024). Estimation and inference of seller's expected revenue in first-price auctions. *Journal of Econometrics*, 241(1), 105734. doi:10.1016/j.jeconom.2024.105734

Doosti, H. (2026b). Estimating seller's expected revenue in first-price sealed-bid auctions via quantile density functions. SSRN preprint. doi:10.2139/ssrn.7071380

Examples

```
set.seed(1)
sim <- fpa_simulate(100, 3, Q = function(u) 10 * u)      # values U[0, 10]
fv  <- fpa_values(sim$bids, method = "bernstein", bandwidth = "bcv")
r   <- 3:8
cbind(r, estimate = fpa_revenue(fv, r, v_s = 2),
      truth = fpa_revenue_model(function(u) 10 * u, n = 3, r = r, v_s = 2))
```

`fpa_revenue_boot` *Bootstrap inference for the expected-revenue estimator*

Description

Recomputes the pseudo-values and the plug-in revenue on ‘B‘ bootstrap samples of whole auctions (by default; see ‘resample‘) (Zincenko 2024, Section 4.3; Doosti 2026b) and returns the bootstrap standard error $\hat{\sigma}^*(r) = \{B^{-1}\sum_b(\hat{R}_b^*(r) - \hat{R}(r))^2\}^{1/2}$, symmetric percentile-$t$ pointwise confidence intervals (critical value: the $1-\alpha$ quantile of $|Z_b^*(r)|$), and a uniform confidence band over ‘r‘ based on the bootstrap distribution of $\sup_r |Z_b^*(r)|$, $Z_b^*(r) = \{\hat{R}_b^*(r) - \hat{R}(r)\}/\hat{\sigma}^*(r)$. (The original simulation code used the $1-\alpha/2$ quantile of Z^* for the pointwise intervals; the symmetric version guarantees that the band contains the intervals.)

Usage

```
fpa_revenue_boot(
  bids,
  r,
  B = 200,
  eta = 0,
  v_s = 0,
  resample = c("auctions", "bids", "blocks"),
  L = "auto",
  alpha = c(0.1, 0.05, 0.01),
  trim = Inf,
  ...
)
```

Arguments

`bids`	‘L‘ by ‘n‘ bid matrix.
`r`	reserve prices.
`B`	number of bootstrap replications.
`eta, v_s`	as in [fpa_revenue()].
`resample`	‘"auctions"‘ (whole auctions with replacement; the default) or ‘"bids"‘ (pooled bids with replacement). The revenue estimator uses the highest pseudo-value in each auction, and pooled resampling breaks that structure: it shifts the bootstrap distribution by an amount of the same order as the estimation error, so the

intervals under-cover (about 0.65-0.80 at nominal 0.90 for the Bernstein estimator with $m = \lceil \sqrt{N} \rceil$ in the design of Zincenko, 2024). ‘"bids"‘ is kept for comparison only.

`L` block length when ‘resample = "blocks"‘: ‘"auto"‘ for the rule of [block_length_auto()], or an integer. ‘L = 1‘ is the i.i.d. auction bootstrap. Ignored by the other schemes.

`alpha` significance levels for the intervals and band.

`trim` discard bootstrap deviations larger than this constant (c_b in Zincenko's eq. 24; default no trimming).

`...` arguments to [fpa_values()] (e.g. ‘method‘, ‘bandwidth‘, ‘monotone‘). A data-driven ‘bandwidth‘ is selected once, on the full sample, and held fixed in every bootstrap replication, as assumed by the theory (Doosti 2026b, Theorems 6–7).

Details

If the auctions have a time order and the same bidders recur, resampling auctions or bids independently understates the sampling uncertainty, in proportion to the dependence; use ‘resample = "blocks"‘ in that case.

Value

an object of class ‘"fpa_revenue_boot"‘: a list with ‘r‘, ‘estimate‘, ‘se‘, ‘ci‘ (array ‘r‘ by ‘c(lower, upper)‘ by ‘alpha‘), ‘band‘ (same layout), the bootstrap critical values, the ‘B‘ by ‘length(r)‘ matrix of bootstrap estimates and, when the bandwidth was selected from the data, ‘bandwidth‘: a list with the selected ‘h‘ and ‘m‘, the ‘selector‘, and ‘at_boundary‘/‘boundary‘ (see [select_bandwidth()]).

Reproducibility

Fold assignment (or bootstrap resampling) uses R's random number generator; call [set.seed()] beforehand to make results exactly reproducible.

Examples

```
set.seed(1)
sim <- fpa_simulate(60, 3, Q = function(u) 10 * u)
rb <- fpa_revenue_boot(sim$bids, r = c(3, 5, 7), B = 50, v_s = 2,
                       method = "bernstein", bandwidth = "sqrt")
rb
```

---

fpa_revenue_model *Expected revenue and optimal reserve price of a known model*

Description

Computes the seller's expected revenue $R(r) = v_s\,\delta(p) + \int_p^1 \beta(u;p)\,d\delta(u)$ for a private-value model with marginal quantile function 'Q', copula multiplier ψ and diagonal δ ([copula_psi()], [copula_delta()]), where $\beta(\cdot;p)$ is the equilibrium bid with reserve price $r = Q(p)$ ([fpa_bid_function()]). Under independence and risk neutrality this is the revenue used as the truth in Zincenko (2024). 'fpa_optimal_reserve_model()' finds the interior maximiser p^* of R from the first-order condition $\delta'(p)(v_s - Q(p)) + q(p)\int_p^1 e^{\Lambda(p)-\Lambda(u)}\delta'(u)\,du = 0$.

Usage

```
fpa_revenue_model(
  Q,
  n,
  r = NULL,
  p = NULL,
  q = NULL,
  family = "independence",
  theta = NULL,
  v_s = 0,
  eta = 0,
  U = fpa_ugrid()
)

fpa_optimal_reserve_model(
  Q,
  n,
  q = NULL,
  family = "independence",
  theta = NULL,
  v_s = 0,
  eta = 0,
  p_lo = 1e-04,
  p_hi = 0.98,
  U = fpa_ugrid()
)
```

Arguments

Q	quantile function of private values (a function of 'u').
n	number of bidders.
r	reserve prices (on the value scale); alternatively give 'p'.
p	reserve quantiles.
q	quantile density of private values (a function of 'u'); if 'NULL' it is obtained by numerical differentiation of 'Q'.
family	'"independence"', '"clayton"', '"gumbel"' or '"frank"'.
theta	copula parameter (ignored for independence).

v_s	seller's valuation.
eta	CRRA risk-aversion parameter of the bidders.
U	quantile grid; see [fpa_ugrid()].
p_lo, p_hi	search interval for the optimal reserve quantile.

Value

'fpa_revenue_model()': numeric vector of expected revenues. 'fpa_optimal_reserve_model()': a list with 'p_star', 'r_star', 'revenue' and 'interior'.

Examples

```
Q <- function(u) (1 - 8 * u / 9)^(-1/2)             # truncated Pareto on [1, 3]
fpa_revenue_model(Q, n = 3, p = c(0.1, 0.3, 0.5), v_s = 1)
fpa_optimal_reserve_model(Q, n = 3, v_s = 1)
fpa_optimal_reserve_model(Q, n = 3, v_s = 1, family = "clayton", theta = 2)
```

---

fpa_revenue_qd *Expected revenue and optimal reserve by the quantile-density functional*

Description

Estimates the seller's expected revenue $R(p)$ at reserve quantile p (reserve price $r = Q_V(p)$) and the optimal reserve from the revenue functional of Doosti (2026c, Proposition 1),

$$R(p) = v_s\,\delta(p) + \int_p^1 \beta(u;p)\,\delta'(u)\,du,$$

where δ is the diagonal section of the copula of the private values ($\delta(u) = u^n$ under independence) and $\beta(u;p)$ is the equilibrium bid in quantile space with reserve $Q_V(p)$. Writing $K(w,u) = e^{\Lambda(w)-\Lambda(u)}$, $\Lambda' = 1/\psi$, the proposition's $\beta(u;p) = Q_V(u) - \int_p^u K(w,u) q_V(w)\,dw$ is equivalently

$$\beta(u;p) = Q_B(u) + \int_0^p K(w,u)\,q_V(w)\,dw,$$

because without a reserve the equilibrium bid quantile function is the observed bid quantile function Q_B. The estimator uses this second form: the empirical bid quantile function carries the upper tail, and the quantile density $\hat{q}_V$ of the pseudo-values is needed only below the reserve quantile, where it is estimated reliably. The optimal reserve maximises $\hat{R}$ over the quantile grid; at an interior optimum it satisfies the generalised Myerson condition $r^* = v_s + q_V(p^*)H(p^*;\theta)$, which reduces to $r^* = v_s + (1-F(r^*))/f(r^*)$ under independence. This is Stage 3 of the copula–quantile density estimator; Stages 1–2 are [fpa_values()] with the same 'family'.

Usage

```
fpa_revenue_qd(
  values,
  r = NULL,
  p = NULL,
  v_s = 0,
  qdf_method = "poisson",
  bandwidth = "rlcv",
  ...
)

fpa_optimal_reserve_qd(
  values,
  v_s = 0,
  qdf_method = "poisson",
  bandwidth = "rlcv",
  p_max = 0.98,
  ...
)
```

Arguments

values	an [fpa_values()] object; its bids, pseudo-values, copula specification (fitted $\hat{\theta}$, or independence) and CRRA parameter are used.
r	reserve prices at which to evaluate $\hat{R}$; alternatively 'p'.
p	reserve quantiles.
v_s	seller's valuation.
qdf_method, bandwidth	quantile density estimator for the pseudo-values, passed to [qdf()] (default: Poisson weights, robust likelihood cross-validation, as in Doosti 2026c).
...	further arguments to [qdf()].
p_max	largest reserve quantile searched by 'fpa_optimal_reserve_qd()'.

