Package ‘robscale’

March 30, 2026

Type Package

Title Accelerated Estimation of Robust Location and Scale

Version 0.5.4

Description Estimates robust location and scale parameters using platform-specific Single Instruction, Multiple Data (SIMD) vectorization and Intel Threading Building Blocks (TBB) for parallel processing. Implements a novel variance-weighted ensemble estimator that adaptively combines all available statistics. Methods include logistic M-estimators, the estimators of Rousseeuw and Croux (1993), the Gini mean difference, the scaled Median Absolute Deviation (MAD), the scaled Interquartile Range (IQR), and unbiased standard deviations. Achieves substantial speedups over existing implementations through an 'Rcpp' backend with fused single-buffer algorithms that halve memory traffic for MAD and M-scale estimation, and a unified dispatcher that automatically selects the optimal estimator based on sample size.

License MIT + file LICENSE

URL https://github.com/davdittrich/robscale, https://doi.org/10.5281/zenodo.18828607

BugReports https://github.com/davdittrich/robscale/issues

Encoding UTF-8

Imports Rcpp (>= 1.0.0), RcppParallel (>= 5.0.0)

LinkingTo Rcpp, RcppParallel, BH

Suggests collapse, devtools, GiniDistance, Hmisc, knitr, revss, rmarkdown, robustbase, testthat (>= 3.0.0)

VignetteBuilder knitr

SystemRequirements GNU make

Config/testthat/edition 3

NeedsCompilation yes

RoxygenNote 7.3.3

Author Dennis Alexis Valin Dittrich [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-4438-8276>)

Maintainer Dennis Alexis Valin Dittrich <davd@economicscience.net>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2026-03-30 14:10:10 UTC

Contents

`adm` *Average Distance to the Median*

Description

Computes the mean absolute deviation from the median, scaled by a consistency constant for asymptotic normality under the Gaussian model.

Usage

```
adm(
  x,
  center = NULL,
  constant = 1.2533141373155,
  na.rm = FALSE,
  ci = FALSE,
  level = 0.95
)
```

Arguments

`x`	A numeric vector.
`center`	Optional numeric scalar giving the central value from which to measure the average absolute distance. Defaults to the median of `x`.
`constant`	Consistency constant for asymptotic normality at the Gaussian. Defaults to $\sqrt{\pi/2} \approx 1.2533$ (Nair, 1947). Set to 1 for the raw (unscaled) mean absolute deviation.
`na.rm`	Logical. If `TRUE`, `NA` values are stripped from `x` before computation. If `FALSE` (the default), the presence of any `NA` raises an error.
`ci`	Logical. If `TRUE`, return a `"robscale_ci"` object with the point estimate and asymptotic confidence interval. Default: `FALSE`.
`level`	Confidence level for the interval (default 0.95).

Details

The average distance to the median (ADM) is defined as

$$\text{ADM}(x) \;=\; C \cdot \frac{1}{n}\sum_{i=1}^{n} |x_i - \text{med}(x)|$$

where C is the consistency constant and $\text{med}(x)$ is the sample median. When `center` is supplied, it replaces the sample median.

Statistical Properties. The default constant $C = \sqrt{\pi/2}$ ensures that the ADM is a consistent estimator of the standard deviation σ under the Gaussian model. At the normal distribution, the ADM achieves an **asymptotic relative efficiency (ARE) of 0.88** compared to the sample standard deviation.

While the ADM is less efficient than the standard deviation for purely Gaussian data, it offers superior resistance to "implosion" (the estimate collapsing to zero). Its implosion breakdown point is $(n-1)/n$, meaning it only collapses if all but one observation are identical. Conversely, its explosion breakdown point is $1/n$, similar to the sample mean. These properties make the ADM the ideal **implosion fallback** for the M-estimator of scale in `robScale`.

Computational Performance. This implementation employs a tiered selection strategy: optimal sorting networks for $n \le 16$ and adaptive $O(n)$ selection (Floyd–Rivest or pdqselect, depending on cache-derived thresholds) for larger datasets. This avoids the full sort required by the standard `median` function.

Value

A single numeric value: the scaled mean absolute deviation from the center. Returns `NA` if `x` has length zero after removal of `NA`s.

References

Nair, K. R. (1947) A Note on the Mean Deviation from the Median. *Biometrika*, **34**(3/4), 360–362. doi:10.2307/2332448

Rousseeuw, P. J. and Verboven, S. (2002) Robust estimation in very small samples. *Computational Statistics & Data Analysis*, **40**(4), 741–758. doi:10.1016/S01679473(02)000786

See Also

mad for the median absolute deviation from the median; robScale for the M-estimator of scale that uses the ADM as an implosion fallback.

