Package 'robscale'

March 9, 2026

Type Package

Title Faster Robustness: Accelerated Estimation of Location and Scale

Version 0.1.5

Description Robust estimation ensures statistical reliability in data contaminated by outliers. Yet, computational bottlenecks in existing 'R' implementations frequently obstruct both very small sample analysis and large-scale processing. 'robscale' resolves these inefficiencies by providing high-performance implementations of logistic M-estimators and the 'Qn' and 'Sn' scale estimators. By leveraging platform-specific Single Instruction, Multiple Data (SIMD) vectorization and Intel Threading Building Blocks (TBB) parallelism, the package delivers speedups of 11–39x for small samples and up to 10x for massive datasets. These performance gains enable the integration of robust statistics into modern, time-critical computational workflows. Replaces 'revss' with an 'Rcpp' backend.

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 testthat (>= 3.0.0), revss, microbenchmark, robustbase

SystemRequirements GNU make, TBB

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>

Repository CRAN

Date/Publication 2026-03-09 21:20:02 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, constant = 1.2533141373155, na.rm = FALSE)
```

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.

Details

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

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

where C is the consistency constant and $\mathrm{med}(x)$ 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 very small samples ($n \leq 8$) and a C++17 introselect algorithm for larger datasets. This approach provides an $O(n)$ worst-case time complexity, typically outperforming pure-R implementations by an order of magnitude. Performance is further enhanced by avoiding 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)
```

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 = 2.2191, finite.corr = TRUE, na.rm = FALSE)
```

Arguments

`x`	A numeric vector of observations.
`constant`	Consistency constant. Default is 2.2191.
`finite.corr`	Logical; if `TRUE`, a finite-sample correction factor is applied.
`na.rm`	Logical; if `TRUE`, `NA` values are removed before computation.

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 package utilizes a tiered execution strategy:

- **Optimal sorting networks** for very small samples ($n \leq 8$). These networks eliminate branch misprediction in the target regimes of extremely small samples.
- A specialized **Johnson–Mizoguchi selection** algorithm for medium-sized datasets.
- **Cache-aware parallelization** via Intel TBB (Threading Building Blocks, when `ROBSCALE_FAST=1`) for large-scale data.

Value

The Q_n estimator of scale.

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

Examples

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

`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 absolute Newton step falls below `tol`. 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 0.95** 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.

Performance and SIMD. The underlying C++ core utilizes platform-specific SIMD (Single Instruction, Multiple Data) backends (SLEEF on Linux, Apple Accelerate on macOS) to vectorize the `tanh` evaluations. This architectural choice delivers substantial performance gains, particularly for large-scale or high-throughput workflows.

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 = 0.0001,
  na.rm = FALSE,
  maxit = 80L,
  tol = sqrt(.Machine$double.eps)
)
```

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 iterations for the multiplicative algorithm. 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)`.

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 0.76** compared to the sample standard deviation.

Numerical Computation. The estimating equation is solved by multiplicative iteration (Rousseeuw & Verboven, 2002, Eq. 27):

$$S^{(k+1)} = S^{(k)} \cdot \sqrt{2 \cdot \frac{1}{n}\sum \psi^2_{\log}\left(\frac{x_i - T}{c \cdot S^{(k)}}\right)}$$

The algorithm starts at the Median Absolute Deviation (MAD). Because location is held fixed at the sample median, the estimator follows a "decoupled" approach that avoids the positive-feedback instabilities often seen in simultaneous location–scale estimation (Proposal 2) at very small sample sizes.

Performance and SIMD. The C++ implementation leverages platform-specific SIMD (Single Instruction, Multiple Data) backends (SLEEF on Linux, Apple Accelerate on macOS) to $\psi_{\log}$ evaluations (via `tanh`). This specialized architecture typically yields an 11–39x speedup over pure-R code for samples of size $n \leq 20$.

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

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"`.

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()
```

`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 = 1.1926, finite.corr = TRUE, na.rm = FALSE)
```

Arguments

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

Details

The S_n estimator is defined as the median of medians:

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

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 ($n \leq 8$). 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.

Value

The S_n estimator of scale.

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

Examples

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

Index