

Package ‘aersn’

October 7, 2026

Type Package

Title Affine-Equivariant Adjusted-Range Self-Normalization for Time-Series Inference

Version 0.2.3

Description Tuning-free inference on fixed-dimensional parameters of dependent time series using affine-equivariant adjusted-range self-normalization. The centered partial-sum path of estimated influence contributions is normalized by its increment hull, the convex hull of all path increments. The gauge of the hull provides an asymptotically pivotal test statistic and an affine-equivariant confidence region without estimating the long-run covariance matrix, and its support function gives simultaneous confidence intervals for linear contrasts. For a single parameter the construction reduces exactly to adjusted-range self-normalization, whose limiting distribution is available in closed form. The Brownian reference law is simulated on a grid matched to the sample size or a supplied common variance-accumulation profile; inference for dependent observations remains asymptotic. Five further methods are provided for comparison on the same estimate and influence contributions: componentwise adjusted ranges after lag-zero partial prewhitening, quadratic self-normalization following Shao (2010) [doi:10.1111/j.1467-9868.2009.00737.x](https://doi.org/10.1111/j.1467-9868.2009.00737.x), kernel long-run covariance estimation with automatic bandwidth selection following Andrews (1991) [doi:10.2307/2938229](https://doi.org/10.2307/2938229) and Newey and West (1994) [doi:10.2307/2297912](https://doi.org/10.2307/2297912), Bartlett fixed-b inference following Kiefer and Vogelsang (2005) [doi:10.1017/S0266466605050565](https://doi.org/10.1017/S0266466605050565), and the equal-weighted cosine method of Lazarus, Lewis, Stock and Watson (2018) [doi:10.1080/07350015.2018.1506926](https://doi.org/10.1080/07350015.2018.1506926). Model interfaces are provided for sample means, linear regression, smooth generalized method of moments, and conditional likelihood scores; other estimators are handled through user-supplied influence contributions. The methods follow Hong, Lin, Linton, Newey and Sun (2026), Cambridge Working Papers in Economics No. 2678 <https://www.janeway.econ.cam.ac.uk/publication/affine-equivariant-adjusted-range-self-normalization>

and, for the scalar case, Hong, Linton,
McCabe, Sun and Wang (2024) <doi:10.1016/j.jeconom.2023.105603>.

License MIT + file LICENSE

URL [https://www.janeway.econ.cam.ac.uk/publication/
affine-equivariant-adjusted-range-self-normalization](https://www.janeway.econ.cam.ac.uk/publication/affine-equivariant-adjusted-range-self-normalization)

Encoding UTF-8

Depends R (>= 4.1.0)

Imports grDevices, graphics, sandwich (>= 3.0.0), lpSolve, stats,
utils

Suggests knitr, rmarkdown, testthat (>= 3.2.0)

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Yongmiao Hong [aut],
Zhuo Lin [aut],
Oliver Linton [aut],
Whitney K. Newey [aut],
Jiajing Sun [aut, cre]

Maintainer Jiajing Sun <jiajing.sun@gmail.com>

Repository CRAN

Date/Publication 2026-10-07 08:10:12 UTC

Contents

aersn	3
aersn_clear_cache	5
aersn_compare	5
aersn_contains	7
aersn_contrast	8
aersn_ewc_lrv	9
aersn_fixed_b_normalizer	11
aersn_gauge	12
aersn_gmm	13
aersn_hac_lrv	15
aersn_hull	17
aersn_hull_normalizer	19
aersn_ldl	19
aersn_ldl_normalizer	20
aersn_lm	22
aersn_mean	23
aersn_methods	24

aersn_mle	25
aersn_normalizer	26
aersn_opg_profile	27
aersn_parametric_reference	28
aersn_path	29
aersn_profile	30
aersn_projection	31
aersn_reference	32
aersn_region	34
aersn_registry	36
aersn_scalar_cdf	37
aersn_shao_normalizer	38
aersn_support	40
aersn_table_reference	40
aersn_target	41
aersn_test	42
aersn_vertices	44
confint.aersn	44
plot.aersn_region	46
reference-queries	47
Index	49

aersn	<i>Adjusted-range self-normalized inference from an estimate and its influence contributions</i>
-------	--

Description

The central constructor of the package. Given a parameter estimate and the observation-level influence contributions of its asymptotically linear representation, it builds the centered influence path, the increment-hull self-normalizer and the diagnostics needed for tests ([aersn_test\(\)](#)), simultaneous confidence intervals ([confint.aersn\(\)](#), [aersn_contrast\(\)](#)) and confidence regions ([aersn_region\(\)](#)). The model interfaces [aersn_mean\(\)](#), [aersn_lm\(\)](#), [aersn_gmm\(\)](#) and [aersn_mle\(\)](#) compute the contributions and call this function.

Usage

```
aersn(
  estimate,
  psi,
  n = NROW(psi),
  profile = NULL,
  names = NULL,
  model = NULL,
  call = NULL
)
```

Arguments

estimate	Numeric vector of length q: the estimate $\hat{\theta}_n$ of the parameters of interest.
psi	Numeric vector or $n \times q$ matrix of estimated influence contributions $\hat{\psi}_{n,t}$, observations in rows, in time order, columns ordered as estimate.
n	Number of observations; must equal <code>nrow(psi)</code> . It is the sample size of the asymptotically linear representation and enters the $n^{-1/2}$ path scaling and the $\sqrt{n}$ scaling of the estimation error.
profile	NULL for calendar-time centering, or a variance-accumulation profile as accepted by <code>aersn_profile()</code> . See that help page for the additional conditions required.
names	Optional parameter names.
model	Optional list describing the estimator (used for printing); set by the model interfaces.
call	The call to record.

Details

Conventions. Parameters are ordered as in `estimate`; column j of `psi` is the contribution to parameter j . The representation assumed is manuscript equation (asymptotic-linear-representation),

$$\sqrt{n}(\hat{\theta}_n - \theta_0) = n^{-1/2} \sum_{t=1}^n \psi_t + o_p(1),$$

with $n = \text{nrow}(\text{psi})$ observations in time order. The centered path is formed by `aersn_path()`: partial sums of the contributions scaled by $n^{-1/2}$ and centered by $\tau(k/n)$ times the full-sample sum, with $\tau(r) = r$ under calendar-time centering. The statistic for a candidate value v is $T_n(v) = \gamma_{K(\hat{G}_n)}\{\sqrt{n}(\hat{\theta}_n - v)\}$ and the level- $(1 - \alpha)$ confidence region is $\hat{\theta}_n + (c_{q,1-\alpha}^{(n)}/\sqrt{n})K(\hat{G}_n)$, where $c_{q,1-\alpha}^{(n)}$ is the matched-grid critical value from `aersn_reference()`.

Nuisance parameters and target transformations. When the parameter of interest is a function $h(\beta)$ of a larger estimated vector, the contributions must already incorporate the first-order effect of estimating the nuisance coordinates: for smooth GMM, $\hat{\psi}_{n,t} = -\hat{h}_n \hat{M}_n m_t(\hat{\beta}_n)$, and for conditional likelihood $\hat{\psi}_{n,t} = \hat{h}_n \hat{J}_n^{-1} s_t(\hat{\beta}_n)$ (manuscript Appendix A). Discarding nuisance coordinates from raw moment contributions or scores does not give the correct path; use `aersn_gmm()`, `aersn_mle()` or `aersn_target()`.

What the package does not check. Validity requires (i) a functional central limit theorem for the partial sums of ψ_t with a nonsingular scale matrix, (ii) the asymptotically linear representation above, and (iii) Hausdorff convergence of the feasible increment hull to that of the infeasible centered path (manuscript Assumption 1). Uniform convergence of the feasible path is sufficient for (iii). These conditions hold for the built-in interfaces under the stated assumptions; for user-supplied contributions they are the user's responsibility. Under strong persistence the finite-sample null rejection rate can exceed the nominal level (manuscript Section 5).

Value

An object of class "aersn": a list with elements `estimate`, `n`, `q`, `names`, `psi`, `path` (an `aersn_path()`), `hull` (an `aersn_hull()`), `centering`, `nodes`, `model` and `call`.

References

Hong, Lin, Linton, Newey and Sun (2026), Sections 2-3 and Appendix A.

See Also

[aersn_test\(\)](#), [confint.aersn\(\)](#), [aersn_contrast\(\)](#), [aersn_region\(\)](#), [aersn_target\(\)](#), [aersn_reference\(\)](#).

Examples

```
## Synthetic example: mean of a bivariate AR(1) series
set.seed(10)
n <- 200; e <- matrix(rnorm(2 * n), n, 2); Y <- e
for (t in 2:n) Y[t, ] <- 0.4 * Y[t - 1, ] + e[t, ]
fit <- aersn(colMeans(Y), sweep(Y, 2, colMeans(Y)), names = c("m1", "m2"))
fit
aersn_test(fit, null = c(0, 0), draws = 500)           # small draws: example only
confint(fit, draws = 500)
```

aersn_clear_cache	<i>Clear the session cache of reference distributions</i>
-------------------	---

Description

Clear the session cache of reference distributions

Usage

```
aersn_clear_cache()
```

Value

Invisibly, the number of cached objects removed.

aersn_compare	<i>Compare inference methods on one estimate</i>
---------------	--

Description

Applies several methods to the same estimate, the same influence contributions, the same null value or linear restriction and the same confidence level, and collects the results in one table. Each row reports the statistic, its critical value, a p-value, the reference law and the tuning values actually used, so that differences between methods can be attributed to the normalizer rather than to a difference in setup.

Usage

```
aersn_compare(
  object,
  null = NULL,
  contrast = NULL,
  level = 0.95,
  methods = aersn_methods(),
  settings = list(),
  draws = NULL,
  seed = 1L,
  cores = 1L,
  verbose = FALSE
)
```

Arguments

object	An aersn() fit.
null	Null value for the point-null test; defaults to zeros.
contrast	Optional m by q matrix of full row rank. When supplied the comparison is of the restriction <code>contrast %*% theta = null</code> and <code>null</code> has length m .
level	Confidence level.
methods	Method identifiers; see aersn_methods() . The default is all six, reported in the order increment hull, LDL componentwise, quadratic self-normalization, HAC, Bartlett fixed-b, EWC.
settings	A named list of per-method tuning arguments, for example <code>list(hac = list(kernel = "Parzen", bandwidth = "andrews"), fixedb = list(b = 0.5), ewc = list(nu = 16))</code> .
draws, seed, cores	Settings for any simulated reference law.
verbose	Report progress while reference laws are simulated.

Details

Statistics are not comparable across rows as numbers. The increment-hull statistic is a gauge, homogeneous of degree one in the estimation error, while the other five are Wald statistics, homogeneous of degree two, and each is referred to its own law. The comparable quantities are the decisions, the `p-values` and the interval widths, which the `reject`, `p_value` and `half_width` columns report. The `half_width` column is the half-width of the confidence interval for the first contrast, or for the first coordinate when no contrast is given, at the stated level.

Methods that fail on the data, for example because a normalizer matrix is numerically singular, are reported with NA statistics and the reason in the `status` column rather than being dropped from the table.

Value

A data frame of class "aersn_comparison" with one row per method and columns method, label, statistic, statistic_scale, critical_value, p_value, p_mcse, reject, half_width, reference_family, reference, tuning and status. The estimate, null value and level are kept as attributes.

See Also

`aersn_test()`, `aersn_normalizer()`.

Examples

```
set.seed(20)
n <- 300
e <- matrix(rnorm(2 * n), n, 2)
Y <- e
for (t in 2:n) Y[t, ] <- 0.4 * Y[t - 1, ] + e[t, ]
fit <- aersn_mean(Y, names = c("m1", "m2"))
aersn_compare(fit, null = c(0, 0), draws = 400, seed = 1)
```

aersn_contains

Membership test for a confidence region

Description

Tests whether candidate parameter vectors belong to a confidence region, equivalently whether the point-null test does not reject them.