Examples

```
adm(c(1:9))

x <- c(1, 2, 3, 5, 7, 8)
adm(x)                      # with consistency constant
adm(x, constant = 1)        # raw mean absolute deviation

# Supply a known center
adm(x, center = 4.0)
```

---

get_consistency_constant
Get Consistency Constant

Description

Returns the consistency constant or finite-sample correction factor for a given scale estimator and sample size.

Usage

```
get_consistency_constant(method, n = NULL)
```

Arguments

method	Character string specifying the estimator. Options: "c4", "gmd", "mad", "iqr", "sn", "qn".
n	Integer. The sample size (used for "c4", "sn", and "qn" which have sample-size-dependent factors).

Details

For "c4", "sn", and "qn", the returned value depends on n. For "gmd", "mad", and "iqr", the returned value is the asymptotic constant (independent of n).

Value

A single numeric value: the consistency constant or correction factor.

Examples

```
get_consistency_constant("mad")      # 1.4826
get_consistency_constant("c4", 5)    # c4(5)
get_consistency_constant("qn", 10)   # finite-sample Qn factor
```

---

gmd *Gini Mean Difference*

Description

Computes the Gini mean difference, scaled by a consistency constant for asymptotic normality under the Gaussian model.

Usage

```
gmd(x, constant = 0.886226925452758, na.rm = FALSE, ci = FALSE, level = 0.95)
```

Arguments

`x`	A numeric vector.
`constant`	Consistency constant for asymptotic normality at the Gaussian. Defaults to $\sqrt{\pi}/2 \approx 0.8862$. Set to 1 for the raw (unscaled) Gini mean difference.
`na.rm`	Logical. If `TRUE`, `NA` values are stripped from `x` before computation. If `FALSE` (the default), the presence of any `NA` raises an error.
`ci`	Logical. If `TRUE`, return a `"robscale_ci"` object with the point estimate and asymptotic confidence interval. Default: `FALSE`.
`level`	Confidence level for the interval (default 0.95).

Details

The Gini mean difference is defined as

$$\mathrm{GMD}(x) = C \cdot \frac{2}{n(n-1)} \sum_{i=1}^{n} (2i - n - 1)\, x_{(i)}$$

where $x_{(1)} \leq \ldots \leq x_{(n)}$ are the order statistics and C is the consistency constant. The computation requires a full sort (O(n log n)).

Statistical Properties. The default constant $C = \sqrt{\pi}/2$ ensures that the GMD is a consistent estimator of σ under the Gaussian model. The GMD achieves an **asymptotic relative efficiency (ARE) of 0.98** compared to the sample standard deviation, making it the most efficient robust alternative in this package. Its breakdown point is $1 - 1/\sqrt{2}$, approximately 29.3%.

Value

If `ci = FALSE` (default), a single numeric value: the scaled Gini mean difference. Returns `0` if `n < 2`; returns `NA` if `x` has length zero after removal of `NA`s. If `ci = TRUE`, an object of class `"robscale_ci"`.

References

Gini, C. (1912) *Variabilita e mutabilita*. Bologna.

See Also

`scale_robust` for the unified dispatcher; `qn` and `sn` for high-breakdown scale estimators.

Examples

```
gmd(c(1:9))

x <- c(1, 2, 3, 5, 7, 8)
gmd(x)                       # with consistency constant
gmd(x, constant = 1)         # raw Gini mean difference

# Asymptotic confidence interval
gmd(x, ci = TRUE)
```

---

iqr_scaled *Scaled Interquartile Range*

Description

Computes the interquartile range, scaled by a consistency constant for asymptotic normality under the Gaussian model.

Usage

```
iqr_scaled(
  x,
  constant = 0.741301109252801,
  na.rm = FALSE,
  ci = FALSE,
  level = 0.95
)
```

Arguments

`x`	A numeric vector.
`constant`	Consistency constant for asymptotic normality at the Gaussian. Defaults to $1/(\Phi^{-1}(0.75) - \Phi^{-1}(0.25)) \approx 0.7413$. Set to 1 for the raw interquartile range.
`na.rm`	Logical. If `TRUE`, `NA` values are stripped from `x` before computation. If `FALSE` (the default), the presence of any `NA` raises an error.
`ci`	Logical. If `TRUE`, return a `"robscale_ci"` object with the point estimate and asymptotic confidence interval. Default: `FALSE`.
`level`	Confidence level for the interval (default 0.95).

Details

The scaled IQR is defined as

$$\text{IQR}_s(x) = C \cdot (Q_{0.75} - Q_{0.25})$$

where Q_p denotes the Type 7 quantile (R default) and C is the consistency constant.

