Package ‘revss’

March 19, 2026

Type Package

Title Robust Estimation in Very Small Samples

Version 3.1.0

Date 2026-03-18

Description Implements and enhances the estimation techniques described in Rousseeuw & Verboven (2002) <doi:10.1016/S0167-9473(02)00078-6> for the location and scale of very small samples.

License BSD_2_clause + file LICENSE

URL https://github.com/aadler/revss

BugReports https://github.com/aadler/revss/issues

Encoding UTF-8

Suggests covr, tinytest

NeedsCompilation yes

Author Avraham Adler [aut, cph, cre] (ORCID: <https://orcid.org/0000-0002-3039-0703>)

Maintainer Avraham Adler <Avraham.Adler@gmail.com>

Repository CRAN

Date/Publication 2026-03-19 08:00:02 UTC

Contents

revss-package	*Robust Estimation in Very Small Samples*

Description

Implements and enhances the estimation techniques described in Rousseeuw & Verboven (2002) <doi:10.1016/S0167-9473(02)00078-6> for the location and scale of very small samples.

Details

The DESCRIPTION file:

Package:	revss
Type:	Package
Title:	Robust Estimation in Very Small Samples
Version:	3.1.0
Date:	2026-03-18
Authors@R:	c(person(given = "Avraham", family = "Adler", role = c("aut", "cph", "cre"), email = "Avraham.Adler@
Description:	Implements and enhances the estimation techniques described in Rousseeuw & Verboven (2002) <doi:1
License:	BSD_2_clause + file LICENSE
URL:	https://github.com/aadler/revss
BugReports:	https://github.com/aadler/revss/issues
Encoding:	UTF-8
Suggests:	covr, tinytest
NeedsCompilation:	yes
Author:	Avraham Adler [aut, cph, cre] (ORCID: <https://orcid.org/0000-0002-3039-0703>)
Maintainer:	Avraham Adler <Avraham.Adler@gmail.com>
Archs:	x64

Index of help topics:

```
adm                     Mean Absolute Deviation from the Median or Mean
admn                    Mean Absolute Deviation from the Median or Mean
                        for Small Samples
madn                    Median Absolute Deviation from the Median or
                        Mean for Small Sample Size
revss-package           Robust Estimation in Very Small Samples
robLoc                  Robust Estimate of Location
robScale                Robust Estimate of Scale
```

Author(s)

Avraham Adler [aut, cph, cre] (ORCID: <https://orcid.org/0000-0002-3039-0703>)

Maintainer: Avraham Adler <Avraham.Adler@gmail.com>

adm — *Mean Absolute Deviation from the Median or Mean*

Description

Compute the mean absolute deviation from the median, also known as the average deviation to the median (ADM) and (by default) adjust by a factor for asymptotically normal consistency.

Usage

```
adm(x, center = NULL, constant = NULL, na.rm = FALSE)
```

Arguments

`x`	**numeric**; A vector of values.
`center`	optional; The central value from which to measure the average distance. It can either be a scalar **numeric** value or a **function** on `x` returning a single numeric value. Defaults to the median of `x`.
`constant`	**numeric**; A scale factor for asymptotic normality. When `NULL`, it defaults to $\sqrt{\frac{\pi}{2}}$.
`na.rm`	**logical**; If `TRUE` then `NA` values are stripped from `x` before computation takes place.

Details

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

where C is the consistency constant and `center` defaults to `median`.

The ADM is the average distance, as an absolute value, between each observation and the central observation—usually the median. In statistical literature this is also called the **mean absolute deviation around the median**. Unfortunately, this shares the same acronym as the median absolute deviation (MAD), which is the median equivalent of this function.

General practice is to adjust the factor for asymptotically normal consistency. In large samples, assuming the Gaussian distribution, the mean absolute deviation from the median approaches $\sqrt{\frac{2}{\pi}}$, the same value as the mean absolute deviation from the mean (Pham-Gia & Hung 2001). This latter statistic may be returned by passing `center = mean(x)` in the function call, where `x` is whatever is being passed in the first position.

Given the asymptotic behavior, the default is to multiple the results by the reciprocal—$\sqrt{\frac{\pi}{2}}$. However, it is important to note that this asymptotic behavior often does **not** hold with the smaller sample sizes for which this package is intended.

If `na.rm` is `TRUE` then `NA` values are stripped from `x` before computation takes place. If this is not done then an `NA` value in `x` will cause `mad` to return `NA`.

Value

A **numeric** value representing the average absolute deviation from the requested central tendency adjusted by the constant.

