Package ‘confintr’

June 4, 2023

Title Confidence Intervals

Version 1.0.2

Description Calculates classic and/or bootstrap confidence intervals for many parameters such as the population mean, variance, interquartile range (IQR), median absolute deviation (MAD), skewness, kurtosis, Cramer's V, odds ratio, R-squared, quantiles (incl. median), proportions, different types of correlation measures, difference in means, quantiles and medians. Many of the classic confidence intervals are described in Smithson, M. (2003, ISBN: 978-0761924999). Bootstrap confidence intervals are calculated with the R package 'boot'. Both one- and two-sided intervals are supported.

License GPL (>= 2)

Depends R (>= 3.1.0)

Encoding UTF-8

RoxygenNote 7.2.3

Imports boot, stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/mayer79/confintr

BugReports https://github.com/mayer79/confintr/issues

NeedsCompilation no

Author Michael Mayer [aut, cre]

Maintainer Michael Mayer <mayermichael79@gmail.com>

Repository CRAN

Date/Publication 2023-06-04 18:40:02 UTC
ci_chisq_ncp

**Description**

This function calculates CIs for the non-centrality parameter (NCP) of the χ²-distribution. A positive lower \((1 - \alpha)\cdot 100\%\)-confidence limit for the NCP goes hand-in-hand with a significant association test at level \(\alpha\).

**Usage**

```r
ci_chisq_ncp(
  x,
  probs = c(0.025, 0.975),
  correct = TRUE,
  type = c("chi-squared", "bootstrap"),
)```

---

**ci_cor**
- CI for the Correlation Coefficient

**ci_cramersv**
- CI for the Cramér’s V

**ci_f_ncp**
- CI for the NCP of the F Distribution

**ci_IQR**
- CI for the Interquartile Range

**ci_kurtosis**
- CI for the Kurtosis

**ci_mad**
- CI for the Median Absolute Deviation

**ci_mean**
- CI for the Mean

**ci_mean_diff**
- CI for the Mean Difference

**ci_median**
- CI for the Median

**ci_median_diff**
- CI for the Median Difference

**ci_oddsratio**
- CI for the Odds Ratio

**ci_proportion**
- CI for the Proportion

**ci_quantile**
- CI for the Quantile

**ci_quantile_diff**
- CI for the Quantile Difference

**ci_rsquared**
- CI for the R-squared

**ci_sd**
- CI for the Standard Deviation

**ci_skewness**
- CI for the Skewness

**ci_var**
- CI for the Variance

**cramersv**
- CI for the Cramér’s V

**is.cint**
- CI for the Is Cint

**kurtosis**
- CI for the Kurtosis

**moment**
- CI for the Moment

**oddsratio**
- CI for the Odds Ratio

**print.cint**
- CI for the Print Cint

**se**
- CI for the Standard Error

**skewness**
- CI for the Skewness

---

**Index**

<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ci_chisq_ncp</td>
<td>2</td>
</tr>
<tr>
<td>ci_cor</td>
<td>4</td>
</tr>
<tr>
<td>ci_cramersv</td>
<td>5</td>
</tr>
<tr>
<td>ci_f_ncp</td>
<td>6</td>
</tr>
<tr>
<td>ci_IQR</td>
<td>7</td>
</tr>
<tr>
<td>ci_kurtosis</td>
<td>8</td>
</tr>
<tr>
<td>ci_mean</td>
<td>9</td>
</tr>
<tr>
<td>ci_mean_diff</td>
<td>10</td>
</tr>
<tr>
<td>ci_median</td>
<td>11</td>
</tr>
<tr>
<td>ci_median_diff</td>
<td>12</td>
</tr>
<tr>
<td>ci_oddsratio</td>
<td>13</td>
</tr>
<tr>
<td>ci_proportion</td>
<td>14</td>
</tr>
<tr>
<td>ci_quantile</td>
<td>15</td>
</tr>
<tr>
<td>ci_quantile_diff</td>
<td>16</td>
</tr>
<tr>
<td>ci_r_squared</td>
<td>17</td>
</tr>
<tr>
<td>ci_sd</td>
<td>18</td>
</tr>
<tr>
<td>ci_skewness</td>
<td>19</td>
</tr>
<tr>
<td>ci_var</td>
<td>20</td>
</tr>
<tr>
<td>cramersv</td>
<td>21</td>
</tr>
<tr>
<td>is.cint</td>
<td>22</td>
</tr>
<tr>
<td>kurtosis</td>
<td>23</td>
</tr>
<tr>
<td>moment</td>
<td>24</td>
</tr>
<tr>
<td>oddsratio</td>
<td>25</td>
</tr>
<tr>
<td>print.cint</td>
<td>26</td>
</tr>
<tr>
<td>se</td>
<td>27</td>
</tr>
<tr>
<td>skewness</td>
<td>28</td>
</tr>
</tbody>
</table>

---

**ci_chisq_ncp**

*CI for the NCP of the Chi-Squared Distribution*
ci_chisq_ncp