Statistical Properties. The default constant ensures consistency for σ under the Gaussian model. The IQR achieves an **asymptotic relative efficiency (ARE) of 0.37** compared to the sample standard deviation. Its breakdown point is 25%.

Computational Performance. Unlike `IQR`, which requires a full sort, this implementation uses O(n) selection via the pdqselect algorithm for each quantile, providing a substantial speedup for large datasets.

Value

If `ci = FALSE` (default), a single numeric value: the scaled interquartile range. Returns `0` if `n < 2`; returns `NA` if `x` has length zero after removal of `NA`s. If `ci = TRUE`, an object of class `"robscale_ci"`.

See Also

`IQR` for the base R (unscaled) interquartile range; `scale_robust` for the unified dispatcher; `mad_scaled` for the scaled median absolute deviation.

Examples

```
iqr_scaled(c(1:9))

x <- c(1, 2, 3, 5, 7, 8)
iqr_scaled(x)                # with consistency constant
iqr_scaled(x, constant = 1)  # raw IQR

# Asymptotic confidence interval
iqr_scaled(x, ci = TRUE)
```

`mad_scaled` *Scaled Median Absolute Deviation*

Description

Computes the median absolute deviation from the median (or a user-supplied center), scaled by a consistency constant for asymptotic normality under the Gaussian model.

Usage

```
mad_scaled(
  x,
  center = NULL,
  constant = 1.4826022185056,
  na.rm = FALSE,
  ci = FALSE,
  level = 0.95
)
```

Arguments

x	A numeric vector.
center	Optional numeric scalar giving the central value from which to measure absolute deviations. Defaults to the median of x.
constant	Consistency constant for asymptotic normality at the Gaussian. Defaults to $1/\Phi^{-1}(0.75) \approx 1.4826$. Set to 1 for the raw median absolute deviation.
na.rm	Logical. If TRUE, NA values are stripped from x before computation. If FALSE (the default), the presence of any NA raises an error.
ci	Logical. If TRUE, return a "robscale_ci" object with the point estimate and asymptotic confidence interval. Default: FALSE.
level	Confidence level for the interval (default 0.95).

Details

The scaled MAD is defined as

$$\mathrm{MAD}_s(x) = C \cdot \mathrm{med}_i\, |x_i - \mathrm{med}(x)|$$

Statistical Properties. The MAD achieves a **50% breakdown point**, meaning it tolerates up to half the data being contaminated. Its **asymptotic relative efficiency (ARE) is 36.8%** compared to the sample standard deviation.

Computational Performance. Unlike mad, this implementation uses O(n) selection with adaptive algorithm dispatch: Floyd-Rivest for moderate n, pdqselect for large n (threshold derived from per-core L2 cache size at startup), and sorting networks for n $\leq$ 16.

Value

If ci = FALSE (default), a single numeric value: the scaled median absolute deviation. Returns 0 if n = 1; returns NA if x has length zero after removal of NAs. If ci = TRUE, an object of class "robscale_ci".

See Also

mad for the base R implementation; scale_robust for the unified dispatcher; robScale for the M-estimate of scale that uses the MAD as a starting value.

Examples

```
mad_scaled(c(1:9))

x <- c(1, 2, 3, 5, 7, 8)
mad_scaled(x)                   # with consistency constant
mad_scaled(x, constant = 1)     # raw median absolute deviation

# Supply a known center
mad_scaled(x, center = 4.0)

# Asymptotic confidence interval
mad_scaled(x, ci = TRUE)
```

`print.robscale_ci` *Print a robscale_ci object*

Description

Print a robscale_ci object

Usage

```
## S3 method for class 'robscale_ci'
print(x, digits = 4, ...)
```

Arguments

`x`	A `"robscale_ci"` object.
`digits`	Number of significant digits to display. Default: 4.
`...`	Further arguments passed to or from other methods.

Value

The object `x`, invisibly.

print.robscale_ensemble_ci *Print a robscale_ensemble_ci object*

Description

Print a robscale_ensemble_ci object

Usage

```
## S3 method for class 'robscale_ensemble_ci'
print(x, digits = 4, ...)
```

Arguments

x	A "robscale_ensemble_ci" object.
digits	Number of significant digits to display. Default: 4.
...	Further arguments passed to or from other methods.

Value

The object x, invisibly.

qn *Robust Estimator of Scale Qn*

Description

Computes the robust estimator of scale Q_n proposed by Rousseeuw and Croux (1993).