Usage

```
aersn_contains(region, v, tol = 1e-09)
```

Arguments

region	An <code>aersn_region()</code> object.
v	A vector of length q or a matrix with one candidate per row.
tol	Numerical tolerance on the gauge.

Value

A logical vector with attribute "gauge" holding the gauge of each candidate relative to the region (at most one inside).

aersn_contrast	<i>Simultaneous intervals for linear contrasts</i>
----------------	--

Description

Confidence intervals for the contrasts $a_k^\top \theta$, rows of a contrast matrix, obtained from the support function of the confidence region: $a_k^\top \hat{\theta}_n \pm (c/\sqrt{n}) h_{K(\hat{G}_n)}(a_k)$, where $h_K(a)$ is the adjusted range of the projected path $a^\top \hat{G}_n$ (manuscript equation (projection-interval)).

Usage

```
aersn_contrast(
  object,
  contrasts,
  level = 0.95,
  type = c("simultaneous", "joint", "marginal"),
  method = "hull",
  ...,
  nu = NULL,
  reference = NULL,
  draws = NULL,
  seed = 1L,
  cores = 1L
)
```

Arguments

object	An aersn() object.
contrasts	A vector of length q or an $m \times q$ matrix with one contrast per row. Row names are used as labels.
level	Confidence level.
type	"simultaneous" uses the q -dimensional critical value (projections of the full joint region; simultaneous over all linear contrasts); "joint" uses the m -dimensional critical value (projections of the joint region of the m specified contrasts, which must have full row rank); "marginal" uses the scalar critical value for each contrast separately.
method	The inference method: a single identifier from aersn_methods() ("hull", the default, "ldl", "shao", "hac", "fixedb", "ewc"), or a prebuilt aersn_normalizer() object.
...	Method-specific tuning arguments passed to aersn_normalizer() , for example kernel and bandwidth for method = "hac", b for method = "fixedb", nu for method = "ewc". An unrecognized argument is an error rather than a silent default.
nu	Number of cosine terms for method = "ewc". It has its own argument because nu is a prefix of null and would otherwise be captured by partial matching.

reference, draws, seed, cores

Reference settings; see [aersn_test\(\)](#). A supplied reference object must match the method, the tuning values and the dimension implied by type.

Value

A data frame with columns estimate, lower, upper, one row per contrast, with attributes type, level, critical.value, dimension and reference. For marginal inference, tuning_by_contrast records the tuning selected separately for each contrast.

Examples

```
set.seed(8)
fit <- aersn_mean(matrix(rnorm(300), 100, 3), names = c("a", "b", "c"))
A <- rbind("a - b" = c(1, -1, 0), "mean" = c(1, 1, 1) / 3)
aersn_contrast(fit, A, draws = 500)
```

aersn_ewc_lrv

Equal-weighted cosine (EWC) covariance estimate

Description

Estimates the long-run covariance matrix by averaging the outer products of the projections of the influence contributions on a cosine basis, and returns it as an [aersn_normalizer\(\)](#). EWC is a heteroskedasticity- and autocorrelation-robust (HAR) method in the fixed-smoothing class: the reference law holds the number of basis terms fixed as the sample grows, retaining the randomness of the covariance estimate through a scaled F distribution. The default number of terms increases with sample size; at each sample size the corresponding scaled F reference is used.

Usage

```
aersn_ewc_lrv(object, nu = NULL, center = TRUE)
```

Arguments

object	An aersn() fit.
nu	Number of cosine terms. The default applies the rule above.
center	Demean the influence contributions first. The estimate does not depend on this because the basis is orthogonal to the constant; the argument exists so that the convention can be stated explicitly.

Details

With ν basis terms the type II cosine functions are

$$\phi_j(t) = \sqrt{2/n} \cos\{\pi j(t - 1/2)/n\}, \quad j = 1, \dots, \nu, \quad t = 1, \dots, n,$$

which are orthonormal on the observation grid and exactly orthogonal to the constant, so demeaning the contributions does not change the estimate. The projections and the estimate are

$$\Lambda_j = \sum_{t=1}^n \phi_j(t) e_t, \quad \hat{\Omega}_{\text{EWC}} = \frac{1}{\nu} \sum_{j=1}^{\nu} \Lambda_j \Lambda_j^{\top}.$$

The statistic for $H_0 : \theta_0 = v$ is $z^{\top} \hat{\Omega}_{\text{EWC}}^{-1} z$ with $z = \sqrt{n}(\hat{\theta}_n - v)$.

Reference law. For a target of dimension m the statistic is referred to a scaled F law: $\{(\nu - m + 1)/(\nu m)\} T$ is compared with $F_{m, \nu - m + 1}$, equivalently the critical value is $\nu m/(\nu - m + 1)$ times the $F_{m, \nu - m + 1}$ quantile. This is the Hotelling form, and it requires $\nu \geq m$. For a linear restriction the dimension m of the restriction is used, not the full parameter dimension. The law is the fixed-smoothing asymptotic reference for dependent data; it is not an exact finite-sample distribution.

Number of terms. The default is $\nu = \lfloor 0.4 n^{2/3} \rfloor$, the rule used in the manuscript's comparison, following the recommendation of Lazarus, Lewis, Stock and Watson (2018). Admissible values satisfy $q \leq \nu \leq n - 1$; the upper limit is the number of cosine terms that remain orthonormal on the grid. Larger ν reduces the variance of the estimate and moves the reference law towards chi-squared, at the cost of a larger bias under strong dependence.

Value

An `aersn_normalizer()` object of class "aersn_normalizer_ewc", whose tuning records `nu`, the rule used and the centering flag.

References

Lazarus, E., Lewis, D. J., Stock, J. H. and Watson, M. W. (2018). HAR inference: recommendations for practice. *Journal of Business and Economic Statistics*, 36(4), 541-559.

See Also

`aersn_normalizer()`, `aersn_hac_lrv()`, `aersn_fixed_b_normalizer()`, `aersn_compare()`.

Examples

```
set.seed(6)
fit <- aersn_mean(matrix(rnorm(600), 300, 2))
nz <- aersn_ewc_lrv(fit)
nz$tuning$nu
aersn_test(fit, method = "ewc")
aersn_test(fit, method = "ewc", nu = 20)
```

aersn_fixed_b_normalizer

Bartlett fixed-b normalizer

Description

Estimates the long-run covariance matrix with the Bartlett kernel and a bandwidth proportional to the sample size, $M = bn$, and returns it as an [aersn_normalizer\(\)](#). Because the bandwidth fraction b is held fixed as the sample grows, the estimate does not converge to the long-run covariance matrix; the randomness that remains in the limit is carried by a nonstandard reference law that depends on b .

Usage

```
aersn_fixed_b_normalizer(object, b = 0.5)
```

Arguments

object	An aersn() fit.
b	The bandwidth fraction, in $(0, 1]$.

Details

The estimate is computed from the centered influence path through the partial-sum identity

$$\hat{\Omega}_b = \frac{2}{b_{\text{grid}}} \cdot \frac{1}{n} \sum_{k=1}^n \hat{G}_{n,k} \hat{G}_{n,k}^\top - \frac{1}{b_{\text{grid}}} \cdot \frac{1}{n} \sum_{k=0}^{n-m} (\hat{G}_{n,k} \hat{G}_{n,k+m}^\top + \hat{G}_{n,k+m} \hat{G}_{n,k}^\top),$$

which is algebraically the Bartlett estimator with bandwidth $m = \text{round}(bn)$ and weights $1 - \ell/m$ for $\ell < m$, applied to the demeaned contributions. In [aersn_hac_lrv\(\)](#) this corresponds to `lag = m - 1`.

Requested and realized b. The bandwidth in grid intervals must be an integer, so the realized fraction is $b_{\text{grid}} = m/n$, which differs from the requested b by at most $1/(2n)$. The realized value is what enters the estimate, is recorded in `tuning$b_grid`, and is the value used to key and simulate the reference law. A request with $\text{round}(bn) = 0$ is rejected rather than silently replaced.

Relation to quadratic self-normalization. At $b = 1$ we have $m = n$, the lagged term involves only the two endpoints of the path, which are zero, and the identity reduces to $\hat{\Omega}_1 = 2V_n^Q$ exactly in finite samples, where V_n^Q is the [aersn_shao_normalizer\(\)](#) matrix. The Wald statistic is therefore exactly half the quadratic self-normalized statistic, and the fixed-b reference law at $b = 1$ is exactly half the quadratic reference law, so the two tests reject on the same samples. The package tests verify both the matrix identity and the equality of the resulting decisions using common reference draws.

Other kernels. Only the Bartlett kernel is supported here, because the package supplies a matching fixed-b reference law only for that kernel. A Parzen or quadratic spectral estimate with a fixed bandwidth fraction would need its own reference law; use [aersn_hac_lrv\(\)](#) with a numeric bandwidth if a consistent chi-squared reference is intended instead.

Value

An `aersn_normalizer()` object of class "aersn_normalizer_fixedb", whose tuning records the requested `b`, the realized `b_grid`, and the integer bandwidth `m`.

References

Kiefer, N. M. and Vogelsang, T. J. (2005). A new asymptotic theory for heteroskedasticity-autocorrelation robust tests. *Econometric Theory*, 21(6), 1130-1164.

See Also

`aersn_normalizer()`, `aersn_shao_normalizer()`, `aersn_hac_lrv()`, `aersn_compare()`.

Examples

```
set.seed(5)
fit <- aersn_mean(matrix(rnorm(400), 200, 2))
nz <- aersn_fixed_b_normalizer(fit, b = 0.5)
nz$tuning$b_grid
## b = 1 gives exactly twice the quadratic self-normalizer
max(abs(aersn_fixed_b_normalizer(fit, b = 1)$matrix -
  2 * aersn_shao_normalizer(fit)$matrix))
```

aersn_gauge

Gauge of an increment hull, or the adjusted-range statistic

Description

The gauge $\gamma_K(x) = \inf\{\lambda > 0 : x \in \lambda K\}$ is the factor by which the increment hull must be enlarged to contain x . It is evaluated by the linear program of manuscript equation (gauge-lp), which uses only the $n + 1$ path values,

$$\gamma_K(x) = \max_{u, \ell} u^\top x \quad \text{subject to} \quad \ell \leq u^\top \hat{G}_{n,j} \leq \ell + 1, \quad j = 0, \dots, n.$$

For $q = 1$ the gauge is $|x|$ divided by the adjusted range of the path (exact scalar reduction, equation (scalar-reduction)).

Usage

```
aersn_gauge(x, v, ...)

## S3 method for class 'aersn_hull'
aersn_gauge(x, v, details = FALSE, ...)

## S3 method for class 'aersn'
aersn_gauge(x, v, details = FALSE, ...)

## S3 method for class 'aersn_region'
aersn_gauge(x, v, details = FALSE, ...)
```

Arguments

x	An "aersn_hull", "aersn" or "aersn_region" object.
v	For a hull, the point(s) at which to evaluate the gauge. For a fit or region, the candidate parameter value(s). A vector of length q or a matrix with one candidate per row.
...	Passed to methods.
details	If TRUE, return a list with the value, the maximizing direction u of the dual program (normalized so that the projected range in that direction is one), and solver information; only for a single candidate.

Details

Applied to an `aersn()` object with a candidate parameter value v, the function returns the test statistic $T_n(v) = \gamma_{K(\hat{G}_n)}\{\sqrt{n}(\hat{\theta}_n - v)\}$. Applied to an `aersn_region()` object it returns the gauge relative to the region, so that values at most one indicate membership.

Value

A numeric vector of gauge values, or a list when details = TRUE.