```r
boot_type = c("bca", "perc", "norm", "basic"),
R = 9999L,
seed = NULL,
...
```

Arguments

- `x`: The result of `stats::chisq.test()`, a matrix/table of counts, or a data.frame with exactly two columns representing the two variables.
- `probs`: Lower and upper probabilities, by default `c(0.025, 0.975)`.
- `correct`: Should Yates continuity correction be applied to the 2x2 case? The default is `TRUE` (also used in the bootstrap), which differs from `ci_cramersv()`.
- `type`: Type of CI. One of "chi-squared" (default) or "bootstrap".
- `boot_type`: Type of bootstrap CI. Only used for `type = "bootstrap"`.
- `R`: The number of bootstrap resamples. Only used for `type = "bootstrap"`.
- `seed`: An integer random seed. Only used for `type = "bootstrap"`.
- `...`: Further arguments passed to `boot::boot()`.

Details

By default, CIs are computed by Chi-squared test inversion. This can be unreliable for very large test statistics. The default bootstrap type is "bca".

Value

An object of class "cint", see `ci_mean()` for details.

References


See Also

- `ci_cramersv()`

Examples

```r
ci_chisq_ncp(mtcars[c("am", "vs")])
```
```
```
Description

This function calculates CIs for a population correlation coefficient. For Pearson correlation, "normal" CIs are available (by `stats::cor.test()`). Also bootstrap CIs are supported (by default "bca", and the only option for rank correlations).

Usage

```r
ci_cor(
  x,
  y = NULL,
  probs = c(0.025, 0.975),
  method = c("pearson", "kendall", "spearman"),
  type = c("normal", "bootstrap"),
  boot_type = c("bca", "perc", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  ...)
```

Arguments

- `x` A numeric vector or a `matrix/data.frame` with exactly two numeric columns.
- `y` A numeric vector (only used if `x` is a vector).
- `probs` Lower and upper probabilities, by default `c(0.025, 0.975)`.
- `method` Type of correlation coefficient, one of "pearson" (default), "kendall", or "spearman". For the latter two, only bootstrap CIs are supported.
- `type` Type of CI. One of "normal" (the default) or "bootstrap" (the only option for rank-correlations).
- `boot_type` Type of bootstrap CI. Only used for `type = "bootstrap"`.
- `R` The number of bootstrap resamples. Only used for `type = "bootstrap"`.
- `seed` An integer random seed. Only used for `type = "bootstrap"`.
- `...` Further arguments passed to `boot::boot()`.

Value

An object of class "cint", see `ci_mean()` for details.

Examples

```r
ci_cor(iris[1:2])
ci_cor(iris[1:2], type = "bootstrap", R = 999) # Use larger R
ci_cor(iris[1:2], method = "spearman", type = "bootstrap", R = 999) # Use larger R
```
Description

This function calculates CIs for the population Cramer's V. By default, a parametric approach based on the non-centrality parameter (NCP) of the chi-squared distribution is utilized. Alternatively, bootstrap CIs are available (default "bca"), also by bootstrapping CIs for the NCP and then mapping the result back to Cramer's V.

Usage