Usage

```
qn(
  x,
  constant = 2.21914446598508,
  finite.corr = TRUE,
  na.rm = FALSE,
  ci = FALSE,
  level = 0.95
)
```

Arguments

`x`	A numeric vector of observations.
`constant`	Consistency constant. Default is 2.2191 (full precision: 2.21914446598508).
`finite.corr`	Logical; if `TRUE`, a finite-sample correction factor is applied.
`na.rm`	Logical; if `TRUE`, `NA` values are removed before computation.
`ci`	Logical. If `TRUE`, return a `"robscale_ci"` object with the point estimate and asymptotic confidence interval. Default: `FALSE`.
`level`	Confidence level for the interval (default 0.95).

Details

The Q_n estimator is defined as the k-th order statistic of the $\binom{n}{2}$ absolute pairwise differences $|x_i - x_j|$ for $i < j$. Specifically, $Q_n = C \cdot d_{(k)}$ where d is the set of absolute differences and $k = \binom{h}{2}$ with $h = \lfloor n/2 \rfloor + 1$.

Statistical Properties. Q_n is a highly robust estimator with a **50% breakdown point**. Unlike the Median Absolute Deviation (MAD), Q_n does not require a prior location estimate, making it more robust in asymmetric distributions. At the Gaussian distribution, it achieves an **asymptotic relative efficiency (ARE) of 0.82**, significantly higher than the 0.37 achieved by the MAD.

Computational Performance. While the naive calculation of Q_n requires $O(n^2)$ space and time, this implementation employs a specialized $O(n \log n)$ algorithm. The implementation uses a tiered execution strategy:

- **Optimal sorting networks** for very small samples ($n \leq 16$). These networks eliminate branch misprediction in the target regimes of extremely small samples.
- A **Croux–Rousseeuw weighted-median refinement** algorithm for medium and large datasets.
- **Cache-aware parallelization** via Intel TBB (Threading Building Blocks) for large-scale data, with thresholds derived from the detected per-core L2 cache size.

Value

If `ci = FALSE` (default), the Q_n estimator of scale (a scalar). If `ci = TRUE`, an object of class `"robscale_ci"`.

References

Rousseeuw, P. J., and Croux, C. (1993). Alternatives to the Median Absolute Deviation. *Journal of the American Statistical Association*, **88**(424), 1273–1283. doi:10.1080/01621459.1993.10476408

Akinshin, A. (2022). Finite-sample Rousseeuw-Croux scale estimators. *arXiv preprint arXiv:2209.12268*.

See Also

`sn` for the S_n scale estimator; `robScale` for the M-estimator of scale; `adm` for the average distance to median.

Examples

```
qn(c(1:9))
x <- c(1, 2, 3, 5, 7, 8)
qn(x)

# Asymptotic confidence interval
qn(x, ci = TRUE)
```

---

robLoc *Robust M-Estimate of Location*

Description

Computes the robust M-estimate of location for very small samples using the logistic ψ function of Rousseeuw & Verboven (2002).

Usage

```
robLoc(
  x,
  scale = NULL,
  na.rm = FALSE,
  maxit = 80L,
  tol = sqrt(.Machine$double.eps)
)
```

Arguments

`x`	A numeric vector.
`scale`	Optional numeric scalar giving a known scale. When supplied, the MAD is replaced by this value and the minimum sample size for iteration is lowered from 4 to 3 (see 'Details').
`na.rm`	Logical. If `TRUE`, `NA` values are stripped from `x` before computation. If `FALSE` (the default), the presence of any `NA` raises an error.
`maxit`	Maximum number of Newton–Raphson iterations. Defaults to 80.
`tol`	Convergence tolerance. Iteration stops when the Newton step satisfies `|v| <= tol * max(|t|, 1.0)`, scaling the tolerance by the location magnitude to ensure convergence for large-valued data. Defaults to `sqrt(.Machine$double.eps)`.

Details

The M-estimate of location T_n is defined as the solution to the estimating equation

$$\sum_{i=1}^{n} \psi_{\log}\left(\frac{x_i - T_n}{S_n}\right) = 0$$

where S_n is a fixed auxiliary scale (defaulting to the MAD) and $\psi_{\log}$ is the logistic psi function:

$$\psi_{\log}(x) = \frac{e^x - 1}{e^x + 1} = \tanh(x/2)$$

Statistical Properties. The logistic psi function is bounded, smooth (C^∞), and strictly monotone. These properties ensure that the resulting M-estimator is both robust to outliers and numerically stable. At the Gaussian distribution, the logistic M-estimator of location achieves high efficiency, with an **asymptotic relative efficiency (ARE) of 0.98** compared to the sample mean.