Value

'fpa_revenue_qd()': numeric vector $\hat{R}$. 'fpa_optimal_reserve_qd()': a list with 'r_star', 'p_star', 'revenue', 'interior' ('FALSE' when $\hat{R}$ is maximised at the smallest reserve, i.e. no binding reserve is optimal) and the evaluated curve 'p', 'R'.

References

Doosti, H. (2026c). Optimal reserve prices in affiliated private value auctions: a copula–quantile density approach. SSRN preprint. doi:10.2139/ssrn.7273869

Examples

```
set.seed(1)
Q <- function(u) (1 - 8 * u / 9)^(-1/2)          # truncated Pareto on [1, 3]
th <- copula_tau_to_theta(0.25, "clayton")
sim <- fpa_simulate(50, 3, Q, family = "clayton", theta = th)
fv  <- fpa_values(sim$bids, method = "poisson", bandwidth = "rlcv", family = "clayton")
fpa_optimal_reserve_qd(fv, v_s = 1)[c("r_star", "p_star", "interior")]
fpa_optimal_reserve_model(Q, 3, family = "clayton", theta = th, v_s = 1)$r_star
```

`fpa_simulate` *Simulate first-price sealed-bid auctions*

Description

Draws private values with a given marginal quantile function and an exchangeable Archimedean copula (independent private values by default), and computes exact equilibrium bids with [fpa_bid_function()].

Usage

```
fpa_simulate(
  L,
  n,
  Q,
  q = NULL,
  family = "independence",
  theta = NULL,
  p0 = 0,
  eta = 0,
  bid_function = NULL
)
```

Arguments

`L`	number of auctions.
`n`	number of bidders per auction.
`Q`	quantile function of private values (a function of ‘u‘).
`q`	quantile density of private values (a function of ‘u‘); if ‘NULL‘ it is obtained by numerical differentiation of ‘Q‘.
`family`	‘"independence"‘, ‘"clayton"‘, ‘"gumbel"‘ or ‘"frank"‘.
`theta`	copula parameter (ignored for independence).
`p0`	screening quantile of the reserve price (default 0, no reserve).
`eta`	CRRA risk-aversion parameter in $[0, 1)$ (0: risk neutral).
`bid_function`	optional precomputed output of [fpa_bid_function()].

Value

a list with matrices ‘values‘ and ‘bids‘ (‘L‘ by ‘n‘) and the bid function used.

Examples

```
set.seed(1)
sim <- fpa_simulate(20, 5, Q = function(u) qlnorm(u, 0, 0.5))
str(sim$bids)
```

`fpa_values` *Recover private values from first-price auction bids*

Description

Nonparametric inversion of the equilibrium bid function of a symmetric first-price sealed-bid auction through the quantile density of bids:

$$\hat{v} = b + \lambda^{-1}\{\psi(\hat{u})\,\hat{q}(\hat{u})\}, \qquad \hat{u} = \hat{G}(b),$$

where $\hat{q}$ is a quantile density estimate ([qdf()]), ψ is the copula multiplier (equal to $u/(n-1)$ under independent private values, so that $\psi q = G/\{(n-1)g\}$ and the formula is that of Guerre, Perrigne and Vuong 2000), and λ^{-1} accounts for risk aversion (Guerre, Perrigne and Vuong 2009): the identity for risk-neutral bidders and $y \mapsto (1-\eta)y$ for CRRA utility $U(t) = t^{1-\eta}$.

‘method = "zincenko"‘ gives the kernel estimator of Zincenko (2024): a triweight kernel for $\hat{G}$ and $\hat{g}$ with normal-reference bandwidths, and linear extrapolation of the inverse bid function within $h_g^{0.99}$ of the boundaries instead of trimming; the implementation is adapted from code provided by F. Zincenko. ‘method = "gpv"‘ gives the conventional benchmark $\hat{v} = b + \hat{G}(b)/\{(n-1)\hat{g}(b)\}$ with a Gaussian kernel and Sheather–Jones bandwidth for $\hat{g}$; ‘"gpv_bc"‘ renormalises a triweight kernel by its mass inside the bid support (a simple boundary correction; the Hickman–Hubbard correction is provided by ‘fpa_density()‘).

Usage

```
fpa_values(
  bids,
  n = NULL,
  method = "indirect_poisson",
  bandwidth = "bcv",
  eta = 0,
  family = "independence",
  theta = NULL,
  u_scheme = c("i/(n+1)", "i/n"),
  monotone = c("none", "spline", "isotonic"),
  trim = 0.1,
  replace = c("all", "boundary"),
  df = 6,
  psi_bw = NULL,
  ...
)
```

Arguments

bids	numeric vector of pooled bids, or an ‘L‘ by ‘n‘ matrix (one row per auction). A matrix is required for ‘family != "independence"‘ with ‘theta = NULL‘, so that the copula parameter can be estimated from within-auction pairs.
n	number of bidders per auction (taken from ‘ncol(bids)‘ for a matrix).
method	quantile-density estimator passed to [qdf()], or ‘"gpv"‘, ‘"gpv_bc"‘, ‘"zincenko"‘ for the kernel benchmarks.
bandwidth	passed to [qdf()]: a selector name, the rule ‘"sqrt"‘ ($m = \lceil\sqrt{N}\rceil$ for ‘method = "bernstein"‘, as in Doosti 2026b), or a numeric bandwidth.
eta	CRRA risk-aversion parameter in $[0, 1)$; ‘0‘ is risk neutrality.
family	copula family of the private values: ‘"independence"‘, ‘"clayton"‘, ‘"gumbel"‘, ‘"frank"‘ (see [copula_psi()]), or ‘"nonparametric"‘ for the flexible copula multiplier of [copula_psi_np()] (the CopQDF-F estimator of Doosti 2026a). With ‘method = "gpv"‘ and a parametric family the inversion is the semiparametric estimator of Hubbard, Li and Paarsch (2012): a kernel bid density combined with a parametric copula.
theta	copula parameter; ‘NULL‘ estimates it by [copula_fit()].
u_scheme	scores $\hat{u}$ attached to the sorted bids: ‘"i/(n+1)"‘ (Hazen ranks, default) or ‘"i/n"‘.
monotone	‘"none"‘, ‘"spline"‘ or ‘"isotonic"‘.
trim	tail share excluded from the spline fit (default 0.1, i.e. the central 80 percent is used).
replace	for ‘monotone = "spline"‘: ‘"all"‘ or ‘"boundary"‘.
df	degrees of freedom of the natural spline (default 6).
psi_bw	beta-kernel bandwidth for ‘family = "nonparametric"‘; see [copula_psi_np()].
...	further arguments to [qdf()] (e.g. ‘kernel‘, ‘H‘, ‘j0‘, ‘m_range‘, ‘grid‘, ‘zero_at_origin‘). In particular ‘zero_at_origin = FALSE‘ switches the Bernstein estimator to the $\hat{Q}(0) = X_{(1)}$ anchor; with ‘method = "bernstein"‘, ‘bandwidth = "sqrt"‘, ‘u_scheme = "i/(n+1)"‘, ‘monotone = "none"‘ and ‘trim = 0‘ this reproduces the pseudo-values of the revenue estimator of Doosti (2026b) exactly.

Value

An object of class ‘"fpa_values"‘: a list with

bids, u sorted pooled bids and their scores

q, psi, markup the quantile density, the copula multiplier and the estimated markup $\lambda^{-1}(\psi q)$ at each bid

values_raw, values pseudo-values before and after the monotonicity step (identical when ‘monotone = "none"‘)

values_mat when ‘bids‘ is a matrix, ‘values‘ rearranged to the original ‘L‘ by ‘n‘ layout

qdf the fitted [qdf()] object (or ‘NULL‘ for the kernel benchmarks)

copula the copula specification / fit

Monotonicity

The equilibrium bid function is increasing, so $\hat{v}$ should be increasing in b. Departures occur mostly near the boundaries. ‘monotone = "spline"‘ follows Doosti, Dewan and Talebian (2025): a natural cubic spline of $\hat{v}$ on b is fitted to the central $1-2\,\mathrm{trim}$ share of the bids and its fitted values replace $\hat{v}$ either everywhere (‘replace = "all"‘, the published implementation) or only in the trimmed tails (‘replace = "boundary"‘). ‘monotone = "isotonic"‘ applies the pool-adjacent-violators rearrangement ([stats::isoreg()]). ‘"none"‘ returns the raw pseudo-values.

References

Guerre, E., Perrigne, I. and Vuong, Q. (2000). Optimal nonparametric estimation of first-price auctions. *Econometrica*, 68, 525–574.

Guerre, E., Perrigne, I. and Vuong, Q. (2009). Nonparametric identification of risk aversion in first-price auctions under exclusion restrictions. *Econometrica*, 77, 1193–1227.

Doosti, H., Dewan, I. and Talebian, M. (2025). *Economics Letters*, 257, 112670.

Doosti, H. (2026a). *Economics Letters*, 261, 112837.

Examples

```
set.seed(1)
sim <- fpa_simulate(20, 5, Q = function(u) qgamma(u, 2, 10))
fit <- fpa_values(sim$bids, method = "bernstein", bandwidth = "bcv")
fit
plot(fit, true = sim$values)
# risk-averse bidders with CRRA eta = 0.25
fit_ra <- fpa_values(sim$bids, method = "bernstein", bandwidth = 0.05, eta = 0.25)
# affiliated values, Clayton copula estimated from the data

sim2 <- fpa_simulate(50, 3, Q = function(u) 1 + 2 * u^2, family = "clayton", theta = 2)
fit2 <- fpa_values(sim2$bids, method = "indirect_poisson", family = "clayton")
fit2$copula$theta
```

---

fpa_virtual_value_model

Affiliated virtual value

Description

The virtual value of affiliated private values in quantile space,

$$\omega(u;\theta) = Q_V(u) - q_V(u)\,H(u;\theta), \qquad H(p) = \frac{1}{\delta'(p)}\int_p^1 K(p,u)\,\delta'(u)\,du,$$

with $K(w,u) = e^{\Lambda(w)-\Lambda(u)}$, $\Lambda' = 1/\psi$ and δ the diagonal section of the copula (Doosti 2026c). The derivative of expected revenue is $R'(p) = \delta'(p)\{v_s - \omega(p)\}$, so no binding reserve is optimal

(a corner) when $\omega(u) \geq v_s$ for all u; under independence ω is the Myerson virtual value $v - (1 - F(v))/f(v)$.