```r
ci_cramersv(
  x,
  probs = c(0.025, 0.975),
  type = c("chi-squared", "bootstrap"),
  boot_type = c("bca", "perc", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  test_adjustment = TRUE,
  ...
)
```

Arguments

- `x`: The result of `stats::chisq.test()`, a matrix/table of counts, or a `data.frame` with exactly two columns representing the two variables.
- `probs`: Lower and upper probabilities, by default `c(0.025, 0.975)`.
- `type`: Type of CI. One of "chi-squared" (default) or "bootstrap".
- `boot_type`: Type of bootstrap CI. Only used for `type = "bootstrap"`.
- `R`: The number of bootstrap resamples. Only used for `type = "bootstrap"`.
- `seed`: An integer random seed. Only used for `type = "bootstrap"`.
- `test_adjustment`: Adjustment to allow for test of association, see Details. The default is `TRUE`. Further arguments passed to `boot::boot()`.

Details

A positive lower \((1 - \alpha) \cdot 100\%\)-confidence limit for the NCP goes hand-in-hand with a significant association test at level \(\alpha\). In order to allow such test approach also with Cramer’s V, if the lower bound for the NCP is 0, we round down to 0 the lower bound for Cramer’s V as well. Without this slightly conservative adjustment, the lower limit for V would always be positive since the CI for V is found by \(\sqrt{(\text{CI for NCP} + \text{df})/(n \cdot (k - 1))}\), where \(k\) is the smaller number of levels in the two variables (see Smithson, p.40). Use `test_adjustment = FALSE` to switch off this behaviour. Note that this is also a reason to bootstrap V via NCP instead of directly bootstrapping V.
Further note that no continuity correction is applied for 2x2 tables, and that large chi-squared test statistics might provide unreliable results with method "chi-squared", see stats::pchisq().

Value
An object of class "cint", see ci_mean() for details.

References

See Also
cramersv(), ci_chisq_ncp()

Examples
# Example from Smithson, M., page 41
test_scores <- as.table(
  rbind(
    Private = c(6, 14, 17, 9),
    Public = c(30, 32, 17, 3)
  )
)
suppressWarnings(X2 <- stats::chisq.test(test_scores))

\[
\begin{align*}
\text{ci_f_ncp} & \quad CI for the Non-Centrality Parameter of the F Distribution \\
\end{align*}
\]

Description
Based on the inversion principle, parametric CIs for the non-centrality parameter (NCP) Delta of the F distribution are calculated. To keep the input interface simple, we do not provide bootstrap CIs here.

Usage
ci_f_ncp(x, df1 = NULL, df2 = NULL, probs = c(0.025, 0.975))

Arguments
\begin{itemize}
\item \texttt{x} \quad The result of stats::lm() or the F test statistic.
\item \texttt{df1} \quad The numerator df. Only used if \texttt{x} is a test statistic.
\item \texttt{df2} \quad The denominator df. Only used if \texttt{x} is a test statistic.
\item \texttt{probs} \quad Lower and upper probabilities, by default \(c(0.025, 0.975)\).
\end{itemize}
Details

A positive lower \((1 - \alpha) \cdot 100\%\)-confidence limit for the NCP goes hand-in-hand with a significant F test at level \(\alpha\). According to `stats::pf()`, the results might be unreliable for very large F values.

Value

An object of class "cint", see `ci_mean()` for details.

References


See Also

`ci_rsquared()`

Examples

```r
fit <- lm(Sepal.Length ~ ., data = iris)
ci_f_ncp(fit)
ci_f_ncp(fit, probs = c(0.05, 1))
```

---

**ci_IQR**

*CI for the IQR*

Description

This function calculates bootstrap CIs (by default "bca") for the population interquartile range (IQR), i.e., the difference between first and third quartile.

Usage

```r
ci_IQR(
  x,
  probs = c(0.025, 0.975),
  type = "bootstrap",
  boot_type = c("bca", "perc", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  ...
)
```
ci_kurtosis

Arguments

x A numeric vector.
probs Lower and upper probabilities, by default c(0.025, 0.975).
type Type of CI. Currently not used as the only type is "bootstrap".
boot_type Type of bootstrap CI c("bca", "perc", "norm", "basic").
R The number of bootstrap resamples. Only used for type = "bootstrap".
seed An integer random seed. Only used for type = "bootstrap".
... Further arguments passed to boot::boot().

Value

An object of class "cint", see ci_mean() for details.

Examples

x <- rnorm(100)
ci_IQR(x, R = 999) # Use larger R
Arguments

- **x**: A numeric vector.
- **probs**: Lower and upper probabilities, by default \(c(0.025, 0.975)\).
- **type**: Type of CI. Currently not used as the only type is "bootstrap".
- **boot_type**: Type of bootstrap CI. Only used for type = "bootstrap".
- **R**: The number of bootstrap resamples. Only used for type = "bootstrap".
- **seed**: An integer random seed. Only used for type = "bootstrap".
- **...**: Further arguments passed to `boot::boot()`.

Value

An object of class "cint", see `ci_mean()` for details.

See Also

`kurtosis()`, `ci_skewness()`

Examples

```r
x <- 1:20
ci_kurtosis(x, R = 999) # Use larger R
```

---

CI for the MAD

Description

This function calculates bootstrap CIs (default: "bca") for the population median absolute deviation (MAD), see `stats::mad()` for more information.

Usage