Small-Sample Strategy. Following Rousseeuw & Verboven (2002), location and scale are estimated separately. In `robLoc`, the auxiliary scale S_n remains fixed throughout the Newton–Raphson iteration. This "decoupled" approach avoids the instabilities often encountered in small samples when using simultaneous location–scale iteration (e.g., Huber's Proposal 2).

Numerical Computation. The estimating equation is solved via Newton–Raphson iteration starting from the sample median. Because the derivative of the logistic psi satisfies $\psi'(x) = \frac{1}{2}(1 - \psi^2(x))$, the Newton step is computationally efficient, requiring no additional transcendental calls beyond the `tanh` evaluations used for the psi function itself.

The denominator of the Newton step is the *observed Fisher information*

$$\hat{I}(t) = \sum_{i=1}^{n} \operatorname{sech}^2\left(\frac{x_i - t}{2S_n}\right),$$

recomputed at each iteration using the current estimate t. This distinguishes the implementation from IRLS (iteratively reweighted least squares), which uses a fixed expected-information denominator and converges linearly. Using the observed Hessian yields true Newton–Raphson: the fixed-point derivative at the solution satisfies $T'(t^*) = 0$, giving *quadratic* local convergence. On typical data, convergence is achieved in two to four iterations; the `maxit` bound of 80 is a conservative safety limit.

Performance and SIMD. The C++ kernel dispatches `tanh` to the fastest available backend: Apple Accelerate on macOS, glibc libmvec (AVX2 4-wide) on Linux x86-64, SLEEF when libmvec is absent, or `#pragma omp simd` as a portable fallback. A fused AVX2 kernel accumulates ψ_i and $\mathrm{d}\psi_i$ in a single pass, halving memory reads.

Fallback Mechanism. For extremely small samples where iteration may be unreliable, the function returns the `median` directly:

- If scale is unknown: $n < 4$ (since the MAD of 3 points is insufficiently robust).
- If scale is supplied: $n < 3$.

Value

A single numeric value: the robust M-estimate of location. Returns `NA` if `x` has length zero (after removal of `NA`s when `na.rm = TRUE`).

References

Rousseeuw, P. J. and Verboven, S. (2002) Robust estimation in very small samples. *Computational Statistics & Data Analysis*, **40**(4), 741–758. doi:10.1016/S01679473(02)000786

See Also

median for the starting value; mad for the auxiliary scale; robScale for the companion scale estimator; qn and sn for high-efficiency scale estimators.

Examples

```
robLoc(c(1:9))

x <- c(1, 2, 3, 5, 7, 8)
robLoc(x)

# Known scale lowers the minimum sample size to 3
robLoc(c(1, 2, 3), scale = 1.5)

# Outlier resistance
x_clean <- c(2.0, 3.1, 2.7, 2.9, 3.3)
x_dirty <- replace(x_clean, 5, 100)
c(robLoc(x_clean), robLoc(x_dirty))   # barely moves
c(mean(x_clean), mean(x_dirty))       # destroyed
```

---

robScale *Robust M-Estimate of Scale*

Description

Computes the robust M-estimate of scale for very small samples using the ρ function of Rousseeuw & Verboven (2002).

Usage

```
robScale(
  x,
  loc = NULL,
  fallback = c("adm", "na"),
  implbound = 1e-04,
  na.rm = FALSE,
  maxit = 80L,
  tol = sqrt(.Machine$double.eps),
  ci = FALSE,
  level = 0.95
)
```

Arguments

x	A numeric vector.

`loc`	Optional numeric scalar giving a known location. When supplied, the observations are centered at `loc` and the minimum sample size for iteration is lowered from 4 to 3 (see 'Details').
`fallback`	Character string specifying the fallback behavior when the MAD collapses to zero or the sample size is too small for iteration. Must be one of `"adm"` (default) or `"na"`. See 'Details'.
`implbound`	Numeric scalar specifying the threshold for MAD implosion. Defaults to `1e-4`. Passing a value of 0 disables implosion checks.
`na.rm`	Logical. If `TRUE`, `NA` values are stripped from `x` before computation. If `FALSE` (the default), the presence of any `NA` raises an error.
`maxit`	Maximum number of Newton–Raphson iterations. Defaults to 80.
`tol`	Convergence tolerance. Iteration stops when the relative change in the scale estimate falls below `tol`. Defaults to `sqrt(.Machine$double.eps)`.
`ci`	Logical. If `TRUE`, return a `"robscale_ci"` object with the point estimate and asymptotic confidence interval. Default: `FALSE`.
`level`	Confidence level for the interval (default 0.95).