'fpa_virtual_value_model()' evaluates ω for a known model; 'fpa_virtual_value()' estimates it from an [fpa_values()] object, with the quantile function and quantile density of the pseudo-values estimated as in [fpa_revenue_qd()].

Usage

```
fpa_virtual_value_model(
  Q,
  q,
  n,
  family = "independence",
  theta = NULL,
  U = fpa_ugrid()
)

fpa_virtual_value(values, qdf_method = "poisson", bandwidth = "rlcv", ...)
```

Arguments

`Q`, `q`	quantile function and quantile density of the private values.
`n`	number of bidders.
`family`, `theta`	copula family and parameter, as in [copula_psi()].
`U`	quantile grid (default [fpa_ugrid()]).
`values`	an [fpa_values()] object fitted with a parametric copula.
`qdf_method`, `bandwidth`	quantile density estimator for the pseudo-values, passed to [qdf()].
`...`	further arguments to [qdf()].

Value

A list with the grid 'U', the virtual value 'omega' and the factor 'H'; 'fpa_virtual_value()' also returns the estimated 'Q' and 'q' on 'U' and the bandwidth 'h' used for the pseudo-values.

References

Doosti, H. (2026c). Optimal reserve prices in affiliated private value auctions: a copula–quantile density approach. SSRN preprint. doi:10.2139/ssrn.7273869

Examples

```
Q <- function(u) (1 - 8 * u / 9)^(-1/2)              # truncated Pareto on [1, 3]
q <- function(u) (4 / 9) * (1 - 8 * u / 9)^(-3/2)
vv <- fpa_virtual_value_model(Q, q, n = 3, family = "clayton", theta = 2)
range(vv$omega)
```

kde_hh *Boundary-corrected kernel density estimator of Hickman and Hubbard (2015)*

Description

Kernel density estimator with the transformation-based boundary correction of Karunamuni and Zhang (2008) at the lower endpoint of the support, as implemented by Hickman and Hubbard (2015) for first-price auctions (their ‘kspdf_bc‘; this function is an R translation of that code, whose authors ask that Hickman and Hubbard (2015) be cited). The bandwidth is Silverman’s rule scaled by Haerdle’s (1991) canonical-kernel constant. The estimate is set to zero outside the range of the data. With ‘side = "both"‘ the same correction is applied at the upper endpoint through the reflected sample.

Usage

```
kde_hh(
  x,
  data,
  kernel = "triweight",
  side = c("lower", "both"),
  hardle = TRUE,
  h = NULL
)
```

Arguments

x	evaluation points.
data	observed sample (bids or pseudo-values).
kernel	‘"triweight"‘ (default), ‘"gaussian"‘, ‘"epanechnikov"‘, ‘"quartic"‘, ‘"uniform"‘ or ‘"triangle"‘.
side	‘"lower"‘ (the original implementation) or ‘"both"‘.
hardle	logical; use Haerdle’s canonical bandwidth constant.
h	optional bandwidth override.

Value

a numeric vector with attribute ‘"h"‘ (the bandwidth used).

References

Hickman, B. R. and Hubbard, T. P. (2015). Replacing sample trimming with boundary correction in nonparametric estimation of first-price auctions. *Journal of Applied Econometrics*, 30, 739–762. doi:10.1002/jae.2385

Karunamuni, R. J. and Zhang, S. (2008). Some improvements on a boundary corrected kernel density estimator. *Statistics & Probability Letters*, 78, 499–507.

Examples

```
x <- rbeta(200, 2, 1)
g <- seq(0.01, 0.99, by = 0.01)
plot(g, kde_hh(g, x), type = "l"); lines(g, dbeta(g, 2, 1), lty = 2)
```

---

loclin_smooth *Local-linear smoother with Gaussian kernel*

Description

Post-smoothing of a wavelet estimate on a grid (Ramirez and Vidakovic 2010; Fan 1992): for each 'x' solves the weighted least-squares line through '(grid, y)' with weights $K((grid - x)/h)$ and returns its value at 'x'. Translated from the MATLAB routine 'loc_lin' of B. Vidakovic.

Usage

```
loclin_smooth(x, grid, y, h)
```

Arguments

x	evaluation points.
grid, y	tabulated function.
h	bandwidth.

Value

numeric vector.

---

make_kernel *Construct and check a kernel for the kernel-type estimators*

Description

'make_kernel()' turns a kernel density function into the object used by [qdf_kernel()], [qdf_kernel_corrected()], [qdf_jones()], [qdf_soni()] and [qdf()] (argument 'kernel'). The distribution function needed by the boundary-corrected and Soni estimators is computed numerically. 'check_kernel()' reports the properties that matter for these estimators: total mass, symmetry, support, order (the first non-vanishing moment), $\int k^2$ and whether the kernel takes negative values; it warns when a property required by the theory fails.

Usage

```
make_kernel(k, name = "custom", support = 1, n_grid = 20001L)

check_kernel(kernel, verbose = TRUE)
```

Arguments

k	a vectorised function, the kernel density.
name	a label.
support	half-width of an interval $[-s, s]$ containing the support (or, for kernels with unbounded support, essentially all of the mass).
n_grid	number of grid points for the numerical integration.
kernel	a kernel name, a function, or an object from ‘make_kernel()‘.
verbose	print the report.

Value

‘make_kernel()‘: an object of class ‘"qdf_kernel"‘ (a list with the density ‘d‘, the distribution function ‘p‘, ‘name‘ and the ‘check‘ results). ‘check_kernel()‘: a list of properties, printed invisibly.

Examples

```
# the biweight (quartic) kernel
biweight <- make_kernel(function(u) 15 / 16 * (1 - u^2)^2 * (abs(u) <= 1), "biweight")
biweight
x <- rgamma(100, 5)
qdf_kernel(c(0.25, 0.5, 0.75), x, h = 0.05, kernel = biweight)
check_kernel("epanechnikov")
```

---

phi_jk *Evaluate scaling functions ϕ_j, k(z) = 2^j/2φ(2^jz − k)*

Description

Evaluate scaling functions $\phi_{j,k}(z) = 2^{j/2}\phi(2^j z - k)$

Usage

```
phi_jk(z, j, k, filter = "db3", levels = 10L)
```

Arguments

z	numeric vector of arguments.
j	resolution level.
k	integer vector of shifts. For a single ‘k‘ a vector is returned; otherwise a matrix ‘length(z)‘ by ‘length(k)‘.
filter	see [wavelet_filter()].
levels	dyadic resolution of the table (default 10, i.e. 1024 points per unit interval).

Value

numeric vector or matrix of values.

Examples

```
z <- seq(0, 1, by = 0.01)
plot(z, phi_jk(z, 3, 2), type = "l")
```

`plot.fpa_density` *Plot an estimated private-value density*

Description

Plot an estimated private-value density

Usage

```
## S3 method for class 'fpa_density'
plot(x, true = NULL, rug = TRUE, ...)
```

Arguments

`x`	an ‘"fpa_density"‘ object.
`true`	optional true density function for simulated data.
`rug`	logical; add a rug of the pseudo-values.
`...`	passed to [graphics::plot()].

Value

No return value, called for its side effect of drawing the estimated density (with an optional rug of pseudo-values and the true density). The object ‘`x`‘ is returned invisibly.

`plot.fpa_density_ci` *Plot a GPV density estimate with confidence bands*

Description

Plot a GPV density estimate with confidence bands

Usage

```
## S3 method for class 'fpa_density_ci'
plot(x, true = NULL, ...)
```

Arguments

`x`	an ‘"fpa_density_ci"‘ object.
`true`	optional true density function for simulated data.
`...`	passed to [graphics::plot()].

Value

No return value, called for its side effect of drawing the density estimate with its pointwise confidence interval and uniform confidence band. The object ‘x‘ is returned invisibly.

---

plot.fpa_revenue_boot *Plot a revenue curve with bootstrap bands*

Description

Plot a revenue curve with bootstrap bands

Usage

```
## S3 method for class 'fpa_revenue_boot'
plot(x, alpha = 0.05, ...)
```

Arguments

x	an ‘"fpa_revenue_boot"‘ object.
alpha	which level to draw.
...	passed to [graphics::plot()].

Value

No return value, called for its side effect of drawing the estimated expected-revenue curve with its pointwise confidence interval (dashed) and uniform confidence band (shaded). The object ‘x‘ is returned invisibly.

---

plot.fpa_values *Plot recovered private values against bids*

Description

Plot recovered private values against bids

Usage

```
## S3 method for class 'fpa_values'
plot(x, true = NULL, ...)
```

Arguments

x	an ‘"fpa_values"‘ object.
true	optional true values (vector or matrix in the layout of ‘bids‘) for simulated data.
...	passed to [graphics::plot()].

Value

No return value, called for its side effect of plotting the recovered pseudo-values against the bids (with the monotone fit and the true values when available). The object ‘x‘ is returned invisibly.

`plot.qdf` *Plot a quantile density estimate*

Description

Plot a quantile density estimate

Usage

```
## S3 method for class 'qdf'
plot(x, true = NULL, ...)
```

Arguments

`x`	a ‘"qdf"‘ object.
`true`	optional function of ‘u‘ giving the true quantile density, drawn for comparison.
`...`	further arguments passed to [graphics::plot()].

Value

No return value, called for its side effect of drawing the estimated quantile density (and the true one, if supplied). The object ‘x‘ is returned invisibly.

`predict.qdf` *Predict from a fitted quantile density estimate*

Description

Predict from a fitted quantile density estimate

Usage

```
## S3 method for class 'qdf'
predict(object, u = NULL, ...)
```

Arguments

`object`	a ‘"qdf"‘ object.
`u`	evaluation points in $[0, 1]$; defaults to the grid stored in ‘object‘.
`...`	unused.