```r
ci_mad(
  x,
  probs = c(0.025, 0.975),
  constant = 1.4826,
  type = "bootstrap",
  boot_type = c("bca", "perc", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  ...)
```
ci_mean

Arguments

- **x**  A numeric vector.
- **probs**  Lower and upper probabilities, by default c(0.025, 0.975).
- **constant**  Scaling factor applied. The default (1.4826) ensures that the MAD equals the standard deviation for a theoretical normal distribution.
- **type**  Type of CI. Currently not used as the only type is "bootstrap".
- **boot_type**  Type of bootstrap CI c("bca", "perc", "norm", "basic").
- **R**  The number of bootstrap resamples. Only used for type = "bootstrap".
- **seed**  An integer random seed. Only used for type = "bootstrap".
- **...**  Further arguments passed to `boot::boot()`.

Value

An object of class "cint", see `ci_mean()` for details.

Examples

```r
x <- rnorm(100)
ci_mad(x, R = 999)  # Use larger R
```

Description

This function calculates CIs for the population mean. By default, Student’s t method is used. Alternatively, Wald and bootstrap CIs are available.

Usage

```r
ci_mean(
  x,
  probs = c(0.025, 0.975),
  type = c("t", "Wald", "bootstrap"),
  boot_type = c("stud", "bca", "perc", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  ...
)
```
ci_mean

Arguments

x  A numeric vector.
probs  Lower and upper probabilities, by default c(0.025, 0.975).
type  Type of CI. One of "t" (default), "Wald", or "bootstrap".
boot_type  Type of bootstrap CI. Only used for type = "bootstrap".
R  The number of bootstrap resamples. Only used for type = "bootstrap".
seed  An integer random seed. Only used for type = "bootstrap".
...  Further arguments passed to boot::boot().

Details

The default bootstrap type for the mean is "stud" (bootstrap t) as it enjoys the property of being second order accurate and has a stable variance estimator (see Efron, p. 188).

Value

An object of class "cint" containing these components:

- parameter: Parameter specification.
- interval: CI for the parameter.
- estimate: Parameter estimate.
- probs: Lower and upper probabilities.
- type: Type of interval.
- info: Additional description.

References


Examples

x <- 1:100
ci_mean(x)
ci_mean(x, type = "bootstrap", R = 999, seed = 1)  # Use larger R
ci_mean_diff  
CI for the Population Mean Difference

Description
This function calculates CIs for the population value of mean(x) - mean(y). The default is Student’s method with Welch’s correction for unequal variances, but also bootstrap CIs are available.

Usage
```r
ci_mean_diff(
x, y,
probs = c(0.025, 0.975),
var.equal = FALSE,
type = c("t", "bootstrap"),
boot_type = c("stud", "bca", "perc", "norm", "basic"),
R = 9999L,
seed = NULL,
...
)
```

Arguments
- `x`: A numeric vector.
- `y`: A numeric vector.
- `probs`: Lower and upper probabilities, by default `c(0.025, 0.975)`.
- `var.equal`: Should the two variances be treated as being equal? The default is `FALSE`. If `TRUE`, the pooled variance is used to estimate the variance of the mean difference. Otherwise, Welch’s approach is used. This also applies to the "stud" bootstrap.
- `type`: Type of CI. One of "t" (default), or "bootstrap".
- `boot_type`: Type of bootstrap CI. Only used for `type = "bootstrap"`.
- `R`: The number of bootstrap resamples. Only used for `type = "bootstrap"`.
- `seed`: An integer random seed. Only used for `type = "bootstrap"`.
- `...`: Further arguments passed to `boot::boot()`.

Details
The default bootstrap type is "stud" (bootstrap t) as it has a stable variance estimator (see Efron, p. 188). Resampling is done within sample. When `boot_type = "stud"`, the standard error is estimated by Welch’s method if `var.equal = FALSE` (the default), and by pooling otherwise. Thus, `var.equal` not only has an effect for the classic Student approach (`type = "t"`) but also for `boot_type = "stud"`. 
ci_median

Description

This function calculates CIs for the population median by calling `ci_quantile()`.

Usage