Details

The M-estimate of scale S_n is defined as the solution to the estimating equation

$$\frac{1}{n}\sum_{i=1}^{n}\rho\left(\frac{x_i - T_n}{S_n}\right) = \beta$$

where the location T_n is fixed at the sample median, $\beta = 0.5$ is the expected value of ρ under the Gaussian model, and ρ is a smooth rho function (Rousseeuw & Verboven, 2002, Sec. 4.2):

$$\rho_{\log}(x) = \psi^2_{\log}\left(\frac{x}{c}\right)$$

The tuning constant $c = 0.373941121$ is chosen to satisfy $E_\Phi[\rho(u)] = 0.5$.

Statistical Properties. This estimator is designed for high robustness and efficiency. It achieves a **50% breakdown point**, meaning the estimate remains reliable even if half the sample is contaminated by outliers. At the Gaussian distribution, the logistic M-estimator of scale achieves an **asymptotic relative efficiency (ARE) of 0.55** compared to the sample standard deviation.

Numerical Computation. The M-scale estimating equation $n^{-1}\sum\rho(u_i) = 1/2$ is solved by Newton–Raphson iteration, starting from the MAD. Each step computes $u_i = (x_i - T)/(2cS)$ and accumulates two sums in a single pass: $\sum\tanh^2(u_i)$ (the rho sum) and $\sum u_i\tanh(u_i)\operatorname{sech}^2(u_i)$ (the derivative sum). The NR update is

$$\Delta S = S \cdot \frac{\bar{\rho} - 1/2}{(2/n)\sum u_i\tanh(u_i)\operatorname{sech}^2(u_i)}$$

with convergence test $|\Delta S|/S \le \mathrm{tol}$. When the derivative sum degenerates, the iteration falls back to a multiplicative half-step. Quadratic convergence yields 3–4 iterations on typical data. Location is held fixed at the sample median, following the decoupled approach of Rousseeuw and Verboven (2002) that avoids the positive-feedback instabilities of simultaneous location–scale estimation.

Performance and SIMD. The C++ kernel dispatches `tanh` to the fastest available backend: Apple Accelerate on macOS, glibc libmvec (AVX2 4-wide) on Linux x86-64, SLEEF when libmvec is absent, or `#pragma omp simd` as a portable fallback.

Known location. When `loc` is supplied, the observations are centered as $x_i - \mu$ and the initial scale is set to $1.4826 \cdot \mathrm{med}(|x_i - \mu|)$ rather than the MAD. This lowers the minimum sample size from 4 to 3 (Rousseeuw & Verboven, 2002, Sec. 5).

Fallback Mechanism and Implosion. Robust scale estimators like the MAD can "implode" (collapse to zero) if more than 50 or the sample size is too small for reliable iteration ($n < 4$, or $n < 3$ if location is known):

- If `fallback = "adm"` (default), the function returns the scaled Average Distance to the Median (`adm`). The ADM is highly resistant to implosion (breakdown point $(n-1)/n$).
- If `fallback = "na"`, the function returns `NA`.

Value

If `ci = FALSE` (default), a single numeric value: the robust M-estimate of scale. Returns `NA` if `x` has length zero (after removal of `NA`s when `na.rm = TRUE`) or if the MAD collapses and `fallback = "na"`. If `ci = TRUE`, an object of class `"robscale_ci"`.

References

Rousseeuw, P. J. and Verboven, S. (2002) Robust estimation in very small samples. *Computational Statistics & Data Analysis*, **40**(4), 741–758. doi:10.1016/S01679473(02)000786

See Also

`adm` for the implosion fallback; `mad` for the starting value and classical alternative; `robLoc` for the companion location estimator; `qn` and `sn` for high-efficiency scale estimators.

Examples

```
robScale(c(1:9))

x <- c(1, 2, 3, 5, 7, 8)
robScale(x)
robScale(x, loc = 5)           # known location

# Outlier resistance
x_clean <- c(2.0, 3.1, 2.7, 2.9, 3.3)
x_dirty <- replace(x_clean, 5, 100)
c(robScale(x_clean), robScale(x_dirty))   # barely moves
c(sd(x_clean), sd(x_dirty))               # destroyed

# MAD implosion: identical values cause MAD = 0
robScale(c(5, 5, 5, 5, 6))     # falls back to adm()

# Asymptotic confidence interval
robScale(x, ci = TRUE)
```

scale_robust — *Robust Ensemble Scale Estimation*

Description

Unified dispatcher for robust scale estimation. Automatically selects between a variance-weighted ensemble of 7 estimators (for small samples) and the Gini Mean Difference (for large samples), or returns a specific estimator by name.