Value

numeric vector of $\hat{q}(u)$.

`proc_check_winning_bid` *Winning-bid check for the dependence specification*

Description

Compares the model's expected winning bid without a ceiling with the sample mean of the auction minima. Under the correct dependence specification the two agree up to sampling error.

Usage

```
proc_check_winning_bid(pv, bids)
```

Arguments

`pv`	A `proc_values` object fitted to a bid matrix.
`bids`	The same bid matrix.

Value

Named vector: `model`, `observed`, `difference`.

`proc_cost` *Expected procurement cost with a price ceiling*

Description

$C(r) = M - R'(M - r; M - v_0)$, where R' is the quantile-density revenue functional (`fpa_revenue_qd`) on the reflected fit. The plug-in `fpa_revenue` is not used: in procurement the unreliable tail of the inversion is the winners' tail.

Usage

```
proc_cost(pv, r = Inf, v0, ...)
```

Arguments

`pv`	A `proc_values` object fitted to a bid matrix.
`r`	Price ceiling(s) on the bid scale; `Inf` for no ceiling.
`v0`	Buyer's cost if no bid is accepted (outside option).
`...`	Passed to `fpa_revenue_qd`.

Value

Numeric vector of expected costs, one per element of `r`.

`proc_cost_boot` *Bootstrap standard errors for expected procurement cost*

Description

Bootstrap standard errors for expected procurement cost

Usage

```
proc_cost_boot(
  bids,
  r = Inf,
  v0,
  B = 499,
  method = "bernstein",
  bandwidth = "sqrt",
  family = "independence",
  resample = c("auctions", "blocks"),
  L = "auto",
  ...
)
```

Arguments

`bids`	$L \times n$ matrix of bids (one row per auction), or a vector with `n` given.
`r`	Price ceiling(s); `Inf` for no ceiling.
`v0`	Buyer's outside option.
`B`	Bootstrap replications.
`method`	First-stage quantile density estimator, passed to `proc_values`.
`bandwidth`	First-stage bandwidth rule or numeric bandwidth. Whatever is given, the bandwidth is selected once on the full sample and held fixed in every bootstrap refit.
`family`	Dependence specification: `"independence"`, `"frank"` or `"nonparametric"` (see `proc_values`).
`resample`	`"auctions"` (i.i.d. auction bootstrap, default) or `"blocks"` (moving-block bootstrap over consecutive auctions; use when the same firms bid repeatedly over time and the rows are in time order).
`L`	Block length for `"blocks"`: `"auto"` (the rule of `block_length_auto`, applied with `winner = "min"`) or an integer. `L = 1` is the i.i.d. auction bootstrap.
`...`	Passed to `fpa_values` (`method`, `bandwidth`, `family`, `theta`, `eta`, ...). `family` must be `"independence"`, `"frank"` or `"nonparametric"`.

Value

List: `r`, `estimate`, `se`, `ci90`, `boot`, `h`, `resample`, `L`.

`proc_M` *Reflection constant for procurement bids*

Description

Returns $M = \max b + \delta \operatorname{range}(b)$. With this choice the reflected sample $M - b$ starts at about zero, so the $Q_n(0) = 0$ anchor of the series quantile density estimators (`"bernstein"`, `"poisson"`, `"indirect_poisson"`) is correct to $O(\delta)$.

Usage

```
proc_M(bids, delta = 1e-06)
```

Arguments

`bids`	Numeric vector or $L \times n$ matrix of procurement bids.
`delta`	Relative offset above the largest bid.

Value

A single number.

`proc_optimal_ceiling` *Optimal price ceiling for a procurement auction*

Description

Mirrors `fpa_optimal_reserve_qd` on the reflected fit. An interior optimum satisfies $r^* + F_c(r^*)/f_c(r^*) = v_0$. The estimate depends on the lower tail of the recovered costs; check `share_below`, the share of pseudo-costs below r^*, before interpreting it.

Usage

```
proc_optimal_ceiling(pv, v0, ...)
```

Arguments

`pv`	A `proc_values` object fitted to a bid matrix.
`v0`	Buyer's cost if no bid is accepted (outside option).
`...`	Passed to `fpa_revenue_qd`.

Value

List: `r_star`, `p_star`, `cost`, `binding`, `share_below`, and the curve `p`, `C`.

proc_simulate *Simulate equilibrium procurement bids*

Description

Simulates the reflected standard auction with value quantile function $Q'(u) = M - Q_c(1-u)$ by `fpa_simulate` and maps back. For costs uniform on [1, 2] this reproduces $b = c + (2-c)/n$ to about 10^{-6}.

Usage

```
proc_simulate(
  L,
  n,
  Qc,
  qc = NULL,
  M,
  family = "independence",
  theta = NULL,
  eta = 0
)
```

Arguments

`L`, `n`	Auctions and bidders per auction.
`Qc`	Cost quantile function.
`qc`	Derivative of `Qc` (optional).
`M`	Any constant above the cost support.
`family`, `theta`, `eta`	Passed to `fpa_simulate`.

Value

List: `costs`, `bids` ($L \times n$), `M`, `reflected`.

proc_values *Recover private costs from procurement (lowest-bid-wins) auctions*

Description

Reflects the bids, $b' = M - b$, calls `fpa_values` on the reflected sample, and maps the result back: costs $c = M - v'$, ranks $u = 1 - u'$, quantile density $q(u) = q'(1-u)$. This implements $c = b - (1-\eta)(1-u)q(u)/(n-1)$ under independence and the copula analogue otherwise.

Usage

```
proc_values(bids, n = NULL, M = NULL, monotone = "isotonic", ...)
```

Arguments

bids	$L \times n$ matrix of bids (one row per auction), or a vector with n given.
n	Number of bidders; taken from ncol(bids) for a matrix.
M	Reflection constant; default proc_M(bids).
monotone	Monotonisation of the pseudo-values; "isotonic" by default, because the spline replacement extrapolates into the lower (winning) tail of procurement bids.
...	Passed to fpa_values (method, bandwidth, family, theta, eta, ...). family must be "independence", "frank" or "nonparametric".

Value

An object of class "proc_values": bids (ascending), u, q, psi, markup ($b-c$), costs_raw, costs, costs_mat (for matrix input), M, n, and reflected, the fpa_values fit on $M - b$ used by the cost functionals.

---

psi_jk *Mother wavelet and periodised wavelet basis on the unit interval*

Description

'psi_jk()' evaluates the mother wavelet $\psi_{j,k}(z) = 2^{j/2}\psi(2^j z - k)$, with ψ obtained from the scaling function by the two-scale relation $\psi(x) = \sqrt{2}\sum_k g_k \phi(2x - k)$, $g_k = (-1)^k h_{N-k}$. 'phi_per()' and 'psi_per()' evaluate the periodised functions $\sum_l \phi_{j,k}(z + l)$ and $\sum_l \psi_{j,k}(z + l)$, which for $2^j \geq N$ form an orthonormal basis of $L_2([0,1])$ (Daubechies 1992, Section 9.3); they are used by the nonlinear estimators [qdf_wavelet_nl()] and 'fpa_density(method = "wavelet_theory")'.

Usage

```
psi_jk(z, j, k, filter = "db3")

phi_per(z, j, k, filter = "db3")

psi_per(z, j, k, filter = "db3")
```

Arguments

z	numeric vector in the unit interval.
j	resolution level.
k	shift(s); for more than one, a matrix 'length(z)' by 'length(k)'.
filter	scaling filter; see [wavelet_filter()].

Value

numeric vector or matrix.

Examples

```
z <- seq(0, 1, length.out = 2001)
P <- psi_per(z, 4, 0:15, "db3")
round(crossprod(P)[1:3, 1:3] / 2000, 3)     # approximately the identity
```

---

qdf *Nonparametric estimation of a quantile density function*

Description

Estimates the quantile density function $q(u) = Q'(u) = 1/g(Q(u))$ (Parzen 1979; also called the sparsity function, Tukey 1965) from an i.i.d. sample, with the smoothing parameter chosen by cross-validation. Ten estimators are available; see [qdf_kernel()], [qdf_poisson()], [qdf_jones()] and [qdf_wavelet()] for their definitions and the simulation evidence in Doosti, Dewan and Talebian (2025), where the indirect Poisson estimator with BCV bandwidth performed best for first-price auction inversion.

Usage

```
qdf(
  x,
 method = c("indirect_poisson", "kernel", "kernel_corrected", "poisson", "bernstein",
    "jones", "soni", "wavelet", "wavelet_hard", "wavelet_block"),
  bandwidth = c("bcv", "rlcv", "wbcv", "sqrt", "cv"),
  m_range = NULL,
  u = NULL,
  H = NULL,
  kernel = "gaussian",
  j0 = NULL,
  loo = NULL,
  zero_at_origin = TRUE,
  grid = NULL,
  u_scheme = NULL,
  optimizer = NULL,
  rescale = TRUE,
  ...
)
```

Arguments

x	numeric sample. The series estimators (‘"poisson"‘, ‘"bernstein"‘, ‘"indirect_poisson"‘) require strictly positive data.

`method` one of ‘"indirect_poisson"‘ (default), ‘"kernel"‘, ‘"kernel_corrected"‘, ‘"poisson"‘, ‘"bernstein"‘, ‘"jones"‘, ‘"soni"‘, or the wavelet estimators ‘"wavelet"‘ (linear), ‘"wavelet_hard"‘, ‘"wavelet_block"‘ of [qdf_wavelet()], all with local-linear smoothing.

`bandwidth` either a selector name (‘"bcv"‘, the default, or ‘"rlcv"‘; ‘"wbcv"‘ is kept only for backward compatibility: it is not scale-invariant, is not recommended and warns, see [bcv_criterion()]; for the wavelet methods ‘"cv"‘, the default for them, see [select_wavelet()]), the rule ‘"sqrt"‘, or a numeric bandwidth h to use directly. ‘"sqrt"‘, for ‘method = "bernstein"‘ only, sets $m = \lceil\sqrt{n}\rceil$, the rule used in Doosti (2026b): it lies inside the range $n^{2/5} \ll m \ll n^{2/3}$ covered by that paper’s asymptotic theory and was calibrated there over the designs of Zincenko (2024).