```r
ci_median(
  x,
  probs = c(0.025, 0.975),
  type = c("binomial", "bootstrap"),
  boot_type = c("bca", "perc", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  ...
)
```

Arguments

- `x`: A numeric vector.
- `probs`: Lower and upper probabilities, by default `c(0.025, 0.975).
- `type`: Type of CI. One of "binomial" (default), or "bootstrap".
- `boot_type`: Type of bootstrap CI. Only used for `type = "bootstrap"`.
- `R`: The number of bootstrap resamples. Only used for `type = "bootstrap"`.
- `seed`: An integer random seed. Only used for `type = "bootstrap"`.
- `...`: Further arguments passed to `boot::boot()`.

Value

An object of class "cint", see `ci_mean()` for details.

Examples

```r
x <- 10:30
y <- 1:30
ci_mean_diff(x, y)
t.test(x, y)$conf.int
ci_mean_diff(x, y, type = "bootstrap", R = 999) # Use larger R
```
ci_median_diff

See Also

    ci_quantile()

Examples

    ci_median(1:100)

---

**ci_median_diff**: CI for the Population Median Difference of two Samples

**Description**

This function calculates bootstrap CIs for the population value of median(x) - median(y) by calling `ci_quantile_diff()`.

**Usage**

```r
    ci_median_diff(
        x, y,
        probs = c(0.025, 0.975),
        type = "bootstrap",
        boot_type = c("bca", "perc", "norm", "basic"),
        R = 9999L,
        seed = NULL,
        ...
    )
```

**Arguments**

- `x`: A numeric vector.
- `y`: A numeric vector.
- `probs`: Lower and upper probabilities, by default `c(0.025, 0.975)`.
- `type`: Type of CI. Currently, "bootstrap" is the only option.
- `boot_type`: Type of bootstrap CI. Only used for `type = "bootstrap"`.
- `R`: The number of bootstrap resamples. Only used for `type = "bootstrap"`.
- `seed`: An integer random seed. Only used for `type = "bootstrap"`.
- `...`: Further arguments passed to `boot::boot()`.

**Value**

An object of class "cint", see `ci_mean()` for details.

**See Also**

    ci_quantile_diff()
Examples

```r
x <- 10:30
y <- 1:30
ci_median_diff(x, y, R = 999) # Use larger value for R
```

---

### ci_oddsratio

**CI for the Odds Ratio**

#### Description

This function calculates a CI for the odds ratio in a 2x2 table/matrix or a data frame with two columns. The CI is obtained through `stats::fisher.test()`. Bootstrap CIs are not available.

#### Usage

```r
ci_oddsratio(x, probs = c(0.025, 0.975))
```

#### Arguments

- `x`: A 2x2 matrix/table of counts, or a data.frame with exactly two columns representing the two binary variables.
- `probs`: Lower and upper probabilities, by default `c(0.025, 0.975)`.

#### Value

An object of class "cint", see `ci_mean()` for details.

#### See Also

`oddsratio()`.  

#### Examples

```r
x <- cbind(c(10, 5), c(4, 4))
ci_oddsratio(x)
```
ci_proportion

CI for a Population Proportion

Description

This function calculates CIs for a population proportion. By default, "Clopper-Pearson" CIs are calculated (via stats::binom.test()). Further possibilities are "Wilson" (without continuity correction), "Agresti-Coull" (using normal quantile instead of +2 correction), and "bootstrap" (by default "bca").

Usage

```r
ci_proportion(
  x,
  n = NULL,
  probs = c(0.025, 0.975),
  type = c("Clopper-Pearson", "Agresti-Coull", "Wilson", "bootstrap"),
  boot_type = c("bca", "perc", "stud", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  ...
)
```

Arguments

- **x**: A numeric vector with one value (0/1) per observation, or the number of successes.
- **n**: The sample size. Only needed if `x` is a vector of length 1.
- **probs**: Lower and upper probabilities, by default `c(0.025, 0.975)`.
- **type**: Type of CI. One of "Clopper-Pearson" (the default), "Agresti–Coull", "Wilson", "bootstrap".
- **boot_type**: Type of bootstrap CI. Only used for `type = "bootstrap"`.
- **R**: The number of bootstrap resamples. Only used for `type = "bootstrap"`.
- **seed**: An integer random seed. Only used for `type = "bootstrap"`.
- **...**: Further arguments passed to `boot::boot()`.

Details

Note that we use the formulas for the Wilson and Agresti-Coull intervals in https://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval. They agree with binom::binom.confint(x, n, method = "ac"/"wilson").

Value

An object of class "cint", see `ci_mean()` for details.
References


Examples