Examples

```
set.seed(4)
Y <- matrix(rnorm(200), 100, 2)
fit <- aersn_mean(Y)
aersn_gauge(fit, c(0, 0)) # statistic for H0: mean = 0
aersn_gauge(fit, c(0, 0), details = TRUE)$direction
```

aersn_gmm

Smooth generalized method of moments

Description

Adjusted-range inference for a target $\theta = h(\beta)$ of a GMM estimator $\hat{\beta}_n$ with moment contributions $m_t(\beta) \in R^{d_m}$, $d_m \geq p$. With $\hat{D} = n^{-1} \sum_t \partial m_t(\hat{\beta}_n) / \partial \beta^\top$, weighting matrix $\hat{W}$ and $\hat{M} = (\hat{D}^\top \hat{W} \hat{D})^{-1} \hat{D}^\top \hat{W}$, the influence contributions are

$$\hat{\psi}_{n,t} = -\hat{h}_n \hat{M} m_t(\hat{\beta}_n),$$

manuscript Proposition "Feasible influence paths for smooth GMM". The matrix $\hat{M}$ uses the Jacobian with respect to all coordinates of β , so the joint estimation of nuisance coefficients is retained in the contributions of the target.

Usage

```
aersn_gmm(
  moments,
  jacobian,
  estimate,
  weight = NULL,
  target = NULL,
  profile = NULL,
  names = NULL
)
```

Arguments

moments	$n \times d_m$ matrix of moment contributions $m_t(\hat{\beta}_n)$ evaluated at the estimate, in time order.
jacobian	$d_m \times p$ matrix $\hat{D}$, the average derivative of the moments with respect to β at the estimate.
estimate	The estimate $\hat{\beta}_n$ (length p).
weight	$d_m \times d_m$ positive-definite weighting matrix; NULL gives the identity (irrelevant when $d_m = p$).
target	The parameters of interest: NULL for all of beta; integer indices or names selecting coordinates; a $q \times p$ matrix of full row rank (linear targets); or a list with elements <code>h</code> (a function of beta) and <code>jacobian</code> (the $q \times p$ derivative, a matrix or a function of beta).
profile	Optional variance-accumulation profile; see aersn_profile() . For GMM the manuscript verifies feasibility only under calendar-time centering ($\tau(r) = r$).
names	Optional names for the target parameters.

Details

The estimator must (approximately) solve the first-order condition $\hat{D}^\top \hat{W} \bar{m}_n(\hat{\beta}_n) = o_p(n^{-1/2})$; the function reports each component of this scaled condition divided by the sample standard deviation of the corresponding component of $\hat{D}^\top \hat{W} m_t$. It warns when the largest absolute standardized residual exceeds `.Machine$double.eps^0.25` (about $1.2e-4$). This numerical convergence check is not a statistical test or a verification of the asymptotic assumptions. The sufficient conditions are manuscript Assumption "Sufficient conditions for smooth GMM": stationarity and ergodicity of the moment process, $\sqrt{n}$ -consistency, full column rank of D_m , consistent Jacobian and weighting matrix, smoothness of the moments, and a joint functional central limit theorem for the moments and their Jacobian. Under these conditions the feasible path is uniformly equivalent to the path based on the true influence function. This includes OLS, IV, and stacked local-projection systems; it does not cover non-smooth moments (for example quantile regression) or estimators with nonstandard rates.

Value

An [aersn\(\)](#) object for the target, with `model$foc` holding the scaled first-order condition $\sqrt{n} \hat{D}^\top \hat{W} \bar{m}_n$. `model$foc_standardized` and `model$foc_tolerance` record the standardized residuals and the numerical warning threshold.

Examples

```
## Instrumental-variables example (synthetic):  $y = 1 + 0.5 x + u$ ,
## x endogenous, instrument z.
set.seed(12)
n <- 300; z <- rnorm(n); v <- rnorm(n)
x <- z + v; u <- 0.5 * v + rnorm(n); y <- 1 + 0.5 * x + u
Zm <- cbind(1, z); Xm <- cbind(1, x)
beta <- solve(crossprod(Zm, Xm), crossprod(Zm, y)) # exactly identified IV
moments <- Zm * as.numeric(y - Xm %*% beta) # z_t (y_t - x_t' beta)
D <- -crossprod(Zm, Xm) / n
fit <- aersn_gmm(moments, D, beta, target = 2, names = "slope")
aersn_test(fit, null = 0.5, draws = 2000)
```

aersn_hac_lrv

HAC long-run covariance estimate

Description

Estimates the long-run covariance matrix of the influence contributions of an [aersn\(\)](#) fit by kernel smoothing of the sample autocovariances, and returns it as an [aersn_normalizer\(\)](#) so that tests, intervals and regions can be built from it exactly as for the other methods.

Usage

```
aersn_hac_lrv(
  object,
  kernel = c("Bartlett", "Parzen", "Quadratic Spectral"),
  bandwidth = "short",
  lag = NULL,
  prewhite = FALSE,
  adjust = FALSE,
  center = TRUE
)
```

Arguments

object	An aersn() fit.
kernel	"Bartlett", "Parzen" or "Quadratic Spectral".
bandwidth	A positive number, or one of "short", "long", "andrews", "newey-west". Ignored when lag is supplied.
lag	Optional nonnegative integer lag truncation for the Bartlett or Parzen kernel, giving bandwidth lag + 1.
prewhite	Apply VAR(1) prewhitening and recoloring.
adjust	Apply the $n/(n - q)$ finite-sample multiplier.
center	Demean the influence contributions before forming the autocovariances. For fitted contributions the column means are already zero and this has no effect.

Details

Write e_t for the influence contributions, demeaned when `center = TRUE`. With kernel weights $w_\ell = w(\ell/h)$ and bandwidth h ,

$$\hat{\Omega} = \hat{\Gamma}_0 + \sum_{\ell=1}^{n-1} w_\ell (\hat{\Gamma}_\ell + \hat{\Gamma}_\ell^\top), \quad \hat{\Gamma}_\ell = \frac{1}{n} \sum_{t=1}^{n-\ell} e_t e_{t+\ell}^\top.$$

The statistic for $H_0 : \theta_0 = v$ is $z^\top \hat{\Omega}^{-1} z$ with $z = \sqrt{n}(\hat{\theta}_n - v)$, and the reference law is χ_q^2 (or χ_m^2 for an m -dimensional linear restriction). All $n - 1$ lags are used; the Bartlett and Parzen weights vanish beyond the bandwidth, while the quadratic spectral kernel has unbounded support and is not truncated.

Kernels. "Bartlett" gives $w(x) = 1 - |x|$ for $|x| \leq 1$; "Parzen" gives $1 - 6x^2 + 6|x|^3$ for $|x| \leq 1/2$ and $2(1 - |x|)^3$ for $1/2 < |x| \leq 1$; "Quadratic Spectral" gives $3\{\sin(y)/y - \cos(y)\}/y^2$ with $y = 6\pi x/5$. These match `sandwich::kweights()`.

Bandwidth. Distinguish three quantities that are easy to confuse. The real-valued bandwidth h enters through $w(\ell/h)$. A lag truncation L for the Bartlett kernel corresponds to $h = L + 1$, since $w_\ell = 1 - \ell/(L + 1)$ is then zero at $\ell = L + 1$. The fixed- b fraction b of `aersn_fixed_b_normalizer()` is a different parameterisation again. `bandwidth` accepts:

- a positive number, used directly as h ;
- "short": $L = \lfloor 4(n/100)^{2/9} \rfloor$, $h = L + 1$, the rule used for the Newey-West comparison in the manuscript;
- "long": $L = \lceil 1.3\sqrt{n} \rceil$, $h = L + 1$;
- "andrews": the plug-in bandwidth of Andrews (1991) for the chosen kernel, from `sandwich::bwAndrews()` with equal weights on the coordinates. For the Parzen and quadratic spectral kernels the returned bandwidths satisfy $h_{\text{Parzen}} = h_{\text{QS}} \times 2.6614/1.3221$ exactly, the ratio of Andrews' kernel constants; the manuscript's replication code uses that identity and the package checks it in its tests.
- "newey-west": the lag-selection rule of Newey and West (1994), from `sandwich::bwNeweyWest()`, applied as $L = \lfloor bw \rfloor$ and $h = L + 1$. This is the convention of `sandwich::NeweyWest()`, which applies Bartlett weights; it is therefore supported for the Bartlett kernel only, and other kernels raise an error rather than silently using a different rule.

Alternatively, `lag` sets the Bartlett or Parzen lag truncation directly, giving $h = L + 1$; it is not available for the quadratic spectral kernel, whose weights do not vanish at any finite lag.

Prewhitening. `prewhite = TRUE` applies the Andrews-Monahan procedure: a VAR(1) is fitted to the contributions by least squares, the kernel estimate is formed from its residuals, and the result is recolored as $(I - \hat{A})^{-1} \hat{\Omega}_e (I - \hat{A})^{-\top}$. Bandwidth selection then uses the residuals, and the scaling divisor stays n although there are $n - 1$ residuals, which is the convention of `sandwich::vcovHAC()`. This is VAR prewhitening of the HAC estimator and is unrelated to the lag-zero LDL partial prewhitening of `aersn_ldl_normalizer()`. The default FALSE reproduces the manuscript's comparison.

Finite-sample adjustment. `adjust = TRUE` multiplies the estimate by $n/(n - q)$, where q is the target dimension, not the total number of fitted coefficients. For a selected target this can differ from `sandwich::meatHAC()` applied to the full model. The default FALSE reproduces the manuscript's comparison. The influence contributions already absorb the estimation of nuisance parameters, so this multiplier is a finite-sample convention rather than a correction derived for this framework.

Value

An `aersn_normalizer()` object of class "aersn_normalizer_hac". tuning records the kernel, the bandwidth rule, the realized numeric bandwidth, the implied Bartlett lag truncation where that is meaningful, and the prewhitening, adjustment and centering flags.

References

- Andrews, D. W. K. (1991). Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica*, 59(3), 817-858.
- Newey, W. K. and West, K. D. (1994). Automatic lag selection in covariance matrix estimation. *Review of Economic Studies*, 61(4), 631-653.
- Andrews, D. W. K. and Monahan, J. C. (1992). An improved heteroskedasticity and autocorrelation consistent covariance matrix estimator. *Econometrica*, 60(4), 953-966.

See Also

`aersn_normalizer()`, `aersn_fixed_b_normalizer()`, `aersn_ewc_lrv()`, `aersn_compare()`.

Examples

```
set.seed(4)
n <- 300
e <- matrix(rnorm(2 * n), n, 2)
Y <- e
for (t in 2:n) Y[t, ] <- 0.5 * Y[t - 1, ] + e[t, ]
fit <- aersn_mean(Y)
aersn_hac_lrv(fit, kernel = "Bartlett", bandwidth = "short")
aersn_hac_lrv(fit, kernel = "Quadratic Spectral", bandwidth = "andrews")
aersn_test(fit, method = "hac", kernel = "Parzen", bandwidth = "andrews")
```

aersn_hull

Increment-hull self-normalizer

Description

Constructs the affine-equivariant normalizing set of the manuscript: the convex hull of all increments of the centered influence path,

$$K(\hat{G}_n) = \text{conv}\{\hat{G}_n(r) - \hat{G}_n(s) : 0 \leq r, s \leq 1\},$$

manuscript equation (increment-hull). The set is compact, convex and centrally symmetric. Its support function in direction u is the adjusted range of the projected path, $h_K(u) = \max_j u^\top \hat{G}_{n,j} - \min_j u^\top \hat{G}_{n,j}$ (equation (support-range)), and its gauge has the dual representation $\gamma_K(x) = \sup_{u \neq 0} |u^\top x| / h_K(u)$ (equation (gauge-dual)). The hull is never stored as a list of pairwise increments; it is represented by the path itself, and the gauge is evaluated by the linear program of equation (gauge-lp).

Usage

```
aersn_hull(path, n_directions = 1000L)
```

Arguments

<code>path</code>	An <code>aersn_path()</code> object.
<code>n_directions</code>	Number of prespecified unit directions used for the projected-range diagnostics (in addition to the coordinate axes).