`m_range` search range for $m = 1/h$ when a selector is used. Defaults to ‘c(10, 50)‘ (‘c(1, 50)‘ for ‘"jones"‘ and ‘"soni"‘), the ranges used in the published simulations.

`u` evaluation grid in $[0, 1]$. Defaults to $i/(n + 1)$.

`H` second bandwidth for ‘method = "soni"‘ (the outer smoother on the probability scale). Defaults to the Sheather–Jones bandwidth of the uniform scores, ‘bw.SJ(rank(x)/n)‘.

`kernel` kernel for the kernel-type estimators (‘"gaussian"‘, ‘"triangular"‘, ‘"epanechnikov"‘); see [qdf_kernel()].

`j0` resolution level for the wavelet methods (default 5 when a numeric ‘h‘ is given; chosen by cross-validation with ‘bandwidth = "cv"‘).

`loo`, `grid`, `u_scheme`, `optimizer` passed to [select_bandwidth()].

`zero_at_origin` logical (default ‘TRUE‘), for ‘method = "bernstein"‘ only: anchor the fitted quantile function at $Q_n(0) = 0$. The default reproduces earlier versions. ‘FALSE‘ anchors at the smallest observation instead and is preferable when the data do not start near zero (for example bids in dollars); it changes the fit only in the lowest few percent of quantiles. Ignored by the other estimators.

`rescale` logical (default ‘TRUE‘): estimate on x/s, s the sample standard deviation, and return $s\,\hat{q}$. This makes every estimator scale-equivariant; it matters for the indirect Poisson, Jones and Soni estimators, whose smoothing parameter lives on the data scale (for them the reported ‘h‘ refers to the standardised data). The kernel, corrected kernel, Poisson and Bernstein estimators are equivariant anyway. ‘FALSE‘ reproduces the original implementations.

`...` currently unused.

Value

An object of class ‘"qdf"‘: a list with components

x the sorted sample

u, q the evaluation grid and the estimated quantile density on it

Q the empirical quantile function on ‘u‘ (for convenience)

method, h, m, H the estimator and its smoothing parameters

selection the output of [select_bandwidth()], or ‘NULL‘ when a numeric bandwidth was supplied

with methods [predict.qdf()], ‘print()‘ and [plot.qdf()].

References

Doosti, H., Dewan, I. and Talebian, M. (2025). Nonparametric estimation of private value distributions in first-price auctions: Evaluating quantile density function approaches. *Economics Letters*, 257, 112670. doi:10.1016/j.econlet.2025.112670

Parzen, E. (1979). Nonparametric statistical data modeling. *Journal of the American Statistical Association*, 74, 105–121.

Examples

```
set.seed(1)
x <- rgamma(80, 5, 1)
fit <- qdf(x, method = "bernstein", bandwidth = "bcv")
fit
predict(fit, u = c(0.1, 0.5, 0.9))
# compare with the truth 1 / dgamma(qgamma(u, 5), 5)
plot(fit, true = function(u) 1 / dgamma(qgamma(u, 5, 1), 5, 1))
```

`qdf_jones` *Indirect estimators of the quantile density*

Description

Estimators that exploit $q(u) = 1/g(Q(u))$ by first estimating the density g.

* ‘qdf_jones()‘ (Jones 1992; estimator 5) is $\hat{q}(u) = 1/\hat{g}_h(Q_n(u))$ with a Gaussian kernel density estimator $\hat{g}_h$. * ‘qdf_soni()‘ (Soni, Dewan and Jain 2012; estimator 6) double-smooths Jones’ estimator on the probability scale. With ‘form = "integrated"‘ (the form printed in Doosti, Dewan and Talebian 2025),

$$\hat{q}(u) = \sum_{i=1}^{n} \frac{K((i/n - u)/H) - K(((i-1)/n - u)/H)}{\hat{g}_h(X_{(i)})},$$

where K is the kernel’s distribution function; with ‘form = "riemann"‘ (the original implementation of Soni, Dewan and Jain)

$$\hat{q}(u) = \frac{1}{nH} \sum_{i=1}^{n} \frac{k((i/n - u)/H)}{\hat{g}_h(X_{(i)})}.$$

Here h is the density bandwidth on the data scale and H the smoothing bandwidth on the probability scale. For right-censored data supply ‘status‘; the empirical scores i/n are then replaced by the Kaplan–Meier estimate $\hat{F}(X_{(i)})$, as in Soni et al. * ‘qdf_indirect_poisson()‘ (Chaubey, Dewan and Li 2024; estimator 7) smooths the empirical *distribution* function with Poisson weights, inverts the result numerically to obtain a smooth quantile estimator, and takes the reciprocal of the implied density at that quantile. It requires positive data.

Usage

```
qdf_jones(u, x, h, kernel = "gaussian")

qdf_soni(
  u,
  x,
  h,
  H,
  kernel = "gaussian",
  form = c("integrated", "riemann"),
  status = NULL
)

qdf_indirect_poisson(
  u,
  x,
  h,
  tol = .Machine$double.eps^0.25,
  invert = c("auto", "newton", "uniroot")
)
```

Arguments

u	numeric vector of evaluation points in $[0, 1]$.
x	numeric sample (sorted internally).
h	for ‘qdf_jones()‘ and ‘qdf_soni()‘ the kernel density bandwidth on the data scale; for ‘qdf_indirect_poisson()‘ the smoothing parameter $1/m$ of the Poisson weights on the data scale.
kernel	‘"gaussian"‘ (default), ‘"triangular"‘, ‘"epanechnikov"‘, or a user-defined kernel from [make_kernel()].
H	for ‘qdf_soni()‘, the bandwidth of the outer Gaussian smoother on the probability scale.
form	for ‘qdf_soni()‘: ‘"integrated"‘ (default) or ‘"riemann"‘.
status	for ‘qdf_soni()‘: optional 0/1 vector (1 = observed, 0 = right-censored) aligned with ‘x‘.
tol	root-finding tolerance passed to [stats::uniroot()] when inverting the smoothed distribution function.
invert	how the smoothed distribution function is inverted: ‘"newton"‘ tabulates it on a grid and refines by Newton steps (efficient for many evaluation points); ‘"uniroot"‘ reproduces the original implementation with [stats::uniroot()] (efficient for a few points; accuracy governed by ‘tol‘); ‘"auto"‘ (default) picks by ‘length(u)‘.

Value

numeric vector of quantile density estimates at ‘u‘.

References

Soni, P., Dewan, I. and Jain, K. (2012). Nonparametric estimation of quantile density function. *Computational Statistics & Data Analysis*, 56, 3876–3886.

Examples

```
x <- rgamma(100, 5, 1)
u <- c(0.25, 0.5, 0.75)
qdf_jones(u, x, h = 0.5)
qdf_soni(u, x, h = 0.5, H = 0.05)
qdf_indirect_poisson(u, x, h = 0.05)
```

---

qdf_kernel *Direct kernel estimators of the quantile density*

Description

Kernel-type estimators that smooth the empirical quantile function directly.

* ‘qdf_kernel()‘ is the Jones (1992) estimator

$$\hat{q}(u) = \sum_{i=1}^{n} X_{(i)} \{k_h((i-1)/n - u) - k_h(i/n - u)\}$$

with a Gaussian kernel $k_h(a) = k(a/h)/h$. * ‘qdf_kernel_corrected()‘ is the boundary-corrected version (estimator 2 of Doosti, Dewan and Talebian 2025), which renormalises the kernel weights by the mass falling inside $[0, 1]$ and adds a correction proportional to the kernel quantile estimator.

Usage

```
qdf_kernel(u, x, h, kernel = "gaussian")

qdf_kernel_corrected(u, x, h, kernel = "gaussian")
```

Arguments

u	numeric vector of evaluation points in $[0, 1]$.
x	numeric sample (sorted internally).
h	bandwidth on the probability scale ($0 < h < 1$).
kernel	‘"gaussian"‘ (default), ‘"triangular"‘ or ‘"epanechnikov"‘, or a user-defined kernel from [make_kernel()] (or a kernel density function). Soni, Dewan and Jain (2012) used the triangular and Epanechnikov kernels.

Value

numeric vector of quantile density estimates at ‘u‘.

References

Jones, M. C. (1992). Estimating densities, quantiles, quantile densities and density quantiles. *Annals of the Institute of Statistical Mathematics*, 44, 721–727.

Chaubey, Y. P., Dewan, I. and Li, J. (2024). On some non parametric estimators of the quantile density function for a stationary associated process. *Communications in Statistics – Theory and Methods*, 53, 5553–5573.

See Also

[qdf()] for automatic bandwidth selection.

Examples

```
x <- rgamma(100, 5, 1)
u <- seq(0.05, 0.95, by = 0.05)
qdf_kernel(u, x, h = 0.05)
qdf_kernel_corrected(u, x, h = 0.05)
```

`qdf_poisson` *Series estimators of the quantile density*

Description

Estimators that smooth the empirical quantile function Q_n with probability weights on a lattice k/m, $m = \lfloor 1/h \rfloor$.

* ‘qdf_poisson()‘ uses Poisson weights (Chaubey, Dewan and Li 2021; estimator 3 of Doosti, Dewan and Talebian 2025):

$$\hat{q}(u) = m \sum_{k=0}^{m} Q_n(k/m)\, w'_k(mu)$$

where $w_k(\mu) = p_k(\mu)/P_m(\mu)$ are truncated Poisson weights. * ‘qdf_bernstein()‘ uses Bernstein polynomials (estimator 4):

$$\hat{q}(u) = m \sum_{k=0}^{m-1} \{Q_n((k+1)/m) - Q_n(k/m)\} b_k(u; m-1).$$

Both require non-negative data because $Q_n(0)$ is taken to be 0; ‘qdf_bernstein()‘ can instead use $Q_n(0) = X_{(1)}$ (‘zero_at_origin = FALSE‘), which is preferable when the data do not start near zero (for example bids in dollars).