Author(s)

Avraham Adler <Avraham.Adler@gmail.com>

References

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

Pham-Gia, T. and Hung, T. L. (2001) "The mean and median absolute deviations," *Mathematical and Computer Modelling*, **34** (7–8), 921–936. doi:10.1016/s08957177(01)001091

See Also

See `admn` for the small-sample bias-corrected version of this function, `mad` in **stats** for the median absolute deviation from the `median`, `madn` for the small-sample bias-corrected version of `mad`, and `robScale` for a robust small-sample estimator of scale.

Examples

```
set.seed(19L)
z <- rnorm(7, 8, 3)
sd(z); sd(z) / 3 - 1
adm(z); adm(z) / 3 - 1
```

---

admn — *Mean Absolute Deviation from the Median or Mean for Small Samples*

Description

Compute the mean absolute deviation from the median, also known as the average deviation to the median (ADM) and adjust by applying factors calculated specifically for small-sample size to achieve unbiased asymptotic normal consistency.

Usage

```
admn(x, center = c("median", "mean"), na.rm = FALSE)
```

Arguments

`x`	**numeric**; A vector of values.
`center`	**character**; either `"median"` or `"mean"` which will be used as the function to calculate the central tendency from which to measure the absolute deviations. Defaults to `"median"`.
`na.rm`	**logical**; If `TRUE` then `NA` values are stripped from `x` before computation takes place.

Details

$$ADM = a_n \sqrt{\frac{\pi}{2}} \frac{1}{n} \sum_{i=1}^{n} |x_i - \underset{j}{\text{center}}\, x_j|$$

where a_n is calculated by Adler (2026) and `center` is either `median` or `mean`.

Inspired by Croux & Rousseeuw (1992), a large-scale Monte-Carlo simulation was performed (2026) to calculate correction factors to make the standard ADM estimate more unbiased for small samples. It is called ADM_n to differentiate it from the more standard ADM, given the a_n multiplier is dependent on n—the size of the sample.

This function differs from its larger-scale version, `adm`, in other ways. First, it only accepts `"median"` or `"mean"` as its central tendency—not a scalar value or any other scalar-valued function—as the factors were only calculated for those two central tendencies. Also, `admn` does not allow passing a user-defined constant, as the intent is to return the unbiased estimate assuming normality.

If `na.rm` is `TRUE` then `NA` values are stripped from `x` before computation takes place. If this is not done then an `NA` value in `x` will cause `madn` to return `NA`.

Value

A **numeric** value representing the average absolute deviation from the requested central tendency adjusted by the asymptotic normality constant and the small-sample bias-reduction constant.

Author(s)

Avraham Adler <Avraham.Adler@gmail.com>

References

Croux, Christophe and Rousseeuw, Peter J. (1992) "Time-Efficient Algorithms for Two Highly Robust Estimators of Scale", *Computational Statistics*, Vol. 1, 411–428. doi:10.1007/9783662-268117_58

Adler, Avraham (2026) "Reducing the Bias in the Robust Estimation of Very Small Samples with the revss package for R", Manuscript in preparation

See Also

See `adm` for the standard definition this function, `mad` in **stats** for the median absolute deviation from the `median`, and `madn` for the small-sample bias-corrected version of `mad`. See `robScale` for a robust M-estimator of scale.

Examples

```
set.seed(19L)
z <- rnorm(7, 8, 3)
sd(z); sd(z) / 3 - 1
adm(z); adm(z) / 3 - 1
admn(z); admn(z) / 3 - 1
```

madn — *Median Absolute Deviation from the Median or Mean for Small Sample Size*

Description

The median absolute deviation (MAD) is actually the median absolute deviation from the median. For small samples, the normal bias correction of 1.4826 has been shown by Croux & Rousseeuw (1992) to be insufficient. This function can apply the n-specific b_n correction factors found in that paper. It also can apply factors found by a significantly larger Monte-Carlo simulation (2026) for either the median absolute distance from the median or from the mean.

Usage

```
madn(x, center = c("median", "mean"), factors = c("AA", "CR"), na.rm = FALSE)
```

Arguments

`x`	**numeric**; A vector of values.
`center`	**character**; either `"median"` or `"mean"` which will be used as the function to calculate the central tendency from which to measure the absolute deviations. Defaults to `"median"`.
`factors`	**character**; `"CR"` will use the factors calculated in Croux & Rousseeuw (1992) while `"AA"` (the default) will use the factors calculated by the package author (2026). Note that if `center == "mean"`, there are no `CR` factors and the `AA` factors will be used.
`na.rm`	**logical**; If `TRUE` then `NA` values are stripped from `x` before computation takes place.