Details

The object records diagnostics recommended in the Supplement (Section "Multivariate critical values and affine transformations"): the singular values of the $(n + 1) \times q$ path matrix, its numerical rank after dividing each nonconstant coordinate by its adjusted range (threshold $\max(n + 1, q) * s_{.1} * \text{eps}$ on this scaled path), the condition number $s_{.1} / s_{.q}$, and the minimum, maximum and spread of the projected adjusted range over the coordinate axes and a fixed set of `n_directions` prespecified unit directions. If the numerical rank is below q the hull is not full dimensional and joint statistics are not evaluated (Algorithm 1, step 2). Scaling prevents units alone from causing numerical rank deficiency; unscaled diagnostics remain available as `singular_values`, `raw_rank` and `condition`. The scanned minimum range is an upper bound on the all-direction minimum.

Value

An object of class "aersn_hull" with elements `path`, `G`, `n`, `q`, `singular_values`, `rank`, `full_rank`, `condition`, `coordinate_ranges`, `min_projected_range`, `max_projected_range`, `projected_range_spread`, and `n_directions`.

References

Hong, Lin, Linton, Newey and Sun (2026), Sections 2.2-2.3 and Supplement Section "Multivariate critical values and affine transformations".

See Also

`aersn_gauge()`, `aersn_support()`, `aersn_path()`.

Examples

```
set.seed(3)
Y <- matrix(rnorm(300), 100, 3)
hull <- aersn_hull(aersn_path(sweep(Y, 2, colMeans(Y))))
hull
aersn_support(hull, c(1, 0, 0))          # adjusted range of column 1
aersn_gauge(hull, c(0.5, -0.2, 0.1))
```

aersn_hull_normalizer *Hull normalizer*

Description

Wraps the increment hull of an `aersn()` fit as an `aersn_normalizer()` object, so that the affine-equivariant adjusted-range method can be handled by the same inference and comparison functions as the other methods. The hull itself is computed by `aersn_hull()` when the fit is created; this function does not recompute it.

Usage

```
aersn_hull_normalizer(object)
```

Arguments

object An `aersn()` fit.

Value

An "aersn_normalizer" object with family = "geometric".

See Also

`aersn_normalizer()`, `aersn_hull()`.

Examples

```
set.seed(1)
fit <- aersn_mean(matrix(rnorm(200), 100, 2))
aersn_hull_normalizer(fit)
```

aersn_ldl *LDL factorization of a symmetric positive-definite matrix*

Description

Factorizes $S = LDL^\top$ with L unit lower triangular and D diagonal with positive entries. The factorization is computed from the Cholesky factor as $L = C\text{diag}(C)^{-1}$ and $D = \text{diag}(C)^2$, where C is the lower Cholesky factor, so that $C = LD^{1/2}$.

Usage

```
aersn_ldl(S, what = "matrix")
```

Arguments

S	A symmetric positive-definite matrix.
what	A label used in error messages.

Details

No ridge term, pseudo-inverse or nearest-positive-definite projection is applied. If the matrix is not numerically positive definite the function stops and reports the smallest eigenvalue, because a silently modified factorization would change the statistic without changing its name.

Value

A list with L (unit lower triangular), D (the diagonal of D , a vector), chol (the lower Cholesky factor C) and reconstruction_error, the maximum absolute deviation of $LDL^\top$ from the symmetrized input.

Examples

```
S <- crossprod(matrix(c(2, 1, 0, 1, 3, 1, 0, 1, 4), 3, 3))
f <- aersn_ldl(S)
f$L
max(abs(f$L %*% diag(f$D) %*% t(f$L) - S))
max(abs(f$chol - f$L %*% diag(sqrt(f$D))))
```

aersn_ldl_normalizer	<i>Componentwise adjusted-range normalizer after LDL partial prewhitening</i>
----------------------	---

Description

Implements the componentwise adjusted-range self-normalizer of Hong, Linton, McCabe, Sun and Wang (2024), as described in the Supplement of the package's main reference. The influence contributions are transformed by the inverse of the unit lower triangular factor of their sample lag-zero covariance, which removes contemporaneous cross dependence, and each transformed coordinate is normalized by its own adjusted range. The joint statistic is the sum of the squared componentwise ratios.

Usage

```
aersn_ldl_normalizer(object, order = NULL)
```

Arguments

object	An <code>aersn()</code> fit.
order	Optional permutation of 1:q giving the coordinate order used for the factorization. The default keeps the order of the fit. The statistic is reported in the original coordinates whichever order is used, but its value generally changes with the order.

Details

Write $\hat{\Gamma}_0$ for the sample lag-zero covariance of the influence contributions, factorized as $\hat{\Gamma}_0 = \hat{L}\hat{D}\hat{L}^\top$ with $\hat{L}$ unit lower triangular (`aersn_ldl()`). With $z = \sqrt{n}(\hat{\theta}_n - v)$, transformed error $\tilde{z} = \hat{L}^{-1}z$ and transformed path $\tilde{G}_n = \hat{L}^{-1}\hat{G}_n$, the statistic is

$$T_n^{\text{comp}}(v) = \sum_{j=1}^q \left(\frac{|\tilde{z}_j|}{\max_k \tilde{G}_{n,k,j} - \min_k \tilde{G}_{n,k,j}} \right)^2,$$

the range being taken over the observation grid including the origin. This is the Wald form $z^\top V^{-1}z$ with $V = \hat{L} \text{diag}(R_j^2) \hat{L}^\top$, where R_j is the adjusted range of transformed coordinate j , so the confidence region is an ellipsoid in the original parameter coordinates and linear contrasts are obtained from V in the usual way.

Cholesky and LDL give the same statistic. Writing the lower Cholesky factor as $\hat{C} = \hat{L}\hat{D}^{1/2}$, the transformed error becomes $\hat{D}^{-1/2}\tilde{z}$ and transformed coordinate j of the path is divided by the same $\hat{D}_{jj}^{1/2}$, so each ratio is unchanged and the diagonal scale cancels exactly. The function returns both factors and reports the difference between the two statistics as a diagnostic.

Conditions. The reference law used for this statistic treats the q componentwise ratios as independent. Diagonalizing the lag-zero covariance does not in general diagonalize the limiting long-run covariance: the independent-component law applies when $\hat{L}^{-1}\Omega\hat{L}^{-\top}$ is diagonal in the limit, where Ω is the long-run covariance of the influence contributions. The prewhitening is partial because serial and lead-lag dependence remain. The statistic depends on the order of the coordinates, and it is not affine equivariant: a rotation or shear of the parameters changes its value. Use `method = "hull"` when equivariance under general nonsingular reparameterization is required.

Value

An `aersn_normalizer()` object of class `"aersn_normalizer_ldl"`. In addition to the common fields it carries `lag_zero` (the sample lag-zero covariance in factorization order), `ldl` (the factorization in that same order), `transform` (the matrix applying the coordinate permutation followed by $\hat{L}^{-1}$ to the error and path in their original column order), `transformed_path`, `component_ranges` and, in `diagnostics`, the reconstruction error, the maximum absolute off-diagonal element of the transformed lag-zero covariance, and the Cholesky agreement check.

References

Hong, Y., Linton, O., McCabe, B., Sun, J. and Wang, S. (2024). Kolmogorov-Smirnov type testing for structural breaks: a new adjusted-range based self-normalization approach. *Journal of Econometrics*, 238(2), 105603. doi:[10.1016/j.jeconom.2023.105603](https://doi.org/10.1016/j.jeconom.2023.105603)

See Also

`aersn_normalizer()`, `aersn_ldl()`, `aersn_compare()`.

Examples

```
set.seed(2)
fit <- aersn_mean(matrix(rnorm(600), 200, 3))
nz <- aersn_ldl_normalizer(fit)
```

```

nz$ldl$L
nz$component_ranges
aersn_test(fit, method = "ldl", draws = 400, seed = 1)

```

aersn_lm

Linear regression by ordinary least squares

Description

Adjusted-range inference for regression coefficients. OLS is the exactly identified GMM estimator with moments $m_t(b) = z_t(y_t - z_t^\top b)$; its influence contributions are $\hat{\psi}_{n,t} = \hat{h}_n(\hat{Q}^{-1} z_t \hat{u}_t)$ with $\hat{Q} = n^{-1} \sum_t z_t z_t^\top$ and residuals $\hat{u}_t$ (manuscript Appendix B, OLS example). The function forms the moments and Jacobian from a fitted `stats::lm()` object and calls `aersn_gmm()`.

Usage

```
aersn_lm(object, ..., target = NULL, profile = NULL, names = NULL)
```

Arguments

object	A formula (fitted with <code>stats::lm()</code> using ...) or a fitted lm object. Rows of the model frame must be in time order.
...	Passed to <code>stats::lm()</code> when object is a formula.
target	Parameters of interest; see <code>aersn_gmm()</code> . The default is all coefficients including the intercept.
profile	Optional variance-accumulation profile; see <code>aersn_gmm()</code> .
names	Optional names for the target parameters.

Details

Weighted least squares, generalized linear models, multiple-response models and models with aliased (NA) coefficients are not supported. Observations removed by `na.action` (including `na.exclude`) are dropped before the path is formed; the function warns because the remaining rows are then treated as consecutive in time. Validity requires the conditions of `aersn_gmm()` for the moments $z_t u_t$, in particular a functional central limit theorem for their partial sums and positive-definite $E(z_t z_t^\top)$.

Value

An `aersn()` object.

Examples

```
set.seed(13)
n <- 250
x <- arima.sim(list(ar = 0.5), n)
u <- arima.sim(list(ar = 0.5), n)
y <- 1 + 0.8 * x + u
fit <- aersn_lm(y ~ x, target = "x")
aersn_test(fit, null = 0.8, draws = 2000)
fit2 <- aersn_lm(y ~ x) # both coefficients
confint(fit2, draws = 1000)
```

aersn_mean	<i>Sample mean of a scalar or vector time series</i>
------------	--

Description

Adjusted-range inference for the mean. The estimate is the sample mean and the influence contributions are the demeaned observations, $\hat{\psi}_{n,t} = Y_t - \bar{Y}_n$ (manuscript Section 2.1). The centered path is the scaled cumulative sum of the demeaned series.

Usage

```
aersn_mean(Y, profile = NULL, names = NULL)
```

Arguments

Y	Numeric vector (scalar series) or $n \times q$ matrix with observations in rows, in time order.
profile	Optional variance-accumulation profile; see aersn_profile() . Because the demeaned observations sum to zero, a profile does not change the path; it changes the reference grid. The manuscript shows that this is not the required profile-centered path for the sample mean unless the profile is linear, so a profile should be supplied here only when its use is justified separately.
names	Optional parameter names.

Details

Validity requires the functional central limit theorem for the partial sums of $Y_t - \theta_0$ with nonsingular long-run covariance (manuscript Assumption 1(i)); the representation is exact and the feasible path equals the infeasible centered path, so Assumption 1(ii)-(iii) hold automatically.

Value

An [aersn\(\)](#) object.

Examples

```
set.seed(11)
x <- arima.sim(list(ar = 0.5), 300)
fit <- aersn_mean(x)
aersn_test(fit, 0, reference = "continuous")
confint(fit, draws = 5000)
```

aersn_methods

Method identifiers

Description

The character identifiers accepted by the method argument of [aersn_test\(\)](#), [confint.aersn\(\)](#), [aersn_contrast\(\)](#), [aersn_region\(\)](#) and [aersn_compare\(\)](#), in the order used by comparison tables.