‘qdf_bernstein_loo()‘ returns the exact leave-one-out Bernstein estimates $\hat{q}_{-i}(u_i)$, $i = 1, \ldots, n$, used by the cross-validation criteria: removing $X_{(i)}$ shifts every higher order statistic down by one position, so the leave-one-out lattice values are obtained by an index shift in $O(nK)$ operations, without refitting.

Usage

```
qdf_poisson(u, x, h)

qdf_bernstein(u, x, h, zero_at_origin = TRUE)

qdf_bernstein_loo(u, x, h, zero_at_origin = TRUE)
```

Arguments

u	numeric vector of evaluation points in $[0, 1]$.
x	numeric sample (sorted internally).
h	bandwidth on the probability scale ($0 < h < 1$).
zero_at_origin	logical; 'TRUE' (default) takes $Q_n(0) = 0$, 'FALSE' takes $Q_n(0) = X_{(1)}$.

Value

numeric vector of quantile density estimates at 'u'; for 'qdf_bernstein_loo()', the vector $(\hat{q}_{-1}(u_1), \ldots, \hat{q}_{-n}(u_n))$, where 'u' has the length of 'x' and is paired with the sorted sample.

References

Chaubey, Y. P., Dewan, I. and Li, J. (2021). On some smooth estimators of the quantile function for a stationary associated process. *Sankhya B*, 83 (Suppl 1), 114–139. doi:10.1007/s13571020-00242x

Examples

```
x <- rgamma(100, 5, 1)
qdf_poisson(c(0.25, 0.5, 0.75), x, h = 0.05)
qdf_bernstein(c(0.25, 0.5, 0.75), x, h = 0.05)
# exact leave-one-out estimates at u_i = i/n agree with refitting
xs <- sort(x); u <- seq_along(xs) / length(xs)
loo <- qdf_bernstein_loo(u, xs, h = 0.05)
all.equal(loo[10], qdf_bernstein(u[10], xs[-10], h = 0.05))
```

---

qdf_wavelet *Wavelet estimators of the quantile density*

Description

Projection estimators of $q(u)$ on a wavelet basis of $[0, 1]$ (Chesneau, Dewan and Doosti 2016; Shirazi and Doosti 2022). The scaling coefficients are estimated by the warped-basis formula

$$\hat{c}_{j_0,k} = \int \phi_{j_0,k}(\hat{F}(x))\, dx = \sum_{i=1}^{n-1} (X_{(i+1)} - X_{(i)})\, \phi_{j_0,k}(i/n),$$

which is exact for the empirical distribution function $\hat{F}$ (‘integration = "exact"‘); ‘"simpson"‘ reproduces the composite Simpson rule of the original MATLAB implementation.

Three estimators are available through ‘threshold‘: * ‘"none"‘: the linear estimator $\hat{g}_L = \sum_k \hat{c}_{j_0,k}\phi_{j_0,k}$; * ‘"hard"‘: term-by-term hard thresholding of a ‘levels‘-level periodic DWT of the coefficient vector; * ‘"block"‘: Cai’s BlockJS block thresholding (Shirazi and Doosti 2022).

With ‘smooth = TRUE‘ the estimate is post-smoothed by the local-linear smoother of Ramirez and Vidakovic (2010) with bandwidth ‘h‘; this "smoothed" version was the best performer in both papers’ simulations.

Usage

```
qdf_wavelet(
  u,
  x,
  j0 = 5L,
  h = 0.15,
  threshold = c("none", "hard", "block"),
  smooth = TRUE,
  filter = "db3",
  dwt_filter = "coif1",
  levels = 3L,
  integration = c("exact", "simpson"),
  ...
)
```

Arguments

u	numeric vector of evaluation points in $[0, 1]$.
x	numeric sample (sorted internally).
j0	coarsest resolution level; the estimate uses 2^{j_0} scaling coefficients. Default 5.
h	bandwidth of the local-linear post-smoother on the probability scale (default 0.15); ignored when ‘smooth = FALSE‘.
threshold	‘"none"‘, ‘"hard"‘ or ‘"block"‘.
smooth	logical; apply local-linear smoothing.
filter	scaling filter for the basis; see [wavelet_filter()]. Default ‘"db3"‘.
dwt_filter	filter for the secondary DWT used in thresholding (default ‘"coif1"‘).
levels	number of DWT levels for thresholding (default 3).
integration	‘"exact"‘ or ‘"simpson"‘.
...	passed to [threshold_coefficients()] (‘const‘, ‘block_length‘, ‘rule‘).

Value

numeric vector of quantile density estimates at ‘u‘.

References

Chesneau, C., Dewan, I. and Doosti, H. (2016). Nonparametric estimation of a quantile density function by wavelet methods. *Computational Statistics & Data Analysis*, 94, 161–174.

Shirazi, E. and Doosti, H. (2022). Nonparametric estimation of a quantile density function under L_p risk via block thresholding method. *Communications in Statistics – Simulation and Computation*, 51, 539–553.

Ramirez, P. and Vidakovic, B. (2010). Wavelet density estimation for stratified size-biased sample. *Journal of Statistical Planning and Inference*, 140, 419–432.

Examples

```
set.seed(1)
x <- rbeta(500, 0.5, 0.5)
u <- seq(0.02, 0.98, by = 0.02)
q_true <- 1 / dbeta(qbeta(u, 0.5, 0.5), 0.5, 0.5)
q_lin  <- qdf_wavelet(u, x, j0 = 5, smooth = FALSE)
q_blk  <- qdf_wavelet(u, x, j0 = 5, threshold = "block", smooth = TRUE)
plot(u, q_true, type = "l"); lines(u, q_lin, col = 2); lines(u, q_blk, col = 4)
```

---

qdf_wavelet_nl *Nonlinear wavelet estimators of the quantile density with theoretical thresholds*

Description

Hard-thresholding and block-thresholding estimators of the quantile density built on the wavelet coefficients of the quantile density itself,

$$\hat{c}_{\tau,k} = \int \phi_{\tau,k}(\hat{F}(x))\,dx, \qquad \hat{d}_{j,k} = \int \psi_{j,k}(\hat{F}(x))\,dx,$$

computed exactly for the empirical distribution function, on a periodised orthonormal basis of $[0,1]$:

$$\hat{q}(u) = \sum_k \hat{c}_{\tau,k}\phi_{\tau,k}(u) + \sum_{j=\tau}^{j_1}\sum_k \hat{d}_{j,k}\,\mathrm{keep}_{j,k}\,\psi_{j,k}(u).$$

* ‘rule = "hard_general"‘: hard thresholding with $\lambda_j = K2^{3j/2}\sqrt{2p\ln n/n}$, $K = \sup|\psi'|$ and $2^{j_1} \approx (n/\ln n)^{1/4}$ (Chesneau, Dewan and Doosti 2016, Theorem 3.2, no smoothness condition beyond the Besov ball); * ‘rule = "hard_lipschitz"‘: hard thresholding with $\lambda_j = \kappa\sqrt{\ln n/n}$ and $2^{j_1} \approx \sqrt{n/\ln n}$ (Theorem 3.4, under a Lipschitz-1/2 condition on q); * ‘rule = "block"‘: block thresholding (Shirazi and Doosti 2022, eq. 6): at each level the coefficients are grouped in blocks of length $l = \lfloor \ln n \rfloor$ and a block is kept when $l^{-1}\sum_{j\in B}|\hat{d}_{j,k}|^p > c\,n^{-p/2}$, with $j_1 = \lfloor \log_2(n/l^2) \rfloor$.

The constants of the theorems are not available in practice. By default $\kappa = \hat{s}\sqrt{2n}$, so that $\lambda = \hat{s}\sqrt{2\ln n}$ (the universal threshold), and $c = 4.505^{p/2}\hat{s}^p n^{p/2}$, so that a block is kept when its mean $|\hat{d}|^p$ exceeds $4.505^{p/2}\hat{s}^p$ (the BlockJS level of Cai 1999), where $\hat{s}$ is the median absolute coefficient

at the finest level divided by 0.6745. With ‘smooth = TRUE‘ the estimate is post-smoothed by local linear regression (Ramirez and Vidakovic 2010), the "smoothed" versions of both papers.

The periodised basis replaces the boundary-corrected basis of Cohen et al. (1993) assumed by the theory; periodisation identifies the endpoints of $[0, 1]$, so near $u = 0$ and $u = 1$ these estimators should be read with care. Local-linear post-smoothing (‘smooth = TRUE‘) helps when q is small at both ends of $[0, 1]$, but when q is large at an endpoint it spreads the error caused by periodisation into the interior, and the unsmoothed estimators are then more accurate. In simulations the cross-validated estimators of [qdf()] were more accurate than all of these rules; the estimators here are mainly intended for studying the theory.

Usage

```
qdf_wavelet_nl(
  u,
  x,
  rule = c("hard_lipschitz", "hard_general", "block"),
  tau = 3L,
  j1 = NULL,
  const = NULL,
  p = 2,
  smooth = FALSE,
  h = 0.15,
  filter = "db3"
)
```

Arguments

u	numeric vector of evaluation points in $[0, 1]$.
x	numeric sample (sorted internally).
rule	‘"hard_general"‘, ‘"hard_lipschitz"‘ or ‘"block"‘.
tau	coarsest level (default 3; must satisfy $2^\tau \geq$ filter length).
j1	finest level; default from the rule as above.
const	κ for ‘"hard_lipschitz"‘, or c for ‘"block"‘; ‘NULL‘ for the data-driven defaults above.
p	risk exponent: enters λ_j for ‘"hard_general"‘ and the block statistic for ‘"block"‘ (default 2).
smooth	logical; local linear post-smoothing.
h	bandwidth of the post-smoother.
filter	scaling filter; see [wavelet_filter()].

Value

numeric vector of estimates at ‘u‘, with attribute ‘"kept"‘, the number of detail coefficients retained at each level.