```r
x <- rep(0:1, times = c(50, 100))
ci_proportion(x)
ci_proportion(x, type = "Wilson")
ci_proportion(x, type = "Agresti-Coull")
```

---

**ci_quantile**

CI for a Population Quantile

Description

This function calculates CIs for a population quantile. By default, distribution-free CIs based on the binomial distribution are calculated, see Hahn and Meeker. Alternatively, bootstrap CIs are available (default "bca").

Usage

```r
ci_quantile(
  x,
  q = 0.5,
  probs = c(0.025, 0.975),
  type = c("binomial", "bootstrap"),
  boot_type = c("bca", "perc", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  ...
)
```

Arguments

- **x**: A numeric vector.
- **q**: A single probability value determining the quantile (0.5 for median).
- **probs**: Lower and upper probabilities, by default c(0.025, 0.975).
- **type**: Type of CI. One of "binomial" (default), or "bootstrap".
- **boot_type**: Type of bootstrap CI. Only used for type = "bootstrap".
- **R**: The number of bootstrap resamples. Only used for type = "bootstrap".
- **seed**: An integer random seed. Only used for type = "bootstrap".
- **...**: Further arguments passed to `boot::boot()`.
Value

An object of class "cint", see `ci_mean()` for details.

References


See Also

`ci_median()`

Examples

```r
x <- 1:100
ci_quantile(x, q = 0.25)
```

---

# ci_quantile_diff

## CI for the Population Quantile Difference of two Samples

### Description

This function calculates bootstrap CIs for the population value of \( q \)-quantile\( (x) - q \)-quantile\( (y) \), by default using "bca" bootstrap. Resampling is done within sample.

### Usage

```r
ci_quantile_diff(
  x,
  y,
  q = 0.5,
  probs = c(0.025, 0.975),
  type = "bootstrap",
  boot_type = c("bca", "perc", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  ...
)
```

### Arguments

- **x**: A numeric vector.
- **y**: A numeric vector.
- **q**: A single probability value determining the quantile (0.5 for median).
- **probs**: Lower and upper probabilities, by default \( c(0.025, 0.975) \).
- **type**: Type of CI. Currently, "bootstrap" is the only option.
- **boot_type**: Type of bootstrap CI. Only used for type = "bootstrap".
ci_rsquared

R
seed
...

The number of bootstrap resamples. Only used for type = "bootstrap".
An integer random seed. Only used for type = "bootstrap".
Further arguments passed to boot::boot()

Value
An object of class "cint", see ci_mean() for details.

See Also
  ci_median_diff()

Examples
x <- 10:30
y <- 1:30
ci_quantile_diff(x, y, R = 999) # Use larger R

---

CI for the Population R-Squared

Description
This function calculates parametric CIs for the population $R^2$. It is based on CIs for the non-centrality parameter $\Delta$ of the F distribution found by test inversion. Values of $\Delta$ are mapped to $R^2$ by $R^2 = \Delta / (\Delta + df_1 + df_2 + 1)$, where the $df_j$ are the degrees of freedom of the F test statistic. A positive lower $(1 - \alpha) \cdot 100\%$-confidence limit for the $R^2$ goes hand-in-hand with a significant F test at level $\alpha$.

Usage
ci_rsquared(x, df1 = NULL, df2 = NULL, probs = c(0.025, 0.975))

Arguments
- x: The result of stats::lm() or the F test statistic.
- df1: The numerator df. Only used if x is a test statistic.
- df2: The denominator df. Only used if x is a test statistic.
- probs: Lower and upper probabilities, by default c(0.025, 0.975).

Details
According to stats::pf(), the results might be unreliable for very large F values. Note that we do not provide bootstrap CIs here to keep the input interface simple.

Value
An object of class "cint", see ci_mean() for details.
References


See Also

ci_f_ncp()

Examples

fit <- lm(Sepal.Length ~ ., data = iris)
summary(fit)$r.squared
ci_rsquared(fit)
ci_rsquared(fit, probs = c(0.05, 1))

---

**ci_sd**  
CI for the Population Std

**Description**

This function calculates CIs for the population standard deviation. They are derived from CIs for the variance by taking the square-root, see `ci_var()`.

**Usage**