Usage

```
aersn_methods()
```

Details

- "hull": the affine-equivariant adjusted-range increment hull, the method of the package's main reference and the default everywhere.
- "ldl": componentwise adjusted ranges after lag-zero partial prewhitening by an LDL factorization ([aersn_ldl_normalizer\(\)](#)).
- "shao": quadratic self-normalization ([aersn_shao_normalizer\(\)](#)).
- "hac": kernel long-run covariance estimation ([aersn_hac_lrv\(\)](#)).
- "fixedb": Bartlett fixed-b ([aersn_fixed_b_normalizer\(\)](#)).
- "ewc": equal-weighted cosine ([aersn_ewc_lrv\(\)](#)).

"hull", "ldl" and "shao" are self-normalized: the normalizer is a random object with a nondegenerate limit and the reference law is nonstandard. "hac" uses a consistent covariance estimate with a chi-squared reference. "fixedb" and "ewc" use inconsistent covariance estimates with fixed-smoothing reference laws.

Value

A character vector of method identifiers.

Examples

```
aersn_methods()
```

aersn_mle

*Conditional likelihood scores***Description**

Adjusted-range inference for a target $\theta = h(\beta)$ of a conditional maximum-likelihood (or M-) estimator from its fitted scores and normalized negative Hessian. The influence contributions are

$$\hat{\psi}_{n,t} = \hat{h}_n \hat{J}_n^{-1} s_t(\hat{\beta}_n), \quad \hat{J}_n = n^{-1} J_{n,n}(\hat{\beta}_n),$$

manuscript Appendix A.2 (equation (fitted-score-clock) and the text following it). Because the fitted scores sum to zero, the calendar-time and profile-centered paths coincide; a profile changes only the reference grid used for matched-grid quantiles.

Usage

```
aersn_mle(
  scores,
  neg_hessian,
  estimate,
  target = NULL,
  profile = "calendar",
  names = NULL
)
```

Arguments

scores	$n \times p$ matrix of fitted conditional scores $s_t(\hat{\beta}_n)$ in time order, one column per model parameter (including nuisance parameters).
neg_hessian	$p \times p$ normalized negative Hessian $\hat{J}_n = -n^{-1} \sum_t \partial s_t(\hat{\beta}_n) / \partial \beta^\top$ (or a consistent estimate of J); it must be symmetric and positive definite.
estimate	The estimate $\hat{\beta}_n$.
target	Parameters of interest; see aersn_gmm() .
profile	"calendar" (default), "opg" for the score outer-product profile, or a known profile as accepted by aersn_profile() .
names	Optional names for the target parameters.

Details

With `profile = "opg"` the variance-accumulation profile is estimated from cumulative score outer products by [aersn_opg_profile\(\)](#). This is justified when the scores form a martingale-difference array and the cumulative conditional score variance and cumulative expected negative Hessian accumulate the same fraction $\tau(r)$ of their full-sample matrices (Supplement, Propositions "Multivariate fitted scores under a common variance-accumulation profile" and "Feasible variance-accumulation profile from score outer products"). Under correct specification and information equality the two profiles agree. For serially correlated scores the outer-product profile is not valid and calendar-time centering should be used. The proportionality measure returned in `model$proportionality` describes departures from a common scalar profile.

Value

An `aersn()` object.

Examples

```
## Gaussian linear model with known variance: the score interface
## reproduces the OLS influence contributions.
set.seed(14)
n <- 200; x <- rnorm(n); y <- 0.5 + x + rnorm(n)
X <- cbind(1, x); b <- solve(crossprod(X), crossprod(X, y))
s <- X * as.numeric(y - X %*% b)          # scores with sigma^2 = 1
J <- crossprod(X) / n
fit <- aersn_mle(s, J, b, target = 2, names = "slope")
confint(fit, draws = 4000)
fit_opg <- aersn_mle(s, J, b, target = 2, profile = "opg")
fit_opg$model$proportionality
```

`aersn_normalizer`

Construct the normalizer used by an inference method

Description

Builds the object that a method uses to normalize the estimation error: the increment hull for "hull", or a q by q positive-definite matrix for the other five methods. The inference functions call this internally; it is exported so that a normalizer can be inspected, reused across several tests, or built once and passed to `aersn_test()` and friends. A prebuilt normalizer is tied to its influence contributions and profile. For a different data set or a lower-dimensional target, pass a method name and tuning arguments to construct the normalizer for that target.

Usage

```
aersn_normalizer(object, method = "hull", ...)
```

Arguments

<code>object</code>	An <code>aersn()</code> fit.
<code>method</code>	A method identifier; see <code>aersn_methods()</code> .
<code>...</code>	Method-specific tuning arguments.

Details

Method-specific tuning arguments are passed through `...` and validated by the individual constructors:

- "hull": none.
- "ldl": order.
- "shao": integration.

- "hac": kernel, bandwidth, lag, prewhite, adjust, center.
- "fixedb": b.
- "ewc": nu, center.

Passing an unknown tuning argument is an error rather than a silent default, so that a misspelled kernel or bandwidth rule cannot be ignored.

Value

An object of class "aersn_normalizer". Common fields are method, label, family ("geometric" or "quadratic"), scale ("gauge" or "wald"), reference_family, q, n, names, matrix (the long-run covariance or quadratic normalizer, NULL for the hull), hull (for the hull method), tuning (the realized tuning values) and diagnostics.

See Also

[aersn_ldl_normalizer\(\)](#), [aersn_shao_normalizer\(\)](#), [aersn_hac_lrv\(\)](#), [aersn_fixed_b_normalizer\(\)](#), [aersn_ewc_lrv\(\)](#), [aersn_compare\(\)](#).

Examples

```
set.seed(1)
fit <- aersn_mean(matrix(rnorm(300), 150, 2))
aersn_normalizer(fit, "shao")
aersn_normalizer(fit, "hac", kernel = "Parzen", bandwidth = "andrews")
```

aersn_opg_profile

Feasible variance-accumulation profile from score outer products

Description

Estimates the variance-accumulation profile τ from fitted conditional-likelihood scores by the normalized trace of cumulative outer products, manuscript equation (opg-trace-clock):

$$\hat{\tau}_n(k/n) = p^{-1} \text{tr}(\hat{\Sigma}_{s,n}^{-1/2} \hat{\Sigma}_{s,k} \hat{\Sigma}_{s,n}^{-1/2}), \quad \hat{\Sigma}_{s,k} = n^{-1} \sum_{t \leq k} s_t s_t^\top.$$

The estimate is nondecreasing, lies in $[0, 1]$, and has endpoints zero and one whenever $\hat{\Sigma}_{s,n}$ is positive definite.

Usage

```
aersn_opg_profile(scores)
```

Arguments

scores n x p matrix of fitted conditional scores $s_t(\hat{\beta}_n)$ in time order (all model parameters, including nuisance parameters).

Details

The estimator is justified (Supplement, Proposition "Feasible variance-accumulation profile from score outer products") for martingale-difference conditional scores whose cumulative conditional variance and cumulative expected negative Hessian accumulate the same fraction $\tau(r)$ of their full-sample matrices. For serially correlated influence contributions, lag-zero outer products do not estimate cumulative long-run covariance, and this profile is not valid. The returned proportionality measure $\mathcal{E}_n = \max_k p^{-1/2} \|\hat{\Xi}_{s,k} - \hat{\tau}_n(k/n)I_p\|_F$ (manuscript equation (clock-proportionality-measure)) is descriptive: large values indicate that the accumulation is not proportional across score components, so a common scalar profile is doubtful.

Value

A numeric vector of length $n + 1$ (the nodes, with attribute "centering" = "profile"), with attributes "proportionality" ($\mathcal{E}_n$) and "score_sum" (the scaled full-sample score $n^{-1/2} \sum_t s_t$, which should be close to zero for fitted scores).

See Also

[aersn_mle\(\)](#), [aersn_profile\(\)](#).

Examples

```
set.seed(2)
s <- matrix(rnorm(400), 200, 2)      # synthetic scores
s <- sweep(s, 2, colMeans(s))        # fitted scores sum to zero
tau <- aersn_opg_profile(s)
range(diff(tau)); attr(tau, "proportionality")
```

aersn_parametric_reference

Parametric reference law for a Wald statistic

Description

Builds an [aersn_reference\(\)](#) object for the two comparator methods whose reference law is available in closed form: the chi-squared law used with a consistent HAC covariance estimate, and the scaled F law used with the equal-weighted cosine estimate.

Usage

```
aersn_parametric_reference(law = c("chisq", "ewc_F"), df, nu = NULL)
```

Arguments

law	"chisq" or "ewc_F".
df	Degrees of freedom, that is the dimension of the parameter or of the linear restriction being tested.
nu	Number of cosine terms, required for "ewc_F".

Details

For law = "chisq" the statistic is referred to χ^2_{df} . For law = "ewc_F" the critical value is $\nu df / (\nu - df + 1)$ times the $F_{df, \nu - df + 1}$ quantile, equivalently $\{(\nu - df + 1) / (\nu df)\} T \sim F_{df, \nu - df + 1}$. Both are asymptotic reference laws for dependent observations, not exact finite-sample distributions.

Value

An object of class "aersn_reference" with type = "parametric".

See Also

[aersn_reference\(\)](#), [aersn_hac_lrv\(\)](#), [aersn_ewc_lrv\(\)](#).

Examples

```
ref <- aersn_parametric_reference("chisq", df = 3)
aersn_critical_value(ref, 0.95)
aersn_parametric_reference("ewc_F", df = 2, nu = 20)
```

aersn_path	<i>Centered influence path</i>
------------	--------------------------------

Description

Computes the centered partial-sum path of estimated influence contributions on the observation grid, including the initial and final zero points. This is the object whose increment hull normalizes the estimation error.

Usage

```
aersn_path(psi, profile = NULL, names = NULL)
```

Arguments

- psi Numeric vector (scalar parameter) or n x q numeric matrix of influence contributions with observations in rows and parameters in columns, in time order.
- profile Variance-accumulation profile; see [aersn_profile\(\)](#). NULL gives calendar-time centering.
- names Optional parameter names (length q).

Details

With contributions $\hat{\psi}_{n,t} \in R^q$ in the rows of `psi` and profile nodes $\tau_k = \tau(k/n)$, the path is

$$\hat{G}_n(k/n) = n^{-1/2} \left\{ \sum_{t=1}^k \hat{\psi}_{n,t} - \tau_k \sum_{t=1}^n \hat{\psi}_{n,t} \right\}, \quad k = 0, \dots, n,$$

manuscript equations (feasible-path) and (feasible-clock-path). Under calendar-time centering $\tau_k = k/n$. The empty sum at $k = 0$ is zero, so $\hat{G}_n(0) = \hat{G}_n(1) = 0$. The scaling by $n^{-1/2}$ uses `n = nrow(psi)`, which must be the number of contributions in the asymptotically linear representation.

Centering subtracts τ_k times the full-sample sum, not the sample mean of the contributions at each date; the two coincide only under calendar-time centering. If the contributions sum to zero (fitted scores or least-squares moments), the second term vanishes and the profile does not change the path; it still changes the reference grid used by `aersn_reference()`.

Value

An object of class "aersn_path": a list with the $(n + 1) \times q$ matrix `G` of path values, `n`, `q`, the profile nodes, the centering label, and `total`, the full-sample sum of the contributions.

See Also

`aersn_hull()`, `aersn()`.

Examples

```
set.seed(1)
Y <- matrix(rnorm(60), 30, 2)
path <- aersn_path(sweep(Y, 2, colMeans(Y)))
path$G[c(1, 31), ] # zero at both ends
plot(path)
```

aersn_profile

Variance-accumulation profile on the observation grid

Description

Validates, or constructs from a function, the nodes $\tau(k/n)$, $k = 0, \dots, n$, of a variance-accumulation profile used for profile centering of the influence path. The profile must be continuous and nondecreasing from $\tau(0) = 0$ to $\tau(1) = 1$; the package only checks these properties on the grid.

Usage

```
aersn_profile(profile, n, tol = 1e-10)
```

Arguments

profile	Either NULL (calendar time, nodes k/n), a function of the sample fraction r in $[0, 1]$, or a numeric vector of length $n + 1$ giving the nodes $\tau(k/n)$ for $k = 0, \dots, n$.
n	Number of observations (the number of grid intervals).
tol	Tolerance for the endpoint and monotonicity checks, between zero and $1e-6$.