References

Chesneau, C., Dewan, I. and Doosti, H. (2016). *Computational Statistics & Data Analysis*, 94, 161–174.

Shirazi, E. and Doosti, H. (2022). *Communications in Statistics – Simulation and Computation*, 51, 539–553.

Examples

```
set.seed(1)
x <- rbeta(500, 0.5, 0.5)
u <- seq(0.05, 0.95, by = 0.05)
q_true <- 1 / dbeta(qbeta(u, 0.5, 0.5), 0.5, 0.5)
cbind(q_true,
      lipschitz = qdf_wavelet_nl(u, x, "hard_lipschitz", smooth = TRUE),
      block     = qdf_wavelet_nl(u, x, "block", smooth = TRUE))
```

`rcopula_archimedean` *Simulate exchangeable Archimedean copula scores*

Description

Marshall–Olkin frailty samplers in base R.

Usage

```
rcopula_archimedean(L, n, family = "independence", theta = NULL)
```

Arguments

`L`	number of auctions (rows).
`n`	number of bidders (columns).
`family`	‘"independence"‘, ‘"clayton"‘, ‘"gumbel"‘ or ‘"frank"‘.
`theta`	copula parameter (ignored for independence).

Value

an ‘L‘ by ‘n‘ matrix with uniform margins.

Examples

```
U <- rcopula_archimedean(500, 2, "frank", theta = 5)
cor(U, method = "kendall")[1, 2]        # about 0.46
```

resample_blocks *Circular moving-block resample of auction indices*

Description

Circular moving-block resample of auction indices

Usage

```
resample_blocks(TT, L)
```

Arguments

TT	number of auctions.
L	block length; 1 reproduces the i.i.d. auction bootstrap.

Value

integer vector of length TT of row indices.

select_bandwidth *Select the smoothing parameter of a quantile density estimator*

Description

Minimises one of the criteria in [bcv_criterion()] over $m = 1/h$ in ‘m_range‘ with [stats::optimize()], optionally after a coarse grid search.

Usage

```
select_bandwidth(
  x,
  fun,
  selector = c("bcv", "rlcv", "wbcv"),
  m_range = c(10, 50),
  loo = NULL,
  grid = NULL,
  u_scheme = NULL,
  optimizer = NULL
)
```

Arguments

x	numeric sample (sorted internally).
fun	an estimator with signature ‘fun(u, x, h)‘, e.g. [qdf_kernel()]. Extra arguments (such as ‘H‘ for [qdf_soni()]) should be bound with a closure before calling.
selector	‘"bcv"‘, ‘"rlcv"‘ or ‘"wbcv"‘.
m_range	numeric length-2 range for $m = 1/h$.
loo	logical; if ‘TRUE‘ (default) leave-one-out fits are computed exactly (by refitting without observation i, or by the index shift of [qdf_bernstein_loo()] for the Bernstein estimator). If ‘FALSE‘ the scaling approximation $\hat{q}_{-i} \approx \hat{q}\,(n-1)/n$ is used. The approximation is not valid for estimators built from order statistics: removing $X_{(i)}$ shifts every higher quantile position, and with it least-squares criteria select the roughest fit in ‘m_range‘. It is kept only for backward compatibility.
grid	search method: ‘0‘ (the default, as in the published simulations) runs [stats::optimize()] directly over ‘m_range‘; an integer ‘grid > 0‘ first evaluates the criterion on that many equally spaced points and then runs ‘optimize()‘ in the bracket around the grid minimum; ‘"exhaustive"‘ evaluates the criterion at every integer ‘m‘ in ‘m_range‘ and returns the global minimiser. When a criterion is multimodal in ‘m‘, ‘optimize()‘ can return an interior local minimum, while the global minimiser can lie at an end of ‘m_range‘; boundary solutions are flagged (see below). ‘NULL‘ is the same as ‘0‘.
u_scheme	grid on which the criterion is evaluated: ‘"i/n"‘ (the published convention) or ‘"i/(n+1)"‘.
optimizer	optional function ‘optimizer(f, interval)‘ used instead of [stats::optimize()]; it must return a list with elements ‘minimum‘ and ‘objective‘. Useful to plug in a global optimiser for criteria with several local minima, e.g. ‘function(f, interval) { g <- seq(interval[1], interval[2], length.out = 200); v <- sapply(g, f); list(minimum = g[which.min(v)], objective = min(v)) }‘ for an exhaustive grid search. When ‘optimizer‘ is supplied it is used, and ‘grid = "exhaustive"‘ is treated as ‘grid = 0‘.

Value

a list with ‘h‘, ‘m‘, ‘selector‘, ‘criterion‘ (the minimised value) and ‘m_range‘; ‘at_boundary‘ (logical) and ‘boundary‘ (‘"lower"‘, ‘"upper"‘ or ‘NA‘), which record a minimiser at an end of ‘m_range‘; with ‘grid = "exhaustive"‘ also the evaluated grid ‘grid_m‘ and criterion values ‘grid_cv‘.

Boundary solutions

If the minimiser lies at either end of ‘m_range‘, whatever the search method, the criterion did not select a bandwidth in that range. A warning is issued and ‘at_boundary = TRUE‘ is returned; widen ‘m_range‘ or give the bandwidth directly.

Examples

```
x <- rgamma(60, 5, 1)
select_bandwidth(x, qdf_poisson, "bcv", m_range = c(10, 50))
```

`select_fpa_density` *Cross-validated choice of the wavelet resolution level and bandwidth*

Description

Chooses ‘j0‘ (and ‘h‘ for ‘"wavelet_ll"‘) by ‘folds‘-fold cross-validation on the bids: the first stage and the unit map are fitted on the training bids and applied to the held-out bids, and the second-stage density $\hat{f}_{-k}$ is scored on the held-out pseudo-values by one of two criteria: * ‘"lscv"‘ (default): least-squares cross-validation (Rudemo 1982; Bowman 1984), $\sum_k\{\int \hat{f}^2_{-k} - 2\,\bar{\hat{f}}_{-k}(\text{held-out})\}$, an unbiased estimate of the integrated squared error up to a constant, so that the selected pair targets MISE; * ‘"loglik"‘: the held-out log-likelihood $\sum_k \sum_i \log \hat{f}_{-k}(\hat{v}_i)$, which targets Kullback–Leibler divergence. Because the held-out points are pseudo-values carrying first-stage error, this criterion tends to favour the smallest bandwidth on the grid; it is kept for comparison.

Both criteria are evaluated on the interior $[\delta, 1-\delta]$ of the unit interval (‘trim‘): pseudo-values that fall outside $[0, 1]$ are projected onto the boundary (Algorithm 1 of the paper), which creates a spurious point mass there that a small bandwidth reproduces; excluding a boundary strip from the score removes this artefact, in the spirit of the boundary trimming of Guerre, Perrigne and Vuong (2000). For ‘"lscv"‘ the trimmed criterion is $\int_\delta^{1-\delta} \hat{f}^2 - 2n_k^{-1}\sum_{\hat{v}_i\in[\delta,1-\delta]} \hat{f}(\hat{v}_i)$, an unbiased estimate of the ISE over the interior up to a constant.

The returned score is oriented so that larger is better in both cases (‘-LSCV‘ for ‘"lscv"‘).

Usage

```
select_fpa_density(
  bids,
  n,
  method = "wavelet_ll",
  first_stage = "gpv_hh",
  j0_grid = 3:6,
  h_grid = c(0.03, 0.05, 0.08, 0.12, 0.2, 0.3, 0.5),
  folds = 5L,
  criterion = c("lscv", "loglik"),
  scale = "auto",
  filter = "db3",
  dwt_filter = "coif1",
  levels = 3L,
  kernel = "triweight",
  side = "lower",
  n_int = 2001L,
  trim = 0.05
)
```

Arguments

`bids`	pooled bids (vector) or an ‘L‘ by ‘n‘ matrix.
`n`	bidders per auction (from ‘ncol(bids)‘ for a matrix).

`method`	second-stage estimator; see Description.
`first_stage`	‘"gpv_hh"‘ (default), ‘"gpv"‘, an [fpa_values()] object, or the name of a [qdf()] method to use through [fpa_values()].
`j0_grid`, `h_grid`	candidate values.
`folds`	number of folds.
`criterion`	‘"lscv"‘ or ‘"loglik"‘.
`scale`	‘"auto"‘, ‘"log"‘, ‘"minmax"‘ or ‘"none"‘.
`filter`, `dwt_filter`, `levels`	wavelet settings; see [qdf_wavelet()].
`kernel`, `side`	kernel and correction side for [kde_hh()] (used in the ‘"gpv_hh"‘ first stage and the ‘"kde_hh"‘ second stage).
`n_int`	number of points for the numerical integral of $\hat{f}^2$.
`trim`	boundary strip excluded from the score on the unit scale (default 0.05; 0 scores the whole interval).

Value

a list with ‘j0‘, ‘h‘, ‘criterion‘ and the matrix of summed scores ‘ll‘ (rows ‘j0_grid‘, columns ‘h_grid‘; larger is better).

Reproducibility

Fold assignment (or bootstrap resampling) uses R’s random number generator; call [set.seed()] beforehand to make results exactly reproducible.

References

Rudemo, M. (1982). Empirical choice of histograms and kernel density estimators. *Scandinavian Journal of Statistics*, 9, 65–78.

Bowman, A. W. (1984). An alternative method of cross-validation for the smoothing of density estimates. *Biometrika*, 71, 353–360.

`select_wavelet` *Cross-validated choice of (j0, h) for the wavelet estimator*

Description

Chooses the resolution level and the post-smoothing bandwidth of [qdf_wavelet()] by ‘folds‘-fold cross-validation. The implied density of the data is $\hat{g}(x) = 1/\hat{q}(\hat{G}_{\mathrm{train}}(x))$. With ‘criterion = "lscv"‘ (default) the pair minimises the least-squares criterion $\int \hat{g}^2 - 2n_k^{-1}\sum \hat{g}(x_i)$ over held-out observations, using $\int \hat{g}(x)^2 dx = \int_0^1 1/\hat{q}(u)\,du$; this targets integrated squared error (Rudemo 1982; Bowman 1984). With ‘criterion = "loglik"‘ it maximises the held-out log-likelihood. Pairs with h below the lattice spacing $1/(2^{j_0}-1)$ are not considered, since the local-linear smoother is then ill-conditioned.