Usage

```
scale_robust(
  x,
  method = c("ensemble", "gmd", "sd", "mad", "iqr", "sn", "qn", "robScale"),
  auto_switch = TRUE,
  threshold = 20L,
  n_boot = 200L,
  na.rm = FALSE,
  ci = FALSE,
  level = 0.95,
  boot_method = c("auto", "bca", "percentile", "parametric", "analytical")
)
```

Arguments

`x`	A numeric vector of observations.
`method`	Character string specifying the estimation method. Options: `"ensemble"` (default), `"gmd"`, `"sd"`, `"mad"`, `"iqr"`, `"sn"`, `"qn"`, `"robScale"`.
`auto_switch`	Logical. If `TRUE` (default), automatically uses GMD when `method = "ensemble"` and `n >= threshold`. Has no effect on named methods: `method = "qn"` always returns the Qn estimator regardless of sample size.
`threshold`	Integer. Sample size at which the ensemble auto-switches to GMD. Default: 20 (research-backed; see 'Details').
`n_boot`	Integer. Number of bootstrap replicates for the ensemble weighting or bootstrap CI. Default: 200.
`na.rm`	Logical. If `TRUE`, `NA` values are stripped from `x` before computation. Default: `FALSE`.
`ci`	Logical. If `TRUE`, return confidence intervals alongside the point estimate. Single methods yield a `"robscale_ci"` object; the ensemble yields a `"robscale_ensemble_ci"` object. Default: `FALSE`.
`level`	Confidence level for the interval (default 0.95).
`boot_method`	CI method. For single named methods: `"auto"` and `"analytical"` (default) return the closed-form ARE-based interval (`sd` uses chi-squared; all others use the normal approximation). `"bca"`, `"percentile"`, and `"parametric"` return a

bootstrap CI via n_boot resamples. For the ensemble: "auto" selects BCa for $n \leq 200$, percentile for $n \leq 5000$, and parametric otherwise. "analytical" is not supported for method = "ensemble".

Details

Ensemble method. When method = "ensemble", the function computes a variance-weighted combination of 7 scale estimators:

1. sd_c4 — unbiased standard deviation
2. gmd — Gini mean difference
3. mad_scaled — median absolute deviation
4. iqr_scaled — scaled interquartile range
5. sn — Sn estimator of Rousseeuw & Croux
6. qn — Qn estimator of Rousseeuw & Croux
7. robScale — logistic M-estimator of scale

The weights are determined by bootstrap resampling: each estimator's inverse variance across n_boot resamples determines its contribution. Estimators with lower sampling variance receive higher weight.

Automatic switching. When auto_switch = TRUE and method = "ensemble" and n >= threshold, the function returns gmd(x) directly. The GMD achieves 98% asymptotic relative efficiency at the Gaussian while being computationally cheaper than the ensemble. Named methods (e.g.\ method = "qn") are always dispatched as requested; auto_switch never overrides an explicit method choice.

Individual methods. When a specific method is requested, scale_robust bypasses the ensemble and calls the corresponding C++ entry point directly. With ci = TRUE, the default boot_method = "auto" returns the analytical interval: chi-squared for "sd", ARE-based normal approximation for all others. Pass boot_method = "bca", "percentile", or "parametric" to obtain a bootstrap CI instead.

Value

If ci = FALSE (default), a single numeric value: the scale estimate (NA when n < 2). If ci = TRUE with a single method, a "robscale_ci" object. If ci = TRUE with method = "ensemble", a "robscale_ensemble_ci" object containing the ensemble estimate, bootstrap CI, and a table of per-estimator results.

See Also

Individual estimators: sd_c4, gmd, mad_scaled, iqr_scaled, sn, qn, robScale; get_consistency_constant for the underlying constants.

Examples

```
x <- c(1, 2, 3, 5, 7, 8)
scale_robust(x)                        # ensemble (n < 20)
scale_robust(x, method = "qn")         # specific method (always qn)

set.seed(42)
```

```
y <- rnorm(50)
scale_robust(y)                         # ensemble auto-switches to GMD
scale_robust(y, method = "qn")          # qn regardless of n
scale_robust(y, auto_switch = FALSE)    # forces ensemble

# Analytical CI for a named method (default)
scale_robust(x, method = "sn", ci = TRUE)

# Bootstrap CI for a named method
scale_robust(x, method = "qn", ci = TRUE, boot_method = "percentile")

# Ensemble with bootstrap CIs
scale_robust(x, ci = TRUE)
```

---

sd_c4 *Unbiased Standard Deviation*

Description

Computes the sample standard deviation corrected by the $c_4(n)$ factor to remove the small-sample bias of the square-root estimator.