Details

Under calendar-time centering the nodes are k/n . The manuscript (Section 4, "Nonuniform Variance Accumulation") replaces k/n by $\tau(k/n)$ when the cumulative long-run covariance satisfies the fixed-shape condition $C(r) = \tau(r) C(1)$. Two additional conditions are required for a *feasible* profile-centered path (manuscript Assumption "Feasible scalar time change"):

- the estimated profile belongs to $\{(t_0, \dots, t_n) : 0 = t_0 \leq \dots \leq t_n = 1\}$ and is uniformly consistent for τ ; and
- the increment hull of the feasible profile-centered path converges in Hausdorff distance to the hull of the infeasible one.

Consistency of the profile alone does not deliver the second condition: for the sample mean, the fitted contributions sum to zero, so substituting any profile leaves the calendar-centered path unchanged, and the resulting path differs from the required one by $\{\tau(r) - r\}S_n(1)$ (Supplement, Section "Cumulative information and profile-based reindexing"). The manuscript establishes feasibility for conditional-likelihood scores whose cumulative score variance and expected negative Hessian accumulate the same fraction of their full-sample matrices; see [aersn_mle\(\)](#) and [aersn_opg_profile\(\)](#). The package does not estimate a valid common profile for arbitrary dependent data.

Value

A numeric vector of length $n + 1$ with attribute "centering" equal to "calendar" or "profile".

Examples

```
aersn_profile(NULL, 4)
aersn_profile(function(r) r^3, 4)
aersn_profile(c(0, 0.1, 0.3, 0.7, 1), 4)
```

Description

`aersn_projection()` returns the polygon of the orthogonal projection of the region onto two coordinates. Because projection is linear, the projection of $\hat{\theta}_n + sK(\hat{G}_n)$ is the same construction applied to the two-dimensional sub-path, so it is exact and has simultaneous coverage. `aersn_slice()` returns the boundary of the intersection of the region with the plane through the center in which all other coordinates are held at their estimates; each boundary point is $\hat{\theta}_n + s d / \gamma_K(d)$ for a direction d in the plane, evaluated by the gauge linear program. A slice is a cross-section, not a confidence set for the two coordinates.

Usage

```
aersn_projection(region, coords = c(1L, 2L))

aersn_slice(region, coords = c(1L, 2L), n_angles = 180L)
```

Arguments

<code>region</code>	An <code>aersn_region()</code> object.
<code>coords</code>	Integer vector of length two: the coordinates to display.
<code>n_angles</code>	Number of directions used for the slice boundary.

Value

A numeric matrix with two columns (polygon vertices in order).

<code>aersn_reference</code>	<i>Reference distribution of a self-normalized statistic</i>
------------------------------	--

Description

Returns the reference law of the statistic $T_q = \gamma_{K(B_q)}(Z_q)$, where B_q is a standard q -dimensional Brownian bridge and Z_q is an independent standard normal vector (manuscript equation (brownian-reference)). With `grid = "matched"` the bridge is evaluated on the same grid as the sample path (n intervals, or the supplied profile nodes), which gives the matched-grid reference statistic $T_q^{(n)}$ whose quantiles the manuscript uses as critical values. With `grid = "continuous"` and $q = 1$ the closed-form law of `aersn_scalar_cdf()` is used.

Usage

```
aersn_reference(
  x,
  n = NULL,
  draws = NULL,
  seed = 1L,
  grid = c("matched", "continuous"),
  nodes = NULL,
```

```

    batches = 20L,
    cores = 1L,
    cache = TRUE,
    verbose = FALSE,
    statistic = "hull_gauge",
    args = list()
)

```

Arguments

x	Either the parameter dimension q, or an aersn() object (in which case q, n and the profile nodes are taken from it).
n	Number of grid intervals (the sample size). Required for grid = "matched" unless x is a fit or nodes is supplied.
draws	Number of Monte Carlo draws; defaults to 50000 for q = 1 and 10000 otherwise.
seed	Integer seed for the reference simulation.
grid	"matched" (bridge on the sample grid) or "continuous" (closed-form law; available for q = 1 only).
nodes	Optional profile nodes (length n + 1, see aersn_profile()) defining a nonuniform reference grid.
batches	Number of interleaved batches for Monte Carlo standard errors.
cores	Number of worker processes, either 1 or 2 (forked workers on Unix-alikes; ignored on Windows). Does not affect the draws.
cache	Use and update the session cache.
verbose	Print progress messages.
statistic	The statistic whose law is simulated: "hull_gauge" (the default, the increment-hull gauge), "componentwise_mq2" (the LDL componentwise statistic), "shao_sq2" (quadratic self-normalization) or "fixedb_bartlett" (Bartlett fixed-b). The chi-squared law used with HAC estimation and the scaled F law used with the equal-weighted cosine method are closed form and are built by aersn_parametric_reference() instead.
args	A list of the tuning values that change the law of statistic: integration ("calendar" or "profile") for "shao_sq2", and b_grid together with the integer bandwidth m for "fixedb_bartlett". Empty for the other statistics. These values enter the cache key and are checked against the statistic they are used with.

Details

Each Monte Carlo draw simulates Brownian increments with variances equal to the node spacings, forms the bridge $W_q(\tau_j) - \tau_j W_q(1)$ at the nodes, and evaluates the gauge of the piecewise-linear bridge at an independent standard normal vector with the same linear program used for the sample statistic ([aersn_gauge\(\)](#)). For nested grids the finite-grid statistic decreases to the continuous-path statistic (Supplement, Lemma "Observed grids and continuous paths"), so matched-grid quantiles

exceed continuous-path quantiles. Corollary "Validity of matched-grid critical values" in the Supplement covers uniform and nonuniform deterministic grids and, when the grid is the estimated profile, conditional quantiles given the sample.

Draws are generated in fixed-size chunks with chunk-specific seeds derived from seed, using the Mersenne-Twister generator with inversion for normal variates; the user's random-number state is saved and restored. Results are therefore reproducible for a given seed, draws, q and grid, independently of cores. Monte Carlo standard errors of quantiles are estimated from batches interleaved batches of draws. Objects are cached in the session by a key that contains every setting that changes the law (statistic, q, grid nodes, draws, seed, chunk size, and the linear-program settings), together with batches. The grid must contain at least $q + 1$ positive increments. If any draw fails numerically, simulation stops without returning or caching a reference; failed draws are never discarded to estimate quantiles. Matching the grid approximates the reference law and does not make inference finite-sample exact for general dependent observations.

Computational cost. Scalar matched-grid simulation stores several matrices with n columns and up to 2000 rows per worker. Peak memory therefore grows with n and $\min(\text{draws}, 2000)$; decreasing draws while keeping it above 2000 mainly reduces runtime, not peak memory. For orientation, a run with $n = 20000$, 50000 draws and one worker used about 3 GB of memory and one minute on an arm64 Mac; requirements vary with the platform. Fewer than 2000 draws reduces peak memory but increases Monte Carlo uncertainty. For $q = 1$, `grid = "continuous"` avoids simulation when a continuous-path approximation is appropriate; it is a different reference law, not the matched-grid calculation.

Value

An object of class "aersn_reference". For Monte Carlo references the element `draws` holds the simulated statistics in generation order (and signed the signed ratios when $q = 1$). Use `quantile()`, `aersn_critical_value()`, `aersn_pvalue()` and `aersn_mcse()` to query it.

See Also

`aersn_scalar_cdf()`, `aersn_registry`.

Examples

```
ref1 <- aersn_reference(1, grid = "continuous")
quantile(ref1, c(0.90, 0.95, 0.99))
ref2 <- aersn_reference(2, n = 50, draws = 400, seed = 1)
ref2
aersn_pvalue(ref2, 2.4)
```

Description

The level- $(1-\alpha)$ confidence region $\mathcal{C}_n = \hat{\theta}_n + (c_{q,1-\alpha}^{(n)} / \sqrt{n}) K(\hat{G}_n)$ (manuscript equation (confidence-region)). The region is a translated and scaled copy of the increment hull; it is convex, centrally symmetric about the estimate, generally not an ellipsoid, and affine equivariant. It is represented by its center, scale, and hull; membership is evaluated through the gauge and projections through the support function. Explicit vertices are available for $q = 2$ ([aersn_vertices\(\)](#)), and two-dimensional projections or slices for plotting in any dimension ([plot.aersn_region\(\)](#)).

Usage

```
aersn_region(
  object,
  level = 0.95,
  method = "hull",
  ...,
  nu = NULL,
  reference = NULL,
  draws = NULL,
  seed = 1L,
  cores = 1L
)
```

Arguments

object	An aersn() object.
level	Confidence level.
method	The inference method: a single identifier from aersn_methods() ("hull", the default, "ldl", "shao", "hac", "fixedb", "ewc"), or a prebuilt aersn_normalizer() object.
...	Method-specific tuning arguments passed to aersn_normalizer() , for example kernel and bandwidth for method = "hac", b for method = "fixedb", nu for method = "ewc". An unrecognized argument is an error rather than a silent default.
nu	Number of cosine terms for method = "ewc". It has its own argument because nu is a prefix of null and would otherwise be captured by partial matching.
reference, draws, seed, cores	Reference settings; see aersn_test() . A supplied reference object must match the method, the tuning values and the dimension implied by type.

Value

An object of class "aersn_region" with elements center, scale ($c/\sqrt{n}$), critical.value, level, hull, q, n, names and reference.

See Also

[aersn_contains\(\)](#), [aersn_support\(\)](#), [aersn_gauge\(\)](#), [aersn_vertices\(\)](#), [plot.aersn_region\(\)](#).

Examples

```
set.seed(9)
fit <- aersn_mean(matrix(rnorm(400), 200, 2))
reg <- aersn_region(fit, draws = 1000)
reg
aersn_contains(reg, rbind(c(0, 0), c(0.5, 0.5)))
plot(reg)
```

aersn_registry	<i>Verified matched-grid critical values from the manuscript replication package</i>
----------------	--

Description

Tabulated matched-grid quantiles of the increment-hull gauge statistic $T_q^{(n)}$ taken from the replication package of Hong, Lin, Linton, Newey and Sun (2026), version of 8 September 2026. They are included as regression targets for the package's own simulations and as a fast tabulated reference ([aersn_test\(\)](#) with reference = "registry").

Usage

```
aersn_registry
```

Format

A data frame with one row per (dimension, grid size, probability):

q parameter dimension (1, 2, 3, 5, 10)
n number of grid intervals (200, 500, 738, 1000)
prob quantile level (0.90, 0.95, 0.975, 0.99)
cv matched-grid quantile (type-8 sample quantile of the draws)
mcse Monte Carlo standard error from 100 interleaved batches
draws number of Monte Carlo draws (2,000,000 for q = 1, 30,000 otherwise)
batches number of batches used for mcse
quantile_type quantile type (8)
seed master seed recorded in the production run manifest
source file of the replication package the row was taken from

Details

The q = 1 rows were produced by `scalar_matched_range_quadratic_cv.R` (endpoint an independent standard normal; bridge the equispaced linear interpolant; range including the zero at the origin). The q >= 2 rows were produced by `multivariate_matched_cv.R` with the same dual linear program used by [aersn_gauge\(\)](#). The production run manifests record master seeds 20260715 and 20260717 and the replication counts stored in the draws column (runs completed 15 July 2026). These tabulated simulation results are distributed under this package's MIT license.

Source

Replication package, files `core_methods/inputs/matched_scalar_critical_values.csv` and `core_methods/inputs/multivariate_matched_cv_registry.csv`.