Details

$$MAD_n = b_n \; 1.4826 \operatorname*{med}_i |x_i - \operatorname*{center}_j x_j|$$

where b_n is calculated either by Croux & Rousseeuw (1992) or by Adler (2026) and `center` is either `median` or `mean`.

Croux & Rousseeuw (1992) used Monte-Carlo methods to calculate correction factors to make the standard MAD estimate more unbiased for small samples. They called this new function MAD_n to differentiate it from MAD, given the b_n multiplier is dependent on n—the size of the sample.

Inspired by Croux & Rousseeuw, the package author (2026) ran a significantly larger Monte-Carlo simulation for both the median absolute deviation from the median and the median absolute deviation from the mean. This function uses those factors as default.

However, unlike `mad`, this function only accepts `"median"` or `"mean"` as its central tendency—not a scalar value or any other scalar-valued function—as the factors were only calculated for those two central tendencies. Also, it does not allow passing a user-defined constant, as the intent is to return the unbiased estimate assuming normality.

If `na.rm` is `TRUE` then `NA` values are stripped from `x` before computation takes place. If this is not done then an `NA` value in `x` will cause `madn` to return `NA`.

Value

A **numeric** value representing the median absolute deviation from the requested central tendency adjusted by the asymptotic normality constant and the small-sample bias constant.

Author(s)

Avraham Adler <Avraham.Adler@gmail.com>

References

Croux, Christophe and Rousseeuw, Peter J. (1992) "Time-Efficient Algorithms for Two Highly Robust Estimators of Scale", *Computational Statistics*, Vol. 1, 411–428. doi:10.1007/9783662-268117_58

Adler, Avraham (2026) "Reducing the Bias in the Robust Estimation of Very Small Samples with the revss package for R", Manuscript in preparation

See Also

See `adm`, `admn`, and `mad` for other basic robust estimators of scale. See `robScale` for a robust M-estimator of scale.

Examples

```
set.seed(19L)
z <- rnorm(7, 8, 3)
sd(z); sd(z) / 3 - 1
mad(z); mad(z) / 3 - 1
madn(z); madn(z) / 3 - 1
madn(z, factors = "CR"); madn(z, factors = "CR") / 3 - 1
```

---

robLoc *Robust Estimate of Location*

Description

Compute a robust estimate of location for very small samples.

Usage

```
robLoc(x, scale = NULL, factors = c("AA", "CR"), na.rm = FALSE, opts = list())
```

Arguments

- `x` — **numeric**; A vector of values.
- `scale` — **numeric**; The scale, if known, can be used to enhance the estimate for the location. Defaults to unknown.
- `factors` — **character**; `"CR"` will use the small-sample bias factors for `madn` calculated in Croux & Rousseeuw (1992) while `"AA"` (default) will use the factors calculated by the package author (2026).
- `na.rm` — **logical**; If `TRUE` then `NA` values are stripped from `x` before computation takes place.
- `opts` — list; Configuration options including:
  - `maxit`: **integer**; The maximum number of iterations. Defaults to 80.
  - `tol`: **numeric**; The desired numeric tolerance for algorithmic convergence. Defaults to the square root of `.Machine$double.eps` or roughly 1.49×10^{-8}.

Details

Computes the M-estimator for location using the logistic ψ function of Rousseeuw & Verboven (2002, 4.1). If there are three or fewer entries, the function defaults to the `median`.

If the scale is known and passed through `scale`, the algorithm uses the suggestion in Rousseeuw & Verboven section 5 (2002), substituting the known scale for the `madn`.

If `na.rm` is `TRUE` then `NA` values are stripped from `x` before computation takes place. If this is not done then an `NA` value in `x` will return an error.

The tolerance and number of iterations are similar to those in existing base R functions.

Value

Solves for the robust estimate of location, T_n, which is the solution to

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

where S_n is fixed at `madn(x)`. Which b_n factors are passed to `madn` for calculating S_n is determined by the value of `factors`.

The ψ-function selected by Rousseeuw & Verboven is:

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

This is equivalent to `2 * plogis(x) - 1`.

