# Package 'CIPerm'

May 7, 2026

**Type** Package

**Title** Computationally-Efficient Confidence Intervals for Mean Shift from Permutation Methods

**Version** 0.2.3

**Date** 2022-06-21

**Description** Implements computationally-efficient construction of confidence intervals from permutation or randomization tests for simple differences in means, based on Nguyen (2009) <doi:10.15760/etd.7798>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**Imports** matrixStats

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**Language** en-US

**URL** https://github.com/ColbyStatSvyRsch/CIPerm/

**BugReports** https://github.com/ColbyStatSvyRsch/CIPerm/issues

**NeedsCompilation** no

**Author** Emily Tupaj [aut],
Jerzy Wieczorek [cre, aut] (ORCID:
<https://orcid.org/0000-0002-2859-6534>),
Minh Nguyen [ctb],
Mara Tableman [ctb]

**Maintainer** Jerzy Wieczorek <jawieczo@colby.edu>

**Repository** CRAN

**Date/Publication** 2022-06-21 13:10:10 UTC

# Contents



---

`cint` *Permutation-methods confidence interval for difference in means*

---

## Description

Calculate confidence interval for a simple difference in means from a two-sample permutation or randomization test. In other words, we set up a permutation or randomization test to evaluate $H_0$ : $\mu_A - \mu_B = 0$, then use those same permutations to construct a CI for the parameter $\delta = (\mu_A - \mu_B)$.

## Usage

```
cint(dset, conf.level = 0.95, tail = c("Two", "Left", "Right"))
```

## Arguments

| | |
|---|---|
| `dset` | The output of `dset`. |
| `conf.level` | Confidence level (default 0.95 corresponds to 95% confidence level). |
| `tail` | Which tail? Either "Two"- or "Left"- or "Right"-tailed interval. |

## Details

If the desired `conf.level` is not exactly feasible, the achieved confidence level will be slightly anti-conservative. We use the default numeric tolerance in `all.equal` to check if `(1-conf.level) * nrow(dset)` is an integer for one-tailed CIs, or if `(1-conf.level)/2 * nrow(dset)` is an integer for two-tailed CIs. If so, `conf.level.achieved` will be the desired `conf.level`. Otherwise, we will use the next feasible integer, thus slightly reducing the confidence level. For example, in the example below the randomization test has 35 combinations, and a two-sided CI must have at least one combination value in each tail, so the largest feasible confidence level for a two-sided CI is 1-(2/35) or around 94.3%. If we request a 95% or 99% CI, we will have to settle for a 94.3% CI instead.

## Value

A list containing the following components:

`conf.int` Numeric vector with the CI's two endpoints.

`conf.level.achieved` Numeric value of the achieved confidence level.

## Examples

```
x <- c(19, 22, 25, 26)
y <- c(23, 33, 40)
demo <- dset(x, y)
cint(dset = demo, conf.level = .95, tail = "Two")
```

---

CIPerm *CIPerm: Computationally-Efficient Confidence Intervals for Mean Shift from Permutation Methods*

---

## Description

Implements computationally-efficient construction of confidence intervals from permutation tests or randomization tests for simple differences in means. The method is based on Minh D. Nguyen's 2009 MS thesis paper, "Nonparametric Inference using Randomization and Permutation Reference Distribution and their Monte-Carlo Approximation," <doi:10.15760/etd.7798> See the nguyen vignette for a brief summary of the method. First use `dset` to tabulate summary statistics for each permutation. Then pass the results into `cint` to compute a confidence interval, or into `pval` to calculate p-values.

## Details

Our R function arguments and outputs are structured differently than the similarly-named R functions in Nguyen (2009), but the results are equivalent. In the nguyen vignette we use our functions to replicate Nguyen's results.

Following Ernst (2004) and Nguyen (2009), we use "permutation methods" to include both randomization tests and permutation tests. In the simple settings in this R package, the randomization and permutation test mechanics are identical, but their interpretations may differ.

We say "randomization test" under the model where the units are not necessarily a random sample, but the treatment assignment was random. The null hypothesis is that the treatment has no effect. In this case we can make causal inferences about the treatment effect (difference between groups) for this set of individuals, but cannot necessarily generalize to other populations.

By contrast, we say "permutation test" under the model where the units were randomly sampled from two distinct subpopulations. The null hypothesis is that the two groups have identical CDFs. In this case we can make inferences about differences between subpopulations, but there's not necessarily any "treatment" to speak of and causal inferences may not be relevant.

## References

Ernst, M.D. (2004). "Permutation Methods: A Basis for Exact Inference," *Statistical Science*, vol. 19, no. 4, 676-685, <doi:10.1214/088342304000000396>.

Nguyen, M.D. (2009). "Nonparametric Inference using Randomization and Permutation Reference Distribution and their Monte-Carlo Approximation" [unpublished MS thesis; Mara Tableman, advisor], Portland State University. *Dissertations and Theses*. Paper 5927. <doi:10.15760/etd.7798>.

dset *Permutation-methods summary statistics*

## Description

Calculate table of differences in means, medians, etc. for each combination (or permutation, if using Monte Carlo approx.), as needed in order to compute a confidence interval using `cint` and/or a p-value using `pval`.

## Usage

```
dset(group1, group2, nmc = 10000, returnData = FALSE)
```

## Arguments

| | |
|---|---|
| `group1` | Vector of numeric values for first group. |
| `group2` | Vector of numeric values for second group. |
| `nmc` | Threshold for whether to use Monte Carlo draws or complete enumeration. If the number of all possible combinations `choose(n1+n2, n1)` <= `nmc`, we use complete enumeration. Otherwise, we take a Monte Carlo sample of `nmc` permutations. You can set `nmc = 0` to force complete enumeration regardless of how many combinations there are. |
| `returnData` | Whether the returned dataframe should include columns for the permuted data itself (if TRUE), or only the derived columns that are needed for confidence intervals and p-values (if FALSE, default). |

## Value

A data frame ready to be used in `cint()` or `pval()`.

## Examples

```
x <- c(19, 22, 25, 26)
y <- c(23, 33, 40)
demo <- dset(x, y, returnData = TRUE)
knitr::kable(demo, digits = 2)
```

`pval` *Permutations-methods p-values for difference in means, medians, or Wilcoxon rank sum test*

## Description

Calculate p-values for a two-sample permutation or randomization test. In other words, we set up a permutation or randomization test to evaluate the null hypothesis that groups A and B have the same distribution, then calculate p-values for several alternatives: a difference in means (`value="m"`), a difference in medians (`value="d"`), or the Wilcoxon rank sum test (`value="w"`).

## Usage

```
pval(
  dset,
  tail = c("Two", "Left", "Right"),
  value = c("m", "s", "d", "w", "a")
)
```

## Arguments

| | |
|---|---|
| `dset` | The output of `dset`. |
| `tail` | Which tail? Either "Two"- or "Left"- or "Right"-tailed test. |
| `value` | Either "m" for difference in means (default); "s" for sum of Group 1 values [equivalent to "m" and included only for sake of checking results against Nguyen (2009) and Ernst (2004)]; "d" for difference in medians; or "w" for Wilcoxon rank sum statistic; or "a" for a named vector of all four p-values. |

## Value

Numeric p-value for the selected type of test, or a named vector of all four p-values if `value="a"`.

## Examples

```
x <- c(19, 22, 25, 26)
y <- c(23, 33, 40)
demo <- dset(x, y)
pval(dset = demo, tail = "Left", value = "s")
pval(dset = demo, tail = "Left", value = "a")
```

# Index