Examples

```
subset(aersn_registry, prob == 0.95)
```

aersn_scalar_cdf	<i>Closed-form scalar adjusted-range reference law</i>
------------------	--

Description

Distribution function, quantile function and density of the continuous-path scalar reference law. Let $Z \sim N(0, 1)$ and let R be the range (supremum minus infimum) of an independent standard Brownian bridge on $[0, 1]$. The signed ratio $M = Z/R$ is the limit of the signed scalar statistic, and $|M|$ is the limit of the two-sided statistic $T_n(\theta_0)$ for $q = 1$ (manuscript Theorem 1 with $q = 1$, equation (scalar-reduction)).

Usage

```
aersn_scalar_cdf(c, method = c("auto", "bessel", "series"), tol = 1e-13)
```

```
aersn_scalar_quantile(p, tol = 1e-12)
```

```
aersn_scalar_density(u, method = c("auto", "bessel", "series"), tol = 1e-11)
```

Arguments

<code>c</code>	Nonnegative evaluation points for the distribution function of $ M $.
<code>method</code>	"auto", "bessel" or "series".
<code>tol</code>	Truncation tolerance for the series representations.
<code>p</code>	Probabilities strictly between 0 and 1.
<code>u</code>	Evaluation points for the density of the signed ratio M .

Details

The Supplement (Proposition "Density and distribution") gives

$$f_M(u) = \frac{1}{2} - 12u^2 \sum_{k \geq 1} \frac{k^2}{(u^2 + 4k^2)^{5/2}}, \quad \Pr(|M| \leq c) = c - 2c^3 \sum_{k \geq 1} (c^2 + 4k^2)^{-3/2},$$

with truncation errors at most $c^3/(8N^2)$ for the distribution function and $3U^2/(16N^2)$ for the density on $|u| \leq U$ after N terms. Poisson summation gives the exponentially convergent tail representation

$$\Pr(|M| > c) = 2\pi c^2 \sum_{k \geq 1} k K_1(\pi k c),$$

where K_1 is the modified Bessel function of the second kind (Supplement, equation (bessel-tail)), and differentiating it gives $f_M(c) = \pi^2 c^2 \sum_k k^2 K_0(\pi k c) - \pi c \sum_k k K_1(\pi k c)$. The "auto" method uses the Bessel representations except very close to zero, where the direct series is used; the two representations are cross-checked in the package tests.

The quantile function inverts the distribution function of $|M|$ with `stats::uniroot()`. Continuous-path critical values at two-sided levels 10, 5 and 1 percent are 1.397390, 1.705776 and 2.367378 (Supplement Table "Continuous-path scalar critical values").

These are continuous-path laws. For a sample of size n , the manuscript uses matched-grid quantiles, which are larger; see `aersn_reference()` with `grid = "matched"`.

Value

Numeric vectors: probabilities $\Pr(|M| \leq c)$, quantiles of $|M|$, or density values of M .

See Also

`aersn_reference()`.

Examples

```
aersn_scalar_cdf(c(1.397390, 1.705776, 2.367378))
aersn_scalar_quantile(c(0.90, 0.95, 0.99))
integrate(aersn_scalar_density, -Inf, Inf)$value # integrates to one
```

aersn_shao_normalizer *Quadratic self-normalizer*

Description

Builds the quadratic self-normalizer of Shao (2010) from the centered influence path of an `aersn()` fit. The normalizer is the discretized integral of the outer product of the path, and the statistic is the Wald form with that matrix. Like the increment hull it needs no bandwidth, kernel or block length, and its reference law is nonstandard.

Usage

```
aersn_shao_normalizer(object, integration = c("auto", "calendar", "profile"))
```

Arguments

object	An <code>aersn()</code> fit.
integration	"calendar" for equal weights $1/n$, "profile" for the increments of the fit's variance-accumulation profile. The default follows the fit: calendar-time centering gives "calendar", profile centering gives "profile".

Details

With the centered path $\hat{G}_n(k/n)$, $k = 0, \dots, n$, the calendar-time normalizer is

$$V_n^Q = \frac{1}{n} \sum_{k=1}^n \hat{G}_n(k/n) \hat{G}_n(k/n)^\top,$$

the origin contributing zero. The statistic for $H_0 : \theta_0 = v$ is $z^\top (V_n^Q)^{-1} z$ with $z = \sqrt{n}(\hat{\theta}_n - v)$, whose limit is $B_q(1)^\top \{\int_0^1 \mathbb{B}_q \mathbb{B}_q^\top\}^{-1} B_q(1)$. For $q = 1$ the two-sided statistic is the square of $|z|/\{n^{-1} \sum_k \hat{G}_n(k/n)^2\}^{1/2}$.

Integration rule. Under nonuniform variance accumulation the manuscript integrates with respect to the variance-accumulation profile rather than calendar time. With `integration = "profile"` the normalizer is

$$V_n^{Q,\tau} = \sum_{k=1}^n \hat{G}_n^\tau(k/n) \hat{G}_n^\tau(k/n)^\top \{\hat{\tau}_n(k/n) - \hat{\tau}_n((k-1)/n)\},$$

the Riemann-Stieltjes sum with the estimated profile increments as weights. With a linear profile the two rules coincide exactly. The reference law is simulated with the same rule and the same nodes, so a calendar-time statistic is never combined with a profile-weighted reference. The adjusted-range methods do not need this correction, because projected ranges are invariant under a monotone change of time.

Value

An `aersn_normalizer()` object of class "aersn_normalizer_shao", carrying the matrix V_n^Q and the integration weights actually used.

References

Shao, X. (2010). A self-normalized approach to confidence interval construction in time series. *Journal of the Royal Statistical Society: Series B*, 72(3), 343-366. doi:10.1111/j.14679868.2009.00737.x

See Also

`aersn_normalizer()`, `aersn_compare()`.

Examples

```
set.seed(3)
fit <- aersn_mean(matrix(rnorm(400), 200, 2))
nz <- aersn_shao_normalizer(fit)
nz$matrix
aersn_test(fit, method = "shao", draws = 400, seed = 1)
```

aersn_support	<i>Support function of an increment hull or confidence region</i>
---------------	---

Description

For an [aersn_hull\(\)](#) object, returns the support function $h_K(u) = \sup_{z \in K} u^\top z$, which equals the adjusted range of the projected path $u^\top \hat{G}_n$ (manuscript equation (support-range)). For an [aersn_region\(\)](#) object, returns the support function of the confidence region $u^\top \hat{\theta}_n + (c/\sqrt{n}) h_K(u)$, which is the upper end of the simultaneous projection interval for the contrast $u^\top \theta$.

Usage

```
aersn_support(x, u, ...)
```

```
## S3 method for class 'aersn_hull'
```

```
aersn_support(x, u, ...)
```

```
## S3 method for class 'aersn_region'
```

```
aersn_support(x, u, ...)
```

Arguments

x	An "aersn_hull" or "aersn_region" object.
u	A direction vector of length q, or a matrix with one direction per row.
...	Unused.

Value

A numeric vector with one value per direction.

aersn_table_reference	<i>Build a reference object from tabulated quantiles</i>
-----------------------	--

Description

Wraps tabulated critical values (for example a row set of [aersn_registry](#)) as an "aersn_reference" of type "table". Such a reference supports [aersn_critical_value\(\)](#) at the tabulated probabilities only, and [aersn_pvalue\(\)](#) returns the bracket of tabulated tail probabilities that contains the p-value.

Usage

```
aersn_table_reference(q, n, probs, values, mcse = NULL, source = "user table")
```

Arguments

q	Parameter dimension.
n	Number of grid intervals.
probs	Tabulated probabilities (upper-tail quantile levels).
values	Tabulated quantiles.
mcse	Optional Monte Carlo standard errors of the quantiles.
source	Character description of the provenance.

Value

An "aersn_reference" object of type "table".

Examples

```
reg <- subset(aersn_registry, q == 2 & n == 200)
ref <- aersn_table_reference(2, 200, reg$prob, reg$cv, reg$mcse,
                           source = "aersn_registry")
aersn_critical_value(ref, 0.95)
```

aersn_target	<i>Transform the target parameter</i>
--------------	---------------------------------------

Description

Maps an [aersn\(\)](#) object for a parameter vector β to one for a target $\theta = h(\beta)$ by transforming the estimate and the influence contributions with the Jacobian of h : $\hat{\theta}_n = h(\hat{\beta}_n)$ and $\hat{\psi}_{n,t}^\theta = \hat{h}_n \hat{\psi}_{n,t}^\beta$. For a linear transformation $A\beta$, the estimate is $A\hat{\beta}_n$ and the contributions are $A\hat{\psi}_{n,t}^\beta$; the increment hull transforms accordingly ($K(Ag) = AK(g)$). Inference from the returned object uses the target dimension's reference law. When dimension is reduced, its region generally differs from the projection of the original region, which retains the original critical value. Use [aersn_contrast\(\)](#) with `type = "simultaneous"` for that projection.

Usage

```
aersn_target(object, transform, jacobian = NULL, names = NULL)
```

Arguments

object	An "aersn" object for the full parameter vector.
transform	Either a numeric $m \times q$ matrix (linear target), or a function $h(\text{beta})$ returning the target vector.
jacobian	For a function transform, its $m \times q$ Jacobian matrix at the estimate, or a function of beta returning it.
names	Optional names for the target coordinates.

Details

The transformation must have full row rank $m \leq q$; otherwise the transformed increment hull cannot be full dimensional. The delta method underlying the function-valued case requires h to be continuously differentiable at the true parameter with a full-row-rank derivative (manuscript Proposition "Feasible influence paths for smooth GMM").

Value

An "aersn" object for the target, with the same sample size and centering.

Examples

```
set.seed(5)
fit <- aersn_mean(matrix(rnorm(300), 100, 3))
diff12 <- aersn_target(fit, matrix(c(1, -1, 0), 1, 3), names = "m1 - m2")
confint(diff12, draws = 500)
```

aersn_test

Point-null test with the increment-hull gauge

Description

Tests $H_0 : \theta_0 = v$ (or $H_0 : A\theta_0 = v$ for a full-row-rank contrast matrix A) using the statistic $T_n(v) = \gamma_{K(\hat{G}_n)}\{\sqrt{n}(\hat{\theta}_n - v)\}$ and the reference law of [aersn_reference\(\)](#). The null is rejected at level $\alpha = 1 - \text{level}$ when $T_n(v)$ exceeds the $(1 - \alpha)$ quantile of the reference law.

Usage

```
aersn_test(
  object,
  null = NULL,
  contrast = NULL,
  level = 0.95,
  alternative = c("two.sided", "greater", "less"),
  method = "hull",
  ...,
  nu = NULL,
  reference = NULL,
  draws = NULL,
  seed = 1L,
  cores = 1L
)
```

Arguments

object	An aersn() object.
null	Null value: a vector of length q (or of length <code>nrow(contrast)</code>). Defaults to zeros.
contrast	Optional $m \times q$ matrix of full row rank defining the linear restriction contrast $%% \theta = \text{null}$.
level	Confidence level $1 - \alpha$ for the reported critical value.
alternative	"two.sided", or, for $q = 1$ and <code>method = "hull"</code> , also "greater" or "less".
method	The inference method: a single identifier from aersn_methods() ("hull", the default, "ldl", "shao", "hac", "fixedb", "ewc"), or a prebuilt aersn_normalizer() object.
...	Method-specific tuning arguments passed to aersn_normalizer() , for example <code>kernel</code> and <code>bandwidth</code> for <code>method = "hac"</code> , <code>b</code> for <code>method = "fixedb"</code> , <code>nu</code> for <code>method = "ewc"</code> . An unrecognized argument is an error rather than a silent default.
nu	Number of cosine terms for <code>method = "ewc"</code> . It has its own argument because <code>nu</code> is a prefix of <code>null</code> and would otherwise be captured by partial matching.
reference	An aersn_reference() object matching the fit, the string "continuous" (closed-form law, $q = 1$ only), the string "registry" (bundled tabulated quantiles, if available for this q and n), or NULL to simulate a matched-grid reference.
draws, seed, cores	Settings passed to aersn_reference() when the reference is simulated. See its computational-cost discussion for memory requirements on long scalar series.