Author(s)

Avraham Adler `<Avraham.Adler@gmail.com>`

References

Croux, Christophe and Rousseeuw, Peter J. (1992) "Time-Efficient Algorithms for Two Highly Robust Estimators of Scale", *Computational Statistics*, Vol. 1, 411–428. doi:10.1007/9783662-268117_58

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

Adler, Avraham (2026) "Reducing the Bias in the Robust Estimation of Very Small Samples with the revss package for R", Manuscript in preparation

See Also

`median`

Examples

```
set.seed(19L)
z <- rnorm(7, 8, 3)
mean(z); mean(z) / 8 - 1
median(z); median(z) / 8 - 1
robLoc(z); robLoc(z) / 8 - 1
robLoc(z, factors = "CR"); robLoc(z, factors = "CR") / 8 - 1
```

`robScale` *Robust Estimate of Scale*

Description

Compute the robust estimate of scale for very small samples.

Usage

```
robScale(x, loc = NULL, na.rm = FALSE, opts = list())
```

Arguments

`x`	**numeric**; A vector of values.
`loc`	**numeric**; The location—if known—can be used to enhance the estimate for the scale. Defaults to unknown.
`na.rm`	**logical**; If `TRUE` then `NA` values are stripped from `x` before computation takes place.
`opts`	list; Configuration options including: • `maxit`: **integer**; The maximum number of iterations. Defaults to 80. • `tol`: **numeric**; The desired numeric tolerance for algorithmic convergence. Defaults to the square root of `.Machine$double.eps` or roughly 1.49×10^{-8}.

- `usefctrs`: **logical**; If `TRUE` (default), then the small-sample bias factors for `robScale` will be used. If `FALSE`, no factors will be applied.
- `madfctrs`: **character**; when passing data to `madn` because of too few observations, `"CR"` will use the factors calculated in Croux & Rousseeuw (1992) while `"AA"` (default) will use the factors calculated by the package author (2026).
- `implbound`: **numeric**; when passing data to `madn` because of too few observations, this is the smallest value `madn` is allowed before switching to `admn` due to implosion. Defaults to 1×10^{-4}.

Details

Computes the M-estimator for scale using a smooth ρ-function defined as the square of the logistic ψ function used in location estimation (Rousseeuw & Verboven, 2002, 4.2). When the sequence of observations is too short for a robust estimate, the scale estimate will default to `madn` so long as `madn` has not "imploded", i.e. it is greater than `implbound` which defaults to 1×10^{-4}. When `madn` has imploded, `admn` is used instead.

If the location is known and passed through `loc`, the algorithm uses the suggestion in Rousseeuw & Verboven section 5 (2002) converting the observations to distances from 0 and iterating on the adjusted sequence.

Inspired by Croux & Rousseeuw, the package author (2026) ran a large Monte-Carlo simulation to estimate bias-reduction factors for both the known and unknown location versions of `robScale`. This function uses those factors by default, but they may be turned off through the `usefctrs` option.

If `na.rm` is `TRUE` then `NA` values are stripped from `x` before computation takes place. If this is not done then an `NA` value in `x` will return an error.

The tolerance and number of iterations are similar to those in existing base R functions.

Value

Solves for the robust estimate of scale, S_n, which is the solution to

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

where T_n is fixed at `median(x)` and β is fixed at 0.5. The ρ-function selected by Rousseeuw & Verboven is based on the square of the ψ-function used in robLoc. Specifically

$$\rho_{log}(x) = \psi_{log}^2\left(\frac{x}{0.37394112142347236}\right)$$

The constant 0.37394112142347236 is necessary so that

$$\beta = \int \rho(u)\; d\Phi(u) = 0.5$$

The calculated value is then adjusted by the appropriate bias-reduction factor unless `usefctrs` is set to `FALSE` in the `opts` list.

Author(s)

Avraham Adler <Avraham.Adler@gmail.com>

References

Croux, Christophe and Rousseeuw, Peter J. (1992) "Time-Efficient Algorithms for Two Highly Robust Estimators of Scale", *Computational Statistics*, Vol. 1, 411–428. doi:10.1007/9783662-268117_58

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

Adler, Avraham (2026) "Reducing the Bias in the Robust Estimation of Very Small Samples with the revss package for R", Manuscript in preparation

See Also

See `adm`, `admn`, `mad`, and `madn` for more basic robust estimators of scale.

Examples

```
set.seed(19L)
z <- rnorm(7, 8, 3)
sd(z); sd(z) / 3 - 1
admn(z); admn(z) / 3 - 1
madn(z); madn(z) / 3 - 1
robScale(z); robScale(z) / 3 - 1
robScale(z, opts = list(usefctrs = FALSE))
robScale(z, opts = list(usefctrs = FALSE)) / 3 - 1
```

Index