Usage

```
sd_c4(x, na.rm = FALSE, ci = FALSE, level = 0.95)
```

Arguments

x	A numeric vector.
na.rm	Logical. If TRUE, NA values are stripped from x before computation. If FALSE (the default), the presence of any NA raises an error.
ci	Logical. If TRUE, return a "robscale_ci" object with the point estimate and exact chi-squared confidence interval. Default: FALSE.
level	Confidence level for the interval (default 0.95).

Details

The standard sd function computes $s = \sqrt{s^2}$ where s^2 is the unbiased sample variance. However, $E[s] \neq \sigma$ for finite n; the square root introduces a downward bias.

The $c_4(n)$ correction factor removes this bias:

$$\hat{\sigma} = \frac{s}{c_4(n)} = \frac{s}{\sqrt{2/(n-1)} \cdot \Gamma(n/2)/\Gamma((n-1)/2)}$$

Statistical Properties. This is a classical (non-robust) estimator with **100% ARE** by construction, but **0% breakdown point** — a single outlier can make it arbitrarily large.

Numerical Stability. Uses Welford's online algorithm for numerically stable variance computation, avoiding catastrophic cancellation that affects naive two-pass formulas.

Value

If `ci = FALSE` (default), a single numeric value: the bias-corrected standard deviation. Returns `NA` if `n < 2`. If `ci = TRUE`, an object of class `"robscale_ci"` with an exact chi-squared confidence interval.

See Also

`sd` for the (biased) sample standard deviation; `scale_robust` for the unified dispatcher; `robScale` for the robust M-estimate of scale.

Examples

```
sd_c4(c(1:9))

# Compare with base sd() --- difference is small for large n
x <- rnorm(1000)
c(sd_c4 = sd_c4(x), sd = sd(x))

# Exact chi-squared confidence interval
sd_c4(c(1:9), ci = TRUE)
```

---

sn — *Robust Estimator of Scale Sn*

Description

Computes the robust estimator of scale S_n proposed by Rousseeuw and Croux (1993).

Usage

```
sn(
  x,
  constant = 1.19259855312321,
  finite.corr = TRUE,
  na.rm = FALSE,
  ci = FALSE,
  level = 0.95
)
```

Arguments

`x`	A numeric vector of observations.
`constant`	Consistency constant. Default is 1.1926 (full precision: 1.19259855312321).
`finite.corr`	Logical; if `TRUE`, a finite-sample correction factor is applied.
`na.rm`	Logical; if `TRUE`, `NA` values are removed before computation.

`ci`	Logical. If `TRUE`, return a `"robscale_ci"` object with the point estimate and asymptotic confidence interval. Default: `FALSE`.
`level`	Confidence level for the interval (default 0.95).

Details

The S_n estimator is defined as the median of medians:

$$S_n = C \cdot \text{med}_i \{\text{med}_j |x_i - x_j|\}$$

Statistical Properties. S_n achieves a **50% breakdown point** and provides superior statistical efficiency compared to the Median Absolute Deviation (MAD). At the Gaussian distribution, it has an **asymptotic relative efficiency (ARE) of 0.58**, which is significantly higher than the 0.37 of the MAD. Unlike M-estimators of scale, S_n is an explicit function of the data and does not require an iterative solution or a prior location estimate.

Computational Performance. This implementation provides a highly optimized $O(n \log n)$ algorithm, avoiding the $O(n^2)$ complexity of the naive definition. The execution strategy is tiered for maximum efficiency:

- **Optimal sorting networks** are used for the target regime of very small samples ($n \leq 16$). These networks minimize control flow overhead and maximize pipeline utilization.
- **Highly tuned kernels** are employed for general datasets, leveraging C++17 features for cache-aware computation.
- **Cache-aware parallelization** via Intel TBB (Threading Building Blocks) for large-scale data, with thresholds derived from the detected per-core L2 cache size.

Value

If `ci = FALSE` (default), the S_n estimator of scale (a scalar). If `ci = TRUE`, an object of class `"robscale_ci"`.

References

Rousseeuw, P. J., and Croux, C. (1993). Alternatives to the Median Absolute Deviation. *Journal of the American Statistical Association*, **88**(424), 1273–1283. doi:10.1080/01621459.1993.10476408

Akinshin, A. (2022). Finite-sample Rousseeuw-Croux scale estimators. *arXiv preprint arXiv:2209.12268*.

See Also

`qn` for the Q_n scale estimator; `robScale` for the M-estimator of scale; `adm` for the average distance to median.

Examples

```
sn(c(1:9))
x <- c(1, 2, 3, 5, 7, 8)
sn(x)

# Asymptotic confidence interval
```

```
sn(x, ci = TRUE)
```

Index