```r
library(boot)

ci_sd(
  x,
  probs = c(0.025, 0.975),
  type = c("chi-squared", "bootstrap"),
  boot_type = c("bca", "perc", "stud", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  ...
)
```

**Arguments**

- `x` A numeric vector.
- `probs` Lower and upper probabilities, by default `c(0.025, 0.975)`.
- `type` Type of CI. One of "chi-squared" (default) or "bootstrap".
- `boot_type` Type of bootstrap CI. Only used for `type = "bootstrap"`.
- `R` The number of bootstrap resamples. Only used for `type = "bootstrap"`.
- `seed` An integer random seed. Only used for `type = "bootstrap"`.
- `...` Further arguments passed to `boot::boot()`. 
ci_skewness

Value

An object of class "cint", see ci_mean() for details.

See Also

ci_var()

Examples

x <- 1:100
ci_sd(x)
ci_sd(x, type = "bootstrap", R = 999) # Use larger R

---

ci_skewness  CI for the Skewness

Description

This function calculates bootstrap CIs for the population skewness. By default, bootstrap type "bca" is used.

Usage

ci_skewness(
  x,
  probs = c(0.025, 0.975),
  type = "bootstrap",
  boot_type = c("bca", "perc", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  ...
)

Arguments

x  A numeric vector.
probs  Lower and upper probabilities, by default c(0.025, 0.975).
type  Type of CI. Currently not used as the only type is "bootstrap".
boot_type  Type of bootstrap CI. Only used for type = "bootstrap".
R  The number of bootstrap resamples. Only used for type = "bootstrap".
seed  An integer random seed. Only used for type = "bootstrap".
...  Further arguments passed to boot::boot().

Value

An object of class "cint", see ci_mean() for details.
**ci_var**

**CI for the Population Variance**

This function calculates CIs for the population variance.

**Usage**

```r
ci_var(
  x,
  probs = c(0.025, 0.975),
  type = c("chi-squared", "bootstrap"),
  boot_type = c("bca", "perc", "stud", "norm", "basic"),
  R = 9999L,
  seed = NULL,
  ...)
```

**Arguments**

- **x** A numeric vector.
- **probs** Lower and upper probabilities, by default `c(0.025, 0.975)`.
- **type** Type of CI. One of "chi-squared" (default) or "bootstrap".
- **boot_type** Type of bootstrap CI. Only used for type = "bootstrap".
- **R** The number of bootstrap resamples. Only used for type = "bootstrap".
- **seed** An integer random seed. Only used for type = "bootstrap".
- **...** Further arguments passed to `boot::boot()`.

**Details**

By default, classic CIs are calculated based on the chi-squared distribution, assuming normal distribution (see Smithson). Bootstrap CIs are also available (default: "bca"). We recommend them for the non-normal case.

The stud (bootstrap t) bootstrap uses the standard error of the sample variance given in Wilks.
Cramer's V

Description

This function calculates Cramer's V, a measure of association between two categorical variables.

Usage

cramersv(x)

Arguments

x

The result of `stats::chisq.test()`, a matrix/table of counts, or a data.frame with exactly two columns representing the two variables.

Details

Cramer's V is a scaled version of the chi-squared test statistic $\chi^2$ and takes values in $[0, 1]$. It is calculated as $\sqrt{\frac{\chi^2}{n \cdot (k - 1)}}$, where $n$ is the number of observations, and $k$ is the smaller of the number of levels of the two variables.

Yates continuity correction is never applied. So in the 2x2 case, if x is the result of `stats::chisq.test()`, make sure no continuity correction was applied. Otherwise, results can be inconsistent.

Value

A numeric vector of length one.
References

See Also
ci_cramersv()

Examples
cramersv(mtcars[cbind("am", "vs")])

is.cint

Type Check

Description
Checks if an object inherits class "cint".

Usage
is.cint(x)

Arguments
x Any object.

Value
A logical vector of length one.

Examples
is.cint(ci_proportion(5, 20))
is.cint(c(1, 2))
**kurtosis**

*Pearson’s Measure of Kurtosis*

**Description**

Defined as the ratio of the 4th central moment and the squared second central moment. Under perfect normality, the kurtosis equals 3. Put differently, we do not show "excess kurtosis" but rather kurtosis.

**Usage**

```r
kurtosis(z, na.rm = TRUE)
```

**Arguments**

- `z` A numeric vector.
- `na.rm` Logical flag indicating whether to remove missing values or not. Default is `TRUE`.

**Value**

Numeric vector of length 1.

**See Also**

`moment()`, `skewness()`

**Examples**

```r
kurtosis(1:10)
kurtosis(rnorm(1000))
```

---

**moment**

*Sample Moments*

**Description**

Calculates central or non-central sample moments.

**Usage**