Usage

```
select_wavelet(
  x,
  j0_grid = 3:6,
  h_grid = c(0.03, 0.05, 0.08, 0.12, 0.2, 0.3),
  folds = 5L,
  criterion = c("lscv", "loglik"),
  ...
)
```

Arguments

x	numeric sample.
j0_grid, h_grid	candidate values.
folds	number of folds.
criterion	‘"lscv"‘ or ‘"loglik"‘.
...	passed to [qdf_wavelet()] (e.g. ‘threshold‘, ‘smooth‘, ‘filter‘).

Value

a list with ‘j0‘, ‘h‘, and the matrix of criterion values (‘ll‘, rows ‘j0_grid‘, columns ‘h_grid‘).

Reproducibility

Fold assignment (or bootstrap resampling) uses R’s random number generator; call [set.seed()] beforehand to make results exactly reproducible.

Examples

```
set.seed(2)
x <- rbeta(300, 0.5, 0.5)
select_wavelet(x, j0_grid = 4:6, h_grid = c(0.05, 0.15), folds = 3)
```

---

threshold_coefficients

Threshold wavelet coefficients

Description

Applies a thresholding rule to the detail coefficients of a periodic DWT.

* ‘"hard"‘: term-by-term hard thresholding, $\hat{d}\,1\{|\hat{d}| > \lambda\}$, with $\lambda = c\,\hat{\sigma}\sqrt{\log(n)/n}$ (Chesneau, Dewan and Doosti 2016; $c = 8$ in their implementation). * ‘"block"‘: the BlockJS rule of Cai (1999) used by Shirazi and Doosti (2022): at each level the coefficients are grouped in blocks of length $L = \lfloor \log n \rfloor$; a block is kept if $\sum_{j \in B} \hat{d}_j^2 \geq \lambda^* L \hat{\sigma}^2$ with $\lambda^* = 4.50524$ (‘rule = "truncate"‘), or shrunk by $(1 - \lambda^* L \hat{\sigma}^2 / S_B^2)_+$ (‘rule = "james-stein"‘).

$\hat{\sigma}$ is the MAD of the finest-level coefficients.

Usage

```
threshold_coefficients(
  w,
  L,
  n,
  type = c("hard", "block"),
  const = NULL,
  block_length = NULL,
  rule = c("truncate", "james-stein")
)
```

Arguments

w	coefficient vector from [dwt_periodic()].
L	number of DWT levels used to produce ‘w‘.
n	sample size (enters the threshold).
type	‘"hard"‘ or ‘"block"‘.
const	threshold constant: ‘c‘ for ‘"hard"‘, λ^* for ‘"block"‘.
block_length	block length for ‘"block"‘; default $\lfloor \log n \rfloor$.
rule	for ‘"block"‘: ‘"truncate"‘ (keep/kill) or ‘"james-stein"‘.

Value

thresholded coefficient vector.

References

Cai, T. T. (1999). Adaptive wavelet estimation: a block thresholding and oracle inequality approach. *Annals of Statistics*, 27, 898–924.

---

timber *USFS timber sealed-bid auctions (homogenised bids)*

Description

Homogenised per-unit bids (dollars per unit of timber) in the sealed-bid subsample of the U.S. Forest Service timber auction data assembled by Lu and Perrigne (2008) and deposited in the Journal of Applied Econometrics Data Archive, split by the number of bidders. The bids are **not** the raw bids: observed lot heterogeneity has been removed as described in Details. The announced reserve price is treated as non-binding in the literature using these data.

Usage

```
timber
```

Format

A list with two numeric matrices, one row per auction and one column per bidder, in the order in which the archive files record them:

n2 107 auctions with 2 bidders (columns ‘bid1‘, ‘bid2‘)

n3 108 auctions with 3 bidders (columns ‘bid1‘, ‘bid2‘, ‘bid3‘)

Details

Each auction in the archive reports the per-unit bid of every bidder together with two lot characteristics: the appraised per-unit value and the timber volume. Observed heterogeneity in these characteristics is removed before the structural analysis, as is standard for these data (Lu and Perrigne 2008; Haile, Hong and Shum 2003), so that the methods of this package can be applied to bids drawn from a common distribution within each stratum.

Within each stratum ($n = 2$ and $n = 3$ separately), the log bid is regressed by ordinary least squares on $X = (1, \log(\text{appraisal}), \log(\text{volume}))$, which is constant within an auction, and every bid is rescaled to the average lot,

$$\tilde{b} = b \exp(\bar{X}'\hat{\beta} - X'\hat{\beta}).$$

The intercept cancels in the difference, so only the slope coefficients matter. The regressions fit the log bids with $R^2 = 0.671$ for $n = 2$ (slopes 0.642 and 0.043) and $R^2 = 0.554$ for $n = 3$ (slopes 0.553 and 0.070). The stored values are rounded to six decimal places.

The script that rebuilds the object from the archive files is ‘data-raw/timber.R‘ of the package sources; the archive files themselves are not redistributed here.

Source

Journal of Applied Econometrics Data Archive, replication files of Lu, J. and Perrigne, I. (2008). Estimating risk aversion from ascending and sealed-bid auctions: the case of timber auction data. *Journal of Applied Econometrics*, 23, 871–896. doi:10.1002/jae.1032; data archive entry doi:10.15456/jae.2022319.0720356598. Please cite that paper when using the data.

References

Haile, P. A., Hong, H. and Shum, M. (2003). Nonparametric tests for common values at first-price sealed-bid auctions. Cowles Foundation Discussion Paper 1445.

Examples

```
data(timber)
fit <- fpa_values(timber$n3, method = "bernstein", bandwidth = "bcv")
summary(fit$values - fit$bids)     # implied markups
```

`wavelet_filter` *Orthonormal wavelet filters*

Description

Returns the scaling (low-pass) filter of a compactly supported orthonormal wavelet. Available: ‘"haar"‘, ‘"db2"‘, ‘"db3"‘, ‘"db4"‘ (Daubechies with 2–4 vanishing moments), ‘"sym4"‘ (Symmlet 8-tap), ‘"coif1"‘, ‘"coif2"‘ (Coiflets). A numeric vector is returned unchanged after checking that it sums to $\sqrt{2}$.

Usage

```
wavelet_filter(filter = "db3")
```

Arguments

`filter`	character name or numeric filter.

Value

numeric filter coefficients.

Examples

```
wavelet_filter("db3")
```

Index

∗ **datasets**
timber, 65

bcv_criterion, 4
bid_icc, 6
block_length_auto, 6, *45*

check_kernel *(*make_kernel*)*, 39
copula_delta *(*copula_psi*)*, 7
copula_delta_prime *(*copula_psi*)*, 7
copula_fit, 7
copula_psi, 7
copula_psi_np, 8
copula_spec_test, 9
copula_tau_to_theta, 10
copula_theta_to_tau
*(*copula_tau_to_theta*)*, 10

density_marmer_shneyerov, 11
dwt_periodic, 12

fpa_bid_function, 13
fpa_block_boot, 14
fpa_corner_test, 15
fpa_density, 17
fpa_density_boot, 19
fpa_density_ci, 21
fpa_density_cv, 23
fpa_evaluate_density
*(*fpa_evaluate_values*)*, 24
fpa_evaluate_values, 24
fpa_montecarlo, 25
fpa_optimal_reserve, 26
fpa_optimal_reserve_model
*(*fpa_revenue_model*)*, 29
fpa_optimal_reserve_qd, *46*
fpa_optimal_reserve_qd
*(*fpa_revenue_qd*)*, 31
fpa_quantile_boot_blocks
*(*fpa_block_boot*)*, 14
fpa_revenue, 27, *44*
fpa_revenue_boot, 28
fpa_revenue_boot_blocks
*(*fpa_block_boot*)*, 14
fpa_revenue_model, 29
fpa_revenue_qd, 31, *44*
fpa_simulate, 33, *47*
fpa_values, 34, *45*, *47*, *48*
fpa_virtual_value
*(*fpa_virtual_value_model*)*, 36
fpa_virtual_value_model, 36

idwt_periodic *(*dwt_periodic*)*, 12

kde_hh, 38

loclin_smooth, 39

make_kernel, 39

phi_jk, 40
phi_per *(*psi_jk*)*, 48
plot.fpa_density, 41
plot.fpa_density_boot
*(*fpa_density_boot*)*, 19
plot.fpa_density_ci, 41
plot.fpa_revenue_boot, 42
plot.fpa_values, 42
plot.qdf, 43
predict.qdf, 43
print.fpa_corner_test
*(*fpa_corner_test*)*, 15
proc_check_winning_bid, 44
proc_cost, 44
proc_cost_boot, 45
proc_M, 46
proc_optimal_ceiling, 46
proc_simulate, 47
proc_values, *45*, 47
psi_jk, 48
psi_per *(*psi_jk*)*, 48

qdf, 49
qdf_bernstein *(*qdf_poisson*)*, 54
qdf_bernstein_loo *(*qdf_poisson*)*, 54
qdf_indirect_poisson *(*qdf_jones*)*, 51
qdf_jones, 51
qdf_kernel, 53
qdf_kernel_corrected *(*qdf_kernel*)*, 53
qdf_poisson, 54
qdf_soni *(*qdf_jones*)*, 51
qdf_wavelet, 55
qdf_wavelet_nl, 57

rcopula_archimedean, 59
resample_blocks, 60
rlcv_criterion *(*bcv_criterion*)*, 4

select_bandwidth, 60
select_fpa_density, 62
select_wavelet, 63

threshold_coefficients, 64
timber, 65

wavelet_filter, 67
wbcv_criterion *(*bcv_criterion*)*, 4