Details

A linear restriction $A\theta_0 = v$ is tested as a point null for the transformed target $A\theta$ (see [aersn_target\(\)](#)); nuisance coordinates are handled through the influence contributions, not by profiling. Unrestricted composite hypotheses are not supported. For $q = 1$ one-sided alternatives use the signed ratio $\sqrt{n}(\hat{\theta}_n - v)$ divided by the adjusted range, whose limit is $M = Z/R$ (Supplement, Section "Scalar specialization").

The p-value is the reference tail probability at the observed statistic. For a Monte Carlo two-sided reference it is the fraction of draws at or above the statistic. One-sided scalar tails use the symmetry of the law, consistently with the critical values. The result includes Monte Carlo uncertainty; see [aersn_pvalue\(\)](#). For the closed-form scalar law no Monte Carlo approximation is used. This is a limiting reference law, not a finite-sample exact test for arbitrary dependent observations.

Value

An object of class `c("aersn_test", "htest")` with the statistic, critical value, p-value, level, estimate, null value, the maximizing direction of the dual program, and the reference summary.

Examples

```
set.seed(6)
fit <- aersn_mean(rnorm(150, mean = 0.2))
```

```
aersn_test(fit, null = 0, reference = "continuous")
aersn_test(fit, null = 0, draws = 2000)
```

aersn_vertices	<i>Vertices of a two-dimensional confidence region</i>
----------------	--

Description

For $q = 2$, the increment hull is a polygon whose vertices are among the pairwise differences of the vertices of the convex hull of the path values ($K(g) = P(g) - P(g)$, Supplement Proposition "Characterization of the bridge-increment hull"). The region's vertices are the center plus the scaled hull vertices, in counter-clockwise order.

Usage

```
aersn_vertices(region)
```

Arguments

region An `aersn_region()` object with $q = 2$, or an `aersn_hull()` object with $q = 2$ (then the unscaled hull vertices are returned).

Value

A numeric matrix with two columns.

confint.aersn	<i>Confidence intervals from the increment hull</i>
---------------	---

Description

Confidence intervals for coordinates of the parameter vector. Three constructions are distinguished:

- "simultaneous": projections of the joint q -dimensional confidence region onto the coordinate axes, manuscript equation (projection-interval), $\hat{\theta}_j \pm (c_q/\sqrt{n})\{\max_k \hat{G}_{n,k,j} - \min_k \hat{G}_{n,k,j}\}$, using the q -dimensional critical value. These intervals hold simultaneously for every linear projection of the region.
- "joint": projections of the joint region of the selected coordinates only (dimension $m = \text{length}(\text{parm})$), using c_m . Coverage is simultaneous over linear combinations of the selected coordinates.
- "marginal": separately constructed scalar intervals, each applying the univariate adjusted-range statistic to the coordinate path with c_1 . They cover each coordinate individually and do not provide simultaneous coverage (manuscript Theorem 1(iii), last paragraph).

Usage

```
## S3 method for class 'aersn'
confint(
  object,
  parm = NULL,
  level = 0.95,
  type = c("simultaneous", "joint", "marginal"),
  method = "hull",
  ...,
  nu = NULL,
  reference = NULL,
  draws = NULL,
  seed = 1L,
  cores = 1L
)
```

Arguments

object	An aersn() object.
parm	Coordinates to report: indices or names; default all.
level	Confidence level.
type	"simultaneous", "joint" or "marginal"; see Details.
method	The inference method: a single identifier from aersn_methods() ("hull", the default, "ldl", "shao", "hac", "fixedb", "ewc"), or a prebuilt aersn_normalizer() object.
...	Method-specific tuning arguments passed to aersn_normalizer() , for example kernel and bandwidth for method = "hac", b for method = "fixedb", nu for method = "ewc". An unrecognized argument is an error rather than a silent default.
nu	Number of cosine terms for method = "ewc". It has its own argument because nu is a prefix of null and would otherwise be captured by partial matching.
reference, draws, seed, cores	Reference settings; see aersn_test() . A supplied reference object must match the method, the tuning values and the dimension implied by type.

Value

A matrix with columns lower and upper and one row per coordinate, of class "aersn_confint", with attributes type, level, critical.value, dimension (the dimension of the reference law used) and reference. For marginal inference, tuning_by_contrast records the tuning selected separately for each coordinate.

Examples

```
set.seed(7)
fit <- aersn_mean(matrix(rnorm(400), 200, 2))
confint(fit, draws = 1000)           # simultaneous
confint(fit, type = "marginal", draws = 4000) # separate scalar intervals
```

plot.aersn_region	<i>Plot a confidence region</i>
-------------------	---------------------------------

Description

For $q = 1$ the interval is drawn; for $q = 2$ the region polygon; for $q \geq 3$ a matrix of two-dimensional views, either projections of the joint region (`type = "projection"`, simultaneous coverage) or slices through the estimate (`type = "slice"`, cross-sections). The estimate is marked, and optionally a null value.

Usage

```
## S3 method for class 'aersn_region'
plot(
  x,
  type = c("projection", "slice"),
  null = NULL,
  coords = NULL,
  col = "steelblue",
  ...
)
```

Arguments

<code>x</code>	An <code>aersn_region()</code> object.
<code>type</code>	For $q \geq 3$, "projection" or "slice"; see <code>aersn_projection()</code> .
<code>null</code>	Optional parameter vector to mark (for example a null value); it is drawn in red when outside the region.
<code>coords</code>	For $q \geq 3$, the coordinates to display (default all).
<code>col</code>	Fill color of the region.
<code>...</code>	Further graphical parameters passed to <code>graphics::plot()</code> .

Value

The region, invisibly.

reference-queries	<i>Query a reference distribution</i>
-------------------	---------------------------------------

Description

Quantiles, critical values, p-values and Monte Carlo standard errors of an `aersn_reference()` object.

Usage

```
## S3 method for class 'aersn_reference'
quantile(x, probs = c(0.9, 0.95, 0.99), type = 8, ...)

aersn_critical_value(ref, level = 0.95)

aersn_mcse(ref, probs = c(0.9, 0.95, 0.99), type = 8)

aersn_pvalue(ref, statistic, alternative = c("two.sided", "greater", "less"))
```

Arguments

<code>x</code> , <code>ref</code>	An "aersn_reference" object.
<code>probs</code>	Probabilities (quantile levels).
<code>type</code>	Quantile type passed to <code>stats::quantile()</code> ; the default (type 8) matches the convention of the manuscript's replication code.
<code>...</code>	Unused.
<code>level</code>	Confidence level; the critical value is the <code>level</code> quantile of the reference law.
<code>statistic</code>	Observed statistic value(s).
<code>alternative</code>	For $q = 1$ Monte Carlo and analytic references, "two.sided" (default, based on $ M $), "greater" or "less" (based on the symmetric signed-ratio law). Joint statistics are two-sided only.

Value

`quantile()` and `aersn_critical_value()` return numeric vectors. `aersn_mcse()` returns the Monte Carlo standard error of each quantile estimated from interleaved batches (NA for analytic references). `aersn_pvalue()` returns a list with elements `p.value`, `mcse`, `resolution` (one over the number of draws), and for table references the bracket `lower/upper`. Monte Carlo results also contain `conf.int`, a 95 percent binomial interval for the reference tail probability, and `conf.level`. One-sided scalar tails use symmetry: half the absolute-tail probability on one side and its complement on the other; their resolution is half that of the two-sided tail. A zero estimated tail probability is not an upper bound on the true probability. Quantile standard errors are NA when fewer than two batches or two draws per batch are available. Type-8 quantile interpolation and empirical tail counting can give decisions differing within one Monte Carlo resolution unit at a boundary.

Examples

```
ref <- aersn_reference(1, n = 100, draws = 4000, seed = 2)
quantile(ref, 0.95); aersn_mcse(ref, 0.95)
aersn_pvalue(ref, 1.9)
```

Index

* datasets

aersn_registry, 36

aersn, 3

aersn(), 6, 8, 9, 11, 13–15, 19, 20, 22, 23, 26, 30, 33, 35, 38, 41, 43, 45

aersn_clear_cache, 5

aersn_compare, 5

aersn_compare(), 10, 12, 17, 21, 24, 27, 39

aersn_contains, 7

aersn_contains(), 35

aersn_contrast, 8

aersn_contrast(), 3, 5, 24, 41

aersn_critical_value
(reference-queries), 47

aersn_critical_value(), 34, 40

aersn_ewc_lrv, 9

aersn_ewc_lrv(), 17, 24, 27, 29

aersn_fixed_b_normalizer, 11

aersn_fixed_b_normalizer(), 10, 16, 17, 24, 27

aersn_gauge, 12

aersn_gauge(), 18, 33, 35, 36

aersn_gmm, 13

aersn_gmm(), 3, 4, 22, 25

aersn_hac_lrv, 15

aersn_hac_lrv(), 10–12, 24, 27, 29

aersn_hull, 17

aersn_hull(), 4, 19, 30, 40, 44

aersn_hull_normalizer, 19

aersn_ldl, 19

aersn_ldl(), 21

aersn_ldl_normalizer, 20

aersn_ldl_normalizer(), 16, 24, 27

aersn_lm, 22

aersn_lm(), 3

aersn_mcse (reference-queries), 47

aersn_mcse(), 34

aersn_mean, 23

aersn_mean(), 3

aersn_methods, 24

aersn_methods(), 6, 8, 26, 35, 43, 45

aersn_mle, 25

aersn_mle(), 3, 4, 28, 31

aersn_normalizer, 26

aersn_normalizer(), 7–12, 15, 17, 19, 21, 35, 39, 43, 45

aersn_opg_profile, 27

aersn_opg_profile(), 25, 31

aersn_parametric_reference, 28

aersn_parametric_reference(), 33

aersn_path, 29

aersn_path(), 4, 18

aersn_profile, 30

aersn_profile(), 4, 14, 23, 25, 28, 29, 33

aersn_projection, 31

aersn_projection(), 46

aersn_pvalue (reference-queries), 47

aersn_pvalue(), 34, 40, 43

aersn_reference, 32

aersn_reference(), 4, 5, 28–30, 38, 42, 43, 47

aersn_region, 34

aersn_region(), 3, 5, 7, 13, 24, 32, 40, 44, 46

aersn_registry, 34, 36, 40

aersn_scalar_cdf, 37

aersn_scalar_cdf(), 32, 34

aersn_scalar_density
(aersn_scalar_cdf), 37

aersn_scalar_quantile
(aersn_scalar_cdf), 37

aersn_shao_normalizer, 38

aersn_shao_normalizer(), 11, 12, 24, 27

aersn_slice (aersn_projection), 31

aersn_support, 40

aersn_support(), 18, 35

aersn_table_reference, 40

aersn_target, 41

aersn_target(), 4, 5, 43

aersn_test, [42](#)
aersn_test(), [3](#), [5](#), [7](#), [9](#), [24](#), [26](#), [35](#), [36](#), [45](#)
aersn_vertices, [44](#)
aersn_vertices(), [35](#)

confint.aersn, [44](#)
confint.aersn(), [3](#), [5](#), [24](#)

graphics::plot(), [46](#)

plot.aersn_region, [46](#)
plot.aersn_region(), [35](#)

quantile(), [34](#)
quantile.aersn_reference
 (reference-queries), [47](#)

reference-queries, [47](#)

sandwich::bwAndrews(), [16](#)
sandwich::bwNeweyWest(), [16](#)
sandwich::kweights(), [16](#)
sandwich::meatHAC(), [16](#)
sandwich::NeweyWest(), [16](#)
sandwich::vcovHAC(), [16](#)
stats::lm(), [22](#)
stats::quantile(), [47](#)
stats::uniroot(), [38](#)