```r
moment(z, p = 1, central = TRUE, na.rm = TRUE)
```
Arguments

z A numeric vector.
p Order of moment.
central Should central moment be calculated? Default is TRUE.
na.rm Logical flag indicating whether to remove missing values or not. Default is TRUE.

Value

Numeric vector of length 1.

See Also

skewness(), kurtosis()

Examples

moment(1:10, p = 1)
moment(1:10, p = 1, central = FALSE)
moment(1:10, p = 2) / stats::var(1:10)

---

oddsratio Odds Ratio

Description

This function calculates the odds ratio of a 2x2 table/matrix, or a data frame with two columns.

Usage

oddsratio(x)

Arguments

x A 2x2 matrix/table of counts, or a data.frame with exactly two columns representing the two binary variables.

Details

The numerator equals the ratio of the top left entry and the bottom left entry of the 2x2 table, while the denominator equals the ratio of the top right entry and the bottom right entry. The result is usually slightly different from the one of stats::fisher.test(), which is based on the ML estimate of the odds ratio.

Value

A numeric vector of length one.
**print.cint**

**See Also**

`ci_oddsratio()`

**Examples**

```r
tab <- cbind(c(10, 5), c(4, 4))
oddsratio(tab)
```

---

**Description**

Print method for an object of class "cint".

**Usage**

```r
## S3 method for class 'cint'
print(x, digits = getOption("digits"), ...)
```

**Arguments**

- `x` A `cint` object.
- `digits` Number of digits used to format numbers.
- `...` Further arguments passed from other methods.

**Value**

Invisibly, the input is returned.

**Examples**

```r
ci_mean(1:100)
```
**se**  

*Standard errors*

**Description**

Functions to calculate standard errors of different statistics. The availability of a standard error (or statistic proportional to it) allows to apply "stud" (bootstrap t) bootstrap.

**Usage**

```r
se_mean(z, na.rm = TRUE, ...)
se_mean_diff(z, y, na.rm = TRUE, var.equal = FALSE, ...)
se_var(z, na.rm = TRUE, ...)
se_proportion(z, na.rm = TRUE, ...)
```

**Arguments**

- `z`  
  Numeric vector.
- `na.rm`  
  Should missing values be removed before calculation? Default is `TRUE`.
- `...`  
  Further arguments to be passed from other methods.
- `y`  
  Numeric vector.
- `var.equal`  
  Should the variances be treated as being equal? Default is `FALSE`.

**Value**

A numeric vector of length one.

**Examples**

```r
se_mean(1:100)
```

---

**skewness**  

*Sample Skewness*

**Description**

Calculates sample skewness. A value of 0 refers to a perfectly symmetric distribution.

**Usage**

```r
skewness(z, na.rm = TRUE)
```
skewness

Arguments

z A numeric vector.
na.rm Logical flag indicating whether to remove missing values or not. Default is TRUE.

Value

Numeric vector of length 1.

See Also

moment(), kurtosis()

Examples

skewness(1:10)
skewness(rexp(100))
Index

boot::boot(), 3–5, 8–14, 16, 17, 19–22

ci_chisq_ncp, 2
ci_chisq_ncp(), 6
ci_cor, 4
ci_cramersv, 5
ci_cramersv(), 3, 24
ci_f_ncp, 6
ci_f_ncp(), 20
ci_IQR(), 7
ci_kurtosis, 8
ci_kurtosis(), 22
ci_mean, 10
ci_mean(), 3, 4, 6–10, 13–16, 18, 19, 21, 23
ci_mean_diff, 12
ci_median, 18
ci_median(), 18
ci_median_diff, 14
ci_median_diff(), 19
ci_oddsratio, 15
ci_oddsratio(), 27
ci_proportion, 16
ci_quantile, 17
ci_quantile(), 13, 14
ci_quantile_diff, 18
ci_quantile_diff(), 14
ci_rquared, 19
ci_r_squared(), 7
ci_sd, 20
ci_sd(), 23
ci_skewness, 21
ci_skewness(), 9
ci_var, 22
ci_var(), 20, 21
cramersv, 23
cramersv(), 6

is.cint, 24

kurtosis, 25
kurtosis(), 9, 26, 29

moment, 25
moment(), 25, 29

oddsratio, 26
oddsratio(), 15

print.cint, 27

se, 28
se_mean (se), 28
se_mean_diff (se), 28
se_proportion (se), 28
se_var (se), 28

skewness, 28
skewness(), 22, 25, 26
stats::binom.test(), 16
stats::chisq.test(), 3, 5, 23
stats::cor.test(), 4
stats::fisher.test(), 15, 26
stats::lm(), 6, 19
stats::mad(), 9
stats::pchisq(), 6
stats::pf(), 7, 19